Additional materials for Instance-specific linear relaxations of semidefinite optimization problems.

Appendix B: Solutions to underlying problems

In Section 5 of the paper, we focused on comparing the objective of the semidefinite relaxation to that of a LP or second order cone relaxation. However, the semidefinite relaxation GW is the relaxation of combinatorial problem, and hence a natural question is whether the linear programs proposed give good solutions for the actual underlying problem. As far as we are aware, there is no algorithm to round the Lovász theta number to obtain an independent sub-graph in a graph, but we can certainly round solutions of the LPs for the max cut problem and for a problem which is not combinatorial: the sparse PCA problem.

In what follows, we present experimental results showing the quality of actual graph cuts obtained using programs $SP_{\mathscr{S}}$ and $SD_{\mathscr{S}}$ and the subsequent rounding using vectors for $\mathscr{S}$ which we describe in the next subsection. We present as well the ratios $\frac{\frac{1}{2}m+\frac{1}{4}z_{SP_{\mathscr{S}}}}{\frac{1}{2}m+\frac{1}{4}z_{SD_{\mathscr{S}}}}$ and $\frac{\frac{1}{2}m+\frac{1}{4}z_{SP_{\mathscr{S}}}}{\frac{1}{2}m+\frac{1}{4}z_{GW}}$ which we call *LP-gap* and *optimality gap*, respectively.

These experiments are done for for Erdős-Rényi random graphs, 16 graphs taken from TSPLIB, 14 graphs from the network repository. We compare them with graph cut values obtained by Mirka and Williamson on the same graph instances [1]. For the Trevisan's algorithm see [10]. The simple and the sweep algorithms are modifications of Trevisan's algorithm presented in [7]. The greedy algorithm for max cut is folklore, and the specifics are detailed in the previous reference as well. In the second subsection, we include results for the sparse PCA problem.

Finding cuts from $SD_{\mathscr{S}}$

A particular advantage of program $SD_{\mathscr{S}}$ is that its solutions are also feasible for *GW* and hence can be employed using the rounding algorithm in [5] to obtain feasible cuts, as the next observation shows.

Observation 1. *Let X be feasible for program $SD_{\mathscr{S}}$ where $\mathscr{S}$ is finite and contains the standard basis $e_1,\ldots,e_n$ of $\mathbb{R}^n$. Then, we can find a cut of value at least*

$$0.878\left(\frac{1}{2}m+\frac{1}{4}z_{SD_{\mathscr{S}}}\right).$$

.

Proof. Notice that since that for i in $1,\ldots,n$, the matrix $e_i(e_i)^T$ is the matrix of all zeros with a 1 on its ii entry. Further, since W has 0 on its diagonal we may assume that any optimal solution to $SD_{\mathscr{S}}$ has all diagonal entries equal to 1. It follows that such a solution is feasible for the Goemans and Williamson semidefinite program, and hence we can use the rounding procedure described in [5] to obtain a cut with the claimed value. □

[1] The results included here were obtained by direct communication with the authors, and will be included in a future version of [7].

In our experiments, we will be using different vectors for programs $SD_{\mathscr{S}}$ and $SP_{\mathscr{S}}$, so to avoid any confusion we denote by $\mathscr{S}'$ the set of vectors used for the relaxation $SD_{\mathscr{S}'}$. An interesting source of vectors for $\mathscr{S}'$ for program $SD_{\mathscr{S}'}$ are the eigenvectors of an optimal solution $\hat{X}$ to $SP_{\mathscr{S}}$ where $\mathscr{S} = \mathscr{E}(W)$. Although $\hat{X}$ is not PSD in general, we can take the eigenvectors $x_i, i \in [k]$ that correspond to positive eigenvalues of $\hat{X}$, and let $\mathscr{S}' = \{x_i, \ldots, x_k\}$. We observe that the computational cost of this procedure comes from solving the LP (or the SDP), whereas producing a random vector in the unit ball to find a cut is computationally cheap. Hence we produce 100 random vectors and report the value of the best cut found using those vectors in all of our experiments. We note that in [7] the same method is used to find cuts from a solution to GW.

Erdős-Rényi random graphs

Letting $\mathscr{S} = \mathscr{E}(W)$, we know, thanks to Corollary 2, that the LP gap and hence the optimality gap converge to 1 as n grows with high probability for Erdős-Rényi graphs when np is not very small. We empirically evaluate the size of cuts produced by $SD_{\mathscr{S}}$ and the subsequent rounding and present these results in table 1, together with the results obtained by Mirka and Williamson on the same graphs. Surprisingly, our procedure generates the best cut value on 15 out of the 20 instances reported in [7]. Furthermore, we obtain cuts better than the ones produced by the Goemans and Williamson rounding procedure -which first solves a semidefinite program- on 18 out of the 20 instances. Our results are reported in Table 1.

Relevant instances

We test our algorithm on 16 complete graphs from TSPLIB [8], an online library of sample instances for the Travelling Salesman Problem and related graph problems. These graph are complete weighted graphs and hence we do not report the number of edges of each graph. In Table 2 we present the optimality gap and the LP gap found for these graphs. We report as well the size of the cuts obtained following the cut generation technique presented in . Our algorithm finds the best cut in 7 of 16 instances, and a better (or equal) cut than the GW relaxation on 14 out of the 16 instances. We then present the same quotients on 14 graph instances taken from the Network Repository [9] in Table 3. Since of these graphs are weighted and some are not, we do not report the number of edges of each graph. Our algorithm finds the best cut in 8 of the 14 instances, and a better (or equal) cut than the GW relaxation on 10 out of the 14 instances.

Sparse PCA

Principal component analysis (PCA) is a popular tool in the statistical and machine learning literature used for dimensionality reduction, data visualisation and analysis.

Table 1: Optimality gap, LP-gap, and other algorithms for Erdős-Rényi random graphs for the max cut problem.

Graph	**m**	**Optimality gap**	**LP Gap**	**LP cut value**	**Greedy**	**Trevisan**	**Simple**	**sweep**	**GW**	**OPT**
$G(50,0.1)$	146	1.076	1.275	114	104	**116**	112	113	113	116
$G(50,0.25)$	313	1.021	1.119	208	199	**210**	209	**210**	199	211
$G(50,0.5)$	613	1.019	1.071	**373**	357	363	372	**373**	371	377
$G(50,0.75)$	915	1.014	1.053	**522**	510	510	510	520	513	524
$G(100,0.1)$	426	1.054	1.213	**304**	272	284	294	296	301	311
$G(100,0.25)$	1243	1.023	1.102	**772**	732	747	766	766	769	782
$G(100,0.5)$	2449	1.013	1.048	**1394**	1350	1356	1391	1391	1375	1416
$G(100,0.75)$	3725	1.006	1.032	**2025**	1978	2008	2014	2019	1982	2035
$G(200,0.1)$	1947	1.031	1.151	**1275**	1204	1242	1257	1257	1267	1296
$G(200,0.25)$	4998	1.015	1.074	**2926**	2796	2892	2920	2922	2861	2975
$G(200,0.5)$	10026	1.007	1.031	5473	5388	5397	5441	5451	**5489**	5542
$G(200,0.75)$	14818	1.005	1.025	**7860**	7731	7743	7848	7852	7835	7904
$G(350,0.1)$	6011	1.018	1.105	3645	3542	3548	3661	**3667**	3513	3735
$G(350,0.25)$	15260	1.009	1.050	**8613**	8344	8426	8535	8553	8349	8709
$G(350,0.5)$	30423	1.006	1.027	**16327**	16110	16225	16253	16298	15904	16482
$G(350,0.75)$	45696	1.003	1.017	**23818**	23613	23678	23791	23811	23674	23967
$G(500,0.1)$	12506	1.0143	1.087	7326	7174	7174	7314	7314	**7372**	7532
$G(500,0.25)$	31127	1.007	1.040	**17177**	16833	17014	17045	17075	16766	17399
$G(500,0.5)$	62167	1.004	1.022	**32978**	32557	32862	32952	32960	32713	33234
$G(500,0.75)$	93343	1.003	1.014	**48326**	47995	47995	48244	48255	47597	48576

The core idea is to find linear combinations of the variables that correspond to directions of maximal variance, called the principal components. Finding these can be accomplished by means of a singular value decomposition. For more details about applications we refer the reader to [1]. One of the main disadvantages of PCA is that the weights in the linear combination of the variables are typically non-zero, thus hindering interpretation and applicability to certain problems, such as biology or finance. In these cases it is desirable to have components that are linear combination

Table 2: Optimality gap, LP-gap, and other algorithms for some graphs on the TSPLIB graph database [8] for the max cut problem.

