Electronic Colloquium on Computational Complexity, Report No. 214 (2025)

A new characterization of VNP via Colored Determinant

Prasad Chaugule

Indian Institute of Technology Delhi, India
chaugule@cse.iitd.ac.in

Abstract. The main theme of this work is to provide a new characterization of the algebraic complexity class VNP. We achieve this via following the two main steps -

- We define a combinatorial variant of the determinant polynomial which we call the colored determinant and show it to be VNP-complete under p-projections for all fields. This adds a new non-monotone VNP-complete polynomial sequence to the already known list of non-monotone VNP-complete polynomial sequences [1].
- Using the colored determinant polynomial, we show that a new variant of stack branching program, called the conditional stack branching program, characterizes the class VNP.

Keywords: Colored Determinant · VNP· Algebraic Complexity

1 Introduction

The main goal of algebraic complexity is to separate the complexity classes VP and VNP. Therefore, to understand these classes better, it is important to study them also through a lens different from the classical definitions of these classes. One well-studied and successful way to characterize a given algebraic class is to study a polynomial sequence complete for that class. For example, the determinant sequence is known to almost characterize the class VP [8] and the permanent sequence is known to characterize the class VNP [8]. Moreover, there are also variants of the determinant sequence known to characterize the classes VP and VNP [2].

Inspired by the study of Non-Auxiliary Pushdown automata (NAuxPDA), Mengel initiated the study of an algebraic branching program appended with memory and gave new characterizations of VP and VNP. Mengel proved that the models *stack branching program* and *random access branching program* characterize the classes VP and VNP respectively [5]. Recently, this line of work was further investigated by S. Chillara and N. Raja [3]. In [5], a *stack branching program* is defined as an algebraic branching program with a single stack memory. In [3], a *stack branching program* with two or more stacks is investigated and shown to characterize the class VNP; in other words, it is shown that adding at least two stacks to an algebraic branching program is sufficient to blow up its power all up to VNP.

ISSN 1433-8092

Table 1. Characterizations of VP and VNP. ✓represents the work in this paper

Class	Characterization	Ref.
VP	Stack Branching Program with one stack (SBP with one stack)	[5]
VNP	Random Access Branching Program (RABP)	[5]
VNP	Stack Branching Program with two stacks (SBP with two stacks)	[3]
VNP	Queue Branching Program (QBP)	[3]
VNP	Conditional Stack Branching Program with one stack (CSBP with one stack)	✓

In this work, we study the trade-off question between the power given to the model to access the stack elements to decide the stack operation to be executed along a given edge and the number of stacks appended to the branching program. We study a new model of computation called the *conditional stack branching program* (with one stack) and show that it characterizes VNP. Our work shows that even a single stack is sufficient to blow the power of an algebraic branching program all up to VNP, given access to the topmost element of the stack to decide the stack operation associated with an edge. We believe that the importance of studying such a model is twofold: one, to give a new characterization of VNP , and the other to investigate and get meaningful insights in the context of the said trade-off. To achieve our result, we show that a variant of determinant, called the colored determinant $\mathsf{ColDet}_n(X)$ is VNP-complete under p-projections for all fields. Moreover, we show that the colored determinant polynomial, $\mathsf{ColDet}_n(X)$ can be computed by a conditional stack branching program (with one stack) of size polynomially bounded in n and for any sequence (f_n) where f_n can be computed by a conditional stack branching program (with one stack) of size polynomially bounded in n is in VNP. This establishes a new characterization of the class VNP via colored determinant sequence. Our result is stronger than existing ABP-with-memory models because it captures a strictly richer class of computations under comparable resource bounds. In particular, the conditional stack–based framework allows dependencies and structural constraints that cannot be simulated efficiently in prior memory-augmented ABP models, thereby yielding a more expressive and robust characterization. We refer the readers to Table 1 for more details regarding various algebraic branching program with memory models studied in the literature.

2 Preliminaries and related work

2.1 Algebraic complexity classes

In this section, we review the standard definitions of p-bounded polynomial sequences, the computational models studied in algebraic complexity, and the associated complexity classes, as well as the notions of hardness and completeness relevant to our results. We refer the readers to [7] for more details about algebraic complexity.

Definition 1 (p-bounded sequence). *A polynomial sequence (f_n) is called a p-bounded sequence if the number of variables and the degree of f_n are both polynomially bounded in n.*

Definition 2 (Arithmetic circuit and formula). *An arithmetic circuit is a directed acyclic graph (DAG) in which the vertices whose in-degree is 0 are called the input vertices. There is a unique out-degree 0 vertex called the output vertex, and all other vertices are called the internal vertices. The internal vertices are labelled with either $+$ or $\times$. An input gate is labelled with a variable from a set of variables X or a constant from $\mathbb{F}$. We define the polynomial computed by a circuit inductively. An input vertex labelled x_i (or $c \in \mathbb{F}$) is said to compute the polynomial x_i (or c, resp.). Let g be a vertex with inputs g_1, g_2. Let $p_1(X), p_2(X)$ be the polynomials computed by g_1, g_2 respectively. If g is labelled with $\times$ (or $+$) then the polynomial computed by g is simply $p_1(X) \times p_2(X)$ (resp. $p_1(X) + p_2(X)$). The polynomial computed by the circuit is the polynomial computed by the output gate. If the out-degree of every vertex in the circuit is one then such a circuit is called a formula.*

Definition 3 (Algebraic Branching Program (ABP) and layered ABP). *An algebraic branching program (ABP) is a directed acyclic graph $G = (V, E)$ with two special designated nodes, called the source node s and the sink node t. The edges are labelled with variables or field constants. For any s to t path ρ, we use f_ρ to denote the product of the variables or constants labelling the edges in ρ. The polynomial computed by an ABP is $\sum_\rho f_\rho$, where ρ is an s to t path. A layered algebraic branching program is the same as the algebraic branching program except that it is a layered, directed acyclic graph DAG with edges connecting vertices only between adjacent layers.*

Definition 4 (VF, VBP and VP). *A p-bounded polynomial sequence (f_n) is in VP (or VF or VBP, respectively), iff f_n can be computed by an arithmetic circuit (or an arithmetic formula or an algebraic branching program, respectively) of size polynomially bounded in n.*

Definition 5 (The complexity class VNP (or VNP_e, respectively)). *A polynomial sequence (f_n) is said to be in VNP (or, VNP_e, respectively) if there is a polynomial sequence (g_n) in VP (or, (g_n) in VF respectively) such that*

$$f_n(X) = \sum_{y_1 \in \{0,1\}, \ldots, y_{m(n)} \in \{0,1\}} g_n(X, Y)$$

Note that $m : \mathbb{N} \to \mathbb{N}$ is polynomially bounded in n.

