Chapter 4
Oracle-Based Primal-Dual Algorithms for Packing and Covering Semidefinite Programs

Khaled Elbassioni and Kazuhisa Makino

Abstract *Packing and covering* semidefinite programs (SDPs) appear in natural relaxations of many combinatorial optimization problems as well as a number of other applications. Recently, several techniques have been proposed that utilize the particular structure of this class of problems in order to obtain more efficient algorithms than those offered by general SDP solvers. For certain applications, it may be necessary to deal with SDPs with a very large number of (e.g., exponentially or even infinitely many) constraints. In this chapter, we give an overview of some of the techniques that can be used to solve this class of problems, focusing on multiplicative weight updates and logarithmic-potential methods.

4.1 Packing and Covering Semidefinite Programs

We denote by $\mathbb{S}^n$ the set of all $n \times n$ real symmetric matrices and by $\mathbb{S}^n_+ \subseteq \mathbb{S}^n$ the set of all $n \times n$ positive semidefinite (psd) matrices. We consider the following pairs of *packing-covering* semidefinite programs (SDPs):

$$\begin{array}{ll} & z_I^* = \max\ C \bullet X \qquad \text{(PACKING- I)} \\ \text{s.t.} & A_i \bullet X \le b_i, \forall i \in [m] \\ & X \in \mathbb{S}^n,\ X \succeq 0 \end{array} \quad \left| \quad \begin{array}{ll} & z_I^* = \min\ b^T y \qquad \text{(COVERING- I)} \\ \text{s.t.} & \sum_{i=1}^m y_i A_i \succeq C \\ & y \in \mathbb{R}^m,\ y \ge 0, \end{array} \right.$$

K. Elbassioni (✉)
Khalifa University of Science and Technology, P.O. Box 127788, Abu Dhabi, United Arab Emirates
e-mail: khaled.elbassioni@ku.ac.ae

K. Makino
Research Institute for Mathematical Sciences (RIMS), Kyoto University, Kyoto 606-8502, Japan
e-mail: makino@kurims.kyoto-u.ac.jp

N. Katoh et al. (eds.), *Sublinear Computation Paradigm*,
https://doi.org/10.1007/978-981-16-4095-7_4

$$\begin{array}{ll} z^*_{II} = \min\ C \bullet X & \text{(COVERING- II)} \\ \text{s.t. } A_i \bullet X \geq b_i, \forall i \in [m] & \\ X \in \mathbb{S}^n,\ X \succeq 0 & \end{array} \qquad \begin{array}{ll} z^*_{II} = \max\ b^T y & \text{(PACKING- II)} \\ \text{s.t.} \sum_{i=1}^{m} y_i A_i \preceq C & \\ y \in \mathbb{R}^m,\ y \geq 0, & \end{array}$$

where $C, A_1, \ldots, A_m \in \mathbb{S}^n_+$ are (non-zero) psd matrices, and $b = (b_1, \ldots, b_n)^T \in \mathbb{R}^m_+$ is a non-negative vector. In the above, $C \bullet X := \mathrm{Tr}(CX) = \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij}$, and "$\succeq$" is the *Löwner order* on matrices: $A \succeq B$ if and only if $A - B$ is psd. This type of SDP arises in many applications. See, for example, [14, 15] and the references therein.

We assume the following throughout this chapter:

(A) $b_i > 0$ and hence $b_i = 1$ for all $i \in [m]$.

It is known that, under assumption (A), *strong duality* holds for problems (PACKING- I) and (COVERING- I) (resp., (PACKING- II) and (COVERING- II)). Let $\epsilon \in (0, 1]$ be a given constant. We say that (X, y) is an ϵ-*optimal* primal-dual solution for (PACKING- I)-(COVERING- I) if (X, y) is a primal-dual feasible pair such that

$$C \bullet X \geq (1 - \epsilon) b^T y \geq (1 - \epsilon) z^*_I. \tag{4.1}$$

Similarly, we say that (X, y) is an ϵ-optimal primal-dual solution for (PACKING- II)-(COVERING- II) if (X, y) is a primal-dual feasible pair such that

$$C \bullet X \leq (1 + \epsilon) b^T y \leq (1 + \epsilon) z^*_{II}. \tag{4.2}$$

In this chapter, we allow the number of constraints m in (PACKING- I) (resp., (COVERING- II)) to be *exponentially* (or even infinitely) large, so we assume the availability of the following *oracle*:

Max(Y)(resp., Min(Y)) : Given $Y \in \mathbb{S}^n_+$, find $i \in \mathrm{argmax}_{i \in [m]} A_i \bullet Y$ (resp., $i \in \mathrm{argmin}_{i \in [m]} A_i \bullet Y$).

Note that an *approximation* oracle computing the above maximum (resp., minimum) within a factor of $(1 - \epsilon)$ (resp., $(1 + \epsilon)$) is also sufficient for our purposes. A primal-dual solution (X, y) to (COVERING- I) (resp., (PACKING- II)) is said to be η-*sparse* if the size of $\mathrm{supp}(y) := \{i \in [m] : y_i > 0\}$ is at most η.

When $C = I = I_n$ (which is the identity matrix in $\mathbb{R}^{n \times n}$) and $b = \mathbf{1}_m$ (which is the vector containing all ones in $\mathbb{R}^m$), we say that the packing-covering SDPs are in *normalized* form. It can be shown (see, e.g., [7, 16]) that, to within a multiplicative factor of $(1 + \epsilon)$ in the objective, any pair of packing-covering SDPs of the form (PACKING- I)-(COVERING- I) can be brought to normalized form in $O(n^3)$ time while increasing the oracle time by only $O(n^\omega)$, where ω is the exponent of matrix multiplication, under the following assumption:

(B-I) There exist r matrices, say $A_1, \ldots, A_r$, such that $\hat{A} := \sum_{i=1}^{r} A_i \succ 0$. In particular, $\mathrm{Tr}(X) \leq \tau := \frac{r}{\lambda_{\min}(\hat{A})}$ for any optimal solution X for (PACKING- I), and we may assume that $r = 1$ and $A_1 = \frac{1}{\tau} I$.

