Partial Lasserre relaxation for sparse Max-Cut

Juan S. Campos*, Ruth Misener† and Panos Parpas†

October 12, 2020

Abstract A common approach to solve or find bounds of polynomial optimization problems like Max-Cut is to use the first level of the Lasserre hierarchy. Higher levels of the Lasserre hierarchy provide tighter bounds, but solving these relaxations is usually computationally intractable. We propose to strengthen the first level relaxation for sparse Max-Cut problems using constraints from the second order Lasserre hierarchy. We explore a variety of approaches for adding a subset of the positive semidefinite constraints of the second order sparse relaxation obtained by using the maximum cliques of the graph's chordal extension. We apply this idea to sparse graphs of different sizes and densities, and provide evidence of its strengths and limitations when compared to the state-of-the-art Max-Cut solver BiqCrunch and the alternative sparse relaxation CS-TSSOS.

Keywords Global Optimization, Max-Cut, Semidefinite Programming, Polynomial Optimization.

1 Introduction

The Lasserre hierarchy [22] is a powerful relaxation for polynomial optimization problems (POPs). Its power comes from the fact that, in practice, the optimal value of the hierarchy converges faster than theoretical results indicate. In practice, the first and second levels often provide a good solution of the original POP. Along with the efficiency of interior point methods to solve semidefinite programming problems, the Lasserre hierarchy can, in theory, solve POPs in polynomial time. However, not even the second hierarchy level can be solved by interior point methods for medium or large problems. This paper explores the idea of partially strengthening the first level relaxation with a subset of the second level relaxations constraints for sparse instances of the classical Max-Cut problem. Although our proposed technique can be applied to any sparse POP, we consider the Max-Cut problem [17] as its semidefinite relaxation has been studied extensively and therefore gives a good point of comparison for the proposed approach.

*Corresponding author. Department of Computing, Imperial College London, London, SW7 2AZ, UK (jsc12@ic.ac.uk). This work was funded by EPSRC grants EP/P016871/1 and EP/M028240/1.

†Department of Computing, Imperial College London, London, SW7 2AZ, UK (r.misener@ic.ac.uk, panos.parpas@ic.ac.uk)

Given a graph $G = (V, E)$ with nodes $V = \{v_1, \ldots, v_n\}$, a set of edges $E = \{(i, j) : 1 \leq i, j \leq n, \text{ if } i \text{ is connected to } j\}$ and a symmetric matrix $W \in S^n$ with value $w_{i,j} \neq 0$ in position (i, j) if $(i, j) \in E$ and 0 otherwise, the Max-Cut problem can be written as the integer program:

$$\begin{aligned} f^\star = \max_{x \in \mathbb{R}^n} & \sum_{(i,j) \in E} w_{i,j} \left(\frac{1 - x_i x_j}{2} \right) \\ \text{s.t. } & x_i \in \{-1, 1\}, \ i = 1, 2, \ldots, n. \end{aligned} \tag{1}$$

Approaches for rigorously computing upper bounds for the Max-Cut problem include linear, semidefinite programming (SDP), convex quadratic and second order cone relaxations. These relaxations may be strengthened with cuts, e.g. triangle or cycle inequalities, or a branch-and-bound algorithm [5, 4, 6, 12, 18]. Codes like Biq Mac [31] and BiqCrunch [19, 21], which implement the SDP relaxation together with triangle inequalities, successfully solve difficult Max-Cut instances. The SDP relaxation corresponds to the first Lasserre hierarchy level [22]. Alternative Lasserre hierarchy versions are relevant for sparse graphs [24, 34]. These alternatives only add a subset of the dense relaxation and thereby reduce the SDP size considerably at every hierarchy level.

In our computational experience, the sparse SDP relaxation of reasonably-sized Max-Cut instances is still too big to be solved at the second hierarchy level or higher. So, our numerical experiments explore different heuristics to add second order information to the standard sparse SDP relaxation of Waki et al. [34]. We show the advantages and limitations of this partial relaxation by comparing with (i) state-of-the-art Max-Cut solver BiqCrunch and (ii) CS-TSSOS [38], a recently-developed sparse POP hierarchy. The results indicate that, for sufficiently sparse problems, there is rich information to be added to strengthen the first order relaxation without using the entire second hierarchy level. The partial Lasserre relaxation is particularly useful if the maximal cliques of a chordal extension of the graph's correlative sparsity matrix are small.

Previous work considered block-diagonal relaxations between the first and second order relaxations [15, 16]. Specific to Max-Cut, Wiegele [40] proposes a submatrix of the second order relaxation constraint to solve Max-Cut problems for dense graphs. Similarly, works like Adams et al. [1] and Ghaddar et al. [13] construct hierarchies by strenghtening the first level of the Lasserre hierarchy. But our paper is, to the best of our knowledge, the first to computationally study sparse versions of the Lasserre hierarchy to form intermediate relaxations between levels. Our work also resembles prior work approximating semidefinite relaxations with linear cutting planes [3, 30, 32, 33] or nonlinear cutting surfaces [11]. These works developing cutting surfaces typically approximate the first order of the Lasserre hierarchy, while we relax the second order [3, 30, 32, 33]. Like Baltean-Lugojan et al. [3], we select submatrices of a semidefinite relaxation. Differently to Baltean-Lugojan et al. [3], we use clique patterns in the Max-Cut graph to select submatrices (rather than data-driven methods).

It is important to note that there are efficient linear methods to solve sparse Max-Cut problems. For example, the linear branch-and-cut algorithm of Liers et al. [28] is faster than Biq Mac when solving the Max-Cut for sparse toroidal grid graphs [31]. This paper uses Max-Cut because it is a well-studied problem, but the ideas explored here apply to any sparse POP.

The rest of the paper is structured as follows. Section 2 provides the notation. Section 3 introduces the dense and sparse hierarchies. Sections 4 and 5 contain our numerical experiments

and the conclusions, respectively.

2 Notation

The monomial $x_1^{\alpha_1}x_2^{\alpha_2}\dots x_n^{\alpha_n}$ will be denoted by $\mathbf{x}^{\boldsymbol{\alpha}}$, where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \dots, \alpha_n] \in \mathbb{N}^n$. If $\phi \subset \{1, 2, \dots, n\}$ and d a positive integer, then $\mathbb{A}_d^\phi = \{\boldsymbol{\alpha} \in \mathbb{N}^n : \alpha_i = 0 \text{ if } i \notin \phi, \sum_i \alpha_i \leq d\}$, and $u_d(x, \phi)$ is the vector containing all the monomials $\mathbf{x}^{\boldsymbol{\alpha}}$ such that $\boldsymbol{\alpha} \in \mathbb{A}_d^\phi$, e.g. if $\phi = \{2, 4\}$, $n = 4$ and $d = 2$, $\mathbb{A}_d^\phi = \{[0,0,0,0], [0,1,0,0], [0,0,0,1], [0,2,0,0], [0,1,0,1], [0,0,0,2]\}$, $u_d(x, \phi) = [1, x_2, x_4, x_2^2, x_2x_4, x_4^2]^\top$.

If $A \in \mathbb{R}^{m\times n}$ is a matrix, then the element in position (i, j) will be denoted by $A_{i,j}$ (if $m = 1$ or $n = 1$, the i^{th} element of the vector will be denoted by A_i). Likewise, if $A, B \in \mathbb{R}^{m\times n}$, we will use the Frobenius inner product $\langle A, B\rangle = \sum_{1\leq i\leq m}\sum_{1\leq j\leq n} A_{i,j}B_{i,j}$ and its induced norm $\|A\|^2 = \langle A, A\rangle$. $\text{Diag}(x_1, x_2, \dots, x_n)$ is the function returning a diagonal matrix of dimensions $n \times n$ with x_i in the entry (i, i) for $i = 1, 2, \dots, n$. The set of symmetric matrices will be denoted by $\mathcal{S}$, and for any matrix $X \in \mathcal{S}$, $X \succeq 0$ ($\succ 0$) means that X is positive semidefinite (resp., definite). $e \in \mathbb{R}^n$ will denote the vector of ones, and if $i \in \mathbb{N}$, $e_i \in \mathbb{R}^n$ is a vector with one in position ith and zeros everywhere else. Finally, the cardinality of any set ϕ will be denoted by $|\phi|$, the set $\{1, 2, \dots, n\}$ will be written as $[n]$, and $\lceil x\rceil$ is the smallest integer such that $\lceil x\rceil \geq x$ for any $x \in \mathbb{R}$.