Graph	Optimality gap	LP Gap	LP cut value	Greedy	Trevisan	Simple	Sweep	GW
bayg29	1.027	1.139	$\mathbf{4.269}\times\mathbf{10^4}$	3.837×10^4	4.225×10^4	$\mathbf{4.269}\times\mathbf{10^4}$	$\mathbf{4.269}\times\mathbf{10^4}$	$\mathbf{4.269}\times\mathbf{10^4}$
bays29	1.025	1.139	$\mathbf{5.399}\times\mathbf{10^4}$	4.831×10^4	5.393×10^4	5.369×10^4	$\mathbf{5.399}\times\mathbf{10^4}$	5.386×10^4
berlin52	1.049	1.186	$\mathbf{4.706}\times\mathbf{10^5}$	4.532×10^5	4.616×10^5	4.465×10^5	4.681×10^5	4.522×10^5
bier127	1.036	1.170	2.323×10^7	2.162×10^7	2.300×10^7	2.322×10^7	$\mathbf{2.330}\times\mathbf{10^7}$	2.320×10^7
brazil58	1.031	1.181	2.313×10^6	$\mathbf{2.319}\times\mathbf{10^6}$	$\mathbf{2.319}\times\mathbf{10^6}$	2.315×10^6	2.315×10^6	2.180×10^6
brg180	1.009	1.029	4.563×10^7	4.118×10^7	$\mathbf{4.616}\times\mathbf{10^7}$	4.531×10^7	4.551×10^7	4.330×10^7
ch130	1.021	1.127	$\mathbf{1.888}\times\mathbf{10^6}$	1.777×10^6	1.885×10^6	$\mathbf{1.888}\times\mathbf{10^6}$	$\mathbf{1.888}\times\mathbf{10^6}$	1.887×10^6
ch150	1.024	1.109	2.525×10^6	2.500×10^6	2.521×10^6	$\mathbf{2.526}\times\mathbf{10^6}$	$\mathbf{2.526}\times\mathbf{10^6}$	2.434×10^6
d198	1.055	1.279	1.289×10^7	9.635×10^6	1.286×10^7	1.292×10^7	$\mathbf{1.293}\times\mathbf{10^7}$	$\mathbf{1.293}\times\mathbf{10^7}$
eil101	1.0218	1.133	$\mathbf{1.071}\times\mathbf{10^5}$	1.052×10^5	1.070×10^5	1.063×10^5	1.064×10^5	1.058×10^5
gr120	1.011	1.158	2.156×10^6	2.123×10^6	2.147×10^6	2.156×10^6	$\mathbf{2.157}\times\mathbf{10^6}$	2.154×10^6
gr137	1.013	1.192	3.068×10^7	2.241×10^7	3.044×10^7	3.066×10^7	$\mathbf{3.070}\times\mathbf{10^7}$	$\mathbf{3.070}\times\mathbf{10^7}$
gr202	1.030	1.180	$\mathbf{1.599}\times\mathbf{10^7}$	1.372×10^7	1.533×10^7	1.559×10^7	1.593×10^7	1.581×10^7
gr96	1.022	1.130	1.165×10^7	8.967×10^6	1.156×10^7	$\mathbf{1.166}\times\mathbf{10^7}$	$\mathbf{1.166}\times\mathbf{10^7}$	1.157×10^7
kroA100	1.007	1.156	$\mathbf{5.897}\times\mathbf{10^6}$	5.848×10^6	5.850×10^6	$\mathbf{5.897}\times\mathbf{10^6}$	$\mathbf{5.897}\times\mathbf{10^6}$	$\mathbf{5.897}\times\mathbf{10^6}$
a280	1.018	1.138	3.209×10^6	2.447×10^6	3.151×10^6	$\mathbf{3.21}\times\mathbf{10^6}$	$\mathbf{3.21}\times\mathbf{10^6}$	2.970×10^6

of just a few variables. Such components are called *sparse* components, and many different techniques have been proposed to obtain them. Cadima and Jolliffe [2] propose an ad-hoc technique consisting in setting to 0 loadings that are small enough. Zou, Hastie, and Tibshirani [12] write the PCA problem as a regression optimization problem, and then impose an ℓ_1 penalization term to encourage sparse solutions. Following the ideas of the previous section, we relax by dropping the constraint $X \succeq 0$ and imposing $v^\top X v \geq 0$ for all $v \in \mathscr{S}$ where we set $\mathscr{S} = \mathscr{E}(C)$. This yields the linear program

$$\begin{aligned} &\max_{X\in\mathbb{S}^n} \langle C, X\rangle \\ \text{s.t: } &tr(X) = 1,\ \vec{1}^\top |X| \vec{1} \leq k, \\ &v^\top X v \geq 0\ \forall v \in \mathscr{S} \\ &X_{ii} \geq 0,\ X_{ii} + X_{jj} - 2\alpha X_{ij} \geq 0\ \forall\ i, j \in [n] \\ &-1 \geq X_{ij} \geq 1,\ \forall i, j \in [n]. \end{aligned} \tag{LSPCA}$$

The linear constraints on X added on the last two lines are valid for X positive semidefinite since the cone of positive semidefinite matrices is self dual, as long as $\alpha \in [0, \sqrt{2}]$. We mention that these constraints are suggested in [11].

Table 3: Optimality gap, LP-gap, and other algorithms for some graphs of the Network repository graph database [9] for the max cut problem.

