Branching Programs with Bounded Repetitions and Flow Formulas

Anastasia Sofronova
St. Petersburg Department of Steklov Mathematical Institute of Russian Academy of Sciences, Russia

Dmitry Sokolov
St. Petersburg Department of Steklov Mathematical Institute of Russian Academy of Sciences, Russia
St. Petersburg State University, Russia

Abstract

Restricted branching programs capture various complexity measures like space in Turing machines or length of proofs in proof systems. In this paper, we focus on the application in the proof complexity that was discovered by Lovasz et al. [14] who showed the equivalence between regular Resolution and read-once branching programs for "unsatisfied clause search problem" (Search_φ). This connection is widely used, in particular, in the recent breakthrough result about the Clique problem in regular Resolution by Atserias et al. [5].

We study the branching programs with bounded repetitions, so-called $(1, +k)$-BPs (Sieling [21]) in application to the Search_φ problem. On the one hand, it is a natural generalization of read-once branching programs. On the other hand, this model gives a powerful proof system that can efficiently certify the unsatisfiability of a wide class of formulas that is hard for Resolution (Knop [13]).

We deal with Search_φ that is "relatively easy" compared to all known hard examples for the $(1, +k)$-BPs. We introduce the first technique for proving exponential lower bounds for the $(1, +k)$-BPs on Search_φ. To do it we combine a well-known technique for proving lower bounds on the size of branching programs [12, 21, 22] with the modification of the "closure" technique [1, 3]. In contrast with most Resolution lower bounds, our technique uses not only "local" properties of the formula, but also a "global" structure. Our hard examples are based on the Flow formulas introduced in [3].

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases proof complexity, branching programs, bounded repetitions, lower bounds

Digital Object Identifier 10.4230/LIPIcs.CCC.2021.17

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2021/028/`

Funding *Anastasia Sofronova*: The research presented in Sections 3 and 4 is supported by Russian Science Foundation (project 18-71-10042).

Acknowledgements The authors would like to thank Dmitry Itsykson, Artur Riazanov for fruitful discussions and comments, Edward Hirsch for comments on the draft, Alexander Knop for a statement of the problem. The authors would also like to thank anonymous reviewers for their valuable comments.

1 Introduction

Branching program is a computational model that generalizes decision tree in the most natural way: the underlying graph of computation can be an arbitrary directed acyclic graph. This is one of the most fundamental models in theoretical computer science: it captures the space complexity of many versions of restricted and unrestricted Turing machines, various proof systems may be described in terms of this model, etc.

A Shannon's style counting argument says that there is a boolean function such that any branching program that computes this function requires an exponential size. However for an explicit function, we are still far from a superpolynomial lower bound, and the best known result is $\frac{n^2}{\log^2 n}$ due to Nechiporuk [16].

For some applications, it is enough to deal with the restricted models of branching programs and many such models were considered. One of the most popular restrictions is the read-once model of branching programs [15] where any input bit may be queried at most once during each computation. This model corresponds to the *eraser Turing machines*. Exponential lower bounds for this model were proven in [24, 26]. For capturing more general machines some natural generalization of read-once branching programs were studied. And one of the most important models among these generalizations is the model with bounded repetitions aka $(1, +k)$-BP that was described in [21]. In this model, we allow our branching programs to requery variables, but on each computation only k input bits may be queried more than one time. There are two natural points of view on this model:

- *syntactic:* if we apply the restriction on *every* path;
- *semantic:* if we apply the restriction on *consistent* paths

(for formal definition see section 2.1). The semantic version is more powerful and may capture strong Turing machine models (for details see [12]).

Exponential lower bounds on $(1, +k)$-BP were shown in [12, 20–22] for various parameters k. Lower bounds from [12] hold for $k = \Omega\left(\frac{n}{\log n}\right)$ and the lower bound from [10] holds even for $k = \Omega(n)$, where n is the number of input bits. We refer the reader to the books [11, 25] with the detailed description of results related to branching programs.

Lower bounds for $(1, +k)$-BP described above are given for "complicated" functions (usually it is characteristic functions of an error-correcting code with additional properties). In particular, these functions are complicated in terms of the certificate complexity. Unfortunately, for some applications, it is not enough. Following [14], we have a connection between proof systems and branching programs in application to the "unsatisfied clause search problem". Hence for the lower bounds in proof complexity we want to deal with this search problem, which is an "easy" problem. Namely, it can be described by a small collection of certificates. In this paper we introduce a technique for proving such lower bounds on the semantic $(1, +k)$-BP where $k = \mathcal{O}(\log n/ \log\log n)$ where n is the number of variables.

1.1 Search Problem and Proof Systems

Consider a **search problem**, defined by a relation $S \subseteq \mathcal{I} \times \mathcal{O}$ for some finite sets $\mathcal{I}$ and $\mathcal{O}$. On input $x \in \mathcal{I}$ the search problem is to find some output in $S(x) := \{o \in \mathcal{O} \mid (x, o) \in S\}$. In this paper we study the "unsatisfied clause search problem" for a CNF formula.

▶ **Definition 1.** *A **unsatisfied search problem** Search_φ for an unsatisfiable CNF formula $\varphi := \bigwedge_{i \in I} C_i$ on n variables is defined as follows:*

- *input: an n-variable assignment $z \in \{0,1\}^n$;*
- *output: an element $i \in I$ such that clause C_i of φ is falsified by z.*

Informally speaking, we may think that if we can solve the Search_φ problem in some computational model $\mathfrak{C}$, then the description of $C \in \mathfrak{C}$ that solves Search_φ is a "certificate of unsatisfiability" of a formula φ. So we may think of this model as a proof system. We do not want to formalize this statement for a general computational model, but we prove the formal statement for the branching programs (see section 5).

The proof system that is defined by the read-once branching programs is equivalent to regular Resolution [14]. This connection is widely used in proof complexity. As an example, we can consider first lower bounds on the regular Resolution and Resolution proofs of the Weak Pigeonhole Principle [17, 18], recent breakthrough result: a lower bound on the regular Resolution proofs of the Clique formulas [5].

The connection between regular Resolution and branching programs makes it interesting to consider some less restricted models of branching programs in application to the Search_φ problems. Some of these models were considered in [13]. In this paper we focus on $(1, +k)$-BPs. Despite the success in proving lower bounds on the Resolution (and hence read-once programs) the lower bounds for $(1, +k)$-BP on the Search_φ are an open question even for $k = 1$.

Previous Techniques

The behaviour of branching programs on functions differs from the behaviour on the Search_φ problem. For example, the unrestricted programs may solve Search_φ for any φ in linear size (we can implement a simple algorithm that checks clauses of φ step by step). Informally speaking, as we said above, the lower bounds for functions on $(1, +k)$-BP [12, 20–22] heavily used the fact that there is no efficient description of these functions in terms of certificates. However Search_φ is defined by a small set of certificates. Considering this difference, it is unclear how to use the classical techniques for our problem.

Another issue is that $(1, +k)$-BP is much stronger than general Resolution on some classes of formulas [13] even for small constant k and syntactic model. This is a crucial observation and it means that we cannot directly apply general techniques for proving lower bounds in proof complexity like [3, 6] etc., since these techniques cannot distinguish between considered classes of formulas and other hard examples for Resolution. Hence if we want to prove lower bound for $(1, +k)$-BP on Search_φ we need some additional arguments in comparison to Resolution lower bounds.

1.2 Our Results

The main result is an exponential lower bound on the size of $(1, +k)$-BPs in application to Search_φ for $k = \mathcal{O}\left(\frac{\log n}{\log\log n}\right)$.

▶ **Theorem 2.** *For $n \in \mathbb{N}$ and $k_0 := k_0(n)$ there is an unsatisfiable formula φ on n variables of size $n^{\mathcal{O}(k_0)}$ such that any semantic $(1, +k_0)$-BP solving Search_φ requires size $\exp\left[\Omega\left(\frac{n}{2^{\mathcal{O}(k_0)}}\right)\right]$.*

We also show that $(1, +k)$-BPs define a proof system in terms of Cook–Reckhow definition [7] $(1, +k)$-BP-PS (see section 5). That is a generalization of the result from [13], where it was shown only for ℓ-CNF where ℓ is an absolute constant.

▶ **Theorem 3.** *A syntactic $(1, +k)$-BP-PS is a proof system in terms of Cook–Reckhow definition for any constant $k \in \mathbb{N}$.*

1.3 Technique

The key ingredients for the lower bound are:

- *garlands:* aka (s, ℓ)-chains, that is a standard technique for proving lower bounds on the branching programs [11, 12, 20–22];
- *closure:* a technique that allows to make large partial restriction and keep the search problem hard for branching programs (and proof systems) [1, 3];

- *amplification:* a trick from [2] that makes formula hard for regular Resolution (and read-once branching programs) and help us to force the branching program to use the repetitions in a very structured way;
- *Flow-Cut:* the famous Theorem [8] that shows the duality between the maximum flow and the minimum cut, that we use to extend partial assignments to total assignments with good properties.

Let us introduce a general sketch of the proof. In section 4.1 we define an unsatisfiable formula Flow_G [3] that states: in graph G we have a source of a flow but there is no sink. We require graph G to be an algebraic expander, but, in fact, we need two properties:

- G is a combinatorial expander; namely, each set of vertices of size at most $r = \Omega\left(\frac{n}{\log n}\right)$ has a lot of neighbours (this is a "local" property, since we care only about small enough sets);
- max-balanced-cut of the graph G is large enough (this is a "global" property of the graph G), where "balanced" means that each piece has size at least $\Omega(r)$.

It is not clear how to show the lower bound for this formula itself and we amplify Flow_G formulas by using the trick from [2]. Denote the result of amplification by φ.

1. For the sake of contradiction we assume that we have a small $(1, +k)$-BP solving Search_φ. We generate a big family of paths and, using the upper bound on the size of our program, we find some paths in the program that form a "garland" structure (see section 4). This idea is similar to the idea from [12].
2. These paths correspond to some assignments and we keep our formula "hard" under these assignments. To do it we use a modification of the "closure" technique [3] (an easier version of this iterative modification was used in [23]). Here we use a combinatorial expansion of the graph G.
3. By using the fact that we deal with an amplified version of Flow_G we show that from the end point of paths that form the garland we cannot reach any leaf that is marked by one of the clauses from some set $T \subseteq \varphi$. Here we use the fact that k is small enough.
4. To conclude the proof, we use the Max-Flow Min-Cut Theorem (and global properties of our graph) to show that there should be some path from the garland to some clause from the set T.

   See section 4 for more details.

2 Preliminaries

Let X be a set of boolean variables. For a variable $x \in X$ we denote $x^1 := x$ and $x^0 := \neg x$. We say that $\alpha\colon X \to \{0, 1, *, ?\}$ is a **generalized partial assignment** and α **assigns** or **touches** $x \in X$ iff $\alpha(x) \in \{0, 1, ?\}$. And an assignment γ is an **instance** of α iff:

- $\alpha(x) \in \{0, 1, *\}$ implies $\gamma(x) = \alpha(x)$;
- $\alpha(x) =?$ implies $\gamma(x) \in \{0, 1\}$.

