Strong Relaxations for Discrete Optimization Problems 22/04/16

Lecture : Extended formulations and communication protocols

Lecturer: Yuri Faenza *Scribes: Jakub Tarnawski*

1 Recap

In the last lecture we considered a polytope $P \subseteq \mathbb{R}^d$ written in two ways: as the convex hull of a set of points and as given by m linear constraints:

$$P = \text{conv}\{v_1, ..., v_n\} = \{x \in \mathbb{R}^d : Ax \leq b\}.$$

Given such a description, we associated with P a matrix $S \in \mathbb{R}^{m \times n}_{\geq 0}$ called the *slack matrix of* P defined as follows:

$$S_{ij} = b_i - A_i v_j,$$

where A_i is the i-th row of A.

Also, given any nonnegative matrix $S \in \mathbb{R}^{m \times d}_{\geq 0}$, we say that a pair of matrices (T, U) is a *rank-r nonnegative factorization* of S if:

$$T \in \mathbb{R}^{m \times r}_{\geq 0}, \quad U \in \mathbb{R}^{r \times d}_{\geq 0}, \quad S = TU.$$

We define the *nonnegative rank of* S as

$$\text{rk}_+(S) = \min\{r : S \text{ has a rank-}r \text{ nonnegative factorization}\}.$$

And the fundamental result which explains our interest in slack matrices was:

Theorem 1 (Yannakakis' Factorization Theorem) *Let P be a polytope and S its slack matrix. Suppose* $\dim(P) \geq 1$. *Then*

$$xc(P) = rk_+(S).$$

2 Deterministic communication complexity

In this lecture, we will see how to obtain nonnegative factorizations (or prove that they do not exist), which translate to extended formulations. We will do this using tools from communication complexity.

2.1 Communication protocols

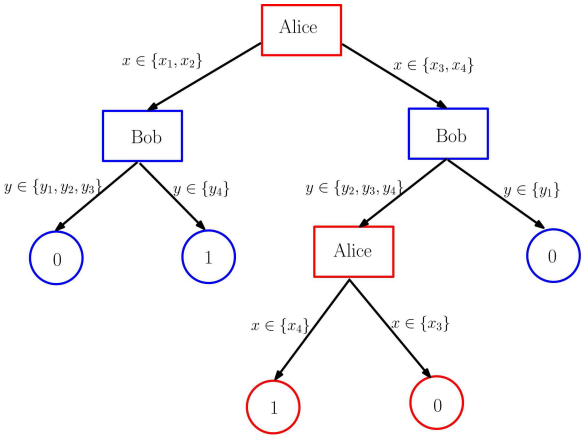
Imagine the following cooperative game played by two players Alice and Bob. They are both given a matrix $S \in \mathbb{R}^{m \times n}_{\geq 0}$ and infinite time together to come up with a strategy. Then they are locked in separate rooms and Alice is given a row index x, while Bob is given a column index y. Their objective is to figure out the value of $S_{x,y}$. To this end, they communicate in rounds, with Alice beginning. In each round, Alice sends one bit of information to Bob. Then Bob either halts the protocol and outputs the answer, or sends one bit of information back to Alice. The complexity of the protocol (for a given S and communication strategy) is the maximum number of rounds it can take Alice and Bob to produce a valid answer (the maximum is taken over all possible choices of (x, y)).[1] We say that the protocol *computes* S.

A protocol can be seen as a binary tree. Each node in this tree is either an Alice-node (which we also call an x-node) or a Bob-node (a y-node). A node can either have two children (the left subtree corresponds to sending a 0 to the other person, and the right corresponds to a 1), or be a leaf (which

[1] Sometimes we think that any natural number of bits can be sent during one round – in this case, the complexity would be the maximum number of bits sent.

	y_1	y_2	y_3	y_4
x_1	0	0	0	1
x_2	0	0	0	1
x_3	0	0	0	0
x_4	0	1	1	1

(a) Matrix $S \in \mathbb{R}_{\geq 0}^{4\times 4}$.



(b) Decision tree of a protocol computing S. Its complexity is 3.

	y_1	y_2	y_3	y_4
x_1	0	0	0	1
x_2	0	0	0	1
x_3	0	0	0	0
x_4	0	1	1	1

(c) Partition of S into monochromatic rectangles.

