# Fast Algorithms for Approximate Semidefinite Programming using the Multiplicative Weights Update Method

Sanjeev Arora* Elad Hazan Satyen Kale
Computer Science Department, Princeton University
35 Olden Street, Princeton, NJ 08544
{arora, ehazan, satyen}@cs.princeton.edu

## Abstract

*Semidefinite programming (SDP) relaxations appear in many recent approximation algorithms but the only general technique for solving such SDP relaxations is via interior point methods. We use a Lagrangian-relaxation based technique (modified from the papers of Plotkin, Shmoys, and Tardos (PST), and Klein and Lu) to derive faster algorithms for approximately solving several families of SDP relaxations. The algorithms are based upon some improvements to the PST ideas — which lead to new results even for their framework— as well as improvements in approximate eigenvalue computations by using random sampling.*

## 1. Introduction

Semidefinite programming (SDP) solves the following general problem:

$$\begin{aligned} \min\, & c \bullet X \\ A_j \bullet X \;&\geq\; b_j \qquad j = 1, 2, \ldots, m \\ X \;&\succeq\; 0 \end{aligned} \tag{1}$$

Here $X \in \mathbb{R}^{n \times n}$ is a matrix of variables and $A_1, A_2, \ldots, A_m \in \mathbb{R}^{n \times n}$. Here, for $n \times n$ matrices $A$ and $B$, $A \bullet B$ is their inner product treating them as vectors in $\mathbb{R}^{n^2}$, and $A \succeq 0$ is notation for "$A$ is positive semidefinite".

The first polynomial-time algorithm (strictly speaking, an approximation algorithm that computes the solution up to any desired accuracy $\varepsilon$ in time polynomial in $\log \frac{1}{\varepsilon}$) used the Ellipsoid method [16] but faster interior-point methods were later given by Alizadeh [3], and Nesterov and Nemirovskii [24]. The running time of Alizadeh's algorithm is $\tilde{O}(\sqrt{m}(m + n^3)L)$ where $L$ is an input size parameter. Here and elsewhere in this paper, the $\tilde{O}$ notation is used to suppress polylog$(\frac{mn}{\varepsilon})$ factors.

Much attention was focused on SDP as a result of the work of Goemans and Williamson [15], who used SDP to design new approximation algorithms for several NP-hard problems such as MAXCUT, MAX 2-SAT, and MAX 3-SAT. In subsequent years, SDP-based approximation algorithms were designed for coloring $k$-colorable graphs, MAX DICUT, etc. Then progress halted for a few years, until recent work of Arora, Rao, Vazirani [8] that gave a new $O(\sqrt{\log n})$-approximation for SPARSEST CUT. The ideas of this paper have been quickly extended to derive similar approximation algorithms for MIN 2CNF DELETION, MIN UNCUT, DIRECTED SPARSEST CUT, and DIRECTED BALANCED SEPARATOR in [2] and NON-UNIFORM SPARSEST CUT in [11, 7]. These new results rely on the so-called *triangle inequality* constraints, which impose a constraint for every *triple* of points. Thus the number of constraints $m = O(n^3)$, and the time to solve such SDPs is $\tilde{O}(n^{4.5})$.

In addition to these well-known approximation algorithms, SDP has also proved useful in a host of other settings. For instance, Linial, London, and Rabinovich [23] observe that given an $n$-point metric space, finding its minimum-distortion embedding into $\ell_2$ is a SDP with $m = O(n^2)$ constraints, which takes $\tilde{O}(n^4)$ time to solve. Recent approximation algorithms for *cut norm* of the matrix [4], and for certain subcases of correlation clustering [10], use a type of SDPs with $m = O(n)$, and hence require time $\tilde{O}(n^{3.5})$. (An intriguing aspect of this work is that the proof that the integrality gap of the SDP used in [4] is $O(1)$ uses the famous *Grothendieck inequality* from analysis.) Halperin and Hazan [17] showed that a biological *probability estimation problem* (HAPLOFREQ), which estimates the frequencies of haplotypes from a noisy sample, can be solved using SDP with $m = O(n^2)$. Chazelle, Kingsford, and Singh [12] use an SDP for *side chain positioning* (SCP), a problem in genomics.

Given the growing popularity of SDP, it would be extremely useful to develop alternative approaches that avoid

*Supported by David and Lucile Packard Fellowship, NSF grants CCR 0205594, CCR 0098180 and CCR 0514993.

the use of general-purpose interior point methods. Even problem-specific approaches would be very useful, but they too seem hard to come by.

A similar situation developed in the past decade in the case of linear programming, after LPs were used to design many approximation algorithms. Subsequent improvements to running times for these algorithms fall into two broad camps: (A) Eliminating use of LP in favor of a direct, combinatorial algorithm that uses the same intuition (in many cases, the same proof of the approximation ratio); (B) Solving the LP approximately instead of exactly. Typically this uses some version of the classical *Lagrangian relaxation* idea. Klein *et al* [21] showed how to do this for a specific multicommodity flow LP, and Plotkin, Shmoys, and Tardos [25] generalized the method to the family of *packing/covering* LPs. Later, Garg and Könemann [14] and Fleischer [13] improved the running times further for flow LPs. A recent survey [6] by the authors of the current paper points out that all such algorithms are a subcase of a more general, widely useful, and older framework they called *Multiplicative Weights Update method* algorithms. From now on we refer to this as the *MW framework*.

Speedups of type (A) and (B) for SDP-based algorithms are not easy. Speedups of type (A) have proved difficult because unlike LP-based approximation algorithms, the approximation ratio of SDP-based algorithms is proved by analyzing a *rounding* algorithm rather than by comparing to the dual. The lone exception we are aware of is the notion of *expander flows* studied by Arora, Rao, and Vazirani (this was presented as an alternative to their more well-known rounding approach that proved useful in most subsequent papers). However, the duality-based framework of ARV also found one use: it was instrumental in the design of a combinatorial, $O(\sqrt{\log n})$-approximation algorithm for (uniform) SPARSEST CUT and ran in $\tilde{O}(n^2)$ time [5], a significant improvement over the $\tilde{O}(n^{4.5})$ running time for the interior point algorithm. Interestingly, this algorithm was also derived in the MW framework. However, the duality-based framework from [8] has yet to be extended to problems other than uniform SPARSEST CUT, though this may yet happen. Thus improvements of type (A) have not been forthcoming for the other problems.

Klein and Lu [20] initiated study of algorithms of type (B) for SDPs. They adapted the PST/MW framework to approximately solve SDPs that arose in the algorithms of Goemans-Williamson and Karger, Motwani, and Sudan. The Klein-Lu approach reduces SDP solving to a sequence of approximate eigenvalue/eigenvector computations, which can be done efficiently using the well-known *power method*.