If α and β are two partial assignments to variables from the set X, we say that a generalized partial assignment $\alpha \uplus \beta\colon X \to \{0, 1, *, ?\}$ is a **joint assignment** iff:

- if $\alpha(x) = a$ and $\beta(x) \in \{a, *\}$, then $\alpha \uplus \beta(x) = a$;
- if $\beta(x) = a$ and $\alpha(x) \in \{a, *\}$, then $\alpha \uplus \beta(x) = a$;
- if $\alpha(x) = a$ and $\beta(x) = 1 - a$, then $\alpha \uplus \beta(x) =?$;
- if $\alpha(x) = \beta(x) = *$, then $\alpha \uplus \beta(x) = *$,

where $a \in \{0, 1\}$.

We will also use the famous Max-Flow Min-Cut Theorem.

▶ **Theorem 4** (Max-Flow Min-Cut [8]). *Let $G := (V, E)$. For any $s, t \in V$ the maximum value of an s-t flow is equal to the minimum capacity over all s-t cuts.*

2.1 Branching Programs

Let $X := \{x_1, \ldots, x_n\}$ be a set of propositional variables and $\mathcal{O}$ be a finite set. A **branching program** is a directed acyclic graph with one source. Every vertex of the graph is labeled by a variable from X, or by an element of the set $\mathcal{O}$ with respect to the following properties:

- if a vertex is labeled by $o \in \mathcal{O}$, then it is a sink;
- if a vertex is labeled by a variable, then it has exactly two outgoing edges: one edge is labeled by 0 and the other one is labeled by 1.

Every branching program B defines a function $f_B\colon \{0,1\}^n \to \mathcal{O}$. We assume that every input $z \in \{0,1\}^n$ induces a path from source to sink in a natural way. If this path ends in a vertex with a label $o \in \mathcal{O}$ then we define $f_B(z) := o$.

We say that B is a **branching program for the relation $S \subseteq \{0,1\} \times \mathcal{O}$** iff f_B is consistent with S: namely if $f_B(z) = o$ then $(z, o) \in S$.

Let D be a branching program and v is a node in it. The **subprogram** of D with the root v we denote by $D(v)$ and define as a subgraph of D that is reachable from v. Also for a partial assignment ρ we define a branching program $D|_\rho$ as the following transformation applied to D:

- for each variable y to which ρ assigns a value a, contract edges $y = a$ and delete edges $y = \neg a$;
- delete all vertices that are unreachable from the root.

These operations only decrease the size of the program.

If p is a consistent path in a branching program, we denote a partial assignment that corresponds to this path by τ_p.

Let us also define some classical restrictions of the general branching programs.

▶ **Definition 5.** *Let B be a branching program. We say that B is a **(syntactic) read-once branching program** or* 1-BP *iff on every path from the source to a sink we can see each variable at most once.*

*We say that B is a **$(1, +k)$-BP** iff on every path p from the source to a sink there is a set of variables X_p of size at most k such that all other variables appear in p at most once. And we can twist this definition a little bit and say that B is a **semantic $(1, +k)$-BP** iff on every consistent path from p source to sink there is a set of variables X_p of size at most k such that all other variables appear in p at most once.*

If a branching program B computes a boolean function, we say that it is **satisfiable** iff f_B is not identically zero.

▶ **Theorem 6** (Savický [19]). *There is an algorithm to check a satisfiability of a syntactic* $(1, +k)$-BP *in time $\mathcal{O}\left[\left(\frac{4en}{k}\right)^k sn\right]$.*

The following algorithm also will be useful for us.

▶ **Theorem 7** (Savický [19]). *The test whether an input branching program is a syntactic* $(1, +k)$-BP *can be done in time $\mathcal{O}\left[\left(\frac{3en}{k+1}\right)^{k+1} s\right]$.*

The next observation is natural and extremely useful for proving lower bounds.

▶ **Lemma 8.** *Let D be a $(1,+k)$-BP for Search_φ, p be a consistent path from the root to some node v. If p has a variable x queried more than one time on it then $D(v)|_{\tau_p}$ is a $(1,+(k-1))$-BP for the $\mathsf{Search}_{\varphi|_{\tau_p}}$. The result holds for both: semantic and syntactic models.*

Proof. A program $D(v)|_{\tau_p}$ is a program for the $\mathsf{Search}_{\varphi|_{\tau_p}}$ by the correctness of the program D. Consider a path s in D from v to some leaf. Let X_s be a set of variables that are queried more than one time on s. If $|X_s| = k$ and $x \notin X_s$, the path ps has at least $k+1$ variables that are queried more that one time. This is a contradiction. If $|X_s| = k$ and $x \in X_s$, note that in $D(v)|_{\tau_p}$ we contract all edges that correspond to the x variable and hence we transform this path into a path with at most $k-1$ repetitions. ◀

3 Expanders

We are given a graph $G := (V, E)$. For two subsets of vertices A, B we write $E(A, B)$ to denote the set of pairs (v, e) where $v \in A$, e is an edge that is incident to v and e connects v with some vertex in B. We will think about it as about set of edges between A and B, but if A and B intersect we count edges within intersection twice. We also use a shortcut notations $E(S) := E(S, V)$ and $\overline{S} := V \setminus S$. If the graph we consider is unclear from the context we specify it as a subscript: $E_G(A, B)$.

▶ Remark 9. Assuming that G is Δ-regular graph this definition allows us to use natural equalities:

- $|E(S)| = \Delta|S|$;
- $|E(A, \overline{A})| = \Delta|A| - |E(A, A)|$.

We write $\mathrm{N}_G(v)$ to denote the set of **neighbours** of v in the graph G. We extend this notion to sets and denote by $\mathrm{N}_G(S) := \{v \mid \exists u \in S,\ (u, v) \in E\}$ the **neighbourhood** of a set of vertices $S \subseteq V$.

A graph $G := (V, E)$ is an **(n, Δ, α)-algebraic expander** (or just **expander**), if:

- $|V| = n$;
- the degree of any vertex $v \in V$ equals Δ;
- the absolute value of the second largest eigenvalue of the adjacency matrix of G is at most $\alpha\Delta$.

▶ **Lemma 10** (Mixing Lemma [4]). *Let $G := (V, E)$ be an (n, Δ, α)-expander. For any two subsets $A, B \subseteq V$ the following holds:*

$$\left| |E(A, B)| - \frac{\Delta|A||B|}{n} \right| \le \alpha\Delta\sqrt{|A||B|}.$$

We also need *combinatorial* edge expansion. We say that $G := (V, E)$ satisfies **(r, β)-(edge) expansion property** for some $r, \beta > 0$, if for all $S \subseteq V$ of size at most r holds $E(S, \overline{S}) \ge \beta\Delta|S|$. The Mixing Lemma says that any expander graph satisfies expansion property for suitable parameters.

▶ **Corollary 11.** *If $G := (V, E)$ is an (n, Δ, α)-expander, then for any $0 < \beta < 1 - \alpha$ the graph G satisfy $((1 - \alpha - \beta)n, \beta)$-expansion property.*

Proof. Consider some $A \subseteq V$ of size at most $(1 - \alpha - \beta)n$. Note that $|E(A, \overline{A})| = \Delta|A| - |E(A, A)|$. By Mixing Lemma:

$$|E(A, A)| \le \frac{\Delta|A|^2}{n} + \alpha\Delta|A| = \Delta|A|\left(\frac{|A|}{n} + \alpha\right) \le \Delta|A|(1 - \beta).$$

Hence $|E(A, \overline{A})| \ge \beta\Delta|A|$ by Remark 9. ◀

The "vertex analog" of the next proposition is well known in the literature (for example [9]). We turn it into edge version.

▶ **Proposition 12.** *Let $G := (V, E)$ be a graph of degree Δ. If G satisfies (r, β)-expansion property then for any set $S \subseteq V$ of size $k \leq r$ there is an enumeration $v_1, v_2, \ldots, v_k \in S$ and a sequence $R_1, \ldots, R_k \subseteq E(S)$ such that:*

- $R_i = E(\{v_i\}, V \setminus \{v_1, v_2, \ldots, v_i\})$;
- $|R_i| \geq \beta\Delta$.

Proof. We create this sequence in reversed order. Since $|S| \leq r$, it holds that $|E(S, \overline{S})| \geq \beta\Delta|S|$ and there is a vertex $v_k \in S$ such that $|E(\{v_k\}, \overline{S})| \geq \beta\Delta$. Let $R_k := |E(\{v_k\}, \overline{S})|$, and repeat the process for $S \setminus \{v_k\}$. ◀

4 Lower Bounds for $(1, +k)$-BP

In this section, we will prove the following theorem:

▶ **Theorem 13** (2)**.** *For $n \in \mathbb{N}$ and $k_0 := k_0(n)$ there is an unsatisfiable formula φ on n variables of size $n^{\mathcal{O}(k_0)}$ such that any semantic $(1, +k_0)$-BP solving Search_φ requires size $\exp\left[\Omega\left(\frac{n}{2^{\mathcal{O}(k_0)}}\right)\right]$.*

Let us describe the main ideas used in the proof. To prove this Theorem we would like to construct an exponentially big set of paths, which cannot be compactly "glued" together in $(1, +k)$-BP, correctly solving Search_φ.

To give a detailed plan we need an auxiliary definition.

▶ **Definition 14.** *A* ***ℓ-garland*** *in a branching program is a pair of paths (a, b) from the root such that $a := v_0 a_1 v_1 a_2 v_2 a_3 \ldots a_\ell v_\ell$ and $b := v_0 b_1 v_1 b_2 v_2 b_3 \ldots b_\ell v_\ell$ where a_i, b_i are possibly empty paths and paths $v_j a_{j+1} v_{j+1}$ and $v_j b_{j+1} v_{j+1}$ are different for all $0 \leq j < \ell$ (see Fig. 1).*

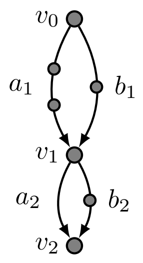

Figure 1 2-garland.

Let us consider the detailed plan.

1. By induction on k we want to show that $\mathsf{Search}_{\varphi|_\rho}$ is hard for $(1, +k)$-BP even after some "good" restriction ρ.
2. For the sake of contradiction we assume that we have a small $(1, +k)$-BP solving $\mathsf{Search}_{\varphi|_\rho}$. In the section 4.2.1 we generate a family of paths starting from the root of the program and find in this family a $(k + 1)$-garland (see Fig. 1). This idea is similar to [12].
3. To argue that we can find a garland we generate exponentially many paths by walking from root (section 4.2). During this process, we have to make sure that on these paths our branching program cannot determine an answer (that would mean that we cannot walk anymore). To avoid it we use the "closure" technique that is motivated by technique from [1,3] and avoid "local contradictions". And hence we have to choose the formula φ very carefully, but we still have some freedom.

4. If we found a repetition while constructing a garland, we use Lemma 8 and apply induction hypothesis. This is a place where we use that formula φ is still hard even after the restriction.
5. In the section 4.2.2 we combine different parts of garland and argue about the reachability of certain leaves. We have to make sure that the paths we consider are consistent and that when we reach the endpoint of the garland, formula φ remains hard. We achieve it by using the following properties.
   - We have already removed repetitions from the garland by using Lemma 8 and induction hypothesis.
   - To show that combinations of different parts of the garland give us consistent paths we equip the closure technique by the notion of "strongly satisfied" (see Section 4.1.1) constraints. This is the second place that requires specific properties of the formula φ.

