Branching Program size lower bounds via Projective Dimension

Sajin Koroth
(joint work with Krishnamoorthy Dinesh and Jayalal Sarma)

Indian Institute of Technology, Madras

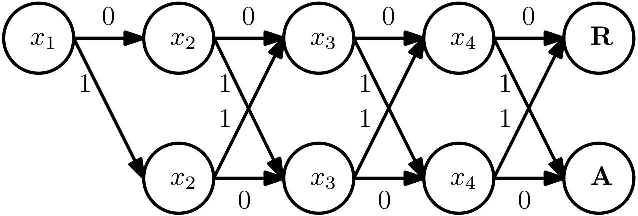
Theory Lunch, Technion

Outline

1. Branching Programs - model and motivation
2. Projective Dimension and BP size lower bounds
3. Gap Between Projective Dimension and BP Size
4. Bridging the Gap : Bitwise Projective Dimension
5. A lower bound for Bitwise Projective Dimension that matches state of the art Branching program lower bound
6. Discussions and Future Work

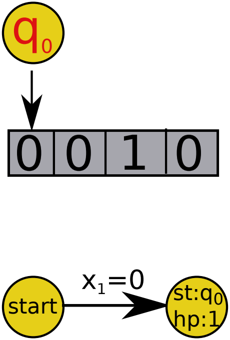
Branching Programs

- Directed Acyclic Graphs with designated start, accept and reject nodes
- Each node queries a variable
- Edges emanating out of a node are labeled by the bit value of the variable queried by the variable

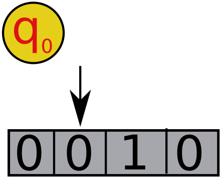


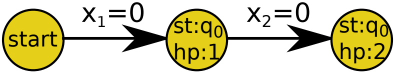
$\mathsf{PARITY}_4 = x_1 \oplus x_2 \oplus x_3 \oplus x_4$

Connection to space bounded Turing Machines

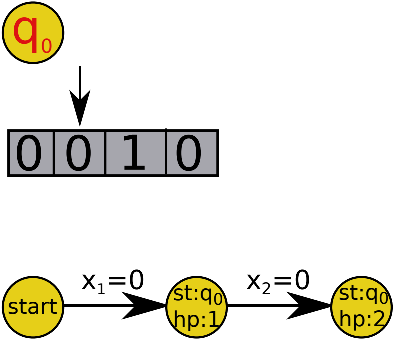


Connection to space bounded Turing Machines

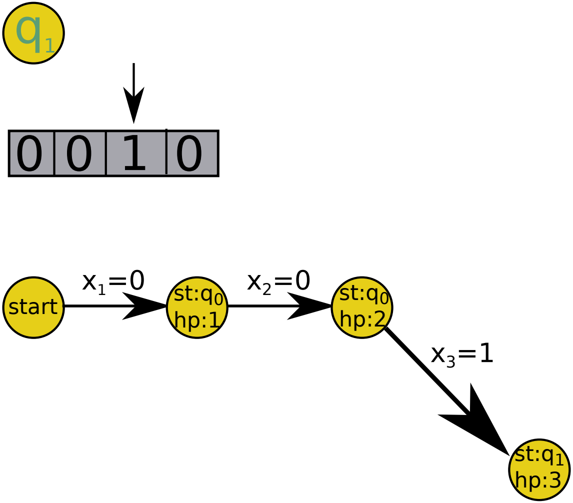




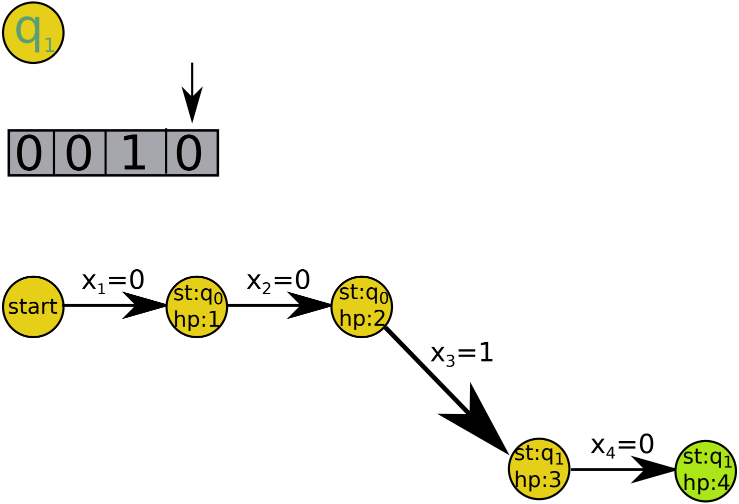
Connection to space bounded Turing Machines