**Our work.** While the Klein-Lu work seemed promising, further progress then stalled. As we discuss in some detail in Sections 1.1 and 2, the main reason has to do with the *width* parameter, which is a value $\rho > 0$ such that the linear functions appearing in the constraints take values in the range $[-\rho, \rho]$. Then the number of iterations in the PST/MW framework is proportional to $\rho^2$. (Aside: In the PST packing-covering framework, the range of values was $[0, \rho]$, in which case the number of iterations is $O(\rho)$. This issue is discussed in [6].) Unfortunately, the width is large in most of the SDP relaxations mentioned above — the SDPs considered by Klein-Lu happened to be among the few where this problem is manageable.

Our first contribution is to modify the MW technique to handle some of these high-width SDPs. Our technique is a hybrid of the MW technique and an "exterior point" (i.e., Ellipsoid-like method) of Vaidya [26]; this lowers the dependence on the width and is very efficient so long as there only "a few" constraints with high width . (Actually the Vaidya algorithm is overkill in most instances, where the number of high-width constraints is a small constant, and one can use simpler ideas, based on binary search, that are reminiscent of fixed-dimension LP algorithms.) Formally one needs a two-level implementation of the multiplicative update idea, that combines the original constraints into new, smaller number of constraints. While this makes intuitive sense —MW methods excel at handling many low-width constraints and exterior point methods excel at handling a few, high-width constraints—this hybrid technique appears to be new. (It is related though to the observation in [25] that their packing-covering problems are solvable in polynomial time using the dual ellipsoid method.)

The above ideas can be used to prove new results about the general PST framework as well; these are described in Section 5.

Our second contribution is to use a better technique for eigenvalue/eigenvector computations than the power method, namely, the Lanczos algorithm. This is the method of choice among numerical analysts, but has not been used in theory papers thus far because worst-case analysis for it is hard to find in the literature. We adapt an analysis for positive semidefinite matrices from [22] to our needs (see Lemma 2).

Then we suggest further speeding up the Lanczos algorithm by first *sparsifying* the matrix via *random sampling*. Our sparsification is quite similar to that of [1], though we get bounds that are more suitable to our applications. In comparison to [1], our sampling performs better or worse depending on some parameters of the input matrix. The details are in the full version of the paper and the sampling itself may be useful for efficiently computing low rank approximations of matrices.

### 1.1. Overview of our results

Our algorithms assume a feasibility version of the SDP (1). Here, we implicitly perform a binary search on the optimum and the objective is converted to a constraint in the standard way. We also assume an additional constraint, a bound on the trace of the solution:

$$\begin{aligned} A_j \bullet X &\geq b_j \qquad j = 1, 2, \ldots, m \\ \sum_i X_{ii} &\leq R \\ X &\succeq 0 \end{aligned} \tag{2}$$

The upper bound on the trace, $\mathbf{Tr}(X) = \sum_i X_{ii}$, is usually absent in the textbooks, but is natural for relaxation SDPs. For instance, in combinatorial optimization, usually we have some unit vectors $v_1, v_2, \ldots, v_n$ associated with, say, the nodes in a graph, and $X_{ij} = v_i \cdot v_j$. Then $\mathbf{Tr}(X) = n$. In any case, $\mathbf{Tr}(X)$ for the optimum $X$ can usually be "guessed" by binary search.

We wish to solve the SDP approximately up to a given tolerance $\varepsilon$, by which we mean that either we find a solution $X$ which satisfies all the constraints up to an additive error of $\varepsilon$, i.e. $A_j \bullet X - b_j \geq -\varepsilon$ for $j = 1, 2, \ldots, m$, or conclude correctly that the SDP is infeasible.

As in the case of LP solving (PST, etc.), the Multiplicative Weights Update idea for solving SDPs is to perform several iterations, indexed by "time" $t$, of the following. Associate a non-negative weight $w_j^{(t)}$ with constraint $j$, where $\sum_j w_j^{(t)} = 1$. A high current weight for a constraint indicates that it was not "satisfied" too well in the past, and is therefore should receive higher priority in the next step. The optimization problem for the next step is to

$$\begin{aligned} \max \sum_j w_j^{(t)} (A_j \bullet X - b_j) \\ X &\succeq 0 \\ \sum_i X_{ii} &\leq R \end{aligned}$$

This is actually an eigenvalue problem in disguise, since the optimum is attained at an $X$ that has rank 1. Thus the Lagrangian relaxation idea would be to solve this eigenvalue problem, and update the weights $w_i$ according to the usual multiplicative update rule, for some constant $\beta$:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} (1 - \beta (A_j X_t - b_j)) / Z^t$$

where $X_t$ was the solution to the eigenvalue problem (expressed as a rank 1 positive semidefinite matrix) at time $t$, and $Z^t$ is the normalization factor to make the weights sum to 1. Then if $\beta$ is small enough, the average $\frac{1}{T} \sum_{t=1}^{T} X_t$ is guaranteed to converge to a near-feasible solution to the original SDP (assuming a feasible solution exists).

| Problem | | |
|---|---|---|
| Previous best | This paper | Improvement |
| MAXQP | | |
| $\tilde{O}(n^{3.5})$ | $\tilde{O}(\frac{n^{1.5}}{\varepsilon^{2.5}} \times \min\{N, \frac{n^{1.5}}{\varepsilon \alpha^*}\})$ | For $N = o(n^2)$ or $\alpha^* = \omega(\frac{1}{\sqrt{n}})$ |
| HAPLOFREQ | | |
| $\tilde{O}(n^4)$ | $\tilde{O}\left(\frac{n^{2.5}}{\varepsilon^{2.5}}\right)$ | $\Omega(n^{1.5})$ |
| SCP | | |
| $\tilde{O}(n^4)$ | $\tilde{O}\left(\frac{n^{1.5}N}{\varepsilon^{4.5}}\right)$ | $\Omega(\frac{n^{3.5}}{N})$ |
| EMBEDDING | | |
| $\tilde{O}(n^4)$ | $\tilde{O}\left(\frac{n^3}{d_{\min}^{2.5}\varepsilon^{3.5}}\right)$ | For $d_{\min} = \omega(n^{-0.4})$ |
| SPARSEST CUT | | |
| $\tilde{O}(n^{4.5})$ | $\tilde{O}(\frac{n^3}{\varepsilon^2})$ | For $\varepsilon = \omega(n^{-0.75})$ |
| MIN UNCUT, BALANCED SEPARATOR, ETC. | | |
| $\tilde{O}(n^{4.5})$ | $\tilde{O}\left(\frac{n^{3.5}}{\varepsilon^2}\right)$ | For $\varepsilon = \omega(\frac{1}{\sqrt{n}})$ |

**Figure 1. Summary table of results.**

Our new contributions to this approach, including the issue of managing the high-width constraints, and of fast eigenvalue computations, were already discussed earlier. Now we describe our main new results. The main point to stress is that in practice our algorithm may run even faster than the worst-case estimates we provide later on. Throughout the paper we carefully list times in terms of number of eigenvalue/eigevector computations required, and these tend run much faster than our worst-case estimate in Lemma 2. By contrast, each iteration of Alizadeh's SDP solver requires *Cholesky decomposition*, i.e. a factorization of a positive semidefinite matrix $X$ as $X = VV^T$ for some matrix $V$. The fastest algorithms for this require $O(n^3)$ time.

