The Stable Set Problem: Clique and Nodal Inequalities Revisited

Adam N. Letchford[*] Fabrizio Rossi[†] Stefano Smriglio[†]

January 2020; Revised June 2020

Abstract

The *stable set* problem is a fundamental combinatorial optimisation problem, that is known to be very difficult in both theory and practice. Some of the solution algorithms in the literature are based on 0-1 linear programming formulations. We examine an entire family of such formulations, based on so-called *clique* and *nodal* inequalities. As well as proving some theoretical results, we conduct extensive computational experiments. This enables us to derive guidelines on how to choose the right formulation for a given instance.

Keywords: stable set problem, clique inequalities, nodal inequalities.

1 Introduction

The *stable set problem* (SSP) is a fundamental and much-studied combinatorial optimisation problem [21, 34]. It is also well-known to be equivalent to the *maximum clique* and *set packing* problems [37]. These problems have a wide range of applications in operational research, computer science and elsewhere [3, 4, 38, 45].

Unfortunately, the SSP is not only $\mathcal{NP}$-hard in the strong sense, but hard even to approximate [23]. It also tends to be very difficult in practice. Current leading exact algorithms (such as those in [26, 42, 44]) struggle with graphs with more than around 300 nodes (unless the graph has some special structure that can be exploited). The SSP is also rather unusual, in the sense that combinatorial algorithms, such as the ones surveyed in [39, 45], are often more effective than mathematical programming algorithms, such as those in [19, 32, 40, 41].

[*]Department of Management Science, Lancaster University, Lancaster LA1 4YX, United Kingdom. E-mail: `A.N.Letchford@lancaster.ac.uk`

[†]DISIM, University of L'Aquila, Via Vetoio, 67010, Coppito (AQ), Italy. E-mail: `fabrizio.rossi, stefano.smriglio@univaq.it`

Another unusual feature of the SSP is that it can be formulated in many ways. In fact, even if one considers only 0-1 linear programming (0-1 LP) formulations, there are several different formulations in the literature [13, 18, 21, 25, 31, 37]. In this paper, we examine an entire *family* of 0-1 LP formulations, based on various combinations of certain constraints known as *clique* and (lifted) *nodal* inequalities. As well as proving some theoretical results, we conduct extensive computational experiments, on both random instances and DIMACS benchmark instances. A careful analysis of the computational results enables us to derive guidelines on how to choose the right formulation for a given instance. All the formulations studied in this paper are publicly available through the web page `http://optimization.disim.univaq.it/stableset`

The paper is structured as follows. In §2, we briefly review the relevant literature. In §3, we consider the nodal inequalities and show how to strengthen them. In §4, we present our family of formulations. The computational experiments and analysis are described in §5 to 6. Finally, concluding remarks are made in §7.

We use the following (standard) terminology and notation in the paper. We are given a (simple, loopless) undirected graph G with vertex set V and edge set E. We sometimes write n for $|V|$ and m for $|E|$. A set of pairwise adjacent (or non-adjacent) vertices in G is called a *clique* (or *stable set*). The maximum cardinality of a clique (or stable set) in G is called the *clique number* (or *stability number*) of G, and denoted by $\omega(G)$ (or $\alpha(G)$). The minimum number of stable sets needed to cover the nodes of G is called the *chromatic number* of G and denoted by $\chi(G)$. The SSP is the problem of determining $\alpha(G)$.

The *complement* of a graph $G = (V, E)$ is the graph $\bar{G} = \big(V, \bar{E}\big)$, where $\bar{E} = \big\{\{i,j\} \subset V : \{i,j\} \notin E\big\}$ is the set of *non-edges* in G. By definition, $\alpha\big(\bar{G}\big) = \omega(G)$ and $\omega\big(\bar{G}\big) = \alpha(G)$. The minimum number of cliques needed to cover the nodes of G, called the *clique covering number* of G, is equal to $\chi\big(\bar{G}\big)$. Given a node $i \in V$, we let $N_G(i)$ denote the set of *neighbours* of i in G, i.e., $N_G(i) = \big\{j \in V \setminus \{i\} : \{i,j\} \in E\big\}$. When no confusion arises, we let $N_G(i) = N(i)$. We also let $\bar{N}(i)$ denote the set of *non-neighbours*, i.e., $\bar{N}(i) = N_{\bar{G}}(i) = V \setminus \big(\{i\} \cup N(i)\big)$. Finally, given some $S \subset V$, we let $G[S] = \big(S, E(S)\big)$ denote the subgraph of G induced by the nodes in S, and we let $r(S)$ denote $\alpha\big(G[S]\big)$(the so-called *rank* of S).

2 Literature Review

We now review the relevant literature. The three subsections in this section cover the standard 0-1 LP formulation, non-standard formulations, and random graphs, respectively. Readers interested in other aspects of the SSP are refered to the surveys [3, 21, 38, 39, 45].

2.1 The classical formulation

The "classical" formulation of the SSP is the following [37]. For each $i \in V$, let x_i be a binary variable, taking the value 1 if and only if node i is selected. Then:

$$\max \quad \textstyle\sum_{i \in V} x_i \tag{1}$$

$$\text{s.t.} \quad x_i + x_j \leq 1 \quad (\{i,j\} \in E) \tag{2}$$

$$x_i \in \{0,1\} \quad (i \in V). \tag{3}$$

The constraints (2) are called *edge inequalities.* The polytope defined by edge and non-negativity inequalities is called the *fractional stable set polytope* and denoted by FRAC(G) [21].

The convex hull of feasible solutions to (1)–(3) is called the *stable set polytope* and denoted by STAB(G). Many valid inequalities are known for this polytope; see, e.g., [4, 7, 9–12, 21, 33, 37]. The following two families of inequalities will be of importance to us:

- *clique* inequalities [37], which take the form $\sum_{i \in C} x_i \leq 1$ for each maximal clique $C \subseteq V$;
- *rank* inequalities [33], which take the form $\sum_{i \in S} x_i \leq r(S)$ for each set $S \subseteq V$.

Note that the non-dominated rank inequalities with right-hand side equal to 1 are the clique inequalities. Clique inequalities define facets of STAB(G) [37]. A sufficient condition for other rank inequalities to define facets is given in [11]. It is shown in [33] that, if a rank inequality defines a facet of STAB$\big(G[S]\big)$, then there exists at least one *lifted* rank inequality of the form

$$\sum_{i \in S} x_i + \sum_{i \in V \setminus S} \beta_i x_i \leq r(S),$$

that defines a facet of STAB(G).

The polytope defined by all clique and non-negativity inequalities is called QSTAB(G) [21]. The upper bound on $\alpha(G)$ obtained by optimising over it is called the *fractional clique covering number* and is denoted by $\chi^f(\bar{G})$. By definition, we have STAB$(G) \subseteq$ QSTAB$(G) \subseteq$ FRAC(G) and $\alpha(G) \leq \chi^f(\bar{G}) \leq \chi(\bar{G})$.

Unfortunately, the clique inequalities can be exponential in number, and the associated separation problem is $\mathcal{NP}$-hard [34]. However, as noted by several authors (e.g., [1,32,41]), one can strengthen the continuous relaxation over FRAC(G), while keeping the number of constraints bounded by $|E|$, by replacing the edge inequalities (2) with a collection of clique inequalities that "covers" the edges in E. This can be done with a greedy heuristic, such as the one described in Algorithm 1.

Algorithm 1: GreedyClqCover(G)

```
   Data: Graph $G = (V, E)$
   Result: A collection $\mathcal{C}_{\text{cov}} = C_1, \ldots, C_k$ of cliques covering all the edges
 1 Set $G'(V, E') := G(V, E)$ and $k := 0$;
 2 while $E' \neq \emptyset$ do
 3     $k := k + 1$;
 4     $i := \arg\max_{j \in V} \{|n_{G'}(j)|\}$;
 5     Set $C_k := \{i\}$;
 6     if $C_k$ is not a maximal clique in $G$ then
 7         Add nodes to make it maximal;
 8     end
 9     $E' := E' \setminus E(C_k)$;
10 end
```

2.2 Other formulations

Della Croce & Tadei [13] suggested a different 0-1 LP formulation. For a given $i \in V$, if we sum up the edge inequalities over all $j \in N(i)$, we obtain:

$$\sum_{j \in N(i)} x_j + |N(i)|\, x_i \;\leq\; |N(i)|. \tag{4}$$

Then, a more compact 0-1 LP formulation of the SSP can be obtained by replacing the m edge constraints with n constraints of the form (4). This is not a good idea in itself, since the continuous relaxation of the resulting 0-1 LP is even weaker than that of the edge formulation. As noted in Murray & Church [31], however, one can do a lot better. For a given $i \in V$, we can replace the inequality (4) with the stronger constraint:

$$\sum_{j \in N(i)} x_j + r\big(N(i)\big)\, x_i \;\leq\; r\big(N(i)\big).$$

Murray & Church [31] also proposed a "hybrid" formulation, having n specially-selected clique inequalities and n constraints of the form:

$$\sum_{j \in S(i)} x_j \,+\, r\big(S(i)\big) x_i \;\leq\; r\big(S(i)\big),$$

where $S(i)$ is a specially-selected subset of $N(i)$. Murray and Church call these last constraints *nodal* inequalities.

There also exist some *extended* 0-1 LP formulations of the SSP, which use additional variables (e.g., [1, 5, 18, 20, 25]). Some of these formulations can be strengthened using semidefinite programming (e.g., [15,17,21,22,25]). There are also formulations of the SSP as a non-linear optimisation problem in continuous variables (e.g., [6, 19, 30]). We omit details, for brevity.

2.3 Random graphs

Finally, we recall a couple of classical results concerned with random graphs. Following Erdös and Rényi [16], we let G_{np} denote a random graph on n nodes in which each edge is present, independently, with probability p. Matula [28] showed that, for fixed $0 < p < 1$, $\omega\big(G_{np}\big)$ is

$$\frac{2 \log n}{\log(1/p)} + o(\log n)$$

with high probability (w.h.p.). Bollobás [2] and Matula & Kučera [29] independently proved that, again for fixed p, $\chi\big(G_{np}\big)$ is w.h.p.

$$\frac{n \log\big(1/(1-p)\big)}{2 \log n} + o(n/\log n).$$

3 On Clique and Nodal Inequalities

From now on, when we say a *nodal inequality*, we mean an inequality of the form

$$\sum_{j \in S} x_j \,+\, r(S) x_i \,\leq\, r(S), \tag{5}$$

for some $i \in V$ and some $S \subseteq N(i)$. In this section, we present some results concerned with clique and nodal inequalities.