It is known that $\mathsf{VF} \subseteq \mathsf{VBP} \subseteq \mathsf{VP} \subseteq \mathsf{VNP}$ [6]. But none of the containments are known to be strict. However, it is known that $\mathsf{VNP} = \mathsf{VNP}_e$ [9].

Lemma 1 (Valiant's criterion). *Let $\phi : \{0,1\}^* \to \mathbb{N}$ be a function in $\#\mathsf{P}/\mathsf{poly}$, then the sequence (f_n) defined by*

$$f_n(X) = \sum_{e \in \{0,1\}^n} \phi(n) \times \prod_{i=1}^{n} X_i^{e_i}$$

is in VNP.

Definition 6 (p-projections, Hardness and Completeness). *A polynomial sequence (f_n) is a p-projection of another polynomial sequence (g_n) if there is a polynomially bounded function $m : \mathbb{N} \to \mathbb{N}$ such that for each $n \in \mathbb{N}$, f_n can be obtained from $g_{m(n)}$ by setting its variables to one of the variables of f_n or field constants. It is denoted by $(f_n) \leq_p (g_n)$. A polynomial sequence (g_n) is said to be hard for a class $\mathcal{C}$ if for every sequence $(f_n) \in \mathcal{C}$, $(f_n) \leq_p (g_n)$. A polynomial sequence (g_n) is said to be complete for a class $\mathcal{C}$, if $(g_n) \in \mathcal{C}$ and is hard for $\mathcal{C}$.*

2.2 Related work

This section provides an overview of the results and models that our work builds upon. We briefly recall the framework of stack-realizable paths and stack branching programs, along with the characterizations of VP and VNP due to S. Mengel [5] and to S. Chillara and N. Raja [3].

Definition 7 (Stack realizable path). *Let Σ be a set of alphabets and k be the number of stacks. Let G be a directed graph with edge label $\phi : E(G) \to \{\texttt{Push}(\square, j), \texttt{Pop}(\square, j), \texttt{No-op}\}$, where $\square \in \Sigma$ and $j \in [k]$. The label $\texttt{Push}(\square, j)$ represents the operation of pushing the symbol $\square$ on stack j and the label $\texttt{Pop}(\square, j)$ represents the operation of popping the symbol $\square$ from the top of the stack. The label $\texttt{No-op}$ represents no operation. Let s, $t \in V(G)$. An s to t directed path is a stack-realizable path if and only if the following conditions are satisfied.*

- *We start with an empty stack and perform all the stack operations along the path and end with an empty stack.*
- *There are no faults while executing the stack operations along the path, that is, we do not attempt to pop an element from the stack that is not at the top of the stack.*

We now state the definition of a stack branching program with k stacks.

Definition 8 (Stack Branching Program with k stacks). *Let Σ denote a finite set of symbols. A stack branching program with k stacks is defined as an algebraic branching program $\mathcal{A}$ (having s and t as the source and the sink vertices, respectively) with two kinds of edge labels, the variable label and the stack label, where the variable label is $\phi_v : E(\mathcal{A}) \to \mathbb{F} \cup X$ and the stack label is $\phi_s : E(\mathcal{A}) \to \{\texttt{Push}(\square, j), \texttt{Pop}(\square, j), \texttt{No-op}\}$, where $\square \in \Sigma$ and $j \in [k]$. The polynomial computed by $\mathcal{A}$ is $\sum_{p \in \mathcal{P}} mon(p)$, where $\mathcal{P}$ denote the set of all stack realizable paths and $mon(p)$ is the monomial formed by mutliplying all the variable labels of the edges in path p. The size of a stack branching program is the number of vertices in it.*

Finally, we restate the characterization results.

Lemma 2 (Characterization of VP and VNP). *[5] [3] A sequence (f_n) is in VP if and only if f_n can be computed by a stack branching program (with one stack) of size polynomially bounded in n. A sequence (f_n) is in VNP if and only if f_n can be computed by a stack branching program (with k stacks) of size polynomially bounded in n, where $k \geq 2$.*

3 Our work

In this section, we present our contributions. We introduce conditional stack branching programs, define properly colored cycle covers, provide a formal definition of the colored determinant, and then state our main results.

Definition 9 (Conditional Stack Branching Program with one stack). *Let Σ be a sequence of finite set of symbols. A conditional stack branching program is defined as an algebraic branching program $\mathcal{A}$ (with s and t as the source and the sink vertices, respectively) with two kinds of edge labels, the variable label Φ_v and the stack label Φ_s, where the variable label is $\Phi_v : E(\mathcal{A}) \to \mathbb{F} \cup X$ and the stack label Φ_s is of the form*

$$if \quad (top == t) \quad \texttt{Pop}(t) \quad else \quad \texttt{Push}(t)$$

where the top represents the topmost element of the stack. An s to t path is stack realizable if and only if the following condition holds.

- *We start with an empty stack and perform all the stack operations along the path and end with an empty stack.*

The polynomial computed by $\mathcal{A}$ is $\sum_{p \in \mathcal{P}} mon(p)$, where $\mathcal{P}$ denote the set of all stack realizable paths and $mon(p)$ is the monomial formed by mutliplying all the variable labels of the edges in path p.

Definition 10 (cycle cover, colored graph and a properly colored cycle cover). *A cycle cover of a directed graph G is a subgraph of G which is a disjoint union of cycles such that every vertex of G participates in exactly one of the cycles. A graph G is called a colored graph if we associate every edge of G with some color. A properly colored cycle cover of a colored graph G is a cycle cover of G such that each cycle in the cover is monochromatic.*

Definition 11 (Colored Determinant). *Let $\mathcal{H}_n(V, E)$ be a complete directed graph. Let $V(\mathcal{H}_n) = \mathcal{H}_1 \cup \mathcal{H}_2$, where $\mathcal{H}_1 = \{1, 2, \ldots, n\}$ and $\mathcal{H}_2 = \{n+1, n+2, \ldots, 2n\}$. Let $E(\mathcal{H}_n) = \{(i,j) | 1 \leq i, j \leq 2n\}$. Let $E(\mathcal{H}_n) = E_1 \cup E_2 \cup E_3$, where E_1 denote the set of edges within the vertices in $\mathcal{H}_1$, the set E_2 denote the set of edges within the vertices in $\mathcal{H}_2$ and the set E_3 denote the set of edges across the vertex sets $\mathcal{H}_1$ and $\mathcal{H}_2$. We color the edges in the sets E_1 and E_2 with color c_r. We color the edges in the set E_3 with c_b. Let $X = \{x_{i,j} | 1 \leq i, j \leq 2n\}$. Let $\Phi : E(\mathcal{H}_n) \to X$, where $\Phi((i,j)) = x_{i,j}$ be the label function. The colored determinant is defined as follows:*

$$\mathsf{ColDet}_n = \sum_{c \in \mathcal{C}} (-1)^{n+k} mon(c)$$

where $\mathcal{C}$ is the set of all properly colored cycle covers of G and $mon(c)$ is formed by multiplying the labels on all the edges in c and k is the number of cycles in cycle cover c (see Figure 1 (left)).