The SDPs we apply our approach to include four relaxations of graph partitioning problems: MAXQP, SPARSEST CUT, MIN UNCUT and BALANCED SEPARATOR, two SDPs arising from computational biology: HAPLOFREQ,SCP, and a metric embedding formulation EMBEDDING. The running times we obtain are shown in Figure 1. We use the notation $\alpha^*$ for the optimum of a given SDP. In this extended abstract we detail only some of the applications, the rest appear in the full version.

The worst-case running time is a function of $\varepsilon$. In some cases, there are also dependencies upon other problem parameters. For instance, in EMBEDDING, where one is seeking the minimum distortion embedding into $\ell_2$, the $\varepsilon$ is benign, say $0.1$. However, there is a dependence on the aspect ratio; in other words, minimum squared internode distance $d_{min}$, where sum of squares of all $\binom{n}{2}$ internode distances is normalized to $n^2$. Our algorithm provides a speedup when

$d_{min}$ is at least $n^{-0.4}$. (This is still an interesting set of metrics.) Likewise, our algorithm for MAXQP provides speedups when either of the following two conditions are true: (a) the number of nonzero entries in the matrix $A$ is $N = o(n^2)$ or (b) the optimum $\alpha^*$ is at least $\frac{1}{\sqrt{n}}$ with the normalization $\sum_{i,j} |A_{ij}| = 1$. Again, this is an interesting class of matrices.

For problems such as (uniform) SPARSEST CUT, BALANCED SEPARATOR, and MIN 2CNF DELETION, the current approximation algorithms require a very small $\varepsilon$, namely, $\alpha^*/|E|$. We improve upon existing SDP solvers when this is at least $1/\sqrt{n}$.

As already noted, in practice our algorithms may run faster.

## 2. An illustration of the method

In this section, we give more details of the method by illustrating its application to the SDP (MAXQP) given below. This SDP arises in many algorithms such as approximating MAXCUT, maximizing the correlation in correlation clustering, approximating the CUTNORM of a matrix, approximating the Grothendieck constant of a graph, etc. See [10] for a discussion.

$$\begin{aligned} \max\ & A \bullet X & & \\ & X_{ii} \leq 1 & & i = 1, 2, \ldots, n \\ & X \succeq 0 & & \end{aligned} \tag{MAXQP}$$

We assume here that diag$(A) \geq 0$. Let $N \geq n$ be the number of non-zero entries of $A$. We wish to get a multiplicative $1 - O(\varepsilon)$ approximation to the optimum value of the SDP. Note that Alizadeh's interior point method solves the SDP in $\tilde{O}(n^{3.5})$ time.

**Step I: Bounding the optimum and trace.** We compute bounds on the optimum of the SDP, $\alpha^*$. For simplicity, assume $\sum_{ij} |A_{ij}| = 1$, this amounts to scaling the optimum by a fixed quantity. Let $X^*$ be the optimum solution. Since $X^*$ is positive semidefinite, for any $i, j$, $(X^*_{ij})^2 \leq X^*_{ii} X^*_{jj} \leq 1$, so $|X^*_{ij}| \leq 1$. Thus, $\alpha^* = A \bullet X^* = \sum_{ij} A_{ij} X^*_{ij} \leq \sum_{ij} |A_{ij}| = 1$. Conversely, the solution $X$ specified by $X_{ij} = \frac{\text{sgn}(A_{ij})}{n}$ and $X_{ii} = 1$ is positive semidefinite, and achieves an objective value of $\frac{1}{n}$. This gives a lower bound on $\alpha$. We also compute a bound on $\mathbf{Tr}(X)$. In this case, this is simply $\sum_i X_{ii} \leq n$.

**Step II: Reduction to feasibility problem.** We "guess" the value of $\alpha$ using binary search in the range computed in Step I. Let $\alpha$ be our current guess. Define a convex set $\mathcal{P} = \{X \succeq 0,\ \sum_i X_{ii} \leq n\}$. We rewrite the SDP as a feasibility problem for the binary search as follows:

$$\begin{aligned} \frac{1}{\alpha} A \bullet X - 1 &\geq 0 & & \\ 1 - X_{ii} &\geq 0 & & i = 1, 2, \ldots, n \\ X &\in \mathcal{P} & & \end{aligned} \tag{3}$$

We now need to estimate the *width* of each constraint. This is defined as the maximum absolute value it can take for $X \in \mathcal{P}$. More specifically, for a generic constraint of the type $A \bullet X - b \geq 0$ where $X \in \mathcal{P}$, assume that the range of values that $A \bullet X - b$ can take is $[-\ell, \rho]$ or $[-\rho, \ell]$ where $1 \leq \ell \leq \rho$. Then $\rho$ is called the width of the constraint. In (3), the range of the constraint $\frac{1}{\alpha} A \bullet X - 1 \geq 0$ for $X \in \mathcal{P}$ is $[-\frac{n}{\alpha}, \frac{n}{\alpha}]$, so the width is $\frac{n}{\alpha}$. The range of the constraint $1 - X_{ii} \geq 0$ for $X \in \mathcal{P}$ is $[-n, 1]$, so the width is $n$.

Note that an additive error of $\varepsilon$ translates to a multiplicative error of $1 - O(\varepsilon)$ to the objective, assuming the binary search guessed the value of the optimum to within a factor of $1 + \varepsilon$.

**Step III: The Multiplicative Weights Update algorithm.** Lagrangian relaxation algorithms assume that there is an algorithm, ORACLE, to solve the following relaxed feasibility problem: given non-negative weights $w_0, w_1, w_2, \ldots, w_n$ on the constraints such that $\sum_{i=0}^n w_i = 1$, consider the weighted combination of constraints $w_0(\frac{1}{\alpha} A \bullet X - 1) + \sum_{i=1}^n w_i(1 - X_{ii})$. Then ORACLE either finds an $X \in \mathcal{P}$ which makes this combination $\geq -\frac{\varepsilon}{2}$ or declares correctly that no $X \in \mathcal{P}$ makes the combination non-negative.

