Strengthening Chvátal-Gomory Cuts for the Stable Set Problem

Adam N. Letchford[1], Francesca Marzi[2], Fabrizio Rossi[2], and Stefano Smriglio[2](✉)

[1] Department of Management Science, Lancaster University, Lancaster, UK
A.N.Letchford@lancaster.ac.uk

[2] Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, L'Aquila, Italy
francesca.marzi@graduate.univaq.it,
{fabrizio.rossi,stefano.smriglio}@univaq.it

Abstract. The stable set problem is a well-known $\mathcal{NP}$-hard combinatorial optimization problem. As well as being hard to solve (or even approximate) in theory, it is often hard to solve in practice. The main difficulty is that upper bounds based on linear programming (LP) tend to be weak, whereas upper bounds based on semidefinite programming (SDP) take a long time to compute. We propose a new method to strengthen the LP-based upper bounds. The key idea is to take violated Chvátal-Gomory cuts and then strengthen their right-hand sides. Although the strengthening problem is itself $\mathcal{NP}$-hard, it can be solved reasonably quickly in practice. As a result, the overall procedure proves to be capable of yielding competitive upper bounds in reasonable computing times.

Keywords: Stable Set Problem · Clique inequalities · Chvátal-Gomory cuts · Cutting plane algorithm

1 Introduction

Given an undirected graph $G = (V, E)$, a *stable set* in G is a set of pairwise non-adjacent vertices. The convex hull of the incidence vectors of all stable sets in G is called the *stable set polytope* and denoted by STAB(G) [19]. The *Stable Set Problem* (SSP) calls for a stable set of maximum cardinality $\alpha(G)$, or, if a weight vector $w \in \mathbb{Q}^n_+$ is given, of maximum weight $\alpha_w(G)$. The SSP is strongly $\mathcal{NP}$-hard even to approximate [22]; and it is naturally stated as the binary program $\max\{\sum_{i\in V} w_i x_i : x_i + x_j \le 1 \ \forall\{i,j\} \in E, x \in \{0,1\}^{|V|}\}$. Optimizing over its continuous relaxation provides very weak upper bounds on $\alpha_w(G)$. Therefore, a great effort has been devoted to improving the basic relaxation FRAC(G) $= \{x \in [0,1]^{|V|} : x_i + x_j \le 1 \ \forall\{i,j\} \in E\}$, by studying valid inequalities for STAB(G). The first steps are due to Padberg, who introduced the *clique* inequalities [28]. These have the form $\sum_{i\in C} x_i \le 1$, for any $C \subseteq V$ inducing a *maximal* clique in G, and induce facets of STAB(G). The polytope defined by all clique and nonnegativity inequalities is denoted by QSTAB(G) [19].

R. Cerulli et al. (Eds.): ISCO 2016, LNCS 9849, pp. 201–212, 2016.
DOI: 10.1007/978-3-319-45587-7_18

Many other valid inequalities, such as the *odd hole* and *odd antihole*, *web* and *antiweb* inequalities, have been derived. We refer the reader to [2,14,19] for detailed surveys. The Chvátal rank of some of these inequalities with respect to FRAC(G) and QSTAB(G) has been investigated in [23]. On the computational side, after the pioneering experience illustrated in [27] with clique and lifted odd-hole inequalities, more extensive results have been obtained with general *rank inequalities* and some of their (non-rank) lifted versions. These have been generated by *project-and-lift* separation heuristics introduced in [30] and recently improved in [8,9,29]. A study concerned with the exact separation of rank inequalities is described in [10]. Despite the fairly sophisticated techniques explored in these papers, the resulting upper bounds are not yet satisfactory for several graph classes.

Much stronger upper bounds can be obtained by Semidefinite Programming (SDP) relaxations. In the seminal paper [25], Lovász introduced the *theta function*, denoted by $\vartheta(G)$, as the optimal value of a SDP problem (we refer the reader to [19] for a comprehensive introduction). It has been proved that $\vartheta(G)$ dominates the bound obtained by optimizing over QSTAB(G) [19], and it is often much stronger in practice. Some classes of graphs for which this occurs are illustrated in [31], while a computational comparison is documented in [14]. Computational experiments with $\vartheta(G)$, or stronger relaxations obtained by adding valid linear inequalities, are presented in [4,11,15,20,24]. These approaches typically require long computing times. In order to manage this difficulty, ellipsoidal relaxations have been introduced [16], which allow one to obtain useful convex programming relaxations and derive effective cutting planes. In fact, this method allows one to achieve upper bounds close to $\vartheta(G)$ by optimizing over a linear relaxation.

