Separating multilinear branching programs and formulas

Zeev Dvir Guillaume Malod
Sylvain Perifel Amir Yehudayoff

Lyon – July 4, 2012

Introduction

- Arithmetic circuits: model for computing polynomials.
- Algebraic variants of P vs NP.

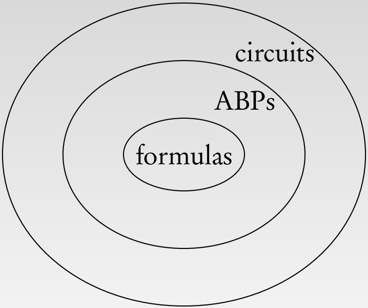
Introduction

- Arithmetic circuits: model for computing polynomials.
- Algebraic variants of P vs NP.
- No strong lower bounds for general circuits.

  (Best nontrivial lower bound for "explicit" polynomials: $\Omega(n \log n)$, Baur-Strassen)

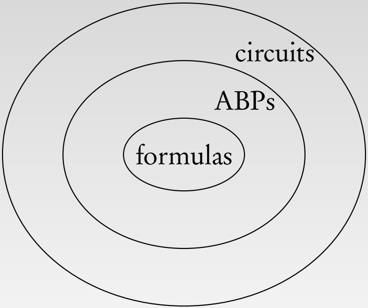
Introduction

- Arithmetic circuits: model for computing polynomials.
- Algebraic variants of P vs NP.
- No strong lower bounds for general circuits.
- Other weaker models:
  formulas and algebraic branching programs (ABPs).



Introduction

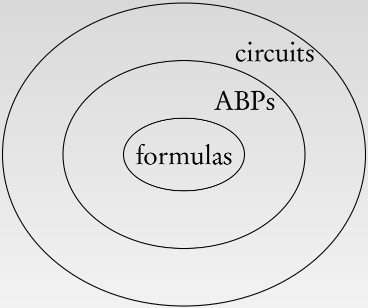
- Important restriction = multilinear computation.
- Raz 2004: separation of multilinear circuits and formulas.



Introduction

- Important restriction = multilinear computation.
- Raz 2004: separation of multilinear circuits and formulas.

Here: $\boxed{\text{separation of multilinear ABPs and formulas.}}$



Outline

1. Formulas, ABPs and multilinearity

2. The rank technique

3. Our separation

Outline

1. Formulas, ABPs and multilinearity

2. The rank technique

3. Our separation

Multilinearity

- A polynomial is multilinear if
  the degree of each variable is at most 1.
- Example: $x_1x_2 + x_1x_3 + x_2 + 1.$
- Counter-example: $x^2y + xyz.$

Multilinearity

- A polynomial is multilinear if
  the degree of each variable is at most 1.
- Example: $x_1x_2 + x_1x_3 + x_2 + 1.$
- Counter-example: $x^2y + xyz.$
- Important multilinear polynomials:
  determinant, permanent…

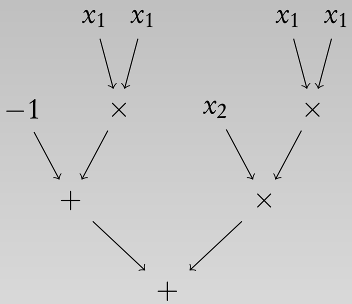
$$\det(x_{1,1}, \ldots, x_{1,n}, x_{2,1}, \ldots, x_{n,n}) = \sum_{\sigma \in \mathcal{S}_n} (-1)^{\epsilon(\sigma)} \prod_{i=1}^{n} x_{i,\sigma(i)}.$$

$$\mathrm{per}(x_{1,1}, \ldots, x_{1,n}, x_{2,1}, \ldots, x_{n,n}) = \sum_{\sigma \in \mathcal{S}_n} \prod_{i=1}^{n} x_{i,\sigma(i)}.$$

Arithmetic formula

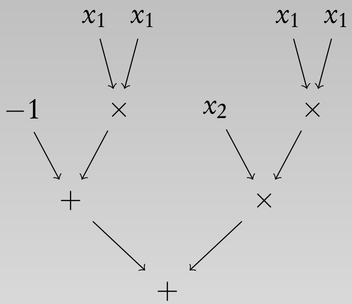
- Weakest model:
  each subcomputation
  can be used only once.

- Underlying graph = tree.



Arithmetic formula

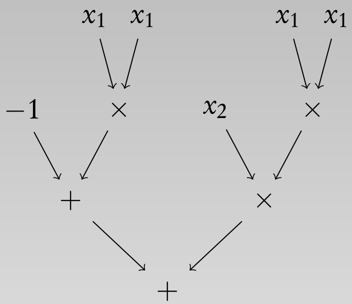
- Weakest model:
  each subcomputation
  can be used only once.



- Underlying graph = tree.

- Formulas can be parallelized (logarithmic depth) = efficient parallel algorithm.