In the latter case, we declare infeasibility of (3) since otherwise any feasible solution would make the weighted combination of constraints non-negative, a contradiction. If the former case holds whenever the ORACLE is presented a set of weights, then we can get an $\varepsilon$ approximate solution to (3), as given in the following theorem, proved in the full version of the paper:

**Theorem 1** *Consider the general SDP (2). Let $\mathcal{P} = \{X \succeq 0,\ \sum_i X_{ii} \leq R\}$. Assume that for any $j$, $A_j \bullet X - b_j$ lies in one of the ranges $[-\ell, \rho]$, $[-\rho, \ell]$. Also, assume that there is an algorithm,* ORACLE*, which runs in time $T_{oracle}$, and given any set of non-negative weights $w_1, w_2, \ldots, w_m$ on the constraints summing to 1, either finds an $X \in \mathcal{P}$ which makes the weighted combination $\sum_{j=1}^m w_j(A_j \bullet X - b_j) \geq -\frac{\varepsilon}{2}$ or declares correctly that no $X \in \mathcal{P}$ makes this combination non-negative. Then there is an algorithm which runs in $O(\frac{\ell\rho}{\varepsilon^2}(T_{oracle} + m))$ time and either gets an $\varepsilon$ approximate solution to (2) or concludes that it is infeasible.*

**Step IV:** ORACLE **from eigenvector computations.** Note that the ORACLE of Theorem 1 needs to maximize the weighted combination of constraints $\sum_{j=1}^m w_j(A_j \bullet X - b_j)$

over the set $\mathcal{P}$ of all positive semidefinite matrices $X$ whose trace is bounded by $R$. We show in section 3 that this amounts to approximately computing the largest eigenvector of the matrix $C = \sum_{j=1}^{m} w_j(A_j - \frac{b_j}{R} I)$ up to tolerance $\delta = \frac{\varepsilon}{2R}$. Define $T_{\text{ev}}(C, \delta)$ to be the time needed for this. In Lemma 1, we show that $T_{\text{oracle}} = \tilde{O}(T_{\text{ev}}(C, \delta))$.

Getting back to our example, SDP (MAXQP), we have the following parameters: $\ell = \rho = \frac{n}{\alpha}$, $m = n + 1$, $R = n$. The running time from Theorem 1 is $\tilde{O}(\frac{n^2}{\varepsilon^2 \alpha^2}(T_{\text{oracle}} + n))$ which is worse than Alizadeh's algorithm for $\alpha = o(n^{-0.25})$ even without factoring in $T_{\text{oracle}}$. We show how to improve the running time in the next step.

**Step V: Inner and Outer SDPs.** Now we indicate our width reduction technique. Observe that there is a single constraint, $\frac{1}{\alpha} A \bullet X - 1 \geq 0$, which has high width ($\frac{n}{\alpha}$). The other constraints have width bounded by $n$. This happens in all our applications: we will find a constant sized set of constraints of high width and the rest will have low width. We devise a hybrid algorithm, using the multiplicative update method to handle the low width constraints and an exterior point algorithm to handle the (few) high width constraints.

The idea is to push the high width constraint, $\frac{1}{\alpha} A \bullet X - 1 \geq 0$, into the convex set, to create a new convex set $\mathcal{Q} = \{X \in \mathcal{P},\ \frac{1}{\alpha} A \bullet X - 1 \geq 0\}$, and run the Multiplicative Weights Update algorithm of Theorem 1 on the other constraints over $\mathcal{Q}$. We call this the *outer SDP*.

The ORACLE now gets a weighted combination of the constraints, $\sum_{i=1}^{n} w_i(1 - X_{ii})$, and needs to find an $X \in \mathcal{Q}$ which makes this $\geq -\frac{\varepsilon}{2}$ or declare that no such $X$ makes the combination non-negative. This can be achieved by approximately solving the 2 constraint SDP of the form

$$\begin{aligned} \sum_{i=1}^{n} w_i(1 - X_{ii}) \ &\geq 0 \\ \frac{1}{\alpha} A \bullet X - 1 \ &\geq 0 \\ X \ &\in \mathcal{P} \end{aligned}$$

We call this the *inner SDP*. The ORACLE for this SDP needs to optimize a weighted combination of *all* constraints over $\mathcal{P}$, this is identical to the one we had in Step IV.

We solve the inner SDP using an exterior point algorithm. The observation is that the ORACLE yields a separation hyperplane for the dual problem, and so we can apply Vaidya's algorithm. Recall that $m$ is the number of constraints. Let $\mathcal{M}(m) = O(m^{2.36})$ be the time needed to multiply two $m \times m$ matrices.

**Theorem 2** *With the setup as in Theorem 1, there is an algorithm which produces an $\varepsilon$ approximate solution to the general SDP (2) or declares correctly its infeasibility in time*

$$\tilde{O}(m \log(\rho) \cdot T_{\textit{oracle}} + m \log(\rho) \mathcal{M}(m \log(\rho)))$$

Note that this algorithm has poor dependence on the number of constraints but handles high width very well. In our example, the number of constraints in the inner SDP is just 2, and the width is poly$(mn)$ from the trace bound. Thus, the algorithm of Theorem 2 solves it in $\tilde{O}(T_{\text{oracle}}) = \tilde{O}(T_{\text{ev}}(C, \frac{\varepsilon}{2n}))$ time.

In all our applications, we have a constant sized set of constraints with much higher (yet, polynomial) width than the rest. Let the other, low width, constraints take values in the range $[-\ell_L, \rho_L]$ or $[-\rho_L, \ell_L]$.

**Corollary 1** *With the given setup, the hybrid algorithm which composes an outer and inner SDP runs in time*

$$\tilde{O}\left(\frac{\ell_L \rho_L}{\varepsilon^2}\left[T_{\textit{ev}}\left(C, \frac{\varepsilon}{2R}\right) + m\right]\right)$$

For SDP (MAXQP), this yields the following theorem, which will be proved in section 4.1:

**Theorem 3** *A multiplicative $1 - O(\varepsilon)$ approximation to SDP* (MAXQP) *can be obtained in time*

$$\tilde{O}\left(\frac{n^{1.5}}{\varepsilon^{2.5}} \cdot \min\left\{N,\ \frac{n^{1.5}}{\varepsilon \alpha^*}\right\}\right)$$

This running time is never worse than the $\tilde{O}(n^{3.5})$ running time of Alizadeh's interior point algorithm. It is asymptotically faster if the matrix $A$ is not dense, i.e. $N = o(n^2)$, or if $\alpha^* = \omega(\frac{1}{\sqrt{n}})$.

We note here the special case of the MAXCUT SDP. For this problem, the matrix $A$ is the combinatorial Laplacian of the input graph, divided by $4m$. We note that the bound $\alpha^* \geq \frac{1}{8}$ is easily obtained from the greedy algorithm. Thus, our algorithm runs in time $\tilde{O}(n^{1.5} \cdot \min\{N, n^{1.5}\})$.

The best algorithm for solving the MAXCUT SDP is due to Klein and Lu [20], with running time $\tilde{O}(nN)$. Our algorithm is a $\sqrt{n}$ factor worse when $N = o(n^2)$. However, our algorithm solves the much more general problem (MAXQP) and the approach of [20] does not extend to this general problem.

## 3. Implementing ORACLE using the approximate eigenvector computations

In this section, we present lemmas which describe how to efficiently implement the ORACLE of Theorem 1 using approximate eigenvector computations. The proofs appear in the full version.

**Lemma 1** *Suppose we have a procedure that given a matrix $C \in \mathbb{R}^{n\times n}$ and a tolerance $\delta > 0$, computes a unit vector $x$ which satisfies $x^T Cx \geq -\delta$, in time $T_{ev}(C,\delta)$, or declares correctly that $C$ is negative definite. Then using this procedure once with $C = \sum_{j=1}^{m} w_i(A_j - \frac{b_j}{R} I)$ and $\delta = \frac{\varepsilon}{2R}$ we can implement* ORACLE.

