# A parallel approximation algorithm for mixed packing and covering semidefinite programs

Rahul Jain*
National U. Singapore

Penghui Yao†
National U. Singapore

January 28, 2012

**Abstract**

We present a parallel approximation algorithm for a class of *mixed packing and covering* semidefinite programs which generalize on the class of positive semidefinite programs as considered by Jain and Yao [6]. As a corollary we get a faster approximation algorithm for positive semidefinite programs with better dependence of the parallel running time on the approximation factor, as compared to that of Jain and Yao [6]. Our algorithm and analysis is on similar lines as that of Young [10] who considered analogous linear programs.

## 1 Introduction

Fast parallel approximation algorithms for semidefinite programs have been the focus of study of many recent works (e.g. [1, 2, 7, 5, 4, 3]) and have resulted in many interesting applications including the well known QIP = PSPACE [3] result. In many of the previous works, the running time of the algorithms had polylog dependence on the size of the input program but in addition also had polynomial dependence of some *width* parameter (which varied for different algorithms). Sometimes (for specific instances of input programs) the width parameter could be as large as the size of the program making it an important bottleneck. Recently Jain and Yao [6] presented a fast parallel approximation algorithm for an important subclass of semidefinite programs, called as *positive semidefinite programs*, and their algorithm had no dependence on any width parameter. Their algorithm was inspired by an algorithm by Luby and Nisan [8] for *positive linear programs*. In this work we consider a more general *mixed packing and covering* optimization problem. We first consider the following feasibility task **Q1**.

**Q1:** Given $n \times n$ positive semidefinite matrices $P_1, \ldots, P_m, P$ and non-negative diagonal matrices $C_1, \ldots, C_m, C$ and $\varepsilon \in (0,1)$, find an *$\varepsilon$-approximate feasible* vector $x \geq 0$ such that (while comparing matrices we let $\geq, \leq$ represent the Löwner order),

$$\sum_{i=1}^{m} x_i P_i \leq (1+\varepsilon)P \quad \text{and} \quad \sum_{i=1}^{m} x_i C_i \geq C$$

*Centre for Quantum Technologies and Department of Computer Science, National University of Singapore, Block S15, 3 Science Drive 2, Singapore 11754. Email: rahul@comp.nus.edu.sg.
†Centre for Quantum Technologies and Department of Computer Science, National University of Singapore, Block S15, 3 Science Drive 2, Singapore 11754. Email: pyao@nus.edu.sg.

or show that the following is infeasible for all $x \geq 0$

$$\sum_{i=1}^{m} x_i P_i \leq P \quad \text{and} \quad \sum_{i=1}^{m} x_i C_i \geq C \ .$$

We present an algorithm for **Q1** running in parallel time $\mathrm{polylog}(n, m) \cdot \frac{1}{\varepsilon^4} \cdot \log \frac{1}{\varepsilon}$. Using this and standard binary search, a multiplicative $(1-\varepsilon)$ approximate solution can be obtained for the following optimization task **Q2** in parallel time $\mathrm{polylog}(n, m) \cdot \frac{1}{\varepsilon^4} \cdot \log \frac{1}{\varepsilon}$.

**Q2:** Given $n \times n$ positive semidefinite matrices $P_1, \ldots, P_m, P$ and non-negative diagonal matrices $C_1, \ldots, C_m, C$,

$$\begin{aligned}
\text{maximize:} \quad & \gamma \\
\text{subject to:} \quad & \sum_{i=1}^{m} x_i P_i \leq P \\
& \sum_{i=1}^{m} x_i C_i \geq \gamma C \\
& \forall i \in [m] : x_i \geq 0.
\end{aligned}$$

The following special case of **Q2** is referred to as a positive semidefinite program.