Graph	**Optimality gap**	**LP Gap**	**LP cut value**	**Greedy**	**Trevisan**	**Simple**	**Sweep**	**GW**
ENZYMES8	1.034	1.269	1.230×10^{2}	1.170×10^{2}	$\mathbf{1.260\times 10^{2}}$	$\mathbf{1.260\times 10^{2}}$	$\mathbf{1.260\times 10^{2}}$	$\mathbf{1.260\times 10^{2}}$
eco-stmarks	1.095	1.393	$\mathbf{1.756\times 10^{3}}$	8.891×10^{2}	1.190×10^{3}	9.354×10^{2}	9.354×10^{2}	9.601×10^{2}
johnson16-2-4	1.000	1.000	3.012×10^{3}	$\mathbf{3.036\times 10^{3}}$	$\mathbf{3.036\times 10^{3}}$	2.958×10^{3}	2.986×10^{3}	2.918×10^{3}
hamming6-2	1.000	1.000	$\mathbf{9.920\times 10^{2}}$	$\mathbf{9.920\times 10^{2}}$	$\mathbf{9.920\times 10^{2}}$	9.680×10^{2}	9.690	9.760×10^{2}
ia-infect-hyper	1.020	1.081	$\mathbf{1.254\times 10^{3}}$	1.213×10^{3}	1.233×10^{3}	1.227×10^{3}	1.227	1.211×10^{3}
soc-dolphins	1.090	1.279	1.160×10^{2}	1.120×10^{2}	1.120×10^{2}	1.190×10^{2}	$\mathbf{1.210\times 10^{2}}$	1.150×10^{2}
email-enron-only	1.113	1.279	4.060×10^{2}	3.920×10^{2}	$\mathbf{4.130\times 10^{2}}$	3.710×10^{2}	3.800×10^{2}	3.960×10^{2}
dwt_209	1.054	1.176	$\mathbf{5.410\times 10^{2}}$	5.250×10^{2}	5.270×10^{2}	5.250×10^{2}	5.270	5400×10^{2}
inf-USAir97	1.332	1.683	1.011×10^{2}	9.961×10^{1}	9.820×10^{1}	8.184×10^{1}	9.337×10^{1}	$\mathbf{1.074\times 10^{2}}$
ca-netscience	1.180	1.334	5.750×10^{2}	5.830×10^{2}	5.880×10^{2}	5.270×10^{2}	5.270×10^{2}	$\mathbf{6.110\times 10^{2}}$
ia-infect-dublin	1.110	1.247	$\mathbf{1.673\times 10^{3}}$	1.648×10^{3}	1.659×10^{3}	1.550×10^{3}	1.558×10^{3}	1.664×10^{3}
road-chesapeake	1.106	1.313	$\mathbf{1.250\times 10^{2}}$	1.230×10^{2}	1.230×10^{2}	1.210×10^{2}	1.230×10^{2}	$\mathbf{1.250\times 10^{2}}$
Erdős991	1.294	1.560	$\mathbf{9.610\times 10^{2}}$	9.330×10^{2}	9.340×10^{2}	7.350×10^{2}	7.580×10^{2}	9.240×10^{2}
dwt_503	1.049	1.174	1.805×10^{3}	1.822×10^{3}	1.822×10^{3}	$\mathbf{1.921\times 10^{3}}$	$\mathbf{1.921\times 10^{3}}$	1.909×10^{3}

We test the quality of our relaxation in terms of sparsity of the recovered components in the examples presented in [4] and in terms of explained variance. Explained variance is the typical way to evaluate the performance of a PCA decomposition. However, we point out that there does not seem to be a consensus in the literature for what the "explained variance" for a sparse PCA decomposition is. The reason, in a nutshell, is that components recovered in the sparse case are not mutually orthogonal [3]. In this paper, the author propose a set of corrected formulas for the the sparse pca which reduce to the usual explained variance formula when the PCs are orthogonal.

0.1 Synthetic experiments and Pit props data set

To evaluate the recovery of sparse principal components with their semidefinite relaxation, [4] use their program on a synthetic data set and on the Pit pros data set. In this subsection, we compare our linear relaxation *LSPCA* to their semidefinite program by checking the sparse components that both methods produce. D'Aspermont et al. [4] generate a synthetic matrix C with sparse components and empirically check that their proposed SDP can indeed recover the components. We repeat this experiment and show that the linear relaxation *LSPCA* obtained by setting $\mathscr{S} = \mathscr{E}(C)$ recovers as well the components. We show the results in Table 4. In the artificial example, three hidden factors are created:

$$V_1 \sim \mathcal{N}(0,290),\ V_2 \sim \mathcal{N}(0,300),\ V_3 = -0.3V_1 + 0.925V_2 + \varepsilon,\ \varepsilon \sim \mathcal{N}(0,300)$$

with V_1, V_2 and ε independent. Then, 10 observed variables are generated as follows:

$$X_i = V_j + \varepsilon_i^j,\ \varepsilon_i^j \sim \mathcal{N}(0,1),$$

with $j = 1$ for $i = 1,2,3,4$, $j = 2$ for $i = 5,6,7,8$ and $j = 3$ for $i = 9,10$ and $\{\varepsilon_i^j\}$ independent for all $i \in [10]$ and $j \in [3]$. To recover the sparse components, a solution X_1 for program *LSPCA* is found and truncated to keep only the dominant -sparse-eigenvector x_1. Then, the covariance matrix C is deflated to obtain

$$C_2 = C - (x_1^\top C x_1) x_1 x_1^\top$$

and iterated to obtain further components. As mentioned in [4], the ideal solution is to use (X_1, X_2, X_3, X_4) for the first principal component to recover factor V_1 and only (X_5, X_6, X_7, X_8) for the second component to recover V_2. We replicate the results of Table 1 in [4] using the true covariance matrix C and the oracle knowledge that the sparcity $k = 4$. We then run our linear relaxation by setting $\mathscr{S} = \mathscr{E}(C)$. We report the results in Table 4. Observe that the components that our linear relaxation are sparse, and have the same support that the ones found by the SDP and are very close in norm (ignoring the signs, which are irrelevant to this application).

Table 4: Loadings for the first two principal components on the synthetic data set with $k = 4$ for both PCs.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
SPCA PC1	0	0	0	0	0.5	0.5	0.5	0.5	0	0
SPCA PC2	0.5	0.5	0.5	0.5	0	0	0	0	0	0
LSPCA PC1	0	0	0	0	−0.598	−0.596	−0.457	−0.28	0	0
LSPCA PC2	0.482	0.366	0.762	0.226	0	0	0	0	0	0

Pit props dataset

We next consider the Pit pros dataset, introduced in [6]. This dataset consists of 180 observations of 13 measured variables. It is a regularly used dataset in the PCA literature, and is notorious for having hard- to-interpret principal components. We replicate the results of Table 2 in [4], where they present two sets of experiments. First, they set $k = 5$ for the first component and then $k = 2$ for components 2 and 3. In the second set of experiments, they set $k = 6$ for the first component and then $k = 2$ for components 2 and 3. For our linear relaxation, We let $\mathscr{S} = \mathscr{E}(C)$ and use the same values of k. We report the results in Table 5 for $k = 5,2,2$. These two tables show that our methods does recover sparse components.

Table 5: Loadings for first three principal components, for the Pit props dataset, $k = 5, 2, 2$.

	topdiam	length	moist	testsg	ovensg	ringtop	ringbud	bowmax	bowdist	whorls	clear	knots	diaknot
SPCA PC1	0.56	0.583	0	0	0	0	0.263	0.098	0.371	0.362	0	0	0
SPCA PC2	0	0	0.707	0.707	0	0	0	0	0	0	0	0	0
SPCA PC3	0	0	0	0	0	0.793	0.610	0	0	0	0	0	−0.012
LSPCA PC1	0.645	0.437	0	0	0	0.019	0.268	0	0.352	0.443	0	0	0
LSPCA PC2	0	0	0.640	0.767	0	0	0	0	0	0	0	0	0
LSPCA PC3	0	0	0	0.0464	0	0.931	0.359	0	0	0	0	0	0

Variance explained

In this subsection, we evaluate the quality of our linear relaxation in terms of the variance explained by the recovered principal components, which is the typical metric to evaluate the quality of principal components. To compute the explained variance, we use the "The fraction of total variance computed" as defined in [3], which is a corrected formula for explained variance when the components are not orthogonal. We compute this value for both the SDP relaxation and our LP relaxation for 40 data sets contained in the *Rdataset* repository, which contains the data sets preinstalled in the core of *R* as well of some of the data sets contained in some of the most popular *R* libraries. We use only data sets which contain between 8 and 20 continuous variables. For each data set, we compute 4 sparse components. We define $ev(SDP)$ to be the largest explained variance by sparse components found by the SDP method by varying the target sparsity k ranging from 1 to 30% of the number of variables in the data set. We define $ev(LP)$ similarly but this time using the linear relaxation. We point out that the number k for which these maximal values are obtained need not be the same. We present our results in figure 1. Each point corresponds to the relative error (in percentages) between the explained variances for the two methods, computed as

$$100 \cdot \frac{ev(SDP) - ev(LP)}{ev(SDP)}.$$

We note that among the 40 data sets used, only 1 has an error larger than 15%, all but 6 have an error larger than 10% and more than half (23 out of 40) have an error smaller than 5%.

