1. Linear relaxations of integer programs.

Lecturer: Massimo Lauria

We can express combinatorial problems using integer programs and, since we can't solve them, we consider relaxed programs in which integer constraints are relaxed to linear ones. Then we round fractional solutions to integer. We consider hierarchies of linear programs and discuss the quality of the corresponding solutions.

This lecture is a sort of scaled down demo of the rest of the course. Here we see that we can express decisions and optimization problems by the means of *integer programs*. This translation works for NP-hard problems, thus there cannot be efficient algorithms to solve integer programs unless $\mathsf{P} = \mathsf{NP}$, which is considered by many to be very unlikely[1].

In any case there is no **known** efficient algorithm that solves integer programs; a viable strategy is to *relax* the integer programs to something easier to manage: for example liner programs.

The most naive way to do that is to transform the integral constraints into fractional linear constraints, e.g. $x \in \{0,1\}$ into $0 \leq x \leq 1$, and leave the other constraints alone[2].

Once we relax the integer program we have a lot of new fractional solutions that were not allowed before. Consider, for example, the program that characterizes the maximum independent sets of graph K_3, i.e., the triangle.

$$\begin{aligned} \text{maximize} \quad & x_1 + x_2 + x_3 \\ \text{subject to} \quad & x_1 + x_2 \leq 1 \\ & x_2 + x_3 \leq 1 \\ & x_1 + x_3 \leq 1 \\ & x_1 \in \{0,1\},\ x_2 \in \{0,1\},\ x_3 \in \{0,1\}. \end{aligned} \tag{1}$$

Its *integer optimum* is 1, but if we relax the constraints and allow $0 \leq x_i \leq 1$, then the linear program has a *fractional optimum* of $\frac{3}{2}$, by setting all variables to $\frac{1}{2}$.

During most of the course we will study systematic techniques to improve the relaxation: we add variables and inequalities in order to **cut away** feasible fractional solutions without changing the set of integer solutions. The quality and the complexity of such techniques is controlled by a parameter called *rank*: the larger the rank, the fewer fractional solutions remain.

Linear programming

The most studied optimization formalism is *linear programming*, which is illustrated by books as [3]: we want to optimize (either minimize of maximize) a linear function $\sum_i c_i x_i$ over the set of variables $\{x_i\}_{i=1}^n$, which are constrained to satisfy a set of linear inequalities and linear equations. Without

[1] Most of the hardness results we will see during the course won't rely on any unproved assumption.

[2] We can assume that all such constraints are affine, namely they have one of the three forms

$$\sum_i a_i x_i \leq b \quad \sum_i a_i x_i \geq b \quad \sum_i a_i x_i = b$$

for a_i and b in $\mathbb{R}$

[3] Jiří Matoušek and Bernd Gärtner. *Understanding and using linear programming*. Springer, Berlin New York, 2007

loss of generality we can assume a linear program to have one of the following forms[4].

[4] While we will discuss the left form more often. The right one is also very common and is called the *standard form*. The standard form is particularly useful in implementations of the simplex algorithm.

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \leq b \end{array} \qquad \begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array} \tag{2}$$

In order to optimize a linear program it we need to check if the program is **feasible**, i.e. if there is a value of x that satisfies the linear constraints. Indeed we can use linear programs to describe either **decision** or **optimization** problems.

The striking feature of linear programming is the possibility to **witness** the unsatisfiability of a set of linear inequalities. This feature is formalized by *Farkas' Lemma*: as it happens often with fundamental results, there are several ways to state Farkas' Lemma.

Lemma 1 (Farkas' Lemma for linear programming)**.** *A set of linear inequalities $Ax \leq b$ is unsatisfiable if and only if there exists a positive vector $y \geq 0$ such that $y^T A = 0$ and $y^T b = -1$.*

Two inequalities $a^T x \leq b$ and $c^T x \leq d$ entail any positive combination as a logical consequence. Thus we can design a proof system LP for sets of linear inequalities with the following inference rule,

$$\frac{a^T x \leq b \quad c^T x \leq d}{(\alpha a + \gamma c)x \leq (\alpha b + \gamma d)x}$$

with $\alpha \geq 0, \gamma \geq 0$. Farkas' Lemma claims that such proof system can deduce the contradiction $0 \leq -1$ from any unsatisfiable set of linear inequalities.