Figure 1: Example of a matrix and a deterministic protocol.

corresponds to outputting the final answer). Figure 1 gives an example of a matrix S and a decision tree of a protocol computing S. Note that since every leaf corresponds to one output value, and the path from the root to this leaf corresponds to gradually shrinking the sets of admissible x-values and y-values, every leaf of the tree corresponds to a monochromatic rectangle in S.

Definition 2 *Given a matrix $S \in \mathbb{R}_{\geq 0}^{m\times n}$, we define its* deterministic communication complexity $\mathrm{DCC}(S)$ *as the minimum complexity of a protocol that computes S.*

Clearly we have $\mathrm{DCC}(S) \leq \min(\lceil \log_2 n \rceil, \lceil \log_2 m \rceil)$ – indeed, it is enough if, say, Alice sends x to Bob; then Bob can easily compute $S_{x,y}$. However, $\mathrm{DCC}(S)$ can be much less than that – say, if all rows of S are equal, then one round is enough since Bob can produce the answer without any input from Alice. In this case, S is a rank-one matrix. It is easy to see that in fact any rank-one matrix has a constant communication complexity.

This triggers the question of whether the nonnegative rank of a matrix can always be upper-bounded using its deterministic communication complexity. Yannakakis answered this in the affirmative:

Theorem 3 ([Yan88]) *For any matrix $S \in \mathbb{R}_{\geq 0}^{m\times n}$ we have*

$$\lceil \log rk_+(S) \rceil \leq \mathrm{DCC}(S).$$

By Theorem 1, this immediately implies:

Corollary 4 *Let P be a polytope with $\dim(P) \geq 1$ and S its slack matrix. Then*

$$\lceil \log xc(P) \rceil \leq \mathrm{DCC}(S).$$

We are going to deduce Theorem 3 as a corollary of a stronger statement about *randomized* communication protocols which we will prove later. Before this, let us see an application of Corollary 4.

2.2 The stable set polytope of perfect graphs

Definition of perfect graph. The following is the most standard definition of a perfect graph:

Definition 5 *We say that a graph G is* perfect *if for each* induced *subgraph $H \subseteq G$ we have $\omega(H) = \chi(H)$, where $\omega(H)$ is the clique number of H (the size of the largest clique in H) and $\chi(H)$ is the chromatic number of H (the lowest number of colors with which it is possible to color H).*

Note that the inequality $\omega(H) \leq \chi(H)$ always holds. Thus, in other words, a graph is perfect if needing k colors to color an induced subgraph of G implies the existence of a k-clique in that subgraph. An example of a non-perfect graph is a 5-cycle, since it needs 3 colors despite not containing a triangle.

Another characterization of perfect graphs has to do with the stable set polytope. Recall that it is defined as the convex hull of all independent sets of vertices:

$$\mathrm{STAB}(G) = \mathrm{conv}\{\chi_I : \ I \subseteq V \text{ is an independent set in } G\}.$$

The following polytope P_G is a relaxation of $\mathrm{STAB}(G)$:

$$P_G = \{x \in \mathbb{R}^V : \quad x \geq 0, \quad x(K) \leq 1 \text{ for any clique } K \subseteq V \text{ in } G\}.$$

It turns out that for perfect graphs, this relaxation is exact:

Theorem 6 ([Chv75]) *A graph G is perfect if and only if* $\mathrm{STAB}(G) = P_G$.

Revisiting the 5-cycle example, we can see that in this case the all-$\frac{1}{2}$'s vector $\bar{x}$ is in P_G but not in $\mathrm{STAB}(G)$ (since $< 1, \bar{x} >= \frac{5}{2}$, while the 5-cycle only has independent sets of size at most 2).

Slack matrix of P_G. Assume that G is perfect. We proceed to describe the slack matrix of $\mathrm{STAB}(G) = P_G$. We focus on its lower part (the one corresponding to the clique constraints) – the upper part is not interesting (see Remark 7). The entries of S are indexed by cliques K and independent sets I:

$$S_{K,I} = 1 - \chi_I(K) = 1 - |I \cap K| = \begin{cases} 0 & \text{if } |I \cap K| = 1, \\ 1 & \text{if } |I \cap K| = 0. \end{cases}$$

(Note that a clique intersects an independent set in at most one vertex.)

Communication protocol for clique & independent set. We will now describe a communication protocol for S, which, by Corollary 4, will imply an upper bound on the extension complexity of $\mathrm{STAB}(G)$ for G perfect. Alice is given a clique K, and Bob is given an independent set I; together they want to figure out whether there exists a vertex in the intersection $K \cap I$ or not.