Appendix C: Additional figures

Here, we provide additional figures from random QCQP experiments for densities $\Delta = 0.25$, 0.75, and 1. Additionally, we include figures for the trust region problem corresponding to these same densities.

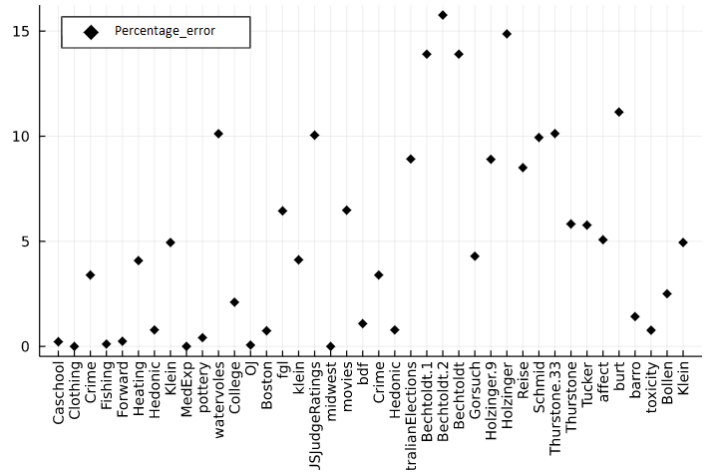

Fig. 1: Relative error in percentages between the explained variances for the two SDP methods and the LP method to recover sparse components.

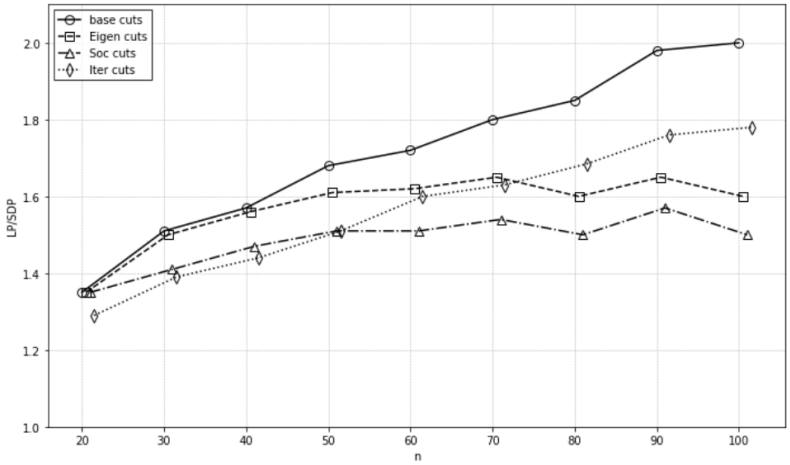

Fig. 2: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for random QCQP instances with density 0.25, $r = 1$, and $q = \frac{n}{5}$.

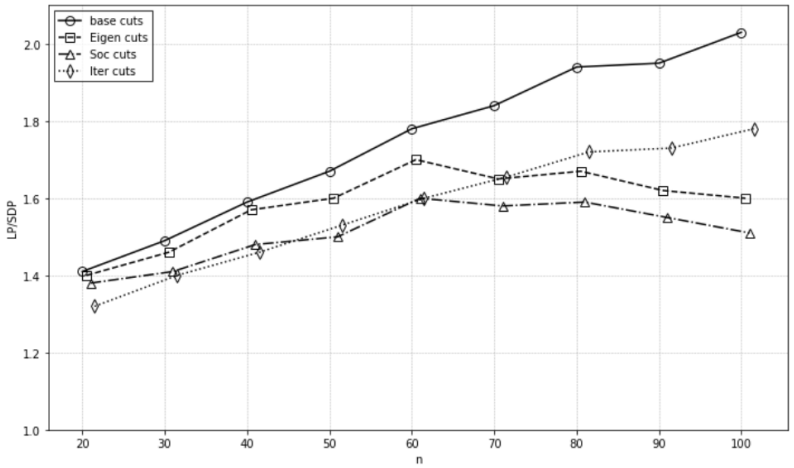

Fig. 3: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for random QCQP instances with density 0.25, $r = 1$, and $q = \frac{n}{10}$.

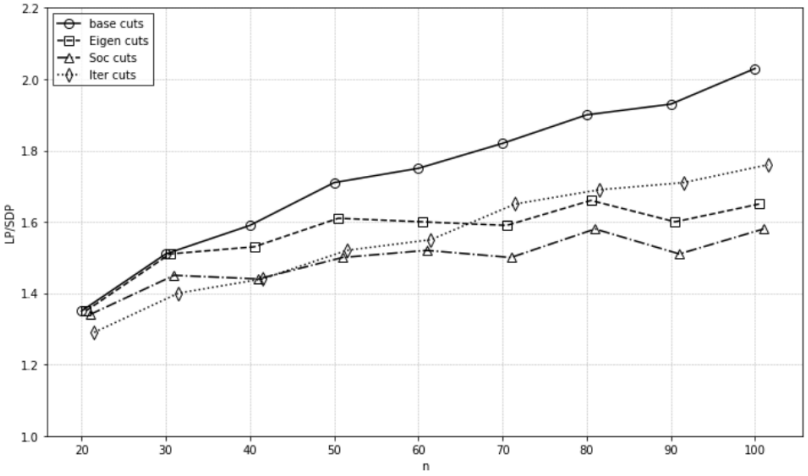

Fig. 4: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for random QCQP instances with density 0.25, $r = \frac{n}{2}$, and $q = \frac{n}{10}$.

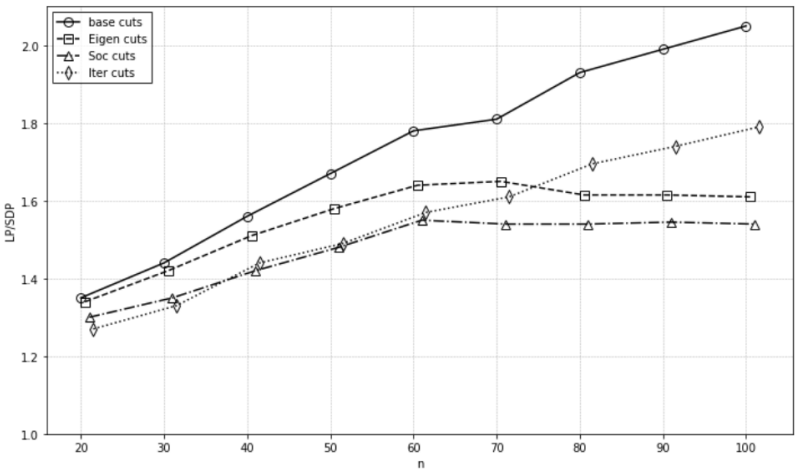

Fig. 5: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for random QCQP instances with density 0.25, $r = n$, and $q = \frac{n}{10}$.