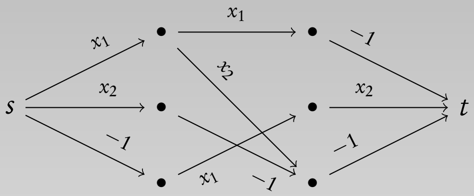
Arithmetic formula



- Weakest model:
  each subcomputation
  can be used only once.
- Underlying graph = tree.
- Formulas can be parallelized (logarithmic depth) =
  efficient parallel algorithm.
- Multilinear =
  each gate computes a multilinear polynomial.

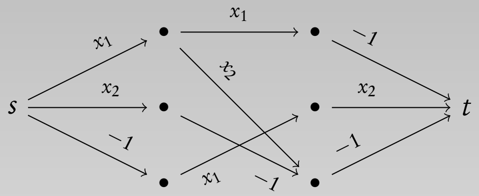
Algebraic Branching Program (ABP)

- DAG from a source s to a sink t with arcs labelled by constants or variables.



Algebraic Branching Program (ABP)

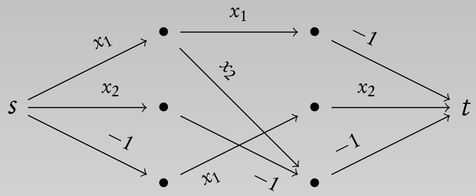
- DAG from a source s to a sink t with arcs labelled by constants or variables.



- Weight of a path = product of the labels.
- Polynomial computed by the ABP = sum of the weights of all paths from s to t.

Algebraic Branching Program (ABP)

- DAG from a source s to a sink t with arcs labelled by constants or variables.



- Weight of a path = product of the labels.

- Polynomial computed by the ABP = sum of the weights of all paths from s to t.

- Multilinear = on each path, each variable appears at most once.

Power of ABPs

- Polynomial-size ABPs capture the complexity of:
  - matrix multiplication
  - computing the determinant.

Power of ABPs

- Polynomial-size ABPs capture the complexity of:
  - matrix multiplication
  - computing the determinant.

- However no multilinear polynomial-size ABP known for the determinant.

Outline

1. Formulas, ABPs and multilinearity

2. The rank technique

3. Our separation

The coefficient matrix

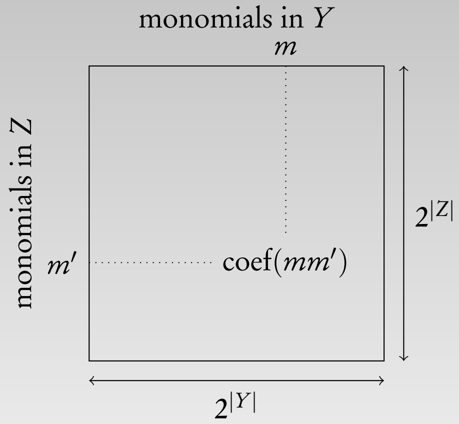
- aka matrix of partial derivatives

The coefficient matrix

- aka matrix of partial derivatives
- f multilinear polynomial over variables X
- a partition of X into Y and Z, $\Pi : X \to Y \cup Z$
- f_Π: renaming the variables according to Π

The coefficient matrix

- aka matrix of partial derivatives
- f multilinear polynomial over variables X
- a partition of X into Y and Z, $\Pi : X \to Y \cup Z$
- f_Π: renaming the variables according to Π
- $M(f_\Pi)$: coefficient matrix of f according to Π.



Example

- $X = \{x_1, x_2, x_3, x_4\}$
- $f(x_1, x_2, x_3, x_4) = 2x_2x_3x_4 + x_1x_2 + 5x_1x_3 - 2x_4 - 3$

Example

- $X = \{x_1, x_2, x_3, x_4\}$
- $f(x_1, x_2, x_3, x_4) = 2x_2x_3x_4 + x_1x_2 + 5x_1x_3 - 2x_4 - 3$
- $Y = \{y_1, y_2\}, \quad Z = \{z_1, z_2\}$
- $\Pi : x_1 \mapsto y_1, \quad x_2 \mapsto y_2,$
  $x_3 \mapsto z_1, \quad x_4 \mapsto z_2$

$$\rightarrow \qquad f_\Pi = 2y_2z_1z_2 + y_1y_2 + 5y_1z_1 - 2z_2 - 3$$

Example

- $X = \{x_1, x_2, x_3, x_4\}$
- $f(x_1, x_2, x_3, x_4) = 2x_2x_3x_4 + x_1x_2 + 5x_1x_3 - 2x_4 - 3$
- $Y = \{y_1, y_2\}, \quad Z = \{z_1, z_2\}$
- $\Pi : x_1 \mapsto y_1, \quad x_2 \mapsto y_2,$
  $x_3 \mapsto z_1, \quad x_4 \mapsto z_2$

$$\rightarrow \qquad f_\Pi = 2y_2z_1z_2 + y_1y_2 + 5y_1z_1 - 2z_2 - 3$$

$$M(f_\Pi) = \begin{array}{c|cccc} & 1 & y_1 & y_2 & y_1y_2 \\ \hline 1 & -3 & 0 & 0 & 1 \\ z_1 & 0 & 5 & 0 & 0 \\ z_2 & -2 & 0 & 0 & 0 \\ z_1z_2 & 0 & 0 & 2 & 0 \end{array}$$