Alice's move is the following: she looks for a vertex $v \in K$ with a small neighborhood, i.e., $|N(v)| \leq |V|/2$. If such a v exists, she sends any such v (its index) to Bob. Bob receives v and first checks whether $v \in I$; if yes, then the intersection is nonempty and Bob outputs $S_{K,I} = 0$. So suppose $v \notin I$. Since for any $v \in K$ we have $K \subseteq N(v)$, it is enough to keep looking for the common vertex in $N(v)$ – so Bob now removes everything except $N(v)$ from G. Thanks to the condition that $|N(v)| \leq |V|/2$, the size of the graph shrinks at least by a factor of two. Bob sends a confirmation (say, a 0 bit) back to Alice, she also removes everything except $N(v)$ in her local copy of G, and the protocol continues.

If, on the other hand, such a vertex v does not exist, Alice sends (say) a 0 to Bob, and Bob now looks for a vertex $w \in I$ with large neighborhood: $|N(w)| > |V|/2$. If such a vertex w exists, he sends any such w to Alice. The moves are now similar as before: Alice first checks whether $w \in K$ and, if yes, outputs $S_{K,I} = 0$. Otherwise, since $w \in I$, none of w's neighbors can also be in I, so Alice and Bob can both remove $N(w)$ from the graph, which again shrinks the size by a factor of two.

The last possibility is that such a vertex w does not exist either. In this case, they output $S_{K,I} = 1$ (to say that $K \cap I = \emptyset$). This is because if there was a vertex u in the intersection $K \cap I$, then of course it would be both in K and in I and it would have $|N(u)|$ either at most $|V|/2$ or more than $|V|/2$.

Complexity of protocol. Since in each round, either the protocol halts or the size of the graph shrinks by a factor of 2, the number of rounds is $O(\log |V|)$. In each round, at most $\log |V|$ bits are sent. The overall complexity is $O\left(\log^2 |V|\right)$.

Remark 7 *Recall that S also has an upper part which corresponds to the nonnegativity constraints. To play a game on S, Alice and Bob can do the following: Alice first sends to Bob one bit informing him in which part the row x is. Then, if it is in the lower part, the game is played as above; otherwise they need to output $\chi_I(v)$, where Alice has v and Bob has I. This is easy to do with $\log |V|$ bits since Alice can just send v to Bob.*

Extension complexity of P_G. By Corollary 4 we have the following for perfect graphs G:

$$\lceil \log \mathrm{xc}(\mathrm{STAB}(G)) \rceil = \lceil \log \mathrm{rk}_+(S) \rceil = O\left(\log |V|\right)^2,$$

so that

$$\mathrm{xc}(\mathrm{STAB}(G)) = O\left(|V|^{\log |V|}\right),$$

i.e. there is an extended formulation for $\mathrm{STAB}(G)$ of this size. This is interesting because:

- we do not know how to do better than this (for perfect graphs),
- as we will see in next lecture, one can prove that the extension complexity of $STAB(G)$ for general graphs is much more than for perfect graphs.

3 Randomized communication complexity

3.1 Randomized protocols

In this section we will introduce the notion of a *randomized communication protocol computing a matrix in expectation.*[2] We will then prove a stronger version of Theorem 3.

In randomized protocols, we allow Alice and Bob to use private random bits. Crucially, given a nonnegative matrix S (and row x and column y privately), the pair should output a value which is *always nonnegative* and equal to S **in expectation**.

To define this formally, we will think of a protocol as a binary tree – but now each internal node will correspond to a randomized decision rather than a deterministic one. Namely, at each Alice-node (x-node) v which is not a leaf, Alice, given a row x, will:

- with probability $p_{0,v}(x)$, send 0 to Bob,
- with probability $p_{1,v}(x)$, send 1 to Bob,
- with the remaining probability $1 - p_{0,v}(x) - p_{1,v}(x)$, halt the protocol and output 0.