If the linear program represents an optimization problem we can even take positive combinations to prove bounds on the function to be optimized. We start with the *primal* program that asks to maximize $c^T x$ under constraints $Ax \leq b$ and $x \geq 0$ (notice that we highlighted the non negativity constraints). Now consider a non negative vector y^T such that $y^T A \geq c^T$. For every feasible solution x of the primal program it holds that

$$c^T x \leq y^T A x \leq y^T b. \qquad \text{(weak duality)}$$

Thus the solution $y \geq 0$ witnesses the **upper bound** $b^T y$ to the maximum achievable in the primal program. **How tight is such upper bound?** We can answer that by looking at *dual* program (D). Notice also that the dual of (D) is (P).

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \leq b \\ & x \geq 0 \end{array} \quad \text{(P)} \qquad \begin{array}{ll} \text{minimize} & b^T y \\ \text{subject to} & A^T y \geq c \\ & y \geq 0 \end{array} \quad \text{(D)}$$

Farkas' Lemma gives a complete solution of the decision case, but Farkas' Lemma also implies the *duality theorem*, which basically claims that there are solutions of the dual program that witness tight bounds for the primal program (and vice versa).

Theorem 2 claims that proof system LP can prove all valid linear inequalities. In proof theoretic jargon: Farkas' Lemma is the completeness of LP, while Theorem 2 is the implicational completeness of LP.

Theorem 2 (Duality theorem)**.** *Consider the linear programs* (P) *and* (D)*, exactly one of the following holds*

- *neither* (P) *nor* (D) *have a feasible solution;*
- *program* (P) *has solutions with arbitrarily large values, and program* (D) *is unsatisfiable;*
- *program* (D) *has solutions with arbitrarily small values, and program* (P) *is unsatisfiable;*
- *both* (P) *and* (D) *have optimal solutions. Let* x^* *and* y^* *such solutions, then*

$$c^T x^* = b^T y^*.$$

Complexity of Linear programming

Deciding the satisfiability of a set of linear inequalities is clearly in NP. Farkas' Lemma and Duality Theorem show that deciding its unsatisfiability is also in NP, thus the problem is in $\mathsf{NP} \cap \mathsf{coNP}$. But actually there are well known efficient algorithms for this problem. All of them are based the geometric interpretation of the system of linear inequalities as a convex polyhedron.

- The *simplex method* is the first algorithm for linear programming and has been invented by Dantzig (1947). It does not run in polynomial time, but it is quite fast in practice. The idea is to walk on the edges of the polyhedron induced by the linear program, in order to reach the vertex with optimal value.
- The first polynomial time algorithm for linear programming is based on the *ellipsoid method* (Khachyian, 1979). The simplex method is much faster in practice, but of course the ellipsoid method has great theoretical value, and runs under more general conditions. For example it allows to optimize over super polynomial size linear programs in polynomial time, assuming the program has an efficient *separator*[5]. The algorithm assumes that the set of feasible solutions is in a 0-centered ball of large enough radius, and that it completely contains some ball of range ϵ. The algorithm checks whether the center of the ball is feasible. If it is not then the separator oracle suggests which half of the ball to save and which to forget. A new ellipsoid of smaller volume "encircles" the useful half. The idea is to encircle the set of feasible solutions with smaller and smaller ellipsoids, until either the center of the last ellipsoid is a feasible solution or the ellipsoid is so small that cannot contain a ball of range ϵ.
- Since the ellipsoid method is so inefficient in practice, the simplex method is usually preferred. The advent of *interior point methods* changed the scenario, providing a family of polynomial time algorithms which are fast in practice. The first member of this family is due to Karmakar (1984): his algorithm is based on the idea of exploring the space of solutions by walking through the interior of the polytope. The original algorithm was not faster than the simplex method but nowadays there are competitive implementations and variants. We must observe that interior point methods were

[5] A separator is an oracle that, given a point x outside the set of feasible solutions F, outputs an hyperplane separating x from F.

already employed in the context of non-linear programming way before Karmakar algorithm.