3.1 Expected strength for random graphs

We begin with investigating the relative strength of clique and nodal inequalities in large random graphs. The result of Matula mentioned in §2.3 implies that, w.h.p.,

$$\alpha\big(G_{np}\big) \,=\, \frac{2 \log n}{\log\big(1/(1-p)\big)} + o(\log n).$$

It also implies that one can find a point in QSTAB(G) by setting x_i to a little less than $\log(1/p)/(2 \log n)$ for all $i \in V$. This implies in turn that, w.h.p.,

$$\chi^f\big(\bar{G}_{np}\big) \geq \frac{n \log(1/p)}{2 \log n}.$$

Moreover, the other result mentioned in §2.3 implies that, w.h.p.,

$$\chi\big(\bar{G}_{np}\big) = \frac{n \log\big(1/p)\big)}{2 \log n} + o(n/\log n).$$

Given that $\chi^f\big(\bar{G}_{np}\big) \le \chi\big(\bar{G}_{np}\big)$, we obtain, w.h.p.:

$$\chi^f\big(\bar{G}_{np}\big) = \frac{n \log(1/p)}{2 \log n} + o(n/\log n).$$

Putting this all together, we have the following ratio w.h.p.:

$$\frac{\chi^f\big(\bar{G}_{np}\big)}{\alpha\big(G_{np}\big)} \;=\; \frac{n\,\log(1/p)\log\big(1/(1-p)\big)}{4\log^2 n} \;+\; o(n/\log^2 n).$$

Thus, for large n, the clique and non-negativity inequalities alone are likely to yield a poor upper bound on $\alpha(G)$.

Now consider the LP relaxation described by n nodal inequalities:

$$\sum_{j \in N(i)} x_j \,+\, r\big(N(i)\big)x_i \,\le\, r\big(N(i)\big) \qquad (i \in V) \tag{6}$$

Summing up these inequalities yields

$$\sum_{i \in V} \Big(|N(i)| + r\big(N(i)\big)\Big)x_i \,\le\, \sum_{i \in V} r\big(N(i)\big). \tag{7}$$

Now, for each i, the expected value of the left-hand side coefficient of x_i in (7) is at least $(n-1)p + 1$. Moreover, the expected value of the right-hand side is $(2n\log n)/\log\big(1/(1-p)\big) + o(n \log n)$. Thus, the upper bound ψ obtained with the nodal inequalities (6) satisfies

$$\psi\big(G_{np}\big) \le \frac{2\log n}{p\log\big(1/(1-p)\big)} + o(\log n).$$

This gives the following ratio w.h.p.:

$$\frac{\psi\big(G_{np}\big)}{\alpha\big(G_{np}\big)} \le \frac{1}{p} + o(1). \tag{8}$$

In other words, for large n and reasonably large p, nodal inequalities are likely to give a much stronger upper bound than clique inequalities. When p is small, however, nodal inequalities can give a poor bound as well. Fortunately, when G is sparse, nodal inequalities can be strengthened considerably, as illustrated in the next three subsections.

3.2 Decomposition of nodal inequalities

It can happen that there exists a partition of S into sets $S_1, \ldots, S_t$ such that

$$\sum_{k=1}^{t} r(S_k) = r(S).$$

In such a case the nodal inequality (5) is dominated by the nodal inequalities

$$x(S_k) + r(S_k)x_i \le r(S_k) \qquad (k = 1, \ldots, t).$$

We call this process *decomposition.* Here are two simple and fast ways to do it:

1. Suppose that $G[S]$ is disconnected. Then we can just let $S_1, \ldots, S_t$ be the node sets corresponding to the connected components.
2. Let us say that a node $j \in S$ is *simplicial* if $j \cup \big(S \cap N(j)\big)$ forms a clique. If this happens, let us denote the clique by C. One can check that every maximum cardinality stable set in $G[S]$ contains exactly one node from C. From this it follows that:

   $$r(S) = r(S \setminus C) + 1.$$

   Therefore, the nodal inequality (5) can be decomposed into the clique inequality on $C \cup \{i\}$ and the nodal inequality

   $$\sum_{j \in S \setminus C} x_j \, + \, r(S \setminus C)x_i \, \le \, r(S \setminus C).$$

   Moreover, if $C \cup \{i\}$ is not a maximal clique in G, then one should enlarge it, in order to strengthen the clique inequality.

Note that, after the second decomposition operation has been performed, the subgraph $G[S \setminus C]$ may be disconnected. Thus, the two decomposition operations should be applied repeatedly until no further decomposition is possible.

3.3 Lifting nodal inequalities

Given an arbitrary nodal inequality of the form (5), one can try to find a stronger valid inequality of the form

$$\sum_{j \in N(i)} \beta_j x_j \, + \, r(S)x_i \, \le \, r(S), \tag{9}$$

where $\beta_j \ge 1$ for $j \in S$ and $\beta_j \ge 0$ for $j \in N(i) \setminus S$. We will call inequalities of this type *lifted nodal inequalities.*

Observe that computing $r(S)$ is itself $\mathcal{NP}$-hard. Moreover, it follows from arguments in [37] that one must solve a weighted SSP instance on $G[N(i)]$ to compute each coefficient β_j. To alleviate this computational burden, we always start by applying decomposition, so that the initial set S is as small as possible. We also make use of the following lemma.

Lemma 1. *Suppose that S is a proper subset of $N(i)$ and we wish to lift the nodal inequality (5). If $k \in N(i) \setminus S$ and k is not adjacent to any node in S, then the variable x_k cannot be lifted (i.e., $\beta_k = 0$).*

Proof. If we increased the LHS coefficient of x_k from 0 to 1, we would obtain

$$\sum_{j \in S} x_j + r(S)x_i + x_k \leq r(S).$$

But this inequality cannot possibly be valid, since there exists a stable set that includes node k together with $r(S)$ nodes from S. $\square$

In other words, for any given $i \in V$, we can treat each connected component of $G[n(i)]$ independently when lifting nodal inequalities.

3.4 Measured strength

To gain insight into the effect of decomposition and lifting and to measure the relative strength of nodal and clique inequalities, we conduct experiments on random graphs. For each pair (n, p) with $n \in \{150, 175, \ldots, 300\}$ and $p \in \{0.1, 0.2, \ldots, 0.9\}$, we generate five G_{np} graphs (making $5 \times 7 \times 9 = 270$ graphs in total). Then, in Figure 1 we report, for each combination of n and p, the average *percentage integrality gap* (i.e., the difference between the LP upper bound and $\alpha(G)$, expressed as a percentage of $\alpha(G)$). The wireframes correspond to:

1. Nodal inequalities (6) strengthened by decomposition (blue)
2. Nodal inequalities (6) strengthened by decomposition and lifting (magenta)
3. Expected percentage gap from expression (8) (grey)

One can observe that decomposition dramatically improves the percentage gap w.r.t. the expected value on graphs with $p \in \{0.1, 0.2\}$. Lifting reduces the gap further, although its benefit decreases with n. Finally, in accordance with the theoretical expectation (8), the percentage gap appears almost insensitive to n when $p > 0.3$.

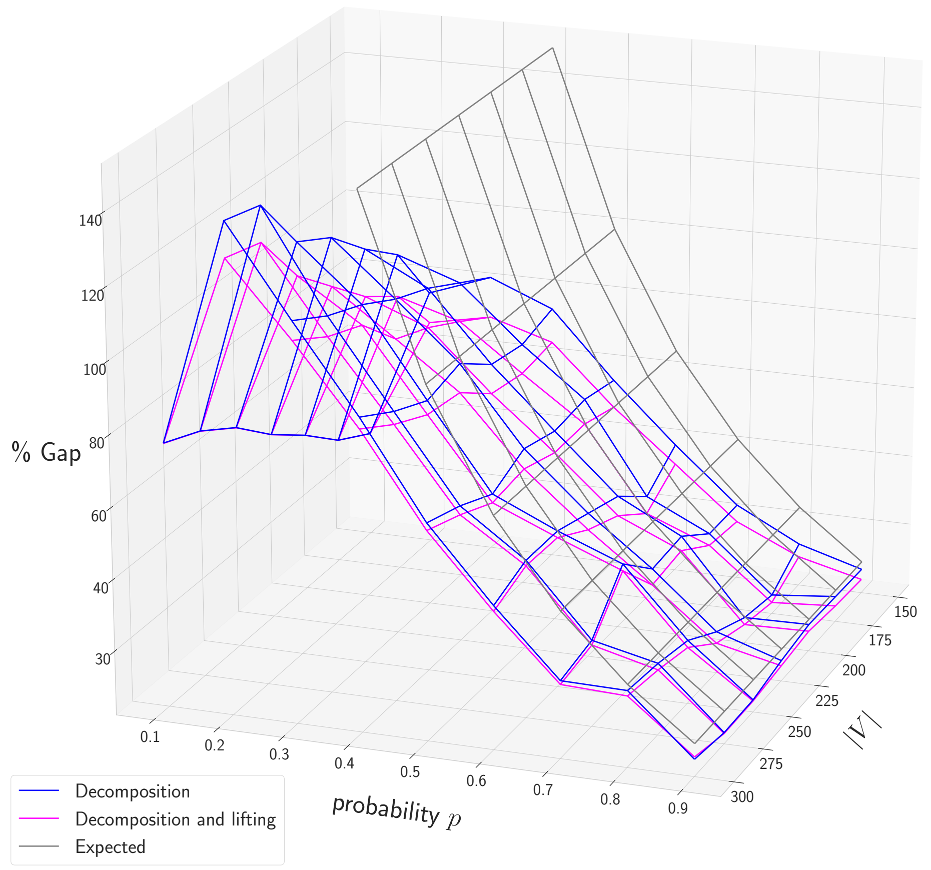

Figure 1: Effect of decomposition and lifting

In Figure 2, upper bounds from nodal inequalities are compared to a close approximation of the upper bounds obtained by optimising over QSTAB(G) [27]. The magenta wireframe depicts the percentage gap obtained by clique inequalities, while the blue wireframe represents the percentage gap obtained by nodal inequalities (without lifting). For sparse graphs, the clique bound outperforms the nodal bound, while for $p > 0.4$ the reverse happens. Also, for $p = 0.1$, the gaps are quite similar. The explanation is that, when $p = 0.1$, decomposition usually splits the $G[\hat{N}(i)]$ into cliques, so that "genuine" nodal inequalities are rare.

Motivated by these experimental observations, we investigate a family of 0-1 LP formulations of the SSP, based on various combinations of clique and (lifted) nodal inequalities, with the aim of exploiting the strong points of both classes of inequalities.

