2012 IEEE 53rd Annual Symposium on Foundations of Computer Science

Faster SDP Hierarchy Solvers for Local Rounding Algorithms

Venkatesan Guruswami*
guruswami@cmu.edu

Ali Kemal Sinop†
asinop@cs.cmu.edu

***Abstract*—Convex relaxations based on different hierarchies of linear/semi-definite programs have been used recently to devise approximation algorithms for various optimization problems. The approximation guarantee of these algorithms improves with the number of *rounds* r in the hierarchy, though the complexity of solving (or even writing down the solution for) the r'th level program grows as $n^{\Omega(r)}$ where n is the input size.**

In this work, we observe that many of these algorithms are based on *local* rounding procedures that only use a small part of the SDP solution (of size $n^{O(1)}2^{O(r)}$ instead of $n^{\Omega(r)}$). We give an algorithm to find the requisite portion in time polynomial in its size. The challenge in achieving this is that the required portion of the solution is not fixed a priori but depends on other parts of the solution, sometimes in a complicated iterative manner.

Our solver leads to $n^{O(1)}2^{O(r)}$ time algorithms to obtain the same guarantees in many cases as the earlier $n^{O(r)}$ time algorithms based on r rounds of the Lasserre hierarchy. In particular, guarantees based on $O(\log n)$ rounds can be realized in polynomial time. For instance, one can (i) get $O(1/\lambda_r)$ approximations for graph partitioning problems such as minimum bisection and small set expansion in $n^{O(1)}2^{O(r)}$ time, where λ_r is the r'th smallest eigenvalue of the graph's normalized Laplacian; (ii) a similar guarantee in $n^{O(1)}k^{O(r)}$ for Unique Games where k is the number of labels (the polynomial dependence on k is new); and (iii) find an independent set of size $\Omega(n)$ in 3-colorable graphs in $(n2^r)^{O(1)}$ time provided $\lambda_{n-r} < 17/16$.

We develop and describe our algorithm in a fairly general abstract framework. The main technical tool in our work, which might be of independent interest in convex optimization, is an efficient ellipsoid algorithm based separation oracle for convex programs that can output a *certificate of infeasibility with restricted support*. This is used in a recursive manner to find a sequence of consistent points in nested convex bodies that "fools" local rounding algorithms.

*Computer Science Department, Carnegie Mellon University.
†Center for Computational Intractability, Department of Computer Science, Princeton University. This work was done while the author was a student at Carnegie Mellon University.

This research was supported in part by NSF grant CCF-1115525 and MSR-CMU Center for Computational Thinking. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

I. Introduction

A rich body of recent research has shown that for many optimization problems, the Unique Games conjecture (UGC) serves as a barrier to further improvements to the approximation factor achieved by efficient algorithms. In many cases, including all constraint satisfaction problems and various graph partitioning problems, the best algorithms are based on fairly simple semi-definite programming (SDP) relaxations. The UGC foretells that for these problems, no tighter relaxation than these simple SDPs will yield a better approximation ratio in the worst-case.

Hierarchies of convex relaxations. A natural question thus is to understand the power and limitations of potentially stronger SDP relaxations, for example those from various *hierarchies* of relaxations. These hierarchies are parameterized by an integer r (called rounds/levels) which capture higher order correlations between (roughly r-tuples of) variables (the basic SDP captures only pairwise correlations, and certain extensions like triangle inequalities pose constraints on triples). Larger the r, tighter the relaxation. The optimum of n'th level of the hierarchy, where n is the number of variables in the underlying integer program, usually equals the integral optimum.

There are several hierarchies of relaxations that have been studied in the literature, such as Sherali-Adams hierarchy of linear programs [1], the Lovász-Schrijver hierarchy [2], a "mixed" hierarchy combining Sherali-Adams linear programs with the base level SDP, and the Lasserre hierarchy [3] (see [4] for a recent survey focusing on their use in approximate combinatorial optimization). Of these hierarchies, the most powerful one is the Lasserre hierarchy (see [5] for a comparison), and therefore holds the most potential for new breakthroughs in approximation algorithms. Arguably, Lasserre SDPs pose the currently strongest known threat to the Unique Games conjecture, as even the possibility of the 4'th level of Lasserre SDP relaxation improving upon the Goemans-Williamson 0.878 approximation factor for Max Cut has *not* been ruled out. Recently, it has also

0272-5428/12 $26.00 © 2012 IEEE
DOI 10.1109/FOCS.2012.58

been shown that $O(1)$ rounds of the Lasserre hierarchy are able to solve all candidate gap instances of Unique Games [6]. (On the other hand, for some of the weaker hierarchies, integrality gaps for super-constant rounds *are* known for various Unique-Games hard problems [7], [8].)

In light of the above, the power and limitations of the Lasserre hierarchy merit further investigation. There has been a fair bit of recent interest in Lasserre hierarchy based approximation algorithms [9], [10], [11], [12], [13], [14], [15]. For instance, our work [11] shows that various graph partitioning problems (including minimum bisection, sparsest cut, and Unique Games) can be well-approximated by $\approx r$ rounds of the Lasserre SDP on graphs whose r'th smallest eigenvalue λ_r (of the Laplacian) is reasonably large.

A (near)-optimal solution to the r'th level Lasserre relaxation can be found in $n^{O(r)}$ time. So understanding the power of these relaxations for small values of r is of particular interest. The main contribution of this work is to improve the running time of various Lasserre-based approximation algorithms to $2^{O(r)}n^{O(1)}$ (from the default $n^{O(r)}$). In particular, the guarantees achieved by $O(\log n)$ rounds of Lasserre SDPs can be realized in polynomial time. Plugging our methods into some of the algorithms in [11] gives us the following application:

Theorem 1. *For the graph partitioning problems such as uniform sparsest cut, small set expansion, and minimum bisection, one can compute a cut with cost at most* $\frac{1.5}{\min\{1,\lambda_r\}}$ *times the optimum in* $n^{O(1)}2^{O(r)}$ *time, where* n *is the number of vertices and* λ_r *the* r*'th smallest eigenvalue of the normalized Laplacian of the input graph.*[1]

For Unique Games with k *labels and* n *variables, an approximation factor of* $3/\min\{1,\lambda_r\}$ *can be achieved for the minimization version in time* $n^{O(1)}k^{O(r)}$*, where* λ_r *the* r*'th smallest eigenvalue of the normalized Laplacian of the constraint graph. (Note the polynomial dependence of the runtime on the number of labels, which is new to this work.)*

Another use of the Lasserre hierarchy to find independent sets in 3-colorable graphs [14], which we can also similarly speed up. A table of several algorithms whose algorithms we are able to improve is given in Section VII.

Theorem 2. *Given a* 3*-colorable regular graph* G*, we can find an independent set of size at least* $n/12$ *in* $2^{O(r)}n^{O(1)}$ *time provided the* r*'th* largest *eigenvalue of the normalized Laplacian of* G *is at most* $17/16$.

[1]For the case of minimum bisection, the cut may have a $o(n)$ imbalance. In fact, an approximation factor of $\frac{1+\varepsilon}{\min\{1,\lambda_r\}}$ can be achieved in $n^{O(1/\varepsilon^2)}2^{O(r/\varepsilon)}$ time.

Our techniques might also be useful in the context of fixed-parameter tractability, which we leave as a potentially interesting avenue for future research.

Local rounding algorithms. Note that even writing down the full r-round Lasserre solution takes $n^{\Omega(r)}$ time. The hope to speed-up the algorithms to a runtime dependence of $2^{O(r)}$ is based on the observation that many of the rounding algorithms have a "local" character that uses only a small portion of the SDP solution. In the simplest setting, the rounding algorithm proceeds in two steps: (i) find a "seed set" S^* of $\approx r$ nodes based only the solution to the base (1-round) SDP, and (ii) use the value of r-round Lasserre solution on the set S^* to sample a partial assignment to S^* and then propagate it to the other nodes. Thus the rounding algorithm only uses the portion of the Lasserre SDP corresponding to the subsets $S^* \cup \{u\}$ for various u. Further, the analysis of the rounding algorithm also relies only on Lasserre consistency constraints for subsets of S^*. More generally, the algorithms might pick a sequence of seed subsets $S_1, S_2, \ldots, S_\ell$ iteratively and the SDP solution restricted to subsets of $S_1 \cup S_2 \cup \cdots S_\ell$ is used for rounding.