We now state our main results.

Lemma 3. *$\mathsf{ColDet}_n(X)$ is VNP-complete for all fields under p-projections.*

Using the Valiant's criterion (Lemma 1), the containment of $\mathsf{ColDet}_n(X)$ in VNP is straightforward. The VNP-hardness proof idea is similar to the classical VNP-hardness proof of permanent by Valiant [8] with minor modifications. The complete proof appears in Sections 4 and 5.

Lemma 4. *The polynomial sequence $\mathsf{ColDet}_n(X)$ can be computed by a conditional stack branching program (with one stack) of size polynomially bounded in n.*

The proof of Lemma 4 is discussed in Sections 6

Lemma 5. *For a given sequence (f_n), if f_n can be computed by a conditional stack branching programs S_n of size polynomially bounded in n, then $(f_n) \in \mathsf{VNP}$.*

The proof of Lemma 5 is presented in section 7

4 Proof of Lemma 3

The proof is a careful modification of the classical VNP-completeness proof of the permanent by Valiant. Section 4.1 discusses the rosette construction, Section 4.2 presents the *Determinantal Colored if and only if* gadget, and Section 4.3 combines these ingredients to construct a graph T_k. We conclude section 4.3 with a technical lemma (Lemma 7) describing the colored determinant of the graph T_k.

4.1 Rosette Construction $R(n)$ for any odd n [1]

Consider a cycle C_n of length n, where $V(C_n) = \{u_1, u_2, \ldots, u_n\}$ and $E(C_n) = \{e_i = (u_i, u_{i+1}) | 1 \leq i \leq n-1\} \cup \{e_n = (u_n, u_1)\}$. We call the vertices and the edges in the set $V(C_n)$ and $E(C_n)$ the connector vertices and the connector edges, respectively. For each edge e_k, we add a new vertex α_k. For each $k \in [n-1]$, we add directed edges from u_k to α_k and from α_k to u_{k+1}. We add directed edges (u_n, α_n) and (α_n, u_1). We add self-loops on all the vertices. We arbitrarily pick a connector vertex and set the weight of its self-loop to 1. We set the weights of the self-loops on all other vertices to -1. We set the weights of all the remaining edges (other than the self-loops) to 1. Let the color of each edge of $R(n)$ be c_r. This completes the construction of $R(n)$ (see Figure 1 (middle)). It is trivial to observe the following properties of Rosette $R(n)$.

1. Each cycle cover is a properly colored cycle cover.
2. Let $\phi \neq S \subseteq E(C_n)$, there exists exactly one cycle cover which includes all the edges in S and excludes all the edges in $\bar{S}$.

[1] In the case of any even n, the above construction works except that the product of the signature and the monomial associated with each cycle cover will be -1 instead of 1. This issue could be handled very easily and is presented in Appendix A.

3. There are exactly two cycle covers which does not use any of the edges in $E(C_n)$. The first one is the cycle cover consisting of a single cycle $(u_1, \alpha_1, u_2, \alpha_2, \ldots, u_n, \alpha_n, u_1)$, and the second one is where every vertex is covered by a self-loop.
4. The total number of properly colored cycle covers is $2^n + 1$.
5. The product of the signature and the weight of the monomial associated with each of the cycle covers in this rosette is the same and is one.

4.2 Determinantal Colored if and only if gadget $\mathcal{G}$

We now discuss the construction of the determinantal *colored if and only if* gadget. This gadget is in the spirit of the classical Valiant's *if and only if* gadget. Let $\mathcal{G} = (V, E)$ be a colored directed graph with $V(\mathcal{G}) = \{a_1, a_2\}$. Let $E(\mathcal{G}) = \{(a_1, a_2), (a_2, a_1)\}$. We set the weight of (a_1, a_2) as -1 and the weight of (a_2, a_1) as 1. We set the colors of both edges as c_b. This finishes the construction of $\mathcal{G}$.

Placing $\mathcal{G}$ between two edges (u, v) and (u', v'): Let G be a colored directed graph with a pair of distinct edges (u, v) and (u', v') which does not share any common vertex. We assume that all the edges of the graph G are set with a color c_r. We say $\mathcal{G}$ is placed between (u, v) and (u', v'), if we delete edges (u, v), (u', v') in graph G, and add edges in the set $\{(u, a_1), (a_1, v), (u', a_2), (a_2, v')\}$. We set the color of all the newly added edges as c_r. Moreover, we set the weight of edge (u, a_1) as the weight of edge (u, v) and the weight of edge (u', a_2) as the weight of edge (u', v'). We set the weight of all other edges as 1 (see Figure 1 (right)).

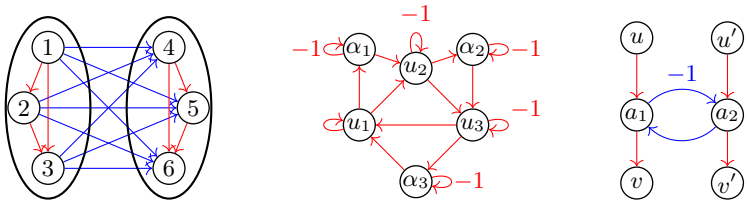

Fig. 1. Graph $\mathcal{H}_3$. To avoid clutter, edge labels are omitted in the figure and only the edges of the form (i, j), where $i < j$ are shown (left), the colored rosette $R(3)$ (middle), the colored determinantal if and only if gadget (right)

We now state our lemma, which describes the properties of the gadget $\mathcal{G}$.

Lemma 6 (Determinantal colored if and only if gadget). *Let G be a directed graph where every edge is colored with c_r. We fix two arbitrary edges (u, v) and (u', v') in graph G that do not share any endpoints. Let $\mathcal{G}$ (with color c_b) is placed between (u, v) and (u', v'). Let us call the new graph G'. Then the following holds.*

- *For any cycle cover C of graph G that uses both the edges (u, v) and (u', v') with signature s and monomial m, there exists a corresponding properly colored cycle cover C' in G' with the signature s' and m' such that $s' = s$ and $m' = m$. In other words, $s \times m = s' \times m'$.*
- *For any cycle cover C of graph that uses none of the edges (u, v) and (u', v') with signature s and monomial m, there exists a corresponding properly colored cycle cover C' in G' with the signature s' and monomial m' such that $s' = -s$ and $m' = -m$. In other words, $s \times m = s' \times m'$.*
- *There are no other properly colored cycle covers in G'.*

The proof of Lemma 6 is presented in Appendix B.