Similarly, at each Bob-node (y-node) w which is not a leaf, Bob, given a column y, will:

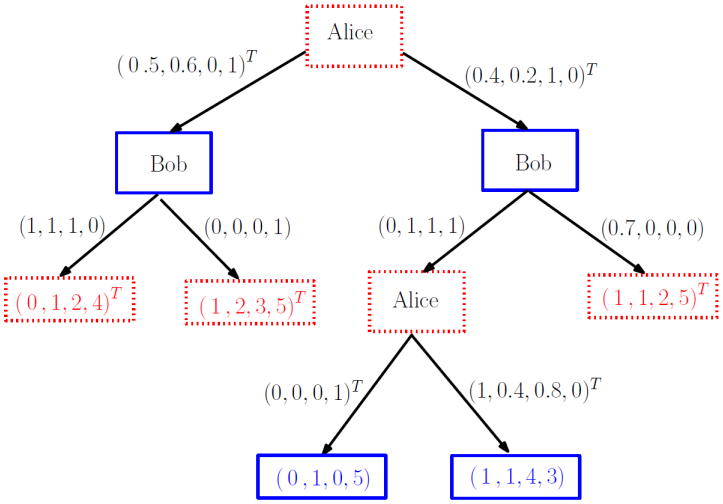
- with probability $q_{0,w}(y)$, send 0 to Alice,
- with probability $q_{1,w}(y)$, send 1 to Alice,
- with the remaining probability $1 - q_{0,w}(y) - q_{1,w}(y)$, halt the protocol and output 0.

Finally, at a leaf Alice-node v, she will deterministically output a nonnegative value $\Lambda_v(x)$. At a leaf Bob-node w, he will output $\Lambda_w(y)$.

Therefore, the protocol can be described by its tree together with functions p, q or Λ for each node.

	y_1	y_2	y_3	y_4
x_1	0.28	0.40	1.60	1.70
x_2	0.74	0.68	0.92	1.44
x_3	1.40	0.80	3.20	2.40
x_4	4.00	4.00	4.00	5.00

(a) Matrix $S \in \mathbb{R}_{\geq 0}^{4\times 4}$.



(b) Tree of a randomized protocol computing S in expectation. Each internal Alice-node v has its left-pointing edge labelled with the vector $p_{0,v}$ and its right-pointing edge with $p_{1,v}$. Similarly each internal Bob-node has edges labelled with q. Leaves have vectors Λ written in them.

Figure 2: Example of a matrix and a randomized protocol.

We say that the protocol *computes S in expectation* if for each x and y, the expected output of the protocol on this input is $S_{x,y}$.

Figure 2 shows an example matrix S and a randomized protocol for it.

We make two observations:

- this is a generalization of a deterministic protocol (i.e., any deterministic protocol can be expressed in this way),
- if $S_{x,y} = 0$, then the output value must always be 0 (since it is required to be nonnegative, and 0 in expectation).

The following definition is just an extension of that of DCC(S):

Definition 8 *Given a matrix $S \in \mathbb{R}_{\geq 0}^{m\times n}$, we define its* randomized communication complexity RCC(S) *as the minimum complexity of a randomized protocol that computes S.*

Because of the first observation above, we have $\mathrm{RCC}(S) \leq \mathrm{DCC}(S)$.

3.2 Equivalence of randomized communication complexity and nonnegative rank

In this section we prove the following.

Theorem 9 ([FFGT15]) *For any matrix $S \in \mathbb{R}_{\geq 0}^{m\times n}$ we have*

$$\lceil \log rk_+(S) \rceil = \mathrm{RCC}(S).$$

Note that this is stronger in two ways:

[2]Our definitions here are not the standard ones in the field, where protocols are asked to compute a matrix *with high probability*.

- if we are looking for a compact extended formulation, then it is enough to give a *randomized* protocol,
- there is equality, so now a lower bound on $\mathrm{RCC}(S)$ will also transfer to a lower bound on $\mathrm{rk}_+(S)$.

Proof **Direction $\leq$.** As we remarked before, a protocol P is described by its tree and vectors p, q, Λ. For any node v let us define

$$\sigma_v(x,y) = \text{probability that an execution of the protocol, given input } (x,y), \text{ will pass through } v.$$