Integer programs and Linear relaxations

Most problems in computer science require solutions that are discrete, integer or $\{0,1\}$. It is not possible express such constraints as linear inequalities, so we need a new formalism called *integer programming*. An integer program may have the form

$$\begin{aligned} \text{maximize} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x_i \in \{0,1\} \text{ or } x_i \in \mathbb{Z}. \end{aligned}$$

It is pretty clear that integer programs are in NP, and that they are expressive enough to encode NP-complete problems, thus integer programming is NP-complete. With a *linear relaxation* we trade expressibility for efficiency: we drop all non-linear constraints and we substitute them with fractional ones (e.g., $x_i \in \mathbb{Z}$ becomes $x_i \in \mathbb{R}$ and $x_i \in \{0,1\}$ becomes $0 \leq x_i \leq 1$).

Example 3 (*Maximum weighted bipartite matching*)**.** Consider the bipartite graph $G = (L, R, E)$, where L and R are two disjoint sets of vertices and $E \subseteq L \times R$, and consider a weight function $w : E \longrightarrow \mathbb{R}$. The Maximum weighted bipartite matching problem looks for the matching of largest weight in the bipartite graph, and we can represent it with an integer program

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} w(e) x_e \\ \text{subject to} \quad & \sum_{e \ni v} x_e = 1 \quad \text{for each } v \in L \cup R \\ & x_e \in \{0,1\}. \end{aligned}$$

This is one of the few cases for which the integer program and its linear relaxation have the **same optimum value**.

The reason for the tightness of the linear relaxation of Example 3 is in the following property of the matrix of constraints.

The criterion of total unimodularity given in Definition 4 is not efficiently verifiable. Instead there are other sufficient criteria which can be verified in polynomial time.

Definition 4. *A matrix M is called* totally unimodular *if every square submatrix of M has determinant in $\{-1,0,1\}$.*

Theorem 5. *Consider the polyhedron $P = \{x | Ax = b, x \geq 0\}$. If A is totally unimodular and b is integer then the vertices of P are all integers.*

The previous theorem implies that an integer program with total unimodular matrix of constraints can be efficiently solved by solving its linear relaxation.

Integrality gaps and Rounding

In the case of Example 3 we saw that the integer program and its linear relaxation have the same optimum, but this is not the case in general. Actually the gap between the integer and the fractional optimum can be very large.[6]

For an optimization problem we may consider OPT, the integer optimum, and FRAC, which is the optimum of the relaxation. A strategy to solve integer program is to relax it in some way, and to *round its fractional solution* in order to get an integer feasible solution. The rounding technique is strongly dependent on the particular optimization problem and it is out of the scope of this lecture. Let ROUND denotes such solution, then for (say) a maximization problem we have

$$\mathsf{FRAC} \geq \mathsf{OPT} \geq \mathsf{ROUND} \tag{3}$$

Definition 6. *Given an integer program and its relaxation, the ratio*

$$\frac{\mathsf{OPT}}{\mathsf{FRAC}} \tag{4}$$

is the integrality gap *of the relaxation.*

Very often the analysis of an approximation algorithm for a maximization problem is achieved by determining the ratio between the fractional optimum and its rounded value, $\frac{\mathsf{ROUND}}{\mathsf{FRAC}}$. The latter is bounded from above by the integrality gap[7].

Improving the linear relaxations

We discussed the gap between an integer program and its linear relaxation, which is a key concept in the analysis of approximation algorithms. To get a tighter relaxation we can add **additional inequalities** to the initial program. Such inequalities must be valid for integer solutions but may still shave off some of the fractional solutions.

