# Cardinality Encodings for Graph Optimization Problems *

**Alexey Ignatiev** [1,2], **Antonio Morgado** [1], **and Joao Marques-Silva** [1]
[1] LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal
[2] ISDCT SB RAS, Irkutsk, Russia
{aignatiev,ajmorgado,jpms}@ciencias.ulisboa.pt

## Abstract

Different optimization problems defined on graphs find application in complex network analysis. Existing propositional encodings render impractical the use of propositional satisfiability (SAT) and maximum satisfiability (MaxSAT) solvers for solving a variety of these problems on large graphs. This paper has two main contributions. First, the paper identifies sources of inefficiency in existing encodings for different optimization problems in graphs. Second, for the concrete case of the maximum clique problem, the paper develops a novel encoding which is shown to be far more compact than existing encodings for large sparse graphs. More importantly, the experimental results show that the proposed encoding enables existing SAT solvers to compute a maximum clique for large sparse networks, often more efficiently than the state of the art.

## 1 Introduction

Different graph optimization problems are of interest for network analysis [Newman, 2004; Palla *et al.*, 2005; Berger-Wolf and Saia, 2006; Whang *et al.*, 2016]. This includes maximum clique, maximum independent set, and minimum vertex cover among others. All these problems are well-known to be NP-hard [Garey and Johnson, 1979]. Propositional Satisfiability (SAT) solving and maximum satisfiability (MaxSAT) are in general not directly applied to solving optimization problems in the analysis of large networks. Nevertheless, the importance of the topic motivated a large body of recent work [Johnson and Trick, 1993; Pardalos and Xue, 1994; Östergård, 2002; Fahle, 2002; Régin, 2003; Tomita and Kameda, 2007; Gelder, 2008; Li and Quan, 2010b; Li and Quan, 2010a; Prosser, 2012; Li *et al.*, 2013; Pattabiraman *et al.*, 2013; Zhou *et al.*, 2014; Fang *et al.*, 2014; McCreesh and Prosser, 2014; Pattabiraman *et al.*, 2015; Rossi *et al.*, 2015; Gouveia and Martins, 2015; Li *et al.*, 2015; Jiang *et al.*, 2016; San Segundo *et al.*, 2016; Fang *et al.*, 2016; San Segundo *et al.*, 2017][1]. For the concrete case of the maximum clique problem (MaxClique), the most efficient approaches implement highly optimized forms of branch-and-bound search.

Identification of maximal and maximum size cliques finds application in the analysis of complex networks, which in most settings correspond to large, but very sparse, graphs. Unfortunately, existing SAT and MaxSAT encodings of MaxClique grow quadratically with the number of vertices, especially if the graph is sparse. For example, for a sparse network $G = (V, E)$ for which the number of edges $|E|$ grows with the number of vertices $|V|$, the number of edges of the complement graph grows with $|V|^2$. For a network with tens to hundreds of thousands of nodes, representing the edges of the complement graph is far beyond the reach of SAT and MaxSAT solvers, but also of most compute clusters. As a result, in the recent past, SAT solvers have not been directly applied to solving optimization problems in large sparse networks. Nevertheless, SAT and MaxSAT techniques have been used as preprocessing steps for solving specific graph optimization problems [Li and Quan, 2010b; Li *et al.*, 2013; Li *et al.*, 2015; Jiang *et al.*, 2016; Fang *et al.*, 2016].

This paper represents a first step towards applying SAT and MaxSAT solvers in solving large scale optimization problems in graphs. The paper has two main contributions. First, the paper shows that in many existing encodings of different optimization problems on graphs, there exist hidden encodings of cardinality constraints, namely AtMost1 constraints, and that the (quadratic) pairwise encoding is used by default. Second, the paper develops a novel encoding for the MaxClique problem, that exploits cardinality constraints. Whereas earlier propositional encodings were based on the complement graph, making them impractical for large sparse graphs, the novel encoding only uses the vertices and the edge information of the original graph. The experimental results show that, not only SAT solvers can be used for solving optimization problems in large sparse graphs, but for specific kinds of complex networks, SAT solvers yield one of the most efficient solution approaches.

The paper is organized as follows. Section 2 introduces the definitions and notation used throughout the paper. Section 3 analyzes one common source of inefficiency in the propo-

*This work was supported by FCT funding of post-doctoral grants SFRH/BPD/103609/2014, SFRH/BPD/120315/2016, and LASIGE Research Unit, ref. UID/CEC/00408/2013.

[1] Due to space restrictions, relevant earlier references to the large body of work on solving graph optimization problems are omitted.

sitional encoding of graph optimization problems. Afterwards, Section 4 develops a new propositional encoding for the MaxClique problem, which is shown to produce significantly tighter encodings for sparse graphs. Section 5 presents results of the encodings proposed in the paper. Section 6 analyzes related work and Section 7 concludes the paper.

## 2 Preliminaries

Standard definitions used in propositional satisfiability (SAT) solving are assumed [Biere *et al.*, 2009], including propositional encodings [Biere *et al.*, 2009, Chapter 2], and also maximum satisfiability (MaxSAT) definitions [Biere *et al.*, 2009, Chapter 19].

### 2.1 Optimization Problems in Undirected Graphs

The paper considers different optimization problems defined on undirected graphs. An undirected graph is defined as a tupple, $G = (V, E)$, where $V$ is a finite set of vertices and the set of edges $E \subseteq A$, with $A = \{\{u, v\} \,|\, u, v \in V \land u \neq v\}$ denoting the set of all distinct pairs of vertices. For simplicity $(u, v)$ will be used to denote each edge of $E$. Also, the complement graph $G^C = (V, E^C)$ of $G$ is such that $E^C = A \setminus E$.