**Q3:** Given $n \times n$ positive semidefinite matrices $P_1, \ldots, P_m, P$ and non-negative scalars $c_1, \ldots, c_m$,

$$\begin{aligned}
\text{maximize:} \quad & \sum_{i=1}^{m} x_i c_i \\
\text{subject to:} \quad & \sum_{i=1}^{m} x_i P_i \leq P \\
& \forall i \in [m] : x_i \geq 0.
\end{aligned}$$

Our algorithm for **Q1** and its analysis is on similar lines as the algorithm and analysis of Young [10] who had considered analogous questions for linear programs. As a corollary we get an algorithm for approximating positive semidefinite programs (**Q3**) with better dependence of the parallel running time on $\varepsilon$ as compared to that of Jain and Yao [6] (and arguably with simpler analysis). Very recently, in an independent work, Peng and Tangwongsan [9] also presented a fast parallel algorithm for positive semidefinite programs. Their work is also inspired by Young [10].

## 2 Algorithm and analysis

We mention without elaborating that using standard arguments the feasibility question **Q1** can be easily transformed, in parallel time $\mathrm{polylog}(mn)$, to the special case when $P$ and $C$ are identity matrices and we consider this special case from now on. Our algorithm is presented in Figure 1 .

### Idea of the algorithm

The algorithm starts with an initial value for $x$ such that $\sum_{i=1}^{m} x_i P_i \leq \mathbb{1}$. It makes increments to the vector $x$ such that with each increment, the increase in $\|\sum_{i=1}^{m} x_i P_i\|$ is not more than

$(1 + \mathcal{O}(\varepsilon))$ times the increase in the minimum eigenvalue of $\sum_{i=1}^m x_i C_i$. We argue that it is always possible to increment $x$ in this manner if the input instance is feasible, hence the algorithm outputs infeasible if it cannot find such an increment to $x$. The algorithm stops when the minimum eigenvalue of $\sum_{i=1}^m x_i C_i$ has exceeded 1. Due to our condition on the increments, at the end of the algorithm we also have $\sum_{i=1}^m x_i P_i \leq (1 + \mathcal{O}(\varepsilon))\mathbb{1}$. We obtain handle on the largest and smallest eigenvalues of concerned matrices via their *soft* versions, which are more easily handled functions of those matrices (see definition in the next section).

---

**Input :** $n \times n$ positive semidefinite matrices $P_1, \ldots, P_m$, non-negative diagonal matrices $C_1, \ldots, C_m$, and error parameter $\varepsilon \in (0, 1)$.

**Output :** Either infeasible, which means there is no $x \geq 0$ such that ($\mathbb{1}$ is the identity matrix),

$$\sum_{i=1}^m x_i P_i \leq \mathbb{1} \quad \text{and} \quad \sum_{i=1}^m x_i C_i \geq \mathbb{1} \ .$$

OR an $x^* \geq 0$ such that

$$\sum_{i=1}^m x_i^* P_i \leq (1 + 9\varepsilon)\mathbb{1} \quad \text{and} \quad \sum_{i=1}^m x_i^* C_i \geq \mathbb{1} \ .$$

1. Set $x_j = \frac{1}{m\|P_j\|}$.
2. Set $N = \frac{1}{\varepsilon}\left(\|\sum_{i=1}^m x_i P_i\| + 2\ln n + \ln m\right)$.
3. While $\lambda_{\min}(\sum_{i=1}^m x_i C_i) < N$ ($\lambda_{\min}$ represents minimum eigenvalue), do
   (a) Set
   $$\text{local}_j(x) = \frac{\mathrm{Tr}(\exp(\sum_{i=1}^m x_i P_i) \cdot P_j)}{\mathrm{Tr}(\exp(-\sum_{i=1}^m x_i C_i) \cdot C_j)} \quad \text{and}$$
   $$\text{global}(x) = \frac{\mathrm{Tr}\exp(\sum_{i=1}^m x_i P_i)}{\mathrm{Tr}(\exp(-\sum_{i=1}^m x_i C_i))}.$$
   (b) If $g$ is not yet set or $\min_j\{\text{local}_j(x)\} > g(1+\varepsilon)$, set $g = \text{global}(x)$.
   (c) If $\min_j\{\text{local}_j(x)\} > \text{global}(x)$ , return infeasible.
   (d) For all $j \in [m]$, set $C_j = \Pi_j \cdot C_j \cdot \Pi_j$, where $\Pi_j$ is the projection onto the eigenspace of $\sum_{i=1}^m x_i C_i$ with eigenvalues at most $N$.
   (e) Choose increment vector $\alpha \geq 0$ and scalar $\delta > 0$ such that
   $$\forall j: \ \alpha_j = x_j \delta \text{ if } \text{local}_j(x) \leq g(1+\varepsilon), \text{ else } \alpha_j = 0, \text{ and}$$
   $$\max\{\left\|\sum_{i=1}^m \alpha_i P_i\right\|, \left\|\sum_{i=1}^m \alpha_i C_i\right\|\} = \varepsilon.$$
   (f) Set $x = x + \alpha$.
4. Return $x^* = x/N$.