Given an arbitrary linear program let $\mathcal{P}$ be the convex hull of its feasible solutions, and $\mathcal{P}_I$ the convex hull of its integer solutions. Of course $\mathcal{P}_I \subseteq \mathcal{P}$. The idea is to find a way to efficiently identify a convex set $\mathcal{E}$ such that $\mathcal{P}_\mathcal{I} \subseteq \mathcal{E} \subseteq \mathcal{P}_I$. If the gap between the optimal solutions in $\mathcal{E}$ and the ones in $\mathcal{P}_I$ is strictly smaller than the integrality gap of the original linear program, then we have a better handle to solve the combinatorial problem. Even better if $\mathcal{E} = \mathcal{P}_I$: in this case there is an efficient way to find an optimal solution for the problem. If $\mathsf{P} \neq \mathsf{NP}$ there is not general way to obtain $\mathcal{P}_I$ by adding just a polynomial number of linear inequalities.

Cutting planes: Gomory[8] introduces a way to cut fractional solutions by noticing that if $a^T x \leq b$ is a valid inequality and a is an integer vector, then

$$a^T x \leq \lfloor b \rfloor \tag{5}$$

is valid over integer solutions (but it is not over fractional ones). This method has been employed effectively in integer programming software.

[6] Consider the problem of computing the maximum independent set of a graph $G = (V, E)$. In this case we want to maximize the objective function $\sum_{v \in V} x_v$ under the assumption that $x_u + x_v \leq 1$ for every pair $\{u, v\} \in E$ for variables $x_v \in \{0, 1\}$. If we consider the **complete graph** then the maximum independent set has size at most 1, but the linear relaxation has a solution of value $|V|/2$, which is the vector $(\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2})$.

[7] For a minimization problem we get $\mathsf{FRAC} \leq \mathsf{OPT} \leq \mathsf{ROUND}$, so the integrality gap actually lower bounds $\frac{\mathsf{ROUND}}{\mathsf{FRAC}}$.

[8] Ralpha E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–278, 1958

Extended formulation: here the idea is to add new variables and add linear constraints on them in such a way that the projection of the extended polytope $\mathcal{P}_E$ on the original variables is closer or even equal to $\mathcal{P}_I$.

Such extended formulation is "problem dependent" and there are very nice connections between the number of inequalities needed to capture $\mathcal{P}_I$ and communication complexity. Fiorini et.al.[9] proved that there is no polynomial size extended formulation for the canonical integer programming representation of the *traveller salesman problem*.

In this course we deal with systematic ways to improve the relaxation of the integer programs. In this lecture we are going to see some techniques specific to linear programming.

[9] Samuel Fiorini, S. Massar, S. Pokutta, H.R. Tiwary, and R. de Wolf. Linear vs. semidefinite extended formulations: exponential separation and strong lower bounds. In *Proceedings of the 44th symposium on Theory of Computing*, pages 95–106. ACM, 2012

Lovász-Schrijver hierarchy

Let us just focus on feasibility problems (i.e., no function to be optimized). The point is to get close to a representation of $\mathcal{P}_I$ in order to determine whether it is empty or not. Notice that if we could efficiently decide feasibility under *quadratic inequalities* we could easily solve integer programs over boolean variables, since $x_i \in \{0,1\}$ is equivalent to $x_i^2 - x_i = 0$.

The Lovász-Schrijver [10] linear relaxation is based on the idea of using quadratic inequalities to determine new **linear inequalities** which are valid for the integer solutions but not for the fractional ones. Once the new linear inequalities are in place, the quadratic inequalities are forgotten. The resulting linear program is tighter than the initial one and we can optimized it or determine its feasibility.

[10] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0–1 optimization. *SIAM Journal on Optimization*, 1:166, 1991

The simplest way to describe Lovász-Schrijver integer programming relaxation is to interpret it as a proof system. Start with a linear relaxation $Ax \geq b$ and $0 \leq x \leq 1$, and call $\mathcal{P}$ the set of its solutions[11].