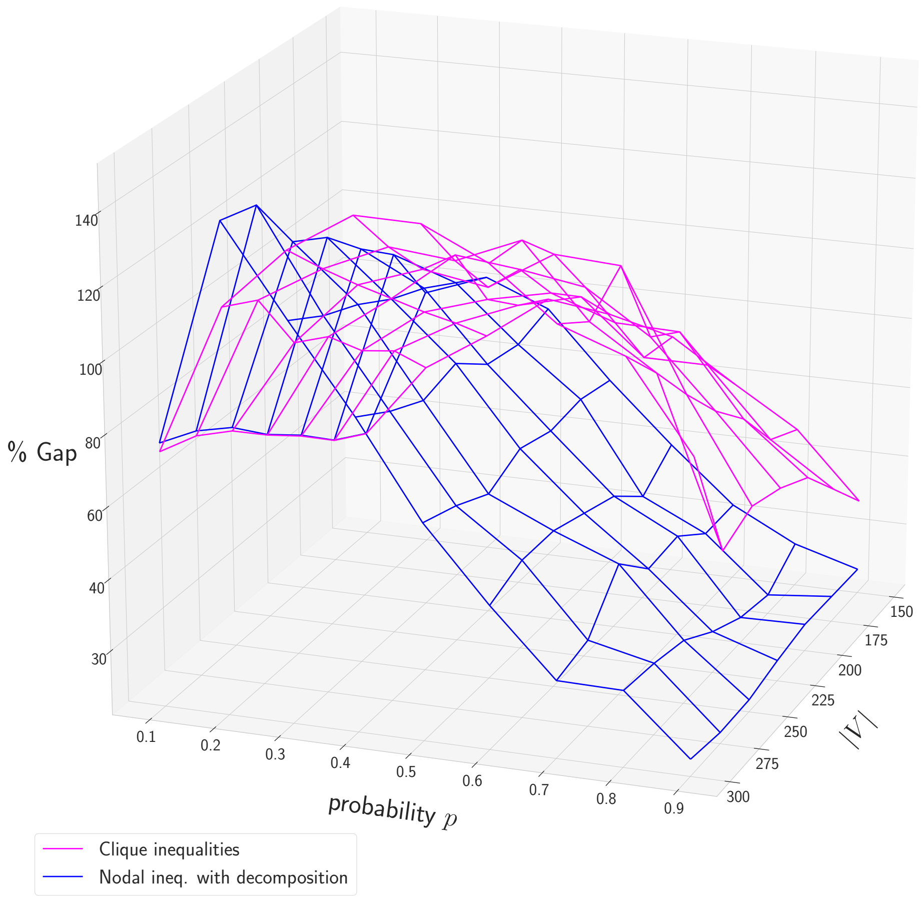

Figure 2: Percentage gap of clique and nodal inequalities

4 A Family of 0-1 LP Formulations

Consider an arbitrary collection of clique inequalities, and let $\mathcal{C}$ be the corresponding collection of cliques in G. We say that an edge $e \in E$ is *covered* by $\mathcal{C}$ if at least one clique in $\mathcal{C}$ includes both its endpoints. Trivially, an edge inequality (2) is implied by the given collection of clique inequalities if and only if $\{i, j\}$ is covered by $\mathcal{C}$.

Let $E(\mathcal{C})$ denote the set of edges covered by $\mathcal{C}$. If $E(\mathcal{C}) = E$, then nodal inequalities are unnecessary. Otherwise, we define $\hat{E}(\mathcal{C}) = E \setminus E(\mathcal{C})$ and denote by $\hat{G}(\mathcal{C})$ the partial graph $\big(V, \hat{E}(\mathcal{C})\big)$. Also, for each $i \in V$, we let $\hat{N}(i) = \big\{j \in N(i) : (i,j) \in \hat{E}(\mathcal{C})\big\}$. Then, we obtain a valid 0-1 LP formulation by adding to the given collection of clique inequalities the following nodal inequalities:

$$\sum_{j \in \hat{N}(i)} x_j + r\big(\hat{N}(i)\big) x_i \le r\big(\hat{N}(i)\big) \qquad \big(i \in V : \hat{N}(i) \neq \emptyset\big). \tag{10}$$

One can then strengthen the nodal inequalities via decomposition (and lifting, if desired).

In what follows, then, we investigate formulations that can be generated by an appropriate combination of a clique collection $\mathcal{C}$ and (if $\hat{E}(\mathcal{C}) \neq \emptyset$) a way to generate nodal inequalities.

4.1 Generating the clique collection

There are many ways in which one could generate the clique collection $\mathcal{C}$. In our preliminary experiments, we found that each of the following five options works well in certain situations:

$\mathcal{C}_\emptyset$: just set $\mathcal{C}$ to $\emptyset$;

$\mathcal{C}_{\text{cov}}$: use Algorithm 1 to generate $\mathcal{C}$;

$\mathcal{C}^{=}_{\text{cov}}$: solve the LP relaxation with one clique inequality for each clique in $\mathcal{C}_{\text{cov}}$, and put into $\mathcal{C}$ only those cliques whose corresponding inequalities are satisfied at equality by the LP solution;

$\mathcal{C}_{\text{cut}}$: start with $\mathcal{C}$ set to $\mathcal{C}_{\text{cov}}$, then solve the same LP relaxation, run a cutting-plane algorithm based on additional clique inequalities (using the separation heuristic described in [27]), and add to $\mathcal{C}$ all cliques generated;

$\mathcal{C}^{=}_{\text{cut}}$: as $\mathcal{C}_{\text{cut}}$, but put into $\mathcal{C}$ only those cliques whose inequalities are satisfied at equality by the final LP solution.

Note that, for the second and fourth options, we have $E(\mathcal{C}) = E$, which makes nodal inequalities unnecessary. We also considered other options, including the one proposed in [31], but none of them gave better results than the above five options.

4.2 Reduced nodal inequalities

In additional preliminary experiments, we found that decomposition should always be applied where possible. Unfortunately, when graphs are fairly dense, decomposition is usually not applicable, meaning that the nodal inequalities tend to be dense as well. This has two drawbacks: first, the evaluation of $r\big(\hat{N}(i)\big)$ may require a large CPU time; second, dense inequalities can put an undue burden on the LP solver. In order to obtain sparser inequalities, one can also consider the following option.

Given a ranking of the vertices and a vertex v in i-th position, a *reduced nodal inequality* is a nodal inequality generated at vertex v after removing from G all vertices in position $h < i$. Notice that a valid 0-1 LP formulation is still obtained by generating reduced nodal inequalities sequentially (according to the ranking), since, if an edge $\{v, w\}$ with v preceding w in the ranking has been removed at step i from G, then $w \in \hat{N}(v)$ and the variables x_v, x_w appear in the inequality

$$\sum_{j \in \hat{N}(v)} x_j + r\big(\hat{N}(v)\big)x_v \leq r\big(\hat{N}(v)\big)$$

with coefficients $r\big(\hat{N}(v)\big)$ and 1, respectively.

A natural ranking of the vertices is by non-increasing order of degree, as it rapidly reduces the density of the inequalities. Note that reduced nodal inequalities may be weaker than standard ones, but not necessarily, since the right-hand sides differ.

4.3 Three-field notation

To distinguish between various formulations in our family, we use the three-field notation "$\mathcal{C}\,|\,\mathcal{N}\,|\,$lift". The first two fields specify respectively the initial clique collection and the nodal inequalities collection. Entries for $\mathcal{C}$ are $\big\{\mathcal{C}_{\emptyset}, \mathcal{C}_{\text{cov}}, \mathcal{C}^{=}_{\text{cov}}, \mathcal{C}_{\text{cut}}, \mathcal{C}^{=}_{\text{cut}}\big\}$, as defined in §4. Entries for $\mathcal{N}$ are $\big\{\mathcal{N}_{\emptyset}, \mathcal{N}, \mathcal{N}_{\text{red}}\big\}$, indicating, respectively, no nodal inequalities, nodal inequalities of the form (10), and reduced nodal inequalities. Decomposition is always applied. Finally, the last field contains the entry l if lifting is applied (and no entry if not). Overall, we have 14 meaningful models, summarized in Table 1.

	$\mathcal{C}_{\emptyset}\vert\mathcal{N}\vert$	$\mathcal{C}_{\emptyset}\vert\mathcal{N}_{\text{red}}\vert$	$\mathcal{C}_{\emptyset}\vert\mathcal{N}\vert l$	$\mathcal{C}_{\emptyset}\vert\mathcal{N}_{\text{red}}\vert l$
$\mathcal{C}_{\text{cov}}\vert\mathcal{N}_{\emptyset}\vert$	$\mathcal{C}^{=}_{\text{cov}}\vert\mathcal{N}\vert$	$\mathcal{C}^{=}_{\text{cov}}\vert\mathcal{N}_{\text{red}}\vert$	$\mathcal{C}^{=}_{\text{cov}}\vert\mathcal{N}\vert l$	$\mathcal{C}^{=}_{\text{cov}}\vert\mathcal{N}_{\text{red}}\vert l$
$\mathcal{C}_{\text{cut}}\vert\mathcal{N}_{\emptyset}\vert$	$\mathcal{C}^{=}_{\text{cut}}\vert\mathcal{N}\vert$	$\mathcal{C}^{=}_{\text{cut}}\vert\mathcal{N}_{\text{red}}\vert$	$\mathcal{C}^{=}_{\text{cut}}\vert\mathcal{N}\vert l$	$\mathcal{C}^{=}_{\text{cut}}\vert\mathcal{N}_{\text{red}}\vert l$

Table 1: Models notation summary

We remark that most of the steps involved in building the models take very little time. However, as mentioned in §3.3, computing the coefficients of the nodal inequalities (9) involves the solution of (small but possibly weighted) SSP instances. Fortunately, as we will see, this can be done reasonably quickly in most cases. (Moreover, the construction of nodal inequalities can be parallelised easily, since each nodal inequality can be handled independently.)

5 Experiments with Random Graphs

In this section we experiment with the collection of 270 random graphs described in §3.4. Since it is well known that weighted SSP instances tend to be easier than unweighted ones for algorithms based on mathematical programming, we report results here only for unweighted instances. In the following three subsections, we evaluate the models according to three criteria: the *percentage integrality gap* (see §3.4), the amount of time taken to construct the model, and the time taken to solve the model by a commercial branch-and-cut solver.

All computations were run on a machine with Intel Xeon CPU E5-2698 v4 clocked at 2.2GHz with 256GB RAM. The coefficients $r(S)$ and β in (9) were computed by the combinatorial max-clique solver CLIQUER [35, 36] (after complementing the graphs). In order to get a good approximation of QSTAB(G) we set the parameter ϵ in [27] to 10^{-3} for graphs with density $< 50\%$ and to 10^{-1} for graphs with density $\geq 50\%$.

5.1 Integrality gaps

We already presented the gaps for $\mathcal{C}_{\emptyset}|\mathcal{N}|$ and $\mathcal{C}_{\text{cut}}|\mathcal{N}_{\emptyset}|$) in Figure 1. Figure 3 displays the gaps for all fourteen models. The first column in Figure 3 corresponds to models with nodal inequalities but no reduction, while the second column refers to those with reduction applied. The gray wireframes represent models without nodal inequalities. The blue and magenta ones represent models with non-lifted and lifted nodal inequalities, respectively.

The first row of Figure 3 contains models from $\mathcal{C}_{\emptyset}$. We see that nodal inequalities yield small gaps when graph density exceeds 30%, but are not competitive when density goes below 20%. Moreover, the effect of both lifting and reduction is surprisingly small.

The second row contains models from $\mathcal{C}^{=}_{\text{cov}}$. It appears that nodal inequalities help significantly for graphs in the density range $[0.3, 0.7]$, but the gap is still larger than that of the $\mathcal{C}_{\emptyset}$ models. Lifting once again gives a marginal contribution. Reduction appears to make the gap slightly worse. Models based on $\mathcal{C}^{=}_{\text{cut}}$, reported in the third row, share a similar behaviour, except that the gap decrease due to nodal inequalities is even smaller.

In summary, nodal inequalities yield a noticable benefit when the density exceeds about 20%. We remark that an additional benefit of nodal inequalities is that they lead to more compact models; see Appendix A for details.

5.2 Model construction

Figure 4 shows the time taken to build the various models. All charts in the figure share the same x-axis, in which is reported the probability p, while on the y-axis is reported the average CPU time (in seconds) needed to construct (sequentially) the models. The marker size is proportional to n and the blue diamond marker represents models that use reduction.