---

Figure 1: Algorithm

## Correctness analysis

We begin with the definitions of soft maximum and minimum eigenvalues of a positive semidefinite matrix $A$. They are inspired by analogous definitions made in Young [10] in the context of vectors.

**Definition 1.** For positive semidefinite matrix $A$, define

$$\mathrm{Imax}(A) \stackrel{\mathrm{def}}{=} \ln \mathrm{Tr} \exp(A),$$

and

$$\mathrm{Imin}(A) \stackrel{\mathrm{def}}{=} -\ln \mathrm{Tr} \exp(-A).$$

Note that $\mathrm{Imax}(A) \geq \|A\|$ and $\mathrm{Imin}(A) \leq \lambda_{\min}(A)$, where $\lambda_{\min}(A)$ is the minimum eigenvalue of $A$.

We show the following lemma in the appendix, which shows that if a small increment is made in the vector $x$, then changes in $\mathrm{Imax}(\sum_{j=1}^m x_j A_j)$ and $\mathrm{Imin}(\sum_{j=1}^m x_j A_j)$ can be bounded appropriately.

**Lemma 2.** *Let $A_1, \ldots, A_m$ be positive semidefinite matrices and let $x \geq 0, \alpha \geq 0$ be vectors in $\mathbb{R}^m$. If $\|\sum_{i=1}^m \alpha_i A_i\| \leq \varepsilon \leq 1$, then*

$$\mathrm{Imax}(\sum_{j=1}^m (x_j+\alpha_j)A_j) - \mathrm{Imax}(\sum_{j=1}^m x_j A_j) \leq \frac{(1+\varepsilon)}{\mathrm{Tr}(\exp(\sum_{i=1}^m x_i A_i))} \sum_{j=1}^m \alpha_j \,\mathrm{Tr}(\exp(\sum_{i=1}^m x_i A_i) A_j),$$

*and*

$$\mathrm{Imin}(\sum_{j=1}^m (x_j+\alpha_j)A_j) - \mathrm{Imin}(\sum_{j=1}^m x_j A_j) \geq \frac{(1-\varepsilon/2)}{\mathrm{Tr}(\exp(-\sum_{i=1}^m x_i A_i))} \sum_{j=1}^m \alpha_j \,\mathrm{Tr}(\exp(-\sum_{i=1}^m x_i A_i) A_j).$$

**Lemma 3.** *At step 3(e) of the algorithm, for any $j$ with $\alpha_j > 0$ we have,*

$$\frac{\mathrm{Tr}(\exp(\sum_{i=1}^m x_i P_i)\cdot P_j)}{\mathrm{Tr}(\exp(\sum_{i=1}^m x_i P_i))} \leq (1+\varepsilon) \frac{\mathrm{Tr}(\exp(-\sum_{i=1}^m x_i C_i)\cdot C_j)}{\mathrm{Tr}(\exp(-\sum_{i=1}^m x_i C_i))}.$$

*Proof.* Consider any execution of step 3(e) of the algorithm. Fix $j$ such $\alpha_j > 0$. Note that,

$$\frac{\mathrm{local}_j(x)}{\mathrm{global}(x)} = \frac{\mathrm{Tr}(\exp(\sum_{i=1}^m x_i P_i)\cdot P_j)\cdot \mathrm{Tr}(\exp(-\sum_{i=1}^m x_i C_i))}{\mathrm{Tr}(\exp(\sum_{i=1}^m x_i P_i))\cdot \mathrm{Tr}(\exp(-\sum_{i=1}^m x_i C_i)\cdot C_j)}.$$

We will show that $\mathrm{global}(x) \geq g$ throughout the algorithm and this will show the desired since that $\mathrm{local}_j(x) \leq (1+\varepsilon)g \leq (1+\varepsilon)\mathrm{global}(x)$.