   At the end of this section, we will have a set of clauses $\mathfrak{C} \subseteq \varphi$ such that leaves marked by elements of this set should be unreachable from the endpoint of the garland.
6. For the last part (section 4.2.3) we consider an arbitrary path r in our garland and note that $\varphi \setminus \mathfrak{C}$ is a satisfiable formula even under the restriction τ_r. It is hard to show this property for the formulas that encode natural combinatorial principles. We use the trick from [2] to change the formula φ to make sure that $\mathfrak{C}$ is large enough.
   Here we use the global structure of our formula φ (in our case we use the Max-Flow Min-Cut Theorem) to satisfy all clauses in $\varphi \setminus \mathfrak{C}$.

We start with defining the hard formulas on a suitable expander graph.

4.1 Hard Formulas

Let $G \coloneqq (V, E)$ be a directed graph. Each edge $e \in E$ has the corresponding variable x_e, where $x_e = 1$ indicates that a flow of size 1 is going through an edge e. Let u be an arbitrary, but fixed vertex of the graph.

The formula $\mathsf{Flow}_{G,u}$ consists of the following constraints written in CNF for all $v \in V$:

$$\sum_{e \in E : \mathrm{st}(e) = v} x_e - \sum_{e \in E : \mathrm{en}(e) = v} x_e \geq c(v),$$

where $e = (\mathrm{st}(e), \mathrm{en}(e))$ and $c \colon V \to \{0, 1\}$ is a labeling function:
- $c(v) = 0$, for all $v \in V \setminus \{u\}$;
- $c(u) = 1$.

This formulas states: for all vertices in the graph the flow is non-negative, and at least for one vertex it is strictly positive. It is easy to see that $\mathsf{Flow}_{G,u}$ is unsatisfiable. We omit index u since in our applications it is an arbitrary vertex.

We use the most naive CNF encoding of these constraints. We represent each constraint separately. Consider a vertex $v \in V$ and a set of edges $E_v \coloneqq \{e_1, e_2, \dots, e_s\} \subseteq E$ that are incident to v. Let $\rho_v \colon E_v \to \{0, 1\}$ be an assignment that violates the constraint in v. In this case we add to the formula a clause C:

$$x_{e_1}^{1-\rho(x_{e_1})} \vee x_{e_2}^{1-\rho(x_{e_2})} \vee \dots \vee x_{e_s}^{1-\rho(x_{e_s})},$$

and we also say that this assignment has a **gap**:

$$g(\rho_v) \coloneqq c(v) - \sum_{e \in E : \mathrm{st}(e) = v} \rho_v(x_e) + \sum_{e \in E : \mathrm{en}(e) = v} \rho_v(x_e).$$

For our purpose we consider Flow_G based on expanders. To be precise, we start with a graph G that is an (n, Δ, α)-expander, where $\Delta = \Theta(\log n)$ and α is some fixed constant, and replace each undirected edge by two directed edges (we say that these edges are **dual**). The exact value of Δ depends on a value of k.

▶ Remark 15. We consider only **proper** partial assignments ρ that satisfy the following property for all pairs of dual edges (e, e'):
- $\rho(x_e) \in \{0, 1\}$ iff $\rho(x_{e'}) \in \{0, 1\}$;
- if $\rho(x_e) = 1$ then $\rho(x_{e'}) = 0$.

We also identify $\mathrm{supp}(\rho)$ with an undirected set of edges that are assigned by ρ.

To make the formula somewhat "confusing" for $(1, +k)$-BP, we would like to add more variables to clauses. These variables do not really affect the physical meaning of the formula, but make it hard for $(1, +k)$-BP to extract additional information from repetitions on paths. This transformation is sensitive to the exact CNF encoding of the constraints that is written above.

▶ **Definition 16.** *Let $G := (V, E)$ be an undirected graph and $\mathcal{C}_v$ be a subset of clauses corresponding to vertex v in Flow_G. Let $\eta_v^k \colon \mathcal{C}_v \to \binom{E}{k}$ be a mapping, and $\eta^k := \{\eta_v^k \mid v \in V\}$ be a family of such mappings. We define $\mathsf{Flow}_G^{\eta^k}$ the following way:*
- *for each $v \in V$ we consider each $C \in \mathcal{C}_v$;*
- *we take $\eta_v^k(C) = \{e_1, \ldots, e_k\}$, which is a set of k edges;*
- *we replace C by 2^{2k} clauses of the form:*

$$C \vee \bigvee_i x_{s_i}^{a_i} \vee x_{s'_i}^{a'_i}$$

enumerated by $a_i, a'_i \in \{0, 1\}$, where $i \in [k]$ and s_i, s'_i are directed copies of the edge e_i.

As described in the plan, at some point in the proof we would like to construct an assignment that leaves certain clauses (to which a certain set of variables was added) unsatisfied. For our purpose, we would like those clauses to "strongly unsatisfy" the condition in their vertices.

Let us describe the construction of η^k. Assume that $\Delta \geq 50 \cdot k \log n$. For each $v \in V$ we define η_v independently. We will be interested in adding variables to clauses which correspond to large incoming flow.

1. Let us consider a set of clauses $\mathcal{C}$ that corresponds to v and a proper partial assignment on edges incident to v with gap equal to $\frac{\Delta}{4} + 1$.
2. Note that $|\mathcal{C}| \geq \binom{\Delta}{\Delta/4} \geq 4^{\Delta/4} \geq n^{4k}$. The first inequality holds since we can choose arbitrary $\Delta/4 + 1$ incoming edges to obtain the desired gap and set all other incident edges to zero.
3. There are at most $\binom{n\Delta}{k} \leq \left(\frac{n\Delta e}{k}\right)^k \leq n^{2k}$ different sets of k edges. Hence we can choose a subset of $B \subseteq \mathcal{C}$ and define η_v^k to be a bijection between B and all possible choices of sets of k edges.

Note that the existence of $(1, +k)$-BP of size S solving $\mathsf{Search}_{\mathsf{Flow}_G^{\eta^k}}$ (for any η^k) implies the existence of $(1, +k)$-BP of size S solving $\mathsf{Search}_{\mathsf{Flow}_G}$.

4.1.1 Locally Consistent Assignments

We need a notion of "good assignments", i.e. assignments that reduce Flow_G formulas to smaller, but "equally hard" instances.

Let $G := (V, E)$ be a graph. A proper assignment ρ δ-satisfies a set of vertices $U \subseteq V$ iff for all $v \in U$ the following holds:

- ρ assigns all edges that are incident to v;
- ρ satisfies the constraint for v;
- $\sum\limits_{e \in E : \mathrm{st}(e) = v} \rho(x_e) \geq \delta \cdot \Delta$.

We also say that a proper assignment ρ is **(r, δ, β)-locally consistent** iff there is a set of vertices V_ρ of size at most r such that:

- ρ δ-satisfies V_ρ;
- $(V \setminus V_\rho, E \setminus \mathrm{supp}(\rho))$ satisfies (r, β)-expansion property.

▶ Remark 17. If ρ is an (r, δ, β)-locally consistent assignment for some $\beta > 0$, then V_ρ is uniquely defined.

Proof. For the sake of contradiction assume that there are two candidates A, B. Wlog $A \setminus B \neq \emptyset$. Pick an arbitrary vertex $v \in A \setminus B$. Since A satisfies the required properties, ρ assigns all edges that are incident to v, which contradicts the fact that $(V \setminus B, E \setminus \mathrm{supp}(\rho))$ satisfies (r, β)-expansion property. ◀

4.2 Proof of Theorem 2

Let G be an (n, Δ, α)-expander and η^{k_0+1} be a mapping defined in section 4.1. In this section we prove an exponential lower bound on $\mathsf{Search}_{\mathsf{Flow}_G^{\eta^{k_0+1}}}$ for $(1, +k_0)$-BP. We assume that n is large enough.

Let us fix some parameters:

- $\Delta := 100 k_0 \log n$ and $\Delta > 200$;
- $\alpha := 0.01$ is the second eigenvalue of the normalized adjacency matrix of G;
- $r := \frac{n}{\Delta}$ and $\beta := 0.96$ is the "combinatorial expansion" of the graph G;
- $\beta' := 0.95$ is an expansion parameter that we try to maintain after removing some vertices and edges from G;
- $\nu_k := \left(\frac{1}{4}(\beta - \beta')\right)^{k+3}$ is a scaling factor that indicates the fraction of edges that we want to assign in our partial assignment.

Note that $r \ll (1-\beta-\alpha)n = 0.03 \cdot n$ and hence by Corollary 11 G satisfies (r, β)-expansion property, hence we can use all combinatorial expansion properties and tools.

To formulate the induction hypothesis we need one more definition. Let $M \subseteq E$ and ρ is a proper assignment. We say that ρ is **γ-minimal local consistent extension** or ***(mlce)*** on M iff:

- ρ is $(r, 0.6, \gamma)$-locally consistent assignment;
- $\mathrm{supp}(\rho) = M \cup E(V_\rho)$;
- $|E(V_\rho, \overline{V_\rho}) \setminus M| < \gamma \Delta |V_\rho|$.

Informally we may think about it in the following way: after we assign edges from M somehow, ρ should assign also V_ρ as a "minimal" set of vertices to take care of in order to be locally consistent.

Let $\varphi := \mathsf{Flow}_G^{\eta^{k_0+1}}$. By induction on $k \leq k_0$ we show the following statement. For all sets of edges $M \subseteq E$ of size at most $\nu_k \Delta r$ and all β'-mlce ρ on M any $(1, +k)$-BP for $\mathsf{Search}_{\varphi|_\rho}$ has size at least $2^{\frac{\nu_k}{4(k+1)^2}\Delta r}$.

Fix some M, ρ, $0 \leq k \leq k_0$ and for the sake of contradiction assume that we have a $(1, +k)$-BP D of size $2^{\frac{\nu_k}{4(k+1)^2}\Delta r}$ for $\mathsf{Search}_{\varphi|_\rho}$.

4.2.1 Construction of the Garland

To fulfill our plan of the proof, described at the beginning of the section, we start constructing the garland by obtaining an exponentially big set of paths with the corresponding assignments. Let us remind that $|M| \leq \nu_k \Delta r$ and ρ is β'-mlce on M.

We say that triple (p, U_p, σ_p) is **γ-good** iff:

- p is a path from the root of the branching program;
- U is a subset of edges such that corresponding variables are queried on p, so-called "branching variables";
- σ_p is a partial assignment such that:
  - σ_p extends $\rho \cup \tau_p$;
  - σ_p is a γ-mlce on $M \cup U_p$;
  - σ_p 0.8-satisfies $V_{\sigma_p} \setminus V_\rho$,

  where τ_p is an assignment that corresponds to p.

We maintain the set of β'-good triples $\mathcal{P}$ and an auxiliary set $\mathcal{S}$ of triples that appear in the set $\mathcal{P}$ at some moment during the process. In the beginning of our construction $\mathcal{P} \coloneqq \{(\emptyset, \emptyset, \rho)\}$ and $\mathcal{S} \coloneqq \mathcal{P}$.

We repeat the following process while we have at least one triple $(p, U_p, \sigma_p) \in \mathcal{P}$ such that $|U_p| \leq \nu_k \Delta r$.

Consider the triple described above. Let v be the end of p and x_e be the variable asked in v.