By comparing the charts in the bottom row with those in the other rows, we see that the cutting-plane algorithm (based on clique inequalities) takes up to around 50 seconds, which is not bad for graphs of this size. Comparing

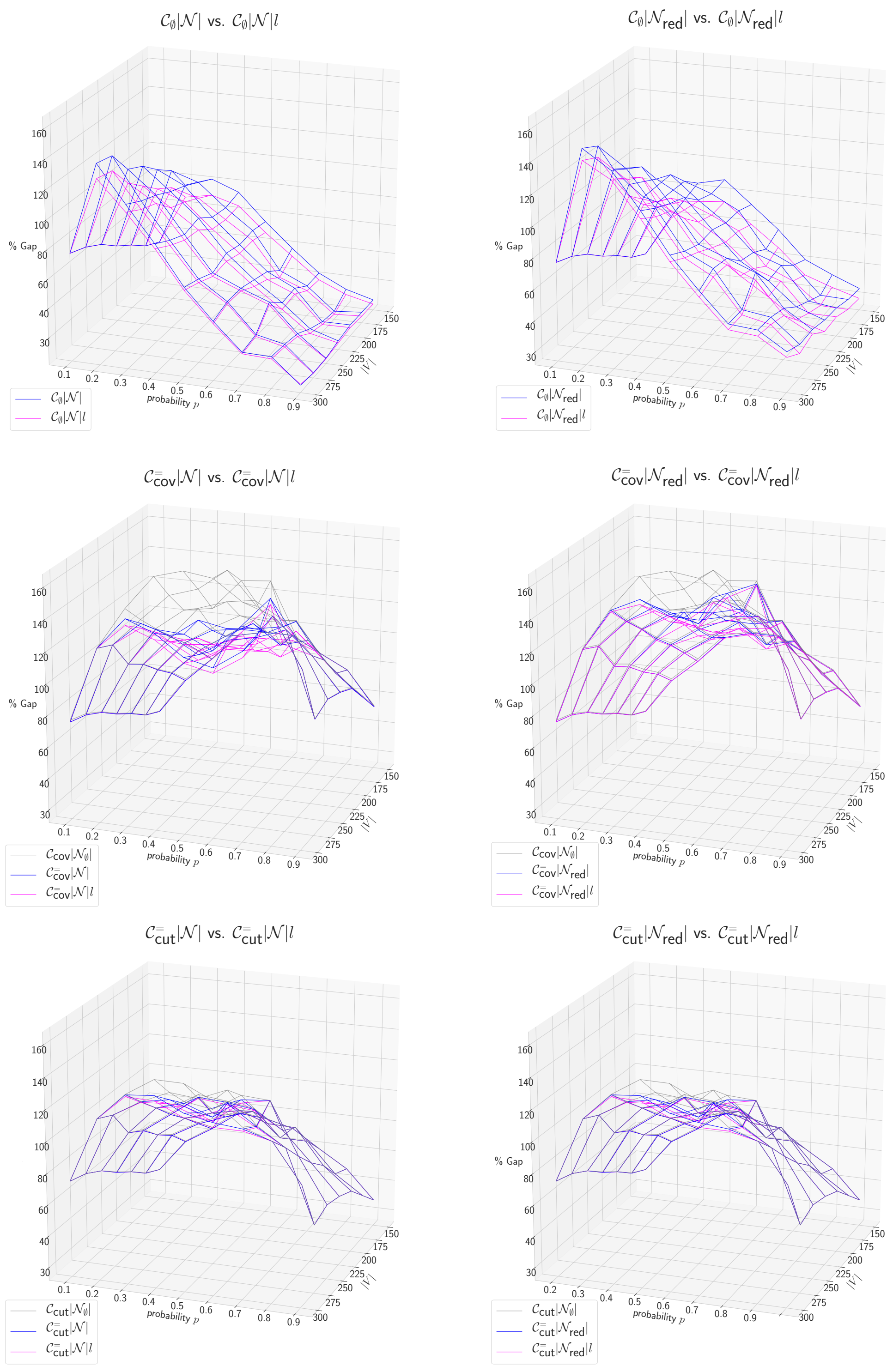

Figure 3: Percentage gap comparison

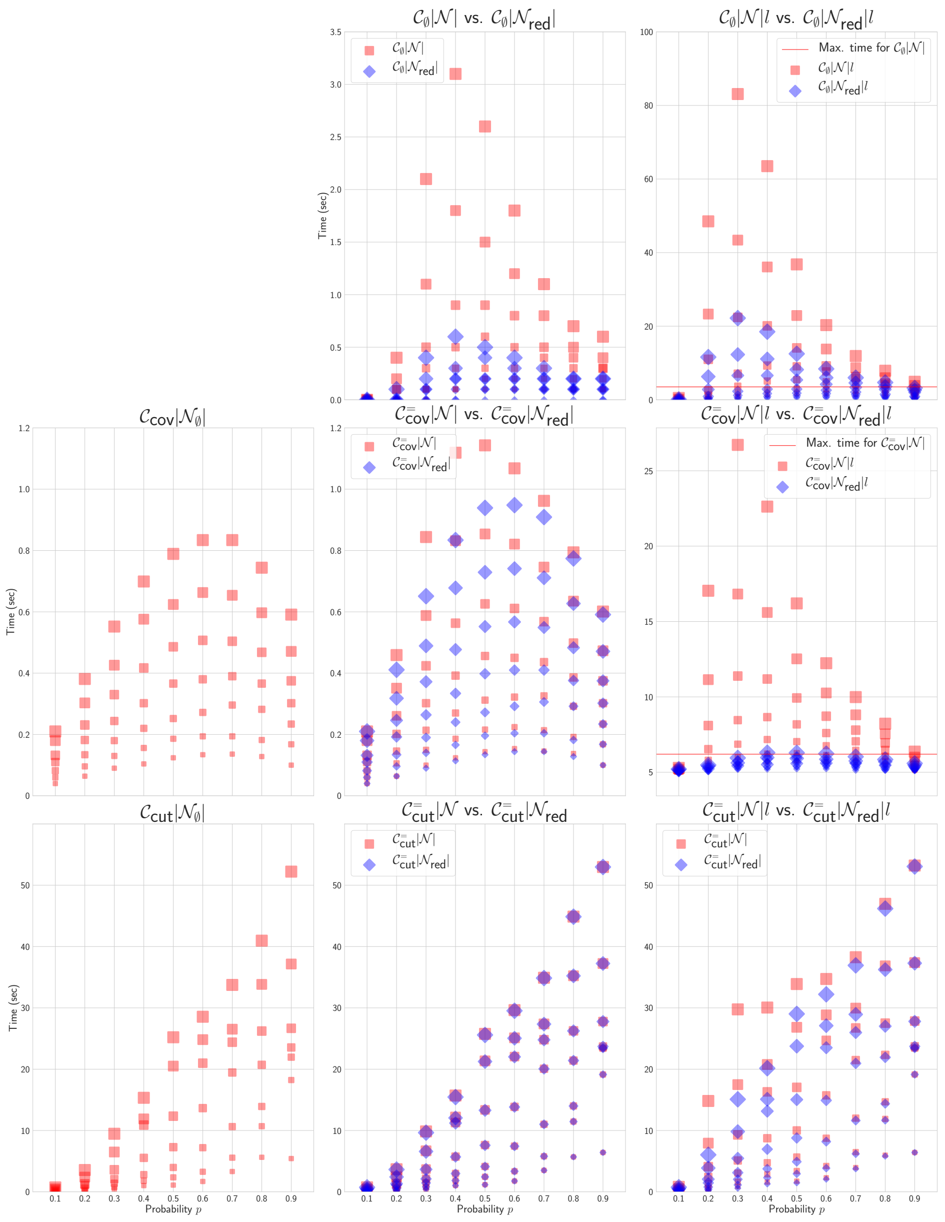

Figure 4: CPU time for model building

the charts in the second column with those in the first, we see that, perhaps surprisingly, computing the right-hand sides of the nodal inequalities is *not* a bottleneck. On the other hand, the top two charts in the third column indicate that lifting can take a significant amount of time, especially when reduction has not been applied. (The red lines in those two charts represent the maximum time spent when no lifting is applied.) Oddly, however, lifting is no problem for $\mathcal{C}^{=}_{\text{cut}}|\mathcal{N}|l$ and $\mathcal{C}^{=}_{\text{cut}}|\mathcal{N}_{\text{red}}|l$. This is probably because most of the edges in G are covered by the cliques in $\mathcal{C}^{=}_{\text{cut}}$.

Overall, the $*|\mathcal{N}_{\text{red}}|$ models (i.e, the ones that use reduction but not lifting) look the most promising. This is because the small unweighted SSPs can be solved quickly and the upper bound degradation due to reduction is marginal. Moreover, such models consistently show the smallest number of nonzeros in the constraint matrix (see Appendix A).

5.3 Model resolution

Now we turn our attention to the time taken to solve the most promising models. We use the MIP solver of `CPLEX` v. 12.8.0, with 4 threads. Default settings are used, with the exception of the branching strategy. The lesson learned from combinatorial algorithms is that branching on a vertex with highest degree typically yields benefits (e.g., [8, 43]). To implement this, we used the CPLEX parameter `branching priority`, with the priority of variable x_i set to $|n_G(i)|$. A time limit of 5,400 seconds per instance was also imposed. The non-binding clique inequalities for the $\mathcal{C}^{=}_{*}$ models, that is the ones in $\mathcal{C}_{\text{cov}} \setminus \mathcal{C}^{=}_{\text{cov}}$ and $\mathcal{C}_{\text{cut}} \setminus \mathcal{C}^{=}_{\text{cut}}$, are transferred to `CPLEX` as "user cuts". The management of possible duplicates of clique inequalities is left to the solver.

Table 2 reports the time taken to build and solve models $\mathcal{C}_{\text{cov}}|\mathcal{N}_{\emptyset}|$, $\mathcal{C}_{\emptyset}|\mathcal{N}_{\text{red}}|$, $\mathcal{C}^{=}_{\text{cov}}|\mathcal{N}_{\text{red}}|$ and $\mathcal{C}^{=}_{\text{cut}}|\mathcal{N}_{\text{red}}|$. It also shows the time taken to build and solve the edge formulation FRAC(G). CPU times are reported in seconds and each figure represents the shifted geometric mean $\prod_{i=1}^{n}(t_i+s)^{1/n}-s$ with $s = 10$ and $n = 25$, corresponding to 5 random instances executed 5 times with different CPLEX random seeds. The presence of an index next to the number declares the number of unsolved instances (in such a case, statistics are collected at time limit). In Appendix B, Table 6 reports the shifted geometric mean of the number of enumerated subproblems. For all tables, instances are sorted by increasing density and size. Finally, Figure 5 shows the relative speedup of each model w.r.t. FRAC(G) (the red line in the charts). A speedup less than 1 represents of course a performance worse than FRAC(G).