At step 3(b) of the algorithm, $g$ can be equal to $\mathrm{global}(x)$. Since $x$ never decreases during the algorithm, at step 3(a), $\mathrm{global}(x)$ can only increase. At step 3(d), the modification of $C_j$s only decreases $\mathrm{Tr}(\exp(-\sum_{i=1}^m x_i C_i))$ and hence again $\mathrm{global}(x)$ can only increase. □

**Lemma 4.** *For each increment of $x$ at step 3(f) of the algorithm,*

$$\mathrm{Imax}(\sum_{j=1}^m (x_j+\alpha_j)P_j) - \mathrm{Imax}(\sum_{j=1}^m x_j P_j) \leq (1+\varepsilon)^3 \left( \mathrm{Imin}(\sum_{j=1}^m (x_j+\alpha_j)C_j) - \mathrm{Imin}(\sum_{j=1}^m x_j C_j) \right).$$

*Proof.* Consider,

$$
\begin{aligned}
&\operatorname{Imax}(\sum_{j=1}^{m}(x_j+\alpha_j)P_j) - \operatorname{Imax}(\sum_{j=1}^{m} x_j P_j) \\
&\le \frac{(1+\varepsilon)}{\operatorname{Tr}(\exp(\sum_{i=1}^{m} x_i P_i))} \sum_{j=1}^{m} \alpha_j \operatorname{Tr}(\exp(\sum_{i=1}^{m} x_i P_i) P_j) \quad \text{(from Lemma 2)} \\
&\le \frac{(1+\varepsilon)^2}{\operatorname{Tr}(\exp(-\sum_{i=1}^{m} x_i C_i))} \sum_{j=1}^{m} \alpha_j \operatorname{Tr}(\exp(-\sum_{i=1}^{m} x_i C_i) C_j) \quad \text{(from Lemma 3 and step 3(e) of the algorithm)} \\
&\le \frac{(1+\varepsilon)^2}{1-\varepsilon/2} \left( \operatorname{Imin}(\sum_{j=1}^{m}(x_j+\alpha_j)C_j) - \operatorname{Imin}(\sum_{j=1}^{m} x_j C_j) \right) \quad \text{(from Lemma 2).}
\end{aligned}
$$

This shows the desired. □

**Lemma 5.** *If the input instance $P_1, \ldots, P_m, C_1, \ldots, C_m$ is feasible, that is there exists vector $y \in \mathbb{R}^m$ such that*

$$
\sum_{i=1}^{m} y_i P_i \le \mathbb{1} \quad \textit{and} \quad \sum_{i=1}^{m} y_i C_i \ge \mathbb{1} \ ,
$$

*then always at step 3(c) of the algorithm, $\min_j\{\mathrm{local}_j(x)\} \le \mathrm{global}(x)$. Hence the algorithm will return some $x^*$.*

*If the algorithm outputs* infeasible*, then the input instance is not feasible.*

*Proof.* Consider some execution of step 3(c) of the algorithm. Let $C'_1, \ldots, C'_m$ be the current values of $C_1, \ldots, C_m$. Note that if the input is feasible with vector $y$, then we will also have

$$
\frac{\operatorname{Tr}(\exp(\sum_{i=1}^{m} x_i P_i)(\sum_{j=1}^{m} y_j P_j))}{\operatorname{Tr}(\exp(\sum_{i=1}^{m} x_i P_i))} \le 1 \le \frac{\operatorname{Tr}(\exp(-\sum_{i=1}^{m} x_i C'_i)(\sum_{j=1}^{m} y_j C'_j))}{\operatorname{Tr}(\exp(-\sum_{i=1}^{m} x_i C'_i))}.
$$

Therefore there exists $j \in [m]$ such that

$$
\frac{\operatorname{Tr}(\exp(\sum_{i=1}^{m} x_i P_i) P_j)}{\operatorname{Tr}(\exp(\sum_{i=1}^{m} x_i P_i))} \le \frac{\operatorname{Tr}(\exp(-\sum_{i=1}^{m} x_i C'_i) C'_j)}{\operatorname{Tr}(\exp(-\sum_{i=1}^{m} x_i C'_i))},
$$

and hence $\mathrm{local}_j(x) \le \mathrm{global}(x)$.