Strong upper bounds have also been obtained by applying the Lovász and Schrijver lift-and-project operators [26] to FRAC(G). The N operator, based on LP, has been tested by Balas *et al.* [3]. The N^+ operator, based on SDP, yields a much stronger relaxation than the N operator, but it is often very hard to solve in practice. Computational experiments are presented in [5]. Finally, the $M(k,k)$ operator has been applied to QSTAB(G) [13,14]: the resulting non-compact linear relaxations turns out to provide upper bounds comparable to those from SDP relaxations at reasonable computational cost.

We propose a new method to strengthen the LP-based upper bounds. The key idea is to take violated Chvátal-Gomory cuts and then strengthen their right-hand sides relative to STAB(G). Although the strengthening problem is itself a SSP, a careful selection of the source cut can make it computationally tractable in practice. We present a cutting-plane algorithm based on the strengthened cuts and show that it is capable of yielding competitive upper bounds in moderate computing times. The algorithm is illustrated in the next section, while the computational experience is described in Sect. 3. Finally, some conclusions are drawn in Sect. 4.

2 Cutting Plane Algorithm

We consider an initial formulation of the SSP based on clique inequalities. In general, G may have exponentially many cliques and the separation problem associated to clique inequalities is strongly $\mathcal{NP}$-hard [27]. Nevertheless, greedy-like separation heuristics perform extremely well: experience shows that a cutting-plane algorithm embedding such a heuristic often achieves upper bounds quite close to those obtained by exactly optimizing over QSTAB(G). We therefore concentrate on the collection $\mathcal{C}$ of cliques generated by such an algorithm (see [13,14] for details) and consider the basic relaxation $\mathcal{Q}(\mathcal{C}) = \{Ax \leq \mathbf{1}, x \geq 0\}$, where $A = A_{\mathcal{C}}$ is the incidence matrix of the cliques in $\mathcal{C}$ versus the vertices of G. We also let $UB_{\mathcal{C}} = \{\max w^T x : x \in \mathcal{Q}(\mathcal{C})\}$ be the associated upper bound on $\alpha(G)$.

We experiment with a cutting plane algorithm based on the classical Chvátal-Gomory (CG) cuts [7,17,18]. These have the form

$$\lfloor u^T A \rfloor x \leq \lfloor u^T b \rfloor, \qquad u \in \mathbb{R}^m_+.$$

The cut generation procedure has two main stages:

1. Identify a violated CG cut $\lambda^T x \leq \lambda_0$
2. Strengthen λ_0 relative to STAB(G)

which are described in the following subsections.

2.1 Cut Separation

The choice of $u \in \mathbb{R}^m_+$ is critical for deriving useful inequalities. The classical idea from Gomory [17,18] is based on *basic* solutions. If one considers $\mathcal{Q}(\mathcal{C})$ in standard form (A, I) and a fractional vertex x^* associated with a basis B, any row i of B^{-1} associated with a fractional component of x^* determines a vector of multipliers u such that the resulting CG inequality cuts off x^*. In what follows, we refer to these inequalities as CG cuts from the tableau.

A different approach to obtaining useful CG cuts has been proposed in [12], where the separation problem associated to CG cuts (for a general MIP) is formulated as a MIP. Let x^* be the current fractional point and denote by $J(x^*) = \{j \in \{1, \ldots, n\} : 0 < x^*_j < 1\}$ the associated fractional support. Let also $\lambda^T x \leq \lambda_0$ be the CG cut to be generated, where $\lambda = \lfloor u^T A \rfloor$ and $\lambda_0 = \lfloor u^T b \rfloor$ for some multiplier $u \in \mathbb{R}_+$. The MIP-CG separation model has the form

$$\begin{aligned}
&\max \; \Big(\sum_{j \in J(x^*)} \lambda_j x^*_j - \lambda_0\Big) - \sum_{i=1}^{m} \gamma_i u_i \\
&\text{s.t.} \\
&\qquad f_j = u^T A_j - \lambda_j, \quad j \in J(x^*) \\
&\qquad f_0 = u^T b - \lambda_0 \\
&\qquad 0 \leq f_j \leq 1 - \delta, \qquad j \in J(x^*) \cup \{0\} \\
&\qquad 0 \leq u_i \leq 1 - \delta, \qquad i = 1, \ldots, m \\
&\qquad \lambda_j \text{ integer} \qquad j \in J(x^*) \cup \{0\}.
\end{aligned}$$