Rank

The rank of the coefficient matrix has nice properties:

- $\mathrm{rank}(M((f+g)_\Pi)) \leq \mathrm{rank}(M(f_\Pi)) + \mathrm{rank}(M(g_\Pi))$

Rank

The rank of the coefficient matrix has nice properties:

- $\operatorname{rank}(M((f+g)_\Pi)) \leq \operatorname{rank}(M(f_\Pi)) + \operatorname{rank}(M(g_\Pi))$
- if f and g are on disjoint variables, then

$$\operatorname{rank}(M((fg)_\Pi)) = \operatorname{rank}(M(f_\Pi))\operatorname{rank}(M(g_\Pi))$$

Rank

The rank of the coefficient matrix has nice properties:

- $\mathrm{rank}(M((f+g)_\Pi)) \leq \mathrm{rank}(M(f_\Pi)) + \mathrm{rank}(M(g_\Pi))$
- if f and g are on disjoint variables, then

$$\mathrm{rank}(M((fg)_\Pi)) = \mathrm{rank}(M(f_\Pi))\mathrm{rank}(M(g_\Pi))$$

- if $Y(f)$ and $Z(f)$ are the numbers of Y and Z variables appearing in f_Π, then

$$\mathrm{rank}(M(f_\Pi)) \leq 2^{\min(Y(f),Z(f))}.$$

The rank technique

Separation of multilinear circuits and formulas (Raz 2004):

- build a polynomial f such that:
  - f is computed by polynomial size circuits;
  - for all partition Π, $M(f_\Pi)$ has full rank ("f_Π is full rank");

The rank technique

Separation of multilinear circuits and formulas (Raz 2004):

- build a polynomial f such that:
  - f is computed by polynomial size circuits;
  - for all partition Π, $M(f_\Pi)$ has full rank ("f_Π is full rank");
- any formula of polynomial size computes a polynomial g which is not full rank according to some partition Π.

The rank technique

Separation of multilinear circuits and formulas (Raz 2004):

- build a polynomial f such that:
  - f is computed by polynomial size circuits;
  - for all partition Π, $M(f_\Pi)$ has full rank ("f_Π is full rank");
- any formula of polynomial size computes a polynomial g which is not full rank according to some partition Π.
- Probabilistic method: g_Π is not full rank if Π is chosen at random.

Outline

1. Formulas, ABPs and multilinearity

2. The rank technique

3. Our separation

The result

Theorem

There exists a polynomial-size *multilinear* ABP computing a polynomial P that has no *multilinear* formula of size $n^{o(\log n)}$.

Strategy

Consider a restricted subset of all partitions:

- small enough so that
  an ABP can compute full-rank polynomials;
- big enough so that
  formulas cannot compute full-rank polynomials.

Strategy

Consider a restricted subset of all partitions:

- small enough so that
  an ABP can compute full-rank polynomials;
- big enough so that
  formulas cannot compute full-rank polynomials.

Lower bound:

- it suffices that polynomials computed by formulas
  are not full-rank for a single partition;
- however probabilistic argument:
  not full-rank for most partitions.

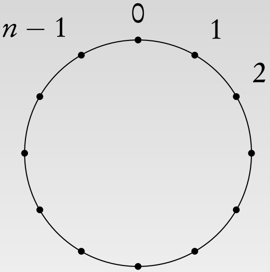
Pairings (1)

- Pairings:
  before partitioning, variables are grouped in pairs.

Pairings (1)

- Pairings:
  before partitioning, variables are grouped in pairs.

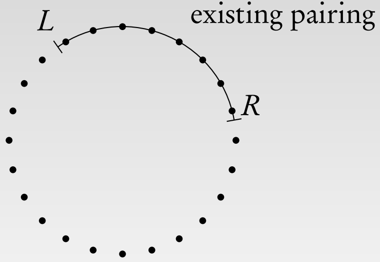
- Set of variables $X \equiv \{0, 1, \ldots, n-1\}$
  seen as the n-cycle C_n.



Pairings (2)

Random pairing: iterative process

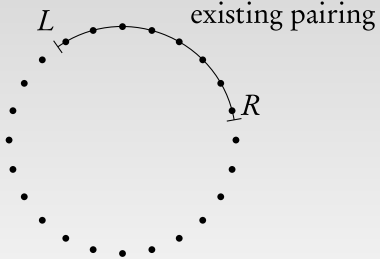
- First pair: $\{\mathtt{0}, 1\}$.
- At any given step,
  the set of vertices grouped in pairs forms an arc $[L, R]$.



Pairings (2)

Random pairing: iterative process

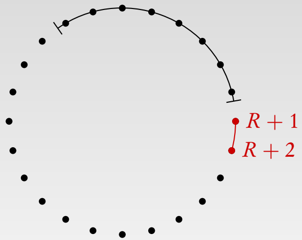
- First pair: $\{0, 1\}$.
- At any given step,
  the set of vertices grouped in pairs forms an arc $[L, R]$.
- Extending a pairing:
  3 equiprobable possibilities for the next pair.