Similarly, it can be shown that, to within a multiplicative factor of $(1 + \epsilon)$ in the objective, any pair of packing-covering SDPs of the form (PACKING- II)-(COVERING- II) can be brought to normalized form in $O(n^3)$ time, while increasing the oracle time by only $O(n^\omega)$. Moreover, we may assume in this normalized form that

(B-II) $\lambda_{\min}(A_i) = \Omega\left(\frac{\epsilon}{n} \cdot \min_{i'} \lambda_{\max}(A_{i'})\right)$ for all $i \in [m]$,

where, for a psd matrix $B \in \mathbb{S}^n_+$, we denote by $\{\lambda_j(B) : \ j = 1, \ldots, n\}$ the eigenvalues of B, and by $\lambda_{\min}(B)$ and $\lambda_{\max}(B)$ the minimum and maximum eigenvalues of B, respectively. Given additional $O(mn^2)$ time, we may also assume that

(B-II') $\frac{\lambda_{\max}(A_i)}{\lambda_{\min}(A_i)} = O\left(\frac{n^2}{\epsilon^2}\right)$ for all $i \in [m]$.

Thus, the remainder of this chapter focuses on normalized problems.

Mixed packing and covering SDPs.
We also consider the following *mixed* packing-covering feasibility SDPs:

$$\begin{aligned} & A_i \bullet X \leq b_i, \ \ \forall i \in [m_p] \qquad \text{(MIX- PACK- COVER)} \\ & B_i \bullet X \geq d_i, \ \ \forall i \in [m_c] \\ & X \in \mathbb{S}^n, \ X \succeq 0, \end{aligned}$$

where $A_1, \ldots, A_{m_p}, B_1, \ldots, B_{m_c} \in \mathbb{R}^{n \times n}$ are psd matrices, and $b = (b_1, \ldots, b_{m_p})^T$, $d = (d_1, \ldots, d_{m_c})^T$ are non-negative real vectors.

A matrix $X \in \mathbb{S}^n_+$ is an ϵ-approximate solution for (MIX- PACK- COVER) if $A_i \bullet X \leq b_i$ for all $i \in [m_p]$ and $B_i \bullet X \geq (1 - \epsilon) d_i$ for all $i \in [m_c]$.

4.2 Applications

4.2.1 SDP relaxation for Robust MAXCUT

Given a simple undirected graph $G = (V, E)$ on $n = |V|$ vertices with non-negative edge weights $w \in \mathbb{R}^E_+$, the objective in the well-known MAXCUT problem is to find a subset of the vertices $X \subset V$ that maximizes the weight of the cut: $w(X, V \setminus X) := \sum_{u \in X, \ v \in V \setminus X} w_{uv}$. The best-known approximation algorithm (with approximation ratio $0.878\ldots$) [10] for MAXCUT is based on the following SDP relaxation:

$$\max\ L(w) \bullet X \qquad \text{(MaxCut- SDP)}$$

$$\begin{aligned} \text{s.t.}\quad & \mathbf{1}_i \mathbf{1}_i^T \bullet X = 1, \quad \forall i \in [n] \\ & X \in \mathbb{R}^{n\times n},\ X \succeq 0. \end{aligned} \tag{4.3}$$

By simply changing the equality in (4.3) into an inequality, this can be written in the form (Packing- I), with $A_i := \mathbf{1}_i \mathbf{1}_i^T$ and $C := L(w) \succeq 0$ being the *Laplacian* matrix of G, defined as follows:

$$L_{ij}(w) = \begin{cases} \sum_{k=1}^{n} w_{ik} & \text{if } i = j, \\ -w_{ij} & \text{if } \{i, j\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Based on this relaxation, the following result is obtained using the *scalar multiplicative weights update (MWU)* method:

Theorem 4.1 ([18]) *There is a randomized algorithm for finding an ϵ-optimal solution for (*MaxCut- SDP*) in time $\tilde{O}(\frac{nm}{\epsilon^3})$, where n and m respectively denote the number of vertices and edges in a given graph.*

Under the *robust optimization* framework, one assumes the weights are *not known precisely*, but instead are given by a *convex uncertainty* set $\mathcal{W} \subseteq \mathbb{R}^n_+$, where it is necessary to find a (near)-optimal solution under the *worst-case* choice $w \in \mathcal{W}$ in the uncertainty set:

$$\max \min_{w\in\mathcal{W}} L(w) \bullet X \qquad \text{Robust- MaxCut- SDP}$$

$$\begin{aligned} \text{s.t.}\quad & \mathbf{1}_i \mathbf{1}_i^T \bullet X \le 1, \quad \forall i \in [n] \\ & X \in \mathbb{R}^{n\times n},\ X \succeq 0. \end{aligned} \tag{4.4}$$

By "guessing" the value τ of an optimal solution (via binary search), (4.4) can be reduced to

$$\text{Robust- MaxCut- SDP} \quad \begin{aligned} \text{s.t.}\quad & \min\ I \bullet X \\ & \mathbf{1}_i \mathbf{1}_i^T \bullet X \ge 1, \quad \forall i \in [n] \\ & \frac{1}{\tau} L(w) \bullet X \ge 1, \quad \forall w \in \mathcal{W} \\ & X \in \mathbb{R}^{n\times n},\ X \succeq 0. \end{aligned}$$

Thus, we obtain a covering SDP (of type (COVERING- II)) with an *infinite* number of constraints, given by a minimization oracle over the convex set $\mathcal{W}$. We can use the *matrix logarithmic-potential* method to obtain the following result:

Theorem 4.2 *There is a randomized algorithm that finds an ϵ-optimal solution for (4.4) in time $\tilde{O}\left(\frac{n^{\omega+1}}{\epsilon^{2.5}} + \frac{n\mathcal{T}}{\epsilon^2}\right)$, where $\mathcal{T}$ is the time needed to optimize a linear function over $\mathcal{W}$.*

Note that for this reduction to remain valid, it is sufficient to find an ϵ-optimal solution to (4.4) for any $\epsilon = o\left(\frac{1}{n}\right)$.

4.2.2 Mahalanobis Distance Learning

Given a psd matrix $X \in \mathbb{S}^n$, the *X-Mahalanobis distance* between two points $a, b \in \mathbb{R}^n$ is defined as

$$d_X(a, b) := \sqrt{(a-b)^T X (a-b)}.$$

The distance function $d_X(\cdot, \cdot)$ is a semi-metric; that is, it is symmetric ($d_X(a, b) = d_X(a, b)$) and satisfies the triangle inequality ($d_X(a, c) \leq d_X(a, b) + d_M(b, c)$), and it is also a metric if $X \succ 0$ (as in this case, $d_X(a, b) = 0$ if and only if $a = b$).