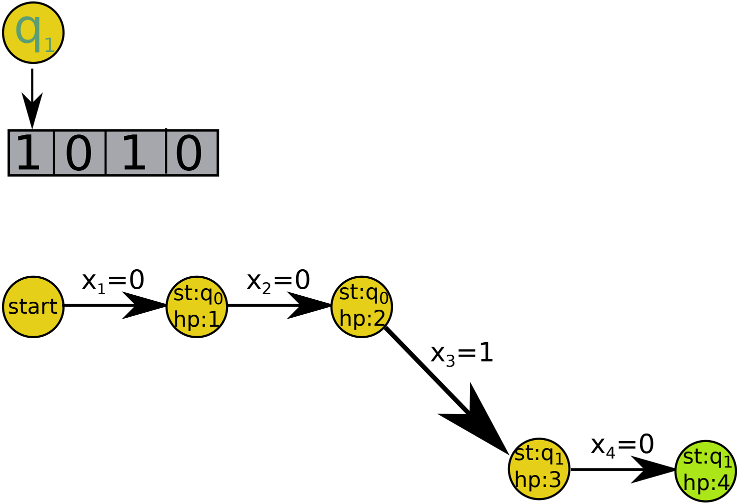
Connection to space bounded Turing Machines



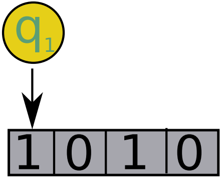
Connection to space bounded Turing Machines