Note that the needed portion of the solution (corresponding to S^*, or more generally $S_1 \cup S_2 \cup \cdots S_\ell$) itself depends on certain other parts of the solution. So one cannot simply project the space down to the relevant dimensions to find the required part of the Lasserre solution. Our main technical contribution is an ellipsoid algorithm that can find the needed partial solution (which satisfies all the local constraints induced on those variables) in time polynomial in the number of variables in the partial solution. We stress that the partial solution we find *may not* extend to a full Lasserre solution. This, however, does not matter for the approximation guarantee as it will *"fool" the rounding algorithm* which can't distinguish the solution we find from a global Lasserre solution.

There are two examples of hierarchy based approximation algorithms which have been speeded up to $2^{O(r)}$ dependence on the number of rounds, both of which rely on weaker hierarchies than Lasserre: (i) the algorithm for sparsest cut on bounded treewidth graphs using the Sherali-Adams hierarchy [16] and (ii) the Unique Games algorithm based on the "mixed" hierarchy [12]. The faster algorithm is for the former case is immediate as the required portion of the solution only depends on the input graph, so one can simply find that part using any LP solver. For the Unique Games algorithm, the seed set S^* depends on the vector solution to the basic SDP relation. The goal is to extend the solution

to local distributions of labels on subsets $S^* \cup \{u\}$ for various nodes u, whose 2-way marginals agree with the vector inner products. As briefly sketched in [12], these constraints form a linear program, and if infeasible, by Farkas' lemma, one can get a new constraint for the vector inner products, which can be fed into an ellipsoid algorithm for solving the basic SDP. Our situation is more complicated as we handle several iterations of seed set selection, and the "extension" problems we solve are no longer simple linear programs. Also, the runtime of the Unique Games algorithm in [12] had an exponential dependence on the number of labels, as opposed to our polynomial dependence.

Main technique: Separation oracle with restricted support. We describe the high level ideas behind our method for finding adequate partial solutions to Lasserre SDPs in Section II. Our approach applies in a fairly general set-up, and therefore we describe our methods in an abstract framework for clarity, both in Section II and later in Section IV where the formal details appear.

In addition to the runtime improvements, our results contribute a useful, and to our knowledge new, basic tool in convex optimization, which is an ***efficient ellipsoid algorithm based separation oracle that can output a certificate of infeasibility with restricted support*** (or more generally belonging to a restricted subspace). For instance, suppose we are given a convex body $K \subseteq \mathbb{R}^n$ via a separation oracle for it. Given a point $y \in \mathbb{R}^U$ (a potential partial solution) for some $U \subset \{1, 2, \ldots, n\}$, we give an algorithm to either find $x \in K$ such that $\text{proj}_U(x) = y$ (if one exists[2]), or find a separating constraint *that is supported on* U.

II. An abstract framework of local rounding and Overview of our techniques

Consider a rounding algorithm with following property: Given an optimal solution $x \in \mathbb{R}^N$ as input, it only reads a much smaller part of this solution, say $T \subseteq N$ with $|T| = o(|N|)$. We call these "local" rounding algorithms: Even though this setting might sound too restrictive and/or unrealistic, observe that several of the known rounding algorithms which use "hierarchies" fit into this framework, [9], [10], [11], [14], [13], [15]. See Section VII for details.

a) Local Rounding.: We first start by outlining a generic *iterative rounding* algorithm. This framework depends on two application specific deterministic[3] procedures, **SEED** and **FEASIBLE**. Without going into the formal details, at a high level, $\textbf{SEED}_S$ procedure chooses next "seed set" designating which fragment of the solution we will read based on current seeds S and $\textbf{FEASIBLE}_S(y)$ is a *strong separation oracle* for a convex body K_S representing the induced solutions on seeds S.

At the end, final seeds and induced solution are fed into another application specific rounding procedure.

b) Formal Framework.: Given two problem specific procedures, **SEED** and **FEASIBLE**, we formalize the generic algorithm described above as follows.

1) Let $x \in \mathbb{R}^N$ be a vector representing an optimal solution for some convex optimization problem, $x \in K_N$.
2) Let $S(0)$ be the initial solution fragment and $y(0) \leftarrow x_{S(0)}$ be the induced solution.
3) For $i \leftarrow 0$ to ℓ:
   a) Fail if $\textbf{FEASIBLE}_{S(i)}(y(i))$ asserts infeasible (i.e. $y(i) \notin K_{S(i)}$).
   b) If $i < \ell$, read next part of solution: $S(i+1) \leftarrow \textbf{SEED}_{S(i)}(y(i))$ and $y(i+1) \leftarrow x_{S(i+1)}$.
4) Perform rounding using $S(\ell)$ and $y(\ell)$.

c) Our Goal.: Suppose $|S(\ell)| \ll |N|$ – the algorithm reads only a negligible portion of the full solution. Then can we find an equivalent rounding algorithm which runs in time $\text{poly}(|S(\ell)|)$ as opposed to $\text{poly}(N)$? Claim 3 shows this can be expected:

Claim 3. *Above rounding algorithm can not distinguish between the following two cases, i.e. any properties satisfied by the output assuming 1 still holds under a weaker condition, 2:*

1) *There exists a feasible solution $x \in \mathbb{R}^N$, i.e. $\textbf{FEASIBLE}_N(x)$ asserts feasible.*
2) *For all $i \in \{0, \ldots, \ell\}$:*
   - *$y(i) \in K_{S(i)}$: $\textbf{FEASIBLE}_{S(i)}(y(i))$ asserts feasible,*
   - *$S(i+1) = \textbf{SEED}_{S(i)}(y(i))$ if $i < \ell$,*
   - *$y(i+1)_{S(i)} = y(i)$ if $i < \ell$.*

Using this insight, we first consider a simple case and give an algorithm whose running time depends on $|S(\ell)|$ instead of $|N|$.

A. An Algorithm for a Simple Case

Suppose that **SEED** procedure does not depend on y. Then the above conditions can easily be expressed as a convex problem of size $|S(\ell)|$, which is much smaller than the original problem. Then we can solve this convex problem using standard ellipsoid procedure and execute the above procedure on this solution instead.

[2]Actually, we need the volume of $K \cap \text{proj}_U^{-1}(y)$ to be at least some small ε

[3]We can allow randomization also, but we stick to the deterministic case for simplicity, since all the seed selection procedures used by the known algorithms can be derandomized.