It is easy to see that getting to v is equivalent to making all the "right" choices at each node in the unique path from the root to v in the tree. That is:

$$\sigma_v(x,y) = \prod_{u\text{: } x\text{-node, ancestor of } v} p_{\alpha_u,u}(x) \cdot \prod_{w\text{: } y\text{-node, ancestor of } v} q_{\alpha_w,w}(y)$$

where α_u and α_w denote the "right" bit at u or w (the one that has to be chosen in order to go to v). Note that this expression has a nice form – the left part depends only on x and the right part depends only on y. This means that σ_v is a rank-one matrix: the product of a column vector containing the first parts, for all x, and of a row vector containing the second parts, for all y. Moreover, since P computes S in expectation, we have

$$\begin{aligned}
S_{x,y} &= \mathbb{E}[\text{output on input } (x,y)] \\
&= \sum_{v\text{: leaf, } x\text{-node}} \Lambda_x(v)\sigma_v(x,y) + \sum_{v\text{: a leaf, } y\text{-node}} \Lambda_y(v)\sigma_v(x,y) \\
&= \sum_{v\text{: leaf, } x\text{-node}} \left(\left(\Lambda_x(v) \cdot \prod_{u\text{: } x\text{-node, ancestor of } v} p_{\alpha_u,u}(x) \right) \cdot \left(\prod_{w\text{: } y\text{-node, ancestor of } v} q_{\alpha_w,w}(y) \right) \right) \\
&+ \sum_{v\text{: leaf, } y\text{-node}} \left(\left(\prod_{u\text{: } x\text{-node, ancestor of } v} p_{\alpha_u,u}(x) \right) \cdot \left(\Lambda_y(v) \cdot \prod_{w\text{: } y\text{-node, ancestor of } v} q_{\alpha_w,w}(y) \right) \right).
\end{aligned}$$

For each leaf v, the corresponding summand is again a product of two parts, the first of which depends only on x and the second only on y. This again means that for each x-node leaf v, the matrix having

$$\left(\left(\Lambda_x(v) \cdot \prod_{u\text{: } x\text{-node, ancestor of } v} p_{\alpha_u,u}(x) \right) \cdot \left(\prod_{w\text{: } y\text{-node, ancestor of } v} q_{\alpha_w,w}(y) \right) \right)$$

as the (x,y)-th entry is rank-one. (The same thing holds for y-node leaves v.) In consequence, S is a sum of ℓ such rank-one matrices, where ℓ is the number of leaves in the tree. It is easy to see that this implies $\mathrm{rk}_+(S) \leq \ell$. But since this is a binary tree, we have $\ell \leq 2^{\text{complexity of } P}$, hence $\lceil \log \mathrm{rk}_+(S) \rceil \leq$ complexity of P.

Direction $\geq$. Take a rank-r factorization of S:

$$S = T \cdot U, \qquad T \in \mathbb{R}^{m\times r}_{\geq 0}, \qquad U \in \mathbb{R}^{r\times n}_{\geq 0}.$$

We will show a protocol of complexity $\lceil \log r \rceil$.

First, assume without loss of generality that the sum of every row of T is at most 1. (If it is not, replace T with ΔT and U with $\frac{1}{\Delta}U$, where Δ is the maximum row sum in T.) We look at entries of T as probabilities.

The protocol is very simple: Alice sends to Bob an index $i = 1, \ldots, r$ randomly with probability $T_{x,i}$ (with the remaining probability $1 - \sum_i T_{x,i}$, she halts and outputs 0). Bob outputs $U_{i,y}$.

Clearly, the protocol uses $\lceil \log r \rceil$ bits and the output value is nonnegative. The matrix of expectations is:

$$\{\mathbb{E}[\text{output on input } (x,y)]\}_{x,y} = \sum_{i=1}^{r} T_{x,i} U_{i,y} = T \cdot U = S,$$

as required. ■

3.3 Example application – the spanning tree polytope

Recall the spanning tree polytope:

$$\begin{aligned} P^{st}(K_n) &= \text{conv}\{\chi_T : T \subseteq E \text{ is a spanning tree of } G\} \\ &= \{x \in \mathbb{R}^E : \quad x \geq 0, \quad x(E) = |V| - 1, \quad x(E(U)) \leq |U| - 1 \text{ for } \emptyset \subsetneq U \subsetneq V\}. \end{aligned}$$

It is known that $P^{st}(K_n)$ has a compact extended formulation of size $O(n^3)$). Thanks to Theorem 9, we can prove the existence of such an extended formulation by showing a low-complexity randomized protocol for the communication game on the slack matrix of $P^{st}(K_n)$.