3 Lasserre dense, sparse and partial relaxations

The Lasserre hierarchy can be obtained by lifting the monomials of the POP. Replace the objective of Problem (1) by $\frac{1}{4}\left\langle L, [x_1, x_2, \dots, x_n]^\top [x_1, x_2, \dots, x_n]\right\rangle$, where L is the Laplacian matrix $L = \text{Diag}(We) - W$; and the integer constraint $x_i \in \{-1, 1\}$ with the equivalent equation $x_i^2 - 1 = 0$. Problem (1) can be written:

$$\begin{aligned} \max_{x\in\mathbb{R}^n}\ & \frac{1}{4}\left\langle L, [x_1, x_2, \dots, x_n]^\top [x_1, x_2, \dots, x_n]\right\rangle \\ \text{s.t. }\ & x_i^2 - 1 = 0,\ i = 1, 2, \dots, n. \end{aligned} \tag{2}$$

Using the fact that $zz^\top \succeq 0$ for any vector z, we can add the redundant constraints $M_d^{[n]}(x) = u_d(x, [n])u_d(x, [n])^\top \succeq 0$ to obtain:

$$\begin{aligned} \max_{x\in\mathbb{R}^n}\ & \frac{1}{4}\left\langle L, [x_1, x_2, \dots, x_n]^\top [x_1, x_2, \dots, x_n]\right\rangle \\ \text{s.t. }\ & x_i^2 - 1 = 0,\ i = 1, 2, \dots, n, \\ & M_d^{[n]}(x) \succeq 0, \end{aligned} \tag{3}$$

where $d \geq 1$ is a positive integer. Let $\widehat{M}_d^{[n]}(x)$ be the matrix obtained after replacing all the occurrences of the monomial x_i^2 by 1 ($i \in [n]$) in the matrix $M_d^{[n]}(x)$, and then deleting the rth column and row if there is $p < r$ such that the pth column is identical to the rth column.

Lifting all the variables by replacing the monomial $\mathbf{x}^{\boldsymbol{\alpha}}$ by the real variable $y_{\boldsymbol{\alpha}}$, and deleting the integrality constraints we obtain the following SDP,

$$\begin{aligned} Q_d = \max_y \ & \frac{1}{4}\left\langle \hat{L}, \widehat{M}_1^{[n]}(y)\right\rangle \\ \text{s.t. } & \widehat{M}_d^{[n]}(y) \succeq 0, \end{aligned} \tag{4}$$

where y is a real vector indexed by the set $\mathbb{A}_{2d}^{[n]}$, and $\hat{L} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & L \end{bmatrix}$. Matrix $\widehat{M}_d^{[n]}(y)$ is the moment matrix of order d. Lasserre [23] proved that the optimal relaxation value converges to the optimal value of the original POP, and that the convergence is finite: if $d \geq n$ then $Q_d = p^\star$. Practically, the optimal value $p^\star$ is found usually for d considerably smaller than n, e.g. small Max-Cut instances have been solved using only the second relaxation [10, 23].

Although, in theory, the SDP relaxation of Max-Cut can be solved using interior point methods, the relaxation size grows exponentially as d increases and using interior point methods is no longer possible. Hence, the relaxation is only useful for small graphs, or for small values of d. Our computational experience on modern desktops is that, for graphs larger than 30 nodes, typically only the first order relaxation can be solved.

For sparse graphs, Waki et al. [34] developed a sparse version of the Lasserre hierarchy that reduces the size of the dense SDP. Using a similar approach as the one for the dense relaxation, we obtain the relaxation:

$$\begin{aligned} Q_d^s = \max_y \ & \frac{1}{4}\sum_{k=1}^{m}\left\langle \hat{L}_k, \widehat{M}_1^{\phi_k}(y)\right\rangle \\ \text{s.t. } & \widehat{M}_d^{\phi_k}(y) \succeq 0,\ k = 1, 2, \ldots, m, \end{aligned} \tag{5}$$

where y is a vector indexed by the set $\cup_{k=1}^m \mathbb{A}_{2d}^{\phi_k}$, $\{\phi_k\}_{k=1}^m$ correspond to the maximal cliques of a chordal extension of the graph $G = (V, E)$, and the matrices $\hat{L}_k$ are such that $\sum_{k=1}^m \left\langle \hat{L}_k, \widehat{M}_1^{\phi_k}(y)\right\rangle = \left\langle \hat{L}, \widehat{M}_1^{[n]}(y)\right\rangle$. For more information about chordal extensions and algorithms to find maximal cliques, see [7, 8, 14]. Lasserre [24] proves that if the sets $\{\phi_k\}_{k=1}^m$ satisfy the running intersection property, which is the case for the maximal cliques used in Waki et al. [34], and some redundant constraints are added to SDP (5), the optimal relaxation value converges to the optimal value of the original polynomial optimization problem as the value of d tends to infinity.

For sparse graphs, i.e. graphs with small $|E|$, Relaxation (5) reduces considerably the size of Problem (4). However, even for sparse graphs, the sparse relaxation of Max-Cut is too large and can not be solved using interior point methods for $d > 1$. For example, we generated graphs with 300 nodes and tried unsuccessfully to solve the Sparse Relaxation (5) for $d = 2$ for many graphs even with a 2% sparsity, i.e. $2|E|/(n^2) \approx 0.02$ (we used Mosek version 8.1 in an Intel Core i7-6700 CPU @ 3.40 Gigahertz Ubuntu 16.04 workstation with 16 gigabytes of RAM).

3.1 A partial and partial augmented second order sparse relaxation

Not all the moment matrices of the sparse second order relaxation can be included, so we propose including a subset. "Partial" Second Order Relaxation (6) relaxes maximal cliques with more than (fewer than or equal to) r nodes using the first (second) level of the Lasserre hierarchy.

$$
\begin{aligned}
Q_r^P = \max_y \ & \frac{1}{4}\sum_{k=1}^{m}\left\langle \hat{L}_k, \widehat{M}_1^{\phi_k}(y)\right\rangle \\
& \widehat{M}_1^{\phi_k}(y) \succeq 0,\ k \notin \Gamma_r, \\
& \widehat{M}_2^{\phi_k}(y) \succeq 0,\ k \in \Gamma_r,
\end{aligned}
\tag{6}
$$

where $\Gamma_r = \{k \in [m] : |\phi_k| \leq r\}$ and the real vector y is indexed by the set $\{\cup_{k\in\Gamma_r}\mathbb{A}_4^{\phi_k}\} \cup \{\cup_{k\notin\Gamma_r}\mathbb{A}_2^{\phi_k}\}$; note that this set is contained in the set $\cup_{k=1}^m \mathbb{A}_4^{\phi_k}$ corresponding to the space of the second order sparse relaxation. Partial Relaxation (6) reduces the number of constraints and variables by including only a subset of the second order semidefinite constraints of the second level of the Lasserre hierarchy.

We consider also an "augmented" version of the previous problem, by also including second order constraints for subsets of the maximal cliques with sizes larger than r.

$$
\begin{aligned}
Q_{r,p,H}^a = \max_y \ & \frac{1}{4}\sum_{k=1}^{m}\left\langle \hat{L}_k, \widehat{M}_1^{\phi_k}(y)\right\rangle \\
& \widehat{M}_1^{\phi_k}(y) \succeq 0,\ k \notin \Gamma_r, \\
& \widehat{M}_2^{\phi_k}(y) \succeq 0,\ k \in \Gamma_r, \\
& \widehat{M}_2^{\phi_{k,i}}(y) \succeq 0,\ \phi_{k,i} \subset \phi_k,\ |\phi_{k,i}| = r,\ i = 1,\ldots,p, k \notin \Gamma_r,
\end{aligned}
\tag{7}
$$

where p denotes the number of second order constraints added for each maximal clique ϕ_k such that $|\phi_k| > r$, while H denotes the heuristic that selects the sets $\phi_{k,i}$. We construct these subsets $\phi_{k,i}$ sequentially, starting by selecting the smallest maximal clique (if there's more than 1 smallest maximal clique, then we select randomly), and making sure that we are not adding repeated subsets. To achieve an equivalent relaxation, i.e. $Q_{r,p,H}^a \leq Q_r^P$, Augmented Relaxation (7) reduces the size of the maximal cliques r with respect to Partial Relaxation (6), i.e. the values of r are different. For fixed r, Augmented Partial Relaxation (7) increases the number constraints over Partial Relaxation (6) by including the semidefinite constraints corresponding to the second order moment matrices of the subsets $\phi_{k,i}$.