The *Mahalanobis distance learning* problem is defined as follows [28]: Given sets $\mathcal{C}_s$ and $\mathcal{C}_d$ of *similar* and *dissimilar* pairs of points in $\mathbb{R}^n$, respectively, a similarity parameter $\sigma_s \in \mathbb{R}_+$ and a dissimilarity parameter $\sigma_d \in \mathbb{R}_+$, the objective is to find a matrix X such that all the pairs in $\mathcal{C}_s$ are "close" and all the pairs in $\mathcal{C}_d$ are "far" with respect to the distance function $d_X(\cdot, \cdot)$:

$$(a-b)^T X (a-b) \leq \sigma_s, \ \forall (a, b) \in \mathcal{C}_s \tag{4.5}$$

$$(a-b)^T X (a-b) \geq \sigma_d, \ \forall (a, b) \in \mathcal{C}_d \tag{4.6}$$

$$X \in \mathbb{S}^n, \ X \succeq 0. \tag{4.7}$$

Note that this can be written in the form (MIX- PACK- COVER), with $|\mathcal{C}_s|$ packing constraints of the form $A_{a,b} \bullet X \leq \sigma_s$, where $A_{a,b} = (a-b)(a-b)^T$ for $(a, b) \in \mathcal{C}_s$, and $|\mathcal{C}_d|$ covering constraints of the form $B_{a,b} \bullet X \geq \sigma_d$, where $B_{a,b} = (a-b)(a-b)^T$ for $(a, b) \in \mathcal{C}_d$.

We can use the *scalar MWU* method to obtain the following result:

Theorem 4.3 *There is a deterministic algorithm that finds an ϵ-feasible solution for (4.5)-(4.2.2) in time $\tilde{O}(\frac{m(m+n^3)}{\epsilon^2})$, where n is the dimension of the point sets and $m := |\mathcal{C}_s|^2 + |\mathcal{C}_d|^2$.*

We remark that it is plausible that further improvements (possibly by another factor of $O(m)$) are possible via *rank-one tricks* and the use of *approximate eigenvalue* computations.

4.2.3 Related Work

Problems (PACKING- I)-(COVERING- I) and (PACKING- II)-(COVERING- II) can be solved using general SDP solvers, such as interior-point methods. For example, the barrier method (see, e.g., [22]) can compute a solution within an *additive* error of ϵ from the optimal in time $O(\sqrt{n}m(n^3 + mn^2 + m^2)\log\frac{1}{\epsilon})$ (see also [1, 27]). However, due to the special nature of (PACKING- I)-(COVERING- I) and (PACKING- II)-(COVERING- II), better algorithms can be obtained. Most of the improvements are obtained by using *first-order methods* [2, 3, 5, 6, 8, 15–18, 21, 23, 24], or second-order methods [13, 14]. In general, we can classify these algorithms according to whether they are *(semi) width-independent*, are *parallel*, *output sparse solutions*, or are *oracle-based*, as follows.

(I) *(Semi) width-independent*: The running time of the algorithm depends *polynomially* on the bit length of the input. For example, in the of case of (PACKING- I)-(COVERING- I), the running time is $\text{poly}(n, m, \mathcal{L}, \log\tau, \frac{1}{\epsilon})$, where $\mathcal{L}$ is the maximum bit length needed to represent any number in the input. In contrast, the running time of a width-dependent algorithm depends polynomially on a "width parameter" ρ, which is polynomial in $\mathcal{L}$ and τ.

(II) *Parallel*: The algorithm takes $\text{polylog}(n, m, \mathcal{L}, \log\tau) \cdot \text{poly}(\frac{1}{\epsilon})$ time on a poly $(n, m, \mathcal{L}, \log\tau, \frac{1}{\epsilon})$ number of processors.

(III) *Sparse*: The algorithm outputs an η-sparse solution to (COVERING- I) (resp., (PACKING- II)) for $\eta = \text{poly}(n, \log m, \mathcal{L}, \log\tau, \frac{1}{\epsilon})$ (resp., $\eta = \text{poly}(n, \log m, \mathcal{L}, \frac{1}{\epsilon})$), where τ is a parameter that bounds the trace of any optimal solution X;

(IV) *Oracle-based*: The only access the algorithm has to the matrices $A_1, \ldots, A_m$ is via the maximization/minimization oracle, and hence the running time is independent of m.

Table 4.1 below gives a summary[1] of the most relevant results together with their classifications according to the four criteria above. We note that almost all of these algorithms for packing/covering SDPs are generalizations of similar algorithms for packing/covering linear programs (LPs), and most of them are essentially based on an *exponential potential function* in the form of *scalar exponentials*, such as [3, 18], or *matrix exponential* [2, 5, 6, 15, 17]. For instance, several of these results use the scalar or matrix versions of the MWU method (see, e.g., [4]), which are extensions of similar methods for packing/covering LPs [9, 11, 25, 29].

In [12], a different type of algorithm was given for covering LPs (indeed, more generally, for a class of concave covering inequalities) based on a *logarithmic* potential function. In [7], it was shown that this approach could be extended to provide sparse solutions for both versions of packing and covering SDPs.

As we can see from the table, among all the algorithms, only the matrix (MWU and logarithmic-potential) algorithms are oracle-based (and hence produce sparse

[1] We provide rough estimates of the bounds, as some of them are not stated explicitly in the corresponding paper in terms of the parameters we consider here.

Table 4.1 Different algorithms for packing/covering SDPs

Paper	Problem	Technique	Most expensive operation	# Iterations	Width-indep.	Parallel	Sparse	Oracle-based
[3, 18]	(PACKING- I) (COVERING- II)	MWU	Max/min eigenvalue of a psd matrix $\tilde{O}(\frac{n^2}{\epsilon})$	$O(\frac{\rho \log m}{\epsilon^2})$	No	No	No*	No
[6]	(PACKING- I) (COVERING- II)	Matrix MWU	Matrix exponential $O(n^3)$	$O(\frac{\rho^2 \tau^2 \log n}{\epsilon^2 (z_I^*)^2})$	No	No	No*	Yes
[13, 15]	(PACKING- I)	Nesterov's smoothing technique [20, 21]	Matrix exponential $O(n^3)$	$O(\frac{\tau \log m}{\epsilon})$	No	No	No	No
[14]	(COVERING- II)	Nesterov's smoothing technique [20, 21]	Min eigenvalue of a non-psdmatrix $O(n^3)$	$O(\frac{\rho^2 \log(nm)}{\epsilon})$	No	No	No	No
[16]	(PACKING- I) & (COVERING- II)	MWU technique [20, 21]	Eigenvalue decomposition $O(n^3)$	$O(\frac{\log^{13} n \log m}{\epsilon^{13}})$	Yes	Yes	No	No
[23, 24]	(PACKING- II)& (COVERING- II)	Matrix MWU	Matrix exponential $O(n^3)$	$O(\frac{\log^3 m}{\epsilon^3})$	Yes	Yes	No	No
[2]	(PACKING- I)& (COVERING- II)	Gradient Descent + Mirror Descent	Matrix exponential $O(n^3)$	$O(\frac{\log^2 (mn) \log \frac{1}{\epsilon}}{\epsilon^2})$	Yes	Yes	No	No
This chapter (Sect. 4.5)	(PACKING- II)& (COVERING- II)	Matrix MWU	Matrix exponential $O(n^3)$	$O(\frac{n \log n}{\epsilon^2})$	Yes	No	Yes	Yes
[7]	(PACKING- II) & (COVERING- II) (PACKING- II) & (COVERING- II)	Matrix Logarithmic potential [12]	Matrix inversion $O(n^\omega)$	$O(n \log(n\mathcal{L}\tau) + \frac{n}{\epsilon^2})$ $O(n \log(n/\epsilon) + \frac{n}{\epsilon^2})$	Yes	No	Yes	Yes