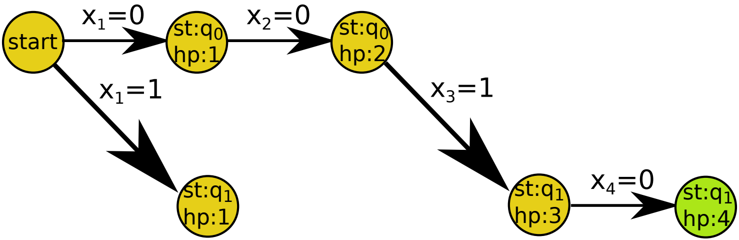


Connection to space bounded Turing Machines

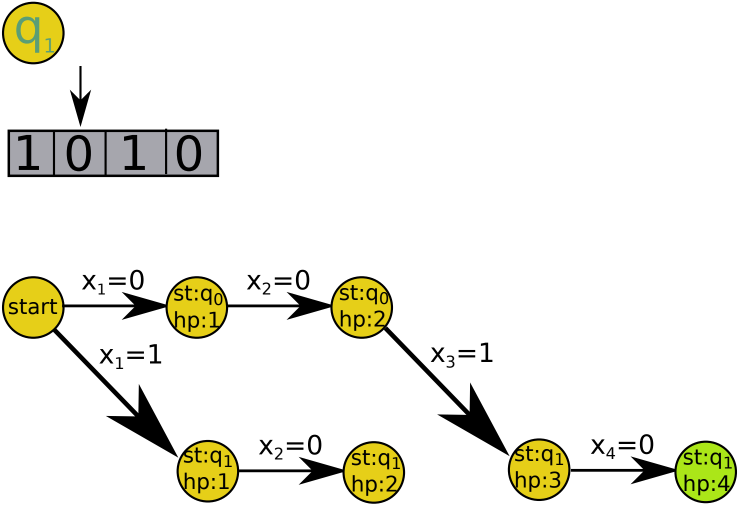


Connection to space bounded Turing Machines

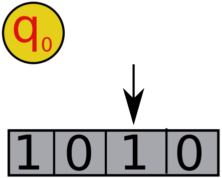


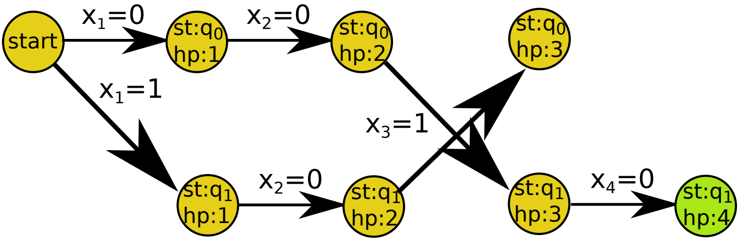


Connection to space bounded Turing Machines

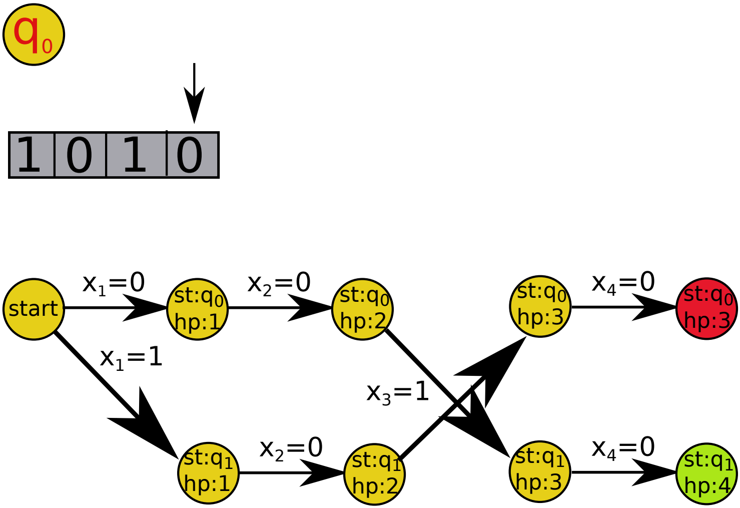


Connection to space bounded Turing Machines

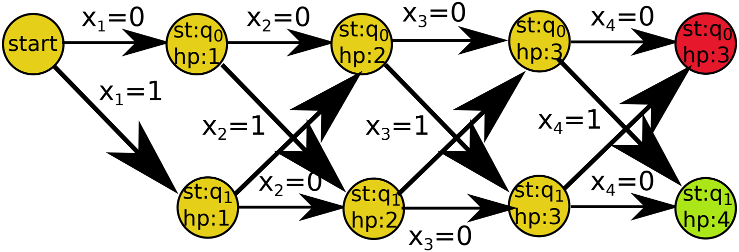




Connection to space bounded Turing Machines

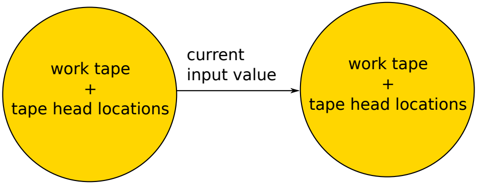


Connection to space bounded Turing Machines



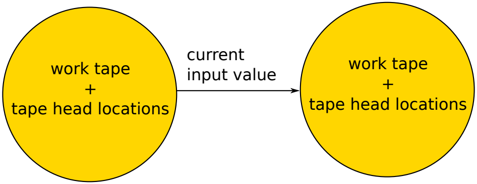
Is $\mathbf{L} \neq \mathbf{P}$

- For every TM with space bound S there is a Deterministic Branching Program with size $2^{O(S)}$
- Thus to prove that **L** the class of log-space solvable problems is separate from **P** the class of polynomial time solvable problems, its enough to prove a super-polynomial size lower bound for BP's



Is $\mathbf{L} \neq \mathbf{P}$

- For every TM with space bound S there is a Deterministic Branching Program with size $2^{O(S)}$
- Thus to prove that **L** the class of log-space solvable problems is separate from **P** the class of polynomial time solvable problems, its enough to prove a super-polynomial size lower bound for BP's