p	$\lvert V\rvert$	FRAC(G)	$\mathcal{C}_{\text{cov}}\lvert\mathcal{N}_\emptyset\rvert$	$\mathcal{C}_\emptyset\lvert\mathcal{N}_{\text{red}}\rvert$	$\mathcal{C}^{=}_{\text{cov}}\lvert\mathcal{N}_{\text{red}}\rvert$	$\mathcal{C}^{=}_{\text{cut}}\lvert\mathcal{N}_{\text{red}}\rvert$
	150	5.1	4.8	4.7	4.8	4.6
	175	26.6	27.0	26.0	25.2	26.8
0.1	200	188.7	204.5	192.7	182.4	175.1
	225	1,704.5	1,782.4	1,518.6	1,638.4	1,527.0
	150	8.0	7.3	6.4	6.8	6.3
	175	41.3	37.3	35.0	30.3	29.0
0.2	200	179.6	156.5	148.1	130.9	110.9
	225	823.2	743.0	851.2	591.9	534.3
	250	2,576.5	2,394.9	3,594.9[5]	1,962.8	1,775.9
	150	7.7	6.2	6.4	4.6	4.9
	175	23.8	20.2	21.4	14.0	13.2
	200	64.4	54.8	77.2	37.2	34.2
0.3	225	220.6	177.5	255.6	118.0	103.9
	250	596.6	480.1	846.6	324.4	293.1
	275	1,809.4	1,419.4	2,428.9	979.6	769.7
	300	4,728.1[9]	3,712.4	5,401.3[25]	2,352.6	1,828.4
	150	6.7	4.8	6.3	3.9	4.0
	175	15.5	11.4	12.8	7.6	7.8
	200	32.3	23.1	35.0	14.9	15.5
0.4	225	84.5	60.1	103.2	44.6	37.4
	250	192.5	139.7	236.0	83.5	78.0
	275	431.6	287.6	530.2	170.8	149.4
	300	792.9	577.3	1,137.3	320.4	280.4
	150	3.4	2.4	3.6	1.9	2.2
	175	12.4	7.6	7.3	4.8	5.9
	200	25.8	14.0	13.3	9.2	11.0
0.5	225	53.4	31.0	37.3	15.1	19.3
	250	104.9	55.5	68.1	30.0	34.8
	275	177.3	102.6	118.2	63.2	70.2
	300	327.1	174.8	229.1	102.7	110.6

(a) $p \in \{0.1, \ldots, 0.5\}$

p	$\lvert V\rvert$	FRAC(G)	$\mathcal{C}_{\text{cov}}\lvert\mathcal{N}_\emptyset\rvert$	$\mathcal{C}_\emptyset\lvert\mathcal{N}_{\text{red}}\rvert$	$\mathcal{C}^{=}_{\text{cov}}\lvert\mathcal{N}_{\text{red}}\rvert$	$\mathcal{C}^{=}_{\text{cut}}\lvert\mathcal{N}_{\text{red}}\rvert$
	150	3.5	2.0	1.3	1.4	2.6
	175	8.4	3.8	3.2	2.8	5.2
	200	22.2	9.7	5.3	6.8	11.6
0.6	225	53.8	22.5	8.6	12.3	22.2
	250	102.0	36.4	13.8	16.5	34.8
	275	147.7	55.7	34.8	26.4	46.6
	300	234.9	80.0	52.2	40.5	59.8
	150	2.0	1.2	0.4	0.9	4.3
	175	4.6	2.9	1.0	1.5	7.2
	200	11.1	5.2	1.8	3.5	13.2
0.7	225	28.0	10.9	3.3	7.1	25.3
	250	74.8	19.2	5.2	12.2	32.9
	275	123.4	27.9	7.9	17.3	37.8
	300	205.5	50.6	11.1	24.6	50.3
	150	1.4	0.8	0.2	0.6	6.2
	175	2.7	1.4	0.3	1.1	12.2
	200	5.1	2.3	0.5	2.3	15.4
0.8	225	7.9	3.8	0.6	3.8	23.9
	250	13.8	8.4	1.3	4.6	28.9
	275	22.1	12.9	1.9	9.0	39.9
	300	42.4	16.2	3.3	14.5	56.1
	150	1.1	0.3	0.1	0.3	6.7
	175	1.7	0.6	0.1	0.6	19.7
	200	3.6	1.0	0.2	1.0	24.4
0.9	225	5.6	1.6	0.2	1.7	24.9
	250	9.0	2.2	0.3	2.2	29.5
	275	13.0	3.0	0.4	3.1	40.0
	300	16.5	4.8	0.4	6.4	57.2

(b) $p \in \{0.6, \ldots, 0.9\}$

Table 2: Total CPU time for G_{np} graphs

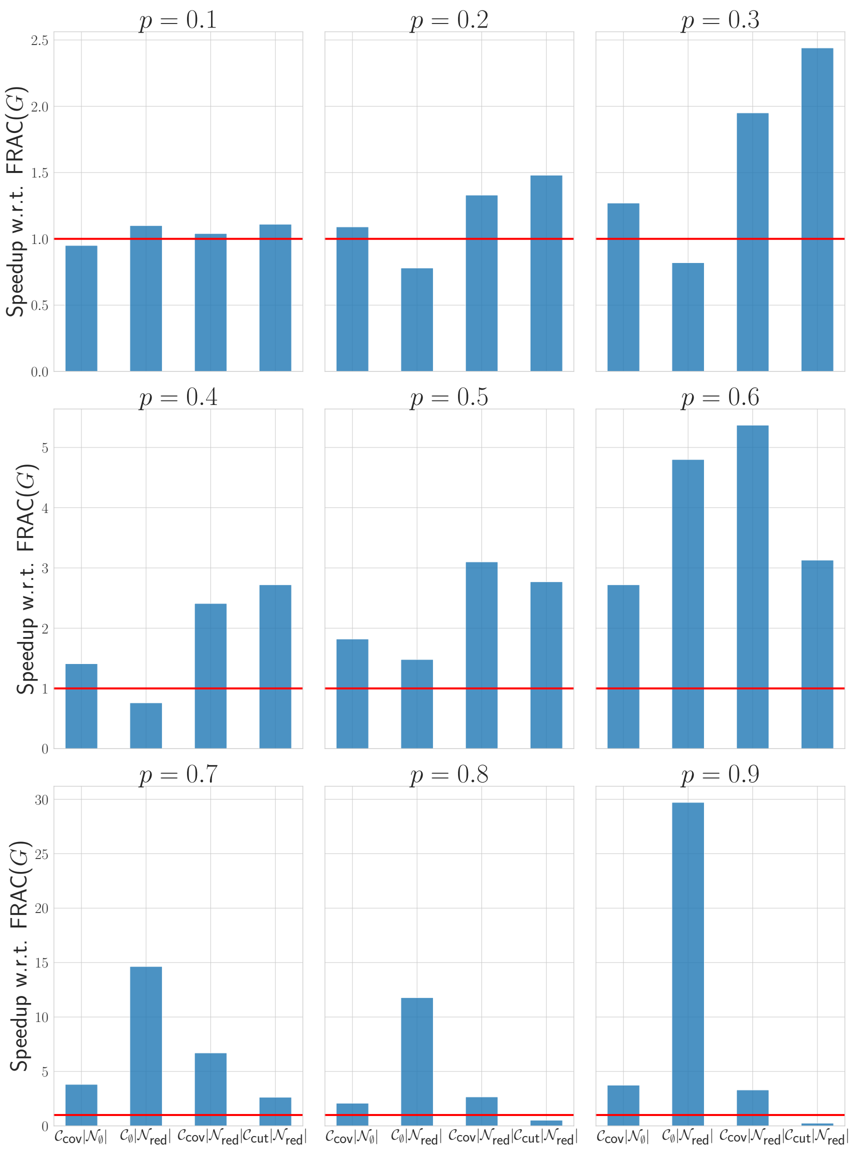

Figure 5

We now discuss the results in increasing order of graph density.

Graphs with $p = 0.1$ Here the solver was able to solve instances with up to 225 nodes within the time limit. One can observe that all models have a very similar performance, although there is some variation when $n \geq 200$. The explanation is that, when the graph is sparse, decomposition tends to reduce most or all of the nodal inequalities to clique inequalities. Moreover, the solver itself generates clique inequalities at the root node for most instances.

Graphs with $p \in \{0.2, 0.3, 0.4\}$ Here, slightly larger instances can be solved. Model $\mathcal{C}_\emptyset|\mathcal{N}_{\text{red}}|$ tends to perform worse than FRAC(G). This performance is consistent with the measured integrality gap. Models $\mathcal{C}_{\text{cov}}|\mathcal{N}_\emptyset|$ slightly outperforms FRAC(G), while models $\mathcal{C}^{=}_{\text{cov}}|\mathcal{N}_{\text{red}}|$ and $\mathcal{C}^{=}_{\text{cut}}|\mathcal{N}_{\text{red}}|$ yield a significant speedup. Closer inspection of the results shows that models with nodal inequalities tend to enumerate a larger number of subproblems than those based on clique inequalities alone. This is more than offset, however, by the fact that each subproblem is solved much more quickly.

Graphs with $p = \{0.5, 0.6\}$ Thanks to its relatively small integrality gaps, the model $\mathcal{C}_\emptyset|\mathcal{N}_{\text{red}}|$ begins to be competitive. In particular, it now outperforms FRAC(G). Moreover, $\mathcal{C}^{=}_{\text{cov}}|\mathcal{N}_{\text{red}}|$ outperforms $\mathcal{C}^{=}_{\text{cut}}|\mathcal{N}_{\text{red}}|$. Closer inspection of the results revealed that, for this density range, the improvement in bound gained by (aggressive) clique separation is outweighed by the increased burden on the LP solver. Moreover, the clique separation heuristic itself gets slower.

Graphs with $p = \{0.7, 0.8, 0.9\}$ The observed trend for $p = \{0.5, 0.6\}$ continues for $p > 0.6$. Here the best times are consistently obtained by $\mathcal{C}_\emptyset|\mathcal{N}_{\text{red}}|$, despite the fact that it is very dense.

To summarise, for medium density, $\mathcal{C}^{=}_{\text{cut}}|\mathcal{N}_{\text{red}}|$ is the clear winner. As density increases, the best perfomance is achieved by $\mathcal{C}^{=}_{\text{cov}}|\mathcal{N}_{\text{red}}|$ and then by $\mathcal{C}_\emptyset|\mathcal{N}_{\text{red}}|$. For low density, the choice of model is relatively unimportant.

6 Experiments with DIMACS graphs

In this section we experiment with the famous DIMACS benchmark max-clique instances (see [14, 24]). As usual, we complemented the graphs, to convert the max-clique instances into SSP instances (`DSJC125*` graphs are not complemented, as these belong to the "color" benchmark set). We excluded the very easy instances (namely, the smallest `johnson`, `c-fat` and `san` graphs), along with all instances which could not be solved by any of the models within 5,400 seconds (namely, the largest `johnson`, `brock`, `C`, `keller`, `p-hat` and `sanr` instances).

We first explored the models' ability to certify the optimality of a given optimal solution. The solver's settings were the same as in the case of the G_{np} graphs, except that the "MIP emphasis" was set to "emphasize optimality" and the primal heuristics were disabled. The parameter ϵ in [27] was set to $\epsilon = 10^{-2}$ for graphs with density $\leq 45\%$ and 0.3 otherwise.