Thus, we need to approximately compute the top eigenvector of the matrix which represents the weighted combination of the constraints. The Lanczos algorithm with a random starting vector is the most efficient algorithm for finding extreme eigenvectors. The running time for the Lanczos algorithm used in our context is the following:

**Lemma 2** *Let $C \in \mathbb{R}^{n\times n}$ be a matrix with $N$ non-zero entries and eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. Let $\delta > 0$ be a given error parameter. Let $\gamma = \max\{\frac{\lambda_1}{\lambda_1+|\lambda_n|}, \frac{\delta}{|\lambda_n|}\}$. Then the Lanczos algorithm with a random start applied to the matrix $C + \Lambda I$ yields with high probability a unit vector $x$ which satisfies $x^T Cx \geq -\delta$ or declares correctly that $C$ is negative definite in time $T_{ev}(C,\delta) = \tilde{O}(\frac{N}{\sqrt{\gamma}})$.*

The parameter $\gamma$ in Lemma 2 is unknown of course, but in applications we will derive lower bounds for it. The lemma shows that the running time of the ORACLE depends on the sparsity of $C$, i.e. the number on non-zero entries in it. We provide a randomized sparsification procedure with the following guarantee:

**Lemma 3** *Let $C \in \mathbb{R}^{n\times n}$ be a matrix with $N$ non-zero entries and let $S = \sum_{ij} |C_{ij}|$. Let $\delta > 0$ be a given error parameter. Then there is a randomized procedure which runs in $\tilde{O}(N)$ time and with high probability produces a matrix $C'$ such that $A'$ has $O(\frac{\sqrt{n}S}{\delta})$ non-zero entries and for all unit vectors $x \in \mathbb{R}^n$, we have $|x^T Cx - x^T C'x| \leq O(\delta)$.*

Thus, $C'$ can be used in place of $C$ in the Lanczos algorithm, *if* it turns out to be sparser: the decision for specific applications depends on the relative values of $N$ and $\frac{\sqrt{n}S}{\delta}$.

# 4. Applications

In this section, we describe applications of the method to the following representative problems: MAXQP and HAPLOFREQ, EMBEDDING, and SPARSEST CUT. More applications of the method to SCP, MIN UNCUT, MIN 2CNF DELETION and related problems can be found in the full version of the paper.

These examples demonstrate the strengths and limitations of the method. It should be noted that the method does not automatically yield faster algorithm; additional fine-tuning (mostly in terms of bounding large negative eigenvalues) is necessary for specific applications.

## 4.1. SDP relaxations of Quadratic Programs

Our first application is the SDP (MAXQP) that we used to illustrate the method, and we complete the proof of Theorem 3.

PROOF: **[Theorem 3]**

We apply Corollary 1. The range of the constraints of the outer SDP, viz. $1 - X_{ii} \geq 0$ for $1 \leq i \leq n$, is $[-n, 1]$ for $X \in \mathcal{Q}$. Thus $\ell_L = 1, \rho_L = n$. Now we bound the running time of the eigenvector computation procedure for the ORACLE.

Given non-negative weights $w_0, w_1, \ldots, w_n$ which sum to 1, the matrix $C$ from Lemma 1 in this case is $w_0(\frac{1}{\alpha}A - \frac{1}{n}I) + \sum_{i=1}^{n} w_i(\frac{1}{n}I - e_i e_i^T)$, where $e_i$ is the $i^{\text{th}}$ standard basis vector, and $\delta = \frac{\varepsilon}{2n}$.

To apply Lemma 2, we need to bound the most negative eigenvalue, $\lambda_n$, of $C$. Observe that $\mathbf{Tr}(C) = w_0(\frac{1}{\alpha}\mathbf{Tr}(A) - 1) \geq -1$. Since $\mathbf{Tr}(C) = \sum_i \lambda_i$, we conclude that $(n-1)\lambda_1 + \lambda_n \geq -1$. This implies that if $|\lambda_n| \geq 2$, then $\frac{\lambda_1}{\lambda_1+|\lambda_n|} \geq \frac{1}{4n}$. Otherwise, if $|\lambda_n| \leq 2$, then $\frac{\delta}{|\lambda_n|} \geq \frac{\varepsilon}{4n}$. Thus, $\gamma = \max\{\frac{\lambda_1}{\lambda_1+|\lambda_n|}, \frac{\delta}{|\lambda_n|}\} \geq \frac{\varepsilon}{4n}$, and by Lemma 2, the eigenvector procedure takes $\tilde{O}(N\frac{\sqrt{n}}{\sqrt{\varepsilon}})$ time.

If we apply the sparsification procedure of Lemma 3, then the relevant parameters are $S = \sum_{ij} |C_{ij}| = O(\frac{1}{\alpha}\sum_{ij} |A_{ij}|) = O(\frac{1}{\alpha})$ (recall $\sum_{ij} |A_{ij}| = 1$). Thus the sparsification procedure yields a matrix $C'$ with $O(\frac{n^{1.5}}{\varepsilon\alpha})$ non-zero entries. Overall, the running time of the Lanczos algorithm becomes $\tilde{O}(\min\{N, \frac{n^{1.5}}{\varepsilon\alpha}\} \cdot \sqrt{\frac{n}{\varepsilon}})$ as stated.

Putting everything together, the final running time of the algorithm becomes

$$\tilde{O}\left(\frac{n^{1.5}}{\varepsilon^{2.5}} \cdot \min\left\{N, \frac{n^{1.5}}{\varepsilon\alpha^*}\right\}\right)$$

□

## 4.2. SDP relaxations of biological probability estimation problems

The following SDP arises in the context of certain biological probability estimation problems, such as in finding haplotype frequencies. See [17] for a discussion.

$$\begin{aligned} \max\ & A \bullet X & & \\ \sum_{ij} & X_{ij} = 1 & & \\ & X_{ij} \geq 0 & & 1 \leq i,j \leq n \\ & X \succeq 0 & & \end{aligned} \qquad \text{(HAPLOFREQ)}$$

We assume here that $A$ is a non-negative matrix. This SDP is a natural relaxation in certain problems where a probability distribution is required. Intuitively, we want to find a

probability distribution $\{p_1, p_2, \ldots, p_n\}$ which maximizes the objective $\sum_{ij} A_{ij} p_i p_j$. In the SDP relaxation, the $X_{ij}$ variables represent $p_i p_j$.

We apply our method to this problem. Step I requires that we bound the optimum and the trace. Let the optimum to this SDP be denoted $\alpha^*$. We claim that $\alpha^*$ is in the range $\max_{ij}\{A_{ij}\} \cdot [\frac{1}{4}, 1]$. The upper bound is trivial since the objective is a convex combination of the $A_{ij}$ values. Let $A_{kl}$ be the maximal $A_{ij}$. Then the lower bound is obtained by taking the unit vector $u = \frac{1}{2}(e_l + e_k)$, where $e_i$ is the $i^{\text{th}}$ standard basis vector, and letting $X$ be the positive semi-definite matrix $uu^T$. Since all $A_{ij}$ are non-negative, this solution has value at least $\frac{1}{4}A_{kl}$.