We use five heuristics to select the sets $\{\phi_{k,i}\}$. If the total number of subsets of size r of ϕ_k is q_k and $I = \{i_1, i_2, \ldots, i_p, \ldots, i_{q_k}\}$ is a set of sub-indices, then we select the subsets $\{\phi_{k,i_1}, \phi_{k,i_2}, \ldots, \phi_{k,i_p}\}$ where:

(H1) I is selected randomly and uniformly form the set $[q_k]$.

(H2) I is such that $\|L_{\phi_{k,i_1}}\| \geq \|L_{\phi_{k,i_2}}\| \geq \cdots \geq \|L_{\phi_{k,i_{q_k}}}\|$, where L_ϕ is the sub-matrix created by deleting columns and rows corresponding to indices not contained in ϕ from the Laplacian matrix. H2 selects variable subsets with large absolute value weights in the graph.

(H3) I is such that $|\Omega_{k,i_1}| \geq |\Omega_{k,i_2}| \geq \cdots \geq |\Omega_{k,i_{q_k}}|$, where for every set $\phi_{k,i} \subset \phi_k$ of size r, $\Omega_{k,i} = \{l : \phi_{k,i} \subseteq \phi_l, l \in [m], l \neq k\}$. H3 selects subsets contained in many maximal cliques.

(H4) I is such that $|\Omega_{k,i_1}| \leq |\Omega_{k,i_2}| \leq \cdots \leq |\Omega_{k,i_{q_k}}|$. H4 selects subsets contained in few maximal cliques.

(H5) I combines H2 and H4: we select subsets that are not repeated in other maximal cliques and contain variables with large weights in absolute value in the graph. Specifically, let $\{i_1, i_2, \ldots, i_z\}$ be the indices of the subsets $\{\phi_{k,i_j}\}$ such that $\left|\Omega_{k,i_j}\right| = 0$ and $\|L_{\phi_{k,i_j}}\| \geq \|L_{\phi_{k,i_{j+1}}}\|$ for $1 \leq j \leq z$. Then: if $p \leq z$ we select the subsets $\{\phi_{k,i_1}, \ldots, \phi_{k,i_p}\}$, if $z < p$ we select the subsets $\{\phi_{k,i_1}, \ldots, \phi_{k,i_z}\}$, and if $z = 0$ we apply the heuristic H2.

The next section explores the power of Relaxations (6) and (7) for sparse graphs.

4 Numerical experiments

This section solves relaxations of the Max-Cut problems using Partial Relaxation (6) and Augmented Partial Relaxation (7). All the experiments were run in an Intel Core i7-6700 CPU @ 3.40 GHz Ubuntu 18.04 workstation with 16 GB RAM. MATLAB version 2018a generated the set of maximal cliques using the code contained in the package SparsePOP version 3.01 [35], an implementation of the Waki et al. [34] sparse relaxation. We created and solved SDPs (6) and (7) with C++ and Fusion-API Mosek version 8.1. All the times are elapsed real times. We limit the time of each interior point run to 5 hours.

4.1 Randomly generated problems

The first set of problems are created randomly using the objective sparsity structure obtained by SparsePOP's `randomUnconst.m`. This function takes as arguments the number of polynomial variables (the graph size n), a lower and upper integer bound (l and u, respectively), and a maximal degree (2). With these parameters, `randomUnconst.m` first constructs randomly a set of maximal cliques $\{\omega_k\}_{k=1}^m$ such that $l \leq |\omega_k| \leq u$ for all k, and then generates a quadratic objective function (see Section 6.1 in Waki et al. [34]). For a given set of parameters (n, l, u), and by setting the maximal degree equal to 2, we construct a graph by assigning a weight $w_{ij} \neq 0$ to the edge $\{i, j\}$ if the monomial $x_i x_j$ has a non-zero coefficient in the function $f(x)$. We select the weight w_{ij} randomly and uniformly from a discrete set $\mathcal{W}$. The graph sparsity depends on parameter u. Note that we only use the sparsity structure of the function $f(x)$ to construct the graph, i.e. the specific function values are irrelevant for constructing the graph, what matters is which coefficients are non-zero. Also note that the resulting graph may not be chordal, and therefore the size of the maximal cliques $\{\phi_k\}_k$ of the chordal extension of the graph (which are the ones used in the actual SDP relaxation) are not necessarily bounded by the parameters l and u.

Taking $n = 300, 500$, $l = 2$ and $u = 4, 6, 8, 10$, we generated 3 different graphs for every combination of the previous parameters by setting $\mathcal{W}$ equal to $\{-1, 1\}$, $\{1, 2, \ldots, 10\}$ and $\{-10, -9, \ldots, -1, 1, 2, \ldots, 10\}$. In this set-up, the sparsity structure of the 3 graphs is identical but the non-zero weights are different. We repeat this procedure 10 times to obtain a total of 240 graphs. Table 1 summarizes the mean of the sparsity of the graphs for the different parameters and presents the mean of different size measures of the maximal cliques found after using the SparsePOP heuristic. Note that, while the parameter u controls the graph sparsity by limiting the maximal clique size, this maximal clique size is not bounded by the parameter u.

n	l	u	Sparsity	$\max\{\lvert\phi_k\rvert\}_k$	$\text{median}\{\lvert\phi_k\rvert\}_k$	$\text{mean}\{\lvert\phi_k\rvert\}_k$	Total
300	**2**	**4**	1.3%	16.2	3.2	3.7	30
	2	**6**	2.4%	38.8	5.1	6.4	30
	2	**8**	3.6%	59.3	6.8	9.3	30
	2	**10**	5.1%	76.8	8.9	12.3	30
500	**2**	**4**	0.8%	25.1	3.2	3.8	30
	2	**6**	1.5%	63.5	5.0	7.0	30
	2	**8**	2.3%	99.3	7.0	10.3	30
	2	**10**	3.1%	124.8	9.0	13.8	30

Table 1: Summary statistics of the random generated graphs. Columns $\max\{|\phi_k|\}_k$, $\text{median}\{|\phi_k|\}_k$, and $\text{mean}\{|\phi_k|\}_k$ are averages over the 30 instances of the maximum, median and mean of the cardinality of the maximal cliques $\{\phi_k\}_k$.

4.1.1 Strength of the partial relaxation

We solved Partial Relaxation (6) for every graph using $r = 3, 4, \ldots, 20$. Notice: (i) given the computational limitations it is not always possible to solve the partial relaxation for all r and (ii) a graph might not contain maximal cliques of a certain size and therefore there is no need to solve the problem for that size, e.g. if a graph does not have maximal cliques of size 7 the set Γ_6 is equal to Γ_7 and the relaxation does not change from $r = 6$ to $r = 7$.

We denote $\hat{r} = \text{argmin}_r\{Q_r^P\}_{r\in\{3,4,\ldots,20\}}$ where Q_r^P is the solution of Partial Relaxation (6) for each graph. In theory, the best solution is obtained for $r = 20$, but (i) we may not be computationally able to solve SDP (6) for such a large r or (ii) we may find the solution for some $r < 20$ (in this case we set $\hat{r}$ as the minimum integer r such that Q_r^P solves the Max-Cut).