**Definition 1 (Minimum Vertex Cover, MinVC)** *Given an undirected graph $G = (V, E)$, a vertex cover $T \subseteq V$ is such that for each $(u, v) \in E$, $\{u, v\} \cap T \neq \emptyset$. A minimum vertex cover is a vertex cover of minimum size.*

**Definition 2 (Maximum Independent Set, MaxIS)** *Given an undirected graph $G = (V, E)$, an independent set $I \subseteq V$ is such that for each $(u, v) \in E$ either $u \notin I$ or $v \notin I$. A maximum independent set is an independent set of maximum size.*

**Definition 3 (Maximum Clique, MaxClique)** *Given an undirected graph $G = (V, E)$, a clique (or complete subgraph) $C \subseteq V$ is such that for every pair $\{u, v\} \subseteq C$, $(u, v) \in E$. A maximum clique is a clique of maximum size.*

**Proposition 1** *$T \subseteq V$ is a vertex cover for $G$ iff $V \setminus T$ is an independent set for $G$ iff $V \setminus T$ is a clique for $G^C$.*

**Definition 4 (Minimum Coloring, MinCol)** *Given an undirected graph $G = (V, E)$ and a set of colors $\mathbb{C} = \{1, \ldots, C\}$, pick a mapping $\kappa : V \to \mathbb{C}$ such that for each $(u, v) \in E$, $\kappa(u) \neq \kappa(v)$. Mapping $\kappa$ is called a coloring of graph $G$. A minimum coloring uses a minimum number of colors.*

It is well-known that all of the above optimization problems are NP-hard [Garey and Johnson, 1979]. Moreover, a wealth of other problems can be encoded to these problems, including set packing or combinatorial auctions [Heras *et al.*, 2008], among many others.

**Example 1** *Figure 1 shows an example undirected graph that will be used throughout the paper. For this graph: (i) the minimum number of colors to color the graph is 5; (ii) there is a maximum clique of size 5, namely $\{u_1, u_2, u_3, u_4, u_5\}$; (iii) there is a maximum independent set of size 3, namely $\{u_1, u_6, u_7\}$; and (iv) there is a minimum vertex cover of size 4, namely $\{u_2, u_3, u_4, u_5\}$.*

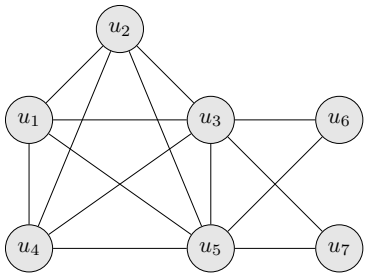


Figure 1: Example graph

### 2.2 Propositional Encodings

There exists a large body of work on solving the graph optimization problems introduced in the previous section with constraint programming, including propositional satisfiability. The paper focuses on propositional encodings for these problems, and revisits in this section what can be viewed as standard propositional encodings [Heras *et al.*, 2008; Gelder, 2008; Biere *et al.*, 2009; Li and Quan, 2010b].

Given a graph $G = (V, E)$, a propositional variable $x_u$ is associated with each vertex $u \in V$, such that $x_u = 1$ iff $u$ is picked (or selected). For the graph coloring problem, the propositional variables used are instead of the form $x_{u,k}$, such that $x_{u,k} = 1$ iff $u$ is assigned color $k$.

**Encoding MaxClique.** For any $(u, v) \in E^C$, add a hard clause $(\neg x_u \lor \neg x_v)$. For each $u \in V$, add a soft clause $(x_u)$.

Although this encoding is commonly used when translating MaxClique to SAT, for large sparse networks it may result in impractical formulas.

**Example 2** *The network `ca-dblp-2012` from the Network Repository[2] [Rossi and Ahmed, 2015] contains 317080 vertices and 1049867 edges. As a result, the complement network contains $50.26970466 \times 10^9 - 1.049867 \times 10^3 = 50.268654793 \times 10^9$ edges. This will be the number of hard binary clauses in the propositional encoding of MaxClique. Clearly, this number of clauses is well beyond the reach of current SAT and MaxSAT solvers, and would be problematic even to represent in most computing servers. This paper proposes in Section 4 a much tighter propositional encoding for the MaxClique problem for this and similar problems.*

**Encoding MinVC.** For any $(u, v) \in E$, add a hard clause $(x_u \lor x_v)$. For each $u \in V$, add a soft clause $(\neg x_u)$. For the remainder of the paper, it will be convenient to consider using instead variables $y_u \triangleq \neg x_u$. Thus, the hard constraints become $(\neg y_u \lor \neg y_v)$, and the soft constraints become $(y_u)$.

**Encoding MaxIS.** By noting the relationship between vertex covers and independent sets, we can adapt the previous encoding. For any $(u, v) \in E$, add a hard clause $(\neg x_u \lor \neg x_v)$. For each $u \in V$, add a soft clause $(x_u)$.

**Encoding MinCol.** Let $C$ be the target number of colors. Each vertex must be assigned a color, and so for each $u \in V$ encode the hard constraint $\sum_{1 \leq k \leq C} x_{u,k} = 1$. (It is well-known that, because it is a minimization problem, it suffices to require $\sum_{1 \leq k \leq C} x_{u,k} \geq 1$ [Walsh, 2000].) Moreover, adjacent vertices must be assigned different colors. Thus, for

[2] http://networkrepository.com.