4.3 Construction of T_k

1. Let $(f_n) \in \mathsf{VNP}$. Then, there exists a sequence $g_n(X, y_1, y_2, \ldots, y_{p(n)}) \in \mathsf{VF}$, such that $f_n = \sum_{y_1,\ldots,y_{p(n)} \in \{0,1\}} g_n(X, y_1, y_2, \ldots, y_{p(n)})$, where $p(n)$ is polynomially bounded in n. Since $(g_n) \in \mathsf{VF}$, it is known that there exists a layered algebraic branching program $\mathcal{B}_n$ of size polynomially bounded in n that computes g_n. Let s and t be the source and the sink vertices of $\mathcal{B}_n$.
2. We add a new vertex θ and edges (θ, s) and (t, θ) to graph $\mathcal{B}_n$. We add self-loops on all the vertices except θ. We now color all the edges of $\mathcal{B}_n$ with color c_r. We call this new graph $\widehat{\mathcal{B}}_n$. Let $\mathbf{Y}_i$ be the number of edges in $\widehat{\mathcal{B}}_n$ labelled by variable y_i.
3. Consider the graph formed with the disjoint union of $\widehat{\mathcal{B}}_n$ and $R(\mathbf{Y}_i)$ for all $i \in [p(n)]$. Recall that the color of every edge of $\widehat{\mathcal{B}}_n$ and $R(\mathbf{Y}_i)$ is c_r. Let us call it $\mathcal{T}'_k$, where k' is the size of this graph.
4. For each $i \in [p(n)]$, let $e_{y_i}^{(1)}, e_{y_i}^{(2)}, \ldots, e_{y_i}^{(\mathbf{Y}_i)}$ be the edges in $\widehat{\mathcal{B}}_n$ labelled with y_i and let $\mathbf{c}_{y_i}^{(1)}, \mathbf{c}_{y_i}^{(2)}, \ldots, \mathbf{c}_{y_i}^{(\mathbf{Y}_i)}$ be the $\mathbf{Y}_i$ connector edges of rosette $R(\mathbf{Y}_i)$. We finally place a *colored determinantal if and only if gadget* (with color c_b) between e_i and $\mathbf{c}_i$. We now set all the Y variables to 1. We call this final graph T_k, where k is the size of the graph, and it is clearly polynomially bounded in n.

Lemma 7. *The colored determinant of the graph T_k is $f_n(X)$.*

We present the proof details of Lemma 7 in Section 5.

5 Proof of Lemma 7

5.1 Cycle covers of graph $\widehat{\mathcal{B}}_n$

In this section, we study the cycle covers of the graph $\widehat{\mathcal{B}}_n$.

Lemma 8. *Let $\mathcal{B}_n$ be the layered algebraic branching program of size $s = \text{poly}(n)$ computing $g_n(X, Y)$. Let $\mathcal{P} = \{\mathcal{P}_i | 1 \leq i \leq \mu\}$ be the set of all $s - t$ paths in $\mathcal{B}_n$. Let m_i be the monomial associated with path $\mathcal{P}_i$, formed by multiplying all the edges of $\mathcal{B}_n$ participating in path $\mathcal{P}_i$. Let $\widehat{\mathcal{B}}_n$ be the graph formed from $\mathcal{B}_n$ as described in point 2 of Section 4.3. The graph $\widehat{\mathcal{B}}_n$ satisfies the following properties.*

- *Every cycle cover of $\widehat{\mathcal{B}}_n$ is properly colored. For each $i \in [\mu]$, there exists a unique cycle cover C in $\widehat{\mathcal{B}}_n$, such that the monomial associated with C is m_i. Moreover, there are no other cycle covers in $\widehat{\mathcal{B}}_n$.*
- *The signature of each cycle cover of graph $\widehat{\mathcal{B}}_n$ is same. We assume it to be one.*

The proof of Lemma 8 is presented in Appendix C.

5.2 Proof of VNP completeness

In this section, we state our main proof. Let $f_n(X) \in \mathsf{VNP}$. Let $\mathcal{T}_{k'}$ and T_k denote the graphs constructed from $f_n(X)$ as stated in Section 4.3. We know that there exists an algebraic branching program $\mathcal{B}_n$ which computes the polynomial $g_n(X, Y)$. Let $\mathcal{P}$ be the set consisting of all the $s-t$ paths in $\mathcal{B}_n$. Let $\mathcal{P} = \{\mathcal{P}_i | 1 \leq i \leq \mu\}$. Let $c_i m_i$ denotes the monomial associated with path $\mathcal{P}_i$ in $\mathcal{B}_n$, where $c_i \in \mathbb{F}$ is the coefficient of m_i. Therefore, $g_n(X, Y) = c_1 m_1 + c_2 m_2 + \ldots + c_\mu m_\mu$. We have, $f_n(X) = \sum_{y_1, y_2, \ldots, y_{p(n)} \in \{0,1\}} c_1 m_1 + c_2 m_2 + \ldots + c_\mu m_\mu$

$= \sum_{y_1, y_2, \ldots, y_{p(n)} \in \{0,1\}} c_1 m_1 + \ldots + \sum_{y_1, y_2, \ldots, y_{p(n)} \in \{0,1\}} c_\mu m_\mu$.

Let $\mathcal{J}_j$ denotes the polynomial $\sum_{y_1, \ldots, y_{p(n)}} c_j m_j$. Therefore, $f_n(X) = \sum_{j=1}^{\mu} \mathcal{J}_j$.

We know that m_j is a monomial over the variable set $X \cup Y$. Therefore, m_j is the product of two monomials $m_j^{(X)}$ and $m_j^{(Y)}$, where $m_j^{(X)}$ is over the variable set X and $m_j^{(Y)}$ is over the variable set Y.
That is,
$g_n(X, Y) = c_1 m_1^{(X)} m_1^{(Y)} + c_2 m_2^{(X)} m_2^{(Y)} + \ldots + c_\mu m_\mu^{(X)} m_\mu^{(Y)}$.