Some parts of this model are redundant and have been introduced in [12] for technical reasons. In detail, the explicit slack variables $f_j = u^T A_j - \lfloor u^T A_j \rfloor$, along with the parameter δ (fixed to 0.01), improve numerical tractability; and the constraints $u_i \leq 1 - \delta$ reduce the chance of generating dominated cuts. Our experience confirmed that these arrangements are indeed helpful. Another key feature of MIP-CG deals with the objective function: besides cut violation $\sum_{j \in J(x^*)} \lambda_j x_j^* - \lambda_0$, it includes a penalty on multipliers $\sum_{i=1}^m \gamma_i u_i$, with $\gamma_i = 10^{-4}$, which helps to make the cut sparser and stronger. According to [12], the penalty term γ_i has to be applied only to tight constraints, that is, those for which $s_i^* = 1 - a_i^T x^* = 0$. Sparsity is, in general, an important feature for a cutting plane to be numerically well-behaved. In our development it is also crucial to make the strengthening stage tractable, as discussed later.

Due to all of these modifications, MIP-CG is not an exact separation oracle any more. However, as pointed out in [12], exceptions are likely to occur only in pathological cases. We solve MIP-CG by the commercial MIP solver IBM Cplex 12.6.3 (default settings). The computation is stopped using two parameters `cutviolation` and `septlim`. In detail, the solver halts if: (*i*) a feasible solution of value greater than or equal to `cutviolation` has been found; (*ii*) the elapsed cpu time reaches `septlim`. In both cases, we store the whole catalog of violated inequalities corresponding to feasible solutions contained in Cplex pool at termination.

2.2 Cut Strengthening

Andersen and Pochet [1] present a general method to strengthen the left-hand side (lhs) coefficients and right-hand side (rhs) of inequalities relative to the mixed-integer hull. Their theoretical development suggests that the rhs should be strengthened before the lhs coefficients are strengthened. In our context, strengthening the rhs of a cut $\lambda^T x \leq \lambda_0$ relative to the integer hull amounts to solving the problem $\lambda_0^* = \max\{\lambda^T x : x \in \text{STAB}(G)\}$ and replacing λ_0 by λ_0^*. This can be translated into the integer program $\{\max \lambda^T x : x \in \mathcal{P}, x \in \{0,1\}\}$, referred to as MIP-RHS = MIP-RHS($\mathcal{P}$), where $\mathcal{P}$ is any linear formulation of the problem. Notice that the number of nonzero entries of λ determines the actual size of the subproblem MIP-RHS to be solved. In other words, the sparser the cut, the easier the strengthening problem. Again, MIP-RHS is solved by Cplex to which a time limit `rhstlim` is imposed. When it is reached, λ_0^* is set equal to the best upper bound returned by the solver at the time limit.

The overall cutting-plane algorithm is summarized in Algorithm 1. The parameter `niter` determines the maximum number of iterations (i.e., the maximum number of cuts generated); `maxpercnz` establishes the maximum percentage cut density allowed for CG cuts from the tableau; and `minimprove` stops the algorithm when tailing-off is reached: it establishes the minimum improvement of the objective value between consecutive iterations required to proceed.

In the separation stage of Algorithm 1 a first CG cut, namely, the one with the smallest *support* (number nnz(λ) of nonzero coefficients), is obtained from the tableau. Ties are broken by selecting the cut that forms the minimum angle with the objective function. Then, if this cut turns out to be too dense, the exact MIP-CG separation is invoked with the aim of detecting a sparser one, with the exception of the first iteration, when the cut from the tableau is always preferred. Notice that MIP-CG always generates rank-1 CG cuts of relaxation $Q(\mathcal{C})$, as the generated cutting planes are never added to it. This turned out to be useful to keep safe the sparsity of the cuts. The rationale for this policy is that, although MIP-CG may be time consuming, it is typically largely counterbalanced by the saving yielded by a sparser source cut when solving MIP-RHS.