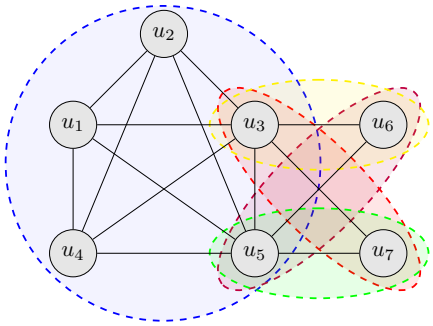


Figure 2: Example edge cover by cliques

each $(u, v) \in E$ and for each color $k$, add the hard clause $(\neg x_{u,k} \vee \neg x_{v,k})$. Observe the minimum-size graph coloring problem is not formulated as a *natural* optimization problem, but as a decision problem instead. To find a minimum size color, one simply reduces the target value of $C$ until the formula becomes unsatisfied; the smallest $C$ for which the formula is satisfiable is the minimum number of colors.

## 3 Hidden Pairwise Encodings

Given a graph $G = (V, E)$, let us consider a clique $T \subseteq V$ of $G$ and a clique $U \in V$ of $G^C$. Given that a binary clause is added for each edge in the clique $T$, for the encodings proposed in the previous section for MinVC, MaxIS and MinCol, the number of hard clauses grows quadratically with the size of $T$. For graph coloring, for each color the number of added clauses also grows quadratically with the size of $T$, for the same reason. Similarly, for MaxClique, the number of hard clauses grows quadratically with the size of $U$, in $G^C$. In general, it can be observed that the hard clauses used in these propositional encodings are essentially capturing one AtMost1 constraint, one for each clique, with a *pairwise encoding* [Biere *et al.*, 2009, Chapter 2]. We refer to the set of clauses associated with each clique in $G$ or $G^C$ as a *hidden pairwise encoding*.

**Example 3** *Consider the problem of computing a Maximum Independent Set for the graph in Figure 1. The standard encoding creates a MaxSAT formula with 7 variables and 7 soft clauses. Additionally, the MaxSAT formula contains 14 hard clauses, that is, one binary hard clause per edge containing the negative literals associated to the vertices of the edge.*

*The set of hard clauses/edges can be partitioned, such that, each partition contains the edges belonging to a maximal clique as shown in Figure 2 (the edges contained within the dashed circle/elipses). The clauses in the partitions are hidden pairwise encodings, and represent the AtMost1 cardinality constraints (using the Pairwise Cardinality encoding)* $AtMost1(x_{u_3}, x_{u_6})$, $AtMost1(x_{u_1}, x_{u_2}, x_{u_3}, x_{u_4}, x_{u_5})$, $AtMost1(x_{u_3}, x_{u_7})$, $AtMost1(x_{u_5}, x_{u_6})$, $AtMost1(x_{u_5}, x_{u_7})$, *and where* $x_{u_i}$ *is the literal associated with vertice* $u_i$.

Clearly, for graphs that include large cliques (or that have complement graphs with large cliques), these *hidden* pairwise encodings may result in unnecessarily large encodings for existing SAT and MaxSAT solvers.

**Function** FINDMXCLQ ($V$, $E$)
**Begin**
  $mxclq \leftarrow \emptyset$
  **while** $E \neq \emptyset$ **do**
    $u \leftarrow$ MAXDEGREE$(V, E)$
    $mxclq \leftarrow mxclq \cup \{u\}$
    $V \leftarrow$ ADJACENT$(E, u)$
    $E \leftarrow$ SUBGRAPH$(E, V)$
  **if** $V \neq \emptyset$ **then** $mxclq \leftarrow mxclq \cup \{V[0]\}$
  **return** $mxclq$

**Algorithm 1:** Algorithm for finding a Maximal Clique

We propose an alternative encoding based on finding an *edge-cover by cliques* (ECC) of $G$ (or of $G^C$).

**Definition 5 (Edge Cover by Cliques)** *Given an undirected graph* $G = (V, E)$*, an edge cover by cliques is a set of cliques* $\{C_1, C_2, \ldots, C_K\}$*, such that* $\cup_{1 \leq i \leq K, \{u,v\} \in C_i} \{(u, v)\} = E$.

Finding a minimal edge cover by cliques is well-known to be NP-hard [Kou *et al.*, 1978]. However, we can compute a heuristic edge cover by cliques as follows. Let $i = 1$ and $G' = G$. Iteratively compute a maximal clique $C_i$ from $G'$, report $C_i$, extract the edges induced by $C_i$ from $G'$, increase $i$ and repeat if $E \neq \emptyset$. In addition, there are well-known polynomial time algorithms for computing a (subset-) maximal clique, e.g. see Algorithm 1. Moreover, for the ECC algorithm outlined above, at each step at least one edge is removed. As a result, a heuristic edge cover by (maximal) cliques can be computed in polynomial time.

Given that we can compute an edge cover by cliques in polynomial time, for each computed clique, we add one AtMost1 constraint to the propositional encoding. Different encodings of AtMost1 constraints can be considered [Bailleux and Boufkhad, 2003; Sinz, 2005; Eén and Sörensson, 2006; Biere *et al.*, 2009; Asín *et al.*, 2011; Ogawa *et al.*, 2013] (among others).

The proposed propositional encoding thus consists of listing the cliques that cover the edges in the graph, and in creating one AtMost1 constraint for each clique, ensuring that a tight encoding of an AtMost1 constraint is used.

Depending on the selected cardinality encoding, the use of the edge cover by cliques technique can reduce the encoding size from quadratic to linear. For example, in the case the original graph is a clique, there will be a single AtMost1 constraint, which can be encoded with a linear number of clauses [Sinz, 2005; Asín *et al.*, 2011] by adding additional variables. In contrast, a result by Erdös et al. [Erdös *et al.*, 1966] gives an upper bound of $|V|^2/4$ on the size of the edge cover by cliques. This bound is tight. Consider a complete bipartite graph with $|V|/2$ vertices on each side of the biparte graph (and $|V|$ even). Then, such a graph will have an edge cover by cliques of size $|V|^2/4$. Thus, the use of AtMost1 constraints for each clique will also yield an upper bound of $\mathcal{O}(|V|^2)$ on the encoding size.

