COMS E6998: Proof Complexity and Applications (Spring'25) February 13, 2025

Lecture 3: Feasible Interpolation

Instructor: *Toniann Pitassi* Scribe: *Steve Marquez*

1 Automatizability

Automatizability is concerned with how efficiently proofs can be found in a given propositional proof system, $\mathcal{P}$. Note that hard-to-refute unsatisfiable CNFs may require superpolynomial-size refutations, and this motivates the following property of a proof system $\mathcal{P}$. Informally $\mathcal{P}$ is *automatizable* if there is an algorithm such that given as input an unsatisfiable CNF F, A outputs a $\mathcal{P}$-refutation of F in time polynomial in the size of the shortest $\mathcal{P}$ refutation.

Definition 1 (Automatizability)**.** *A proof system $\mathcal{P}$ is (polynomially) automatizable if there exists an algorithm A such that for all unsatisfiable CNF formulas F given as input to A, $A(F)$ outputs a $\mathcal{P}$-refutation of F, and the runtime of A on F is polynomial in the size of the minimum $\mathcal{P}$ proof of F.*

Note that the weaker a proof system is, the easier it should be to find proofs, For example, if $\mathcal{P}$ requires $2^{\Omega(n)}$-size refutations for every unsatisfiable CNF F (over n variables), then $\mathcal{P}$ is trivially automatizable. However, for stronger proof systems that admit short proofs for many unsat formulas (e.g., Frege systems), it is highly nontrivial to find a refutation in time polynomial in the size of the shortest proof.

1.1 Remarks

An automatizable proof system is obviously desirable for SAT-solving and automated theorem proving, especially if the proof system in question is fairly strong. It has been shown that tree-like Resolution is automatizable in quasi-polynomial-time, and dag-like Resolution is automatizable in sub-exponential time. We will see later in the course that many algebraic and semialgebraic proof systems (Sherali-Adams, Sum-of-Squares, Nullstellensatz, Polynomial Calculus) are *degree* automatizable, meaning that refutations in these systems can be found in time $n^{O(d)}$ where d is the minimal degree of the refutation. Of these systems, Sum-of-Squares (SOS) is the most powerful, and we will see later that degree-automatizability of SOS has been exploited to give new approximation algorithms for a variety of distributional learning problems.

We note that there is another more relaxed notion of automatizability, called *weak automatizability.* A proof system $\mathcal{P}$ is weakly automatizable if there is an algorithm A such that for any unsatisfiable CNF formula F, A returns a refutation in *some* Cook-Reckhow proof system (that can be stronger than $\mathcal{P}$ itself, and moreover $A(F)$ runs in time polynomial in the size of the shortest $\mathcal{P}$-refutation of F. It is an open problem whether or not Resolution is weakly automatizable. We note that the difference between automatizable and weakly automatizable is analogous to the difference between proper learnability and learnability.

Next we define the notion of feasible interpolation, which is connected to automatizablity.

2 Feasible Interpolation

Consider an unsatisfiable CNF formula of the form $A(\vec{x}, \vec{z}) \wedge B(\vec{y}, \vec{z})$, where A is a CNF formula over the variables $\vec{x}, \vec{z}$, and similarly B is a CNF formula over the variables $\vec{y}, \vec{z}$. We think of the variables $\vec{z}$ as shared, and the variables $\vec{x}$ (respectively $\vec{y}$) as private to A (B respectively). A CNF formula of this type is called a "split" formula.