Table 3 shows the total solution time, in seconds, for 31 instances and 5 models. The last row reports the average speedup w.r.t. FRAC(G) on the

Graph	FRAC(G)	$\mathcal{C}_{\text{cov}}\lvert\mathcal{N}_{\emptyset}\rvert$	$\mathcal{C}_{\emptyset}\lvert\mathcal{N}_{\text{red}}\rvert$	$\mathcal{C}^{=}_{\text{cov}}\lvert\mathcal{N}_{\text{red}}\rvert$	$\mathcal{C}^{=}_{\text{cut}}\lvert\mathcal{N}_{\text{red}}\rvert$	CLIQUER
C125-9	0.7	0.7	0.7	0.7	0.6	4.9
DSJC125.1	0.8	0.8	0.7	0.7	0.7	10.1
DSJC125.5	2.1	1.0	0.7	0.7	0.8	0.0
MANN_a45	3.5	4.0	6.0	5.1	5.0	†
MANN_a81	354.1	191.5	115.6	139.4	138.3	†
brock200_1	74.7	64.7	123.2	51.0	40.4	6.4
brock200_2	15.1	9.2	8.4	6.8	6.4	0.0
brock200_3	31.3	19.8	48.9	11.6	11.6	0.1
brock200_4	31.5	27.7	54.5	15.1	13.4	0.4
c-fat200-5	4.6	19.9	0.5	1.4	1.6	0.1
c-fat500-1	2.2	4.4	1.2	3.8	86.9	0.0
c-fat500-10	3.0	15.2	5.2	12.2	14.0	0.0
c-fat500-2	1.7	6.2	4.8	6.5	21.7	0.0
c-fat500-5	2.1	13.6	7.3	11.8	13.1	0.0
keller4	2.0	1.3	10.4	1.3	1.7	0.1
p_hat300-1	117.5	12.4	1.8	7.7	12.6	0.0
p_hat300-2	51.3	7.8	12.5	5.0	6.6	0.3
p_hat300-3	156.5	103.8	90.8	62.3	50.6	323.2
p_hat500-1	92.1	104.1	30.4	63.0	149.7	0.0
p_hat500-2	148.4	146.7	218.7	103.2	110.7	71.5
p_hat700-1	433.8	489.8	16.6	256.8	358.7	0.1
p_hat700-2	2,007.0	1,285.8	†	756.6	756.8	†
san1000	22.3	64.9	209.9	23.4	115.2	0.1
san400_0.5_1	2.0	3.2	1.8	1.7	5.2	0.0
san400_0.7_1	4.2	4.7	3.5	3.5	9.1	†
san400_0.7_2	2.9	4.0	2.8	3.0	7.7	1291.0
san400_0.7_3	1.8	1.9	0.9	1.6	46.0	3.7
san400_0.9_1	3.6	3.0	2.3	3.0	1.2	†
sanr200_0.7	52.5	44.7	103.2	30.1	27.1	2.0
sanr200_0.9	86.4	87.8	81.9	66.4	60.4	†
sanr400_0.5	2,570.7	1,340.5	1,655.6	662.6	804.0	1.2
Speedup	**1.0**	**1.5**	**0.8**	**2.7**	**2.2**	

Table 3: DIMACS Instances: CPU time to prove optimality

whole testbed. In the last column we have also have included the solution time taken by CLIQUER.

Table 4 shows, for the models that involve nodal inequalities, the total time spent by CLIQUER and the number of CLIQUER calls. Table 7 in Appendix B reports the number of enumerated subproblems.

A first observation is that, on the whole, the model $\mathcal{C}_{\emptyset}|\mathcal{N}_{\text{red}}|$ performs worse than FRAC(G) (since the overall speedup is less than 1). A closer inspection of the results shows that $\mathcal{C}_{\emptyset}|\mathcal{N}_{\text{red}}|$ works best when decomposition is effective (that is, the number of CLIQUER calls is low). If this is not the case, the performance observed with nodal inequalities turns out to be worsened.

	$\mathcal{C}_\emptyset \lvert\mathcal{N}_{\text{red}}\rvert$		$\mathcal{C}^{=}_{\text{cov}} \lvert\mathcal{N}_{\text{red}}\rvert$		$\mathcal{C}^{=}_{\text{cut}} \lvert\mathcal{N}_{\text{red}}\rvert$	
	Time	# Calls	Time	# Calls	Time	# Calls
Graph						
C125-9	0.00	1	0.00	0	0.00	1
DSJC125.1	0.00	0	0.00	0	0.00	0
DSJC125.5	0.02	110	0.00	86	0.00	85
MANN_a45	0.00	0	0.00	0	0.00	0
MANN_a81	0.00	0	0.00	0	0.00	0
brock200_1	0.02	143	0.01	117	0.01	112
brock200_2	0.08	184	0.02	168	0.02	169
brock200_3	0.06	181	0.02	149	0.02	149
brock200_4	0.05	165	0.02	142	0.01	141
c-fat200-5	0.05	191	0.05	188	0.05	188
c-fat500-1	0.64	489	0.06	418	0.11	447
c-fat500-10	0.54	492	0.54	492	0.53	492
c-fat500-2	0.68	489	0.27	473	0.27	473
c-fat500-5	0.50	488	0.49	486	0.50	486
keller4	0.02	135	0.00	72	0.00	59
p_hat300-1	0.40	283	0.04	255	0.03	247
p_hat300-2	0.96	219	0.13	206	0.13	203
p_hat300-3	0.39	171	0.10	159	0.10	150
p_hat500-1	2.13	488	0.24	454	0.20	459
p_hat500-2	37.70	374	3.52	352	2.93	352
p_hat700-1	7.87	682	1.10	661	0.98	664
p_hat700-2	1,004.52	539	45.61	521	49.03	521
san1000	0.57	581	0.00	0	0.00	0
san400_0.5_1	0.09	190	0.00	0	0.00	0
san400_0.7_1	0.01	40	0.00	0	0.00	0
san400_0.7_2	0.03	133	0.00	0	0.00	0
san400_0.7_3	0.05	214	0.00	0	0.00	0
san400_0.9_1	0.00	0	0.00	0	0.00	0
sanr200_0.7	0.04	163	0.01	127	0.01	134
sanr200_0.9	0.00	15	0.00	6	0.00	7
sanr400_0.5	1.79	379	0.40	365	0.39	366

Table 4: DIMACS Instances: CLIQUER time and calls.

The results for $\mathcal{C}^{=}_{\text{cov}}|\mathcal{N}_{\text{red}}|$ and $\mathcal{C}_{\text{cov}}|\mathcal{N}_{\emptyset}|$ are more promising, with $\mathcal{C}^{=}_{\text{cov}}|\mathcal{N}_{\text{red}}|$ in particular showing a significant speedup. A partial explanation is that the time spent by CLIQUER is drastically smaller for $\mathcal{C}^{=}_{\text{cov}}|\mathcal{N}_{\text{red}}|$ than for $\mathcal{C}_{\emptyset}|\mathcal{N}_{\text{red}}|$. Note that, for the `san` graphs in particular, $\mathcal{C}^{=}_{\text{cov}}|\mathcal{N}_{\text{red}}|$ needs no calls at all to CLIQUER. This is due to decomposition.

As for $\mathcal{C}^{=}_{\text{cut}}|\mathcal{N}_{\text{red}}|$, its performance is good, but not quite as good as $\mathcal{C}^{=}_{\text{cov}}|\mathcal{N}_{\text{red}}|$. As in the case of the random graphs, aggressive generation of clique inequalities does not pay off for dense graphs. On the other hand, clique separation is (significantly) rewarding for `brock200_1`, `brock200_4`, `p_hat300-3` and `sanr200_0.7`.

A complementary comment concerns with the comparison between MIP models and CLIQUER. The solution times taken by the latter, reported in the last column of Table 3, reveal that CLIQUER is generally faster by one or two order of magnitude. Nevertheless, in six cases, it fails to solve the problem within the time limit and in two cases, namely, `p_hat300-3` and `san400_0.7_2`, it is largely outperformed.

We remark that the DIMACS test-set considered in this study extends those used in previous papers on exact MIP algorithms [1, 18–20, 40, 41]. In particular, `mann_a81`, `p_hat500-1`, `p_hat500-2`, `p_hat700-1`, `p_hat700-2` and `sanr400_0.5` have not been considered before, to the best of our knowledge. Moreover, based on the insight gained in our experiments, we were able to tackle three even harder instances, namely `brock400_4`, `p_hat500-3` and `keller5`. CLIQUER takes 176.6 CPU secs to solve `brock400_4` while fails on the others.

For `brock400_4` and `p_hat500-3`, it was enough to let CPLEX use 32 threads and use one of $\mathcal{C}_{\text{cov}}|\mathcal{N}_{\emptyset}|$, $\mathcal{C}^{=}_{\text{cov}}|\mathcal{N}_{\text{red}}|$ and $\mathcal{C}^{=}_{\text{cut}}|\mathcal{N}_{\text{red}}|$. Details are in the following table (speedup is measured w.r.t. $\mathcal{C}_{\text{cov}}|\mathcal{N}_{\emptyset}|$).

	$\mathcal{C}_{\text{cov}}\|\mathcal{N}_{\emptyset}\|$	$\mathcal{C}^{=}_{\text{cov}}\|\mathcal{N}_{\text{red}}\|$		$\mathcal{C}^{=}_{\text{cut}}\|\mathcal{N}_{\text{red}}\|$	
Graph	Time	Time	Speedup	Time	Speedup
brock400_4	3,686.4	2,732.3	1.4	1,646.8	2.24
p_hat500-3	3,890.5	1,962.3	1.98	1,215.0	3.20

Table 5: solution of `brock400_4` and `p_hat500-3`

By contrast, `keller5` required some specific tuning, as all previous models failed in the time limit. The best result was achieved by using $\mathcal{C}_{\emptyset}|\mathcal{N}|$, solving the LP relaxation, running the clique separation algorithm [27], and adding all generated cliques to `CPLEX` as "user cuts". The final formulation had 776 rows, 150,196 nonzeros, and 27,858 user cuts. It required about 840 seconds to be generated and about 170 seconds to be solved. It is worthwhile to mention that `keller5` turns out to be a challenging instance even for state-of-the-art combinatorial algorithms (see Table 3 of [45]).

To sum up the results of this section and the previous one, $\mathcal{C}^{=}_{\text{cov}}|\mathcal{N}_{\text{red}}|$

and $\mathcal{C}^{=}_{\text{cut}}|\mathcal{N}_{\text{red}}|$ are the most promising models.

7 Conclusions

This paper presents, to our knowledge, the first detailed study of alternative stable set formulations in the "natural" space of the node variables. We have shown that a careful selection and strengthening of clique and nodal inequalities can have a significant impact on the performance of MIP-based approaches to the stable set problem. In particular, the models $\mathcal{C}^{=}_{\text{cov}}|\mathcal{N}_{\text{red}}|$ and $\mathcal{C}^{=}_{\text{cut}}|\mathcal{N}_{\text{red}}|$ can be particularly useful, and have enabled us to solve some instances that have never before been solved with MIP-based approaches.

We remark that, in this paper, we have used (lifted) nodal inequalities only in the initial formulation. An interesting topic for future research is the dynamic generation of (lifted) nodal inequalities in a cut-and-branch or branch-and-cut algorithm.

To end the paper, we make some remarks about the performance of MIP algorithms relative to combinatorial algorithms. It is widely held (see, e.g., [45]) that combinatorial algorithms are superior. This is indeed true for the majority of the instances tested. For instance, in the `brock` graph family, instances with $n = 800$ have been solved by combinatorial methods, but are beyond the scope of current MIP models. However, for some instances, our best MIP configurations clearly outperform the combinatorial algorithms. Prominent examples are `mann_a45`, `mann_81` and `keller5`. So we believe that MIP approaches still warrant further investigation. Recall also that MIP algorithms tend to perform much better on weighted instances, whereas most combinatorial methods are geared toward cardinality instances.