## 4 Cardinality Encoding for MaxClique

As shown in the previous section, propositional encodings for a number of graph optimization problems naturally represent (hidden) pairwise encodings of AtMost1 constraints,

concretely due to the existence of cliques in graphs. This observation is also true for the MaxClique problem, but for the complement graph. However, for the MaxClique problem the situation can be far more acute, since the encoding considers the edges in the complement graph. As a result, for large sparse graphs (including those exhibiting community structure), the complement graphs will have large cliques, which cause propositional encodings of MaxClique not to scale in practice. Although the technique of finding a (disjoint) edge cover by cliques is in general effective at reducing the encoding size, for the concrete case of the MaxClique problem, the main issue is the need to analyze the complement graph (or at least the non-edges of $G$). This section proposes an alternative encoding for the MaxClique problem, that solves this issue, by ensuring the encoding size grows linearly with the size of the graph (and not of its complement). The encoding assumes a large sparse graph $G$, where $V$ can be very large, but where $|E| = \Theta(|V|)$. As shown below, under this working hypothesis, the encoding size is in $\mathcal{O}(|V| \times \gamma(|V|, K))$, where $K$ is the target clique size, and $\gamma(|V|, K)$ is some function over $|V|$ and $K$. (Clearly, $\gamma$ must be chosen such that the encoding size is not quadratic on $|V|$.)

The novel encoding for the MaxClique problem can be summarized as follows. Let the goal be to decide whether there exists a clique of exactly size $K$ in $G$. Clearly, if this is the case, then we must satisfy the following cardinality constraint:

$$\sum_{u \in V} x_u = K \tag{1}$$

i.e. exactly $K$ vertices must be included in the selected clique. In addition, if some vertex $u$ is picked to be included in the clique, then of its adjacent vertices, exactly $K-1$ must be picked as well:

$$x_u \rightarrow \left( \sum_{v \in \mathsf{Adj}(u)} x_v = K - 1 \right) \tag{2}$$

It should be noted that, to simplify the encoding, one can replace $=$ with $\geq$ in (2). Moreover, observe that the total number of cardinality constraints used is $|V|+1$, in the worst-case.

As can be concluded, the proposed model encodes a decision problem and not an optimization problem. In principle, one could modify the problem formulation to represent an optimization problem. Instead, we propose to solve the MaxClique problem using an iterative SAT solving approach, by solving the decision problem formulation at each step. One approach is to start from some pre-computed lower bound on the value of the maximum clique, and iteratively check the existence of cliques of larger size. Moreover, we will also be interested in pre-computing an upper bound on the size of the maximum clique, e.g. to use binary search instead of linear search or to search over unsatisfiable instances.

**Correctness.** As can be observed, the proposed problem formulation essentially captures the definition of a maximum size clique. Correctness can be argued as follows.

**Proposition 2** *An undirected graph $G = (V, E)$ has a clique of size $K$ iff there exists a selection of vertices such that (1) and (2) are simultaneously satisfied.*

*Proof.* [Sketch] Suppose there exists a clique of size $K$. Simply pick the $K$ vertices, i.e. for each $u$ set $x_u = 1$ and for the others set $x_v = 0$. This will satisfy (1) and (2).
Conversely, suppose there is an assignment that satisfies (1) and (2). For each picked vertex, there are exactly $K-1$ adjacent vertices that are also picked. In total, exactly $K$ vertices are picked. Thus, for any picked vertex $u$, and the picked $K-1$ adjacents they *must* represent the same set of vertices; otherwise the number of selected vertices would be larger than $K$. Thus, the $K$ picked vertices represent a clique. □

**Encoding Size.** Each cardinality constraint on $n$ variables and right-hard side (rhs) $k$ can be encoded with $\mathcal{O}(n \times \gamma(n,k))$ clauses, where $\gamma(n,k)$ can be $n$ [Bailleux and Boufkhad, 2003], $k$ [Sinz, 2005], $\sqrt{n}$ [Ogawa *et al.*, 2013], $\log^2 n$ [Eén and Sörensson, 2006], and $\log^2 k$ [Asín *et al.*, 2011]. Moreover, the encoding size depends on the average degree of each vertex in $G$. If the encoding used is the same for all constraints, each cardinality constraint for vertex $u$ sums over $\mathsf{Deg}(u)$ vertices. In total, $|E|$ vertices are summed over. Since, by hypothesis $|E| = \Theta(|V|)$, then the overall encoding size becomes $\mathcal{O}(|V| \times \gamma(|V|, K))$, with the asymptotically tightest being $\mathcal{O}(|V| \times \log^2 K)$ [Asín *et al.*, 2011]. (Observe that it is implicit that $K = o(|V|)$.)

**Practical Optimizations & Implementation.** There are several optimizations that can be used to simplify the problem formulation, but also to simplify the SAT instances to solve.

Any (maximal) clique represents a lower bound on the size of the maximum clique. As a result, we compute a number of maximal cliques, e.g. using randomization, and pick the largest computed maximal clique. This will represent the lower bound ($L$) where the algorithm starts from. Moreover, to estimate where to stop, a simple upper bound can be computed as follows. Pick the largest value $U$ for which there are $U$ vertices in $V$, each of which with a degree of size $U-1$.