Let $\text{gap}_r = \frac{Q_r^P}{f^\star} - 1$, where $f^\star$ is the optimal solution of Max-Cut Problem (1). BiqCrunch calculated the optimal solution $f^\star$ for each graph. We limited the total BiqCrunch time to one hour and used default parameters (`BiqCrunch\problems\max-cut\biq_crunch.param`). We set $f^\star$ equal to the best feasible solution found by BiqCrunch (we did not obtain a certificate of optimality for all problems). Table 2 groups the results by dimension and the parameter u that controls the graph sparsity. The third column contains the mean of the gaps for $r = 0$ and $r = \hat{r}$, i.e. the gap for the first order relaxation and the best solution found using the partial relaxation respectively ($\mathbf{gap}_0$ is missing information from 7 problems that Mosek could not solve), while the fourth column shows the mean of the differences between the size of the largest maximal clique and $\hat{r}$. The fifth column presents the total number of problems with a gap smaller than 1×10^{-7} (30 problems in total). Additionally, given that all the weights of the graphs are integer numbers, the last column presents how many of the partial relaxation solutions are at most a unit away from the Max-Cut solutions, i.e. $Q_{\hat{r}}^P - f^\star \leq 1$. The partial relaxation reaches a gap smaller than 1% in average for the sparser problems corresponding to $u = 4, 6$ (these problems have in average maximal cliques with size smaller than 7, see Table 1), and solves most of the problems for $u = 4$. Also notice that these results are achieved without using all the possible maximal cliques, with an average difference between $\hat{r}$ and the largest maximal cliques ranging from 4 to 40. When the graph density increases, the partial relaxation starts to lose its effectiveness. Table 3 additionally groups results by the weights generating the

graph and shows that the average gap for the graphs with weights in the set $\{1, 2, \ldots, 10\}$ are smaller than for the other two types of weights.

n	u	$(\mathbf{gap}_0, \mathbf{gap}_{\hat{r}})$	$\max\{\lvert\phi_k\rvert\}_k - \hat{r}$	$\mathbf{gap}_{\hat{r}} \leq 10^{-7}$	$Q^P_{\hat{r}} - f^\star \leq 1$
300	**4**	(4.84%, 0.00%)	4.3	29	30
	6	(7.55%, 0.09%)	20.4	10	22
	8	(9.79%, 1.07%)	39.6	0	1
	10	(12.16%, 2.51%)	56.8	0	0
500	**4**	(4.46%, 0.00%)	9.4	24	30
	6	(8.59%, 0.80%)	43.6	0	1
	8	(10.96%, 2.67%)	79.3	0	0
	10	(13.86%, 5.04%)	104.8	0	0

Table 2: Optimality results using the best solution found ($f^\star$) with the partial relaxation ($Q^P_{\hat{r}}$) applied to the random instances grouped by dimension and u. Each row represents 30 instances.

		$\mathbf{gap}_{\hat{r}}$		
n	u	$w_{ij} \in \{-1, 1\}$	$w_{ij} \in [10]$	$w_{ij} \in \{-10, \ldots, 10\}$
300	**4**	0.00%	0.00%	0.00%
	6	0.14%	0.05%	0.09%
	8	1.80%	0.40%	1.02%
	10	3.52%	0.75%	3.25%
500	**4**	0.00%	0.00%	0.00%
	6	1.41%	0.29%	0.71%
	8	3.84%	1.08%	3.10%
	10	7.46%	1.52%	6.12%

Table 3: Gaps using the best solution found with the partial relaxation ($Q^P_{\hat{r}}$) applied to the random instances grouped by dimension, u and type of weight of the graph.

4.1.2 Comparison with triangle inequalities

When using SDP relaxations for the Max-Cut problem, one of the most common and efficient approaches combines the triangle inequalities with the first level of the dense Lasserre hierarchy. State-of-the-art codes Biq Mac [31, 40] and BiqCrunch [19] (approximately) solve this relaxation at every branch-and-bound node. Given $r < s < t \leq n$, the triangle inequalities are:

$$\begin{aligned} y_{e_r+e_s} + y_{e_r+e_t} + y_{e_s+e_t} &\geq -1, \\ y_{e_r+e_s} - y_{e_r+e_t} - y_{e_s+e_t} &\geq -1, \\ -y_{e_r+e_s} + y_{e_r+e_t} - y_{e_s+e_t} &\geq -1, \\ -y_{e_r+e_s} - y_{e_r+e_t} + y_{e_s+e_t} &\geq -1. \end{aligned} \tag{8}$$

The *metric polytope* (MET) is the space defined by the vectors satisfying the set of Inequalities (8) for all $r < s < t \leq n$. The vertices of this convex set contain all the possible cuts of a

graph with n nodes, and is therefore a relaxation for the Max-Cut problem by itself. Laurent [25] further discusses the MET vertices. Laurent [26] also characterizes the vertices of the dense Lasserre relaxation of order 1 and proves that they correspond exactly to the cuts of the graph. Furthermore, the second order dense relaxation is contained in the metric polytope (see Anjos and Wolkowicz [2][1]), which means that the second level of the Lasserre dense relaxation is contained in the intersection of the metric polytope and the first level of the hierarchy (we will refer to this intersection as MET_+^{1st}).

For every solution $Q_{\hat{r}}^P$, we calculated the time that BiqCrunch takes to find a solution as good as $Q_{\hat{r}}^P$ (software download: `https://biqcrunch.lipn.univ-paris13.fr/`). For each instance, we ran four implementations of BiqCrunch:

1. Standard branch-and-bound using defaults: `BiqCrunch\problems\max-cut\biq_crunch.param`,
2. Root node solve only with default parameter change: `minAlpha` $= 1 \times 10^{-12}$,
3. Root node solve only with default parameter change: `minTol` $= 1 \times 10^{-12}$, or
4. Root node solve only with default parameter change: `minAlpha` $= 1 \times 10^{-12}$ and `minTol` $= 1 \times 10^{-12}$.

For each Max-Cut instance, our comparisons are with respect to the fastest time of these four implementations that reached the solution $Q_{\hat{r}}^P$. These changes improve the accuracy of the root node solution to prevent an early stop of the algorithm (see Section 4.1 of Krislock et al. [19] and Section 2.1 of the BiqCrunch manual [20] for more information). Finally, we set a limit of 5 hours to BiqCrunch. Given that the weights of all the graphs are integers, the solution found using BiqCrunch is not strictly better than $Q_{\hat{r}}^P$, in particular if $BC^\star$ is the BiqCrunch solution then $BC^\star < \lceil Q_{\hat{r}}^P \rceil$.

Table 4 compares the times of the partial relaxation (t) and BiqCrunch (t_{BC}), grouping the instances by the type of weight in the graph, the parameter u (see Section 4.1), and the size of the graph (10 instances in total for a fixed type of weight, value of u and size of the graph). Table 4 shows the average of the ratio of the times (t_{BC}/t), number of problems where the partial relaxation was faster than BiqCrunch out of the 10 instances ($t_{BC} > t$), and the average time taken by the partial relaxation (t). The partial relaxation is faster than BiqCrunch for the problems with the smallest maximal cliques ($u = 4$) independent of the size (one particular instance of size 500 could not be solved by BiqCrunch for any of the 4 implementations), but loses its efficiency as the size of the maximal cliques increases. For instances generated using $u = 8, 10$, BiqCrunch is always faster. As the size of the graphs increases from 300 to 500, the number of problems solved faster by the partial relaxation increases for some cases, and for those where it does not (remaining at zero) the average ratio of the times has a small increase. With respect to the type of weight, BiqCrunch tends to perform better for weights in $\{-10, \ldots, 10\}$.

[1] Anjos and Wolkowicz [2] prove this result for a different hierarchy than that of Lasserre. But the latter is contained in the former and the proof directly applies to the Lasserre case.

		$n = 300$			$n = 500$		
Weight	u	t_{BC}/t	$t_{BC} > t$	t	t_{BC}/t	$t_{BC} > t$	t
$w_{ij} \in \{-1, 1\}$	**4**	31.18	10	2.83	37.93	10	28.25
	6	1.51	3	95.29	1.26	8	146.79
	8	0.34	0	113.73	0.48	0	270.16
	10	0.18	0	171.87	0.23	0	421.23
$w_{ij} \in [10]$	**4**	44.96	10	1.77	29.60	10	33.91
	6	1.16	4	71.92	1.37	9	145.84
	8	0.43	0	112.38	0.53	0	262.28
	10	0.20	0	173.86	0.26	0	409.63
$w_{ij} \in \{-10, \ldots, 10\}$	**4**	44.01	10	2.52	32.32	10	34.43
	6	0.62	1	76.28	1.01	5	147.23
	8	0.34	0	114.78	0.40	0	267.70
	10	0.16	0	173.61	0.22	0	412.88

Table 4: Comparison between the time used by BiqCrunch to reach a solution as good as $Q^P_{\hat{r}}$ for all the random instances.