Pairings (2)

Random pairing: iterative process

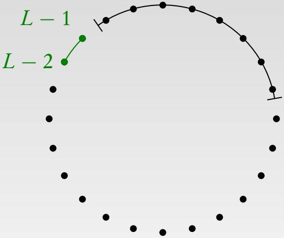
- First pair: $\{0, 1\}$.
- At any given step,
  the set of vertices grouped in pairs forms an arc $[L, R]$.
- Extending a pairing:
  3 equiprobable possibilities for the next pair.



Pairings (2)

Random pairing: iterative process

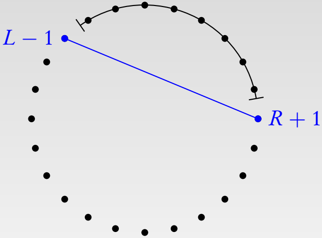
- First pair: $\{0, 1\}$.
- At any given step,
  the set of vertices grouped in pairs forms an arc $[L, R]$.
- Extending a pairing:
  3 equiprobable possibilities for the next pair.



Pairings (2)

Random pairing: iterative process

- First pair: $\{0, 1\}$.
- At any given step,
  the set of vertices grouped in pairs forms an arc $[L, R]$.
- Extending a pairing:
  3 equiprobable possibilities for the next pair.



Arc-partitions

X to be partitioned into
$Y = \{y_1, \ldots, y_{n/2}\}$ and $Z = \{z_1, \ldots, z_{n/2}\}$.

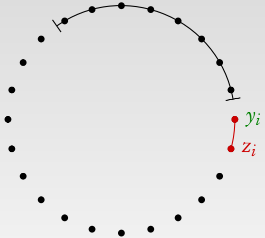
Arc-partitions

X to be partitioned into
$Y = \{y_1, \ldots, y_{n/2}\}$ and $Z = \{z_1, \ldots, z_{n/2}\}$.

Definition of a random arc-partition:

from a random pairing $P = P_1, \ldots, P_{n/2}$, if $P_i = \{j, k\}$ then

- with probability $1/2$, x_j is mapped to y_i and x_k to z_i;
- with probability $1/2$, x_j is mapped to z_i and x_k to y_i.



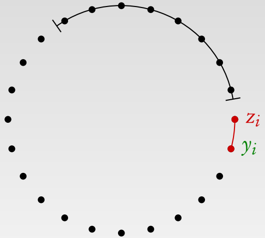
Arc-partitions

X to be partitioned into
$Y = \{y_1, \ldots, y_{n/2}\}$ and $Z = \{z_1, \ldots, z_{n/2}\}$.

Definition of a random arc-partition:

from a random pairing $P = P_1, \ldots, P_{n/2}$, if $P_i = \{j, k\}$ then

- with probability $1/2$, x_j is mapped to y_i and x_k to z_i;
- with probability $1/2$, x_j is mapped to z_i and x_k to y_i.



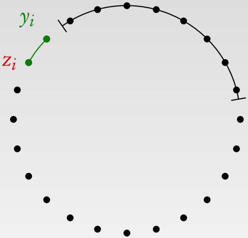
Arc-partitions

X to be partitioned into
$Y = \{y_1, \ldots, y_{n/2}\}$ and $Z = \{z_1, \ldots, z_{n/2}\}$.

Definition of a random arc-partition:

from a random pairing $P = P_1, \ldots, P_{n/2}$, if $P_i = \{j, k\}$ then

- with probability $1/2$, x_j is mapped to y_i and x_k to z_i;
- with probability $1/2$, x_j is mapped to z_i and x_k to y_i.



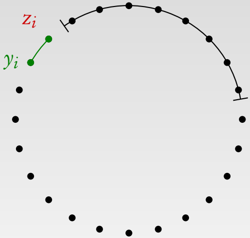
Arc-partitions

X to be partitioned into
$Y = \{y_1, \ldots, y_{n/2}\}$ and $Z = \{z_1, \ldots, z_{n/2}\}$.

Definition of a random arc-partition:

from a random pairing $P = P_1, \ldots, P_{n/2}$, if $P_i = \{j, k\}$ then

- with probability $1/2$, x_j is mapped to y_i and x_k to z_i;
- with probability $1/2$, x_j is mapped to z_i and x_k to y_i.



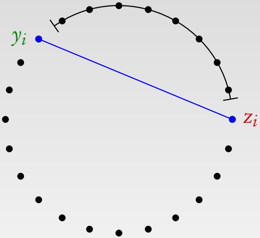
Arc-partitions

X to be partitioned into
$Y = \{y_1, \ldots, y_{n/2}\}$ and $Z = \{z_1, \ldots, z_{n/2}\}$.

Definition of a random arc-partition:

from a random pairing $P = P_1, \ldots, P_{n/2}$, if $P_i = \{j, k\}$ then

- with probability $1/2$, x_j is mapped to y_i and x_k to z_i;
- with probability $1/2$, x_j is mapped to z_i and x_k to y_i.