Given some target clique size $K$, the generation of the CNF formula for checking the existence of a clique of size $K$ or greater can be simplified. Let $\mathsf{Deg}(u)$ denote the number of edges for vertex $u$, i.e. the degree of $u$. Then, if the $\mathsf{Deg}(u) < K-1$ it is *guaranteed* that $u$ cannot be included in a clique of size $K$. Thus, the propositional encoding can be optimized as follows. For any vertex $u \in V$, with $\mathsf{Deg}(u) < K-1$, add a unit clause $(\neg x_u)$; these nodes are referred to as *filtered*. For any other vertex $u \in V$, with $\mathsf{Deg}(u) \geq K-1$ (i.e. non-filtered vertices), encode the constraint:

$$x_u \rightarrow \left( \sum_{v \in \mathsf{Adj}(u) \wedge \mathsf{Deg}(v) \geq K-1} x_v = K - 1 \right) \tag{3}$$

Finally, add the modified constraint over (some of) the vertices in $V$:

$$\sum_{u \in V \wedge \mathsf{Deg}(u) \geq K-1} x_u = K \tag{4}$$

The above optimization can be further improved by considering the *effective* degree of each vertex, i.e. the adjacent vertices not yet assigned value 0. The effective degree can also be used to refine the upper bound $U$.

The concept of *filtering* can be extended to edges. For any

edge $(u, v) \in E$, if $|\mathsf{Adj}(u) \cap \mathsf{Adj}(v)| < K - 2$, then it is *guaranteed* that $(u, v)$ cannot be included in a size $K$ clique. Thus, the edge is declared *filtered* and the degree information for both $u$ and $v$ is updated.

Although for large graphs it is infeasible to find maximal cliques in the complement graph, $G^C$, the following heuristic can be used in practice. Sort vertices by decreasing degree. For each vertex $u \in V$, pick the first $J$ vertices of higher degree that are not adjacent to $u$. (Observe that only $\mathsf{Deg}(u)+J$ vertices need to be checked. Thus, the heuristic runs in linear time on $|V|$ if $J$ is constant.) For each picked vertex $v \in V$, add clause $(\neg x_u \vee \neg x_v)$.

Moreover, the iterative SAT solving approach can exploit standard techniques commonly used in SAT-based problem solving. One concrete example is to exploit the incremental SAT solving features of modern SAT solvers, which enable reusing learned clauses.

Finally, it should be noted that the encoding proposed in this section can be used with decision procedures other than SAT, including constraint programming (CP), integer linear programming (ILP), satisfiability modulo theories (SMT), and answer set programming (ASP), among others. The same applies for the optimizations outlined earlier in this section.

## 5 Experimental Results

This section investigates the application of the new encodings, for the concrete case of the MaxIS and MaxClique problems, to large sparse graphs. These graphs represent large networks exhibiting community structure. The networks considered were obtained from two sources. First, a well-known benchmark generator of networks with community structure [Lancichinetti *et al.*, 2008]. Second, a selection of examples from comprehensive collections of networks [Leskovec and Krevl, 2014; Rossi and Ahmed, 2015].

### 5.1 Problem Instances & MaxIS

Most of the benchmarks, namely the collaboration networks (*ca-*$*$), interaction networks (*ia-*$*$), retweet networks (*rt-*$*$), social networks (*soc-*$*$), technological networks (*tech-*$*$), and web graphs (*web-*$*$), were selected from the well-known publicly available sources of large real networks SNAP [Leskovec and Krevl, 2014] and Network Repository [Rossi and Ahmed, 2015]. When selecting benchmark instances, we were mostly interested in fairly large networks (with 10000–500000 nodes) that are sparse, contain both large and small communities and have a large maximum clique. The motivation for this was to obtain challenging instances whose maximum cliques are hard to compute. The easiest to solve instances were excluded from the evaluation.

We also considered 2 sparse networks having 1000 and 10000 nodes generated randomly with the use of the *Benchmark* tool [Lancichinetti *et al.*, 2008] and also exhibiting community structure (in Table 1 these are named *comm-n1000* and *comm-n10000*, respectively). Additionally and in order to get to the limits of what the state-of-the-art MaxClique solvers can deal with, we decided to construct some relatively small crafted networks with the desired properties described above following the known reduction of SAT to MaxClique [Karp, 1972]. For this, we considered a family of unsatisfiable CNF formulas, which are proved to be hard to refute by resolution-based reasoning, namely *pigeon-hole principle* formulas $PHP_n$. More precisely, we considered small formulas for $n \in \{5, 6\}$ having 20/30 variables and 45/81 clauses, respectively. These formulas are known to be minimally unsatisfiable, i.e. removing a clause makes it satisfiable. Therefore, the corresponding graphs are known to have 45 and 81 maximum cliques of size 44 and 80, respectively. The graphs were then "sparsified" by introducing 5000 additional vertices and randomly generated sparse edges. In some cases, we also added to the result 10 cliques of size 20 and again sparsified the graphs by introducing connections between the components. Note that by doing this we enforce community structure in the resulting networks, i.e. they have at least 10 communities of size $\geq 20$, a larger community containing the maximum cliques and all of them are sparsely connected by randomly generated edges. The $PHP_n$-based instances are called *p?sparse?-*$*$[3] in Table 1.

**Encoding Maximum Independent Set.** We have tested the encoding of edge cover by cliques as previously described. For the instances tested, we noticed a reduction on the number of clauses that ranges from 10% up to around 70% with respect to the initial value. For example, for the *web-arabic-2005* instance, the number of hard clauses of the standard propositional encoding is 1747269 whereas the edge cover by cliques encoding contains 475891 hard clauses; a reduction on the number of clauses of 70%.