1. If x_e was queried on p we stop the process. In this case we return "Repetition" and we remember the path p.
2. Erase the triple (p, U_p, σ_p) from $\mathcal{P}$.
3. If $\sigma_p(x_e) \in \{0, 1\}$, then we continue along the edge $x_e = \sigma_p(x_e)$. Consider a path p' that is the extension of p along this edge, $U_{p'} \coloneqq U_p$ and $\sigma_{p'} \coloneqq \sigma_p$. Put $(p', U_{p'}, \sigma_{p'})$ into $\mathcal{P}$ and $\mathcal{S}$ and repeat the process from the beginning.
4. If $\sigma_p(x_e) = *$, then it is a "branching node", and we call this step a **branching step**.
   a. Let p' be a path obtained by continuing p along the edge $x_e = 0$, and p'' be a path obtained by continuing p along the edge $x_e = 1$.
   b. $U_{p'} \coloneqq U_p \cup e$, $U_{p''} \coloneqq U_p \cup e$.
   c. $\tau' \coloneqq \sigma_p \cup \{x_e = 0, x_{e'} = 0\}$, $\tau'' \coloneqq \sigma_p \cup \{x_e = 1, x_{e'} = 0\}$, where $x_{e'}$ is a dual edge.
   d. $(p', U_{p'}, \tau')$ is $(\beta' - 0.01)$-good triple. We extend an assignment τ' to make this triple β'-good. For the formal statement see Lemma 18. Here we describe an idea. Let $R \subseteq E$ be a set of edges that are unassigned by τ' (or τ''), and $B \subseteq V \setminus V_{\sigma_p}$ be the maximal set of vertices that satisfies:
      - $|B| \leq r$;
      - $|E(B, \overline{B}) \cap R| \leq \beta' \Delta |B|$.

      Let κ be an assignment on variables that correspond to edges in the set $E(B) \setminus \mathrm{supp}(\tau')$ such that $\tau' \cup \kappa$ 0.8-satisfies the constraints for all $v \in B$. This assignment κ always exists (and moreover it is independent of the value of x_e, but we do not use this fact).
   e. We denote $\sigma_{p'} \coloneqq \tau' \cup \kappa$, $\sigma_{p''} \coloneqq \tau'' \cup \kappa$ and put $(p', U_{p'}, \sigma_{p'})$ and $(p'', U_{p''}, \sigma_{p''})$ into $\mathcal{P}$ and into $\mathcal{S}$.

To conclude the construction we want to show the following claims.

- **Repetition case.** In the first case of the proof (if we have a repetition) we can reduce the problem to a lower bound on $(1, +(k-1))$-BP.
- **Correctness.** The branching step can be done and triples $(p', U_{p'}, \sigma_{p'})$ and $(p'', U_{p''}, \sigma_{p''})$ satisfy the required properties.
- **Garland extraction.** Among these paths we can find an k-garland (a, b) and a locally consistent extension of ρ.

4.2.1.1 Correctness

We show that if we have a triple (p, U_p, τ_p) which is β'-good then after processing it with our algorithm we also put in our sets β'-good triples. Let us formulate the general Lemma that helps us with it.

▶ **Lemma 18.** *Let (p, U_p, σ_p) and (q, U_q, σ_q) be* 0.9*-good triples. Then there is an assignment κ such that:*

- *for any γ that is an instance of $\sigma_p \uplus \sigma_q$ an assignment $\gamma \cup \kappa$ is a β'-mlce on* $\operatorname{supp}(\sigma_p) \cup \operatorname{supp}(\sigma_q)$*;*
- $|\operatorname{supp}(\gamma \cup \kappa)| \leq \nu_{k-1}\Delta r$.

Moreover if $p = q$ then triple $(p, U_p, \sigma_p \cup \kappa)$ is β'-good.

Proof. The proof was motivated by the closure technique developed in [1,3]. For the full version of the proof see Appendix A. ◀

If the branching step was not done, then we do not change U and τ, and we extend the path p according to the assignment τ hence the triple remains β'-good. We are left with the branching step. Note that $(p', U_{p'}, \tau')$ is 0.9-good and we apply Lemma 18 to a pair composed of two identical triples $(p', U_{p'}, \tau')$ and obtain κ that satisfies the required properties.

4.2.1.2 Repetition case

First let us note that if there is a repetition, then $k > 0$. Suppose we found a repetition while considering a triple (p, U_p, σ_p). The size of $M \cup U_p$ is at most $2\nu_k\Delta r$ and σ_p is β'-mlce on $M \cup U_p$. Let v be an end node of p. The program $D(v)|_{\sigma_p}$ is a $(1, +(k-1))$-BP for $\mathsf{Search}_{\mathsf{Flow}_G^{\eta^k}|_{\sigma_p}}$ by Lemma 8. Thus by induction hypothesis we have a lower bound of $2^{\frac{\nu_{k-1}}{4k^2}\Delta r} \geq 2^{\frac{\nu_k}{4(k+1)^2}\Delta r}$ on the size of $D(v)|_{\sigma_p}$ and in this case we are done.

4.2.1.3 Garland extraction

The following Lemma gives us a pair of triples $(p, U_p, \sigma_p), (q, U_p, \sigma_q) \in \mathcal{P}$ such that (p, q) forms a $(k+1)$-garland.

▶ **Lemma 19.** *There are $(p, U_p, \sigma_p), (q, U_q, \sigma_q) \in \mathcal{S}$ such that (p, q) forms a $(k+1)$-garland.*

Proof. For the proof see Appendix B. ◀

To continue the proof we need some additional property that we can "avoid repetitions" in this garland. We say that there is a **repetition in a garland** $p = v_0p_1v_1p_2v_2p_3 \dots p_{k_0+1}v_{k_0+1}$ and $q = v_0q_1v_1q_2v_2q_3 \dots q_{k_0+1}v_{k_0+1}$ iff there is **path in the garland**, i.e. path r of the form $v_0r_1v_1r_2v_2r_3 \dots r_{k_0+1}v_{k_0+1}$, such that some variable is queried more than one time on it, where $r_i \in \{p_i, q_i\}$.

Consider a path r in our garland (p, q) that contains a repetition and $r' \subseteq r$ the largest initial segment of r without repetitions. Let v be its end node. We apply Lemma 18 to triples $(p, U_p, \sigma_p), (q, U_q, \sigma_q)$, which gives us assignment κ, and choose a instance γ of $\sigma_p \uplus \sigma_q$ that is consistent with $\tau_{r'}$. Moreover, $|\operatorname{supp}(\gamma \cup \kappa)| \leq \nu_{k-1}\Delta r$, and $\gamma \cup \kappa$ is a β'-mlce on $\operatorname{supp}(\sigma_p) \cup \operatorname{supp}(\sigma_q)$. Hence by Lemma 8 we can use the induction hypothesis for $(1, +k-1)$-BP $D(v)|_{\tau_{r'}}$ and formula $\varphi|_{\gamma\cup\kappa}$. The size of $D(v)|_{\tau_{r'}}$ is at least $2^{\frac{\nu_{k-1}}{4k^2}\Delta r} \geq 2^{\frac{\nu_k}{4(k+1)^2}\Delta r}$.

For the rest of the proof we can assume that on any path r of the form described above there are no repetitions.

4.2.2 Unreachable Leaves

Let us summarize what we have from the previous section. We created a pair of triples: (p, U_p, σ_p) and (q, U_q, σ_q) such that:

- (p, q) forms $(k_0 + 1)$-garland:
  - $p = v_0 p_1 v_1 p_2 v_2 p_3 \dots p_{k_0+1} v_{k_0+1}$;
  - $q = v_0 q_1 v_1 q_2 v_2 q_3 \dots q_{k_0+1} v_{k_0+1}$;
- (p, U_p, σ_p) and (q, U_q, σ_q) are β'-good;
- there are no repetitions on any path in the garland (p, q).

We use Lemma 18 for (p, U_p, σ_p) and (q, U_q, σ_q) and get an assignment κ. Let us fix an assignment γ that is an instance of $\sigma_p \uplus \sigma_q$ consistent with:

- τ_p;
- values in σ_q that do not contradict τ_p

and denote $\zeta := \gamma \cup \kappa$. Note that, by construction:

- $|\zeta| \le \nu_{k-1} \Delta r$;
- $|\zeta|$ is $(r, 0.6, \beta')$-locally consistent.

In this section we describe a set of clauses that should be unreachable from the vertex v_{k_0+1}. Note that on each segment of a garland $(v_i p_i v_{i+1}, v_i q_i v_{i+1})$ we query at least one variable in both assignments τ_p and τ_q and get the different values. Denote any variable that satisfies this property by x_i.

We remind that $\varphi := \mathsf{Flow}_G^{\eta^{k_0+1}}$. Let $\mathfrak{D}, \mathfrak{C}$ be the subsets of clauses:

$$\mathfrak{D} := \{D \in \mathsf{Flow}_G \mid \text{for every } e \text{ that corresponds to some } x_i : e \in \eta^{k_0+1}(D)\}.$$

and

$$\mathfrak{C} := \{C \in \varphi \mid C \text{ is obtained from some } D \in \mathfrak{D} \text{ by the amplification trick}\}.$$

For the sake of contradiction suppose that there is a path s from v_{k_0+1} such that:

- s is a consistent path and τ_s is consistent with ζ and hence ps is also consistent;
- s ends in a clause $C \in \mathfrak{C}$.

Consider a family of paths $r_i := v_0 p_1 v_1 p_2 v_2 p_3 \dots p_{i-1} v_{i-1} q_i v_i p_{i+1} q_{i+1} p_{i+2} \dots p_{k_0+1} v_{k_0+1}$, where $i \in [k_0 + 1]$. All paths r_i are consistent since there are no repetitions in the garland (p, q). Hence if r_i is inconsistent with s then on s we requery some variable x'_i from the segment $v_{i-1} q_i v_i$ and get an inconsistent value.

By construction, τ_s is consistent with ζ, and $\zeta := \gamma \cup \kappa$, where γ is an instance of $\sigma_p \uplus \sigma_q$. If x'_i appeared in $v_{i-1} q_i v_i$, but not in $v_{i-1} p_i v_i$ (note that it cannot appear in any other segment of the garland, since there are no repetitions on the garland), then $(\sigma_p \uplus \sigma_q)(x'_i) \in \{\sigma_q(x'_i), ?\}$ and $\tau_p(x'_i) = *$ thus $\gamma(x'_i) = \sigma_q(x'_i)$ by the choice of γ. It follows that $\zeta(x'_i) = \sigma_q(x'_i)$ as well, and since τ_s is consistent with ζ, we cannot obtain an inconsistent with τ_{q_i} value for x'_i while requerying it. Hence x'_i had appeared in $v_{i-1} p_i v_i$ as well, and on s we requeried a variable from $v_{i-1} p_i v_i$ in consistent way. Moreover if all paths from some set $\{r_i\}_{i \in L}$ where $L \subseteq [k_0 + 1]$ are inconsistent with s we requery at least $|L|$ variables from the path p on the path s. Hence at least one of the paths r_{i_0} is consistent with s, where $i_0 \in [k_0 + 1]$ (or on the path ps we requery at least $k_0 + 1$ variables).

▶ Remark 20. This is the only place there we use the property that there are no repetitions on the garland.

Consider two paths ps and $r_{i_0}s$:

- these paths are consistent;
- $\tau_{ps}(x_{i_0}) \neq \tau_{r_{i_0}s}(x_{i_0})$.

These properties imply that clause C is not a legal answer for at least one these paths, and we have a contradiction with the assumption that there is a consistent path from v_{k_0+1} to this clause. That gives us the desired description of leaves that should be unreachable for v_{k_0+1}.

To conclude the proof it remains to show that there should be a path from v_{k_0+1} to at least one leaf marked by a clause $C \in \mathfrak{C}$. We do it in the next section.