Consider a monomial $m_j = c_j m_j^{(X)} m_j^{(Y)}$, let $V_j \subseteq Y$ is the set of variables whose degree is at least one in $m_j^{(Y)}$. Let $V_j = \{y_{t_1}, y_{t_2}, \ldots, y_{t_\ell}\}$ and the corresponding index set $V_j' = \{t_1, t_2, \ldots, t_\ell\}$, where $1 \leq t_1 < t_2 < \ldots < t_\ell \leq p(n)$. An evaluation of monomial $c_j m_j$ at any point where at least one variable from $V_j \subseteq Y$ is set to 0 vanishes. The evaluations where the monomial $c_j m_j$ survives is only at the points of the form $y_{t_1} = 1, \ldots, y_{t_\ell} = 1$ and for all $\Gamma \notin V_j'$, $y_\Gamma = *$, where $* \in \{0, 1\}$. That is, the coefficient of $c_j m_j^{(X)}$ is equal to the $2^{p(n)-|V_j|}$ in $\mathcal{J}_j$. This is true for every monomial $c_j m_j$, for all $1 \leq j \leq \mu$.
Cycle covers of $\mathcal{T}_{k'}$: Consider the graph $\mathcal{T}_{k'}$ as described in point 3 of Section 4.3. We analyse the properly colored cycle covers of graph $\mathcal{T}_{k'}$. Observe that any properly colored cycle cover of $\mathcal{T}_{k'}$ is a disjoint union of the properly colored cycle covers of each of its disjoint components. Since the graph $\mathcal{T}_{k'}$ is monochromatic, we hereafter drop the mention of cycle covers as properly colored cycle covers. For the component $\widehat{\mathcal{B}}_n$, any cycle cover must cover the vertex θ by a cycle that uses the edge (θ, s), followed by a directed $s - t$ path in the underlying branching program followed by (t, θ) and all other vertices must get covered by self-loops. Let $\mathcal{C}_j$ be a cycle cover of graph $\widehat{\mathcal{B}}_n$, where the vertex θ

is covered by an edge (θ, s), followed by a directed $s-t$ path $\mathcal{P}_j$ followed by an edge (t, θ) and all other vertices are covered with self-loops. We assume the signature of every cycle cover of $\widehat{\mathcal{B}}_n$ to be one. For each $i \in [p(n)]$, we know that the rosette $R(\mathbf{Y}_i)$ can be covered in $2^{\mathbf{Y}_i}+1$ ways. Moreover, the product of the signature and the monomial associated with each rosette is one. Therefore, the colored determinant of graph $\mathcal{T}_{k'}$ is $\sum_{j\in[\mu]}\left(c_j m_j \prod_{t=1}^{p(n)}(2^{\mathbf{Y}_t}+1)\right)$.

Cycle covers of T_k: Recall that $V_j = \{y_{t_1}, y_{t_2}, \ldots, y_{t_\ell}\}$ is the set of variables whose degree is at least one in m_j. By the property of *colored determinantal if and only if* gadget, the cycle cover of the rosettes $R(\mathbf{Y}_\alpha)$ for $\alpha \in V_j'$ must pick a subset of its connector edges which can be done in only way (by property 2 of rosette) and the cycle cover of the rosettes $R(\mathbf{Y}_\beta)$ for $\beta \in \overline{V_j'}$ must not pick any of the connector edges which can be done in two ways (by property 3 of rosette). Therefore, for all $\alpha \in V_j$, the factor $(2^{\mathbf{Y}_\alpha}+1)$ drops to one whereas for all $\beta \in \overline{V_j'}$ the factor $(2^{\mathbf{Y}_i}+1)$ drops to 2. Therefore, the determinant of the graph T_k will be $\sum_{j\in[\mu]} 2^{p(n)-|V_j|} c_j m_j^{(X)}$.

6 Proof of Lemma 4

We prove Lemma 4 in the following three steps.

- We define a variant of colored determinant which is the sum of over all properly colored clow sequences instead of cycle covers (see Subsection 6.1).
- We then show that such a variant is the same as the colored determinant (see Subsection 6.2).
- We construct a conditional stack branching program to compute the colored determinant over properly colored clow sequences instead of properly colored cycle covers (See Subsection 6.3).

6.1 Colored Determinant as a sum of clow sequences

We first recall the definitions of a closed walk and a clow sequence from [4].

Definition 12 (A closed walk, A clow sequence). *[4] A closed walk (clow) over a directed graph $G = (V, E)$, where $V = [n]$ is a walk where the minimum numbered vertex is visited exactly once. The minimum numbered vertex is called the head of a clow. A sequence $C = \langle c_1, c_2, \ldots, c_k\rangle$ is a clow sequence if for all $i \in [k]$, c_k is a clow. Moreover, $Head(c_1) < Head(c_2) < Head(c_3) \ldots < Head(c_k)$. The degree of the clow sequence C is the sum of the total number of edges participating in each clow in c (counted with multiplicity).*

We now extend these definitions to colored graphs and define a properly colored clow and a properly colored clow sequence.

Definition 13 (Properly colored clow). *A clow of a colored graph is called a properly colored clow, if every simple cycle within the clow is colored with the same color (see Figure 2).*

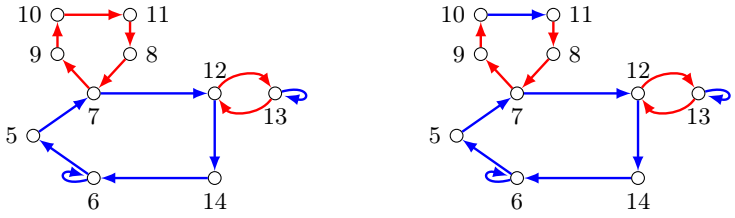

Fig. 2. A properly colored clow $\langle 5, 7, 9, 10, 11, 8, 7, 12, 13, 13, 12, 14, 6, 6, 5\rangle$ with head 5 and degree 14 (left). A clow with head 5 and degree 14 which is not properly colored (right).

Definition 14 (Properly colored clow sequence). *A sequence $C = \langle c_1, \ldots, c_k\rangle$ over a colored graph is called a properly colored clow sequence if C is a clow sequence and for all $i \in [k]$, c_k is a properly colored clow.*

We define a new variant of Colored Determinant, which, instead of summing over all properly colored cycle covers, it sums over all properly colored clow sequences.