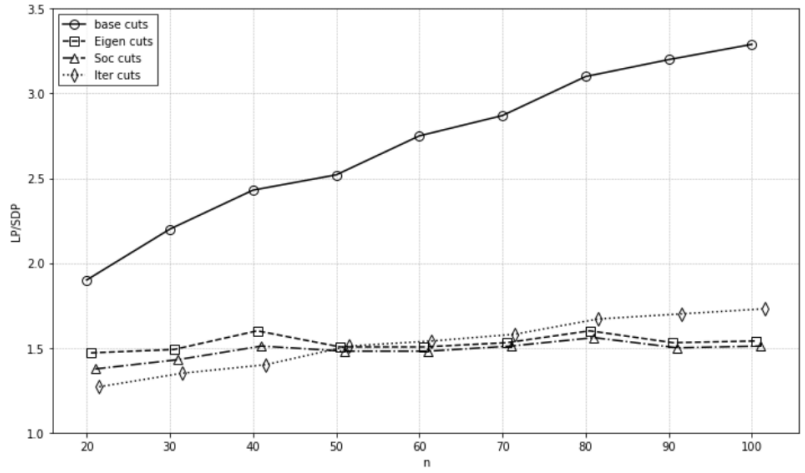

Fig. 6: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for random QCQP instances with density 0.75, $r = 1$, and $q = \frac{n}{5}$.

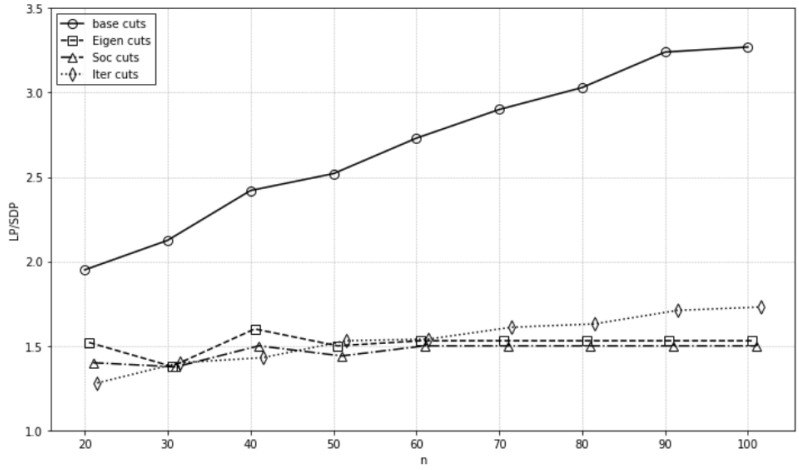

Fig. 7: Quality of the ratios $\frac{z_{\mathscr{E}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for random QCQP instances with density 0.75, $r = 1$, and $q = \frac{n}{10}$.

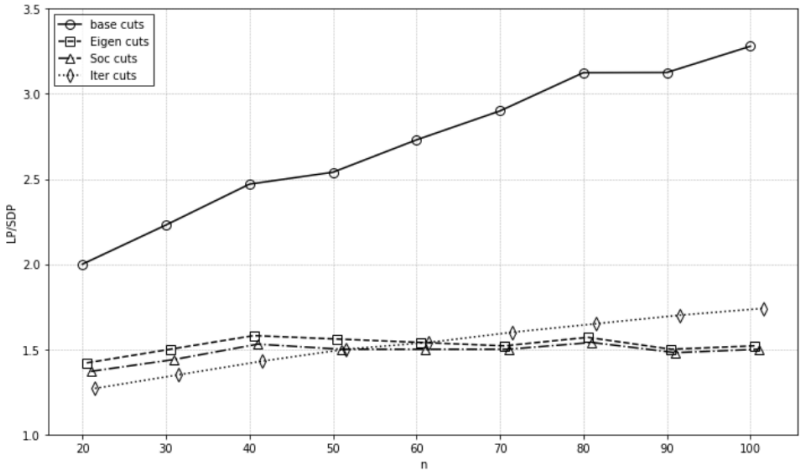

Fig. 8: Quality of the ratios $\frac{z_{\mathscr{E}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for random QCQP instances with density 0.75, $r = \frac{n}{2}$, and $q = \frac{n}{10}$.

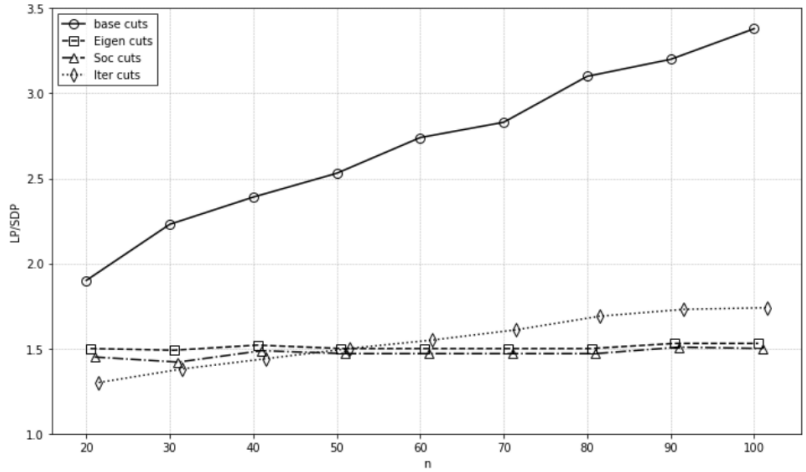

Fig. 9: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for random QCQP instances with density 0.75, $r = n$, and $q = \frac{n}{10}$.

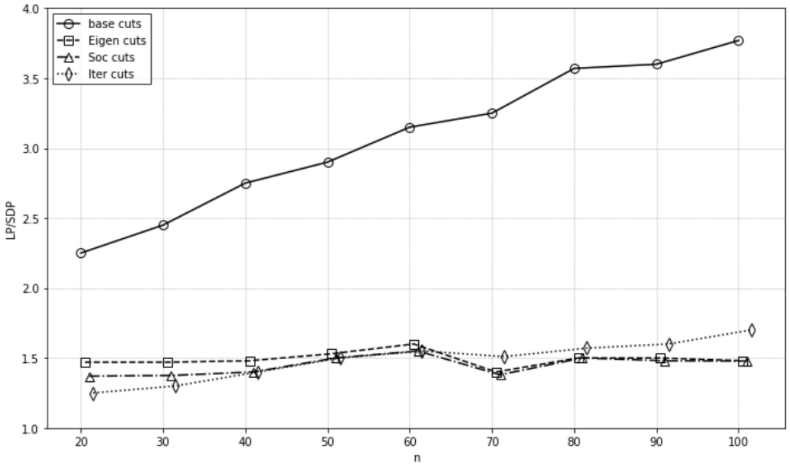

Fig. 10: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for random QCQP instances with density 1, $r = 1$, and $q = \frac{n}{5}$.

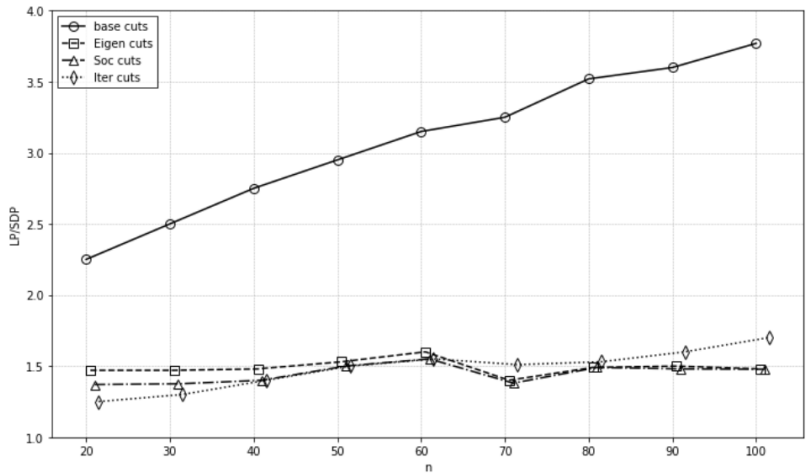

Fig. 11: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for random QCQP instances with density 1, $r = 1$, and $q = \frac{n}{10}$.