4.2.3 Directing the Flow

Let us remind that we deal with $\varphi := \mathsf{Flow}_G^{\eta^{k_0+1}}$. To show that there is a path consistent with ζ from v_{k_0+1} to a leaf with a label $C \in \mathfrak{C}$ we show that $(\varphi \setminus \mathfrak{C})|_\zeta$ is satisfiable and hence there should be an extension of ζ that violates only clauses from $\mathfrak{C}$.

▶ Remark 21. If we do not care about assignment ζ, the statement is trivial, since φ is so-called minimally unsatisfiable formula (that becomes satisfiable after removing any clause). But ζ transforms our formula to "heavily unsatisfiable" formula, since ζ 0.6-satisfies a lot of vertices (that was the crucial property that we used to create a garland).

Note that by construction of η^{k_0+1} for each $v \in V$ there exists a clause $D \in \mathfrak{D}$ that had originated from the constraint for v. For each v, we pick any such clause and denote it by D_v. We divide the rest of the proof into two parts.

1. "Local part". We find a carefully chosen large enough set of vertices $U \in V$ and an assignment $\tau \supseteq \zeta$ such that there is a set $V_\tau \supseteq (U \cup V_\zeta)$:
   - $(V \setminus V_\tau, E \setminus \operatorname{supp}(\tau))$ satisfies (r, β')-expansion property;
   - for all $v \in U$ the assignment τ violates D_v and hence τ assigns all edges incident to v;
   - for all $v \in V_\tau \setminus U$ the assignment τ satisfies constraint for v.

   For this part we use the simplified version of technique used for the garland creation.
2. "Global part". By using Max-Flow Min-Cut Theorem we show that τ can be extended to total assignment that satisfies constraints for vertices whose constraints are neither satisfied nor falsified by τ yet.

Since we satisfy all the constraints of $(\mathsf{Flow}_G \setminus \mathfrak{D})_\zeta$ this assignment also satisfies all constraints in $(\varphi \setminus \mathfrak{C})|_\zeta$ by the construction of the formula φ (clauses of φ are the weakened versions of the clauses Flow_G).

Before we proceed with the proof let us define the "overflow".

▶ **Definition 22.** *The* ***overflow*** *introduced by a locally consistent assignment σ is:*

$$\mathsf{of}_\sigma := 1 + \sum_{v \in V_\sigma} \left(\sum_{e \in E:\mathrm{st}(e)=v} x_e - \sum_{e \in E:\mathrm{en}(e)=v} x_e - c(v) \right).$$

Note that $\mathsf{of}_\zeta \leq |\zeta| + 1 \leq \nu_{k-1}\Delta r + 1$.

4.2.3.1 Local part

We start with the local part of the proof. In the beginning of our construction $U_0 := \emptyset$, $\tau_0 := \zeta$, $V_{\tau_0} := V_\zeta$ and $i := 0$.

We repeat the following process while $\mathsf{of}_{\tau_i} > 0$.

1. Choose a vertex u_i that is untouched by τ_i.
2. Let ρ_{u_i} be an assignment to edges that are incident to u_i such that D_{u_i} is unsatisfied by ρ_{u_i}.
3. $\tau' := \tau_i \cup \rho_{u_i}$. Since u_i is untouched by τ_i there is no intersection between ρ_{u_i} and τ_i.
4. Let $H_i \subseteq V \setminus V_{\tau_i}$ be the maximal set of vertices that satisfies:
   - $|H_i| \le r$;
   - $|E(H_i, \overline{H_i} \setminus \{u_i\}) \setminus \mathrm{supp}(\tau_i)| \le \beta' \Delta |H_i|$.

   Let κ_i be an assignment on variables that correspond to edges in the set $E(H) \setminus \mathrm{supp}(\tau')$ such that for all $v \in H_i$:
   $$\sum_{e \in E:\mathrm{st}(e)=v} (\tau' \cup \kappa_i)(x_e) - \sum_{e \in E:\mathrm{en}(e)=v} (\tau' \cup \kappa_i)(x_e) = c(v).$$
5. $U_{i+1} := U_i \cup \{u_i\}$, $\tau_{i+1} := \tau' \cup \kappa_i$ and $V_{\tau_{i+1}} := V_{\tau_i} \cup H_i \cup \{u_i\}$.
6. $i := i + 1$.

Let ℓ be a number of iterations in this process. Let $U := U_\ell$ and $\tau := \tau_\ell$.

At first we give an upper bound on ℓ. Since for all i an assignment κ_i exactly satisfies vertices in H, inclusion of H into V_τ does not change the overflow. Assignment ρ_{u_i} violates $D_{u_i} \in \mathfrak{D}$ and by definition of η^{k_0+1}:
$$-\frac{\Delta}{4} - 1 \le \sum_{e \in E:\mathrm{st}(e)=u_i} \rho_{u_i}(x_e) - \sum_{e \in E:\mathrm{en}(e)=u_i} \rho_{u_i}(x_e) \le -\frac{\Delta}{4}.$$

Hence on each iteration $\mathsf{of}_{\tau_{i+1}} \le \mathsf{of}_{\tau_i} - \frac{\Delta}{4}$ and $|U| \le \frac{4|\zeta|}{\Delta}$ and $-\frac{\Delta}{4} - 1 \le \mathsf{of}_\tau \le 0$.

▶ **Lemma 23.** *For all $i \le \ell$:*
- *κ_i exists;*
- *$|V_{\tau_i}| \le \frac{1}{(\beta-\beta')\Delta}(\mathrm{supp}(\zeta) + \Delta|U_i|)$ and hence $|\tau_i| \le \frac{2}{(\beta-\beta')}(|\mathrm{supp}(\zeta)| + \Delta|U_i|)$;*
- *$(V \setminus V_{\tau_i}, E \setminus \mathrm{supp}(\tau_i))$ satisfies (r, β')-expansion property.*

Proof. This Lemma may be considered as simplified version of Lemma 18. For the proof see Appendix A. ◀

To conclude the construction note that $\tau_i \le \frac{10}{4}\nu_{k-2}\Delta r \le \frac{\Delta}{4} r$ for all $i \le \ell$ and we always can find the vertex untouched by τ_i.

▶ Remark 24. This is the only place where we use that $r \le \frac{n}{\Delta}$.

4.2.3.2 Global part

Let $B := V_\tau \setminus V_\zeta$. For the vertex $v \in V$ the **overflow of v** is defined in the following way:
$$\mathsf{of}(v) := -\sum_{\substack{e \in \mathrm{supp}(\tau) \\ \mathrm{st}(e)=u}} \tau(x_e) + \sum_{\substack{e \in \mathrm{supp}(\tau) \\ \mathrm{en}(e)=u}} \tau(x_e) + c(v).$$

We want to create an auxiliary graph. Let $F^+ := \{v \in V \setminus V_\tau \mid \mathsf{of}(v) > 0\}$ and $F^- := \{v \in V \setminus V_\tau \mid \mathsf{of}(v) < 0\}$. See Fig. 2.

We define a graph $G' := (V', E')$ on vertices $V' := (V \setminus V_\tau) \cup \{s\} \cup \{t\}$, where s is a source and t is a sink. Edges E' include four groups:

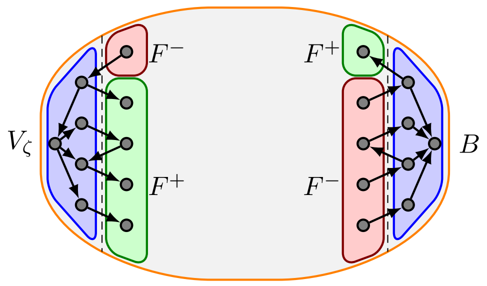

Figure 2 Set after assignment.

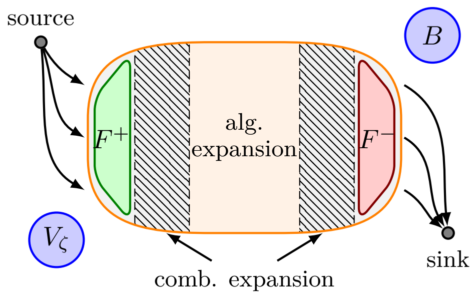

Figure 3 Graph G' with cuts.

- $E \setminus \mathrm{supp}(\tau)$;
- we connect s with all $v \in F^+$ by $\mathsf{of}(v)$ number of edges;
- we connect t with all $v \in F^-$ by $-\mathsf{of}(v)$ number of edges;
- if $\mathsf{of}_\tau < 0$ we choose an arbitrary set of vertices $S \in V \setminus V_\tau$ of size $|\mathsf{of}_\tau|$ and connect all $v \in S$ with s by one more edge.

See Fig. 3.

▶ Remark 25. **1.** $\mathsf{deg}(s) = \mathsf{deg}(t)$;
2. If $A \subseteq V'$ then $E(\{s\}, A) \leq \frac{\Delta}{4} + 1 + \sum_{v \in A} \mathsf{of}(v)$ and $E(\{t\}, A) = -\sum_{v \in A} \mathsf{of}(v)$.

Proof. The first property follows from the construction of τ and the second one follows from definition of G'. ◀

Let $f := \mathsf{deg}(s)$. To conclude the proof we want to show that there is an s-t flow in G' of size f (assuming that capacity of each edge is 1) and that if this flow exists, then we have an extension of τ that satisfies $\mathsf{Flow}_G \setminus \mathfrak{D}$. As we mention above together these facts imply that $(\mathsf{Flow}_G \setminus \mathfrak{D})|_\tau$ is satisfiable hence $(\mathsf{Flow}_G \setminus \mathfrak{D})|_\varsigma$ is satisfiable and $(\varphi \setminus \mathfrak{C})|_\varsigma$ is also satisfiable hence there is a path from v_{k_0+1} to a leaf marked by some $C \in \mathfrak{C}$ which is a contradiction with an existence of a garland and an assumption about size of the branching program.

We start with the second part. Suppose that we have a flow of size f. Fix the flow that achieves this value. We define a total proper assignment $\sigma \supseteq \tau$ in the natural way. Consider an edge $e \in E' \cup E$ and $a = (u, v), a' = (v, u)$ its directed copies. If there is a flow on the edge e:

- from u to v then $x_a = 1$ and $x_{a'} = 0$;
- from v to u then $x_a = 0$ and $x_{a'} = 1$.

otherwise we set $x_a = 0$ and $x_{a'} = 0$.

Note that $f = \mathsf{deg}(s)$ hence we use all edges that connect s with other vertices to push the flow. That implies for all $v \in V \setminus V_\tau$:

$$\sum_{\substack{e \in \mathrm{supp}(\sigma) \setminus \mathrm{supp}(\tau) \\ \mathrm{st}(e) = v}} \sigma(x_e) + \sum_{\substack{e \in \mathrm{supp}(\sigma) \setminus \mathrm{supp}(\tau) \\ \mathrm{en}(e) = v}} \sigma(x_e) = |E(s, v)| = \mathsf{of}(v)$$

and hence

$$\sum_{e \in E : \mathrm{st}(e) = v} \sigma(x_e) + \sum_{e \in E : \mathrm{en}(e) = v} \sigma(x_e) = c(v).$$

and constraints for all vertices in $V \setminus V_\tau$ are satisfied, but τ itself satisfied all constraints in $\mathsf{Flow}_G \setminus \mathfrak{D}$ that correspond to vertices in V_τ. Altogether it says that σ satisfies all constraints in $\mathsf{Flow}_G \setminus \mathfrak{D}$ as desired.