Definition 15. *Let $\mathcal{H}_n(V, E)$ be a complete directed graph. Let $V(\mathcal{H}_n) = \mathcal{H}_1 \cup \mathcal{H}_2$, where $\mathcal{H}_1 = \{1, 2, \ldots, n\}$ and $\mathcal{H}_2 = \{n+1, n+2, \ldots, 2n\}$. Let $E(\mathcal{H}_n) = \{(i, j) | 1 \leq i, j \leq 2n\}$. Let $E(\mathcal{H}_n) = E_1 \cup E_2 \cup E_3$, where E_1 denote the set of edges within the vertices in $\mathcal{H}_1$, the set E_2 denote the set of edges within the vertices in $\mathcal{H}_2$ and the set E_3 denote the set of edges across the vertex sets $\mathcal{H}_1$ and $\mathcal{H}_2$. We color the edges in the sets E_1 and E_2 with color c_r. We color the edges in the set E_3 with c_b. Let $X = \{x_{i,j} | 1 \leq i, j \leq 2n\}$. Let $\Phi : E(\mathcal{H}_n) \to X$, where $\Phi((i, j)) = x_{i,j}$ be the label function. The colored determinant (over clow sequences) is defined as follows:*

$$\mathsf{ColDet}^*_n(X) = \sum_{c \in \mathcal{C}} (-1)^{n+k} mon(c)$$

where $\mathcal{C}$ is the set of all properly colored clow sequences of degree $2n$ of graph $\mathcal{H}_n$ and $mon(c)$ is formed by multiplying the labels on all the edges in c and k is the number of clows in c.

6.2 Involution on the set of properly colored clow sequences

Like in the case of the determinant [4], we show that the colored determinant defined as a signed sum of properly colored cycle covers can also be expressed as the signed sum of properly colored clow sequences. Lemma 9 formalizes this.

Lemma 9. $\mathsf{ColDet}^*_n(X) = \mathsf{ColDet}_n(X)$

The main idea is to show that the properly colored clow sequences that are not cycle covers always appear in pairs with opposite signatures and, therefore, contribute 0 to the overall sum. The formal proof of Lemma 9 appears in Appendix D.

6.3 Construction of a conditional stack Branching program $\mathcal{A}'_{\mathcal{H}_n}$ to compute $\mathsf{ColDet}^*_n(X)$

We present the construction in three major steps.

- We recall the structure of an $s-t$ path in the algebraic branching program computing the determinant [4] (see Subsection 6.3.1 for details)..
- We modify the above construction to remember the color of the last traversed edge in the clow sequence (see Subsection 6.3.2).
- We finally use the stack operation labels on the edges of the modified branching program such that it computes $\mathsf{ColDet}^*_n(X)$ (see Subsection 6.3.3 for more details).

6.3.1 Determinant as a sum of signed clow sequences We recall the definition of the determinant, which is defined as the sum of signed clow sequences.

Definition 16. *[4] Consider a directed graph $G = (V, E)$ with $V = [n]$ and $E = \{e_{i,j} = (i,j) | 1 \leq i,j \leq n\}$. Let $X = \{x_{i,j} | 1 \leq i,j \leq n\}$ be the set of variables. Let $\Phi : E(G) \to X$, where $\phi(e_{i,j}) = x_{i,j}$ The determinant is $\mathsf{Det}_n(X) = \sum_{c \in \mathcal{C}} (-1)^{n+k} mon(c)$ where the sum is over all clow sequences $\mathcal{C}$ of degree n and k is the number of clows in the clow sequence c.*

We know that there exists an algebraic branching program, say $\mathcal{F}_n$ of size $\mathcal{O}(n^3)$ that computes $\mathsf{Det}_n(X)$. Moreover, the graph $\mathcal{F}_n$ satisfies the following properties [4].
For every clow sequence $\mathcal{C} = \langle \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k \rangle$ of degree n and positive signature (or negative signature), let $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k$ be the paths formed by unwinding the clows $\mathcal{C}_1$, $\mathcal{C}_2$, ..., $\mathcal{C}_k$, respectively. For every clow sequence $\mathcal{C} = \langle \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k \rangle$, there exists a unique $s-t$ path $\mathcal{P}$ in $\mathcal{F}_n$ such that the path $\mathcal{P}$ is an edge (s, s') followed by paths $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k$ and then followed by a single edge $\hat{e}$ labelled by $+1$ (-1, respectively). There are no $s-t$ paths in $\mathcal{F}_n$ other than the kind of paths stated above.

6.3.2 Determinant over a colored graph G We present the construction of an algebraic branching program $\mathcal{A}_G$ to compute the determinant polynomial $\mathsf{Det}_n(X)$ over graph G. Our construction is essentially the same as the construction stated in [4], except for a few minor modifications. Let $G = (V, E)$ be a colored directed graph such that $|V(G)| = n$ and a function Φ such that $\Phi((i,j)) \to x_{i,j}$. Let $\Phi' : E(G) \to \{c_b, c_r\}$ be the color function. Let H_n denote the vertex set of $\mathcal{A}_G$ where, $H_n = \{s, t\} \cup H'_n$ and

$$H'_n = \{[p, h, u, i, c] | p \in \{0,1\}, h \in [n], u \in [n], i \in [n], c \in \{c_r, c_b, \otimes\}\}.$$

The meanings associated with p, h, u, and i in our construction are the same as given in [4]. However, in our construction, since we are working with a colored graph, we also wish to remember the color of the last traversed edge in our traversal; therefore, a new component c is added in the tuple to remember the color of the last traversed edge. We now add the following edges in $\mathcal{A}_G$.

- $(s, [p, h, h, 0, \otimes])$ for $h \in \{1, \ldots, n\}$, where $p = n \bmod 2$.
- $([p, h, u, i, c], [p, h, v, i+1, c_r])$ if $v > h$ and $i+1 \leq n$; the color of the edge (u, v) is c_r, the weight of this edge is set to x_{uv}.
- $([p, h, u, i, c], [p, h, v, i+1, c_b])$ if $v > h$ and $i+1 \leq n$; the color of the edge (u, v) is c_b and the weight of this edge is set to x_{uv}.
- $([p, h, u, i, c], [\bar{p}, h', h', i+1, c_r])$ if $h' > h$ and $i+1 \leq n$; the color of the edge (u, h) is c_r and the weight of this edge is x_{uh}.
- $([p, h, u, i, c], [\bar{p}, h', h', i+1, c_b])$ if $h' > h$ and $i+1 \leq n$; the color of the edge (u, h) is c_b and the weight of this edge is x_{uh}.
- $([1, h, u, n, c], t)$; the weight of this edge is $+1$
- $([0, h, u, n, c], t)$; the weight of this edge is -1.

This completes the construction of $\mathcal{A}_G$ (see Figure 3). We now state our lemma formally.

Lemma 10. *The algebraic branching program $\mathcal{A}_{\mathcal{H}_n}$ computes the determinant $\mathsf{Det}_n(X)$ over graph $\mathcal{H}_n$.*

The proof of Lemma 10 is straightforward and immediately follows from [4].