The feasible space of the partial relaxation depends on the parameter r. If $r = 0$, this space is equivalent to the feasible space of Relaxation (5) with $d = 1$, and MET^{1st}_+ is then a tighter relaxation. Recall that, for graphs of size 3 or 4, the vertices of the metric polytope correspond exactly to the cuts of the graph [25]. Therefore, if $r \leq 4$, the space defined by MET^{1st}_+ is at least as tight or tighter than the feasible space of Partial Relaxation (6). However, it is not difficult to find a point y belonging to MET^{1st}_+ that does not satisfy $\widehat{M}_2^{\phi_k}(y) \succeq 0$ for $|\phi_k| \geq 5$.[2] Given that the feasible space MET^{1st}_+ is not contained in the feasible space of the partial relaxation if $r \geq 5$, these two feasible spaces are then not equal for $r \geq 5$. Although the feasible spaces are not the same, Tables 2 to 4 show that, for sparse graphs, the partial relaxation can provide competitive solutions compared to the standard MET plus first order Lasserre relaxation.

4.1.3 Augmented relaxation

This section concentrates on Augmented Partial Relaxation (7). First we compare heuristics H1 - H5 and then compare the partial and the augmented relaxations. In the implementation of the heuristics H1 to H5 for the augmented formulation, we did not consider all the possible subsets of ϕ_k as this number can be very large. Instead, we select randomly 20 subsets and apply the heuristics to those sets. For example, if $r = 5$ and the maximal clique k contains 10 elements ($|\phi_k| = 10$), the total number of possible subsets of size 5 is $q_k = 252$. Rather than considering all 252 subsets when applying any particular heuristic, we select randomly 20 subsets and assume those are all the possible subsets of size 5 for the maximal clique k and then apply the heuristic. If $p > q_k$, we include all the subsets (without adding repeated subsets).

We solved every Augmented Partial Relaxation (7) using $r = 3, 4, \ldots, 20$, $p = 1, 2, 3$, and

[2]Let $\phi_k = [5]$ and the set $\Omega = \{y : y_{\boldsymbol{\alpha}} = 0 \text{ if } \sum_i \alpha_i = 2 \text{ and } \alpha_i = 0 \text{ for } i \in [5], y_{e_1+e_2} = -0.907, y_{e_1+e_3} = 0.668, y_{e_1+e_4} = -0.591, y_{e_1+e_5} = -0.907, y_{e_2+e_3} = -0.761, y_{e_2+e_4} = 0.498, y_{e_2+e_5} = 0.814, y_{e_3+e_4} = -0.737, y_{e_3+e_5} = -0.761, y_{e_4+e_5} = 0.498\}$. Any $y \in \Omega$ satisfies MET^{1st}_+, but we can numerically verify that the feasibility problem of finding y such that $\widehat{M}_2^{\phi_k}(y) \succeq 0$ and $y \in \Omega$ does not have a solution.

the 5 types of heuristics, i.e. for a fixed r and p we have 5 different solutions. For each value of r and p, we ranked the optimal solutions found by the different heuristics from the best (ranked 1) to worst (rank 5): the best heuristic produces the augmented relaxation with the smallest optimal objective value. If two or more heuristics produce the same solution, we assign the same rank. Table 5 presents the performance of each heuristic (the results do not change drastically between the type of weight used in the graph, or the number of subsets p). For example, the third row and second column indicates that, in 428 out of the 8905 solutions (4.8%), the random heuristic (H1) found the best possible solution out of all the heuristics. Notice that the total number of problems is not the same between the 5 heuristics, this is because we exclude the solution of the augmented relaxations that do not improve the objective value, i.e. if $Q^a_{r,p,H}$ is already the solution of the original Max-Cut problem we do not include $Q^a_{r+j,p,H}$ for $j > 0$. Therefore, heuristics that perform better will have fewer total problems. The results indicate that the best heuristic, with more than 90% of its solutions ranked first or second, consists on selecting the subsets that do not repeat in other maximal cliques and for which the norm of the weights between the variables of the subset is as large as possible (H5). Interestingly, the random heuristic (H1) performed better than trying to select the most repeated subsets (H3).

	H1		**H2**		**H3**		**H4**		**H5**	
Rank	Total	%	Total	%	Total	%	Total	%	Total	%
1	428	4.8%	2215	25.4%	459	4.9%	1761	20.1%	5177	59.8%
2	451	5.1%	3706	42.5%	155	1.7%	1542	17.6%	2805	32.4%
3	1085	12.2%	1878	21.5%	212	2.3%	4956	56.5%	561	6.5%
4	6812	76.5%	906	10.4%	258	2.8%	483	5.5%	95	1.1%
5	129	1.4%	19	0.2%	8224	88.4%	28	0.3%	18	0.2%
Total	8905		8724		9308		8770		8656	

Table 5: Comparison objective value of Augmented Partial Relaxation (7) using heuristics H1-H5

For fixed p and H, let r_a be the minimum value such that the solution $Q^a_{r_a,p,H}$ of the augmented relaxation satisfies $Q^a_{r_a,p,H} \leq Q^P_{\hat{r}}$. We found $Q^a_{r_a,p,H}$ for the 240 random graphs using H5 as heuristic, $p = 1, 2, 3$, and calculated: the time t and t_a needed to obtain $Q^P_{\hat{r}}$ and $Q^a_{r_a,p,H}$ respectively, and $\Delta r = \hat{r} - r_a$. Recall that $\hat{r}$ and r_a will typically have different values, so a large value of Δr indicates that Augmented Partial Relaxation (7) is working particularly well. Table 6 compares the two relaxations grouped by the size of the graph, the upper bound parameter (u) used to construct the random graph, and three different values of p, i.e. for a fixed n, u and p, Table 6 represents 30 problems. For the three values of p, Table 6 shows the mean of Δr, the mean of the ratios of the time (t/t_a), and how many problems out of 30 were solved faster using the augmented relaxation ($t > t_a$). For Table 6, the times t and t_a include: time to create the maximal cliques and the subsets, time to formulate the relaxation in Mosek, and the relaxation solution time used by the interior point method. As expected, increasing the number of included subsets (p) reduces the largest clique size (smaller r_a, larger Δr). Table 6 shows that the augmented relaxation may often be more efficient, but there is not a clear pattern.

Our code for creating the maximal cliques and the subsets (which is written in MATLAB) can be improved considerably as our intention was not at this point to generate a very efficient

code for these heuristics. Table 7 is equivalent to Table 6 except that Table 7 excludes from t and t_a the time spent constructing the maximal cliques and the subsets (the values of Δr do not change and are not included again). In Table 7, a clearer pattern appears: larger values of p and sparser problems (smaller values of u) make the augmented relaxation faster than the partial one (the only exception is for $n = 300$ and $u = 4$). Finally, there were not substantial differences when the results were also grouped by the type of weight of the graph.

$n = 300$	$p = 1$			$p = 2$			$p = 3$		
u	Δr	t/t_a	$t > t_a$	Δr	t/t_a	$t > t_a$	Δr	t/t_a	$t > t_a$
4	3.80	1.22	10	4.87	1.16	8	5.73	1.02	9
6	6.50	5.69	29	8.37	7.41	29	9.10	7.36	29
8	6.37	2.00	29	8.23	2.89	29	9.17	3.35	30
10	5.53	1.20	20	7.33	1.32	18	8.53	1.81	25
$n = 500$	$p = 1$			$p = 2$			$p = 3$		
u	Δr	t/t_a	$t > t_a$	Δr	t/t_a	$t > t_a$	Δr	t/t_a	$t > t_a$
4	6.03	3.77	18	6.83	3.76	17	8.23	4.48	18
6	6.53	2.04	29	8.43	2.88	30	9.37	3.07	30
8	5.83	1.07	19	7.70	1.45	27	8.83	1.49	26
10	5.50	0.74	2	7.33	1.15	23	8.43	0.97	12

Table 6: Comparison between the time used to find the best solution using Partial Relaxation (6) and the time used by the smallest Augmented Partial Relaxation (7) with a solution as good as the partial (using heuristic H5). Times include creating maximal cliques $\{\phi_k\}_k$ and subsets $\{\phi_{k,i}\}_{k,i}$, creating the SDP, and solving the SDP.