It remains to show that we have an s-t flow of size f in G'. To do it we use the Max-Flow Min-Cut Theorem and show that minimal s-t cut has size f. Consider such a cut (S, T), where S, T are disjoint subsets of V' such that $s \in S$ and $t \in T$. We consider two cases:

- either S or T is small enough, then we use the (r, β')-expansion property that we have after removing $\operatorname{supp}(\tau)$ and V_τ from G;
- S and T are large enough, then we use the Mixing Lemma to show that even removing $\operatorname{supp}(\tau)$ from G cannot destroy balanced cuts.

see Fig. 3.

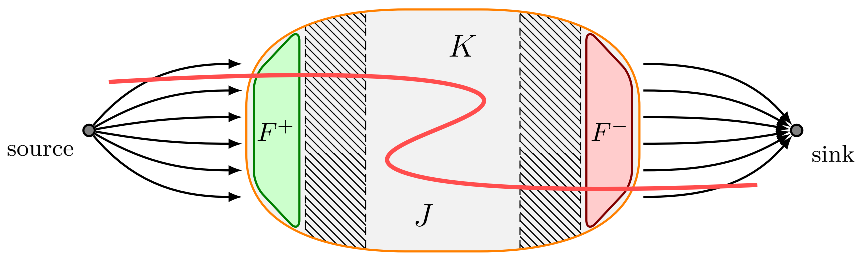

Figure 4 Graph s-t cut.

Consider an arbitrary s-t cut $S \cup T$. Let $J \coloneqq S \setminus \{s\}$ and $K \coloneqq T \setminus \{t\}$ (see Fig. 4). Consider the following cases.

1. If $J = \emptyset$ or $K = \emptyset$ then size of (S, T) cut equals $\mathsf{deg}(s)$ or $\mathsf{deg}(t)$ respectively and we are done.
2. $0 < |J| \le r$ or $0 < |K| \le r$. Wlog assume that $|J| \le r$. Note that:
$$E_{G'}(S, T) = E_{G'}(\{s\}, K) + E_{G'}(\{t\}, J) + E_{G'}(J, K).$$
$E_{G'}(\{s\}, K) = \sum_{v \in F^+ \cap K} \mathsf{of}(v)$, so by Remark 25 to give a lower bound on the size of cut it is enough to show that $E_{G'}(J, K) \ge \frac{\Delta}{4} + 1 + \sum_{v \in F^+ \cap J} \mathsf{of}(v)$. But $(V \setminus V_\tau, E \setminus \operatorname{supp}(\tau))$ satisfies (r, β')-expansion property. Hence
   - for all $v \in V \setminus V_\tau$: $|\mathsf{of}(v)| \le 0.1 \cdot \Delta$;
   - $|E_{G'}(J, K)| \ge 0.9 \cdot \Delta |J|$,

   that implies that $|E_{G'}(J, K)| - \frac{\Delta}{4} - 1 \ge 2 \sum_{v \in F^+ \cap J} \mathsf{of}(v)$.
3. $|J| > r, |K| > r$. Wlog assume that $|J| \le |K|$. By Mixing Lemma:
$$|E_G(J, \overline{J})| = \Delta |J| - E_G(J, J) \ge \Delta |J| - \frac{\Delta}{n} |J|^2 - \alpha \Delta |J| \ge \Delta |J| - |J| - \alpha \Delta |J| \ge 0.9 \cdot \Delta r,$$
and
$$|E_{G'}(J, K)| \ge |E_G(J, \overline{J})| - |\operatorname{supp}(\tau)| \ge 0.6 \cdot \Delta r.$$
On the other hand:
$$f = \sum_{v \in F^+} \mathsf{of}(v) =$$
$$\sum_{v \in F^+} \left(- \sum_{\substack{e \in \operatorname{supp}(\tau) \\ \operatorname{st}(e) = u}} \tau(x_e) + \sum_{\substack{e \in \operatorname{supp}(\tau) \\ \operatorname{en}(e) = u}} \tau(x_e) + c(v) \right) \le$$
$$|\operatorname{supp}(\tau)| \le \frac{\Delta}{4} r.$$

Hence in all cases (S, T) has size at least f which by Max-Flow Min-Cut Theorem implies the existence of flow in G' of size at least f. That as mentioned above implies the desired lower bound on the size of branching program.

5 Cook–Reckhow Proof Systems

In this section we illustrate that syntactic $(1, +k)$-BP give us a proof system in terms of Cook–Reckhow. This result is a generalization of the same result for formulas of bounded width [13]. We start with the most general definition of a proof system for a language of unsatisfiable formulas.

▶ **Definition 26** (Cook, Reckhow [7]). *A* ***proof system*** *is a polynomial-time algorithm $\Pi(\varphi, w)$ that satisfies two properties:*

correctness: if there is some $w \in \{0,1\}^*$ such that $\Pi(\varphi, w) = 1$ then φ is unsatisfiable;

soundness: if φ is an unsatisfiable boolean formula then there is a string $w \in \{0,1\}^*$ such that $\Pi(\varphi, w) = 1$.

We say that w is a ***witness of unsatisfiability*** *of φ.*

$(1, +k)$-BP can be used to define a natural proof system. We assume that the witness of unsatisfiability of a CNF formula φ is a description of a $(1, +k)$-BP that solves Search_φ problem; denote it by $(1, +k)$-BP-PS. This definition is equivalent to the definition of $(1, +k)$-BP-PS from [13] but we erase some technicalities.

For our purpose we need to show that there is a polynomial-time algorithm that checks whether a given description is a $(1, +k)$-BP and that it solves Search_φ.

▶ **Lemma 27.** *There is an algorithm that for given syntactic $(1, +k)$-BP of size s and boolean CNF formula φ with m clauses and n variables checks whether this program solves Search_φ in time $\mathcal{O}\left[\left(\frac{4en}{k}\right)^k sn^2m\right]$.*

We defer the proof of this Lemma to section 5.1.

▶ **Theorem 5.1** (3). *A syntactic $(1, +k)$-BP-PS is a proof system in terms of Cook–Reckhow definition for any constant $k \in \mathbb{N}$.*

Proof. Given a description of a branching program B we can use an algorithm from Theorem 7 to check whether it is $(1, +k)$-BP. After that we can use an algorithm from Lemma 27 to check whether this program solves Search_φ problem. If k is an absolute constant both algorithms work in time $\mathsf{poly}(|\varphi|, |B|)$. ◀

5.1 Proof of Lemma 27

Fix some syntactic $(1, +k)$-BP B and some CNF formula $\varphi := \bigvee_{i=1}^{m} C_i$. Leaves of B are marked by clauses of φ. We construct an auxiliary branching programs B_i that are obtained by replacing the labels C_i of sinks by 1 and other labels by 0.

The clause C is a solution of the Search_φ problem for an assignment z iff $C(z) = 0$. Hence B makes a mistake on the assignment z iff the path that corresponds to z ends in sink marked by C and $C(z) = 1$. But it means that B makes a mistake iff there is a variable x_i such that an assignment $x_i \leftarrow z_i$ satisfies C and there is a path in B from source to sink labeled by C consistent with an assignment $x_i \leftarrow z_i$.

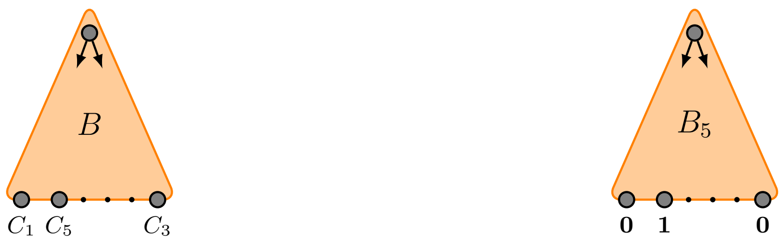

Figure 5 Construction of B_i.

The last observation gives useful criteria of correctness. We have path in B from source to sink that is labeled by C_i consistent with some assignment $x_j \leftarrow a$ iff $B_i|_{x_j \leftarrow a}$ is satisfiable. We are ready to describe an algorithm:

1. enumerate all clauses $C_i \in \varphi$;
2. enumerate variables $x_j \in C_i$ and consider a constant a such that $C_i|_{x_j \leftarrow a} = 1$;
3. check whether $B_i|_{x_j \leftarrow a}$ is satisfiable, if yes return "NO";
4. if B passes all tests then return "B is correct".

The correctness of this algorithm follows from previous observation. And we run satisfiability algorithm at most nm times hence the running time is at most $\mathcal{O}\left[\left(\frac{4en}{k}\right)^k sn^2m\right]$.

6 Open Problems

In conclusion we want to mention some open problems. We start with the obvious ones.

1. Find a formula that is hard for $(1, +k)$-BP where $k := n^{\varepsilon}$.
2. Find a formula that is hard for read-twice branching programs (programs that on any path may read each variable at most twice).

Another problems are more technical, but in our opinion the solution of these problems may lead to new techniques for proving lower bounds.

1. Find a "natural" formula that is hard for $(1, +k)$-BP for any $k > 0$. The main problem with the current bound is that we amplify our formula by an η function. This is an artificial trick that prevents generalization of our main Theorem.
2. More difficult question: can we prove a lower bound on random Δ-CNF formulas? This is a canonical example of the hard formulas. Typically, only the "local" structure is used for proving lower bounds on these formulas, which is one of the important barriers for proving lower bounds on these formulas in $\mathbf{AC}_0$-Frege proof system.

References

1 Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Pseudorandom generators in propositional proof complexity. *SIAM J. Comput.*, 34(1):67–88, 2004. `doi:10.1137/S0097539701389944`.

2 Michael Alekhnovich, Jan Johannsen, Toniann Pitassi, and Alasdair Urquhart. An exponential separation between regular and general resolution. *Theory Comput.*, 3(1):81–102, 2007. `doi:10.4086/toc.2007.v003a005`.

3 Michael Alekhnovich and Alexander A. Razborov. Lower bounds for polynomial calculus: Non-binomial case. *Proceedings of the Steklov Institute of Mathematics*, 242:18–35, 2003. Available at `http://people.cs.uchicago.edu/~razborov/files/misha.pdf`. Preliminary version in *FOCS '01*.

4 Noga Alon and Fan R. K. Chung. Explicit construction of linear sized tolerant networks. *Discret. Math.*, 72(1-3):15–19, 1988. doi:10.1016/0012-365X(88)90189-6.

5 Albert Atserias, Ilario Bonacina, Susanna F. de Rezende, Massimo Lauria, Jakob Nordström, and Alexander A. Razborov. Clique is hard on average for regular resolution. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 866–877. ACM, 2018. doi:10.1145/3188745.3188856.

6 Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow – resolution made simple. *J. ACM*, 48(2):149–169, 2001. doi:10.1145/375827.375835.

7 Stephen Cook and Robert Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, March 1979. URL: https://projecteuclid.org:443/euclid.jsl/1183740343.

8 L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. doi:10.4153/CJM-1956-045-5.

9 Konstantinos Georgiou, Avner Magen, and Madhur Tulsiani. Optimal sherali-adams gaps from pairwise independence. In Irit Dinur, Klaus Jansen, Joseph Naor, and José Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 125–139, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

10 Stasys Jukna. Expanders and time-restricted branching programs. *Theor. Comput. Sci.*, 409(3):471–476, 2008. doi:10.1016/j.tcs.2008.09.012.