If the algorithm outputs infeasible, then at that point $\min_j\{\mathrm{local}_j(x)\} > \mathrm{global}(x)$ and hence from the argument above $P_1, \ldots, P_m, C'_1, \ldots, C'_m$ is infeasible which in turn implies that $P_1, \ldots, P_m, C_1, \ldots, C_m$ is infeasible. □

**Lemma 6.** *If the algorithm returns some $x^*$, then*

$$
\sum_{i=1}^{m} x_i^* P_i \le (1+9\varepsilon)\mathbb{1} \quad \textit{and} \quad \sum_{i=1}^{m} x_i^* C_i \ge \mathbb{1}.
$$

*Proof.* Because of the condition of the while loop, it is clear that $\sum_{i=1}^{m} x_i^* C_i \ge \mathbb{1}$.

For $x \in \mathbb{R}^m$, define

$$
\Phi(x) \stackrel{\text{def}}{=} \operatorname{Imax}(\sum_{j=1}^{m} x_j P_j) - (1+\varepsilon)^3 \cdot \operatorname{Imin}(\sum_{j=1}^{m} x_j C_j).
$$

Note that the update of $C_j$'s at step 3(d) only increase $\text{Imin}(\sum_{j=1}^m x_j C_j)$. Hence using Lemma 4, we conclude that $\Phi(x)$ is non-decreasing during the algorithm. At step 1 of the algorithm,

$$\begin{aligned}\Phi(x) &\leq \text{Imax}(\sum_{j=1}^m x_j P_j) = \ln \text{Tr}(\exp(\sum_{i=1}^m x_i P_i)) \\ &\leq \ln(n \exp(\left\|\sum_{i=1}^m x_i P_i\right\|)) \leq \ln(n \exp(\sum_{i=1}^m \|x_i P_i\|)) = \ln n + 1.\end{aligned}$$

Hence just before the last increment,

$$\begin{aligned}\left\|\sum_{i=1}^m x_i P_i\right\| \leq \text{Imax}(\sum_{j=1}^m x_j P_j) &\leq \Phi(x) + (1+\varepsilon)^3 \cdot \text{Imin}(\sum_{j=1}^m x_j C_j) \\ &\leq \ln n + 1 + (1+\varepsilon)^3 \cdot \text{Imin}(\sum_{j=1}^m x_j C_j) \\ &\leq \ln n + 1 + (1+\varepsilon)^3 \cdot \lambda_{\min}(\sum_{j=1}^m x_j C_j) \\ &\leq \ln n + 1 + (1+\varepsilon)^3 N \leq (1+8\varepsilon)N \;.\end{aligned}$$

In the last increment, because of the condition on step 3(e) of the algorithm, $\|\sum_{i=1}^m x_i P_i\|$ increase by at most $\varepsilon$. Hence $\sum_{i=1}^m x_i^* P_i \leq (1+9\varepsilon)\mathbb{1}$. □

## Running time analysis

**Lemma 7.** *Assume that the algorithm does not return* infeasible *for some input instance. The number of times $g$ is increased at step 3(b) of the algorithm is $\mathcal{O}(N/\varepsilon)$.*

*Proof.* At the beginning of the algorithm $\text{Tr}(\exp(-\sum_{i=1}^m x_i C_i)) \leq n$ since each eigenvalue of $\exp(-\sum_{i=1}^m x_i C_i)$ is at most 1. Also $\text{Tr}\exp(\sum_{i=1}^m x_i P_i) \geq 1$. Hence

$$g = \text{global}(x) = \frac{\text{Tr}\exp(\sum_{i=1}^m x_i P_i)}{\text{Tr}(\exp(-\sum_{i=1}^m x_i C_i))} \geq \frac{1}{n} \geq \frac{1}{\exp(N)}.$$