$n = 300$	$p = 1$		$p = 2$		$p = 3$	
u	t/t_a	$t > t_a$	t/t_a	$t > t_a$	t/t_a	$t > t_a$
4	3.33	27	3.47	29	3.49	28
6	7.61	30	10.56	30	10.97	30
8	2.13	29	3.24	29	3.86	30
10	1.23	22	1.36	18	1.91	27
$n = 500$	$p = 1$		$p = 2$		$p = 3$	
u	t/t_a	$t > t_a$	t/t_a	$t > t_a$	t/t_a	$t > t_a$
4	8.58	28	7.58	29	10.81	29
6	2.25	29	3.42	30	3.78	30
8	1.11	19	1.52	27	1.58	26
10	0.75	2	1.18	23	1.00	15

Table 7: Comparison between the time used to find the best solution using Partial Relaxation (6) and the time used by the smallest Augmented Partial Relaxation (7) with a solution as good as the partial (using heuristic H5). Times include creating and solving the SDP but do not include creating the maximal cliques $\{\phi_k\}_k$ and subsets $\{\phi_{k,i}\}_{k,i}$.

4.2 Max-Cut instances from applications in statistical physics

Now that we have established the usefulness of the SDP relaxations (6) and (7) for random instances, we apply the partial and augmented relaxations to toroidal grid graphs [27, 28]. These problems are in the Biq Mac library (`http://biqmac.aau.at/biqmaclib.html`). There are 6 different graphs structures and, for each structure, there are 3 different sets of weights, i.e. same set of edges but with different weights. Table 8 summarizes the 6 graph structures and the maximal cliques obtained using the SparsePOP heuristics. Note that $t3$ instances have larger maximal cliques in average (around 13) compared with $t2$ (approximately 8).

Instance	n	**Sparsity**	$\max\{\lvert\phi_k\rvert\}_k$	$\text{median}\{\lvert\phi_k\rvert\}_k$	$\text{mean}\{\lvert\phi_k\rvert\}_k$
t2g10	100	4.0%	23	5	7.2
t2g15	225	1.8%	39	5	8.4
t2g20	400	1.0%	52	5	8.1
t3g5	125	4.8%	53	7	13.5
t3g6	216	2.8%	72	7	12.3
t3g7	343	1.8%	119	7	16.1

Table 8: Summary of the graph structure and maximal cliques for the toroidal grid graphs [27, 28]. The columns $\max\{|\phi_k|\}_k$, $\text{median}\{|\phi_k|\}_k$ and $\text{mean}\{|\phi_k|\}_k$, correspond to the the maximum, median and mean of the cardinality of the maximal cliques $\{\phi_k\}_k$.

We found (if possible) the smallest $r \in \{3, 4, \ldots, 20\}$ such that $\frac{Q^a_{r,p,H}}{f^\star} - 1 \leq 1 \times 10^{-7}$, for $p = 0, 1, 2, 3$ and $H = H5$. Note that $p = 0$ corresponds to Partial Relaxation (6) (heuristic H does not apply in this case), and $p = 1, 2, 3$ to Augmented Partial Relaxation (7). For each p value, we calculated the time BiqCrunch needs to find a solution at least as good as $Q^a_{r_a,p,H}$. We used the same 4 parameter settings used in Section 4.1.2, and additionally we repeated the 4 experiments but now using the default parameters provided by BiqCrunch for Ising problems (`BiqCrunch\problems\max-cut\biq_crunch.param.ising`). Since the graphs only contain integer weights, we again consider a solution $BC^\star$ from BiqCrunch as good as $Q^a_{r,p,H}$ if $BC^\star < \left\lceil Q^a_{r,p,H} \right\rceil$. We selected the best BiqCrunch time from the 8 parameter settings (once more we limit the experiment to a 5 hour limit)[3].

Once more, t and t_a correspond to the time (including creation of maximal cliques and subsets) to solve the partial relaxation ($p = 0$) and the augmented one ($p = 1, 2, 3$), and t_{BC} the time used by BiqCrunch. Table 10 presents the minimum value r, t, t_a, and gap, as well as the time ratios t_{BC}/t and t_{BC}/t_a, for each instance for the different values of p. Partial Relaxation (6) finds the solution for all the $t2$ instances at least 6 times faster than BiqCrunch, except for $t2g20_6666$ where even using $r = 19$ is only enough to reduce the gap to 4^{-4}. For the $t3$ instances, which have larger maximal cliques, the partial relaxation can not find the solution for all the problems and is slower than BiqCrunch. Increasing the value of p and using the heuristic

[3]For the instance $t2g20_6666$ when $Q^a_{r_a,p,H}$ provided an accuracy to the 10^{-7} or lower, BiqCrunch was not able to find a solution with any setting under the time limit. In this case we found that the following setting was able to find a solution: default Ising parameters changing `minAlpha` and `minTol` to 10^{-12}, `nitermax` to 20000, and `maxNiter` to 1000.

H5, Augmented Partial Relaxation (7) improves the gap for all the problems (when $p = 2$ all the gaps are smaller than 1×10^{-7} except one), and for some of the $t3$ problems the augmented relaxation is faster than BiqCrunch. The augmented relaxation is slower than the partial one for many problems. Most of this difference is explained by the time selecting the subsets in the augmented relaxation.

4.3 Comparison with CS-TSSOS

This section compares the partial relaxation with the moment-SOS hierarchy CS-TSSOS [38]. This 2 level hierarchy exploits sparsity by combining the correlative sparsity used in Waki et al. [34] (which is the same one used for constructing the maximal cliques in our partial and augmented relaxations) with term sparsity [36, 37, 39]. Wang et al. [38] use large-scale, sparse randomly generated graphs to show the efficiency of CS-TSSOS as a relaxation of the Max-Cut. For the instances generated, the second order sparse Relaxation (5), i.e. second level of the Waki et al. [34] relaxation, is too large to be solved, while the second level of CS-TSSOS is not only solvable but also provides better bounds than the first order of the sparse Relaxation (5).

We solved the 10 Wang et al. [38] instances using Partial Relaxation (6) for values of $r = 4, 5, \ldots, 10$, and using the CS-TSSOS second level relaxation. Both the instances and the code to generate and solve the CS-TSSOS relaxations (which also uses Mosek to solve the SDP) can be found in `https://wangjie212.github.io/jiewang/code.html` (accessed 25/06/2020). We did not use Augmented Partial Relaxation (7) because generating the subsets of the maximal cliques with any of the heuristics takes considerable time compared to the time of solving the resulting relaxation. These instances are larger than the graphs studied so far (the largest graph has 5005 nodes), have maximal cliques with average size no larger than 8, and the largest maximal clique have 16 elements (see Table 9).

Instance	n	**Sparsity**	$\max\{\lvert\phi_k\rvert\}_k$	$\text{median}\{\lvert\phi_k\rvert\}_k$	$\text{mean}\{\lvert\phi_k\rvert\}_k$
g20	505	1.61%	15	7	7.3
g40	1005	0.68%	15	6	6.7
g60	1505	0.43%	15	6	6.7
g80	2005	0.30%	15	6	6.3
g100	2505	0.23%	16	6	6.2
g120	3005	0.19%	14	5	6.0
g140	3505	0.16%	15	5	5.9
g160	4005	0.13%	15	5	5.9
g180	4505	0.12%	15	5	6.1
g200	5005	0.11%	15	5	5.9

Table 9: Summary of the graph structure and maximal cliques for the instances in used in [38]. The columns $\max\{|\phi_k|\}_k$, $\text{median}\{|\phi_k|\}_k$ and $\text{mean}\{|\phi_k|\}_k$, correspond to the the maximum, median and mean of the cardinality of the maximal cliques $\{\phi_k\}_k$.