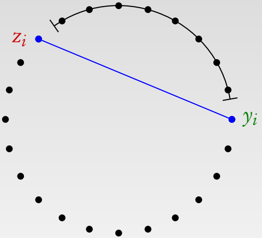
Arc-partitions

X to be partitioned into
$Y = \{y_1, \ldots, y_{n/2}\}$ and $Z = \{z_1, \ldots, z_{n/2}\}$.

Definition of a random arc-partition:

from a random pairing $P = P_1, \ldots, P_{n/2}$, if $P_i = \{j, k\}$ then

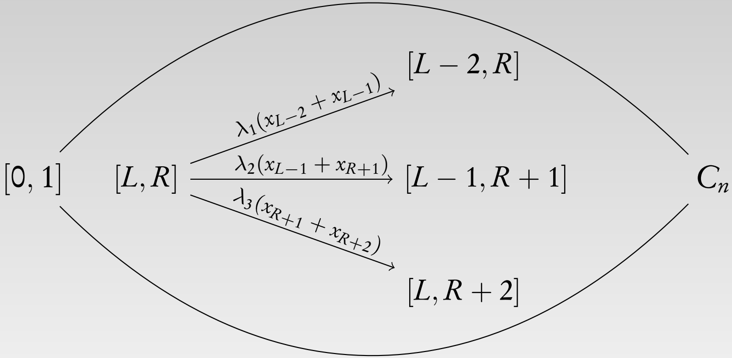
- with probability $1/2$, x_j is mapped to y_i and x_k to z_i;
- with probability $1/2$, x_j is mapped to z_i and x_k to y_i.



The branching program

The ABP is built according to the iterative process of pairing:

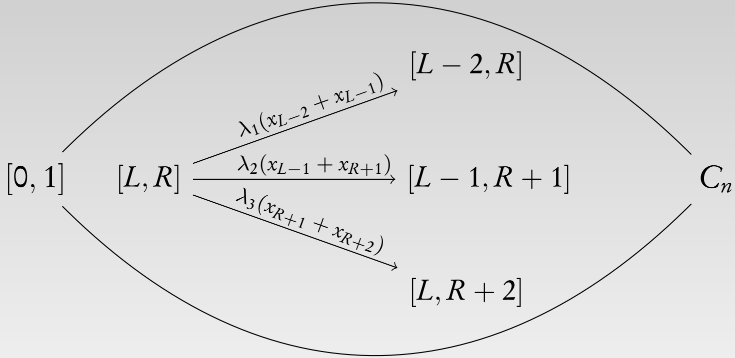
- vertices = arcs $[L, R]$ of the pairing
- start node $[0, 1]$, end node C_n
- one path = one pairing



The branching program

The ABP is built according to the iterative process of pairing:

- vertices = arcs $[L, R]$ of the pairing
- start node $[0, 1]$, end node C_n
- one path = one pairing



Independent paths, two choices per edge $\rightarrow$ full rank.

(K, T)-products

Definition

A polynomial $g(x_1, \ldots, x_n)$ is a (K, T)-product if $g = g_1 g_2 \cdots g_K$ where:

- g_i is on the set of variables X_i;
- $X_1, \ldots, X_K$ are pairwise disjoint;
- and $|X_i| \geq T$.

(K, T)-products

Definition

A polynomial $g(x_1, \ldots, x_n)$ is a (K, T)-product if $g = g_1 g_2 \cdots g_K$ where:

- g_i is on the set of variables X_i;
- $X_1, \ldots, X_K$ are pairwise disjoint;
- and $|X_i| \geq T$.

Example: $(x_1 x_2 - 3x_1)(x_3 + 1)(5x_5 x_6 - x_6)$

is a $(3, 2)$-product.

Restricting to (K, T)-products

Lemma (Shpilka&Yehudayoff)

If $f(x_1, \ldots, x_n)$ is computed by a formula of size s, then $f = f_1 + \cdots + f_{s+1}$ where f_i is a $(\frac{\log n}{100}, n^{7/8})$-product.

Restricting to (K, T)-products

Lemma (Shpilka&Yehudayoff)

If $f(x_1, \ldots, x_n)$ is computed by a formula of size s, then $f = f_1 + \cdots + f_{s+1}$ where f_i is a $(\frac{\log n}{100}, n^{7/8})$-product.

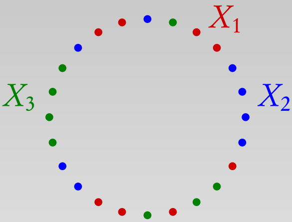
Since $\mathrm{rank}(M((f_i + f_j)_\Pi)) \leq \mathrm{rank}(M((f_i)_\Pi)) + \mathrm{rank}(M((f_j)_\Pi))$,

we restrict the study to one (K, T)-product $g = g_1 g_2 \cdots g_K$

$\rightarrow$ we must argue that g is low rank
(instead of only "not full rank")

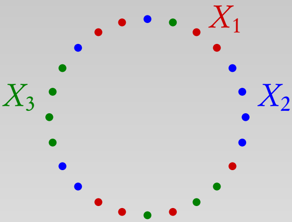
Combinatorics

- K disjoint subsets of the variables = K colors.