### 5.2 Maximum Clique

The approach to the MaxClique problem proposed in Section 4 was implemented in a prototype as a Python script instrumenting calls to the Glucose 3.0 SAT solver [Audemard *et al.*, 2013]. The prototype is referred to as *SATClq*. Cardinality encoding used in SATClq is a variant of modulo totalizer [Ogawa *et al.*, 2013; Morgado *et al.*, 2015]. Lower bounding the maximum cliques in SATClq is implemented by enumerating a few maximal cliques and choosing the largest among them. The upper bounds are computed as described in Section 4. Computing lower and upper bounds are linear time procedures, whose contribution to the total running time is negligible if carefully implemented in a low-level programming language. However, in our Python prototype this can be time consuming for some of the largest networks. Therefore, in order to evaluate the efficiency of the approach, we only account for the SAT solving time, which is the most time consuming part of the proposed approach.

The considered competition represents the state of the art in branch-and-bound MaxClique solving and comprises the following known reference solvers: Cliquer 1.21 [Östergård, 2002], FMC [Pattabiraman *et al.*, 2013; Pattabiraman *et al.*, 2015], IncMaxCLQ [Li *et al.*, 2013], and LMC [Jiang *et al.*, 2016].[4] The experiments were performed in Ubuntu Linux on an Intel Xeon E5-2630 2.60GHz processor with 64GByte of memory. The time limit was set to 3600s and the memory limit to 10GByte for each process to run.

[3] Prefixes *p5* and *p6* represent formulas $PHP_5$ and $PHP_6$, respectively. Infixes *sparse1*, *sparse2*, *sparse3* denote different values of the density parameter.

[4] BBMCSP [San Segundo *et al.*, 2016] was not assessed since this tool is no longer accessible from the reported download link.

Table 1: Running time (in seconds) per solver for the considered set of benchmarks. The new SAT-based approach (SATClq) is represented by SATC here while IncMaxCLQ is denoted by IMC.

| Instance | SATC | Cliquer | FMC | IMC | LMC |
|---|---|---|---|---|---|
| comm-n1000 | 0.19 | 0.02 | 0.05 | 0.11 | 0.5 |
| comm-n10000 | — | 0.99 | — | 12.33 | 0.93 |
| ca-AstroPh | 0 | 101.17 | 0.43 | — | 0.69 |
| ca-citeseer | 0 | 354.46 | 0.92 | — | 1.03 |
| ca-coauthors-dblp | 0 | — | 29.65 | — | 9.42 |
| ca-CondMat | 0 | 71 | 0.13 | — | 0.55 |
| ca-dblp-2010 | 0 | 353.85 | 0.87 | — | 0.92 |
| ca-dblp-2012 | 0 | — | 1.39 | — | 1.07 |
| ca-HepPh | 0 | 44.61 | 0.57 | — | 0.6 |
| ca-HepTh | 0 | 27.84 | 0.06 | — | 0.49 |
| ca-MathSciNet | 0 | — | 1.27 | — | 1.07 |
| ia-email-EU | 2.47 | 7.15 | 0.08 | — | 0.49 |
| ia-reality-call | 0 | 3.98 | 0.03 | — | 0.44 |
| ia-retweet-pol | 1.76 | 2.35 | 0.16 | — | 0.49 |
| ia-wiki-Talk | — | 60.48 | 4.21 | — | 0.73 |
| rt-pol | 1.7 | 2.39 | 0.19 | — | 0.49 |
| rt_barackobama | 0 | 0.46 | 0.45 | 6.63 | 0.46 |
| soc-advogato | 0.15 | 0.25 | 0.11 | 4.91 | 0.49 |
| soc-epinions | — | 101.67 | 0.21 | — | 0.82 |
| soc-gplus | 0.01 | 2.82 | 0.45 | — | 0.47 |
| tech-as-caida2007 | 0.01 | 5.26 | 0.09 | — | 0.48 |
| tech-internet-as | 0.02 | 12.23 | 0.45 | — | 0.52 |
| tech-pgp | 3.05 | 0.71 | 0.07 | — | 0.45 |
| tech-WHOIS | — | 10.13 | — | 6.31 | 0.49 |
| web-arabic-2005 | 0 | 151.31 | 2.43 | — | 1.57 |
| web-baidu-baike-related | 0.94 | — | — | — | 2.54 |
| web-it-2004 | 0 | — | 25.32 | — | 4.87 |
| web-NotreDame | 0 | — | 3.76 | — | 1.37 |
| web-sk-2005 | 0 | 97.44 | 0.34 | — | 0.64 |
| p5sparse1 | 2.88 | 1031.15 | — | 12.17 | 0.48 |
| p5sparse2+10clq20 | 1.9 | 24.42 | — | — | 0.54 |
| p5sparse3+10clq20 | 3.62 | 150.15 | — | — | 0.58 |
| p6sparse1 | 48.34 | — | — | — | 0.53 |
| p6sparse2+10clq20 | 42.65 | — | — | — | 0.64 |
| p6sparse3+10clq20 | 50.88 | — | — | — | 0.7 |

Table 1 shows the running time for each solver for the considered benchmark set. As one can observe, IncMaxCLQ fails to solve most of the considered instances. As indicated by the author, the reason is that IncMaxCLQ makes use of a static adjacency matrix for at most 11000 vertices and it is not designed for larger graphs (as a result, in our evaluation it reports a segmentation fault for most of the problem instances). Also, as expected, the state-of-the-art solver LMC is able to solve all the considered instances within at-most 10 seconds. Another observation that can be made is that the SAT solving time of SATClq is negligible for all the collaboration networks (see *ca-*$*$ instances). However, these instances are harder to FMC and most of all Cliquer. The same holds for the considered web graphs (*web-*$*$).