State of the art of BP size lower bounds

- For deterministic branching programs it is $n^2/\log^2 n$ by Nechiporuk **from 60's**
- Nechiporuk's method applies for many functions. We consider the **Element Distinctness** function
  - $\mathrm{ED}_m : \{0,1\}^{n=m2\log m} \to \{0,1\}$
  - m inputs $x_1,\ldots,x_m$ each representing a number in $[m^2]$
  - $f(x_1,\ldots,x_m) = 1$ iff no two x_i, x_j are equal
- Let there be a size S branching program computing ED_n. Let S_i be the number of nodes in the BP which queries a bit from x_i (x_i is a $2\log m$ bit input).
- The number of different branching programs on S_i nodes is at most $2^{3S_i \log S_i}$
- $\mathrm{ED}_4(1,*,3,4) \not\equiv \mathrm{ED}_4(1,*,2,3)$. There are $2^{\Omega(n)}$ restrictions which give different restrictions of ED_n for each $i \in [m]$.
- For every i, $2^{3S_i \log S_i} \geq 2^{\Omega(n)}$, that is $S_i = \Omega(n/\log n)$
- $S = \sum_{i=1}^{m=n/\log n} S_i = \Omega(n^2/\log^2 n)$.

State of the art of BP size lower bounds

- For deterministic branching programs it is $n^2/\log^2 n$ by Nechiporuk **from 60's**
- Nechiporuk's method applies for many functions. We consider the **Element Distinctness** function
  - $\mathrm{ED}_m : \{0,1\}^{n=m2\log m} \to \{0,1\}$
  - m inputs $x_1,\ldots,x_m$ each representing a number in $[m^2]$
  - $f(x_1,\ldots,x_m) = 1$ iff no two x_i, x_j are equal
- Let there be a size S branching program computing ED_n. Let S_i be the number of nodes in the BP which queries a bit from x_i (x_i is a $2\log m$ bit input).
- The number of different branching programs on S_i nodes is at most $2^{3S_i \log S_i}$
- $\mathrm{ED}_4(1,*,3,4) \not\equiv \mathrm{ED}_4(1,*,2,3)$. There are $2^{\Omega(n)}$ restrictions which give different restrictions of ED_n for each $i \in [m]$.
- For every i, $2^{3S_i \log S_i} \geq 2^{\Omega(n)}$, that is $S_i = \Omega(n/\log n)$
- $S = \sum_{i=1}^{m=n/\log n} S_i = \Omega(n^2/\log^2 n)$.

State of the art of BP size lower bounds

- For deterministic branching programs it is $n^2/\log^2 n$ by Nechiporuk **from 60's**
- Nechiporuk's method applies for many functions. We consider the **Element Distinctness** function
  - $\mathrm{ED}_m : \{0,1\}^{n=m2\log m} \to \{0,1\}$
  - m inputs $x_1, \ldots, x_m$ each representing a number in $[m^2]$
  - $f(x_1, \ldots, x_m) = 1$ iff no two x_i, x_j are equal
- Let there be a size S branching program computing ED_n. Let S_i be the number of nodes in the BP which queries a bit from x_i (x_i is a $2\log m$ bit input).
- The number of different branching programs on S_i nodes is at most $2^{3S_i \log S_i}$
- $\mathrm{ED}_4(1,*,3,4) \not\equiv \mathrm{ED}_4(1,*,2,3)$. There are $2^{\Omega(n)}$ restrictions which give different restrictions of ED_n for each $i \in [m]$.
- For every i, $2^{3S_i \log S_i} \geq 2^{\Omega(n)}$, that is $S_i = \Omega(n/\log n)$
- $S = \sum_{i=1}^{m=n/\log n} S_i = \Omega(n^2/\log^2 n)$.

State of the art of BP size lower bounds

- For deterministic branching programs it is $n^2/\log^2 n$ by Nechiporuk **from 60's**
- Nechiporuk's method applies for many functions. We consider the **Element Distinctness** function
  - $\mathrm{ED}_m : \{0,1\}^{n=m2\log m} \to \{0,1\}$
  - m inputs $x_1, \ldots, x_m$ each representing a number in $[m^2]$
  - $f(x_1, \ldots, x_m) = 1$ iff no two x_i, x_j are equal
- Let there be a size S branching program computing ED_n. Let S_i be the number of nodes in the BP which queries a bit from x_i (x_i is a $2\log m$ bit input).
- The number of different branching programs on S_i nodes is at most $2^{3S_i \log S_i}$
- $\mathrm{ED}_4(1,*,3,4) \not\equiv \mathrm{ED}_4(1,*,2,3)$. There are $2^{\Omega(n)}$ restrictions which give different restrictions of ED_n for each $i \in [m]$.
- For every i, $2^{3S_i \log S_i} \geq 2^{\Omega(n)}$, that is $S_i = \Omega(n/\log n)$
- $S = \sum_{i=1}^{m=n/\log n} S_i = \Omega(n^2/\log^2 n)$.