*In fact, these algorithms find sparse solutions in the sense that the dependence of the size of the support of the dual solution on m is at most logarithmic. However, the dependence of the size of the support on the bit length $\mathcal{L}$ is not polynomial

solutions) in the sense described above. However, the overall running time of the matrix MWU algorithm is larger by a factor of (roughly) $\Omega(n^{3-\omega})$ than that of the logarithmic-potential algorithm, where ω is the exponent of matrix multiplication. Moreover, we cannot extend the matrix MWU algorithm to solve (PACKING- I)-(COVERING- I) (in particular, it seems tricky to bound the number of iterations).

4.3 General Framework for Packing-Covering SDPs

Given a pair of packing-covering SDPs (PACKING- I)-(COVERING- I) or (COVERING-II)-(PACKING- II), we consider the following general framework in which each constraint is assigned a weight reflecting how satisfied the constraint is given the current solution:

```
1  Initialize constraint weights
2  while the stopping criterion is not satisfied do
3  |  Form a "weighted average" of all the inequalities into a single inequality
4  |  /* If we maintain weights for primal ⟶ scalar version */
5  |  /* If we maintain weights for dual ⟶ matrix version */
6  |  Solve a fractional knapsack problem to determine the direction of the next
   |  update
7  |  Update the primal (or sometimes dual) variables in the chosen direction
8  |  Update the weights to reflect which constraints become more satisfied
9  |  /* Weights (essentially) ⟷ dual (or sometimes primal) variables */
10 end
```

Algorithm 1: A general framework for solving packing-covering SDPs

We obtain different algorithms depending on how the weights are defined. We write $a_i := A_i \bullet X \geq 0$. Since $a_{\max} := \max\{a_1, \ldots, a_m\}$ (resp., $a_{\min} := \min\{a_1, \ldots, a_m\}$) is *not a smooth* function (in X), it is more convenient to work with a *smooth approximation* of it, which is provided by the weighted average formed in step 3 in the framework. There are several ways to do this, for example:

- *Exponential averaging*: The weights are $\overline{p}_i := \frac{(1+\epsilon)^{a_i}}{\sum_{i'=1}^{m}(1+\epsilon)^{a_{i'}}}$ (resp., $\overline{p}_i := \frac{(1-\epsilon)^{a_i}}{\sum_{i'=1}^{m}(1-\epsilon)^{a_{i'}}}$). The following claim justifies the use of these sets of weights.

Lemma 4.1 *If* $a_{\max} \geq \frac{1+\epsilon}{\epsilon}\log_{1+\epsilon}\frac{m}{\epsilon}$ $\left(\text{resp., } a_{\min} \geq \frac{1}{\epsilon}\log_{\frac{1}{1-\epsilon}}\left(\frac{m\cdot a_{\max}}{\epsilon\cdot a_{\min}}\right)\right)$, *then*

$$\frac{a_{\max}}{1+\epsilon} \leq \sum_{i=1}^{m}\overline{p}_i a_i \leq a_{\max} \quad \left(\text{resp., } a_{\min} \leq \sum_{i=1}^{m}\overline{p}_i a_i \leq (1+\epsilon)a_{\min}\right).$$

- *Logarithmic potential averaging*: The weights are $\overline{p}_i = \frac{\epsilon}{m}\frac{\theta^*}{\theta^*-a_i}$ (resp., $\overline{p}_i = \frac{\epsilon}{m}\frac{\theta^*}{a_i-\theta^*}$), where θ^* is the minimizer (resp., maximizer) of the potential function

$$\Phi(\theta) = \ln\left(\theta \cdot \sqrt[\epsilon/m]{\prod_{i=1}^{m} \frac{1}{\theta - a_i}}\right) \quad \left(\text{resp.,} \Phi(\theta) = \ln\left(\theta \cdot \sqrt[\epsilon/m]{\prod_{i=1}^{m} (a_i - \theta)}\right)\right).$$

(It can be easily verified that $\sum_i \overline{p}_i = 1$.) The following claim justifies the use of these sets of weights.

Lemma 4.2

$$\frac{(1-\epsilon)a_{\max}}{1-\epsilon/m} \leq \sum_{i=1}^{m} \overline{p}_i a_i \leq a_{\max} \quad \left(\textit{resp.,} a_{\min} \leq \sum_{i=1}^{m} \overline{p}_i a_i \leq \frac{a_{\min}(1+\epsilon)}{1+\epsilon/m}\right).$$

4.4 Scalar Algorithms

4.4.1 Scalar MWU Algorithm for (PACKING-I)-(COVERING-I)

Given a normalized pair of packing-covering SDPs of type I (PACKING- I)-(COVERING- I), and a feasible primal solution X, we use the exponential weight $p_i := (1+\epsilon)^{A_i \bullet X}$, for $i \in [m]$. Averaging the inequalities with respect to the weights $\overline{p}_i := \frac{p_i}{\sum_i p_i}$, we arrive at the following problem:

$$\begin{aligned} & \max\ I \bullet X \\ \text{s.t.}\quad & \sum_i \overline{p}_i A_i \bullet X \leq 1, \quad \forall i \in [m] \\ & X \in \mathbb{R}^{n \times n},\ X \succeq 0. \end{aligned} \tag{4.8}$$

Letting $\overline{A} := \sum_i \overline{p}_i A_i$ and writing $X = \sum_{v \in B_n} \lambda_v v v^T$, where $B_n := \{v \in \mathbb{R}^n : \|v\| = 1\}$ and $\lambda_v \geq 0$ for all $v \in B_n$, we obtain the following (*infinite-dimensional*) *knapsack* problem

$$\begin{aligned} & \max \sum_{v \in B_n} \lambda_v \\ \text{s.t.}\quad & \sum_{v \in B_n} \lambda_v \overline{A} \bullet v v^T \leq 1, \quad \forall i \in [m] \\ & \lambda_v \geq 0, \quad \forall v \in B_n. \end{aligned} \tag{4.9}$$

An optimal solution is attained at a vector $v \in B_n$ which *minimizes* $v^T \overline{A} v$. This is the basis vector corresponding to $\lambda_{\min}(\overline{A})$.