B. Our Algorithm

Unfortunately for all algorithms we consider in this paper, the procedure **SEED** heavily depends on y. In particular, at the i^{th} level, $0 \leq i < \ell$, we are trying to solve the following induced problem on $S(i+1)$. Given $y(i) \in K_{S(i)}$:

$$\begin{aligned} \text{Find} \quad & y(i+1) \\ \text{st} \quad & y(i+1)_{S(i)} = y(i),\ y(i+1) \in K_{S(i+1)}; \\ & \exists y(i+2) \in K_{S(i+1)} : y(i+2)_{S(i+1)} = y(i+1) \\ & \qquad \text{where } S(i+2) = \mathbf{SEED}_{S(i+1)}(y(i+1)); \\ & \qquad \vdots \\ & \exists y(\ell) \in K_{S(\ell)} : y(\ell)_{S(\ell-1)} = y(\ell-1) \\ & \qquad \text{where } S(\ell) = \mathbf{SEED}_{S(\ell-1)}(y(\ell-1)). \end{aligned} \tag{1}$$

Observe that if we can construct a weak separation oracle for eq. (1) at $(i+1)^{th}$ level, then we can combine it with ellipsoid algorithm to solve the problem at i^{th} level also. Thus if we can convert this ellipsoid algorithm to a weak separation oracle, then we can call these separation oracles recursively starting from 0^{th} level all the way down to ℓ^{th} level:

Recursive Separation Oracle. (Template for i^{th} level)

1. Given $S(i)$ and $y(i)$, if $\mathbf{FEASIBLE}_{S(i)}(y(i))$ asserts infeasible and returns c, then assert infeasible and return c (to the $(i-1)^{th}$ level).
2. If $i = \ell$, then return the solution $y(\ell)$.
3. Let $S(i+1) \leftarrow \mathbf{SEED}_{S(i)}(y(i))$.
4. Use ellipsoid method to find $y(i+1)$ such that $y(i+1)_j = y(i)_j$ for all $j \in S(i)$ with separation oracle being a recursive call for the $(i+1)^{th}$ level (which takes inputs $S(i+1)$ and $y(i+1)$).
5. If ellipsoid method fails to find such solution $y(i+1)$, return a separating hyperplane.

The key question now is how one might implement (the currently vague) step 5. Let us inspect a simple option, and see what goes wrong with it.

Return an arbitrary hyperplane seen so far. Any inequality returned by the recursive separation oracle call is a valid separating hyperplane, so consider returning an arbitrary one. What goes wrong in this case? The problem is that the running time now might be as large as polynomial in $|N|$. To see this, suppose that $\mathbf{FEASIBLE}_{S(\ell)}(y(\ell))$ returned an inequality on support $S(\ell)$. Then the parent ellipsoid procedure needs to keep track of the additional variables from this particular $S(\ell)$, call it $\tilde{S}$. At some later stage, the algorithm may backtrack and change an earlier seed set, say $S(\ell-5)$, which will need to a new $S(\ell)$. But the algorithm would still need to keep the values of variables from the $\tilde{S}$, the old value of $S(\ell)$. Continuing in this fashion, the set of variables the algorithm has to track might end up being N, which is equivalent to constructing the whole solution on $\mathbb{R}^N$!

This attempt has not been futile though, as it shows what kind of hyperplanes we need:

Any hyperplane returned by step 5 at $(i+1)^{st}$ level should have support $S(i)$. (2)

We outline our proposed solution in the next section.

C. Our Contribution: A Separation Oracle with Restricted Support

Our solution to 2 is based on a new ellipsoid algorithm for finding separating constraints with restricted support. Specifically, the main technical contribution of this paper is Algorithm 1 with the following guarantee: Given a feasibility problem of the form

$$\text{Find } y \in \mathbb{R}^n \text{ subject to } \Pi y = y_0,\ y \in \text{int}(K),$$

where Π is a projection matrix, $y_0 \in \text{span}(\Pi) \subseteq \mathbb{R}^n$; along with separation oracle for convex body K; it either finds feasible y or asserts that the problem is infeasible and outputs a separating hyperplane $c \in \text{span}(\Pi)$. This algorithm coupled with the recursive separation oracle meets both our correctness and running time requirements. In particular, the running time instead of being the trivial bound of $|N|^{O(1)}$ will be roughly $|S(\ell)|^{O(\ell)}$. Assuming the exponential-time hypothesis, the exponential dependence on the number of seed selection stages ℓ cannot be avoided (a sub-exponential dependence would lead to a $f(k)n^{o(k)}$ time algorithm to decide if an n-vertex graph has a k-clique).

Remark 1. Our algorithm can be thought as a weak separation oracle for eq. (1) at level i given a weak separation oracle for level $i+1$. When each convex body $K_{S(2)}, \ldots, K_{S(\ell)}$ is guaranteed to be a polytope, such as LPs from the Sherali-Adams Hierarchy, it is known that one can obtain a strong separation oracle at level i by using only using a strong separation oracle at level $i+1$ (see Corollary 6.5.13 in [17]). However in the case of semi-definite programming, it is an open question [18] whether one can obtain a strong separation oracle from another strong separation oracle in polynomial time.

III. Preliminaries

For any positive integer n, let $[n] \triangleq \{1, 2, \ldots, n\}$. We will use $\emptyset$ to denote empty set. Given set A, let 2^A be its power set, i.e. set of all subsets. For any real k, we will use $A_k, A_{\leq k}$ and $A_{\geq k}$ to denote the set of all subsets of A having size exactly k, at most k and at least k respectively. Observe that $2^A = A_{\geq 0}$, $\emptyset = A_0$. Finally note that $A_{\geq 1}$ is the set of non-empty subsets of A.

Given sets A, B and a field $\mathbb{F}$ ($\mathbb{R}$ for reals, $\mathbb{Q}$ for rationals) , we will use $\mathbb{F}^A$ and $\mathbb{F}^{A \times B}$ to denote vectors

and matrices over $\mathbb{F}$ whose rows and columns are identified with elements of A and B respectively. For any function $f : A \to \mathbb{F}$ (resp. $g : A \times B \to \mathbb{F}$), we will use $[f(u)]_{u\in A}$ (resp. $[g(u,v)]_{(u,v)\in A\times B}$) to denote the vector (resp. matrix) whose value at row u (resp. row u and column v) is equal to $f(u)$ (resp. $g(u,v)$). Given vector $x \in \mathbb{F}^A$ and matrix $Y \in \mathbb{F}^{A\times B}$, for any subset C and D let $x_C \in \mathbb{F}^C$ and $Y_{C,D} \in \mathbb{F}^{C\times D}$ denote the minors of x, Y on rows $A \cap C$ and columns $B \cap D$ with 0's everywhere else.

Finally we will use $\mathbf{S}^A \supset \mathbf{S}^A_+ \supset \mathbf{S}^A_{++}$ to denote the set of symmetric, positive semi-definite and positive definite matrices on rows and columns A.

A. Convex Geometry and Ellipsoid Method

The main crux of our algorithm relies on an ellipsoid solver method which can also return a certificate of infeasibility. Throughout this section, we assume the underlying space is n-dimensional whose coordinates are identified with $[n]$. So all our vectors and matrices (unless noted otherwise) will have $[n]$ as their rows.

Notation 4 (Projection)**.** *We will use $\Pi \in \mathbf{S}^{[n]}_+$ to denote a projection matrix representing some linear subspace $\mathrm{span}(\Pi) \subseteq \mathbb{R}^{[n]}$ and $\Pi^\perp$ to denote the projection matrix onto null space of Π, i.e. $\Pi^\perp = \mathrm{identity} - \Pi$.*

Given vector $y_0 \in \mathbb{R}^{[n]}$, we will use $y_0 \in \Pi$ if y_0 is in the span of Π, i.e. $\Pi y_0 = y_0$ and we will use $\Pi^{-1}(y_0)$ to denote the following set of vectors:

$$\Pi^{-1}(y_0) \triangleq \left\{ y \in \mathbb{R}^{[n]} \middle| \Pi^\perp(y - y_0) = 0 \right\}.$$

Notation 5 (Balls)**.** *Given a set $K \subseteq \mathbb{R}^{[n]}$ and non-negative real $\varepsilon \geq 0$, we define $\mathbb{B}(K, \pm\varepsilon)$ in the following way.*

$$\mathbb{B}(K,\varepsilon) \triangleq \left\{ x \in \mathbb{R}^{[n]} | \exists y \in K \text{ s.t. } \|y - x\|_2 \leq \varepsilon \right\}.$$

$$\mathbb{B}(K,-\varepsilon) \triangleq K \setminus \mathbb{B}(\mathbb{R}^{[n]} \setminus K, \varepsilon).$$

Observe that for $y \in \mathbb{R}^{[n]}$, $\mathbb{B}(y,\varepsilon)$ is the n-dimensional sphere with origin y, with $\mathbb{B}(K,\varepsilon)$ being Minkowski addition of sphere of radius ε to K and $\mathbb{B}(K,-\varepsilon)$ being Minkowski subtraction of sphere of radius ε from K.