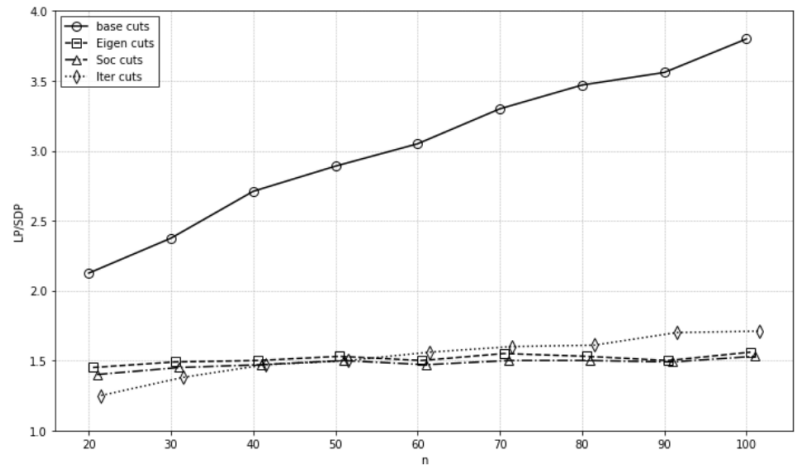

Fig. 12: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for random QCQP instances with density 1, $r = \frac{n}{2}$, and $q = \frac{n}{10}$.

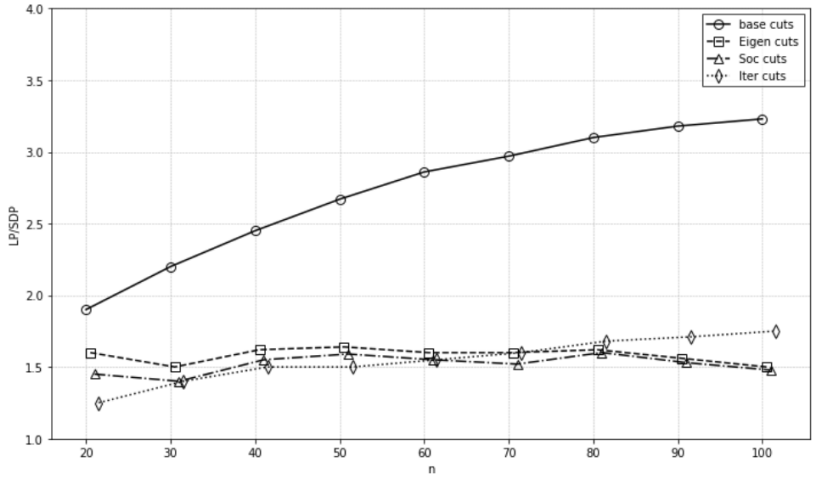

Fig. 13: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for random QCQP instances with density 1, $r = n$, and $q = \frac{n}{10}$.

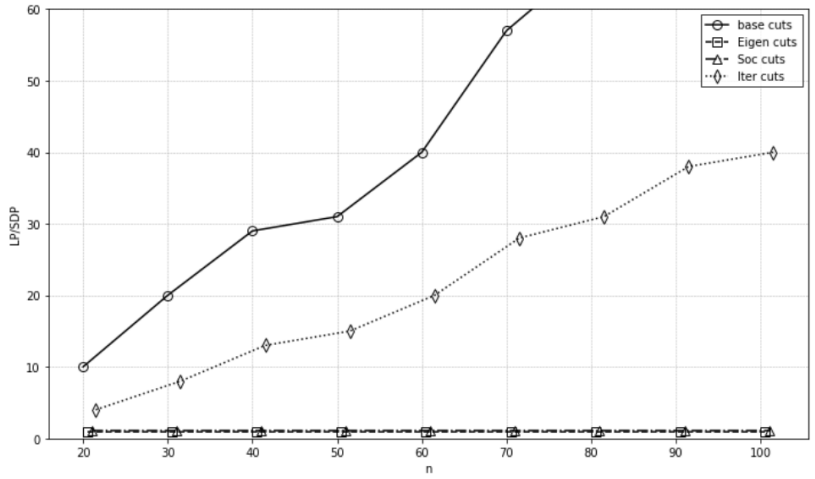

Fig. 14: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for instances of the extended trust region problem with density 0.25, $r = 1$, and $q = \frac{n}{5}$.

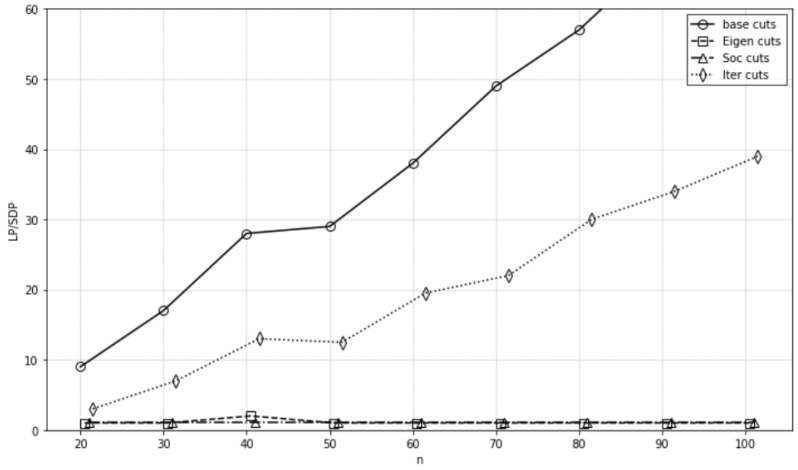

Fig. 15: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for instances of the extended trust region problem with density 0.25, $r = 1$, and $q = \frac{n}{10}$.

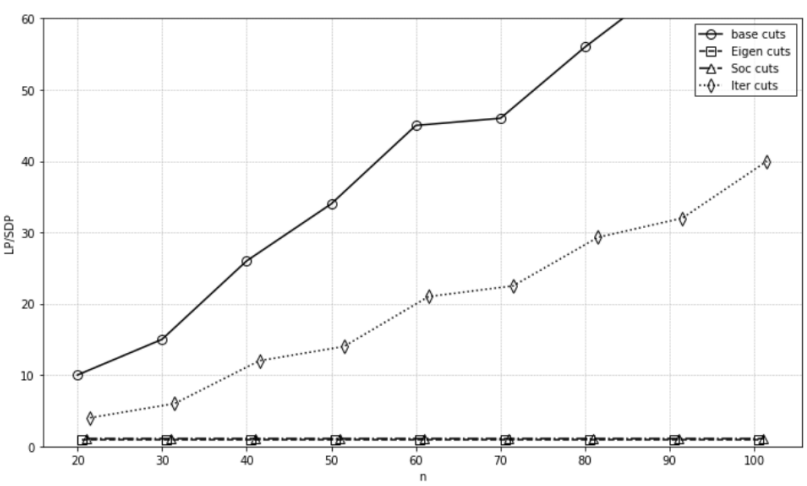

Fig. 16: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for instances of the extended trust region problem with density 0.25, $r = \frac{n}{2}$, and $q = \frac{n}{10}$.