Thus, using this set of weights in our general framework (Algorithm 1) yields the following procedure (for a vector $p \in \mathbb{R}^m$, we write $\overline{p}_i := \frac{p_i}{\sum_i p_i}$):

1 $t \leftarrow 0$; $X(0) \leftarrow 0$; $y(0) \leftarrow 0$; $M(0) \leftarrow 0$; $T \leftarrow \epsilon^{-2} \ln m$
2 **while** $M(t) < T$ **do**
3 $\quad p_i(t) = (1+\epsilon)^{A_i \bullet X(t)}$ /* Update the weights */
4 $\quad v(t) = \text{argmin}_{v:||v||=1} \sum_i \overline{p}_i(t) A_i \bullet vv^T$ /* Find an eigenvector corresponding to the smallest eigenvalue of the average inequality matrix */
5 $\quad \delta(t) = 1/\max_i A_i \bullet v(t)v(t)^T$ /* Define the update step size */
6 $\quad X(t+1) = X(t) + \delta(t)v(t)v(t)^T$; $y(t+1) \leftarrow y(t) + \delta(t)\overline{p}_i(t)$ /* Update the primal-dual solutions */
7 $\quad M(t+1) = \max_i A_i \bullet X(t+1)$ /* Compute the largest LHS */
8 $\quad t \leftarrow t+1$
9 **end**
10 **output** $(\hat{X}, \hat{y}) = \left(\frac{X(t)}{M(t)}, \frac{y(t)}{(1-1.5\epsilon)M(t)}\right)$

Algorithm 2: Scalar MWU algorithm for (PACKING- I)-(COVERING- I)

The stopping criterion is that the left-hand side (LHS) of at least one inequality in (PACKING- I) reaches some threshold $T := \epsilon^{-2} \ln m$, with respect to the current solution $X(t)$. The step size (step 5) is chosen such that in each iteration of the while-loop, this right-hand size increases by at least 1, thus guaranteeing termination in mT iterations.

Theorem 4.4 *Given a real $\epsilon \in (0, 1]$, Algorithm 2 outputs an ϵ-optimal solution for (*PACKING- I*)-(*COVERING- I*) in $O(m \log m/\epsilon^2)$ iterations, where each iteration requires an oracle call that computes an eigenvector corresponding to the minimum eigenvalue of a psd matrix.*

For a given matrix $M \in \mathbb{R}^{n \times n}$, computing $\lambda_{\min}(M)$ (almost) exactly requires $O(n^3)$ time via a full eigenvalue decomposition of the matrix. If M is psd, a faster approximation of $\lambda_{\min}(M)$ can be obtained (using Lanczos' algorithm with a random start) via the following result.

Theorem 4.5 ([19]) *Let $M \in \mathbb{S}^n_+$ be a psd matrix with N non-zeros and $\gamma \in (0, 1)$ be a given constant. Then, there is a randomized algorithm that computes, with high (i.e., $1 - o(1)$) probability a unit vector $v \in \mathbb{R}^n$ such that $v^T M v \geq (1-\gamma)\lambda_{\max}(M)$. The algorithm takes $O\left(\frac{\log n}{\sqrt{\gamma}}\right)$ iterations, each requiring $O(N)$ arithmetic operations.*

By applying the lemma to $(\overline{A})^{-1}$, we can approximate $\lambda_{\min}(\overline{A})$ in $\tilde{O}(n^\omega)$ time.

4.4.2 Scalar Logarithmic Potential Algorithm For (PACKING-I)–(COVERING-I)

Given a normalized pair of packing-covering SDPs of type I (PACKING- I)-(COVERING- I) and a feasible primal solution X, we use the logarithmic-potential weights $\overline{p}_i = \frac{\epsilon}{m} \frac{\theta^*}{\theta^* - A_i \bullet X}$ for $i \in [m]$. Averaging the inequalities with respect to this

set of weights, we arrive at the knapsack problem (4.9). This gives rise to the following procedure:

$s \leftarrow 0$; $\varepsilon_0 \leftarrow \frac{1}{4}$; $t \leftarrow 0$; $v(0) \leftarrow 1$; $X(0) \leftarrow \mathbf{1}_i \mathbf{1}_i^T$ (for an arbitrary $i \in [n]$)
while $\varepsilon_s > \epsilon$ **do**
  $\delta_s \leftarrow \frac{\varepsilon_s^3}{32m}$ **while** $v(t) > \varepsilon_s$ **do**
    $\theta(t) \leftarrow \theta^*(t)^{\delta_s}$, where $\theta^*(t)$ is the smallest positive root of the equation
    $$\frac{\varepsilon_s \theta}{m} \sum_i \frac{1}{\theta - A_i \bullet X(t)} = 1$$
    $y(t) \leftarrow \overline{p}(t)$, where $\overline{p}_i(t) := \frac{\varepsilon_s \theta(t)}{m} \frac{1}{\theta(t) - A_i \bullet X(t)}$, for $i \in [n]$ /* Set the dual solution */ $v(t) = \operatorname{argmin}_{v:||v||=1} \overline{A}(t) \bullet vv^T$, where $\overline{A}(t) := \sum_i \overline{p}_i(t) A_i$ /* Find the eigenvector corresponding to the smallest eigenvalue of the average inequality matrix */
    $$v(t+1) \leftarrow \frac{\overline{A}(t) \bullet X(t) - \overline{A}(t) \bullet v(t)v(t)^T}{\overline{A}(t) \bullet X(t) + \overline{A}(t) \bullet v(t)v(t)^T}$$ /* Compute the error */
    $$\tau(t+1) \leftarrow \frac{\varepsilon_s \theta(t) v(t+1)}{4m(\overline{A}(t) \bullet X(t) + \overline{A}(t) \bullet v(t)v(t)^T)}$$ /* Compute the step size */ $X(t+1) \leftarrow (1 - \tau(t+1))X(t) + \tau(t+1)v(t)v(t)^T$ /* Update the primal solution */ $t \leftarrow t+1$
  end
  $\varepsilon_{s+1} \leftarrow \frac{\varepsilon_s}{2}$
  $s \leftarrow s+1$
end
output $(\hat{X}, \hat{y}) = \left(\frac{X(t-1)}{(1-\varepsilon_{s-1}/m)\theta(t-1)}, \frac{(1+\varepsilon_{s-1})y(t-1)}{(1-\varepsilon_{s-1})(1-2\varepsilon_{s-1})\theta(t-1)} \right)$

Algorithm 3: Scalar logarithmic-potential algorithm for (PACKING- I)-(COVERING- I)

In the above, for given numbers $x \in \mathbb{R}_+$ and $\delta \in (0, 1)$, we define the δ-(upper) approximation x^δ of x to be a number satisfying: $x \le x^\delta < (1+\delta)x$.