Notation 6 (Volumes)**.** *Given $K \subseteq \mathbb{R}^{[n]}$, we will use $\mathrm{vol}_d(K)$ to denote d-dimensional volume of K, provided it exists. Furthermore for any non-negative real $\varepsilon \geq 0$, let $\mathrm{vol}_d(\varepsilon)$ be the volume of d-dimensional ball of radius ε. We will use $\mathrm{vol}_d^{-1}(K)$ to denote the radius of a d-dimensional sphere whose volume is equal to $\mathrm{vol}_d(K)$ so that*

$$\mathrm{vol}_d(K) = \mathrm{vol}_d(\mathrm{vol}_d^{-1}(K)).$$

Notation 7 (Polytope)**.** *Given matrix $A \in \mathbb{R}^{m\times n}$ and vector $b \in \mathbb{R}^m$, let $\mathrm{poly}(A,b) \triangleq \left\{x \in \mathbb{R}^{[n]} | Ax \leq b\right\}$.*

Definition 8 (Separation Oracle)**.** *Given a convex body $K \subseteq \mathbb{R}^{[n]}$, $\textbf{\textit{SEP}}_\delta(y)$ is a separation oracle for K if the following holds. On inputs a rational vector $y \in \mathbb{Q}^{[n]}$ and rational number $\delta > 0$, $\textbf{\textit{SEP}}_\delta(y)$ asserts feasible if $y \in K$. Otherwise, if $y \notin K$, it returns c such that $\|c\|_\infty = 1$ and*

$$\forall x \in K : \langle c, x\rangle \leq \langle c, y\rangle + \delta.$$

We will use $T(\textbf{\textit{SEP}}_\delta)$ to denote the worst case running time of $\textbf{\textit{SEP}}_\delta$.

Theorem 9 (Central-Cut Ellipsoid Method [17])**.** *There exists an algorithm, called the central-cut ellipsoid method,*

$$\textbf{\textit{CCUT-E}}(\textbf{\textit{SEP}}_\delta, \Pi, y_0, \varepsilon_0)$$

that solves the following problem. Given a projection matrix $\Pi \in \mathbf{S}^{[n]}_+$ of rank m, vector $y^0 \in \Pi$, a convex body $K \subseteq [-\Delta,\Delta]^{[n]}$ for some positive Δ with $\textbf{\textit{SEP}}_\delta$ (see Definition 8) and rational number $\varepsilon_0 > 0$, it runs in time $|\log\Delta| N\left[\mathrm{poly}(n) + T(\textbf{\textit{SEP}}_{2^{-N}})\right]$, where $N \leq 6(n-m)(|\log\varepsilon_0| + (n-m))$, after which it outputs:

1) *Either a vector $a \in \mathbb{Q}^{[n]}$ such that $a \in K \cap \Pi^{-1}(y^0)$;*
2) *Or a polytope of the form $P = \mathrm{poly}(C,d)$, where $C \in \mathbb{Q}^{[N]\times[n]}, d \in \mathbb{Q}^{[N]}$ with $K \subseteq P$ and $\mathrm{vol}_{n-m}\left(P \cap \Pi^{-1}(y^0)\right) < \varepsilon_0$.*

Proof: Such algorithm can be obtained by trivial modifications to the central-cut ellipsoid algorithm [17], which we outline here. Handling the constraint $\Pi a = y^0$ can be done by projecting the covariance matrix of ellipsoid onto $\Pi^{-1}(y^0)$. At k^{th} iteration, for all k, we add hyperplanes returned by $\textbf{SEP}_\delta$ to P.

Algorithm terminates with a feasible $a \in \mathbb{Q}^{[n]}$ with $\Pi a = y^0$ only if $\textbf{SEP}_\delta(a)$ asserts feasible for some $\delta \leq \varepsilon_0$, in which case $a \in K$. Otherwise, when the maximum number of iterations is reached we simply return. ■

IV. Finding Separating Hyperplanes on a Subspace

We now describe our main technical contribution: An ellipsoid algorithm which can output a certificate of infeasibility **on a restricted subspace** using only the separation oracle $\textbf{SEP}_\delta$ as in Definition 8. The procedure uses the central-cut ellipsoid method [17] as a sub-routine. The main technical ingredient of our algorithm is Theorem 12, which is stated and proven in Section IV-A: It allows us to express this as another convex programming problem in terms of the "history" of constraints returned by separation oracle. Finally in Section IV-B we present our ellipsoid algorithm, bound its running time and prove its correctness.

A. An Equivalent Convex Problem

We first state some useful propositions. Recall our goal: Given convex body K, a subspace Π and $y_0 \in \Pi$, we have a polytope P separating various points $\{y\} \subset \Pi^{-1}(y_0)$ from K. We want to compute a separating hyperplane on Π. Our approach is formulated in Lemma 11, see also Figure 1. We first show that points interior in K have far off projections from Πy_0. The proof appears in the full version.

Lemma 10. *Given convex body $K \subseteq \mathbb{R}^{[n]}$, a projection matrix $\Pi \in \mathbf{S}_+^{[n]}$ with $\operatorname{rank}(\Pi) = m$, vector $y_0 \in \mathbb{R}^{[n]}$ and positive real $\delta > \operatorname{vol}_{n-m}^{-1}\left(\Pi^{-1}(y_0) \cap K\right)$, for all $y \in \mathbb{B}(K, -2\delta)$, we have $\|\Pi(y - y_0)\| \geq \delta$.*

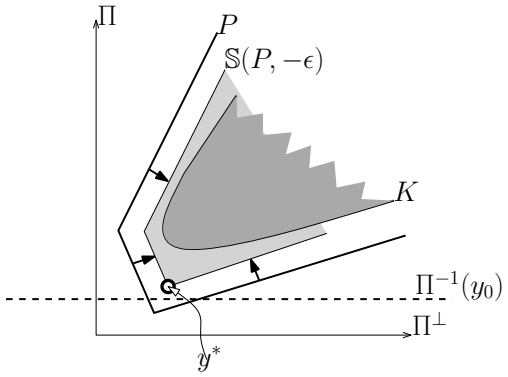

Fig. 1. We want to find a hyperplane parallel to $\Pi^{\perp}$ separating $\Pi^{-1}(y_0)$ and K, using only the inequalities returned by separation oracle, polytope P. The optimal solution of Lemma 11 is given by y^* with corresponding hyperplane $\Pi^{-1}(y^*)$.

Having shown that there is a δ-neighborhood of Πy_0 disjoint from interior of K whenever their intersection has small volume, we can immediately use Minkowski's Separating Hyperplane Theorem to infer the existence of such hyperplane. In fact, any hyperplane perpendicular to the line from y_0 to the closest point in K has this property. We formalize this in the following lemma, whose proof is deferred to the full version.

Lemma 11. *Given convex body $K \subseteq \mathbb{R}^{[n]}$, a projection matrix $\Pi \in \mathbf{S}_+^{[n]}$ with $\operatorname{rank}(\Pi) = m$, vector $y_0 \in \mathbb{R}^{[n]}$ and positive real $\delta > \operatorname{vol}_{n-m}^{-1}\left(\Pi^{-1}(y_0) \cap K\right)$, the hyperplane perpendicular to the projection of direction from y to closest point in the interior of ΠK separates y_0 and interior of ΠK:*

Formally any optimal solution y^* to eq. (3) satisfies eq. (4) for all $x \in \mathbb{B}(K, -2\delta)$:

$$\min \|\Pi(y - y_0)\|^2 \text{st } y \in \mathbb{B}(K, -2\delta), \tag{3}$$

$$\langle \Pi(y^* - y_0), x - y_0 \rangle \geq \|\Pi(y^* - y_0)\|^2. \tag{4}$$

Given Lemma 11, we can choose our separating hyperplane c as $c = -\frac{\Pi(y^* - y_0)}{\|\Pi(y^* - y_0)\|_\infty}$. But this does not quite work for two reasons:

1) Hyperplane c only separates "strict interior" of K as it is, whereas we need to separate K itself.
2) Depending on K, it might not be possible to represent optimal c using polynomially many bits, thus we need to account for near optimal solutions.