11 Stasys Jukna. *Boolean Function Complexity — Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-24508-4.

12 Stasys Jukna and Alexander A. Razborov. Neither reading few bits twice nor reading illegally helps much. *Discret. Appl. Math.*, 85(3):223–238, 1998. doi:10.1016/S0166-218X(98)00042-0.

13 Alexander Knop. IPS-like Proof Systems Based on Binary Decision Diagrams. *Electron. Colloquium Comput. Complex.*, 24:179, 2017. URL: https://eccc.weizmann.ac.il/report/2017/179.

14 László Lovász, Moni Naor, Ilan Newman, and Avi Wigderson. Search problems in the decision tree model. *SIAM J. Discret. Math.*, 8(1):119–132, 1995. doi:10.1137/S0895480192233867.

15 William Masek. A fast algorithm for the string editing problem and decision graph complexity. *Master Thesis, Massachusetts Institute of Technology*, 1976.

16 E. I. Nechiporuk. On a boolean function. *Dokl. Akad. Nauk SSSR*, 169:765–766, 1966.

17 Toniann Pitassi and Ran Raz. Regular resolution lower bounds for the weak pigeonhole principle. *Comb.*, 24(3):503–524, 2004. doi:10.1007/s00493-004-0030-y.

18 Ran Raz. Resolution lower bounds for the weak pigeonhole principle. *J. ACM*, 51(2):115–138, 2004. doi:10.1145/972639.972640.

19 Petr Savický. A probabilistic nonequivalence test for syntactic (1,+k)-branching programs. *Electron. Colloquium Comput. Complex.*, 5(51), 1998. URL: http://eccc.hpi-web.de/eccc-reports/1998/TR98-051/index.html.

20 Petr Savický and Stanislav Žák. A lower bound on branching programs reading some bits twice. *Theor. Comput. Sci.*, 172(1-2):293–301, 1997. doi:10.1016/S0304-3975(96)00183-1.

21 Detlef Sieling. New lower bounds and hierarchy results for restricted branching programs. *J. Comput. Syst. Sci.*, 53(1):79–87, 1996. doi:10.1006/jcss.1996.0050.

22 Detlef Sieling and Ingo Wegener. New lower bounds and hierarchy results for restricted branching programs. In Ernst W. Mayr, Gunther Schmidt, and Gottfried Tinhofer, editors, *Graph-Theoretic Concepts in Computer Science, 20th International Workshop, WG '94, Herrsching, Germany, June 16-18, 1994, Proceedings*, volume 903 of *Lecture Notes in Computer Science*, pages 359–370. Springer, 1994. doi:10.1007/3-540-59071-4_61.

23 Dmitry Sokolov. (semi)algebraic proofs over ± 1 variables. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 78–90. ACM, 2020. doi:10.1145/3357713.3384288.

24 Ingo Wegener. On the complexity of branching programs and decision trees for clique functions. *J. ACM*, 35(2):461–471, 1988. `doi:10.1145/42282.46161`.

25 Ingo Wegener. *Branching Programs and Binary Decision Diagrams*. SIAM, 2000. URL: `http://ls2-www.cs.uni-dortmund.de/monographs/bdd/`.

26 Stanislav Žák. An exponential lower bound for real-time branching programs. *Information and Control*, 71(1):87–94, 1986. `doi:10.1016/S0019-9958(86)80018-3`.

A Missed Lemmas

A.1 Lemma 18

At first we prove an auxiliary Lemma.

▶ **Lemma 28.** *If $G := (V, E)$ satisfies (r, a)-expansion property, $M \subseteq E$, and $S \subseteq V$ of size at most r, such that $|E(S, \overline{S}) \setminus M| \le b\Delta|S|$ then $|S| \le \frac{|M|}{(a-b)\Delta}$.*

Proof. The size of S is at most r, hence:

$$b\Delta|S| \ge |E(S, \overline{S}) \setminus M| \ge a\Delta|S| - |M|.$$

Thus $|S| \le \frac{|M|}{(a-b)\Delta}$. ◀

▶ **Lemma A.1** (18). *Let (p, U_p, σ_p) and (q, U_q, σ_q) be 0.9-good triples. Then there is an assignment κ such that:*

- *for any γ that is an instance of $\sigma_p \uplus \sigma_q$ an assignment $\gamma \cup \kappa$ is a β'-mlce on $\mathrm{supp}(\sigma_p) \cup \mathrm{supp}(\sigma_q)$;*
- *$|\mathrm{supp}(\gamma \cup \kappa)| \le \nu_{k-1}\Delta r$.*

Moreover if $p = q$ then triple $(p, U_p, \sigma_p \cup \kappa)$ is β'-good.

Proof. Let $S := V_{\sigma_p} \cup V_{\sigma_q}$, $E_\sigma := \mathrm{supp}(\sigma_p) \cup \mathrm{supp}(\sigma_q)$ and $B \subseteq V \setminus S$ be the maximal set of vertices that satisfies:

- $|B| \le r$;
- $|E(B, \overline{B}) \setminus E_\sigma| \le \beta'\Delta|B|$.

At first we give an upper bound on the size of set B.

Partial assignment σ_p is 0.9-mlce on $M \cup U_p$. $\beta'\Delta|V_{\sigma_p}| \ge |E(V_{\sigma_p}, \overline{V}_{\sigma_p}) \setminus (M \cup U_p)|$ and by Lemma 28

$$|V_{\sigma_p}| \le \frac{|M \cup U_p|}{(\beta - \beta')\Delta} \le 2\frac{\nu_k}{(\beta - \beta')} r \le \frac{1}{2}\nu_{k-1} r.$$

By analogy the same holds for V_{σ_q}.

The equality $E(B, \overline{B}) \cap E_\sigma = E(B, S) \cup (E(B) \cap |M \cup U_p \cup U_q|)$ together with $|E(B, \overline{B}) \setminus E_\sigma| \le \beta'\Delta|B|$ implies:

$$(1 - \beta')\Delta|B| - |M \cup U_p \cup U_q| \le |E(B, S)|$$

By Mixing Lemma:

$$|E(B, S)| \le \frac{\Delta}{n}|B||S| + \alpha\Delta\sqrt{|S||B|}.$$

For the sake of contradiction assume that $|B| \ge |S|$ thus:

$$|E(B, S)| \le \frac{\Delta}{n}|B||S| + \alpha\Delta|B|.$$

Altogether:

$$(1-\beta')\Delta|B| \le \frac{\Delta r}{n}\nu_{k-1}|B| + \alpha\Delta|B| + 3\nu_k\Delta r \le 2\alpha\Delta|B|,$$

that contradicts the choice of α and β', hence $|B| \le |S| \le \nu_{k-1}r$.

At first we show that $(V \setminus (S \cup B), E \setminus (E_\sigma \cup E(B)))$ satisfies (r, β')-expansion property. By contradiction, suppose that there is a set $B' \subseteq V \setminus (S \cup B)$ of size at most r such that $|E(B', \overline{B'}) \setminus (E_\sigma \cup E(B))| < \beta'\Delta|B'|$.

Again by Lemma 28 we conclude that:

$$|B'| \le \frac{|M \cup U_p \cup U_q| + \Delta|S \cup B|}{(\beta - \beta')\Delta} \le \nu_{k-1}r + \frac{1}{2}r \le \frac{3}{4}r.$$

But it implies that $|B \cup B'| \le r$, moreover:

$$\begin{aligned} &|E(B \cup B', \overline{B \cup B'}) \setminus E_\sigma| \le \\ &\qquad \beta'\Delta|B| + \beta'\Delta|B'| = \\ &\qquad\quad \beta'\Delta|B \cup B'|. && B \text{ and } B' \text{ are disjoint} \end{aligned}$$

That contradicts the choice of B.

Now we find a proper assignment κ on the $E(B) \setminus E_\sigma$ such that for all $v \in B$:

$$\sum_{e \in E:\mathrm{st}(e)=v} x_e \ge 0.8 \cdot \Delta.$$

Since σ_p is an $(r, 0.6, 0.9)$-locally consistent assignment, then $(V \setminus V_{\sigma_p}, E \setminus \mathrm{supp}(\sigma_p))$ satisfies $(r, 0.9)$-expansion property. By analogy we have the same property for σ_q that implies: $(V \setminus S, E \setminus E_\sigma)$ satisfies $(r, 0.8)$-expansion property. Indeed, consider a set $C \subseteq V \setminus S$ of size at most r:

$$\begin{aligned} |E(C, \overline{C}) \setminus E_\sigma| &= |E(C, \overline{C})| - |E(C, \overline{C}) \cap E_\sigma| \\ &\ge |E(C, \overline{C})| - |E(C, \overline{C}) \cap \mathrm{supp}(\sigma_p)| - |E(C, \overline{C}) \cap \mathrm{supp}(\sigma_q)| \\ &= |E(C, \overline{C}) \setminus \mathrm{supp}(\sigma_p)| - 0.1 \cdot \Delta|C| \\ &\ge 0.8 \cdot \Delta|C|. \end{aligned}$$

By Proposition 12 there is an enumeration of vertices in B: $v_1, v_2, \ldots, v_{|B|} \in B$ and a sequence $R_1, \ldots, R_{|B|} \subseteq (E(B) \setminus E_\sigma)$ such that:
- $R_i = E(\{v_i\}, V \setminus \{v_1, v_2, \ldots, v_i\}) \setminus E_\sigma$;
- $|R_i| \ge 0.8\Delta$.

We define κ in the following way:
- for an $e \in R_i$ we assign corresponding variables to direct the flow outside of the vertex v_i (i.e. if e' is a directed copy of e that goes outside of v_i we set $x_{e'}$ to 1 and set the dual edge to 0);
- for all loops inside the set B we assign corresponding variables to 0.

Let γ be an instance of $\sigma_p \uplus \sigma_q$, $\zeta := \gamma \cup \kappa$ and $V_\zeta := S \cup B$. We have already shown that the graph $(V \setminus V_\zeta, E \setminus \mathrm{supp}(\zeta))$ satisfies (r, β')-expansion property. We want to show that vertices in V_ζ are 0.6-satisfied by ζ. Consider four cases.

1. $v \in V_\rho$. Both assignments σ_p and σ_q extend an assignment ρ hence γ agreed with both assignments on edges incident to V_ρ. Thus γ 0.6-satisfies v.

2. $v \in V_{\sigma_p} \setminus V_\rho$. Let E_v be a set of edges that are incident to v. At least $0.8 \cdot \Delta$ of those edges carry outgoing flow from v in σ_p. Denote those edges as E_{σ_p}.
   If $v \notin V_{\sigma_q}$ then σ_q may assign at most $0.1 \cdot \Delta$ edges in E_v. That means that in γ at least $0.7 \cdot \Delta$ edges from E_{σ_p} still carry outgoing flow from v.
   If $v \in V_{\sigma_q}$ then σ_p and σ_q both 0.8-satisfy v. Let $E_{\sigma_q} \subseteq E_v$ be the set of edges that carry outgoing flow from v in σ_q. Then $E_{\sigma_p} \cap E_{\sigma_q} \geq 0.6 \cdot \Delta$, and all those edges carry outgoing flow from v in γ.
   Note that if $\sigma_p = \sigma_q$, then we 0.8-satisfy v.
3. $v \in V_{\sigma_p} \setminus V_\rho$. By analogy with the previous case.
4. $v \in B$. We direct the flow on at least $0.8 \cdot \Delta$ edges from E_v outside of v hence κ 0.8-satisfies v.