The trace of $X$ is trivially bounded by 1 from the first constraint. Note also that w.l.o.g. we can relax the first constraint to be $\sum_{ij} X_{ij} \leq 1$, in the optimum the sum obviously equals 1 since all the quantities are non-negative. Let $N$ be the number of non-zero entries of $A$.

**Theorem 4** *SDP* (HAPLOFREQ) *can be approximated up to a multiplicative error of* $1 - O(\varepsilon)$ *in* $\tilde{O}(\frac{n^{2.5}}{\varepsilon^{2.5}})$ *time.*

PROOF: According to step II, we "guess" $\alpha$ using binary search and write the following feasibility SDP. Here, $\mathcal{P}$ is the convex set $\{X \succeq 0\,, \sum_i X_{ii} \leq 1\}$.

$$\begin{aligned} \frac{1}{\alpha} A \bullet X - 1 &\geq 0 \\ 1 - \sum_{ij} X_{ij} &\geq 0 \\ X_{ij} &\geq 0 \qquad 1 \leq i, j \leq n \\ X &\in \mathcal{P} \end{aligned}$$

The width of the first constraint is $(\frac{1}{\alpha}\sum_{ij} |A_{ij}|)^2 = O(n^4)$. This is the high width constraint which we will put into the inner SDP. The width for the rest of the constraints is $O(1)$, these constitute the outer SDP. Thus, $\ell_L = \rho_L = 1$, and $\delta = \frac{\varepsilon}{2}$. Let $C$ represent the weighted combination of the constraints for the ORACLE. According to Corollary 1, the SDP can be $\varepsilon$ approximately in $\tilde{O}(\frac{1}{\varepsilon^2} \cdot [T_{ev}(C, \frac{\varepsilon}{2}) + n^2])$ time.

It remains to estimate $T_{ev}(C, \frac{\varepsilon}{2})$. The matrix $C$ is of the form $w_0(\frac{1}{\alpha}A - I) + w_1(I - J) + \sum_{ij} w_{ij} E_{ij}$, where $J$ is the all 1's matrix, and $w_0, w_1, w_{ij}$ for $1 \leq i, j \leq n$ are non-negative weights summing to 1.

To bound the most negative eigenvalue, $\lambda_n$, of $C$, we use the Gershgorin circle theorem [18], which implies that $|\lambda_n| \leq \max_i\{\sum_j |C_{ij}|\}$. For the matrix $C$, the dominant contributors to this maximum are the matrices $\frac{1}{\alpha}A$ and $J$. For any $i$, we have $\sum_j \frac{1}{\alpha}|A_{ij}| \leq 4n$ since $\alpha \geq \frac{1}{4}\max_{ij} A_{ij}$. Also, for any $i$, $\sum_j |J_{ij}| = n$. Thus, the bound on $|\lambda_n|$ is $O(n)$.

Thus, the $\gamma$ of Lemma 2 is $\geq \Omega(\frac{\varepsilon}{n})$, and hence $T_{ev}(C, \frac{\varepsilon}{2}) = \tilde{O}(\frac{n^{2.5}}{\sqrt{\varepsilon}})$ because $C$ is a dense matrix. Since $\sum_{ij} |C_{ij}|$ can be as large as $\Omega(n^2)$, sparsification does not help here.

Finally, the total running time comes to $\tilde{O}(\frac{n^{2.5}}{\varepsilon^{2.5}})$. □

For comparison, the best known interior point algorithm solves this SDP in $\tilde{O}(n^4)$ time.

### 4.3. Embedding of finite metric spaces into $\ell_2$

Given a finite metric space on $n$ points specified by the pairwise distances $\{D_{ij}\}$, embedding into $\ell_2$ with minimum distortion amounts to solving the following mathematical program. For convenience of notation, let $d_{ij} = D_{ij}^2$.

$$\begin{aligned} &\min \alpha \\ d_{ij} \;\leq\; ||v_i - v_j||^2 \;\leq\; \alpha \cdot d_{ij} &\qquad 1 \leq i < j \leq n \\ v_i \;\in\; \mathbb{R}^n &\qquad 1 \leq i \leq n \end{aligned}$$

(EMBEDDING)

By Bourgain's theorem [9] the minimum distortion is $O(\log n)$. Thus, the optimum value $\alpha^*$ of SDP EMBEDDING is $O(\log^2 n)$. We assume that the distances are scaled so that $\sum_{ij} d_{ij} = n^2$. We claim that this implies that there is an optimal solution $v_1, v_2, \ldots, v_n$ which satisfies $\sum_i ||v_i||^2 \leq \alpha^* n$: we may assume that the optimal solution satisfies $\sum_i v_i = 0$, otherwise we can shift the origin to the sum of the vectors; this does not change the pairwise distances $||v_i - v_j||^2$. Thus we have $\alpha^* n^2 = \alpha^* \sum_{ij} d_{ij} \geq \sum_{ij} ||v_i - v_j||^2 = n \sum_i ||v_i||^2$. The claim follows.

**Theorem 5** *SDP* (EMBEDDING) *can be approximated up to a* $1 + O(\varepsilon)$ *multiplicative factor in* $\tilde{O}(\frac{n^3}{d_{min}^{2.5}\varepsilon^{3.5}})$ *time.*

PROOF: Guess $\alpha$ using binary search, and formulate the following SDP. Define the convex set $\mathcal{P} = \{X \succeq 0 \mid \sum_i X_{ii} \leq \alpha n\}$.

$$\begin{aligned} \frac{\alpha}{d_{ij}}(X_{ii} - 2X_{ij} + X_{jj}) - 1 \;&\geq\; 0 \qquad 1 \leq i < j \leq n \\ 1 - \frac{1}{d_{ij}}(X_{ii} - 2X_{ij} + X_{jj}) \;&\geq\; 0 \qquad 1 \leq i < j \leq n \\ X \;&\in\; \mathcal{P} \end{aligned} \tag{4}$$

Since $X \succeq 0$, the expression $(X_{ii} - 2X_{ij} + X_{jj})$ is always positive. The bound on the trace implies that these terms are bounded by $\alpha n$. Hence the width of the constraints of SDP (4) is bounded by $\rho_L \leq O(\frac{n\alpha}{d_{ij}}) = \tilde{O}(\frac{n}{d_{\min}})$, where $d_{\min} = \min_{ij}\{d_{ij}\}$. Also, $\ell_L = 1$, $m = 2n^2$, and $\delta = \frac{\varepsilon}{2\alpha n}$. Let $C$ be the matrix representing the weighted combination of all the constraints. Thus, an $\varepsilon$ approximate solution to the SDP can be found in time $\tilde{O}(\frac{n}{d_{\min}\varepsilon^2} \cdot \left[n^2 + T_{ev}\left(C, \frac{\varepsilon}{2\alpha n}\right)\right])$.