We now show how to overcome these problems.

Theorem 12. *Given convex body $K \subseteq [-\Delta, \Delta]^n$ for some $\Delta > 0$, a projection matrix $\Pi \in \mathbf{S}_+^{[n]}$ with $\operatorname{rank}(\Pi) = m$, a vector $y_0 \in [-\Delta, \Delta]^n$, for any positive real $\delta > 0$ with $\delta > \operatorname{vol}_{n-m}^{-1}\left(\Pi^{-1}(y_0) \cap K\right)$, the following holds: If y' is an $\frac{\delta^2}{2\Delta\sqrt{m}}$- approximate solution to eq. (5):*

$$\min \|\Pi(y - y_0)\|^2 \text{ st } y \in \mathbb{B}\left(K, -\left(2 + \frac{\delta}{2\sqrt{m}\Delta}\right) \cdot \delta\right) \tag{5}$$

then $\Pi(y' - y_0) \neq 0$ and for c being

$$c \triangleq -\frac{\Pi(y' - y_0)}{\|\Pi(y' - y_0)\|_\infty} \implies \forall x \in K : \langle c, x \rangle \leq \langle c, y_0 \rangle + 2\delta\sqrt{m}. \tag{6}$$

Proof: Before we begin, we set $\varepsilon \triangleq \frac{\delta}{2\sqrt{m}\Delta}$. Let y^* be an optimal solution of eq. (5) with $\|y^* - y'\| \leq \varepsilon\delta$. Since $y^* \in \mathbb{B}(K, -(2+\varepsilon)\delta)$, we have $\mathbb{B}(K, -2\delta) \supseteq \mathbb{B}(y^*, \varepsilon\delta) \ni y'$. By Lemma 10, this implies $\|\Pi(y' - y_0)\| \geq \delta$, proving $\Pi(y' - y_0) \neq 0$. For any $x \in K$, we can decompose x as $x = x' + z$ for some $x' \in \mathbb{B}(K, -2\delta)$ and $\|z\|_2 \leq 2\delta$. Then, $\langle \Pi(y' - y_0), x - y_0 \rangle$ can be expressed as:

$$\begin{aligned}
&\langle \Pi(y' - y_0), x' - y_0 \rangle + \langle \Pi(y' - y_0), z \rangle \\
&\geq \langle \Pi(y' - y^*), x' - y_0 \rangle + \langle \Pi(y^* - y_0), x' - y_0 \rangle \\
&\quad - \|z\| \cdot \|\Pi(y' - y_0)\| \\
&\geq -\underbrace{\|\Pi(y' - y^*)\|}_{\leq \varepsilon\delta} \underbrace{\|\Pi(x' - y_0)\|}_{\leq 2\Delta\sqrt{m}} \\
&\quad + \underbrace{\delta^2}_{\text{by Lemma 11}} - 2\delta\|\Pi(y' - y_0)\| \\
&\geq -2\delta\|\Pi(y' - y_0)\| \text{ (by the choice of } \varepsilon = \frac{\delta}{2\Delta\sqrt{m}}) \\
&\geq -2\delta\sqrt{m}\|\Pi(y' - y_0)\|_\infty.
\end{aligned}$$

Since $c = -\frac{\Pi(y' - y_0)}{\|\Pi(y' - y_0)\|_\infty}$, we have $\langle c, x \rangle \leq \langle c, y_0 \rangle + 2\delta\sqrt{m}$ for any $x \in K$. ∎

B. Ellipsoid Algorithm with Certificate of Infeasibility

Our solver is given in Algorithm 1.

The proof of the following theorem follows by combining various ingredients we have so far, especially Theorems 9 and 12. The proof is deferred to full version.

Theorem 13 (Main technical tool)**.** *Algorithm 1 runs in time $N \cdot T(\textbf{\textit{SEP}}_{2^{-N}}) + \operatorname{poly}(n) \log^2 \frac{1}{\varepsilon_0}$ where*

Algorithm 1 CERTIFY-E$(\mathbf{SEP}_\delta, \Pi, y_0, \varepsilon_0)$: Ellipsoid method with certificate of infeasibility.

Input. • Convex body $K \subseteq [0,1]^n$ and separation oracle $\mathbf{SEP}_\delta$ as in Definition 8,
• Projection matrix $\Pi \in \mathbf{S}_+^{[n]}$ with $\mathrm{rank}(\Pi) = m$, $y_0 \in \mathbb{R}^{[n]}$ and positive real $0 < \varepsilon_0 < 1$.

Output. • Either a vector $y \in \mathbb{Q}^n$ st $y \in K \cap \Pi^{-1}(y_0)$.
• Or $c \in \Pi$ with $\|c\|_\infty = 1$ and $\forall x \in K : \langle c, x\rangle \le \varepsilon_0 + \langle c, y_0\rangle$.

Procedure. 1. Run **CCUT-E**$(\mathbf{SEP}_\delta, \Pi, y_0, \varepsilon)$ where $\varepsilon \leftarrow \mathrm{vol}_{n-m}\left(\frac{\varepsilon_0}{2\sqrt{m}}\right)$.
2. If it returns $y \in K \cap \Pi^{-1}(y_0)$, then return y.
3. Else let $P = \mathrm{poly}(C, d)$ be the polytope it returns. Set $\delta \leftarrow \frac{\varepsilon_0}{2\sqrt{m}}$. $\varepsilon' \leftarrow \frac{\delta}{2\Delta\sqrt{m}}$ and
4. Solve eq. (7) using regular ellipsoid method to find an $\varepsilon'\delta$-approximate solution, $y^* \in \mathbb{Q}^n$:

$$\text{Minimize } \|\Pi(y - y_0)\|^2 \text{ subject to } Cy \le d - (2 + \varepsilon')\delta\sqrt{\mathrm{diag}(C^T C)}. \tag{7}$$

5. Return $c \leftarrow -\frac{\Pi(y^* - y_0)}{\|\Pi(y^* - y_0)\|_\infty}$.

$N = O\left((\textit{\# of free variables})^2 \log \frac{\textit{\# of fixed variables}}{\varepsilon_0}\right) = O\left((n-m)^2 \log \frac{m}{\varepsilon_0}\right)$, *and provides the following guarantee: If* $\mathrm{vol}_{n-m}^{-1}(K \cap \Pi^{-1}(y_0)) > \frac{\varepsilon_0}{2\sqrt{m}}$ *then it outputs* $y \in K \cap \Pi^{-1}(y_0)$.

V. Faster Solver for Local Rounding Algorithms

We return back to our motivating example. Assume we have n variables, and we want to find a discrete labeling $\widetilde{x} \in L^{[n]}$ of those from a set of labels L, under various constraints and objective. Suppose we "lifted" this problem into a higher dimension $\mathbb{R}^N$ where $|N| \gg n$, and obtained a family of increasingly tight convex relaxations defined over various subspaces of $\mathbb{R}^N$.