Theorem 4.6 *Given $\epsilon \in (0, 1]$, Algorithm 3 outputs an ϵ-optimal solution for (COVERING- I)-(PACKING- I) in $O(m \log \psi + m/\epsilon^2)$ iterations, where $\psi := \frac{\lambda_{\max}(\overline{A}(0))}{\lambda_{\min}(\overline{A}(0))}$ and each iteration requires an oracle call that computes an eigenvector corresponding to the minimum eigenvalue of a psd matrix.*

4.5 Matrix Algorithms

4.5.1 *Matrix MWU Algorithm For (COVERING-II)-(PACKING-II)*

Let $F(y) := \sum_{i=1}^{m} y_i A_i$. Then, we can rewrite the normalized version of (PACKING-II) as follows:

$$\begin{aligned} & z_I^* = \max\ \mathbf{1}^T y && \text{(PACKING- II)} \\ \text{s.t.}\ & \lambda_j(F(y)) \le 1, \ \ \forall j \in [n] \\ & y \in \mathbb{R}^m,\ y \ge 0. \end{aligned}$$

Averaging the inequalities with respect to the weights $\overline{p}_j := \frac{p_j}{\sum_j p_j}$, where $p_j := (1+\epsilon)^{\lambda_j(F(y))}$, we get

$$\begin{aligned} & \max\ \mathbf{1}^T y \\ \text{s.t.}\ & \sum_j \overline{p}_j \lambda_j(F(y)) \le 1, \ \ \forall j \in [n] \\ & y \in \mathbb{R}^m,\ y \ge 0. \end{aligned}$$

Using the *eigenvalue decomposition*: $F(y) = U \Lambda U^T$, where Λ is the diagonal matrix containing the eigenvalues of $F(y)$ and $UU^T = I$, and letting

$$\overline{P} := U \begin{bmatrix} \overline{p}_1 & 0 & \cdots & 0 \\ 0 & \overline{p}_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \overline{p}_n \end{bmatrix} U^T = \frac{(1+\epsilon)^{F(y)}}{\mathrm{Tr}((1+\epsilon)^{F(y)})},$$

we obtain the following knapsack problem:

$$\begin{aligned} & \max\ \mathbf{1}^T y \\ \text{s.t.}\ & \sum_i (\overline{P} \bullet A_i) y_i \le 1, \ \ \forall j \in [n] \\ & y \in \mathbb{R}^m,\ y \ge 0. \end{aligned}$$

An optimal solution is attained at the basis vector $y = \mathbf{1}_i \in \mathbb{R}^m_+$ that minimizes $\overline{P} \bullet A_i$. This gives rise to the following matrix MWU algorithm:

Theorem 4.7 *Given an real $\epsilon \in (0, 1]$, Algorithm 2 outputs an ϵ-optimal solution for (COVERING- II)-(PACKING- II) in $O(n \log n/\epsilon^2)$ iterations, where each iteration requires matrix exponential computation, two oracle calls that computes the max-*

$t \leftarrow 0$; $y(0) \leftarrow 0$; $X(0) \leftarrow 0$; $M(0) \leftarrow 0$; $T \leftarrow \epsilon^{-2} \ln n$
while $M(t) < T$ **do**
  $P(t) = (1+\epsilon)^{\sum_{i=1}^{m} y_i(t) A_i}$ /* Update the weight matrix by exponentiation */
  $i(t) \leftarrow \text{argmin}_i \, A_i \bullet P(t)$ $\delta(t) \leftarrow 1/\lambda_{\max}(A_{i(t)})$ /* Define the update step size */
  $X(t+1) \leftarrow X(t) + \frac{\delta(t) P(t)}{I \bullet P(t)}$; $y(t+1) \leftarrow y(t) + \delta(t) \mathbf{1}_{i(t)}$ /* Update the primal-dual solution */
  $M(t+1) \leftarrow \lambda_{\max}(\sum_i y_i(t) A_i)$ /* Compute the largest eigenvalue of LHS of dual */
  $t \leftarrow t+1$
end
$L(t) \leftarrow \min_i \, A_i \bullet X(t)$
output $(\hat{X}, \hat{y}) = \left(\frac{X(t)}{L(t)}, \frac{y(t)}{M(t)} \right)$

Algorithm 4: Matrix MWU algorithm for (PACKING- II)-(COVERING- II)

imum eigenvalue of a psd matrix, and a single oracle call to the minimization in step 4.

The most demanding step in the above algorithm is the matrix exponential computation, which can be done in $O(n^3)$ time via a complete eigenvalue decomposition. A more efficient approximation, particularly when the matrices A_i are sparse, can be obtained via the following result.

Theorem 4.8 ([26]) *There is an algorithm for approximating the matrix exponential e^F in time $O(n^2 r \log^3 \frac{1}{\epsilon})$, where r denotes the number of non-zeros in $F \in \mathbb{S}^n$, and ϵ is the approximation accuracy.*

We remark that a matrix MWU algorithm and a theorem similar to Algorithm 4 and Theorem 4.7 for (PACKING- I)-(COVERING- I) have not yet been discovered and are left as open problems.