Observation 2. *If f is unsatisfiable, then it must be true that for any assignment $\vec{\alpha}$ for $\vec{z}$, either $A(\vec{x}, \vec{\alpha})$ or $B(\vec{y}, \vec{\alpha})$ must be unsatisfiable.*

Definition 3 (Interpolant Function). *An interpolant function is a function that on input an assignment $\vec{\alpha}$ to the shared variables, says which of the two CNFs, $A(\vec{x}, \vec{\alpha})$, or $B(\vec{y}, \vec{\alpha})$ is unsatisfiable. Note that both can be unsatisfiable, in which case an interpolant function can give either as a legal answer. This leads to the following definition of an interpolant function associated with a CNF of this form.*

Let $A(\vec{x}, \vec{z}) \wedge B(\vec{y}, \vec{z})$ be unsatisfiable. An interpolant function, $f_{A \wedge B}$ for this CNF formula takes as input an assignment $\vec{\alpha}$ for $\vec{z}$, and satisfies:

$$f_{A \wedge B}(\vec{\alpha}) = \begin{cases} 1 & \textit{if } A(\vec{x}, \vec{\alpha}) \textit{ is SAT} \\ 0 & \textit{if } B(\vec{y}, \vec{\alpha}) \textit{ is SAT} \\ * & \textit{otherwise} \end{cases}$$

Definition 4 (Feasible Interpolation). *$\mathcal{P}$ has feasible interpolation if there is a circuit $C(\vec{\alpha})$ that computes $f_{A \wedge B}$ for every UNSAT split formula $A \wedge B$. Furthermore, the size of C is polynomial in size of the shortest $\mathcal{P}$-refutation of $f_{A \wedge B}$.*

Definition 5 (Monotone Feasible Interpolation). *$\mathcal{P}$ has monotone feasible interpolation if whenever the $\vec{z}$ variables occur only negatively in B and positively in A, then there exists a monotone circuit C that computes $f_{A \wedge B}$ and the size of C is polynomial in the size of the shortest $\mathcal{P}$-refutation of $f_{A \wedge B}$.*

3 Automatizability and Feasible Interpolation

Next we relate automatizability and feasible interpolation.

Theorem 6 ([BPR00]). *For any propositional proof system $\mathcal{P}$, if $\mathcal{P}$ is automatizable, $\mathcal{P}$ has the feasible interpolation property.*

Proof. Let $A(\vec{x}, \vec{z}) \wedge B(\vec{y}, \vec{z})$ have a $\mathcal{P}$-refutation π. Then, let us define an automatizable algorithm C to solve the interpolant function $f_{A \wedge B}$ that does the following:
"On input $\vec{\alpha}$, run algorithm C for $|\pi|$ steps on $A(\vec{x}, \vec{\alpha})$. If it outputs a refutation, output 'A UNSAT', else output "B UNSAT"
Note that if $B(\vec{y}, \vec{\alpha})$ is satisfiable, then let $\vec{\rho}$ be the satisfying assignment to $\vec{y}$. Our split formula can be solved as $A(\vec{x}, \vec{\alpha}) \wedge B(\vec{\rho}, \vec{\alpha}) = A(\vec{x}, \vec{\alpha}) \wedge 1$. Therefore, it must be true that $\Pi|_{\vec{z}=\vec{\alpha}, \vec{y}=\vec{\rho}}$ is a refutation of $A(\vec{x}, \vec{\alpha})$. □

4 Proof System: Cutting Planes

In this section, we will study the Cutting Planes proof system, and prove that it has feasible interpolation. Then we will use this property to prove exponential lower bounds for Cutting Planes proofs. Cutting

Planes is a refutation system used for proving unsolvability of a system of linear inequalities where the variables are $x_1, \dots, x_n \in \mathbb{Z}$.

Definition 7 (Cutting Planes Refutations)**.** *Let $Ax \geq b$ be a system of integer linear inequalities such that $\{a^1x \geq b^1, a^2x \geq b^2, \dots, a^mx \geq b^m\}$ where $A \in \mathbb{Z}^{m\times n}, b \in \mathbb{Z}^m$.*

A Cutting Planes (CP) refutation of $Ax \geq b$ is a finite sequence of inequalities such that each line (inequality) is either one of the original ones or ones that follow from previously derived inequalities by an application of one of the CP inference rules. The goal of CP is to derive the inequality $0 \geq 1$ from the initial set of inequalities, showing that the initial set of inequalities is unsatisfiable over the reals. The rules are as follows:

(Division Rule) *Given a CP refutation and an integer d where d divides every a, then we can form the following logical implication:*

$$\frac{ax \geq c}{\frac{a}{d}x \geq \left\lceil \frac{c}{d} \right\rceil}$$

(Non-Negative Linear Combination Rule) *Given two linear inequalities, we can derive a new valid linear inequality that is true for some $\alpha, \beta \geq 0$.*

$$\frac{ax \geq c \qquad bx \geq d}{(\alpha a + \beta b)x \geq \alpha c + \beta d}$$

Remark 1. *To refute a CNF $F = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, we must first convert each clause to a linear inequality. For example, consider the clause $C = x_1 \vee \neg x_2 \vee x_3$; this converts to the inequality: $x_1 + (1 - x_2) + x_3 \geq 1$. In addition to the inequalities associated with each clause C_i, we additional add $2n$ additional inequalities that force the values of all variables to be in $[0, 1]$: $\{x_i \geq 0, x_i \leq 1 | i \in (n)\}$.*

Remark 2. *The linear combination rule is sound even over $\mathbb{R}$. However the division rule will preserve only integer solutions.*

Remark 3. *For refuting CNFs, we can assume without loss of generality that all coefficients have magnitude $m \leq 2^{n^2}$. Therefore the size of a refutation can be measured by the number of lines (inequalities) in the CP refutation.*

Remark 4. *CP can be proven to be sound and complete.*

4.1 Complexity

Cutting Planes is a natural proof system that generalizes Resolution. Next we examine the relative complexity of CP versus Resolution proofs.

Claim 8. *CP p-simulates Resolution*

Proof. Consider the following application of the Resolution rule:

$$\frac{f = (x_1 \vee x_2 \vee x_3)(\neg x_1 \vee x_2 \vee x_4)}{x_2 \vee x_3 \vee x_4}$$

Then we can do the following steps as shown below. First, we convert each clause in f to a linear inequality and add them together. Then we can introduce new converted clauses in order to apply the division rule. This will leave us with our resolution of interest.

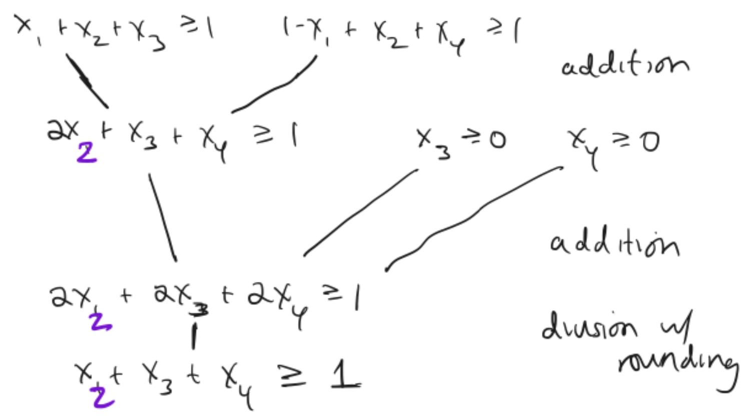


Figure 1: Example refutation of our initial clauses using CP rules.

In order to simulate Resolution with CP, we simply apply the above conversion to each application of the Resolution rule. Since each step can be simulated in CP's by a constant number of steps, it follows that any Resolution refutation can be simulated by a CP refutation of size polynomial in the size of the Resolution refutation. □

The following lemma shows that on the other hand, Resolution cannot p-simulate Cutting Planes.

Lemma 9. *Resolution does not p-simulate CP. In particular, the Pigeonhole Principle (PHP^n_{n-1}) has* polysize *CP refutations but requires exponential-size Resolution refutations.*

Proof. Recall that Resolution refutations of the PHP require exponential size. Thus it suffices to prove that the PHP has polynomial-size CP refutations. Let us recall the initial clauses (converted into equivalent inequalities) for PHP^n_{n-1}:
(a) $\forall i \in [n] : P_{i,1} + P_{i,2} + \cdots + P_{i,n-1} \geq 1$
(b) $\forall i \neq i' \in [n], j \in [n-1] : P_{i,j} + P_{i',j} \leq 1$
Then, we can do the following steps:

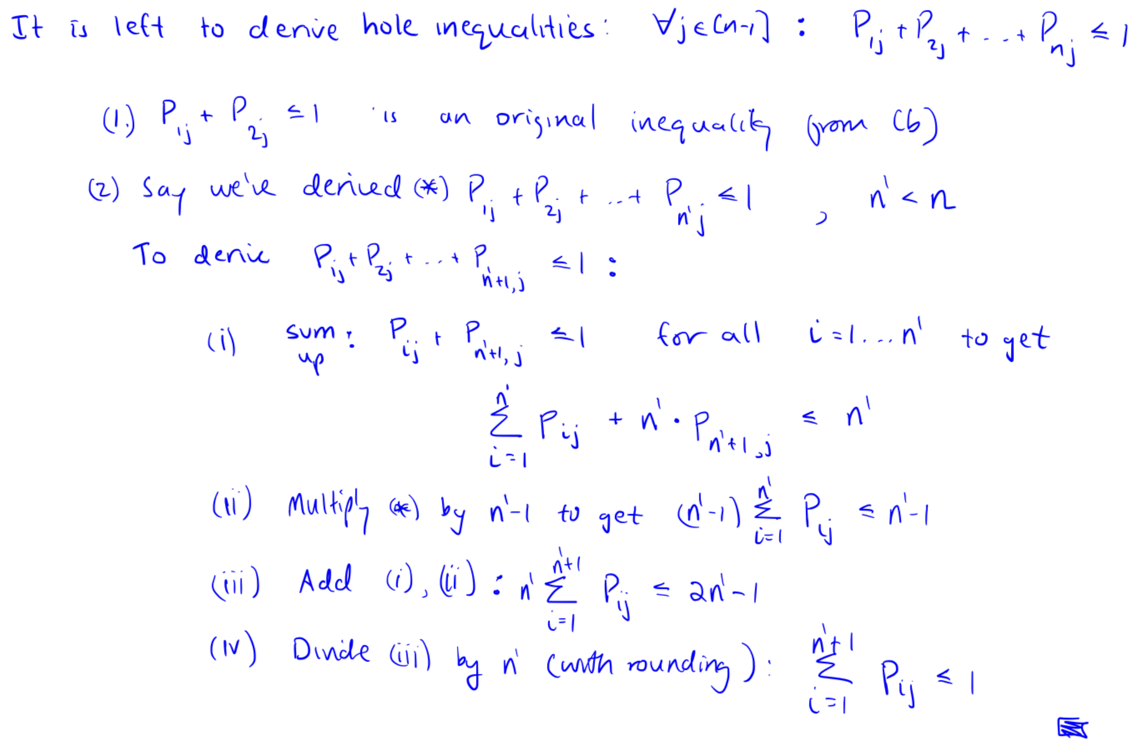
(1) Derive the Hole inequalities: $P_{1,j} + P_{2,j} + \ldots + P_{n,j} \leq 1$ for all $j \in [n-1]$ as shown in *Figure 2*
(2) Add up all equations from step (1) to derive $\Sigma_{i\in[n],j\in[n-1]} P_{i,j} \leq n-1$
(3) Add up all equations from initial equations (a) to derive $\Sigma_{i\in[n],j\in[n-1]} P_{i,j} \geq n$
(4) Add the derived equations from steps (2) and (3) to derive $n-1 \geq n$; then subtracting $n-1$ from both sides we derive $0 \geq 1$ as desired.

□

4.2 Exponential Lower Bounds for CPs using Feasible Interpolation

In this section we prove Exponential lower bounds for CPs refutations using feasible interpolation.