Formally, we have a set of subspaces $\{\Pi_S\}_{S \subseteq [N]}$, represented by their projection matrices and associated with subsets of $[N]$, and with each subspace Π_S, we have an associated convex body, $K_S \subseteq \Pi_S[0,1]^N$ with such that

$$\Pi_S \subseteq \Pi_T \implies \Pi_S K_T \subseteq K_S.$$

We are given functions **FEASIBLE**, **ROUND** and **SEED**, along with positive integers n, s such that:

- $\mathbf{FEASIBLE}_S : \Pi_S\mathbb{Q}^N \to \{\text{feasible}, \Pi_S\mathbb{Q}^N\}$. On input $S \subseteq N$,$y \in \Pi_S Q^N$, it asserts feasible if $y \in K_S$ or returns $c \in \Pi_S\mathbb{Q}^N : \|c\|_\infty = 1$ such that[4] $\forall x \in K_S : \langle c, x\rangle < \langle c, y\rangle$ in time $\mathrm{poly}(\mathrm{rank}(\Pi_S))$.
- $\mathbf{SEED}_S : K_S \to 2^N$. Given $S \subseteq [N]$ and $y \in \Pi_S\mathbb{Q}^N$, it returns subset $S' \supseteq S$ such that $\frac{\mathrm{rank}(\Pi_{S'})}{\mathrm{rank}(\Pi_S)} \le s$ when $S \neq \emptyset$, and $\mathrm{rank}(\Pi_{S'}) \le n$ when $S = \emptyset$. Its worst case running time is bounded by $\mathrm{poly}(\mathrm{rank}(\Pi_S))$ (or $\mathrm{poly}(n)$ in the case of $S = \emptyset$).
- $\mathbf{ROUND}_S : K_S \to L^{[n]}$. On inputs $S \subseteq N$ and $y \in K_S$, returns an approximation to the original problem in time $\mathrm{poly}(\mathrm{rank}(\Pi_S))$.

We now describe our main solver. Note that once the algorithm outputs $y^* \in K_{S(\ell)}$, the final output labeling will simply be $\mathbf{ROUND}_{S(\ell)}(y^*)$.

Algorithm 3 SEP$_{S(i),\varepsilon_0}(y)$:Separation Oracle.

Input. • Positive real $\varepsilon_0 > 0$, current iteration i, current seeds $S(i)$, vector $y \in \mathbb{Q}^{S(i)}$.

Output. • Either asserts feasible, and sets values of global variables $S(i+1), \ldots, S(\ell)$ along with y^* so that:
1) $\Pi_{S(j)} y^* \in K_{S(j)}$ for all $j : i \le j \le \ell$,
2) $S(j+1) = \mathbf{SEED}_{S(j)}(y^*)$ for all $j : i \le j \le \ell - 1$.
• Or returns $c \in \Pi_{S(i)}$ with $\|c\|_\infty = 1$ such that $\forall x \in K_{S(i)} : \langle c, x - y\rangle < \varepsilon_0$.

Procedure. 1. If $\mathbf{FEASIBLE}_{S(i)}(y)$ returns $c \in \Pi_{S(i)}$, return c.
2. Else if $i \ge \ell$, set $y^* \leftarrow y$. Assert feasible and return.
3. $S(i+1) \leftarrow \mathbf{SEED}_\emptyset(y)$.
4. Run **CERTIFY-E**$(\mathbf{SEP}_{\Pi_{S(i+1)},\delta}, \Pi_{S(i)}, y, \varepsilon_0)$ (see Algorithm 1).
5. If it returns c, return c.
6. Else assert feasible.

The proof of the following theorem is easy given the ingredients so far, and is given in the full version.

Theorem 14. *Algorithm 2 runs in time* $\left[s^\ell n \log(1/\varepsilon_0)\right]^{O(\ell)}$ *(compare this with the straightforward algorithm which runs in time* $N^{O(1)} \log(1/\varepsilon_0)$*) and achieves the following guarantee:*

[4]We can handle $\mathbf{FEASIBLE}_S$ that only returns a weak separation oracle, but since in our application to SDPs we have access to a strong separation oracle, we assume this for simplicity.

Algorithm 2 Fast Solver (to fool local rounding algorithm)

Input. • Maximum number of iterations ℓ and positive real $\varepsilon_0 > 0$,
• n, r, $(K_S)_{S\subseteq[N]}$ with separation oracle **FEASIBLE**, **SEED**, $\Pi_\emptyset$ and $y(0) \in K_\emptyset \mathbb{Q}^N$ all as described in Section V.

Output. • Either asserts $\operatorname{vol} K \le \varepsilon_0$,
• Or outputs $y^* \in K_{S(\ell)}$ and $S(0), \dots, S(\ell)$ st for all i: 1) $\Pi_{S(i)} y^* \in K_{S(i)}$; 2) $S(i+1) = \textbf{SEED}_{S(i)}(y^*)$.

Procedure. 1. Initialize global variables $S(1), \dots, S(\ell)$ representing seed sets and global sparse vector $y^* \in \mathbb{Q}^{[N]}$ representing the final solution (it will be in span of $\Pi_{S(\ell)}$).
2. Set $S(0) \leftarrow \{\emptyset\}$.
3. Run **CCUT-E**($\textbf{SEP}_{S(0),\delta}, 0, 0, \varepsilon_0$) (see Theorem 9) where **SEP** is given in Algorithm 3.
4. If it asserts feasible, output $S(0), \dots, S(\ell)$ and y^*.
5. Else assert $\operatorname{vol} K \le \varepsilon_0$.

Provided that $\operatorname{vol} K > \varepsilon_0$*, it outputs* $y^* \in K_{S(\ell)}$ *and* $S(0), \dots, S(\ell)$ *st for all* i*:*

$$\Pi_{S(i)} y^* \in K_{S(i)}, \tag{8}$$
$$S(i+1) = \textbf{SEED}_{S(i)}(y^*). \tag{9}$$

Otherwise it asserts $\operatorname{vol} K \le \varepsilon_0$.

Furthermore there is no algorithm which runs in time $n^{o(\ell)}$ *assuming Exponential Time Hypothesis.*

We will review Lasserre Hierarchy in Section VI and finally in Section VII, we will show a sample of how some approximation algorithms using Lasserre Hierarchy relaxation fit into our framework.

VI. Lasserre Hierarchy

In this section, we present a general derivation of Lasserre Hierarchy relaxation. Although our relaxation is only given for $0/1$ programming problems, it can easily be adapted to work on any set of finite labels.

A. Preliminaries

For some positive integer d, let $Q \in \mathbb{R}^{[n]\le d}$ (resp. $\mathcal{P} = \{P_1, \dots, P_M\} \subset \mathbb{R}^{[n]\le d}$) be a degree $\le d$ multilinear polynomial (resp. subset of degree $\le d$ multilinear polynomials) over n variables representing objective function (resp. constraints). We want to find a binary labeling of n variables, $\widetilde{x} \in \{0,1\}^n$, which satisfies eq. (10):

$$\begin{aligned} \min_{\widetilde{x}\in\{0,1\}^n} \quad & \textstyle\sum_{S\in[n]\le d} Q_S \prod_{u\in S} \widetilde{x}_u \\ \text{st} \quad & \textstyle\sum_{S\in[n]\le d} P_S \prod_{u\in S} \widetilde{x}_u \ge 0 \text{ for all } P \in \mathcal{P}. \end{aligned} \tag{10}$$

Observe that we can convert any constraint satisfaction problem on $\{0,1\}^{[n]}$ to this form easily. In this section, we will first express eq. (10) as an **equivalent** SDP problem, from which Lasserre relaxation can be obtained by enforcing positivity only on certain principal minors of this matrix. This exposition of Lasserre relaxation through constraints on certain minors will be convenient when we are presenting our partial SDP solver.

Notation 15. *Given positive integers* n, d*, for any vector* $P \in \mathbb{R}^{[n]\le d}$ *and* $y \in \mathbb{R}^{2^{[n]}}$*, we define* $P * y \in \mathbb{R}^{2^{[n]}}$ *as* $(P*y)_S \triangleq \sum_{T\in[n]\le d} P_T y_{T\cup S}$.