Algorithm 1. Cutting plane algorithm

Input:	Formulation $\mathcal{Q}(\mathcal{C})$
Output:	An updated formulation $\mathcal{P}$, the upper bound CG-S;
Parameters:	`niter,cutviolation,septlim,` `rhstlim,maxpercnz,minimprove`

```
$\mathcal{P} \leftarrow \mathcal{Q}(\mathcal{C})$
Optimize over $\mathcal{P}$, get $x^*$
for ($i := 1$ to niter and $x^*$ is fractional) do
  Evaluate all violated Gomory cuts from the current tableau
  Select the sparsest cut $(\lambda, \lambda_0)$
  if (nnz $(\lambda) >$ maxpercnz $* |V|$) and $i > 1$ then
    Solve MIP-CG($\mathcal{Q}(\mathcal{C})$, cutviolation,septlim)
    Select the sparsest violated cut $(\beta, \beta_0)$
    if nnz $(\lambda) >$ nnz $(\beta)$ then
      $\lambda := \beta, \lambda_0 := \beta_0$
    end if
  end if
  $\lambda_0^* \leftarrow$ Solve MIP-RHS($\mathcal{P}$, rhstlim)
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{\lambda^T x \leq \lambda_0^*\}$
  $\bar{x} \leftarrow x^*$
  Optimize over $\mathcal{P}$, get $x^*$
  if $(w^T\bar{x} - w^T x^*) <$ minimprove then
    return CG-S $= w^T x^*$
  end if
end for
return CG-S $= w^T x^*$
```

3 Computational Achievements

The upper bound CG-S computed by Algorithm 1 is now compared to those achieved by other methods in the literature. The test-bed consists of the DIMACS Second Challenge (Johnson and Trick [21]) benchmark collection, available at the web site [6], representing the standard benchmark for evaluating MSS and max-clique algorithms. We consider the complemented version of these graphs, as they were originally created for the max-clique problem. We include all the graphs with $n \leq 400$ except the "easy" ones, i.e., those for which that upper bound $UB_{\mathcal{C}}$ is close to the integer optimum $\alpha(G)$. The latter include the whole family of `johnson` graphs and most of the `c-fat`, `hamming` and `san` graphs. All instances are unweighted instances, as these tend to be the most difficult in practice. The computations are run on a machine with processor AMD Opteron 6376 (64 cores) clocked at 1.4 GHz with 64 GB RAM. The LP-MIP solver is IBM CPLEX 12.6.3 (using 32 threads): settings are default for MIP-CG, while `mipemphasis` is set to *moving best bound* for CG-RHS. The parameter settings for Algorithm 1 are as follows: `niter` $= 30$, `cutviolation` $= 0.2$, `septlim` $\in \{5, 50\}$, `rhstlim` $\in \{100, 150\}$, `maxpercnz` $\in \{0.9, 0.7\}$, `minimprove` $= 0.01$. Pairs $\{x, y\}$ of values indicate that the parameter assumes the value x for $|V| \leq 200$ and y otherwise. Before going through the evaluation of CG-S we analyze the strength of the first Chvátal closure of $\mathcal{Q}(\mathcal{C})$.

3.1 On the Strength of the First Chvátal Closure of QSTAB(G)

Let us denote by $\mathcal{Q}^1(\mathcal{C})$ the first Chvátal closure of $\mathcal{Q}(\mathcal{C})$ and by $UB_{\mathcal{Q}^1(\mathcal{C})} = \{\max \mathbb{1}^T x : x \in \mathcal{Q}^1(\mathcal{C})\}$. A close approximation to $UB_{\mathcal{Q}^1(\mathcal{C})}$ (unless pathological cases) is obtained by a cutting plane algorithm which uses MIP-CG as separation oracle. Table 1 compares $UB_{\mathcal{Q}(\mathcal{C})}$ and $UB_{\mathcal{Q}^1(\mathcal{C})}$ and shows the percentage gap closed by the first Chvátal closure.

In 16 out of 26 cases the gap closed is less than 2 %, in 21 cases less than 6 % and only in two cases greater than 10 %. Overall it turns out to that $\mathcal{Q}^1(\mathcal{C})$ is almost as tight as $\mathcal{Q}(\mathcal{C})$. This also gives a strong pointer about the strength of the Chvátal closure of QSTAB(G), which includes well known inequalities, such as odd-hole, odd-antihole and antiweb inequalities [23]. These results provide a benchmark to demonstrate the remarkable effect of cut strengthening, as documented below.

3.2 Evaluation of CG-S

Table 2 compares the upper bound CG-S returned by Algorithm 1 to the following upper bounds: $UB_{\mathcal{C}}$; $\vartheta(G)$; BCS, obtained by separation algorithms for rank inequalities and local cuts [29]; CMDK, obtained by separation algorithms for rank and non-rank inequalities [8]; MKK, computed by the Lovász and Schrijver $M(k, k)$ lifting operator applied to QSTAB(G) [14]; BLP derived from MKK through projection [13]; GR, obtained by strengthening the Lovász theta relaxation with odd circuit and triangle inequalities [20]; DR, obtained by tailored SDP algorithms to compute ϑ^+ [11]; BV, computed by optimizing over the Lovász and Schrijver lifting operator M_+ applied to FRAC(G) [5]; L, computed by strengthening the ϑ bound with non valid inequalities [24]; ELL achieved by outer approximation of ellipsoidal relaxations [16]. An asterisk in the DR or BV columns means that result was not reported.