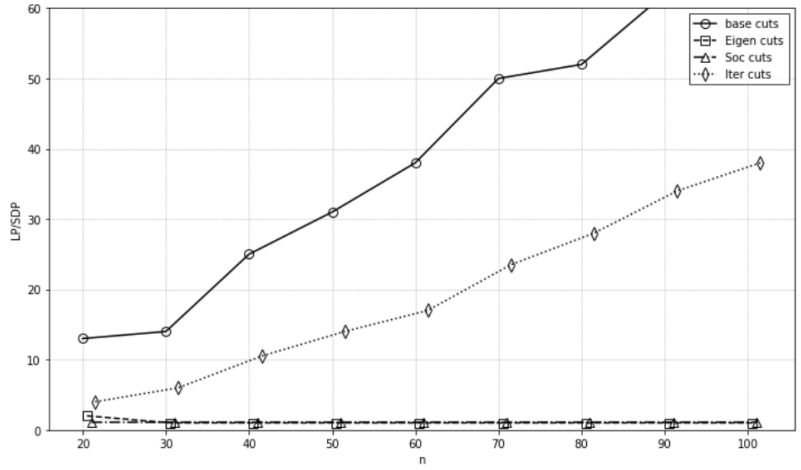

Fig. 17: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for instances of the extended trust region problem with density 0.25, $r = n$, and $q = \frac{n}{10}$.

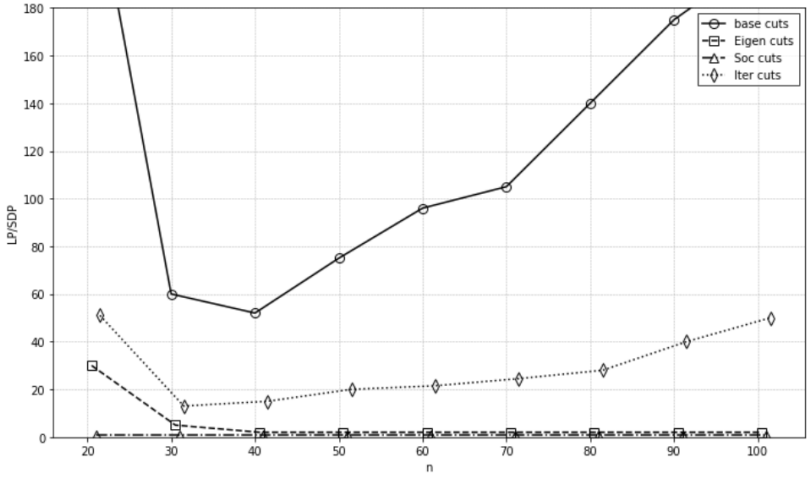

Fig. 18: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for instances of the extended trust region problem with density 0.75, $r = 1$, and $q = \frac{n}{5}$.

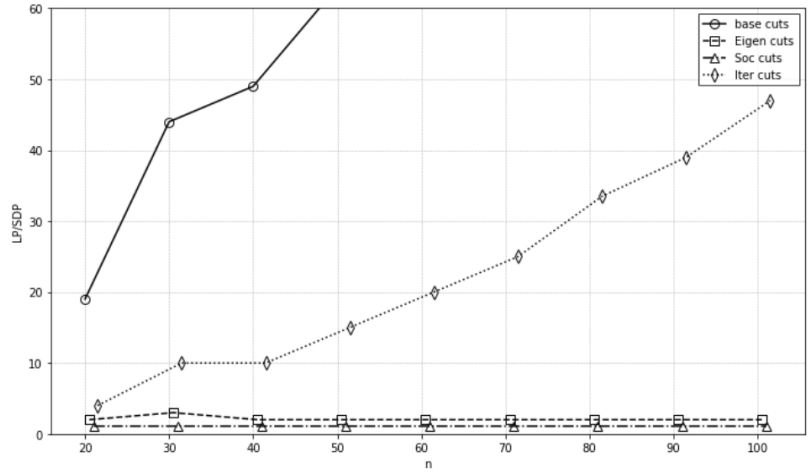

Fig. 19: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for instances of the extended trust region problem with density 0.75, $r = 1$, and $q = \frac{n}{10}$.

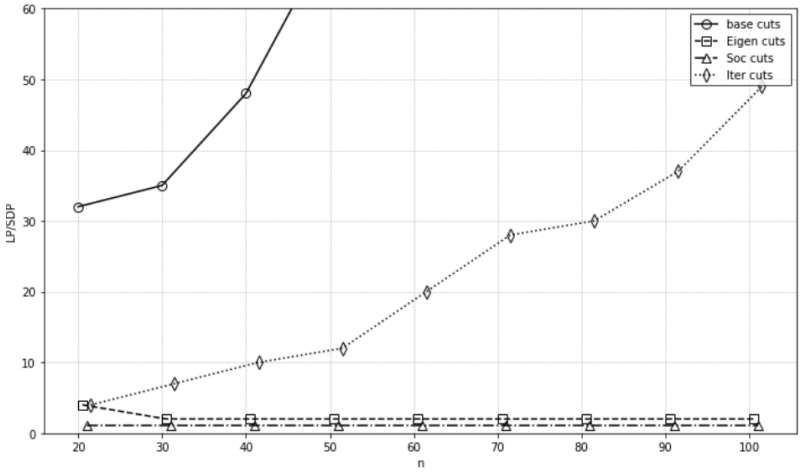

Fig. 20: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for instances of the extended trust region problem with density 0.75, $r = \frac{n}{2}$, and $q = \frac{n}{10}$.

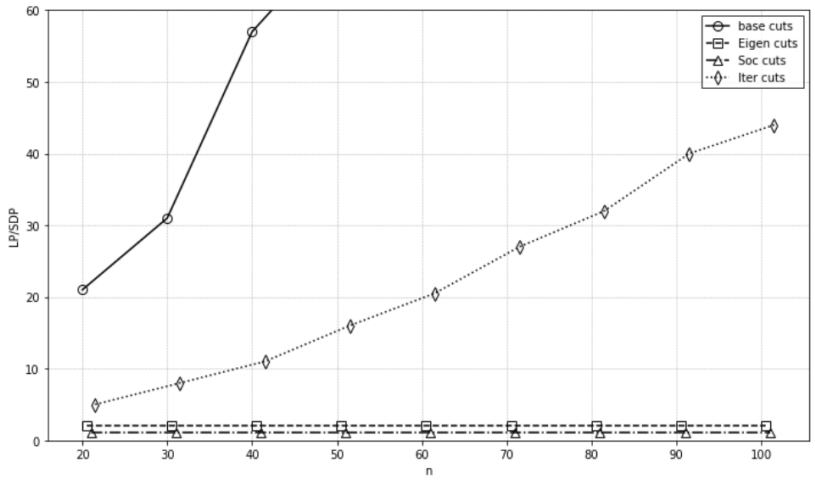

Fig. 21: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for instances of the extended trust region problem with density 0.75, $r = n$, and $q = \frac{n}{10}$.