Definition 16 (Multivariate Moment Matrix on $\mathcal{P}$ and Q)**.** *Given positive integer* n *let* $\mathbb{M}^n : \mathbb{R}^{2^{[n]}} \to \mathbb{R}^{2^{[n]}\times 2^{[n]}}$ *be the following linear matrix function:*

$$\mathbb{M}^n(y) = [y_{A\cup B}]_{A,B\subseteq[n]}.$$

For any $\mathcal{P} = \{P_1, \dots, P_M\} \subseteq \mathbb{R}^{[n]\le d}$ *representing constraints,* $Q \in \mathbb{R}^{[n]\le d}$ *and* $q \in \mathbb{R}$ *representing objective polynomial and a guess for its value, let* $\mathbb{M}^{n,\mathcal{P}}_{Q,q} : \mathbb{R}^{2^{[n]}} \to \mathbb{R}^{([m+2]\times 2^{[n]})\times([m+2]\times 2^{[n]})}$ *be the following linear function on block diagonal matrices:*

$$\mathbb{M}^{n,\mathcal{P}}_{Q,q}(y) \triangleq \begin{bmatrix} \mathbb{M}^n(y) & 0 & 0 & \\ 0 & qy_\emptyset - \langle Q, y\rangle & 0 & \cdots \\ 0 & 0 & \mathbb{M}^n(P_1 * y) & \\ & \vdots & & \ddots \end{bmatrix}.$$

Having defined the moment matrix, we can state the following:

Theorem 17 (See [3])**.** *eq.* (10) *has optimal value* $\le q$ *if and only if there exists* y *with* $y \ne 0$ *such that* $\mathbb{M}^{n,\mathcal{P}}_{Q,q}(y) \succeq 0$.

Definition 18 (Principal Minors of Multivariate Moment Matrices)**.** *Given* $\mathcal{P} \subseteq \mathbb{R}^{[n]\le d}$, $Q \in \mathbb{R}^{[n]\le d}$ *and* $q \in \mathbb{R}$ *as described, for any* $T : T \subseteq 2^{[n]}, T \supseteq \operatorname{support}(Q)$ *being a family of sets over* $[n]$ *containing the support of* Q*, we will refer to the following principal minor of* $\mathbb{M}^{n,\mathcal{P}}_{Q,q}(y)$ *as* $\mathbb{M}\big|^n_T(y)$*:*

$$\begin{bmatrix} (\mathbb{M}^n(y))_{T,T} & 0 & 0 & \\ 0 & qy_\emptyset - \langle Q, y\rangle & 0 & \cdots \\ 0 & 0 & (\mathbb{M}^n(P_1 * y))_{T,T} & \\ & \vdots & & \ddots \end{bmatrix}.$$

Observation 19. *Given a family of sets over $[n]$, $T \subseteq 2^{[n]}$, for any $P \in \mathbb{R}^{[n]_{\leq d}}$, the minor $(\mathbb{M}^n(P*y))_{T,T}$ is only a function of $y_{\mathrm{ex}(T,d)}$ where*

$$\mathrm{ex}(T,d) \triangleq \{A \cup B \cup C : A \in T, B \in T, C \in [n]_{\leq d}\}.$$

Proof: Consider $(\mathbb{M}^n(P*y))_{A,B}$ with $A, B \in U$. By Definition 16, this is equal to

$$(\mathbb{M}^n(P*y))_{A,B} = (P*y)_{A\cup B} = \sum_{C \in [n]_{\leq d}} P_C y_{A\cup B\cup C},$$

where $A \cup B \cup C \in \mathrm{ex}(U,d)$ by definition. The second part follows immediately from the definition of $\mathbb{M}\big|^n_T(y)$. ■

Using Theorem 17 and observation 19, we can easily state and prove the following:

Theorem 20 (Lasserre Relaxation [3])**.** *Given positive integers n, r and d, polynomials $\mathcal{P} \subset \mathbb{R}^{[n]_{\leq d}}$ and $Q \in \mathbb{R}^{[n]_{\leq d}}$ with $q \in \mathbb{R}$, the following is r^{th} round Lasserre Hierarchy relaxation of eq. (10):*

$$\textit{Find } y \in \mathbb{R}^{[n]_{\leq 2r+d}} \textit{ st } \mathbb{M}\Big|^n_{[n]_{\leq r}}(y) \succeq 0 \textit{ and } y_\emptyset = 1. \tag{11}$$

Note that the straightforward SDP relaxation of eq. (10) corresponds to d rounds of Lasserre hierarchy relaxation (and the "basic" SDP relaxation for the case of quadratic polynomials).

B. Separation Oracle for Lasserre Hierarchy

Given a binary labeling problem, we first cast $\tilde{r}$ rounds of Lasserre Hierarchy relaxation in our framework:

- The set of labels is $L = \{0,1\}$.
- Lifted space N is $\binom{[n]}{\leq r'}$, the subsets of $[n]$ of size at most r',
- For any subset $S \subseteq [n]$, we define Π_S as the projection matrix onto $\mathbb{R}^{\mathrm{ex}(S,2)}$ so that

$$[\Pi_S x]_T = \begin{cases} 1 & \text{if } T \in \mathrm{ex}(S,2), \\ 0 & \text{else.} \end{cases}$$

The associated convex body, K_S, is defined as

$$K_S = \left\{ y \in \mathbb{R}^{\mathrm{ex}(S,2)} : y_\emptyset = 1,\ \mathbb{M}\big|^n_{\mathrm{ex}(S,2)}(y) \succeq 0 \right\}.$$

Before stating the **FEASIBLE** procedure, we need the following well known result:

Proposition 21. *Given a symmetric matrix $A \in \mathbf{S}^B$, there exists an algorithm which asserts if $A \succeq 0$ or returns $x \in \mathbb{Q}^B$ such that $x^T A x < 0$ in time at most polynomial in size of A.*

Then our **FEASIBLE**$_S(y)$ procedure is trivial: It asserts feasible if $\mathbb{M}\big|^n_{\mathrm{ex}(S,2)}(y) \succeq 0$. Else it returns $x \in \mathbb{Q}^{\mathrm{ex}(S,2)}$ for which $x^T \mathbb{M}\big|^n_{\mathrm{ex}(S,2)}(y) x < 0$.

VII. Approximation Algorithms for Combinatorial Optimization Problems

In this section, we finally apply our main algorithm as given in Algorithm 2 to various rounding algorithms for Lasserre Hierarchy relaxations as stated in the introduction. For all these problems, our separation oracle is the same procedure as described in Section VI-B. The running times we obtained as well as approximation factors and other guarantees are summarized in Figure 2. The last two columns list the value of s (the factor by which $\mathrm{rank}(\Pi_S)$ increases in each step of seed selection) and ℓ (the number of iterations of seed selection) used by the rounding algorithm in each case. The parameter r refers to the index of the eigenvalue governing the approximation guarantee, and ε is a positive parameter.

The claimed running times follow from the $\approx s^{O(\ell^2)} n^{O(\ell)}$ runtime guaranteed by Theorem 14 for our solver (Algorithm 2). The rounding algorithm in each case runs within the same time. For problems marked with *, check the caption for required conditions.

Below, in Algorithms 4 and 5, we present seed selection procedures for binary graph partitioning algorithms from [11] and semi-coloring algorithm from [14], respectively.

Algorithm 4 SEED-QIP$_S(y)$: Seed selection procedure for approximation algorithms given in [11] for two-way partitioning problems.

Input. Subset $S \subseteq [n]$, and $y \in \mathbb{Q}^{\mathrm{ex}(S,2)}$ provided $\mathbb{M}\big|^{[n]}_{\mathrm{ex}(S,2)}(y) \succeq 0$ and $y_\emptyset = 1$, positive integer r'.

Output. T with $|T| \leq r' \cdot |S|$.

Procedure. 1. Let $(x_T)_T$ be vectors corresponding to Cholesky factorization of $\mathbb{M}\big|^{[n]}_{\mathrm{ex}(S,2)}(y)$. For each $S \in \mathrm{ex}(S,2)$ and $f \in \{0,1\}^S$, set

$$x_S(f) \leftarrow \sum_{T \subseteq f^{-1}(0)} (-1)^{|T|} x_{f^{-1}(1)\cup T}.$$

2. Let $\Pi_S^\perp \leftarrow I - \sum_{f\in\{0,1\}^S} \frac{1}{\|x_S(f)\|^2} x_S(f) x_S(f)^T$.
3. Use volume sampling [19], [20] to choose S': an r'-subset of vectors from $\left(\Pi_S^\perp x_u(1)\right)_{u\in[n]}$.
4. Return $S \cup S'$.