At the end of the algorithm $\lambda_{\min}(\sum_{i=1}^m x_i C_i) \leq N + \varepsilon \leq 2N$. Hence

$$\text{Tr}(\exp(-\sum_{i=1}^m x_i C_i)) \geq \left\|\exp(-\sum_{i=1}^m x_i C_i)\right\| = \exp(-\lambda_{\min}(\sum_{i=1}^m x_i C_i)) \geq \exp(-2N).$$

Also (using Lemma 6)

$$\text{Tr}(\exp(\sum_{i=1}^m x_i P_i)) \leq n \left\|\exp(\sum_{i=1}^m x_i P_i)\right\| \leq n \exp((1+9\varepsilon)N) \leq \exp(11N).$$

Hence $g \leq \text{global}(x) \leq \exp(13N)$.

Whenever $g$ is updated at step 3(b) of the algorithm, we have

$$\text{global}(x) \geq \min_j\{\text{local}_j(x)\} > (1+\varepsilon)g$$

just before the update and $\text{global}(x) = g$ just after the update. Thus $g$ increases by at least $(1+\varepsilon)$ multiplicative factor. Hence the number of times $g$ increases is $\mathcal{O}(N/\varepsilon)$. □

**Lemma 8.** *Assume that the algorithm does not return* infeasible *for some input instance. The number of iterations of the while loop in the algorithm for a fixed value of $g$ is $\mathcal{O}(N\log(mN)/\varepsilon)$.*

*Proof.* From Lemma 6 and step 3(d) of the algorithm we have $\max\{\|\sum_{i=1}^m x_iP_i\|, \|\sum_{i=1}^m x_iC_i\|\} = \mathcal{O}(N)$ throughout the algorithm. On the other hand we have $\max\{\|\sum_{i=1}^m \delta x_iP_i\|, \|\sum_{i=1}^m \delta x_iC_i\|\} = \varepsilon$ at step 3(e). Hence $\delta = \Omega(\varepsilon/N)$ throughout the algorithm.

Let $x_j$ be increased in the last iteration of the while loop for a fixed value of $g$. Note that $x_j$ is initially $1/(m\|P_j\|)$ and at the end $x_j$ is at most $10N/\|P_j\|$ (since, using Lemma 6, $\|x_jP_j\| \leq \|\sum_{i=1}^m x_jP_j\| \leq 10N$). Hence the algorithm makes at most $\mathcal{O}(\log(mN)/\delta) = \mathcal{O}(N\log(mN)/\varepsilon)$ increments for each $x_j$.

Note that $\mathrm{local}_j(x)$ only increases throughout the algorithm (easily seen for steps 3(d) and 3(e) of the algorithm). Hence since the last iteration of the while loop (for this fixed $g$) increases $x_j$, it must be that each iteration of the while loop increases $x_j$. Hence, the number of iterations of the while loop (for this fixed $g$) is $\mathcal{O}(N\log(mN)/\varepsilon)$. □

We claim (without further justification) that each individual step in the algorithm can be performed in parallel time $\mathrm{polylog}(mn)$. Hence combining the above lemmas and using $N = \mathcal{O}(\frac{\ln(mn)}{\varepsilon})$, we get

**Corollary 9.** *The parallel running time of the algorithm is upper bounded by* $\mathrm{polylog}(mn) \cdot \frac{1}{\varepsilon^4} \cdot \log\frac{1}{\varepsilon}$.

# References


[1] S. Arora, E. Hazan, and S. Kale. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 339–348, 2005.

[2] S. Arora and S. Kale. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, pages 227–236, 2007.

[3] R. Jain, Z. Ji, S. Upadhyay, and J. Watrous. QIP = PSPACE. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC '10, pages 573–582, New York, NY, USA, 2010. ACM.