The results clearly show the high quality of CG-S. In 10 out of 26 cases CG-S is the (unique) best upper bound while this holds 9 times for $\vartheta(G)$ and MKK.

Table 1. Upper bounds from the first Chvátal closure

Graph	$\alpha(G)$	$UB_{\mathcal{Q}(\mathcal{C})}$	$UB_{\mathcal{Q}^1(\mathcal{C})}$	$\frac{UB_{\mathcal{Q}(\mathcal{C})}-UB_{\mathcal{Q}^1(\mathcal{C})}}{UB_{\mathcal{Q}(\mathcal{C})}-\alpha(G)}\%$
brock200_1	21	38.02	37.83	1.12
brock200_2	12	21.21	21.12	0.98
brock200_3	15	27.3	27.22	0.65
brock200_4	17	30.66	30.54	0.88
brock400_1	27	63.96	63.92	0.11
brock400_2	29	64.39	64.34	0.14
brock400_3	31	64.18	64.13	0.15
brock400_4	33	64.21	64.16	0.16
C125.9	34	43.05	42.59	5.08
C250.9	44	71.39	70.99	1.46
c-fat200-5	58	66.67	65.76	10.5
DSJC125.1	34	43.16	42.64	5.68
DSJC125.5	10	15.39	15.25	2.6
mann_a9	16	18	17	50
mann_a27	126	135	134.15	9.44
hamming6-4	4	5.33	5.23	7.52
keller4	11	14.82	14.76	1.57
p_hat300-1	8	15.26	15.24	0.28
p_hat300-2	25	33.59	33.54	0.58
p_hat300-3	36	54.33	54.18	0.82
san200_0.7-2	18	20.36	20.28	3.39
san200_0.9-3	42	45.13	44	36.1
sanr200_0.7	18	33.34	33.17	1.11
sanr200_0.9	42	59.82	59.39	2.41
sanr400_0.5	13	41.29	41.26	0.11
sanr400_0.7	21	57.02	56.96	0.17

In 11 out of 24 cases it outperforms $\vartheta(G)$ (to the best of our knowledge $\vartheta(G)$ has never been computed for the two `sanr400` graphs) and in all the remaining cases it is quite close to $\vartheta(G)$. Looking at the other bounds, CG-S is the best bound ever computed for all instances in the hard classes `brock`, `p_hat` and `sanr`. It is also evident that BCS and CMDK, obtained by inequalities with a combinatorial structure, tend to be weaker in general. However, CMDK performed pretty well in some specific instances.

Table 3 reports the times in seconds required to compute several among the tightest upper bounds. The computing times of MKK, BLP, ELL are reported as in the original papers. These refer to different computers: all of them have