On the other hand, there are instances from other families, e.g. *ia-wiki-Talk*, *soc-epinions*, *tech-WHOIS*, that are hard for SATClq but at the same time are relatively easy for Cliquer and FMC. The largest of the generated instances with community structure (*comm-n10000*) is also hard for SATClq but not for Cliquer and IncMaxCLQ. A possible explanation for this could be that these instances possibly have symmetries that deteriorate SAT solver's performance, which suggests that applying modern symmetry breaking techniques [Biere *et al.*, 2009] should further improve the performance of the proposed approach.

Regarding the $PHP_n$-based instances, SATClq demonstrates the best performance solving all of them. Cliquer succeeds to solve a half of them while FMC cannot solve any. This is unexpected because these instances are quite small and sparse (up to 22000 nodes and 360000 edges).

Overall, the number of instances solved by SATClq is 31 out of the considered 35 benchmarks (the number of SAT oracle calls varies from 1 to 83; on average, it is 13) while LMC solves 35, Cliquer and FMC solve 26 instances each, and IncMaxCLQ can solve 6. Nevertheless, these results should not be misconstrued. LMC, Cliquer and FMC (but also IncMaxCLQ) are highly optimized tools, all with several years of development and continuous improvement. In contrast, SATClq is an initial prototype. For other classes of instances, the performance of SATClq may well lag behind LMC, FMC, Cliquer or IncMaxCLQ. On the other hand, and to the best of our knowledge, the instances in Table 1 would be well beyond the reach of earlier SAT encodings of MaxClique [Heras *et al.*, 2008; Li and Quan, 2010b]; we believe this is the main contribution of SATClq.

## 6 Related Work

The problems of MinVC, MaxIS, MaxClique, MinCol have been studied in a wide range of settings (see Section 1 for a non-comprehensive list of references). Although these problems find natural reductions to SAT [Heras *et al.*, 2008; Gelder, 2008; Biere *et al.*, 2009; Li and Quan, 2010b; Li and Quan, 2010a] (and of course from SAT [Karp, 1972]), it is also the case that the most efficient practical solutions are not SAT-based. Different approaches for compact representations were investigated elsewhere [Rintanen, 2006]. For MaxClique, the most efficient approaches are based on branch-and-bound search (references in Section 1), with an ongoing effort on integrating branch and bound search with SAT and MaxSAT reasoning techniques [Li and Quan, 2010b; Zhou *et al.*, 2014; Fang *et al.*, 2014; Li *et al.*, 2015; Jiang *et al.*, 2016; Fang *et al.*, 2016]. The practical applications in network science, where one often needs to analyze large sparse graphs, motivate the recent interest in efficient algorithms for solving these problems. The problem of edge-cover by cliques (ECC) finds different applications, and has been investigated in recent work [Conte *et al.*, 2016].

## 7 Conclusions

Despite the success of SAT solvers, and SAT-based problem solvers, SAT has not been applied in solving graph optimization problems in practical settings, with analysis of complex networks being a concrete example. This paper identifies one source of inefficiency in propositional encodings of a number of graph optimization problems, that include minimum vertex cover, maximum independent set, maximum clique, and minimum graph coloring. More importantly, the paper develops a novel encoding for the maximum clique problem, which eliminates the need to analyze the complement graph (or at least the non-edges of a graph). This novel encoding enables the successful application of SAT to computing the maximum clique of large sparse graphs, to our best knowledge for the first time.

A number of optimizations can be envisioned, including the breaking of symmetries. This is the subject of future work.

## References


[Asín *et al.*, 2011] R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell. Cardinality networks: a theoretical and empirical study. *Constraints*, 16(2):195–221, 2011.

[Audemard *et al.*, 2013] G. Audemard, J. Lagniez, and L. Simon. Improving Glucose for incremental SAT solving with assumptions: Application to MUS extraction. In *SAT*, pages 309–317, 2013.

[Bailleux and Boufkhad, 2003] O. Bailleux and Y. Boufkhad. Efficient CNF encoding of boolean cardinality constraints. In *CP*, pages 108–122, 2003.

[Berger-Wolf and Saia, 2006] T. Y. Berger-Wolf and J. Saia. A framework for analysis of dynamic social networks. In *KDD*, pages 523–528, 2006.

[Biere *et al.*, 2009] A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*. IOS Press, 2009.

[Conte *et al.*, 2016] A. Conte, R. Grossi, and A. Marino. Clique covering of large real-world networks. In *SAC*, pages 1134–1139, 2016.

[Eén and Sörensson, 2006] N. Eén and N. Sörensson. Translating pseudo-boolean constraints into SAT. *JSAT*, 2(1-4):1–26, 2006.

[Erdös *et al.*, 1966] P. Erdös, A. W. Goodman, and L. Pósa. The representation of a graph by set intersections. *Canad. J. Math*, 18(106-112):86, 1966.

[Fahle, 2002] T. Fahle. Simple and fast: Improving a branch-and-bound algorithm for maximum clique. In *SODA*, pages 485–498, 2002.

[Fang *et al.*, 2014] Z. Fang, C. Li, K. Qiao, X. Feng, and K. Xu. Solving maximum weight clique using maximum satisfiability reasoning. In *ECAI*, pages 303–308, 2014.

[Fang *et al.*, 2016] Z. Fang, C. Li, and K. Xu. An exact algorithm based on maxsat reasoning for the maximum weight clique problem. *J. Artif. Intell. Res. (JAIR)*, 55:799–833, 2016.

[Garey and Johnson, 1979] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[Gelder, 2008] A. V. Gelder. Another look at graph coloring via propositional satisfiability. *Discrete Applied Mathematics*, 156(2):230–243, 2008.