The overview of the proof is as follows. First, we show that CPs has monotone feasible interpolation. In particular, for any split formula, $F = A(\vec{x}, \vec{z}) \wedge B(\vec{y}, \vec{z})$ we have:



It is left to derive hole inequalities: $\forall j \in [n-1]$: $P_{1j} + P_{2j} + \dots + P_{nj} \le 1$

(1.) $P_{1j} + P_{2j} \le 1$ is an original inequality from (6)

(2) Say we've derived (*) $P_{1j} + P_{2j} + \dots + P_{n'j} \le 1$, $n' < n$

To derive $P_{1j} + P_{2j} + \dots + P_{n'+1,j} \le 1$:

(i) sum up: $P_{ij} + P_{n'+1,j} \le 1$ for all $i = 1 \dots n'$ to get

$$\sum_{i=1}^{n'} P_{ij} + n' \cdot P_{n'+1,j} \le n'$$

(ii) Multiply (*) by $n'-1$ to get $(n'-1) \sum_{i=1}^{n'} P_{ij} \le n'-1$

(iii) Add (i), (ii) : $n' \sum_{i=1}^{n'+1} P_{ij} \le 2n'-1$

(iv) Divide (iii) by n' (with rounding): $\sum_{i=1}^{n'+1} P_{ij} \le 1$

∎

Figure 2: Detailed instructions on how to derive the Hole inequalities.

1. If F has a polynomial-size CP refutation where all coefficients are small (the length of all coefficients in binary is $O(polylog(n))$, then there is a polynomial-size monotone circuit for computing the interpolant function;

2. And if F has a polynomial-size CP refutation without any bound on the coefficient length (recall that the coefficient length is without loss of generality at most $poly(n)$, then there is a polynomial-size monotone *real* circuit for computing the interpolant function.

A monotone real circuit is a generalization of a monotone circuit, given by the following definition. We note that to get the main idea behind the CP lower bound, the reader should focus on the simpler case where the coefficients are bounded in length (case 1 above).

Definition 10 (Monotone Real Circuit). *A monotone real circuit for $f : \{0,1\}^n \rightarrow \{0,1\}$ is a sequence of functions $g_1, \ldots, g_s$ where $g_s = f$ and $\forall i < s$, g_i satisfies at least one of these conditions: $g_i = x_i$ or there is a monotone real function $\Phi : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$, and $j, k < i$ such that $g_i = \Phi(g_j, g_k)$*

We prove below the special case corresponding to (1) above. Consider an unsatisfiable split CNF formula $F = A(\vec{x}, \vec{z}) \wedge B(\vec{y}, \vec{z})$ that is monotone with respect to $\vec{z}$. Let π be a CP refutation of F, where all coefficients in π have length at most $polylog(n)$. Given an assignment $\vec{\alpha}$ to the $\vec{z}$ variables, we want to show that there exists a monotone circuit $C(\vec{\alpha})$, of size polynomial in the size of π, that outputs $f_{A \wedge B}(\vec{\alpha})$.

Lemma 11. *There is a polynomial-sized circuit $C(\vec{\alpha})$ that outputs $f_F(\vec{\alpha})$.*

Proof. Let $\vec{\alpha}$ be an assignment to the variables $\vec{z}$. We want to show that for each line $f(\vec{x})+g(\vec{y})+h(\vec{\alpha}) \geq D$ in π, there exists D_0 and D_1, where $D_0 + D_1 \geq D - h(\vec{\alpha})$ such that, in Cutting Planes, we can derive $f(\vec{x}) \geq D_0$ from $A(\vec{x}, \vec{\alpha})$ and $g(\vec{y}) \geq D_1$ from $B(\vec{y}, \vec{\alpha})$. We prove this by induction on the number of derivation steps in the refutation.
(Base Case): This is trivially true for the initial clauses (when the proof has size 1).

(Linear Combination): Suppose that the j^{th} line of the proof is derived from the following two previously derived inequalities by the addition rule: $f_1(\vec{x})+g_1(\vec{y})+h_1(\vec{\alpha}) \geq C$ and $f_2(\vec{x})+g_2(\vec{y})+h_2(\vec{\alpha}) \geq D$. Then, by induction, we can derive the following linear inequalities:

$$f_1(\vec{x}) \geq C_0 \qquad g_1(\vec{y}) \geq C_1$$

where $C_0 + C_1 \geq C - h_1(\vec{\alpha})$.