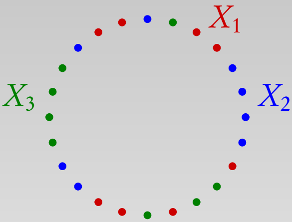
Combinatorics

- K disjoint subsets of the variables = K colors.
- $\text{rank}(M(g_\Pi)) = \prod_i \text{rank}(M((g_i)_\Pi))$
  $\rightarrow g$ is low rank if
  one of the g_i is low rank.



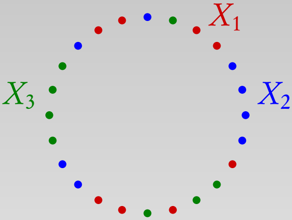
Combinatorics

- K disjoint subsets of the variables = K colors.
- $\mathrm{rank}(M(g_\Pi)) = \prod_i \mathrm{rank}(M((g_i)_\Pi))$
  $\rightarrow g$ is low rank if
  one of the g_i is low rank.
- Since $\mathrm{rank}(M((g_i)_\Pi)) \leq 2^{\min(Y(g_i), Z(g_i))}$,
  it suffices that some color has
  much more Y than Z variables.



Combinatorics

- K disjoint subsets of the variables = K colors.
- $\text{rank}(M(g_\Pi)) = \prod_i \text{rank}(M((g_i)_\Pi))$ $\rightarrow g$ is low rank if one of the g_i is low rank.
- Since $\text{rank}(M((g_i)_\Pi)) \leq 2^{\min(Y(g_i),Z(g_i))}$, it suffices that some color has much more Y than Z variables.



From now on the argument is only combinatorial.

Unbalanced colors

Definition

For a given partition Π, a color is balanced if it has roughly the same number of Y and Z variables.

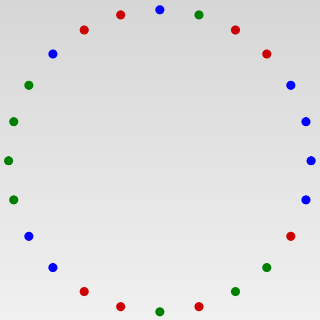
$\rightarrow$ We want to show that some colors are unbalanced.

Unbalanced colors

Definition

For a given partition Π, a color is balanced if it has roughly the same number of Y and Z variables.

$\rightarrow$ We want to show that some colors are unbalanced.

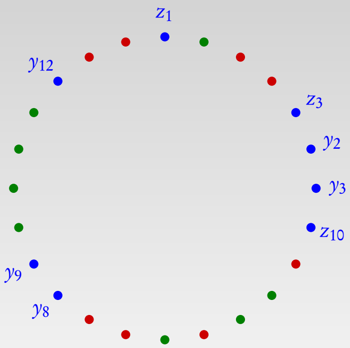


Unbalanced colors

Definition

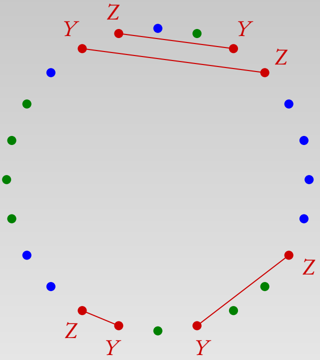
For a given partition Π, a color is balanced if it has roughly the same number of Y and Z variables.

$\rightarrow$ We want to show that some colors are unbalanced.



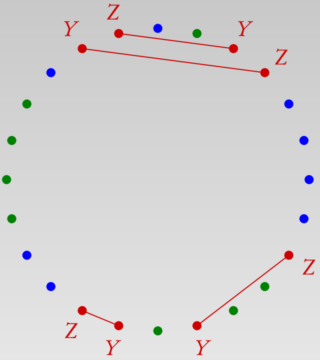
A sufficient condition of balance

If all pairs containing a red vertex has its other vertex red, then color red is balanced.



A sufficient condition of balance

If all pairs containing a red vertex has its other vertex red, then color red is balanced.

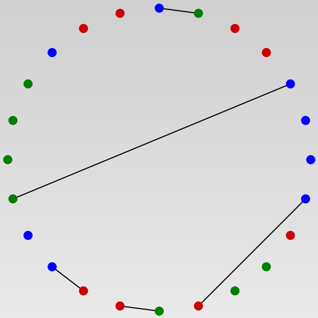


$\rightarrow$ look for pairs whose vertices have different colors

= "violations"

Violations

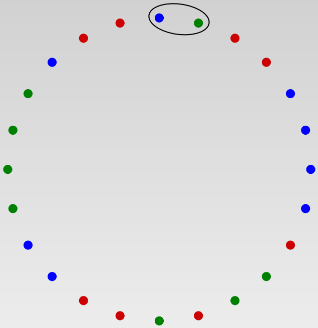
Examples of violations in a pairing:



Jumps

Violation for consecutive vertices = "jump"

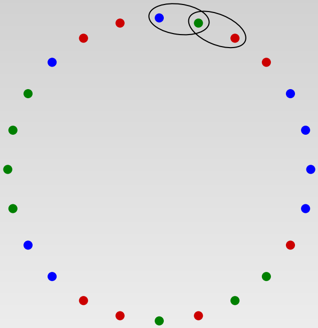
(=change of color)



Jumps

Violation for consecutive vertices = “jump”

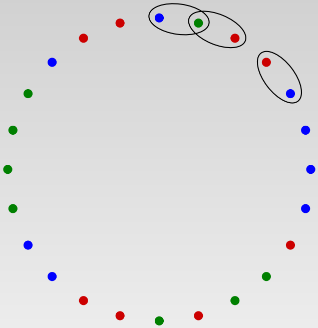
(=change of color)



Jumps

Violation for consecutive vertices = “jump”

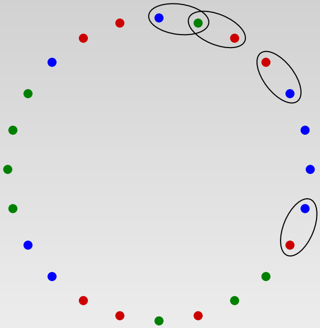
(=change of color)



Jumps

Violation for consecutive vertices = “jump”

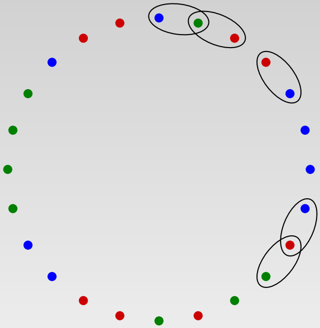
(=change of color)



Jumps

Violation for consecutive vertices = “jump”

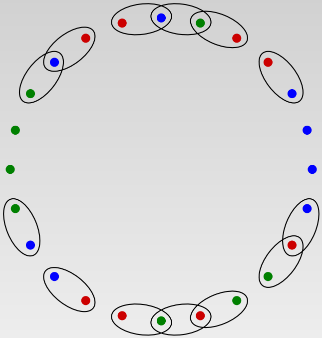
(= change of color)



Jumps

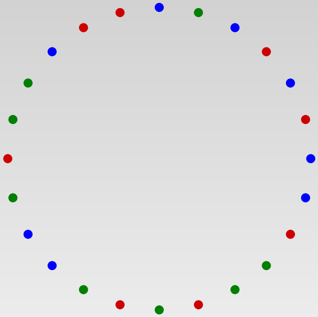
Violation for consecutive vertices = "jump"

(=change of color)



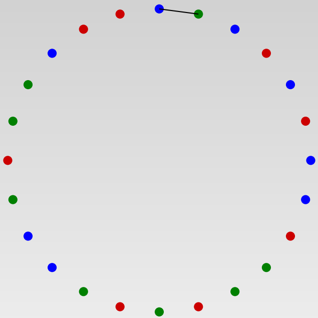
First case: Many colors with many jumps

- If $[R, R+1]$ is a jump,
  it is chosen in the pairing with probability 1/3.
- Many jumps
  $\rightarrow$ one third of them yield violations.



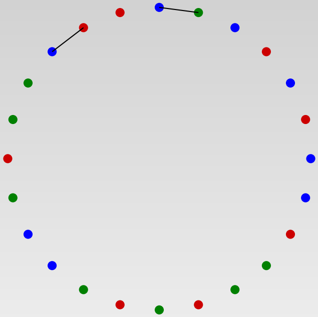
First case: Many colors with many jumps

- If $[R, R+1]$ is a jump,
  it is chosen in the pairing with probability 1/3.
- Many jumps
  $\rightarrow$ one third of them yield violations.



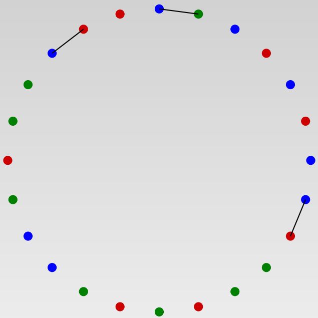
First case: Many colors with many jumps

- If $[R, R+1]$ is a jump,
  it is chosen in the pairing with probability $1/3$.
- Many jumps
  $\rightarrow$ one third of them yield violations.



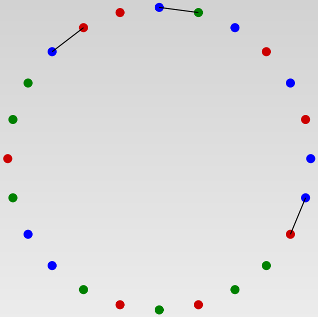
First case: Many colors with many jumps

- If $[R, R+1]$ is a jump,
  it is chosen in the pairing with probability 1/3.
- Many jumps
  $\rightarrow$ one third of them yield violations.