[11] We focus on integer programs with boolean solutions here, thus the presence of inequalities $0 \leq x \leq 1$ in the relaxation is assumed. Thus we can also assume that we deal with polytopes here.

Every inequality $\sum_j a_{i,j}x_j - b_i \leq 0$ is an axiom of the proof system, together with axioms $-x_i \leq 0$, $x_i - 1 \leq 0$ and $x_i - x_i^2 \leq 0$. The system allows to multiply a linear inequality by a variable or by the complement of a variable

$$\frac{\sum_j c_j x_j - d \leq 0}{\sum_j c_j(x_j x_i) - d x_i \leq 0} \qquad \frac{\sum_j c_j x_j - d \leq 0}{\sum_j c_j x_j(1-x_i) - d(1-x_i) \leq 0}, \tag{6}$$

and to infer positive combinations of known inequalities

$$\frac{c + \sum_j c_j x_j + \sum_{\{i,j\}} c_{\{i,j\}} x_i x_j \leq 0 \qquad d + \sum_j d_j x_j + \sum_{\{i,j\}} d_{\{i,j\}} x_i x_j \leq 0}{\alpha(c + \sum_j c_j x_j + \sum_{\{i,j\}} c_{\{i,j\}} x_i x_j) + \beta(d + \sum_j d_j x_j + \sum_{\{i,j\}} d_{\{i,j\}} x_i x_j) \leq 0} \text{ assuming } \alpha, \beta \geq 0. \tag{7}$$

Quadratic inequalities can be used to derive new linear inequalities, which in turn can be multiplied again. We will keep track on how many multiplication steps are needed to derive a particular inequality. This is the *rank of an inequality* in Lovász-Schrijver.

Definition 7 (Rank of Lovász-Schrijver)**.** *Consider a derivation of an inequality in Lovász-Schrijver. The* rank of the derivation *of an axiom is* 0*; the rank of a multiplication step is one plus the rank of the premise; the rank of a positive sum is the maximum among the rank of the premises. The* rank of an inequality *is the smallest among the ranks of all possible derivations of that inequality. The* rank of a point *is the smallest among the rank of all linear inequalities that are falsified by that point. The rank of the empty polytope is the rank of the inequality* $1 \leq 0$*.*

The set of solutions of all linear inequalities of rank 0 is denoted as $\mathcal{P}_0$, and it is equal to $\mathcal{P}$. The polytope characterized by the points compatible with all linear inequalities of rank t is denoted as $\mathcal{P}_t$. It is important to stress that while derivations can to use quadratic inequalities, the polytopes are defined in terms of the linear ones. It holds that

$$\mathcal{P} = \mathcal{P}_0 \supseteq \mathcal{P}_1 \supseteq \cdots \supseteq \mathcal{P}_{n-1} \supseteq \mathcal{P}_n = \mathcal{P}_I. \tag{8}$$

where the last equation is proved in[12].

[12] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0–1 optimization. *SIAM Journal on Optimization*, 1:166, 1991

Theorem 8 (Lovász-Schrijver, 1991)**.** *The polyhedron* $\mathcal{P}_n$ *is equal to the convex hull of the integer feasible solutions of* $\mathcal{P}$*.*

The most common definition of Lovász-Schrijver relaxation is geometric, and it is also the original definition. Given the a polytope $\mathcal{P} \subseteq \mathbb{R}^n$, we consider a matrix variable $X \in \mathbb{R}^{(n+1)\times(n+1)}$ with row and column indices going from 0 to n. Matrix X must satisfy the following inequalities: $x_{i0} = x_{0i} = x_{ii}$ for all $i \in [n]$; $x_{ij} = x_{ji}$ for every $i, j \in [n]$; if $\sum_j a_j x_j - b \leq 0$ then X must satisfy

$$\sum_j a_j x_{ij} - b x_{i0} \leq 0$$

for every $i \in \{0\} \cup [n]$. Let's call $\mathcal{M}$ the set of points satisfying these inequalities.