4.5.2 Matrix Logarithmic Potential Algorithm For (PACKING-I)-(COVERING-I)

Let $F(y) := \sum_{i=1}^{m} y_i A_i$. Then, we can rewrite the normalized version of (COVERING-I) as

$$\begin{aligned} & z_I^* = \min \mathbf{1}^T y && \text{(PACKING- II)} \\ \text{s.t.} \quad & \lambda_j(F(y)) \geq 1, \quad \forall j \in [n] \\ & y \in \mathbb{R}^m, \; y \geq 0. \end{aligned}$$

Averaging the inequalities with respect to the weights $\overline{p}_j := \frac{\epsilon}{n} \frac{\theta^*}{\lambda_j(F(y)) - \theta^*}$, we get

$$\begin{aligned} & \min \ \mathbf{1}^T y \\ \text{s.t.} \quad & \sum_j \overline{p}_j \lambda_j(F(y)) \geq 1, \quad \forall j \in [n] \\ & y \in \mathbb{R}^m, \ y \geq 0. \end{aligned}$$

Using the *eigenvalue decomposition*: $F(y) = U\Lambda U^T$, where Λ is the diagonal matrix containing the eigenvalues of $F(y)$ and $UU^T = I$, and letting

$$\overline{P} := U \begin{bmatrix} \overline{p}_1 & 0 & \cdots & 0 \\ 0 & \overline{p}_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \overline{p}_n \end{bmatrix} U^T = \frac{\epsilon\theta^*}{n}(F(y) - \theta^* I)^{-1},$$

we obtain the following knapsack problem:

$$\begin{aligned} & \min \quad \mathbf{1}^T y \\ \text{s.t.} \quad & \sum_i (\overline{P} \bullet A_i) y_i \geq 1, \forall j \in [n] \\ & y \in \mathbb{R}^m, \ y \geq 0. \end{aligned}$$

An optimal solution is attained at the basis vector $y = \mathbf{1}_i \in \mathbb{R}^m_+$ that maximizes $\overline{P} \bullet A_i$. This gives rise to the following matrix logarithmic-potential algorithm:

1 $s \leftarrow 0$; $\varepsilon_0 \leftarrow \frac{1}{2}$; $t \leftarrow 0$; $v(0) \leftarrow 1$; $y(0) \leftarrow \frac{1}{r}\sum_{i=1}^r \mathbf{1}_i$
2 **while** $\varepsilon_s > \epsilon$ **do**
3 $\quad \delta_s \leftarrow \frac{\varepsilon_s^3}{32n}$
4 $\quad$ **while** $v(t) > \varepsilon_s$ **do**
5 $\quad\quad \theta(t) \leftarrow \theta^*(t)_{\delta_s}$, where $\theta^*(t)$ is the smallest positive root of the equation $\frac{\varepsilon_s\theta}{n}\mathrm{Tr}(F(y(t)) - \theta I)^{-1} = 1$
$\quad\quad X(t) \leftarrow \frac{\varepsilon_s\theta(t)}{n}(F(y(t)) - \theta(t)I)^{-1}$ /* Set the primal solution */ $i(t) \leftarrow \mathrm{argmax}_i \ A_i \bullet X(t)$ /* Call the maximization oracle */
6 $\quad\quad v(t+1) \leftarrow \frac{X(t) \bullet A_{i(t)} - X(t) \bullet F(y(t))}{X(t) \bullet A_{i(t)} + X(t) \bullet F(y(t))}$ /* Compute the error */
7 $\quad\quad \tau(t+1) \leftarrow \frac{\varepsilon_s\theta(t)v(t+1)}{4n(X(t) \bullet A_{i(t)} + X(t) \bullet F(y(t)))}$ /* Compute the step size */
$\quad\quad y(t+1) \leftarrow (1 - \tau(t+1))y(t) + \tau(t+1)\mathbf{1}_{i(t)}$ /* Update the dual solution */ $t \leftarrow t+1$
8 $\quad$ **end**
9 $\quad \varepsilon_{s+1} \leftarrow \frac{\varepsilon_s}{2}$
10 $\quad s \leftarrow s+1$
11 **end**
12 **output** $(\hat{X}, \hat{y}) = \left(\frac{(1-\varepsilon_{s-1})X(t-1)}{(1+\varepsilon_{s-1})^2\theta(t-1))}, \frac{y(t-1)}{\theta(t-1)}\right)$

Algorithm 5: Matrix logarithmic-potential algorithm for (PACKING- I)-(COVERING- I)

The most demanding steps are the computation of $\theta(t)$ and $X(t)$ in steps 5 and 5, respectively. Computing $\theta(t)$ can be done via binary search over a region determined by repeated matrix multiplications and approximate minimum eigenvalue computa-

tion (cf. Theorem 4.5). Once $\theta(t)$ is determined, computing $X(t)$ requires a single matrix inversion. The overall running time per iteration is $\tilde{O}(n^{\omega})$ plus the time needed by the maximization oracle in step 5.

Theorem 4.9 *Given $\epsilon \in (0, 1]$, Algorithm 5 outputs an ϵ-optimal solution for (*COVERING- I*)-(*PACKING- I*) in $O(n \log \psi + \frac{n}{\epsilon^2})$ iterations, where $\psi := \frac{r \cdot \max_i \lambda_{\max}(A_i)}{\lambda_{\min}(\hat{A})}$ and each iteration requires $O(\log \frac{n}{\epsilon})$ matrix multiplications and a single oracle call to the maximization in step 5.*

4.5.3 Matrix Logarithmic Potential Algorithm For (PACKING-II)-(COVERING-II)

A symmetric version of Algorithm 5 for (PACKING- II)-(COVERING- II) can be given as follows:

$s \leftarrow 0$; $\varepsilon_0 \leftarrow \frac{1}{4}$; $t \leftarrow 0$; $\nu(0) \leftarrow 1$; $y(0) \leftarrow \mathbf{1}_i$ (for an arbitrary $i \in [m]$)
while $\varepsilon_s > \epsilon$ **do**
  $\delta_s \leftarrow \frac{\varepsilon_s^3}{32n}$ **while** $\nu(t) > \varepsilon_s$ **do**
    $\theta(t) \leftarrow \theta^*(t)^{\delta_s}$, where $\theta^*(t)$ is the smallest positive root of the equation $\frac{\varepsilon_s \theta}{n} \mathrm{Tr}(\theta I - F(y(t)))^{-1} = 1$ $X(t) \leftarrow \frac{\varepsilon_s \theta(t)}{n} (\theta(t) I - F(y(t)))^{-1}$ /* Set the primal solution */ $i(t) \leftarrow \mathrm{argmin}_i A_i \bullet X(t)$ /* Call the minimization oracle */
    $\nu(t+1) \leftarrow \frac{X(t) \bullet F(y(t)) - X(t) \bullet A_{i(t)}}{X(t) \bullet A_{i(t)} + X(t) \bullet F(y(t))}$ /* Compute the error */
    $\tau(t+1) \leftarrow \frac{\varepsilon_s \theta(t) \nu(t+1)}{4n(X(t) \bullet A_{i(t)} + X(t) \bullet F(y(t)))}$ /* Compute the step size */
    $y(t+1) \leftarrow (1 - \tau(t+1)) y(t) + \tau(t+1) \mathbf{1}_{i(t)}$ /* Update the dual solution */
    $t \leftarrow t + 1$
  end
  $\varepsilon_{s+1} \leftarrow \frac{\varepsilon_s}{2}$
  $s \leftarrow s + 1$
end
output $(\hat{X}, \hat{y}) = \left(\frac{(1+\varepsilon_{s-1}) X(t-1)}{(1-2\varepsilon_{s-1})^2 \theta(t-1)}, \frac{y(t-1)}{\theta(t-1)} \right)$