The most negative eigenvalue of $C$, $\lambda_n$, can be bounded in absolute value by $O(\frac{1}{d_{\min}})$. This is because all constraints

have only $O(1)$ terms, each bounded in absolute value by $O(\frac{\alpha}{d_{\min}}) = \tilde{O}(\frac{1}{d_{\min}})$. Since $C$ is a convex combination of the constraints, we conclude that for all $j$, $\sum_{ij} |C_{ij}| = \tilde{O}(\frac{1}{d_{\min}})$, and this bounds $|\lambda_n|$ by the Gershgorin circle theorem. Thus, by Lemma 2, $T_{\text{ev}}(C, \frac{\varepsilon}{n})$ can be bounded by $\tilde{O}(\frac{N\sqrt{n}}{\sqrt{\varepsilon d_{\min}}})$, where $N$ is the number of non-zero entries in $C$.

Sparsification could potentially reduce the number of matrix entries. Since $S = \sum_{ij} |C_{ij}| = \tilde{O}(\frac{1}{d_{\min}})$, so by Lemma 3 the number of entries could be reduced to $\tilde{O}(\frac{n^{1.5}}{\varepsilon d_{\min}})$.

Thus the total running time comes to

$$\tilde{O}\left(\min\left\{\frac{n^3}{d_{\min}^{2.5}\varepsilon^{3.5}}, \frac{n^{3.5}}{d_{\min}^{1.5}\varepsilon^{2.5}}\right\}\right)$$

For comparison, interior point methods can solve this SDP in time $\tilde{O}(n^4)$. Note that in order to improve over the running time of interior point methods, the first expression is always better.

□

### 4.4. SDP relaxation of Sparsest Cut

For a graph $G = (V, E)$ with $V = \{1, 2, \ldots, n\}$, the following SDP arises as a relaxation for the Sparsest Cut problem in [8]:

$$\begin{aligned} \min \sum_{\{i,j\}\in E} ||v_i - v_j||^2 & \\ ||v_i - v_j||^2 + ||v_i - v_k||^2 - ||v_j - v_k||^2 &\geq 0 \quad i,j,k \in [n] \\ \sum_{i<j} ||v_i - v_j||^2 &= n \end{aligned}$$

(SPARSEST CUT)

Note that the RHS of the last constraint is $n$ rather than 1 as in [8]. This effectively scales the range of the optimum, $\alpha^*$, to be $[0, n]$. We desire an additive $\varepsilon$ approximate solution.

**Theorem 6** *SDP* (SPARSEST CUT) *can be approximated up to an additive error $\varepsilon$ in $\tilde{O}(\frac{n^3}{\varepsilon^2})$ time.*

PROOF: As usual, we guess $\alpha$ using binary search, and write the following SDP, with the convex set $\mathcal{P} = \{X \succeq 0, \sum_i X_{ii} = 1\}$:

$$\begin{aligned} \alpha - \sum_{ij\in E} (X_{ii} - 2X_{ij} + X_{jj}) &\geq 0 \\ (X_{ii} - 2X_{ij} + X_{jj}) + (X_{ii} - 2X_{ik} + X_{kk}) & \\ -(X_{jj} - 2X_{jk} + X_{kk}) &\geq 0 \\ X &\in \mathcal{P} \end{aligned}$$

Here, just as in the embeddings problem from the previous section, we make the assumption that $\sum_i v_i = 0$ w.l.o.g. Then the constraint $\sum_{ij} ||v_i - v_j||^2 \leq n$ is equivalent to $\sum_i ||v_i||^2 \leq 1$.

Since $\sum_i X_{ii} = 1$, for any $i, j$, $|X_{ij}| \leq \sqrt{X_{ii}X_{jj}} \leq 1$. Thus, the width of the first constraint is $O(n^2)$. The width for the rest of the constraints is $O(1)$. Since there are $n^3$ constraints anyway, for the ORACLE we solve the eigenvector problem up to arbitrary precision using a standard algorithm such as QR, which runs in $n^3$ time as well. Finally, using the inner and outer SDPs of Corollary 1,the SDP can be approximated up to $\varepsilon$ in time:

$$\tilde{O}(\frac{1}{\varepsilon^2} \cdot [T_{\text{oracle}} + n^3]) = \tilde{O}\left(\frac{n^3}{\varepsilon^2}\right)$$

□

## 5. Extensions to the PST framework

In this section we outline how the techniques discussed earlier can be applied to extend and in some cases improve running times for algorithms in the PST framework. In the PST framework, we have the following kind of feasibility problem:

| GENERAL FEASIBILITY: $\exists ? x \in \mathcal{P}$ such that $Ax \geq b$, where $A \in \mathbb{R}^{m\times n}$ and $\mathcal{P}$ is a convex set. |
|---|

We assume that we have an algorithm, ORACLE, which given non-negative weights $w_1, w_2, \ldots, w_m$ summing to 1, finds an $x \in \mathcal{P}$ which makes the weighted combination $\sum_{j=1}^m w_j(A_j x - b_j) \geq -\frac{\varepsilon}{2}$ or declares correctly that no such $x$ makes the combination non-negative. Let $T_{oracle}$ be the running time of the oracle. We wish to devise an algorithm which finds an $x \in \mathcal{P}$ that satisfies all the constraints up to an additive error of $\varepsilon$ or declares correctly that the system is infeasible. If for every $x \in \mathcal{P}$, each $A_j x \in [-\rho, \rho]$ then the PST algorithm solves this in time $\tilde{O}(\frac{\rho^2}{\varepsilon^2}(T_{oracle} + m))$.

### 5.1. Composition of Lagrangian Relaxation Algorithms

Now we consider systems where "a few" of the constraints have high width and the rest do not.

We can extend the MW algorithm to this setting without incurring a high penalty for the high width constraints. We separate the few, $m_H$, high width constraints, which have width $\rho_H$. These constitute the inner feasibility problem as in Step V of Section 2. The rest of the $m_L = m - m_H$ constraints have low width $\rho_L$ and take values in one of the

ranges $[-\ell_L, \rho_L]$ or $[-\rho_L, \ell_L]$. These constitute the outer feasibility problem.