This matrix satisfies every rank 1 inequality derivable in Lovász-Schrijver from $\mathcal{P}$: if the quadratic terms $x_i x_j$ is mapped to x_{ij}, if x_i is mapped to x_{0i} and each degree zero term is multiplied by x_{00}, then $\mathcal{M}$ satisfies every line of these rank 1 derivations. We define[13] $N(\mathcal{P})$ as the projection over the variable $x_{00}, x_{01}, \ldots, x_{0n}$, with $x_{00} = 1$. So it is clear that $N(\mathcal{P}) = \mathcal{P}_1$ and that $N(\mathcal{P}_t) = \mathcal{P}_{t+1}$. Set $N^t(\mathcal{P})$ is the t-th iteration of the operaton N over $\mathcal{P}$.

[13] this is often called the *Lovász-Schrijver operator*.

Lovász-Schrijver is a *Lift and project* method to improve on the linear relaxation. The reason for this name is clear: first the linear program is augmented to $\approx n^2$ variables, and then is projected again to the original variables. Moving to the larger space allows to get tighter inequalities in the original space.

Theorem 9. [14] *Let be $\mathcal{P}$ a polytope described by $n^{O(1)}$ inequalities. It is possible to optimize any linear function over $\mathcal{P}_t$ in time $n^{O(t)}$.*

[14] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0–1 optimization. *SIAM Journal on Optimization*, 1:166, 1991

Proof Sketch. In order to optimize over a linear program using the ellipsoid algorithm it is sufficient to have a weak separator oracle. In the paper they show a polynomial time reduction from a weak separator oracle for $N(\mathcal{P})$ to a weak separator oracle for $\mathcal{P}$. □

Sherali-Adams hierarchy

Here we start with a set of polynomial inequalities in variables $x_1 \dots x_n$ over $\{0,1\}$.

$$p_1 \geq 0, p_2 \geq 0, \dots, p_m \geq 0. \tag{9}$$

A Sherali-Adams proof of a polynomial inequality $p \geq 0$ over the same variables is an equation of the form

$$\sum_{1 \leq l \leq m} p_l g_l + \sum_i (x_i^2 - x_i) h_i + g_0 = p \tag{10}$$

where each

$$g_l = \alpha_l \prod_{i \in A_l} x_i \prod_{j \in B_l} (1 - x_j) \tag{11}$$

with $\alpha_l \geq 0$ and $A_l \cap B_l = \emptyset$; and each h_i is an arbitrary polynomial.

The *rank of a Sherali-Adams* proof is equal the the maximum degree among the polynomials

$$g_0 \qquad g_l p_l \qquad h_i(x_i^2 - x_i). \tag{12}$$

A refutation of a set of polynomials is a Sherali-Adams proof of $-1 \geq 0$.

There are other definitions of rank in literature. For example[15] define the Sherali-Adams rank as the degree of the polynomials g_l. In that way you get that Sherali-Adams rank simulates Lovász-Schrijver rank if the initial inequalities have degree 1.

[15] Monique Laurent. A comparison of the sherali-adams, lovász-schrijver and lasserre relaxations for 0-1 programming. *Mathematics of Operations Research*, 28:470–496, 2001

While Sherali-Adams is a natural proof system for non-linear inequalities, the linear case is the one we are interested to.

If $\{p_l \geq 0\}_{l=1}^m$ are linear inequalities, then they define a bounded polytope since we are always under the assumption that $0 \leq x_i \leq 1$. Let's call this polytope $\mathcal{P}$. Then we can define $S_t(\mathcal{P})$ as the polytope obtained by all linear inequalities obtained through a proof of rank at most t. It is clear from the definition that $N^t(\mathcal{P}) \supseteq S_{t+1}(\mathcal{P}) \supseteq \mathcal{P}_I$, and that

$$\mathcal{P} = S_0(\mathcal{P}) \supseteq S_1(\mathcal{P}) \supseteq S_{n+1}(\mathcal{P}) = \mathcal{P}_I. \tag{13}$$

Sherali-Adams can be seen as a Lift-and-Project relaxation as well. Consider $Y \in \mathbb{R}^s$ where $s = \binom{n}{\leq t}$, where every coordinate of Y is naturally indexed by a subset of $[n]$ of size at most t.