Table 2. Comparison among upper bounds

Graph	$\alpha(G)$	UB_C	$\vartheta(G)$	BCS	CMDK	MKK	BLP	GR	DR	BV	L	ELL	CG-S
brock200_1	21	38.20	27.5	*	34.85	30.25	33.59	*	*	27.98	*	28.53	24.56
brock200_2	12	21.53	14.22	20.99	16.29	16.09	18.27	*	*	17.08	*	15.16	13.79
brock200_3	15	27.73	18.82	*	23.26	21.16	23.55	*	*	20.79	*	19.6	16.97
brock200_4	17	30.84	21.29	29.93	26.81	23.8	26.77	*	*	22.8	*	22.35	20.34
brock400_1	27	68.47	39.70	*	*	*	*	*	*	*	*	41.83	45.08
brock400_2	29	68.28	39.56	63.84	63.27	*	*	*	*	*	*	42.82	44.90
brock400_3	31	68.42	39.48	*	*	*	*	*	*	*	*	41.05	45.71
brock400_4	33	64.21	39.60	63.89	63.17	*	*	*	*	*	*	41.26	43.77
C.125.9	34	43.06	37.89	41.26	38.84	36.53	37.81	*	*	*	37.06	38.48	38.72
C.250.9	44	71.50	56.24	69.76	65.35	59.96	63.95	*	*	*	*	58.09	56.45
c-fat200-5	58	66.67	60.34	58.89	58	58	58	*	*	58.17	*	*	62.27
DSJC125.1	34	43.15	38.39	*	39.41	36.99	38.22	*	*	*	*	39.16	38.66
DSJC125.5	10	15.60	11.47	*	11.97	11.41	13.21	*	11.4	*	11.46	12.24	11.69
mann_a9	16	18.50	17.47	*	18	16.85	17.11	17.47	*	17.17	17.29	*	16.89
mann_a27	126	135.00	132.76	*	135	131.39	132.44	132.76	*	*	*	*	132.05
hamming6-4	4	5.33	5.33	*	4	4	4.64	*	4	4.54	*	*	4.47
keller4	11	14.82	14.01	14.83	14.2	13.17	14.29	*	*	15.41	13.15	14.55	13.60
p_hat300_1	8	15.68	10.1	*	12.43	11.4	13.45	*	*	18.66	*	11.13	11.39
p_hat300_2	25	34.01	27	33.81	33.5	30	30.73	*	*	30.1	*	28.5	28.60
p_hat300_3	36	54.74	41.16	54.12	51.82	47.32	49.79	*	*	43.32	*	43.02	41.08
san200_0.7-2	18	21.14	18	18.5	18.54	18	18	*	*	20.01	*	*	18.70
san200_0.9-3	44	45.13	44	44	44	44	*	44	*	44.4	*	*	44
sanr200_0.7	18	33.48	23.8	*	30.21	26.12	29.45	*	*	24.97	*	24.75	22.58
sanr200_0.9	42	60.04	49.3	*	54.18	50.73	54.52	49.27	*	49.31	*	50.6	47.74
sanr400_0.5	13	41.30	*	*	*	*	*	*	*	*	*	*	24.08
sanr400_0.7	21	57.02	*	*	*	*	*	*	*	*	*	*	38.61

Table 3. Computational times (sec.) and details

Graph	MKK	BLP	ELL	CG-S	MIP-CG time	MIP-RHS time	#cuts	#cuts tableau	Average frac	$\|\mathcal{C}\|$	$UB_{\mathcal{C}}$ time
brock200_1	17,670	373	9.88	87.95	2.61	82.71	16	14	116.75	2,127	1.10
brock200_2	26,501	190	5.8	93.03	14.92	72.74	12	1	101	3,902	0.65
brock200_3	22,386	338	14.37	115.8	0	111.82	16	16	97.81	2,867	3.84
brock200_4	25,362	196	20.12	92.88	7.11	83.03	15	9	109.8	2,567	2.21
brock400_1	*	*	11.43	661.99	110.93	510.07	18	1	269.28	5,676	15.93
brock400_2	*	*	5.88	838.44	110.29	681.67	21	1	268.24	5,713	17.26
brock400_3	*	*	5.44	923.55	172.79	683.1	26	1	278.69	5,673	16.89
brock400_4	*	*	5.83	1,091.41	143.9	896.7	16	1	285.5	7,325	12.74
C.125.9	227	391	0.6	9.48	4.28	5.1	15	8	111.33	489	0.06
C.250.9	9,397	8,908	5.02	323.94	11.71	309.63	20	9	201.85	1,724	0.40
c-fat200-5	265	45	*	16.17	0	6.85	16	16	191.12	7561	1.34
DSJC125.1	274	297	0.44	6.48	1.76	4.62	14	11	106.93	464	0.06
DSJC125.5	377	27	2.27	19.55	3.55	15.21	11	5	81.09	1,522	0.17
mann_a9	0.41	0.26	*	3.95	0.14	3.83	18	4	40.44	48	0.01
mann_a27	393	120	*	1.63	0	1.55	20	20	370.05	468	0.07
hamming6-4	4	5	*	48.5	43.89	4.53	16	1	30.06	149	0.02
keller4	15,324	9,586	0.64	30.07	7.51	22.12	13	1	86.85	868	0.54
p_hat300_1	4,910	767	4.23	66.26	5.61	59.54	12	1	78.25	1,124	1.78
p_hat300_2	24,337	2,207	3.35	95.92	13.76	80.69	11	5	127.09	2,016	1.03
p_hat300_3	46,408	2,419	25.94	255.36	18.68	228.91	12	7	192.17	4,074	9.41
san200_0.7-2	300	151	*	47.93	20.2	26.36	14	1	138.43	1,537	1.16
san200_0.9-3	143	*	*	0.24	0	0.23	1	1	152	1,143	0.14
sanr200_0.7	9,971	762	6.02	64.95	5.29	57.46	14	8	114.43	2,280	1.64
sanr200_0.9	8,483	949	1.42	56.68	5.15	50.87	15	8	156.6	1,150	0.25
sanr400_0.5	*	*	*	1,637.08	137.21	1,454.1	23	1	207.65	4,886	9.80
sanr400_0.7	*	*	*	1,551.21	285.11	1,152.83	25	1	267.84	8,540	25.36