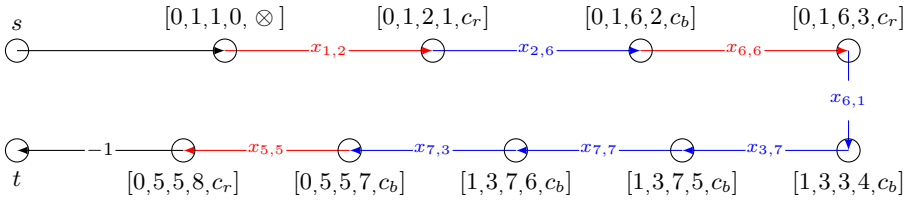

Fig. 3. A $s-t$ path in $\mathcal{A}_{\mathcal{H}_4}$ corresponding to clow sequence $\langle c_1, c_2, c_3 \rangle$ of degree 8, where $c_1 = (1, 2, 6, 6, 1)$, $c_2 = (3, 7, 7, 3)$, $c_3 = (5, 5)$

6.3.3 Modifying $\mathcal{A}_{\mathcal{H}_n}$ to $\mathcal{A}'_{\mathcal{H}_n}$ such that $\mathcal{A}'_{\mathcal{H}_n}$ computes colored determinant In this section, we add the conditional stack operations to the edges of $\mathcal{A}_{\mathcal{H}_n}$ to construct a conditional stack branching program $\mathcal{A}'_{\mathcal{H}_n}$ such that $\mathcal{A}'_{\mathcal{H}_n}$ computes $\mathsf{ColDet}^*_n(X)$. The main idea here is to remember in which clow and at what vertex in the traversal of the clow sequence, there is a color change and record it onto the stack. Therefore, we define an alphabet set sequence $\Sigma_n = \{\triangle_{h,a} | h \in [n], a \in [n]\}$, where h denotes the head of the clow in which the color change occurs and a denotes the vertex at which the color change occurs. For each edge e in $\mathcal{A}_{\mathcal{H}_n}$ of the form $([p, h, u, \delta, r])$ to $([p, h, v, \delta + 1, b])$ (or $([p, h, u, \delta, b])$ to $([p, h, v, \delta + 1, r])$), we set the stack label as

$$if(top == \triangle_{h,u}) \quad \mathtt{Pop}(\triangle_{h,u}) \qquad else \qquad \mathtt{Push}(\triangle_{h,u}).$$

A closing edge of a clow is the last edge of the clow. Note that edges directed from $([p, h, u, \delta, r])$ to $([p, h', v, \delta + 1, b])$ (or, $([p, h, u, \delta, b])$ to $([p, h', v, \delta + 1, r])$) are labelled with `No-op` (no operation) as these edges are closing edges of a clow and a color change here is not a true color change in the clow traversal. We set the stack operation labelings of all other edges to `No-op` (no operation) (see Figure 4). We state our lemma formally.

Lemma 11. *The conditional algebraic branching program $\mathcal{A}'_{\mathcal{H}_n}$ computes the colored determinant $\mathsf{ColDet}^*_n(X)$ of graph $\mathcal{H}_n$.*

The proof of Lemma 11 appears in Appendix E.

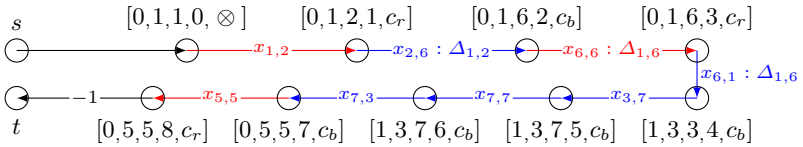

Fig. 4. A $s - t$ path in $\mathcal{A}'_{\mathcal{H}_4}$ corresponding to clow sequence $\langle c_1, c_2, c_3 \rangle$ of degree 8, where $c_1 = (1, 2, 6, 6, 1)$, $c_2 = (3, 7, 7, 3)$, $c_3 = (5, 5)$. Since a conditional stack operation can be uniquely identified by the stack symbol, we write $x : \gamma$ as a shorthand to mean x as the variable label and γ as the conditional stack operation with γ as the stack symbol

7 Proof of Lemma 5

Proof. Since checking if a path through a conditional stack branching program is realizable or not is easy and can be done in P, (f_n) is in VNP follows from Valiant's criterion (Lemma 1). □

Acknowledgements. The author thanks Nikhil Balaji for helpful discussions and the anonymous reviewers for their feedback, which improved the presentation of this work.

References

1. Chaugule, P., Limaye, N.: On the closures of monotone algebraic classes and variants of the determinant. Algorithmica **86**(7), 2130–2151 (2024)
2. Chaugule, P., Limaye, N., Pandey, S.: Variants of the determinant polynomial and the VP-completeness. In: The 16th International Computer Science Symposium in Russia. vol. 12730, pp. 31–55. Springer (2021), `https://doi.org/10.1007/978-3-030-79416-3_3`
3. Chillara, S., Raja, N.: Branching programs with extended memory: New insights. In: International Conference on Algorithms and Complexity. pp. 91–104. Springer (2025)

4. Mahajan, M., et al.: Determinant: Combinatorics, algorithms, and complexity. Chicago Journal of Theoretical Computer Science **1997**(5) (1997)
5. Mengel, S.: Arithmetic branching programs with memory. In: International Symposium on Mathematical Foundations of Computer Science. pp. 667–678. Springer (2013)
6. Saptharishi, R.: A survey of lower bounds in arithmetic circuit complexity. Github survey **95** (2015)
7. Shpilka, A., Yehudayoff, A.: Arithmetic circuits: A survey of recent results and open questions. Foundations and Trends® in Theoretical Computer Science **5**(3–4), 207–388 (2010)
8. Valiant, L.G.: Completeness classes in algebra. In: Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing. pp. 249–261. STOC '79 (1979)
9. Valiant, L.G.: Reducibility by algebraic projections. University of Edinburgh, Department of Computer Science (1980)

A $R(n)$ for any even n

For the given even n, we first construct the rosette as described in Section 4.1 and call it $\widehat{R(n)}$. Consider a directed loop X with weight set to -1. The final rosette $R(n)$ is the disjoint union of the rosette $\widehat{R(n)}$ and graph X.

B Proof of Lemma 6

Let G be a colored directed graph G, where the color of each edge of graph G is c_r. We place $\mathcal{G}$ between edges (u, v) and (u', v'). Recall that the color of each edge of $\mathcal{G}$ is c_b. The new graph formed is denoted by G'.