First case: Many colors with many jumps

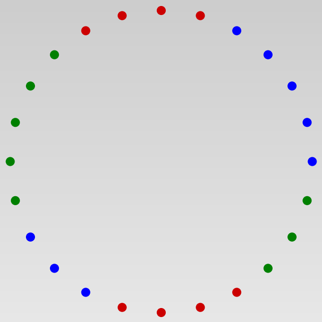
- If $[R, R+1]$ is a jump,
  it is chosen in the pairing with probability 1/3.
- Many jumps
  $\rightarrow$ one third of them yield violations.



$\rightarrow$ the color is unbalanced with sufficiently high probability.

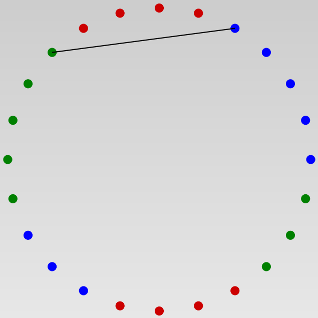
Second case: Many colors with few jumps

- Large monochromatic arcs.
- $\rightarrow$ A large number of cords give violations since each violating cord is chosen with probability 1/3.



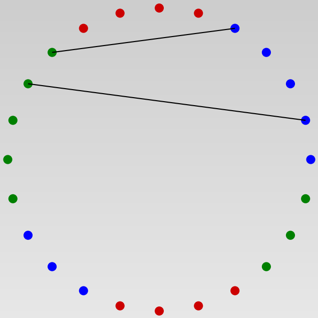
Second case: Many colors with few jumps

- Large monochromatic arcs.
- $\rightarrow$ A large number of cords give violations
  since each violating cord is chosen with probability 1/3.



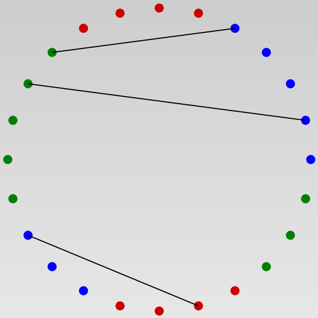
Second case: Many colors with few jumps

- Large monochromatic arcs.
- $\rightarrow$ A large number of cords give violations
  since each violating cord is chosen with probability 1/3.



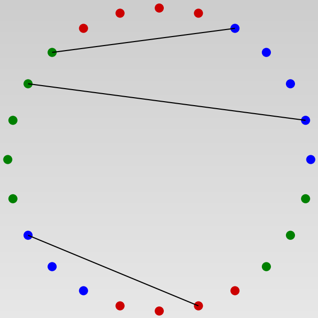
Second case: Many colors with few jumps

- Large monochromatic arcs.
- $\rightarrow$ A large number of cords give violations
  since each violating cord is chosen with probability $1/3$.



Second case: Many colors with few jumps

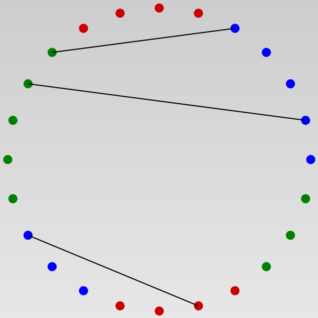
- Large monochromatic arcs.
- $\rightarrow$ A large number of cords give violations
  since each violating cord is chosen with probability $1/3$.



$\rightarrow$ the color is unbalanced with sufficiently high probability.

Second case: Many colors with few jumps

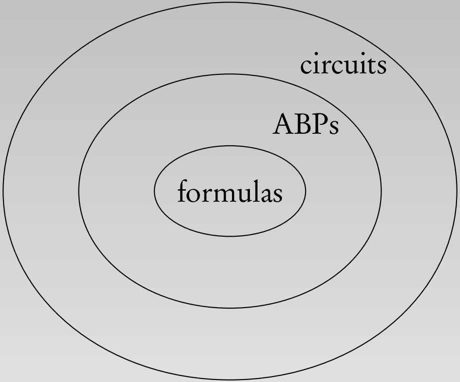
- Large monochromatic arcs.
- $\rightarrow$ A large number of cords give violations
  since each violating cord is chosen with probability 1/3.



$\rightarrow$ the color is unbalanced with sufficiently high probability.

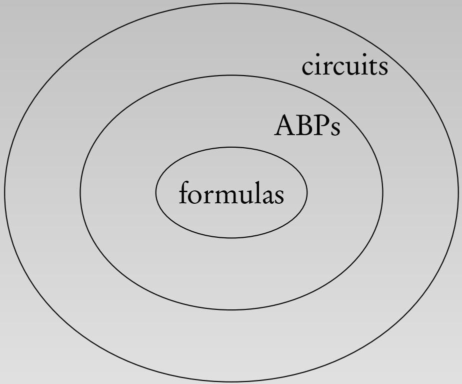
(Formal analysis = 2D random walk on a chessboard.)

Future directions



- Separate multilinear circuits and ABPs?

Future directions



- Separate multilinear circuits and ABPs?
- Are there polynomial-size multilinear ABPs for the determinant?