CPU-s with higher clock frequency but a smaller number of cores. Although the comparison is not rigorously documented, these values can be considered reliable enough for a general judgment. In the column CG-S the total time required by Algorithm 1 is reported, while the successive two columns contain the overall time spent for solving MIP-CG and MIP-RHS respectively. The remaining columns report: the number of cuts generated; the number of cuts generated from the tableau; the average size of the fractional support and, finally, the number of clique inequalities in the initial formulation $Q(\mathcal{C})$ and the time required to construct it and compute $UB_{\mathcal{C}}$.

Table 3 shows that the strong bounds are achieved by a few cuts. Indeed, the very first cuts turn out to close a significant portion of the integrality gap. Notice also that the number of cuts generated by solving MIP-CG is large, which highlights that the adjustments of MIP-CG towards sparsification play a role. In 12 cases only one cut from the tableau is selected. It is indeed the first cut: at the first iteration even the cuts from MIP-CG are very dense and are not worth generating.

The average time for a single cut separation is quite reasonable and, as one can expect, most of this time is spent in solving MIP-RHS. These facts suggest that strengthened CG cuts can be cost-effective when embedded in a branch-and-cut framework.

Looking at computing times, the proposed method is outperformed only by ELL [16], while it is competitive with the other LP-based methods. Even if a direct comparison cannot be done, methods GR, DR, BV, L, based on sophisticated SDP approaches tend to be slower.

Another important fact is that the size of the average fractional support is often around $0.5|V| - 0.7|V|$. In our experience this nice effect is rarely observable with other cutting planes which typically keep $|J(x^*)| \simeq |V|$. This of course impacts on the efficiency of the method, which turns out to practical for graphs with 400 vertices.

It is worthwhile to remark that these experiments were carried out with general-purpose parameter settings, and that results on specific graphs may improve significantly with dedicated tuning.

The overall picture of this experience is that strengthened CG cuts can be quite effective even for a very structured combinatorial optimization problem such as the SSP. In fact, they seem to be competitive with inequalities that are derived from polyhedral studies. Notice also that the latter tend to be sparser than general CG cuts, a feature that usually guarantees a better numerical behaviour. Nevertheless, the above results, along with our previous experience with other general cutting planes, show that some denser cuts are required in order to achieve very strong bounds. This is a key issue for the development of IP algorithms for the SSP and deserve further investigation.

4 Conclusions

We showed that strengthening the right-hand-side of rank-1 CG cuts from a clique relaxation relative to the stable set polytope is extremely effective. In particular, the upper bounds obtained are competitive to those from sophisticated SDP approaches. This is so even though our implementation of the strengthening procedure is rather rudimentary and has significant room for improvement. In fact, one major research direction deals with speeding up the strengthening stage, either by using a combinatorial solver for the weighted SSP instances, or by using upper bounds on the weighted stability number that are faster to compute. Overall, our feeling is that the method can be improved so as to tackle larger graphs. A natural development will also be testing these cutting planes in a branch-and-cut framework. Finally, a theoretical study of strengthened rank-1 CG cuts would be interesting. It can be shown, for example, that the odd hole, odd antihole, web and antiweb inequalities, along with certain lifted versions of them, are cuts of this type.