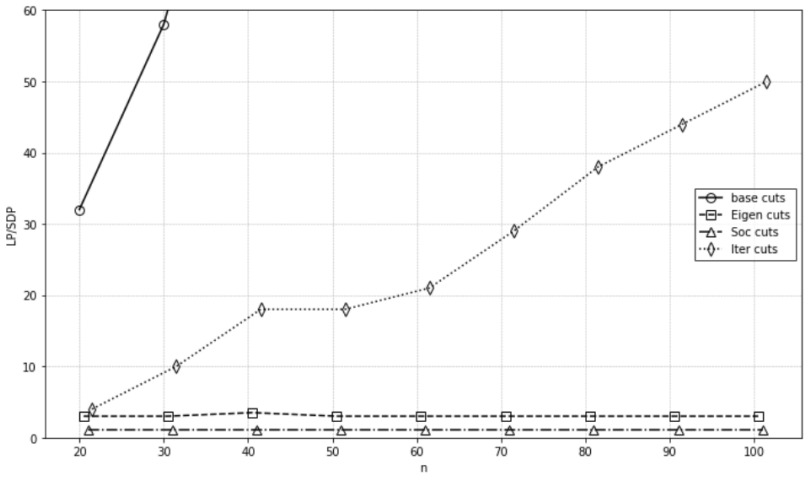

Fig. 22: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for instances of the extended trust region problem with density 1, $r = 1$, and $q = \frac{n}{5}$.

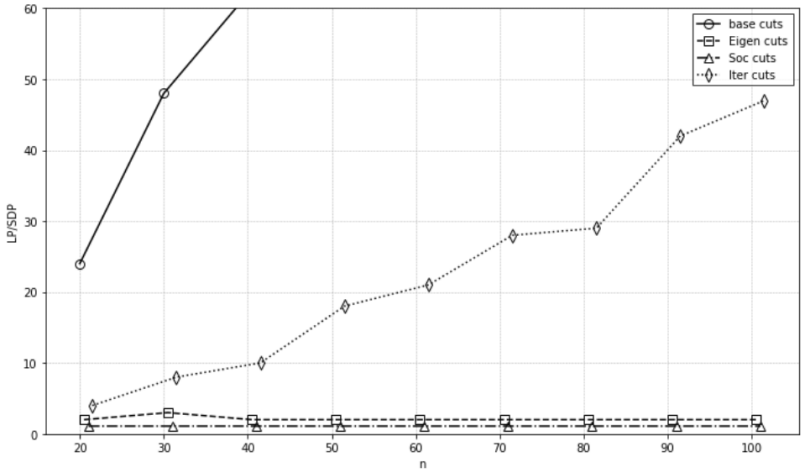

Fig. 23: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for instances of the extended trust region problem with density 1, $r = 1$, and $q = \frac{n}{10}$.

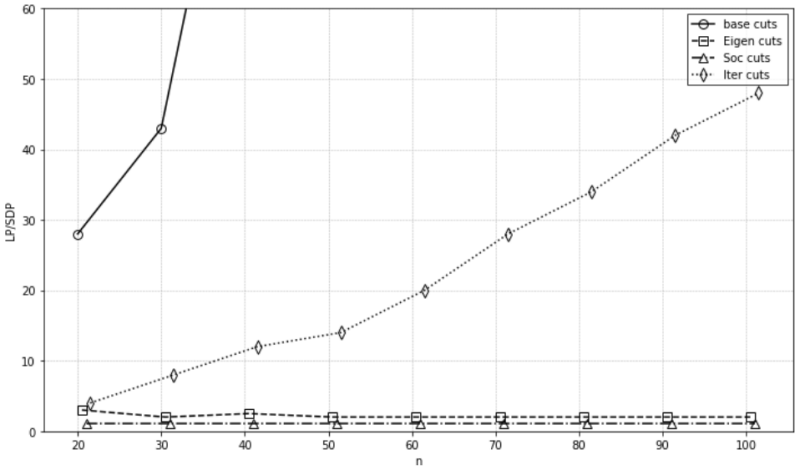

Fig. 24: Quality of the ratios $\frac{z_{\mathscr{S}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for instances of the extended trust region problem with density 1, $r = \frac{n}{2}$, and $q = \frac{n}{10}$.

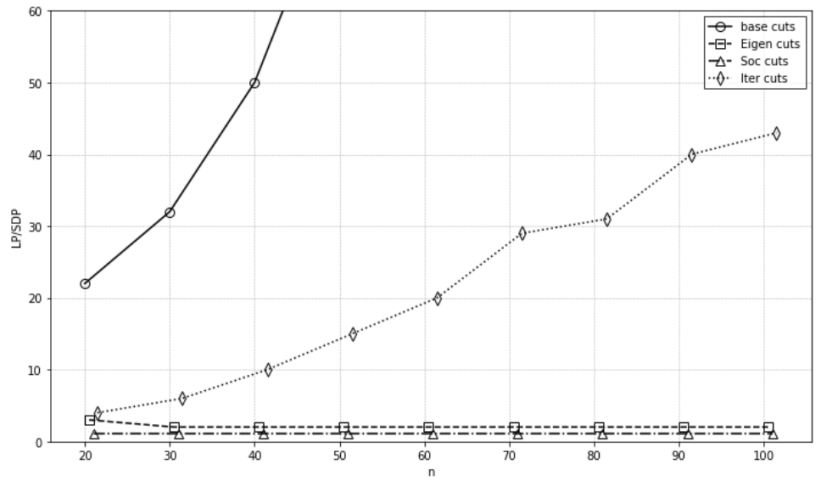

Fig. 25: Quality of the ratios $\frac{z_{\mathscr{L}}}{z_{sdp}}$ (eigen cuts), $\frac{z_n}{z_{sdp}}$ (oracle cuts), $\frac{z_0}{z_{sdp}}$ (base cuts), and $\frac{z_{soc}}{z_{sdp}}$ for instances of the extended trust region problem with density 1, $r = n$, and $q = \frac{n}{10}$.

References

1. Abdi, H., Williams, L.J.: Principal component analysis. Wiley interdisciplinary reviews: computational statistics **2**(4), 433–459 (2010)
2. Cadima, J., Jolliffe, I.T.: Loading and correlations in the interpretation of principle compenents. Journal of Applied Statistics **22**(2), 203–214 (1995)
3. Camacho, J., Smilde, A.K., Saccenti, E., Westerhuis, J.A.: All sparse pca models are wrong, but some are useful. part i: computation of scores, residuals and explained variance. Chemometrics and Intelligent Laboratory Systems **196**, 103907 (2020)
4. d'Aspremont, A., Ghaoui, L., Jordan, M., Lanckriet, G.: A direct formulation for sparse pca using semidefinite programming. Advances in neural information processing systems **17** (2004)
5. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM (JACM) **42**(6), 1115–1145 (1995)
6. Jeffers, J.N.: Two case studies in the application of principal component analysis. Journal of the Royal Statistical Society: Series C (Applied Statistics) **16**(3), 225–236 (1967)
7. Mirka, R., Williamson, D.P.: An experimental evaluation of semidefinite programming and spectral algorithms for max cut. In: 20th International Symposium on Experimental Algorithms (SEA 2022). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2022)
8. Reinelt, G.: Tsplib—a traveling salesman problem library. ORSA journal on computing **3**(4), 376–384 (1991)
9. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: AAAI (2015). URL `http://networkrepository.com`
10. Trevisan, L.: Max cut and the smallest eigenvalue. SIAM Journal on Computing **41**(6), 1769–1786 (2012)
11. Wang, Y., Tanaka, A., Yoshise, A.: Polyhedral approximations of the semidefinite cone and their application. Computational Optimization and Applications **78**(3), 893–913 (2021)
12. Zou, H., Hastie, T., Tibshirani, R.: Sparse principal component analysis. Journal of Computational and Graphical Statistics **15**(2), 265–286 (2006)