Table 11 shows the results. Because our code is specific to construct the partial relaxation for the Max-Cut problem while the CS-TSSOS code is general for any POP, we included the time used by the interior point method to solve the SDP relaxation (Time IPM), besides the

	Partial				Augmented: $p = 1$				Augmented: $p = 2$				Augmented: $p = 3$			
Instance	r	t_{BC}/t	t	**Gap**	r	t_{BC}/t_a	t_a	**Gap**	r	t_{BC}/t_a	t_a	**Gap**	r	t_{BC}/t_a	t_a	**Gap**
t2g10_5555	9	10.58	0.63	**3.9E-09**	5	3.78	1.77	**2.4E-08**	5	3.62	1.84	**1.6E-08**	4	2.51	2.66	**5.3E-08**
t2g10_6666	5	20.61	0.30	**3.5E-08**	5	3.65	1.70	**3.3E-08**	5	3.62	1.71	**7.0E-08**	4	3.14	1.98	**1.3E-08**
t2g10_7777	9	8.35	0.68	**7.1E-08**	6	5.44	1.04	**3.5E-08**	5	3.43	1.65	**2.9E-08**	4	3.00	1.88	**1.7E-08**
t2g15_5555	7	22.45	1.76	**1.5E-08**	6	7.57	5.23	**5.0E-08**	5	8.48	4.67	**3.2E-09**	5	8.13	4.87	**9.7E-10**
t2g15_6666	11	12.88	2.93	**8.7E-08**	6	7.32	5.22	**2.3E-09**	5	7.55	5.06	**2.1E-09**	5	6.65	5.75	**5.1E-11**
t2g15_7777	13	6.04	8.13	**2.9E-08**	8	9.58	5.12	**1.6E-08**	6	9.84	4.99	**3.3E-09**	5	8.15	6.03	**3.5E-08**
t2g20_5555	13	11.00	19.12	**6.6E-08**	7	14.80	14.54	**2.8E-08**	5	14.40	14.94	**2.2E-08**	4	6.49	33.18	**1.2E-08**
t2g20_6666	19	0.66	169.54	3.9E-04	12	10.13	38.41	**7.7E-10**	8	14.97	26.01	**1.4E-08**	7	19.89	19.58	**3.7E-10**
t2g20_7777	9	38.12	5.18	**1.2E-08**	5	14.57	13.55	**1.3E-10**	5	13.49	14.63	**3.5E-10**	4	5.52	35.75	**2.5E-08**
t3g5_5555	19	0.12	83.45	**9.1E-12**	8	1.91	5.33	**4.8E-09**	8	1.82	5.60	**1.7E-09**	8	1.59	6.42	**3.1E-09**
t3g5_6666	19	0.12	82.40	**3.5E-11**	13	0.49	20.37	**7.3E-09**	11	0.71	14.09	**1.9E-08**	8	1.48	6.71	**3.2E-08**
t3g5_7777	19	0.08	132.43	**2.6E-08**	9	1.58	6.69	**7.0E-09**	8	1.82	5.82	**3.9E-10**	8	1.73	6.12	**1.2E-09**
t3g6_5555	7	4.50	9.21	**1.6E-09**	7	3.53	11.76	**1.5E-08**	6	2.20	18.90	**9.5E-08**	6	2.13	19.56	**6.4E-08**
t3g6_6666	19	0.20	218.39	1.4E-03	20	0.62	1778.53	8.8E-07	19	0.29	3833.02	**4.4E-08**	16	1.95	579.46	**2.3E-08**
t3g6_7777	19	0.29	188.46	2.7E-07	13	1.47	38.13	1.0E-07	10	2.75	20.87	**1.5E-08**	10	2.27	25.24	**1.0E-11**
t3g7_5555	19	0.27	364.32	**1.5E-08**	10	0.48	202.09	**1.3E-10**	9	0.53	185.65	**2.1E-08**	8	0.43	227.89	**6.8E-08**
t3g7_6666	19	0.27	301.38	6.6E-04	13	0.52	273.66	**5.7E-11**	10	0.64	221.13	**3.2E-09**	10	0.59	238.25	**5.7E-09**
t3g7_7777	19	0.35	282.06	6.9E-03	19	0.05	3270.75	3.3E-03	15	0.32	413.42	3.4E-03	15	0.24	652.69	2.8E-03

Table 10: Results for the toroidal grid graphs problems. Times from partial (t) and augmented relaxation (t_a) compared to BiqCrunch (t_{BC}). The Gap values that are less than 1×10^{-7} are in bold font.

total time used by each method (Total time), which includes the time to create the relaxation and solve it. In general, Partial Relaxation (6) achieves better solutions faster (both in terms of the total time and the interior point time) than the CS-TSSOS relaxation after adding maximal cliques of size 7 or smaller ($r \leq 7$). In particular, for the largest instances, $r = 4$ was enough to improve the bound found by CS-TSSOS. These solutions can be improved using larger values of r and while still obtaining competitive times compared to CS-TSSOS.

Instance	**Results**	**CS-TSSOS**	**Waki et al.**	**Partial**						
		2nd	**1st**	$r = 4$	$r = 5$	$r = 6$	$r = 7$	$r = 8$	$r = 9$	$r = 10$
g20	Total time	**56.2**	0.6	0.6	0.7	0.9	**1.4**	2.2	3.4	5.1
	Time IPM	**41.5**	0.3	0.3	0.4	0.6	**0.9**	1.6	2.5	4.0
	Opt. value	**537.4**	570.8	565.7	555.7	540.0	**527.6**	517.3	509.2	502.0
g40	Total time	**73.1**	1.2	1.4	1.7	**2.1**	2.8	4.0	6.2	11.3
	Time IPM	**36.9**	0.5	0.6	0.9	**1.1**	1.7	2.5	4.4	8.7
	Opt. value	**992.2**	1032.6	1013.7	987.1	**968.0**	947.7	933.9	922.3	907.5
g60	Total time	**117.4**	2.1	2.4	**2.9**	3.7	5.6	8.2	12.2	17.4
	Time IPM	**47.7**	0.8	0.9	**1.1**	1.6	2.7	4.8	7.8	12.3
	Opt. value	**1387.2**	1439.9	1411.1	**1379.8**	1347.2	1318.9	1295.8	1275.5	1259.2
g80	Total time	**151.5**	3.4	3.9	**4.7**	6.2	8.9	11.6	18.8	26.0
	Time IPM	**40.3**	1.0	1.2	**1.5**	2.2	3.8	6.0	11.9	18.4
	Opt. value	**1838.1**	1899.2	1857.4	**1811.5**	1771.4	1737.2	1710.3	1685.7	1663.8
g100	Total time	**220.5**	4.8	5.7	**6.9**	8.9	13.1	16.5	22.3	32.4
	Time IPM	**56.7**	1.2	1.5	**2.0**	2.7	5.1	7.4	12.4	21.2
	Opt. value	**2328.3**	2398.7	2341.2	**2288.1**	2238.2	2193.5	2161.3	2133.9	2112.1
g120	Total time	**284.2**	7.2	8.2	**10.4**	13.1	16.9	23.7	31.6	42.6
	Time IPM	**45.5**	1.3	1.8	**2.4**	3.2	5.7	10.0	17.7	26.4
	Opt. value	**2655.4**	2731.7	2659.0	**2592.9**	2539.5	2492.1	2452.6	2418.3	2394.1
g140	Total time	**339.4**	10.3	12.0	**14.6**	17.9	23.0	28.2	40.4	55.3
	Time IPM	**47.5**	1.7	2.1	**2.7**	3.9	6.7	11.2	20.2	32.7
	Opt. value	**3027.2**	3115.8	3030.4	**2957.4**	2893.6	2837.4	2794.0	2760.6	2733.1
g160	Total time	**414.7**	15.3	**17.2**	19.8	25.0	30.6	36.7	48.5	63.5
	Time IPM	**56.9**	2.0	**2.3**	2.8	4.4	7.3	12.2	22.2	35.1
	Opt. value	**3589.0**	3670.7	**3572.9**	3497.1	3424.6	3365.8	3321.8	3278.2	3249.0
g180	Total time	**551.3**	20.9	**23.0**	26.4	31.7	38.1	45.7	57.9	76.5
	Time IPM	**60.2**	2.1	**2.6**	3.4	5.1	8.3	13.3	22.8	37.8
	Opt. value	**3953.1**	4054.7	**3950.6**	3864.3	3788.6	3723.6	3676.6	3631.2	3597.7
g200	Total time	**640.6**	29.4	**31.0**	36.1	45.2	47.3	60.1	75.6	96.2
	Time IPM	**54.4**	2.8	**3.0**	4.0	6.3	8.9	16.7	28.5	44.8
	Opt. value	**4472.3**	4584.6	**4465.4**	4363.6	4270.1	4205.1	4145.9	4096.6	4060.7

Table 11: Comparison of the solutions of the CS-TSSOS and Partial Relaxation (6) for the Wang et al. [38] large-scale instances. Total time includes the time spent creating the relaxation and the time used by the interior point method in Mosek to solve it (Time IPM in the table). The CS-TSSOS results are in bold font as well the results for the smallest value of r such that the corresponding partial relaxation has an optimal solution as good as the solution of the CS-TSSOS relaxation.

5 Conclusions

We explored the idea of using information of the second order of the Lasserre hierarchy to strengthen the standard relaxation used to solve Max-Cut in the case of sparse graphs. We used two basic formulations by: (i) limiting the size of the maximal cliques added to the relaxation and (ii) including subsets of the maximal cliques that were too large.