References

1. Andersen, K., Pochet, Y.: Coefficient strengthening: a tool for reformulating mixed-integer programs. Math. Program. **122**, 121–154 (2010)
2. Borndörfer, R.: Aspects of Set Packing, Partitioning and Covering. Doctoral thesis, Technical University of Berlin (1998)
3. Balas, E., Ceria, S., Cornuéjols, G., Pataki, G.: Polyhedral methods for the maximum clique problem. In: Johnson, D.S., Trick, M.A. (eds.) Cliques, Coloring and Satisfiability. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 26, pp. 11–28 (1996)
4. Bomze, I.M., Frommlet, F., Locatelli, M.: Copositivity cuts for improving SDP bounds on the clique number. Math. Program. **124**, 13–32 (2010)
5. Burer, S., Vandenbussche, D.: Solving lift-and-project relaxations of binary integer programs. SIAM J. Optim. **16**, 726–750 (2006)
6. DIMACS repository. ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/clique
7. Chvátal, V.: Edmonds polytopes and a hierarchy of combinatorial problems. Discr. Math. **4**, 305–337 (1973)
8. Corrêa, R.C., Delle Donne, D., Koch, I., Marenco, J.: General cut-generating procedures for the stable set polytope (2015). arXiv:1512.08757v1
9. Corrêa, R.C., Delle Donne, D., Koch, I., Marenco, J.: A strengthened general cut-generating procedure for the stable set polytope. Elec. Notes Discr. Math. **50**, 261–266 (2015)
10. Coniglio, S., Gualandi, S.: On the exact separation of rank inequalities for the maximum stable set problem. Optimization (2014). http://www.optimization-online.org/DB_HTML/2014/08/4514.html
11. Dukanovic, I., Rendl, F.: Semidefinite programming relaxations for graph coloring and maximal clique problems. Math. Program. **109**, 345–365 (2007)
12. Fischetti, M., Lodi, A.: Optimizing over the first Chvátal closure. Math. Program. **110**, 3–20 (2007)
13. Giandomenico, M., Rossi, F., Smriglio, S.: Strong lift-and-project cutting planes for the stable set problem. Math. Program. **141**, 165–192 (2013)
14. Giandomenico, M., Letchford, A., Rossi, F., Smriglio, S.: An application of the Lovász-Schrijver $M(K, K)$ operator to the stable set problem. Math. Program. **120**, 381–401 (2009)
15. Giandomenico, M., Letchford, A.N., Rossi, F., Smriglio, S.: Approximating the Lovász theta function with the subgradient method. Elec. Notes Discr. Math. **41**, 157–164 (2013)
16. Giandomenico, M., Letchford, A.N., Rossi, F., Smriglio, S.: Ellipsoidal relaxations of the stable set problem: theory and algorithms. SIAM J. Optim. **25**(3), 1944–1963 (2015)
17. Gomory, R.E.: Outline of an algorithm for integer solutions to linear programs. Bull. Amer. Math. Soc. **64**, 275–278 (1958)
18. Gomory, R.E.: An algorithm for integer solutions to linear programs. In: Graves, R.L., Wolfe, P. (eds.) Recent Advances in Mathematical Programming, pp. 269–302. McGraw-Hill, New York (1963)
19. Grötschel, M., Lovász, L., Schrijver, A.J.: Geometric Algorithms in Combinatorial Optimization. Wiley, New York (1988)
20. Gruber, G., Rendl, F.: Computational experience with stable set relaxations. SIAM J. Optim. **13**, 1014–1028 (2003)

21. Johnson, D.S., Trick, M.A. (eds.): Cliques, Coloring, Satisfiability: Observation of Strains: The 2nd DIMACS Implementation Challenge. American Mathematical Society, Providence (2011)
22. Håstad, J.: Clique is hard to approximate within $n^{1-\epsilon}$. Acta Math. **182**, 105–142 (1999)
23. Holm, E., Torres, L.M., Wagler, A.K.: On the Chvátal rank of linear relaxations of the stable set polytope. Int. Trans. Oper. Res. **17**, 827–849 (2010)
24. Locatelli, M.: Improving upper bounds for the clique number by non-valid inequalities. Math. Prog. **150**, 511–525 (2015)
25. Lovász, L.: On the Shannon capacity of a graph. IEEE Trans. Inform. Theor. **25**, 1–7 (1979)
26. Lovász, L., Schrijver, A.J.: Cones of matrices and set-functions and 0–1 optimization. SIAM J. Optim. **1**, 166–190 (1991)
27. Nemhauser, G.L., Sigismondi, G.: A strong cutting plane/branch-and-bound algorithm for node packing. J. Oper. Res. Soc. **43**, 443–457 (1992)
28. Padberg, M.W.: On the facial structure of set packing polyhedra. Math. Program. **5**, 199–215 (1973)
29. Rebennack, S., Oswald, M., Theis, D.O., Seitz, H., Reinelt, G., Pardalos, P.M.: A branch and cut solver for the maximum stable set problem. J. Comb. Opt. **21**, 434–457 (2011)
30. Rossi, F., Smriglio, S.: A branch-and-cut algorithm for the maximum cardinality stable set problem. Oper. Res. Lett. **28**, 63–74 (2001)
31. Juhász, F.: The asymptotic behaviour of lovász' θ function for random graphs. Combinatorica. **2**(2), 153–155 (1982)