- Let C be a colored cycle cover with signature s and monomial m which uses both the edges (u, v) and (u', v') of G. Then there exists another cycle cover C' in G', such that the product of the signature and the monomial of C is the same as the product of the signature and the monomial of C'. Observe that the cycle cover C' is the same as the cycle cover C, except that the edges (u, v) and (u', v') in C are replaced by length two directed paths (u, a_1, v) and (u', a_2, v'), respectively.
- Let C be a colored cycle cover with signature s and monomial m, which does not use any of the edges from (u, v) and (u', v') of graph G. Then there exists another cycle cover C' in G' with the signature s' and monomial m' such that $s' = -s$ and $m' = -m$. The cycle cover C' is simply the disjoint union of C and the two-length cycle (a_1, a_2, a_1).
- Let C be a colored cycle cover of G which uses the edge (u, v) but not (u', v'). The corresponding cycle cover, say C' in G' will consume the vertices u, a_1, and v, but there is no way to cover a_2, and therefore, C' vanishes in G'. A similar argument follows for the case of a cycle cover of G that uses (u', v') but not (u, v).
- We now consider the last case. Let C' be a cycle cover of G' that either uses both the edges from set $\{(u, a_1), (a_2, v')\}$ or from set $\{(u', a_2), (a_1, v)\}$. Then C' must use either the edges $(u, a_1), (a_1, a_2), (a_2, v')$ or $(u', a_2), (a_2, a_1), (a_1, v)$ in the given order in one of its cycle. Since such a cycle is not monochromatic, C' would vanish in graph G'. □

C Proof of Lemma 8

Proof. The only way to cover the vertex θ in $\widehat{\mathcal{B}}_n$ is by a cycle starting from the vertex θ followed by the edge (θ, s), followed by a directed path in $\widehat{\mathcal{B}}_n$, follwed by the edge (t, θ). Consider a cycle cover C of the graph $\widehat{\mathcal{B}}_n$. The cycle cover C must consist of a cycle C' which starts at θ and traverses an $s - t$ path in it and closes back at vertex θ. The remaining vertices of $\widehat{\mathcal{B}}_n$ in C must be covered by self-loops. Therefore, there is a one-to-one correspondence between the set of monomials $\mathcal{M} = \{m_k | 1 \leq k \leq \mu\}$ and the set of cycle covers of $\widehat{\mathcal{B}}_n$. Since the

algebraic branching program $\mathcal{B}_n$ is layered, the number of cycles in every cycle cover of $\widehat{\mathcal{B}}_n$ is the same; therefore, the signature is the same for all cycle covers. Moreover, since $\widehat{\mathcal{B}}_n$ is monochromatic, every cycle cover is properly colored. □

D Proof of Lemma 9

Proof. Consider a properly colored clow sequence $\mathcal{C} = \langle C_1, C_2, \ldots, C_k \rangle$ of $\mathcal{H}_n$. Let j be the minimum numbered index such that all the clows $C_{j+1}, \ldots, C_k$ form a set of disjoint simple cycles. If $j = 0$ then the sequence $\mathcal{C}$ is a properly colored cycle cover. If not, then we traverse the clow C_j starting from its head, until one of the following two things happens.

- We encounter a vertex that touches one of the cycles from $C_{j+1}, \ldots, C_k$.
- We encounter a vertex that completes a simple cycle within itself.

Let us call such a vertex α. Note that the vertex α cannot satisfy both the above conditions. We now discuss both cases in detail.
Case 1: Let α touches the cycle C_β.
Let $\widehat{\mathcal{C}} = \langle C_1, C_2, \ldots, C_{j-1}, C'_j, C_{j+1}, \ldots, C_{\beta-1}, C_\beta, \ldots, C_k \rangle$ where C'_j is formed by combining the sequences C_j and C_β. The head of the new clow $C_{j'}$ is the same as the head of clow C_j. The clow $C_{j'}$ inserts the cycle C_j into the clow C_i at the first occurrence, traversing from its head. Observe that the clow sequence $\widehat{\mathcal{C}}$ is also a properly colored clow sequence. The monomials associated with the clow sequences $\mathcal{C}$ and $\widehat{\mathcal{C}}$ are the same but with opposite signatures.
Case 2: Let C_j finishes a simple cycle T within itself at vertex α. Note that the simple cycle T cannot touch any of the later cycles; otherwise, we would be in case 1. Let us modify the sequence $\mathcal{C}$ by removing the cycle T from C_j and placing T as a new clow in our modified sequence.

Since the head of the cycle T is greater than the head of C_j, and is distinct from the heads of all the cycles later than C_j (in the sequence $\mathcal{C}$), the cycle T can be placed after the clow C_j at an appropriate place in our modified clow sequence. Let us call this modified clow sequence as $\widehat{\mathcal{C}}$. Clearly, $\widehat{\mathcal{C}}$ is a properly colored clow sequence. Moreover, the monomials associated with both $\mathcal{C}$ and $\widehat{\mathcal{C}}$ are the same but with opposite signatures. □

E Proof of Lemma 11

Proof. Let us call a clow sequence which is not a cycle cover a *true clow sequence*. Let S denote the set of all $s-t$ paths in $\mathcal{A}'_{\mathcal{H}_n}$. We partition the set S into two sets S_1 and S_2, where,

- S_1 consists of all $s-t$ paths corresponding cycle covers of $\mathcal{H}_n$,
- S_2 consists of all $s-t$ paths corresponding to the true clow sequences of $\mathcal{H}_n$.

For any path P in set S_1, the path P is stack-realizable in $\mathcal{A}'_{\mathcal{H}_n}$ if and only if the path P corresponds to a properly colored cycle cover in $\mathcal{H}_n$. This follows trivially.

For any properly colored true clow sequence $\mathcal{C}$ of $\mathcal{H}_n$, there exists a unique stack-realizable path P corresponding to $\mathcal{C}$ in $\mathcal{A}'_{\mathcal{H}_n}$. This follows trivially as well.

We now need to show that for any true clow sequence $\mathcal{C}$, which is not properly colored, the corresponding $s-t$ path in $\mathcal{A}'_{\mathcal{H}_n}$ is not stack-realizable. Instead, we show that if P is a stack-realizable path in $\mathcal{A}'_{\mathcal{H}_n}$ corresponding to some true clow sequence $\mathcal{C}$ then $\mathcal{C}$ must be properly colored. Let the sequence of executed stack operations associated with path P be Ψ_P. The length of any stack-realizable sequence is even, and any stack-realizable sequence of stack operations can be inductively formed by the following steps.

- The empty sequence is stack-realizable.
- If A and B are two stack-realizable sequences, then AB is a stack-realizable sequence.
- If A is a stack-realizable sequence, then $\mathrm{Push}(\alpha)\mathrm{APop}(\alpha)$ is a stack-realizable sequence, for some stack alphabet α.

Therefore, this implies that the clow sequence $\mathcal{C}$ must be properly colored. □

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il