And similarly,

$$f_2(\vec{x}) \geq D_0 \qquad g_2(\vec{y}) \geq D_1$$

where $D_0 + D_1 \geq D - h_2(\alpha)$.

From these derivations, we can apply the addition rule to each side separately to derive the following inequalities: $f_1(\vec{x}) + f_2(\vec{x}) \geq C_0 + D_0$ and $g_1(\vec{y}) + g_2(\vec{y}) \geq C_1 + D_1$.
(Division): Suppose that the j^{th} line of the proof was derived via the division rule from the linear inequality: $f(\vec{x}) + g(\vec{y}) + h(\vec{\alpha}) \geq D$. Then by induction, we can derive the following linear inequalities:

$$f(\vec{x}) \geq D_0 \qquad g(\vec{y}) \geq D_1$$

where $D_0 + D_1 \geq D - h(\vec{\alpha})$. Then we can apply the division rule using a value d that divides $f(\vec{x})$ and $g(\vec{y})$ to get:

$$\frac{1}{d} f(\vec{x}) \geq \lceil \frac{D_0}{d} \rceil \qquad \frac{1}{d} g(\vec{y}) \geq \lceil \frac{D_1}{d} \rceil$$

where $\lceil \frac{D_0}{d} \rceil + \lceil \frac{D_1}{d} \rceil \geq \lceil \frac{D_0 + D_1}{d} \rceil \geq \lceil D \rceil - \frac{h(\vec{\alpha})}{d}$.

Since we were able to derive $\lceil \frac{D_0+D_1}{d} \rceil \geq \lceil D \rceil - \frac{h(\vec{\alpha})}{d}$ by using the Division rule for CP, then we also derive $f(\vec{x}) \geq D_0$ from $A(\vec{x}, \vec{\alpha})$ and $g(\vec{y}) \geq D_1$ from $B(\vec{y}, \vec{\alpha})$.

Thus we can obtain CP derivations of $0 \geq D_0$ from $A(\vec{x}, \vec{\alpha})$ and $0 \geq D_1$ from $B(\vec{y}, \vec{\alpha})$. Since $D_0 \geq 1$ or $D_1 \geq 1$, we have a refutation $0 \geq 1$ from either $A(\vec{x}, \vec{\alpha})$ or $B(\vec{y}, \vec{\alpha})$. So our algorithm A computes $D_0 + D_1$ and outputs 1 if $D_1 \geq 1$ and 0 if $D_0 \geq 0$ in polytime. ☐

4.2.1 Lower Bounds for the Clique-Coclique Formula

Let us use everything we have learned so far and apply it to the Clique-Coclique formula.

Definition 12 (**Clique**). *In graph theory, a clique is a subset of vertices within a graph where every single vertex is directly connected to every other vertex in that subset.*

Definition 13 (**Coclique**). *A coclique is a set of vertices in a graph where no two vertices are connected by an edge.*

The Clique-Coclique formula is a split CNF formula defined as: $F = Clique_k(\vec{x}, \vec{z}) \wedge Color_{k-1}(\vec{y}, \vec{z})$, where $\vec{z}$-variables are an $m \times n$ matrix that represents an undirected graph $G = (V, E)$ where $|V| = n$. Furthermore, the $\vec{x}$-variables are a $k \times n$ matrix that represents a clique of size k and has a subset $S \subseteq V$ where $|V| = k$, and the $\vec{y}$-variables are an $(k-1) \times n$ matrix that represents a $(k-1)$ coloring of V.

Definition 14 ($Clique_k(\vec{x}, \vec{z})$). *$Clique_k(\vec{x}, \vec{z})$ is the conjunction of the following constraints:*
1) $\forall i \in [k] : \vee_{v=1}^{n} x_{i,v}$
2) $\forall i \neq j \in [k], \forall v \in [n] : \neg x_{i,v} \vee \neg x_{j,v}$
3) $\forall u \neq v \in [n], \forall i \neq j \in [k] : \neg x_{i,u} \vee \neg x_{j,v} \rightarrow z_{u,v}$
Constraint 1) and 2) says that $\vec{x}$ defines a subset $S \subseteq [n]$ of size $\geq k$. Constraint 3) says that all edges between clique vertices are in G.