References

[1] E. Balas, S. Ceria, G. Cornuéjols & G. Pataki (1996) Polyhedral methods for the maximum clique problem. In Johnson & Trick (eds.) *op. cit.*, pp. 11–28.

[2] B. Bollobás (1988) The chromatic number of random graphs. *Combinatorica*, 8, 49–55.

[3] I.M. Bomze, M. Budinich, P.M. Pardalos & M. Pelillo (1999) The maximum clique problem. In D.-Z. Du & P.M. Pardalos (eds.) *Handbook of Combinatorial Optimization: Supplement Volume A*, pp. 1–74. Dortrecht: Kluwer.

[4] R. Borndörfer (1998) *Aspects of Set Packing, Partitioning and Covering.* Doctoral Thesis, Technical University of Berlin.

[5] S. Burer & D. Vandenbussche (2006) Solving lift-and-project relaxations of binary integer programs. *SIAM J. Optim.*, 16, 726–750.

[6] S. Busygin (2006) A new trust region technique for the maximum weight clique problem. *Discr. Appl. Math.*, 154, 2080–2096.

[7] L. Cánovas, M. Landete & A. Marín (2000) New facets for the set packing polytope. *Oper. Res. Lett.*, 27, 153–161.

[8] R. Carraghan & P.M. Pardalos (1990) An exact algorithm for the maximum clique problem. *Oper. Res. Lett.*, 9, 375–382.

[9] E. Cheng & W.H. Cunningham (1997) Wheel inequalities for stable set polytopes. *Math. Program.*, 77, 389–421.

[10] E. Cheng & S. de Vries (2002). Antiweb-wheel inequalities and their separation problems over the stable set polytopes. *Math. Program.*, 92, 153–175.

[11] V. Chvátal (1975) On certain polytopes associated with graphs. *J. Combin. Th. Ser. B*, 18, 138–154.

[12] R.C. Corrêa, D. Delle Donne, I. Koch & J. Marenco (2017) General cut-generating procedures for the stable set polytope. *Discr. Appl. Math.*, to appear.

[13] F. Della Croce & R. Tadei (1994) A multi-KP modeling for the maximum-clique problem. *Eur. J. Oper. Res.*, 73, 555–561.

[14] DIMACS Repository
`http://archive.dimacs.rutgers.edu/pub/challenge/graph/benchmarks/`, last accessed June 2020.

[15] I. Dukanovic & F. Rendl (2007) Semidefinite programming relaxations for graph coloring and maximal clique problems. *Math. Program.*, 109, 345–365.

[16] P. Erdős & A. Rényi (1959) On random graphs. I. *Publicationes Mathematicae*, 6, 290–287.

[17] M. Giandomenico & A.N. Letchford (2006) Exploring the relationship between max-cut and stable set relaxations. *Math. Program.*, 106, 159-175.

[18] M. Giandomenico, A. Letchford, F. Rossi & S. Smriglio (2009) An application of the Lovász-Schrijver $M(K, K)$ operator to the stable set problem. *Math. Program.*, 120, 381–401.

[19] M. Giandomenico, A.N. Letchford, F. Rossi, & S. Smriglio (2014) Ellipsoidal relaxations of the stable set problem: theory and algorithms. *SIAM J. Optim.*, 25, 1944–1963.

[20] M. Giandomenico, F. Rossi & S. Smriglio (2013) Strong lift-and-project cutting planes for the stable set problem. *Math. Program.*, 141, 165–192.

[21] M. Grötschel, L. Lovász & A.J. Schrijver (1988) Stable sets in graphs. Chapter 9 of *Geometric Algorithms and Combinatorial Optimization.* New York: Wiley.

[22] G. Gruber & F. Rendl (2003) Computational experience with stable set relaxations. *SIAM J. Optim.*, 13, 1014–1028.

[23] J. Håstad (1999) Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math.*, 182, 105–142.

[24] D.S. Johnson & M.A. Trick (eds.) (1996) *Cliques, Coloring and Satisfiability: the 2nd DIMACS Implementation Challenge.* Providence, RI: American Mathematical Society.

[25] L. Lovász & A.J. Schrijver (1991) Cones of matrices and set-functions and 0-1 optimization. *SIAM J. Optim.*, 1, 166–190.

[26] E. Maslov, M. Batsyn & P.M. Pardalos (2013) Speeding up branch and bound algorithms for solving the maximum clique problem. *J. Glob. Optim.*, 59, 1–21.

[27] F. Marzi, F. Rossi & S. Smriglio (2019) Computational study of separation algorithms for clique inequalities. *Soft Computing*, 23(9), 3013—3027.

[28] D.W. Matula (1972) The employee party problem. *Notes Amer. Math. Soc.*, 19, A-382.

[29] D. Matula & L. Kučera (1990) An expose-and-merge algorithm and the chromatic number of a random graph. In M. Karonski, J. Jaworski & A. Rucinski (eds.) *Random Graphs '87.* New York: Wiley.

[30] T.S. Motzkin & E.G. Straus (1965) Maxima for graphs and a new proof of a theorem of Turán. *Canad. J. Math.*, 17, 533–540.

[31] A.T. Murray & R.L. Church (1997) Facets for node packing. *Eur. J. Oper. Res.*, 101, 598–603.

[32] G.L. Nemhauser & G. Sigismondi (1992) A strong cutting plane/branch-and-bound algorithm for node packing. *J. Oper. Res. Soc.*, 43, 443–457.

[33] G.L. Nemhauser & L.E. Trotter (1974) Properties of vertex packing and independence system polyhedra. *Math. Program.*, 6, 48–61.

[34] G.L. Nemhauser & L.A. Wolsey (1988) *Integer and Combinatorial Optimization.* New York: Wiley.

[35] S. Niskanen and P.R.J. Östergård, Routines for clique searching, http://users.aalto.fi/ pat/cliquer.html

[36] P.R.J. Östergård (2002) A fast algorithm for the maximum clique problem. *Discr. Appl. Math.*, 120, 197–207.

[37] M.W. Padberg (1973) On the facial structure of set packing polyhedra. *Math. Program.*, 5, 199–215.

[38] P.M. Pardalos & J. Xue (1994) The maximum clique problem. *J. Glob. Optim.*, 4, 301–328.

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

[40] S. Rebennack, M. Oswald, D.O. Theis, H. Seitz, G. Reinelt & P.M. Pardalos (2011) A branch and cut solver for the maximum stable set problem. *J. Comb. Optim.*, 21, 434–457.

[41] F. Rossi & S. Smriglio (2001) A branch-and-cut algorithm for the maximum cardinality stable set problem. *Oper. Res. Lett.*, 28, 63–74.

[42] P.S. Segundo, D. Rodrıguez-Losada & A. Jiménez (2011) An exact bit-parallel algorithm for the maximum clique problem. *Comput. Oper. Res.*, 38, 571–581.

[43] E.C. Sewell (1998) A branch and bound algorithm for the stability number of a sparse graph. *INFORMS J. Comput.*, 10, 438–447.

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

[45] Q. Wu & J.K. Hao (2015) A review on algorithms for maximum clique problems. *Eur. J. Oper. Res.*, 242, 693–709.

A Model sizes for random graphs

In this appendix, we report some figures concerned with the size of the various 0-1 LP models. Figures 6 and 7 show, for each model, the number of linear inequalities, the number of non-zeroes in the constraint matrix, and the number of genuine nodal inequalities. As in the previous charts, marker

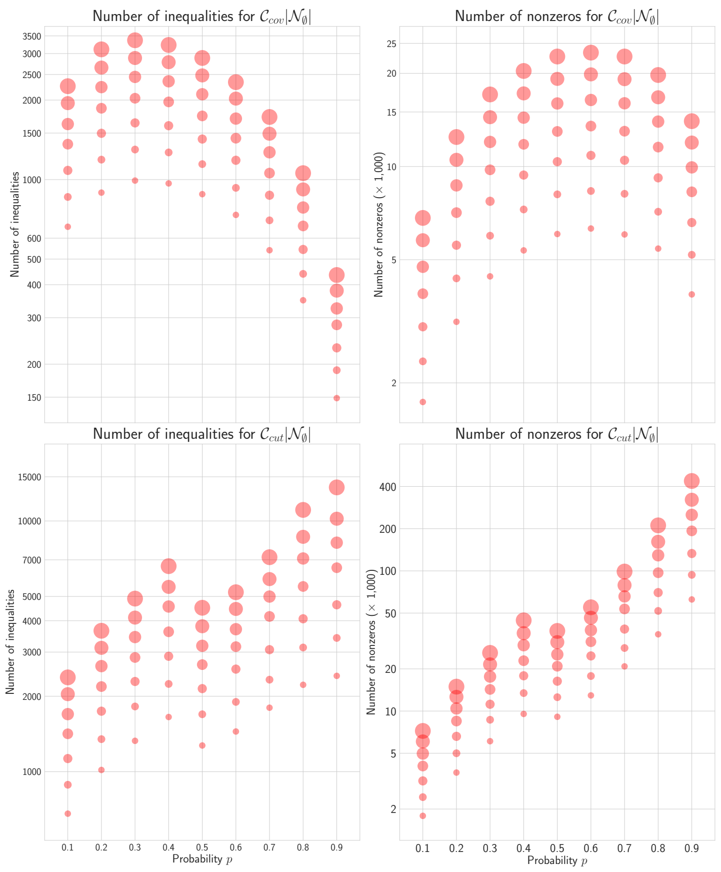

Figure 6: Features of models without nodal inequalities

size is proportional to n and blue markers refer to models in which reduction is applied. Charts reporting the number of inequalities and non-zeros have a log scale.

Figure 6 reports the data for $\mathcal{C}_{\text{cov}}|\mathcal{N}_\emptyset|$ and $\mathcal{C}_{\text{cut}}|\mathcal{N}_\emptyset|$. The decrease in model size when density reaches 0.5 is due to the fact that we change the clique violation parameter ϵ from 0.001 to 0.1 at that point.

Figure 7 reports on the models with nodal inequalities. When the graph density is 0.1, decomposition converts most or all nodal inequalities into clique inequalities, leading to 0-1 LPs with lots of constraints but low density. As density increases ($\geq 20\%$), the number of non-zeroes increases and the number of nodal inequalities approaches n. On the other hand, reduction of nodal inequalities significantly decreases the number of non-zeroes. (As one might expect, it is typically halved.) Finally, note that the number of constraints and non-zeroes for all models based on $\mathcal{C}_{\text{cov}}$ and $\mathcal{C}_{\text{cut}}$ are similar.