The works [12] and [14] use greedy seed selection, but these can be replaced by the above column selection procedure as well (we describe the procedure for the semi-coloring algorithm in [14] below). We conclude the paper by listing some Lasserre based approximation algorithms for which we are not able to get a runtime improvement: the algorithm for independent sets in 3-uniform hypergraphs [9], and the algorithm for directed

Problem Name		Running Time	OPT	Rounding	s	ℓ
Maximum Cut	[11]	$2^{O(r/\varepsilon^3)}n^{O(1/\varepsilon)}$	$1-\eta$	$1-\frac{1+\varepsilon}{\lambda_{n-r}}\eta$	$2^{O(r/\varepsilon)}$	$O(1/\varepsilon)$
k-Unique Games	[11]	$k^{O(r/\varepsilon)}n^{O(1)}$	$1-\eta$	$1-\frac{2+\varepsilon}{\lambda_r}\eta$	$k^{O(r/\varepsilon)}$	1
Minimum Bisection	[11]	$2^{O(r/\varepsilon^3)}n^{O(1/\varepsilon)}$	η	$\frac{1+\varepsilon}{\lambda_r}\eta - o(1)$	$2^{O(r/\varepsilon)}$	$O(1/\varepsilon)$
Maximum Bisection	[11]	$2^{O(r/\varepsilon^3)}n^{O(1/\varepsilon)}$	$1-\eta$	$1-\frac{1+\varepsilon}{\lambda_{n-r}}\eta - o(1)$	$2^{O(r/\varepsilon)}$	$O(1/\varepsilon)$
Sparsest Cut*	[15]	$2^{O(r/\varepsilon)}n^{O(1)}$	η	$\frac{\eta}{\varepsilon}$	$2^{O(r/\varepsilon)}$	1
Independent Set*	[11]	$2^{O(r)}n^{O(1)}$	η	$\Omega(\eta)$	$2^{O(r)}$	$O(1)$
	[14]	$2^{O(r)}n^{O(1)}$	η	$\frac{\eta}{16}$	$2^{O(r)}$	1
Maximum 2-CSPs*	[12]	$k^{O(rk^2/\varepsilon^3)}n^{O(k/\varepsilon^2)}$	η	$\eta-\varepsilon$	$k^{O(r/\varepsilon)}$	$O\left(k/\varepsilon^2\right)$

Fig. 2. Running times and approximation guarantees for various Lasserre Hierarchy relaxation rounding algorithms using our faster solver. For sparsest cut, the spectral assumption is $\lambda_r \geq \eta/(1-\varepsilon)$. For independent set [11], the spectral assumption is $\lambda_{n-r} \leq 1 + O(1/\Delta)$ where Δ is the maximum degree. For independent set [14], the assumptions are that G is 3-colorable and $\lambda_{n-r} \leq 17/16$. Finally for maximum 2-CSPs [12], the assumption is that $\lambda_r \geq 1 - \Omega(k^2/\varepsilon^2)$.

Algorithm 5 SEED-COLOR$_S(y)$: Seed selection procedure for semi-coloring algorithm as given in [14] on graph G.

Input. Graph G on nodes $[n]$, positive integer r'.

Output. $S \in [n]_{\leq r'}$.

Procedure. 1. Let $X_u \leftarrow \sum_{i=1}^{3} e_i \otimes x_\emptyset^\perp x_u(i)$.
2. Use volume sampling [19], [20] to choose S, an r'-subset of vectors from $(X_u)_{u\in[n]}$.
3. Return S.

Steiner tree [21]. This is because these algorithms are adaptive with a large number of stages ℓ in the rounding procedure.

Acknowledgments

We thank Anupam Gupta and László Lovász for useful discussions.

References

[1] H. D. Sherali and W. P. Adams, "A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems," *SIAM J. Discrete Mathematics*, vol. 3, pp. 411–430, 1990. 1

[2] L. Lovász and A. Schrijver, "Cones of matrices and set-functions and 0-1 optimization," *SIAM J. Optimization*, vol. 1, pp. 166–190, 1991. 1

[3] J. B. Lasserre, "An explicit equivalent positive semidefinite program for nonlinear 0-1 programs," *SIAM J. Optimization*, vol. 12, no. 3, pp. 756–769, 2002. 1, 8, 9

[4] E. Chlamtac and M. Tulsiani, "Convex relaxations and integrality gaps," in *Handbook on Semidefinite, Cone and Polynomial Optimization*, 2011. [Online]. Available: http://www.cs.princeton.edu/~chlamtac/sdpchapter.pdf 1

[5] M. Laurent, "A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0-1 programming," *Math. Oper. Res.*, vol. 28, no. 3, pp. 470–496, 2003. 1

[6] B. Barak, A. Harrow, J. Kelner, D. Steurer, and Y. Zhou, "Hypercontractivity, Sum-of-Squares proofs, and their applications," in *STOC*, 2012, pp. 307–326. 2

[7] S. Khot and R. Saket, "SDP integrality gaps with local ℓ_1-embeddability," in *FOCS*, 2009, pp. 565–574. 2

[8] P. Raghavendra and D. Steurer, "Integrality gaps for strong SDP relaxations of Unique Games," in *FOCS*, 2009, pp. 575–585. 2

[9] E. Chlamtac and G. Singh, "Improved approximation guarantees through higher levels of SDP hierarchies," in *APPROX-RANDOM*, 2008, pp. 49–62. 2, 3, 9

[10] A. R. Karlin, C. Mathieu, and C. T. Nguyen, "Integrality gaps of linear and semi-definite programming relaxations for knapsack," *CoRR*, vol. abs/1007.1283, 2010. 2, 3

[11] V. Guruswami and A. K. Sinop, "Lasserre hierarchy, higher eigenvalues, and approximation schemes for graph partitioning and quadratic integer programming with PSD objectives," in *FOCS*, 2011, pp. 482–491. 2, 3, 9, 10

[12] B. Barak, P. Raghavendra, and D. Steurer, "Rounding semidefinite programming hierarchies via global correlation," in *FOCS*, 2011, pp. 472–481. 2, 3, 9, 10

[13] P. Raghavendra and N. Tan, "Approximating CSPs with global cardinality constraints using SDP hierarchies," in *SODA*, 2012. 2, 3

[14] S. Arora and R. Ge, "New tools for graph coloring," in *APPROX-RANDOM*, 2011, pp. 1–12. 2, 3, 9, 10

[15] V. Guruswami and A. K. Sinop, "Lasserre SDPs, ℓ_1-embeddings, and approximating non-uniform sparsest cut via generalized spectra," 2012, submitted. 2, 3, 10

[16] E. Chlamtac, R. Krauthgamer, and P. Raghavendra, "Approximating sparsest cut in graphs of bounded treewidth," in *APPROX-RANDOM*, 2010, pp. 124–137. 2

[17] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, 1993. 4, 5

[18] L. Porkolab and L. Khachiyan, "On the complexity of semidefinite programs," *J. Global Optimization*, vol. 10, pp. 351–365, 1997. 4

[19] A. Deshpande and L. Rademacher, "Efficient volume sampling for row/column subset selection," in *FOCS*, 2010, pp. 329–338. 9, 10

[20] V. Guruswami and A. K. Sinop, "Optimal column-based low-rank matrix reconstruction," in *SODA*, 2012, pp. 1207–1214. 9, 10

[21] T. Rothvoß, "Directed Steiner Tree and the Lasserre hierarchy," *CoRR*, vol. abs/1111.5473, 2011. 10