The outer problem will be solved using the Multiplicative Weights Update algorithm while the inner problem will be solved using Vaidya's exterior point algorithm. We get the following theorem:

**Theorem 7** *Then there is an algorithm which either gets an $\varepsilon$ approximate solution to* GENERAL FEASIBILITY *or declares correctly its infeasibility in time*

$$\tilde{O}\left(\frac{\ell_L \rho_L}{\varepsilon^2}\left[\left(m_H \log(\rho_H) T_{oracle} + \quad m_H \log(\rho_H)\mathcal{M}(m_H \log(\rho_H))\right) + m_L\right]\right)$$

### 5.2. Mixed packing and covering constraints

Mixed packing and covering problems are defined as follows. The width $\rho$ is defined as above.

| MIXED PACKING-COVERING $\exists ? x \in \mathcal{P}$ such that $Ax \leq b$ and $\hat{A}x \geq \hat{b}$, where $A \in \mathbb{R}^{\hat{m} \times n}$, $\hat{A} \in \mathbb{R}^{m-\hat{m} \times n}$, $b, \hat{b} > 0$ and $\mathcal{P}$ is a convex set such that $Ax, \hat{A}x \geq 0$ for $x \in \mathcal{P}$. |
|---|

The original PST paper showed how to $\varepsilon$-approximately solve (i.e. RHS of constraints violated by a at most a factor of $1 \pm \varepsilon$) mixed packing and covering formulations in time proportional to the width squared. For the special case of linear programming, Young [27] provides an algorithm that is independent of the width completely. Recently Jansen [19] obtained an approximation algorithm for general fractional mixed covering and packing problems that is independent of the width, at the expense of an extra factor of $m$, the number of constraints, in the running time.

Our generalized proof for the multiplicative weights update method in [6] allows us to reduce the dependance on the width in the general setting from quadratic to linear. The following theorem gives an algorithm with faster running time than Jansen's algorithm provided the width is no more than the number of constraints. This is the case in many of our applications where the number of constraints is $O(n^2)$ (eg. HAPLOFREQ) or $O(n^3)$ (eg. SPARSEST CUT).

**Theorem 8** *A $\varepsilon$-approximate solution to* MIXED PACKING-COVERING *can be computed in time*

$$\tilde{O}\left(\frac{\rho}{\varepsilon^2}(m + T_{oracle})\right)$$

## 6. Conclusions and Future Work

We have designed new hybrid Lagrangian relaxation algorithms for solving SDPs. Our ideas are general though we customize them for some interesting SDPs. Each iteration step is an approximate eigenvector computation, which is very efficient in practice, even though the theoretical worst case bounds listed here do not show this. (Even so, our worst-case bounds provide speedups for specific SDPs over interior point methods.) In every iteration of the interior point algorithm, one needs to compute the Cholesky decomposition of a positive semidefinite matrix. This takes much as $O(n^3)$ time, whereas the top eigenvector of a matrix can be computed much more efficiently. This is where our method gets an edge over interior point methods.

Another advantage of our method is that the Cholesky decomposition of the final solution is obtained automatically because the solution is a convex combination of many rank 1 matrices. Typically, the first step of rounding in approximation algorithms relying on SDP is to compute the Cholesky decomposition of the optimal solution, and this step comes for free as noted.

The chief limitation of this method is from the polynomial dependence on $\frac{1}{\varepsilon}$. Some applications (such as general SPARSEST CUT) requires $\varepsilon$ to be very tiny and then this method is rendered useless. The main goal of future work will be to reduce the dependence on $\frac{1}{\varepsilon}$. It is expected that ideas which helped us reduce the dependence on the width may help here.

Our hybrid approach for the PST/MW framework may also be useful for other convex optimization problems.

## Acknowledgements

We would like to thank Beresford Parlett and Daniel Spielman for helpful discussions.

## References

[1] D. Achlioptas and F. McSherry. Fast computation of low rank matrix approximations. In *33rd ACM STOC*, pages 611–618, 2001.

[2] A. Agarwal, M. Charikar, K. Makarychev, and Y. Makarychev. $O(\sqrt{\log n})$ approximation algorithms for Min UnCut, Min 2CNF deletion, and directed cut problems. In *37th ACM STOC*, pages 573–581, 2005.

[3] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM J. Optim.*, 5(1):13–51, 1995.

[4] N. Alon and A. Naor. Approximating the cut-norm via grothendieck's inequality. In *36th ACM STOC*, pages 72–80, 2004.

[5] S. Arora, E. Hazan, and S. Kale. $O(\sqrt{\log n})$ approximation to sparsest cut in $\tilde{O}(n^2)$ time. In *45th IEEE FOCS*, pages 238–247, 2004.

[6] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta algorithm and applications. *Manuscript*, 2005.

[7] S. Arora, J. R. Lee, and A. Naor. Euclidean distortion and the sparsest cut. In *37th ACM STOC*, pages 553–562, 2005.

[8] S. Arora, S. Rao, and U. V. Vazirani. Expander flows, geometric embeddings and graph partitioning. In *36th ACM STOC*, pages 222–231, 2004.

[9] J. Bourgain. On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel J. Math.*, 52(1–2):46–52, 1985.

[10] M. Charikar and A. Wirth. Maximizing quadratic programs: Extending grothendieck's inequality. In *45th IEEE FOCS*, pages 54–60, 2004.

[11] S. Chawla, A. Gupta, and H. Räcke. Embeddings of negative-type metrics and an improved approximation to generalized sparsest cut. In *SODA*, pages 102–111, 2005.

[12] B. Chazelle, C. Kingsford, and M. Singh. A semidefinite programming approach to side-chain positioning with new rounding strategies. *INFORMS Journal on Computing, Special Issue on Computational Molecular Biology/Bioinformatics*, pages 380–392, 2004.

[13] L. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM J. Discrete Math.*, 13(4):505–520, 2000.

[14] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *39th IEEE FOCS*, pages 300–309, 1998.

[15] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

[16] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer Verlag, 1988.

[17] E. Halperin and E. Hazan. Haplofreq - estimating haplotype frequencies efficiently. In *9th RECOMB*, pages 553–568, 2005.

[18] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1994.

[19] K. Jansen. Approximation algorithms for mixed fractional packing and covering problems. In *Proceedings of the 3rd IFIP International Conference on Theoretical Computer Science*, pages 223–236, 2004.

[20] P. Klein and H.-I. Lu. Efficient approximation algorithms for semidefinite programs arising from MAX CUT and COLORING. In *28th ACM STOC*, pages 338–347, 1996.

[21] P. Klein, S. Plotkin, C. Stein, and E. Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM J. Comput.*, 23(3):466–487, 1994.

[22] J. Kuczyński and H. Woźniakowski. Estimating the largest eigenvalue by the power and lanczos algorithms with a random start. *SIAM Journal on Matrix Analysis and Applications*, 13(4):1094–1122, 1992.

[23] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15:215–245, 1995.

[24] Y. Nesterov and A. Nemirovskii. *Interior Point Polynomial Methods in Convex Programming: Theory and Applications*. Society for Industrial and Applied Mathematics, Philadelphia, 1994.

[25] S. A. Plotkin, D. B. Shmoys, and T. Tardos. Fast approximation algorithm for fractional packing and covering problems. In *32nd IEEE FOCS*, pages 495–504, 1991.

[26] P. M. Vaidya. A new algorithm for minimizing convex functions over convex sets. *Math. Program.*, 73(3):291–341, 1996.

[27] N. Young. Sequential and parallel algorithms for mixed packing and covering. In *42nd IEEE FOCS*, pages 538–546, 2001.