We tested these ideas on randomly generated graphs of different densities and sizes, as well as on sparse graphs coming from applications of statistical physics. Our results showed that Partial Relaxation (6) and Augmented Partial Relaxation (7) can be very effective for solving sparse graphs with small maximal cliques but lose power as the density and/or the size of the maximal cliques increases. The results also showed that this idea provides strong bounds when compared with CS-TSSOS.

Although the partial relaxation showed potential, there is still the question of how to select the parameters r and p in the partial relaxation. This is an interesting question for future work. In particular, multilevel techniques like the ones used in [9, 10, 29] can be useful in this context.

References

[1] Elspeth Adams, Miguel F Anjos, Franz Rendl, and Angelika Wiegele. A hierarchy of subgraph projection-based semidefinite relaxations for some NP-hard graph optimization problems. *INFOR: Information Systems and Operational Research*, 53(1):40–48, 2015.

[2] Miguel F Anjos and Henry Wolkowicz. Strengthened semidefinite relaxations via a second lifting for the max-cut problem. *Discrete Applied Mathematics*, 119(1-2):79–106, 2002.

[3] Radu Baltean-Lugojan, Pierre Bonami, Ruth Misener, and Andrea Tramontani. Scoring positive semidefinite cutting planes for quadratic optimization via trained neural networks. 2018. URL `http://www.optimization-online.org/DB_FILE/2018/11/6943.pdf`.

[4] Francisco Barahona and Laszlo Ladanyi. Branch and cut based on the volume algorithm: Steiner trees in graphs and max-cut. *RAIRO-Operations Research*, 40(1):53–73, 2006.

[5] Francisco Barahona, Michael Jünger, and Gerhard Reinelt. Experiments in quadratic 0–1 programming. *Mathematical Programming*, 44(1-3):127–137, 1989.

[6] Alain Billionnet and Sourour Elloumi. Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. *Mathematical Programming*, 109(1):55–68, 2007.

[7] Jean RS Blair and Barry Peyton. *An introduction to chordal graphs and clique trees*, pages 1–29. Springer, 1993.

[8] Immanuel M Bomze, Marco Budinich, Panos M Pardalos, and Marcello Pelillo. The maximum clique problem. In *Handbook of combinatorial optimization*, pages 1–74. Springer, 1999.

[9] Juan S. Campos and Panos Parpas. A multigrid approach to SDP relaxations of sparse polynomial optimization problems. *SIAM Journal on Optimization*, 28(1):1–29, 2018.

[10] Juan S Campos, Ruth Misener, and Panos Parpas. A multilevel analysis of the Lasserre hierarchy. *European Journal of Operational Research*, 277(1):32–41, 2019.

[11] Hongbo Dong. Relaxing nonconvex quadratic functions by multiple adaptive diagonal perturbations. *SIAM Journal on Optimization*, 26(3):1962–1985, 2016.

[12] Ilse Fischer, Gerald Gruber, Franz Rendl, and Renata Sotirov. Computational experience with a bundle approach for semidefinite cutting plane relaxations of max-cut and equipartition. *Mathematical Programming*, 105(2-3):451–469, 2006.

[13] Bissan Ghaddar, Juan C Vera, and Miguel F Anjos. A dynamic inequality generation scheme for polynomial programming. *Mathematical Programming*, 156(1-2):21–57, 2016.

[14] Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*, volume 57. Elsevier, 2004.

[15] Nebojša Gvozdenović and Monique Laurent. Computing semidefinite programming lower bounds for the (fractional) chromatic number via block-diagonalization. *SIAM Journal on Optimization*, 19(2):592–615, 2008.

[16] Nebojša Gvozdenović, Monique Laurent, and Frank Vallentin. Block-diagonal semidefinite programming hierarchies for 0/1 programming. *Operations Research Letters*, 37(1):27 – 31, 2009.

[17] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[18] Sunyonga Kim and Masakazu Kojima. Second order cone programming relaxation of nonconvex quadratic optimization problems. *Optimization Methods and Software*, 15(3-4): 201–224, 2001.

[19] Nathan Krislock, Jérôme Malick, and Frédéric Roupin. Improved semidefinite bounding procedure for solving max-cut problems to optimality. *Mathematical Programming*, 143 (1-2):61–86, 2014.

[20] Nathan Krislock, Jérôme Malick, and Frédéric Roupin. BiqCrunch 2.0 user guide. `https://biqcrunch.lipn.univ-paris13.fr/BiqCrunch/repository/papers/biqcrunch-manual.pdf`, 2016. Accessed: 2020-08-06.

[21] Nathan Krislock, Jérôme Malick, and Frédéric Roupin. Biqcrunch: A semidefinite branch-and-bound method for solving binary quadratic problems. *ACM Transactions on Mathematical Software (TOMS)*, 43(4):32, 2017.

[22] Jean B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

[23] Jean B Lasserre. An explicit equivalent positive semidefinite program for nonlinear 0-1 programs. *SIAM Journal on Optimization*, 12(3):756–769, 2002.

[24] Jean B. Lasserre. Convergent SDP-relaxations in polynomial optimization with sparsity. *SIAM Journal on Optimization*, 17(3):822–843, 2006.

[25] Monique Laurent. Graphic vertices of the metric polytope. *Discrete Mathematics*, 151(1-3): 131–153, 1996.

[26] Monique Laurent and Svatopluk Poljak. On a positive semidefinite relaxation of the cut polytope. *Linear Algebra and its Applications*, 223:439–461, 1995.

[27] Frauke Liers. *Contributions to Determining Exact Ground States Of Ising Spin Glasses And To Their Physics*. PhD thesis, Verlag nicht ermittelbar, 2004.

[28] Frauke Liers, Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi. Computing exact ground states of hard Ising spin glass problems by branch-and-cut. *New optimization algorithms in physics*, 50(47-68):6, 2004.

[29] Wann-Jiun Ma, Jakub Marecek, and Martin Mevissen. A fine-grained variant of the hierarchy of Lasserre. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 580–586. IEEE, 2019.

[30] Andrea Qualizza, Pietro Belotti, and François Margot. Linear programming relaxations of quadratically constrained quadratic programs. In *Mixed Integer Nonlinear Programming*, pages 407–426. Springer, 2012.

[31] Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121(2): 307, 2010.

[32] Anureet Saxena, Pierre Bonami, and Jon Lee. Convex relaxations of non-convex mixed integer quadratically constrained programs: projected formulations. *Mathematical Programming*, 130(2):359–413, 2011.

[33] Hanif D Sherali, Evrim Dalkiran, and Jitamitra Desai. Enhancing RLT-based relaxations for polynomial programming problems via a new class of v-semidefinite cuts. *Computational Optimization and Applications*, 52(2):483–506, 2012.

[34] Hayato Waki, Sunyoung Kim, Masakazu Kojima, and Masakazu Muramatsu. Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity. *SIAM Journal on Optimization*, 17(1):218–242, 2006.

[35] Hayato Waki, Sunyoung Kim, Masakazu Kojima, Masakazu Muramatsu, and Hiroshi Sugimoto. Algorithm 883: SparsePOP—A sparse semidefinite programming relaxation of polynomial optimization problems. *ACM Transactions on Mathematical Software (TOMS)*, 35 (2):15, 2008.

[36] Jie Wang, Haokun Li, and Bican Xia. A new sparse sos decomposition algorithm based on term sparsity. In *Proceedings of the 2019 on International Symposium on Symbolic and Algebraic Computation*, pages 347–354, 2019.

[37] Jie Wang, Victor Magron, and Jean-Bernard Lasserre. TSSOS: A Moment-SOS hierarchy that exploits term sparsity. *arXiv preprint arXiv:1912.08899*, 2019.

[38] Jie Wang, Victor Magron, Jean B Lasserre, and Ngoc Hoang Anh Mai. CS-TSSOS: Correlative and term sparsity for large-scale polynomial optimization. *arXiv preprint arXiv:2005.02828*, 2020.

[39] Jie Wang, Victor Magron, and Jean-Bernard Lasserre. Chordal-TSSOS: a moment-SOS hierarchy that exploits term sparsity with chordal extension. *arXiv preprint arXiv:2003.03210*, 2020.

[40] Angelika Wiegele. *Nonlinear optimization techniques applied to combinatorial optimization problems*. Ph.D. thesis, Alpen-Adria-Universität Klagenfurt, 2006.