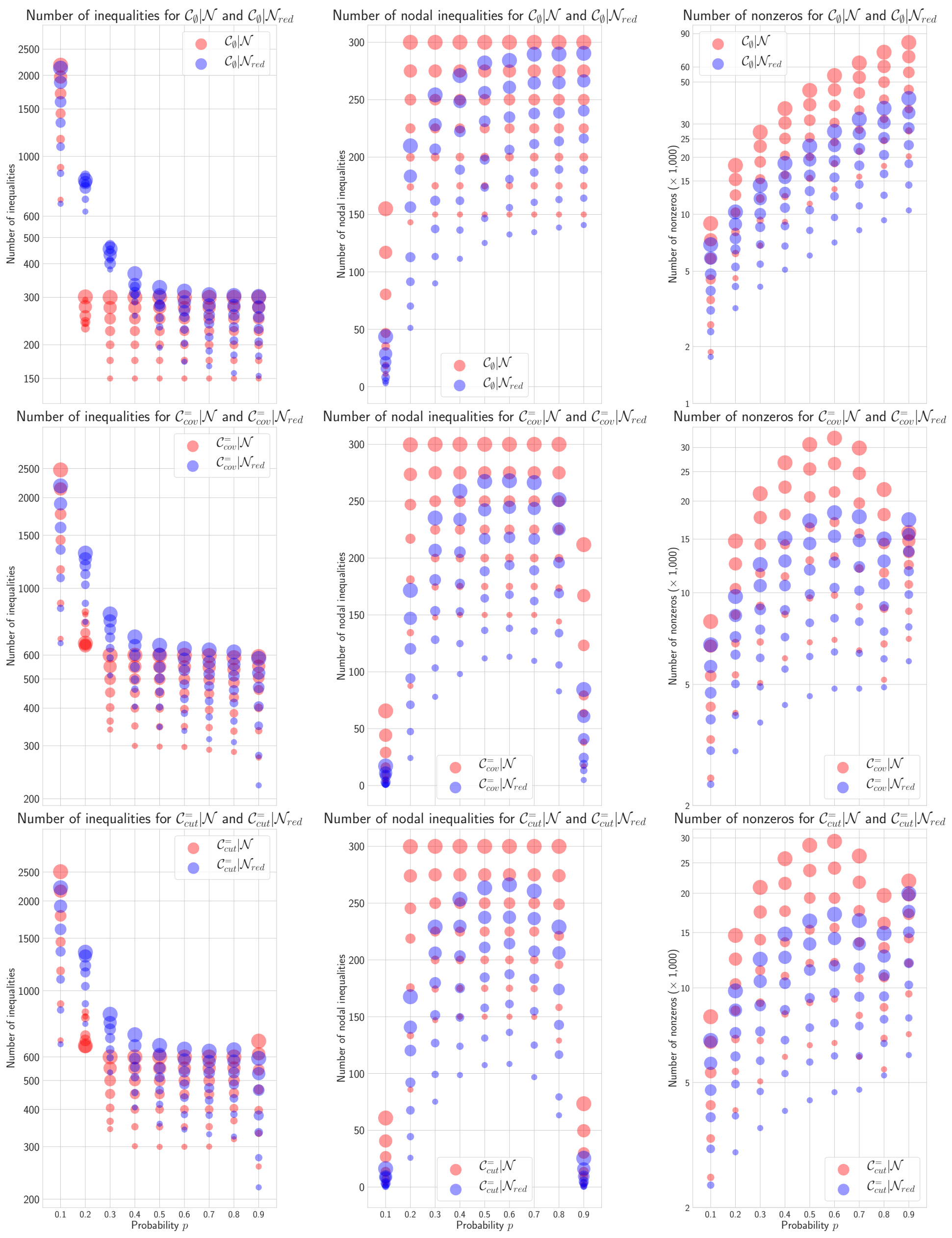

Figure 7: Features of models with nodal inequalities

B Enumerated subproblems

In this appendix, we report the number of branch-and-bound nodes needed to solve the various models for the test instances considered.

p	$\lvert V\rvert$	FRAC(G)	$\mathcal{C}_{\text{cov}}\lvert\mathcal{N}_{\emptyset}\rvert$	$\mathcal{C}_{\emptyset}\lvert\mathcal{N}_{\text{red}}\rvert$	$\mathcal{C}^{=}_{\text{cov}}\lvert\mathcal{N}_{\text{red}}\rvert$	$\mathcal{C}^{=}_{\text{cut}}\lvert\mathcal{N}_{\text{red}}\rvert$
0.1	150	18.4	16.5	17.5	16.7	15.9
	175	107.4	100.2	109.4	99.2	100.8
	200	741.9	705.3	840.0	668.9	615.6
	225	4,998.4	4,538.7	5,428.2	4,321.1	3,902.2
0.2	150	20.8	21.4	20.4	19.0	15.7
	175	104.7	125.5	106.2	75.5	69.6
	200	399.9	497.7	378.4	292.9	210.2
	225	1,645.0	2,200.2	1,658.8	1,102.0	915.3
	250	3,912.9	7,503.0	4,185.1	2,681.5	2,279.0
0.3	150	11.1	31.9	10.7	13.2	12.0
	175	36.7	120.9	34.1	40.9	34.8
	200	95.1	459.9	96.0	123.4	91.9
	225	273.6	1,501.5	295.5	369.3	297.6
	250	698.8	4,536.3	688.4	918.2	813.2
	275	1,802.2	10,889.2	1,774.9	2,181.0	1,673.3
	300	3,642.1	14,105.1	3,755.4	4,558.1	3,458.3
0.4	150	7.2	38.2	7.0	10.7	8.3
	175	13.2	85.9	12.4	19.3	16.4
	200	29.1	221.6	27.6	48.4	37.0
	225	83.9	593.5	78.9	159.4	114.8
	250	157.5	1,174.5	158.6	301.2	218.9
	275	325.1	2,272.2	299.7	495.5	418.1
	300	569.6	4,124.9	530.1	874.5	713.3
0.5	150	2.7	19.5	2.6	4.5	3.1
	175	7.4	40.6	6.7	10.3	8.1
	200	11.7	78.8	10.9	21.1	15.8
	225	25.6	187.1	22.6	43.2	34.7
	250	44.9	300.9	42.0	85.4	59.7
	275	80.8	480.3	74.1	172.4	129.0
	300	157.7	843.8	136.2	260.1	205.3

(a) $p \in \{0.1, \ldots, 0.5\}$
(Figures expressed in thousands)

p	$\lvert V\rvert$	FRAC(G)	$\mathcal{C}_{\text{cov}}\lvert\mathcal{N}_{\emptyset}\rvert$	$\mathcal{C}_{\emptyset}\lvert\mathcal{N}_{\text{red}}\rvert$	$\mathcal{C}^{=}_{\text{cov}}\lvert\mathcal{N}_{\text{red}}\rvert$	$\mathcal{C}^{=}_{\text{cut}}\lvert\mathcal{N}_{\text{red}}\rvert$
0.6	150	1,533.7	6,276.7	1,402.5	2,010.1	1,580.1
	175	1,632.3	12,126.2	2,345.1	3,759.1	2,611.5
	200	6,748.0	22,588.3	6,016.2	9,036.7	6,302.9
	225	11,715.5	39,803.3	11,311.5	15,650.3	12,459.0
	250	14,532.2	68,790.0	17,313.6	25,187.9	17,983.3
	275	27,744.3	136,705.3	24,271.8	43,816.5	29,471.6
	300	35,232.7	184,069.4	31,118.6	64,957.8	41,453.5
0.7	150	1.0	1,751.1	7.9	21.6	323.2
	175	3.2	3,323.2	347.4	120.3	1,000.6
	200	367.4	6,006.7	2,420.0	3,933.5	2,692.2
	225	2,347.2	8,830.0	4,745.8	6,961.0	5,191.0
	250	11,410.7	14,883.7	8,458.2	11,567.7	8,392.2
	275	12,371.2	18,507.0	9,521.9	9,565.9	9,562.1
	300	17,100.2	22,864.4	15,200.9	17,146.6	12,069.4
0.8	150	1.0	4.9	1.0	1.0	1.0
	175	1.0	30.5	1.0	6.1	68.8
	200	1.0	107.4	1.0	120.5	74.0
	225	1.0	33.9	2.7	660.6	652.4
	250	1.0	670.3	270.1	364.4	1,375.9
	275	1.0	2,947.7	1,300.0	3,406.7	2,413.4
	300	55.1	4,782.1	2,395.7	6,782.2	4,335.0
0.9	150	1.0	1.0	1.0	1.0	96.1
	175	1.0	1.0	1.0	1.0	233.1
	200	1.0	1.0	1.0	1.0	297.9
	225	1.0	1.0	1.0	1.0	340.6
	250	1.0	1.0	1.0	1.0	397.3
	275	1.0	1.0	1.0	2.8	428.7
	300	1.0	1.0	1.0	28.0	514.9

(b) $p \in \{0.6, \ldots, 0.9\}$

Table 6: Enumerated subproblems for G_{np} graphs

	FRAC(G)	$\mathcal{C}_{\text{cov}}\vert\mathcal{N}_\emptyset\vert$	$\mathcal{C}_\emptyset\vert\mathcal{N}_{\text{red}}\vert$	$\mathcal{C}^{=}_{\text{cov}}\vert\mathcal{N}_{\text{red}}\vert$	$\mathcal{C}^{=}_{\text{cut}}\vert\mathcal{N}_{\text{red}}\vert$
C125-9	1,512.9	1,473.0	1,566.9	1,665.3	1,595.8
DSJC125.1	3,017.0	3,005.8	2,958.6	3,211.4	3,045.5
DSJC125.5	731.6	673.5	4,702.0	957.0	923.2
MANN_a45	5,457.6	7,861.1	7,400.4	7,960.5	7,960.5
MANN_a81	110,173.7	213,099.1	156,374.6	52,447.5	52,447.5
brock200_1	127,422.3	124,203.9	436,194.0	97,132.8	69,965.0
brock200_2	5,225.9	5,698.7	34,047.4	8,541.9	6,883.7
brock200_3	16,757.9	14,330.8	175,927.2	20,123.9	15,173.3
brock200_4	26,172.4	28,087.1	224,541.2	30,943.5	23,619.3
c-fat200-5	1.0	6.0	163.4	144.0	144.0
c-fat500-1	1.0	1.0	1.0	1.0	1.0
c-fat500-10	1.0	1.0	300.4	1.0	1.0
c-fat500-2	1.0	1.0	111.3	1.0	1.0
c-fat500-5	1.0	1.0	265.5	1.0	1.0
keller4	2,228.5	2,825.3	58,430.9	2,643.5	1,987.8
p_hat300-1	6,556.0	5,305.2	2,694.7	7,372.7	6,725.2
p_hat300-2	3,691.7	2,505.9	14,252.0	3,552.2	2,990.6
p_hat300-3	89,316.1	74,430.7	109,680.7	46,717.6	27,655.8
p_hat500-1	38,984.1	36,781.7	47,159.5	45,403.8	41,892.4
p_hat500-2	31,292.3	30,262.0	146,402.8	29,770.6	26,977.7
p_hat700-1	87,524.8	71,018.0	2,815.6	97,360.2	93,692.1
p_hat700-2	242,023.2	201,541.8	603,979.8	131,010.8	129,413.6
san1000	1.0	1.0	552.0	1.0	1.0
san400_0.5_1	1.0	1.0	1.0	1.0	1.0
san400_0.7_1	1.0	1.0	1.0	1.0	1.0
san400_0.7_2	1.0	1.0	1.0	1.0	1.0
san400_0.7_3	1.0	1.0	1.0	1.0	1.0
san400_0.9_1	1.0	1.0	1.0	1.0	1.0
sanr200_0.7	72,649.2	69,740.9	462,536.1	62,815.5	50,354.8
sanr200_0.9	268,008.5	262,676.4	246,479.6	203,523.0	181,005.4
sanr400_0.5	771,069.4	628,429.6	3,886,181.9	764,598.5	743,241.5

Table 7: Enumerated subproblems for DIMACS graphs