By construction $V_\zeta := V_{\sigma_p} \cup V_{\sigma_q} \cup B$ hence $|V_\zeta| \leq \nu_{k-1} r$ and $|\operatorname{supp}(\zeta)| \leq \nu_{k-1} r$. In order to check that ζ is β'-mlce note that:

$$|E(B, \overline{B}) \setminus E_\sigma| \leq \beta' \Delta |B| \leq \beta' \Delta |V_\zeta|,$$

but

$$|E(B, \overline{B}) \setminus E_\sigma| = |E(V_\zeta, \overline{V_\zeta}) \setminus E_\sigma|$$

since σ_p and σ_q together assign all edges that are incident to $V_{\sigma_p} \cup V_{\sigma_q}$. Thus:

$$|E(V_\zeta, \overline{V_\zeta}) \setminus E_\sigma| \leq \beta' \Delta |V_\zeta|$$

that concludes the proof.

In case of $(p, U_p, \sigma_p) = (q, U_q, \sigma_q)$ it remains to show that ζ is β'-mlce on $M \cup U_p$. Again we note that:

$$|E(B, \overline{B}) \setminus E_\sigma| \leq \beta' \Delta |B|,$$

and also

$$|E(V_{\sigma_p}, \overline{V}_{\sigma_p}) \setminus E_\sigma| \leq \beta' \Delta |V_{\sigma_p}|,$$

hence

$$|E(V_{\sigma_p} \cup B, \overline{V_{\sigma_p} \cup B}) \setminus E_\sigma| \leq \beta' \Delta (|V_{\sigma_p}| + |B|) \leq \beta' \Delta |V_{\sigma_p} \cup B|,$$

where the last inequality holds since B and V_{σ_p} are disjoint, that concludes the proof. ◀

A.2 Lemma 23

▶ **Lemma A.2** (23). *For all $i \leq \ell$:*

- *κ_i exists;*
- *$|V_{\tau_i}| \leq \frac{1}{(\beta-\beta')\Delta}(\operatorname{supp}(\zeta) + \Delta|U_i|)$ and hence $|\tau_i| \leq \frac{2}{(\beta-\beta')}(|\operatorname{supp}(\zeta)| + \Delta|U_i|)$;*
- *$(V \setminus V_{\tau_i}, E \setminus \operatorname{supp}(\tau_i))$ satisfies (r, β')-expansion property.*

Proof. We show by induction on i that:

- $(V \setminus V_{\tau_i}, E \setminus \operatorname{supp}(\tau_i))$ satisfies (r, β')-expansion property;
- $|E(V_{\tau_i}, \overline{V}_{\tau_i}) \setminus (\operatorname{supp}(\zeta) \cup E(U_i))| < \beta' \Delta |V_{\tau_i}|$;
- $|V_{\tau_i}| \leq \frac{1}{(\beta-\beta')\Delta}(\operatorname{supp}(\zeta) + \Delta|U_i|)$ and hence $|\tau_i| \leq \frac{2}{(\beta-\beta')}(|\operatorname{supp}(\zeta)| + \Delta|U_i|)$.

Assignment τ_0 is ζ and ζ is $(r, 0.6, \beta')$-locally consistent, in particular, $(V \setminus V_\zeta, E \setminus \operatorname{supp}(\zeta))$ satisfies (r, β')-expansion property and $E(V_\zeta, \overline{V}_\zeta) \setminus \operatorname{supp}(\zeta)) = \emptyset$.

By definition of H_i:

$$\beta'\Delta|H_i| > |E(H_i, \overline{H}_i \setminus \{u_i\}) \setminus \operatorname{supp}(\tau_i)| \geq |E(H_i, \overline{H}_i) \setminus (\operatorname{supp}(\tau_i) \cup E(H_i, \{u_i\}))|$$

and by Lemma 28

$$|H_i| \leq \frac{|\operatorname{supp}(\tau_i) \cup E(H_i, u_i)|}{(\beta - \beta')\Delta} \leq \frac{|\operatorname{supp}(\tau_i) \cup E(H_i, u_i)|}{(\beta - \beta')\Delta} \leq \frac{1}{2}\nu_{k-2}(r+1).$$

Hence $|H_i \cup V_{\tau_i} \cup \{u_i\}| \leq r$ that together with:

$$\begin{aligned} &|E(H_i \cup V_{\tau_i} \cup \{u_i\}, \overline{H_i \cup V_{\tau_i} \cup \{u_i\}}) \setminus (\operatorname{supp}(\zeta) \cup E(U_{i+1}))| \leq \\ &|E(V_{\tau_i}, \overline{H_i \cup V_{\tau_i}}) \setminus (\operatorname{supp}(\zeta) \cup E(U_{i+1}))| + |E(H_i, \overline{H}_i) \setminus (\operatorname{supp}(\zeta) \cup E(U_{i+1}) \cup E(V_{\tau_i})| \leq \\ &\beta'\Delta|V_{\tau_i}| + \beta'\Delta|H_i| \leq \\ &\beta'\Delta|H_i \cup V_{\tau_i}| \leq \\ &\beta'\Delta|H_i \cup V_{\tau_i} \cup \{u_i\}| \end{aligned}$$

implies $|V_{\tau_{i+1}}| = |H_i \cup V_{\tau_i} \cup \{u_i\}| \leq \frac{1}{(\beta-\beta')\Delta}(|\operatorname{supp}(\zeta)| + \Delta|U_{i+1}|)$ by Lemma 28. Also $|\tau_{i+1}| \leq \frac{2}{(\beta-\beta')}(|\operatorname{supp}(\zeta)|+\Delta|U_{i+1}|)$ since by construction τ_{i+1} assigns only edges in $\operatorname{supp}(\zeta) \cup E(U_i \cup V_{\tau_{i+1}})$.

Now we show that a graph $(V \setminus V_{\tau_{i+1}}, E \setminus \operatorname{supp}(\tau_{i+1}))$ satisfies (r, β')-expansion property. For the sake of contradiction assume that there is a set $S \subseteq V \setminus V_{\tau_{i+1}}$ of size at most r such that: $E(S, \overline{S}) \setminus \operatorname{supp}(\tau_{i+1}) \leq \beta'\Delta|B|$.

By Lemma 28 $|S| \leq \frac{|\operatorname{supp}(\tau_{i+1})|}{(\beta-\beta')\Delta} \leq \frac{1}{2}\nu_{k-2}(r+1)$. Hence $|H_i \cup S| \leq r$ that together with:

$$\begin{aligned} &E(H_i \cup S, \overline{H_i \cup S} \setminus \{u_i\}) \setminus \operatorname{supp}(\tau_i)| \leq \\ &E(H_i, \overline{H_i \cup S} \setminus \{u_i\}) \setminus \operatorname{supp}(\tau_i)| + E(S, \overline{H_i \cup S} \setminus \{u_i\}) \setminus \operatorname{supp}(\tau_i)| \leq \\ &\beta'\Delta|H_i| + \beta'\Delta|S| = \\ &\beta'\Delta|H_i \cup S| \end{aligned}$$

contradicts the choice of H_i.

To conclude the proof we have to show the existence of κ_i. Note that $(V \setminus V_{\tau_i}, E \setminus \operatorname{supp}(\tau_i))$ satisfies (r, β')-expansion property. Consider an arbitrary set $B \subseteq V \setminus (V_{\tau_i} \cup \{u_i\})$ of size at most r:

$$|E(B, \overline{B}) \setminus (\operatorname{supp}(\tau_i) \cup E(\{u_i\}))| \geq \beta'\Delta|B| - E(B, \{u\}).$$

By Mixing Lemma:

$$|E(B, \{u\})| \leq \frac{\Delta}{n}|B| + \alpha\Delta\sqrt{B} \leq 0.05 \cdot \Delta|B|,$$

and hence

$$|E(B, \overline{B}) \setminus (\operatorname{supp}(\tau_i) \cup E(\{u_i\}))| \geq 0.9 \cdot \Delta|B|$$

and graph $(V \setminus V_{\tau_i} \setminus \{u_i\}, E \setminus \operatorname{supp}(\tau_i))$ satisfies $(r, 0.9)$-expansion property.

By Proposition 12 there is an enumeration of vertices in H_i: $v_1, v_2, \ldots, v_{|H_i|} \in H_i$ and a sequence $R_1, \ldots, R_{|H_i|} \subseteq E(H_i) \setminus (\operatorname{supp}(\tau_i) \cup E(\{u_i\}))$ such that:

- $R_k = E(\{v_k\}, V \setminus \{v_1, v_2, \dots, v_k\}) \setminus (\mathrm{supp}(\tau_i) \cup E(\{u_i\}))$;
- $|R_i| \geq 0.9 \cdot \Delta$.

We define κ_i for vertices $v_1, \dots, v_{H_i}$ step by step, such that κ_i on $E(v_k)$ satisfies the constraint:

$$\sum_{e \in E:\mathrm{st}(e)=v_k} (\tau' \cup \kappa_i)(x_e) - \sum_{e \in E:\mathrm{en}(e)=v_k} (\tau' \cup \kappa_i)(x_e) = c(v_k).$$

Since we have an access to the $0.9 \cdot \Delta$ edges and others are already assigned, we can always choose the right values (loops are always assigned to zero). ◀

B Garland in the Paths

▶ **Lemma B.1** (19). *There are $(p, U_p, \sigma_p), (q, U_q, \sigma_q) \in \mathcal{S}$ such that (p, q) forms a $(k+1)$-garland.*

Proof. Note that we can describe elements in $\mathcal{P}$ by a sequence of bits of size $s := \nu_k \Delta r$. Each bit of this sequence describes an assignment for an edge e that we choose on "branching step". From the construction it follows that different sequences generate different paths in the branching program and hence different elements of $\mathcal{P}$.

Let $s_k := \lfloor \frac{s}{k+1} \rfloor$. We construct our garland by the iterative algorithm. After i-th iteration we have a set S_i of sequences of size is_k such that any two of the corresponding paths form i-garland and all paths end in the same node. The size of S_i will be at least $\exp\left[s_k - \frac{i}{2k} s_k\right]$ for all $1 \leq i \leq k+1$.

1. For $i = 1$ consider all possible strings of length s_k and paths that correspond to them. The branching program has size at most $2^{\frac{s_k}{2k}}$, hence there exists a node such that at least $2^{\frac{s_k(2k-1)}{2k}}$ paths end there. The set S_1 consists of all corresponding sequences.
2. For the step i, $2 \leq i \leq k+1$, we consider all sequences in S_{i-1}. Let v be the end node of all paths corresponding to sequences in the set. To each sequence $s \in S_{i-1}$ we append a string u_s of s_k bits in such a way that for any pair $r, r' \in S_{i-1}$ paths that corresponds to ru_r and $r'u_{r'}$ differ at some node after v. Since $2^{s_k} \geq |S_{i-1}|$, it is possible to do this. For the resulting sequences, we consider the set of the corresponding paths. The set of paths has size at least $2^{\frac{s_k(2k-i+1)}{2k}}$, and the size of the program is at most $2^{\frac{s_k}{2k}}$. Hence there exists a node such that $2^{\frac{s_k(2k-i)}{2k}}$ paths end there. Let S_i be the set of sequences corresponding to those paths.

After $k+1$ steps we have a set S_{k+1}, $|S_{k+1}| \geq 2$, such that any two sequences in it correspond to a $(k+1)$-garland. ◀