Definition 15 ($Color_{k-1}(\vec{y}, \vec{z})$). *$Color_{k-1}(\vec{y}, \vec{z})$ is defined by the following constraints:*
1) $\forall u \in [n] : \vee_{v=1}^{n} x_{i,v}$
2) $\forall i \neq j \in [k-1], \forall u \in [n] : \neg y_{i,u} \vee \neg y_{j,u}$
3) $\forall u \neq v \in [n], \forall i \in [k-1] : \neg y_{i,u} \vee \neg y_{i,v} \vee \neg z_{u,v}$
Constraints 1) and 2) say that $\vec{y}$ represents a legal coloring of n. Constraint 3) says that there is no edge in G between vertices of different colors.

By construction we can define the following interpolant function:

$$f(\vec{z}) = \begin{cases} 1 & \text{if graph encoded by } \vec{z} \text{ contains a } k\text{-clique} \\ 0 & \text{if graph encoded by } \vec{z} \text{ has a } k-1 \text{ coloring} \\ * & \text{otherwise} \end{cases}$$

The Clique-Coclique formula has two properties: minterm and maxterm, which will be used to calculate the lower bounds of the formula.

Definition 16. *For a fix k, a minterm is a graph containing a k-clique and no other edges. A maxterm is a maximal graph that is $(k-1)$ colorable such that its vertices are partitioned into $k-1$ groups and $E_{i,j} = 1$ if and only if vertices i and j belong to different groups.*

Theorem 17 ([Raz85])**.** *Based on the construction of the formula and its min/max term properties, let $k = \sqrt{n}$. Then any monotone circuit C that accepts all minterms and rejects all maxterms has* $\text{size}(2^{\Omega(n^{\epsilon})})$ *for some $\epsilon > 0$.*

Now that we have analyzed the properties of this formula, let us apply concepts from CP refutations to develop a strong theorem on the lower bounds of this proof.

Theorem 18. *Any CP refutation of Clique-Coclique formula requires exponential time.*

Proof. By monotone feasible interpolation for CP, a size(s) CP refutation implies poly(s) size monotone real circuit for separating k-cliques from $(k-1)$ colorable graphs for all k-values. Furthermore, by Razborov (low coefficient cases) and Cook-Haken (for the more general case of coefficients of length $poly(s)$), monotone circuits for the clique-coclique formula requires $\text{size}(exp(n^{\epsilon}))$, therefore for $k = \sqrt{n}$, this implies a $2^{\Omega(n^{\epsilon})}$ lower bound on the size of CP refutations. □

5 Final Remarks and Further Research

For a long time the only lower bound for Cutting Planes was via monotone feasible interpolation, and therefore only for split-form formulas. However, recent work by [FPPR17] and [HP17] proved exponential lower bounds for random kCNFs, where $k = O(logn)$, by generalizing the feasible interpolation method for arbitrary formulas. Work in this area has led to the open question of whether we can improve the lower bounds for random kCNFs for $k = O(1)$.

We note that for Resolution, exponential lower bounds were proven for random kCNFs for $k = O(1)$ by Chvatal and Szemeredi [CS88].

References

[BPR00] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. On interpolation and automatization for frege systems. *SIAM Journal on Computing*, 29(6):1939–1967, 2000.

[CS88] Vasek Chvátal and Endre Szemerédi. Many hard examples for resolution. *J. ACM*, 35(4):759–768, 1988.

[FPPR17] Noah Fleming, Denis Pankratov, Toniann Pitassi, and Robert Robere. Random cnfs are hard for cutting planes. 03 2017.

[HP17] Pavel Hrubes and Pavel Pudlak. Random formulas, monotone circuits, and interpolation. pages 121–131, 10 2017.

[Raz85] A. A. Razborov. Lower bounds on the monotone complexity of some Boolean functions. *Dokl. Akad. Nauk SSSR*, 281(4):798–801, 1985.