[Gouveia and Martins, 2015] L. Gouveia and P. Martins. Solving the maximum edge-weight clique problem in sparse graphs with compact formulations. *EURO J. Computational Optimization*, 3(1):1–30, 2015.

[Heras *et al.*, 2008] F. Heras, J. Larrosa, S. de Givry, and T. Schiex. 2006 and 2007 Max-SAT evaluations: Contributed instances. *JSAT*, 4(2-4):239–250, 2008.

[Jiang *et al.*, 2016] H. Jiang, C. M. Li, and F. Manyà. Combining efficient preprocessing and incremental MaxSAT reasoning for MaxClique in large graphs. In *ECAI*, pages 939–947, 2016.

[Johnson and Trick, 1993] D. S. Johnson and M. A. Trick, editors. *Cliques, Coloring, and Satisfiability*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 1993.

[Karp, 1972] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

[Kou *et al.*, 1978] L. T. Kou, L. J. Stockmeyer, and C. K. Wong. Covering edges by cliques with regard to keyword conflicts and intersection graphs. *Commun. ACM*, 21(2):135–139, 1978.

[Lancichinetti *et al.*, 2008] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110, 2008.

[Leskovec and Krevl, 2014] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[Li and Quan, 2010a] C. M. Li and Z. Quan. Combining graph structure exploitation and propositional reasoning for the maximum clique problem. In *ICTAI*, pages 344–351, 2010.

[Li and Quan, 2010b] C. M. Li and Z. Quan. An efficient branch-and-bound algorithm based on MaxSAT for the maximum clique problem. In *AAAI*, 2010.

[Li *et al.*, 2013] C. Li, Z. Fang, and K. Xu. Combining MaxSAT reasoning and incremental upper bound for the maximum clique problem. In *ICTAI*, pages 939–946, 2013.

[Li *et al.*, 2015] C. Li, H. Jiang, and R. Xu. Incremental MaxSAT reasoning to reduce branches in a branch-and-bound algorithm for MaxClique. In *LION*, pages 268–274, 2015.

[McCreesh and Prosser, 2014] C. McCreesh and P. Prosser. Reducing the branching in a branch and bound algorithm for the maximum clique problem. In *CP*, pages 549–563, 2014.

[Morgado *et al.*, 2015] A. Morgado, A. Ignatiev, and J. Marques-Silva. MSCG: Robust core-guided MaxSAT solving. *JSAT*, 9:129–134, 2015.

[Newman, 2004] M. E. Newman. Detecting community structure in networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):321–330, 2004.

[Ogawa *et al.*, 2013] T. Ogawa, Y. Liu, R. Hasegawa, M. Koshimura, and H. Fujita. Modulo based CNF encoding of cardinality constraints and its application to MaxSAT solvers. In *ICTAI*, pages 9–17, 2013.

[Östergård, 2002] P. R. J. Östergård. A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, 120(1-3):197–207, 2002.

[Palla *et al.*, 2005] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[Pardalos and Xue, 1994] P. M. Pardalos and J. Xue. The maximum clique problem. *J. Global Optimization*, 4(3):301–328, 1994.

[Pattabiraman *et al.*, 2013] B. Pattabiraman, M. M. A. Patwary, A. H. Gebremedhin, W. Liao, and A. N. Choudhary. Fast algorithms for the maximum clique problem on massive sparse graphs. In *WAW*, pages 156–169, 2013.

[Pattabiraman *et al.*, 2015] B. Pattabiraman, M. M. A. Patwary, A. H. Gebremedhin, W. Liao, and A. N. Choudhary. Fast algorithms for the maximum clique problem on massive graphs with applications to overlapping community detection. *Internet Mathematics*, 11(4-5):421–448, 2015.

[Prosser, 2012] P. Prosser. Exact algorithms for maximum clique: A computational study. *Algorithms*, 5(4):545–587, 2012.

[Régin, 2003] J. Régin. Using constraint programming to solve the maximum clique problem. In *CP*, pages 634–648, 2003.

[Rintanen, 2006] J. Rintanen. Compact representation of sets of binary constraints. In *ECAI*, pages 143–147, 2006.

[Rossi and Ahmed, 2015] R. Rossi and N. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, pages 4292–4293, 2015.

[Rossi *et al.*, 2015] R. A. Rossi, D. F. Gleich, and A. H. Gebremedhin. Parallel maximum clique algorithms with applications to network analysis. *SIAM J. Scientific Computing*, 37(5), 2015.

[San Segundo *et al.*, 2016] P. San Segundo, A. Lopez, and P. M. Pardalos. A new exact maximum clique algorithm for large and massive sparse graphs. *Computers & OR*, 66:81–94, 2016.

[San Segundo *et al.*, 2017] P. San Segundo, A. Lopez, J. Artieda, and P. M. Pardalos. A parallel maximum clique algorithm for large and massive sparse graphs. *Optimization Letters*, 11(2):343–358, 2017.

[Sinz, 2005] C. Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In *CP*, pages 827–831, 2005.

[Tomita and Kameda, 2007] E. Tomita and T. Kameda. An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. *J. Global Optimization*, 37(1):95–111, 2007.

[Walsh, 2000] T. Walsh. SAT v CSP. In *CP*, pages 441–456, 2000.

[Whang *et al.*, 2016] J. J. Whang, D. F. Gleich, and I. S. Dhillon. Overlapping community detection using neighborhood-inflated seed expansion. *IEEE Trans. Knowl. Data Eng.*, 28(5):1272–1284, 2016.

[Zhou *et al.*, 2014] Z. Zhou, C. M. Li, C. Huang, and R. Xu. An exact algorithm with learning for the graph coloring problem. *Computers & OR*, 51:282–301, 2014.