[4] R. Jain, S. Upadhyay, and J. Watrous. Two-message quantum interactive proofs are in PSPACE. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '09, pages 534–543, Washington, DC, USA, 2009. IEEE Computer Society.

[5] R. Jain and J. Watrous. Parallel approximation of non-interactive zero-sum quantum games. In *Proceedings of the 2009 24th Annual IEEE Conference on Computational Complexity*, CCC '09, pages 243–253, Washington, DC, USA, 2009. IEEE Computer Society.

[6] R. Jain and P. Yao. A parallel approximation algorithm for positive semidefinite programming. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science*, FOCS '11, Washington, DC, USA, 2011. IEEE Computer Society.

[7] S. Kale. *Efficient algorithms using the multiplicative weights update method*. PhD thesis, Princeton University, 2007.

[8] M. Luby and N. Nisan. A parallel approximation algorithm for positive linear programming. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, STOC '93, pages 448–457, New York, NY, USA, 1993. ACM.

[9] R. Peng and K. Tangwongsan. Faster and Simpler Width-Independent Parallel Algorithms for Positive Semidefinite Programming. *ArXiv e-prints : arXiv:1201.5135*, January 2012.

[10] N. E. Young. Sequential and parallel algorithms for mixed packing and covering. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 538–546, 2001.

# A Deferred proofs

**Proof of Lemma 2:** We will use the following Golden-Thompson inequality.

**Fact 10.** For Hermitian matrices $A, B$ : $\mathrm{Tr}(\exp(A+B)) \leq \mathrm{Tr}\exp(A)\exp(B)$.

We will also need the following fact.

**Fact 11.** Let $A$ be positive semidefinite with $\|A\| \leq \varepsilon \leq 1$. Then,

$$\exp(A) \leq \mathbb{1} + (1+\varepsilon)A \quad \text{and} \quad \exp(-A) \leq \mathbb{1} - (1-\varepsilon/2)A.$$

Consider,

$$\begin{aligned}
&\mathrm{lmax}(\sum_{j=1}^{m}(x_j+\alpha_j)A_j) - \mathrm{lmax}(\sum_{j=1}^{m} x_j A_j) \\
&= \ln\left(\frac{\mathrm{Tr}\exp(\sum_{i=1}^{m}(x_i+\alpha_i)A_i)}{\mathrm{Tr}\exp(\sum_{i=1}^{m} x_i A_i)}\right) \\
&\leq \ln\left(\frac{\mathrm{Tr}\exp(\sum_{i=1}^{m} x_i A_i)\exp(\sum_{j=1}^{m}\alpha_j A_j)}{\mathrm{Tr}\exp(\sum_{i=1}^{m} x_i A_i)}\right) \quad \text{(from Fact 10)} \\
&= \ln\left(\frac{\mathrm{Tr}\exp(\sum_{i=1}^{m} x_i A_i)(\mathbb{1}+(1+\varepsilon)(\sum_{j=1}^{m}\alpha_j A_j))}{\mathrm{Tr}\exp(\sum_{i=1}^{m} x_i A_i)}\right) \quad \text{(from Fact 11)} \\
&= \ln\left(1+\frac{(1+\varepsilon)\,\mathrm{Tr}\exp(\sum_{i=1}^{m} x_i A_i)(\sum_{j=1}^{m}\alpha_j A_j)}{\mathrm{Tr}\exp(\sum_{i=1}^{m} x_i A_i)}\right) \\
&\leq \frac{(1+\varepsilon)\,\mathrm{Tr}\exp(\sum_{i=1}^{m} x_i A_i)(\sum_{j=1}^{m}\alpha_j A_j)}{\mathrm{Tr}\exp(\sum_{i=1}^{m} x_i A_i)} \quad \text{(since } \ln(1+a) \leq a \text{ for all real } a\text{)}
\end{aligned}$$

The desired bound on $\mathrm{lmin}(\sum_{j=1}^{m}(x_j+\alpha_j)A_j) - \mathrm{lmin}(\sum_{j=1}^{m} x_j A_j)$ follows by analogous calculations. $\square$