Map every monomial m over variables $x_1, \dots, x_n$ into the variable $Y_{I(m)}$ where $I(m)$ is the set of indices of the variables in m. For example

$$x_2^2 x_3 x_5 \mapsto Y_{\{2,3,5\}}$$

This map translate any polynomial inequality of degree at most t into a linear inequality of over $\mathbb{R}^s$. Consider all inequalities of degree at most t of the form $gp_l \geq 0$ where p_l is an initial inequality; $g = \alpha \prod_{i \in A} x_i \prod_{j \in B}(1 - x_j)$; $\alpha \geq 0$ and $A \cap B = \emptyset$, and take the linear inequalities obtained by translating monomials into Y variables.

Take $\mathcal{E}$ to be the set of all vectors Y that satisfy all such linear inequalities. The polytope $\mathcal{E}$ satisfies all linear inequalities over the original variables which are derivable in Sherali-Adams proofs of rank at most t, after projecting over $(y_\emptyset, y_1, \ldots, y_n)$.

Efficiency of SA: if the linear system has $n^{O(1)}$ initial inequalities then the polytope $S_t(\mathcal{P})$ can be described by $n^{O(t)}$ linear inequalities over the Y_S variables, with $|S| \leq t$. Since this is a linear system, both optimization and feasibility can be computed in time $n^{O(t)}$.

An example of Sherali-Adams relaxation

Consider the maximum independent set integer program for a graph $G = (V, E)$.

$$\begin{aligned} \text{maximize} \quad & \sum_{v \in V} x_v \\ \text{subject to} \quad & x_u + x_v - 1 \leq 0 \qquad \text{for } \{u, v\} \in E \\ & 0 \leq x_u \leq 1. \end{aligned} \tag{14}$$

The Sherali-Adams relaxation of rank t is

$$\begin{aligned} \text{maximize} \quad & \sum_{v \in V} Y_{\{v\}} \\ \text{subject to} \quad & \sum_{T' \subseteq T} (-1)^{|T'|} \cdot \left[Y_{S \cup T' \cup \{u\}} + Y_{S \cup T' \cup \{v\}} - Y_{S \cup T'} \right] \leq 0 \\ & \qquad \text{for } \{u, v\} \in E, |S \cup T \cup \{u\}| \leq t, |S \cup T \cup \{v\}| \leq t \end{aligned}$$

$$0 \leq \sum_{T' \subseteq T} (-1)^{|T'|} Y_{S \cup T' \cup \{u\}} \leq \sum_{T' \subseteq T} (-1)^{|T'|} Y_{S \cup T'}$$
$$\text{for } \{u, v\} \in E,\ |S \cup T \cup \{u\}| \leq t, |S \cup T \cup \{v\}| \leq t\,.$$

References

[FMP+12] Samuel Fiorini, S. Massar, S. Pokutta, H.R. Tiwary, and R. de Wolf. Linear vs. semidefinite extended formulations: exponential separation and strong lower bounds. In *Proceedings of the 44th symposium on Theory of Computing*, pages 95–106. ACM, 2012.

[Gom58] Ralpha E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–278, 1958.

[Lau01] Monique Laurent. A comparison of the sherali-adams, lovász-schrijver and lasserre relaxations for 0-1 programming. *Mathematics of Operations Research*, 28:470–496, 2001.

[LS91] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0–1 optimization. *SIAM Journal on Optimization*, 1:166, 1991.

[MG07] Jiří Matoušek and Bernd Gärtner. *Understanding and using linear programming*. Springer, Berlin New York, 2007.