The [interesting part of] the slack matrix S of $P^{st}(K_n)$ is:

$$S_{U,T} = |U| - 1 - |T \cap E(U)| \geq 0$$

where U is a proper subset of V and T is a spanning tree. For any U, $T[U]$ is a forest and

$$S_{U,T} = (\text{the number of connected components of } T[U]) - 1.$$

The protocol will be as follows:

- Alice sends an arbitrary vertex $u \in U$ to Bob.
- Bob picks an edge $e \in T$ from the tree uniformly at random. When T is rooted at u, one endpoint of e is the parent of the other; say $e = (z, w)$, where z is the parent. Bob sends (z, w) to Alice.
- If $w \in U$ but $z \notin U$, then Alice outputs $|V| - 1$. Otherwise she outputs 0.

Clearly, the protocol uses $3 \log |V|$ random bits and its output is nonnegative. Let us show that it works in expectation. We have

$$\mathbb{E}[\text{output on input } (U,T)] = (|V| - 1) \cdot \mathbb{P}(\text{output is } |V| - 1),$$

so it is enough to show that

$$\mathbb{P}(\text{output is } |V| - 1) = \frac{(\text{the number of connected components of } T[U]) - 1}{|V| - 1} = \frac{S_{U,T}}{|V| - 1},$$

or in other words, that the number of edges $(z, w) \in T$ with $w \in U, z \notin U$ is equal to the number of connected components of $T[U]$ minus one. To see this, note that for every connected component of $T[U]$, there is exactly one edge $(z, w) \in T$ exiting that component and going up towards the root u (except for the component containing u, which has no such edge). (The vertex w is the root, in T, of such a connected component, and z is its parent.)

This concludes the analysis of the protocol. By Theorem 9, it follows that $P^{st}(K_n)$ has an extended formulation of size $O(n^3)$.

3.4 Separation between deterministic and randomized protocols

In the *Set Disjointness* problem, Alice and Bob are given respectively sets $A, B \subseteq [n]$, and they want to decide if they intersect. Clearly, one can devise a protocol of complexity n to decide it. In fact, it can be proved that one cannot do really better. Let D^n be the $2^n \times 2^n$ matrix defined as follows.

$$S_{A,B} = \begin{cases} 1 & \text{if } A \cap B = \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 10 *[Raz92] Every protocol computing D^n with high probability has complexity $\Omega(n)$.*

Let $S \in \mathcal{R}_{\geq 0}^{m \times p}$. A *negative embedding* of D^n into S is a pair of maps $\alpha : 2^n \to m$ and $\beta : 2^n \to p$ such that, for each $A, B \subseteq [n]$,

$$D^n_{A,B} = 1 \text{ if and only if } S_{\alpha(A),\beta(B)} = 0.$$

From Theorem 10 one immediately deduces the following.

Lemma 11 *If there exists a negative embedding of D^n into S, then $DCC(S) = \Omega(n)$.*

Hence, in order to show an exponential separation between DCC(S) and RCC(S), it is enough to show the following.

Lemma 12 *[FFGT15] There exists a negative embedding of D^n into the slack matrix of the spanning tree polytope of K_{2n+1}.*

Proof We will map each (A, B) into an entry of $S_{U,T}$ as defined in the previous section. For $A \subseteq [n]$, define $U = \alpha(A)$ as follows: add $2n+1$ to U, and then add $n+i$ to U for all $i \in A$. For $B \subseteq [n]$, define $T = \beta(B)$ as follows: add to T edges $\{i, 2n+1\}$ for all i; add to T edges $\{i, n+i\}$ for all $i \in B$, add to T edges $\{n+i, 2n+1\}$ for all $i \notin B$. One easily verifies that the maps α, β define a negative embedding. ■

We remark that Lemma 11 can be generalized to protocols with little *normalized variance*. Intuitively, the normalized variance is a measure of the "amout of randomness" used by a protocol: see [FFGT15] for details.

References

[Chv75] Vašek Chvátal. On certain polytopes associated with graphs. *Journal of Combinatorial Theory, Series B*, 18(2):138–154, 1975.

[FFGT15] Yuri Faenza, Samuel Fiorini, Roland Grappe, and Hans Raj Tiwary. Extended formulations, nonnegative factorizations, and randomized communication protocols. *Mathematical Programming, Series B*, 153(1):75–94, 2015.

[Raz92] Alexander Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci*, 106(2):385–390, 1992.

[Yan88] Mihalis Yannakakis. Expressing combinatorial optimization problems by linear programs. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 223–228. ACM, 1988.