Algorithm 6: Materix logarithmic-potential algorithm for (PACKING- II)-(COVERING- II)

Theorem 4.10 *Given $\epsilon \in (0, 1]$, Algorithm 6 outputs an ϵ-optimal solution for (*PACKING- II*)-(*COVERING- II*) in $O(n \log \psi + \frac{n}{\epsilon^2})$ iterations, where $\psi := O(\log \frac{n}{\epsilon})$ and each iteration requires $O(\log \frac{n}{\epsilon})$ matrix inversions and a single oracle call to the minimization in step 4.*

Acknowledgements We thank Waleed Najy for many helpful discussions on this topic. This work was partially supported by JST CREST JPMJCR1402 and Grants-in-Aid for Scientific Research. The research of the first author was partially supported by Abu Dhabi Education & Knowledge – Abu Dhabi Award for Research Excellence (AARE18-152).

References

1. F. Alizadeh, Interior point methods in semidefinite programming with applications to combinatorial optimization. SIAM J. Optim. **5**(1), 13–51 (1995)
2. Z. Allen-Zhu, Y.T. Lee, L. Orecchia, Using optimization to obtain a width-independent, parallel, simpler, and faster positive sdp solver, in *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16 (Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2016), pp. 1824–1831
3. S. Arora, E. Hazan, S. Kale, Fast algorithms for approximate semidefinite programming using the multiplicative weights update method (2005), pp. 339–348
4. S. Arora, E. Hazan, S. Kale, The multiplicative weights update method: a meta-algorithm and applications. Theory Comput. **8**(1), 121–164 (2012)
5. S. Arora, S. Kale, A combinatorial, primal-dual approach to semidefinite programs (2007), pp. 227–236
6. S. Arora, S. Kale. A combinatorial, primal-dual approach to semidefinite programs. J. ACM **63**(2), 12:1–12:35 (2016)
7. K. Elbassioni, K. Makino, Oracle-based primal-dual algorithms for packing and covering semidefinite programs, in *27th Annual European Symposium on Algorithms, ESA 2019, September 9–11, 2019, Munich/Garching, Germany* (2019), pp. 43:1–43:15
8. D. Garber, E. Hazan, Sublinear time algorithms for approximate semidefinite programming. Math. Program. **158**(1–2), 329–361 (2016)
9. N. Garg, J. Könemann, Faster and simpler algorithms for multicommodity flow and other fractional packing problems. SIAM J. Comput. **37**(2), 630–652 (2007)
10. M.X. Goemans, D.P. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM **42**(6), 1115–1145 (1995)
11. M.D. Grigoriadis, L.G. Khachiyan, A sublinear-time randomized approximation algorithm for matrix games. Operat. Res. Lett. **18**(2), 53–58 (1995)
12. M.D. Grigoriadis, L.G. Khachiyan, L. Porkolab, J. Villavicencio, Approximate max-min resource sharing for structured concave optimization. SIAM J. Optim. **41**, 1081–1091 (2001)
13. G. Iyengar, D. J. Phillips, C. Stein, Approximation algorithms for semidefinite packing problems with applications to maxcut and graph coloring, in *Integer Programming and Combinatorial Optimization (IPCO)*, eds. by M. Jünger, V. Kaibel (Berlin, Heidelberg, 2005), pp. 152–166
14. G. Iyengar, D.J. Phillips, C. Stein, Feasible and accurate algorithms for covering semidefinite programs, in *Algorithm Theory—SWAT 2010*, ed. by H. Kaplan (Berlin, Heidelberg, 2010), pp. 150–162.
15. G. Iyengar, D.J. Phillips, C. Stein, Approximating semidefinite packing programs. SIAM J. Optim. **21**(1), 231–268 (2011)
16. R. Jain, P. Yao, A parallel approximation algorithm for positive semidefinite programming. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011* (2011), pp. 463–471
17. R. Jain, P. Yao, A parallel approximation algorithm for mixed packing and covering semidefinite programs. CoRR (2012). arXiv:abs/1201.6090
18. P. Klein, H.-I. Lu, Efficient approximation algorithms for semidefinite programs arising from max cut and coloring, in *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96 (ACM, New York, NY, USA, 1996), pp. 338–347

19. Z. Leyk, H. Woźniakowski, Estimating a largest eigenvector by lanczos and polynomial algorithms with a random start. Numer. Linear Algebra Appl. **5**(3), 147–164 (1999)
20. Y. Nesterov, Smooth minimization of non-smooth functions. Math. Program. **103**(1), 127–152 (2005)
21. Y. Nesterov, Smoothing technique and its applications in semidefinite optimization. Math. Program. **110**(2), 245–259 (2007)
22. Y. Nesterov, A. Nemirovskii, Interior-Point Polynomial Algorithms in Convex Programming. Society for Industrial and Applied Mathematics (1994)
23. R. Peng, K. Tangwongsan, Faster and simpler width-independent parallel algorithms for positive semidefinite programming, in *Proceedings of the Twenty-fourth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '12 (ACM, New York, NY, USA, 2012), pp. 101–108
24. R. Peng, K. Tangwongsan, P. Zhang, Faster and simpler width-independent parallel algorithms for positive semidefinite programming. CoRR (2016). arXiv:abs/1201.5135
25. S.A. Plotkin, D.B. Shmoys, É. Tardos, Fast approximation algorithms for fractional packing and covering problems (1991), pp. 495–504
26. J. van den Eshof, M. Hochbruck, Preconditioning lanczos approximations to the matrix exponential. SIAM J. Sci. Comput. **27**(4), 1438–1457 (2006)
27. L. Vandenberghe, S. Boyd, Semidefinite programming. SIAM Rev. **38**(1), 49–95 (1996)
28. E.P. Xing, M.I. Jordan, S.J. Russell, A.Y. Ng, Distance metric learning with application to clustering with side-information, in *Advances in Neural Information Processing Systems 15*, ed. by S. Becker, S. Thrun, K. Obermayer (MIT Press, Cambridge, 2003), pp. 521–528
29. N.E. Young, Sequential and parallel algorithms for mixed packing and covering (2001), pp. 38–546