State of the art of BP size lower bounds

- For deterministic branching programs it is $n^2/\log^2 n$ by Nechiporuk **from 60's**
- Nechiporuk's method applies for many functions. We consider the **Element Distinctness** function
  - $\mathrm{ED}_m : \{0,1\}^{n=m2\log m} \to \{0,1\}$
  - m inputs $x_1,\ldots,x_m$ each representing a number in $[m^2]$
  - $f(x_1,\ldots,x_m) = 1$ iff no two x_i, x_j are equal
- Let there be a size S branching program computing ED_n. Let S_i be the number of nodes in the BP which queries a bit from x_i (x_i is a $2\log m$ bit input).
- The number of different branching programs on S_i nodes is at most $2^{3S_i \log S_i}$
- $\mathrm{ED}_4(1,*,3,4) \not\equiv \mathrm{ED}_4(1,*,2,3)$. There are $2^{\Omega(n)}$ restrictions which give different restrictions of ED_n for each $i \in [m]$.
- For every i, $2^{3S_i \log S_i} \geq 2^{\Omega(n)}$, that is $S_i = \Omega(n/\log n)$
- $S = \sum_{i=1}^{m=n/\log n} S_i = \Omega(n^2/\log^2 n)$.

State of the art of BP size lower bounds

- For deterministic branching programs it is $n^2/\log^2 n$ by Nechiporuk **from 60's**
- Nechiporuk's method applies for many functions. We consider the **Element Distinctness** function
  - $\mathrm{ED}_m : \{0,1\}^{n=m2\log m} \to \{0,1\}$
  - m inputs $x_1,\ldots,x_m$ each representing a number in $[m^2]$
  - $f(x_1,\ldots,x_m) = 1$ iff no two x_i, x_j are equal
- Let there be a size S branching program computing ED_n. Let S_i be the number of nodes in the BP which queries a bit from x_i (x_i is a $2\log m$ bit input).
- The number of different branching programs on S_i nodes is at most $2^{3S_i \log S_i}$
- $\mathrm{ED}_4(1,*,3,4) \not\equiv \mathrm{ED}_4(1,*,2,3)$. There are $2^{\Omega(n)}$ restrictions which give different restrictions of ED_n for each $i \in [m]$.
- For every i, $2^{3S_i \log S_i} \geq 2^{\Omega(n)}$, that is $S_i = \Omega(n/\log n)$
- $S = \sum_{i=1}^{m=n/\log n} S_i = \Omega(n^2/\log^2 n)$.

State of the art of BP size lower bounds

- For deterministic branching programs it is $n^2/\log^2 n$ by Nechiporuk **from 60's**
- Nechiporuk's method applies for many functions. We consider the **Element Distinctness** function
  - $\mathrm{ED}_m : \{0,1\}^{n=m2\log m} \to \{0,1\}$
  - m inputs $x_1,\ldots,x_m$ each representing a number in $[m^2]$
  - $f(x_1,\ldots,x_m) = 1$ iff no two x_i, x_j are equal
- Let there be a size S branching program computing ED_n. Let S_i be the number of nodes in the BP which queries a bit from x_i (x_i is a $2\log m$ bit input).
- The number of different branching programs on S_i nodes is at most $2^{3S_i \log S_i}$
- $\mathrm{ED}_4(1,*,3,4) \not\equiv \mathrm{ED}_4(1,*,2,3)$. There are $2^{\Omega(n)}$ restrictions which give different restrictions of ED_n for each $i \in [m]$.
- For every i, $2^{3S_i \log S_i} \geq 2^{\Omega(n)}$, that is $S_i = \Omega(n/\log n)$
- $S = \sum_{i=1}^{m=n/\log n} S_i = \Omega(n^2/\log^2 n)$.

State of the art of BP size lower bounds

- For deterministic branching programs it is $n^2/\log^2 n$ by Nechiporuk **from 60's**
- Nechiporuk's method applies for many functions. We consider the **Element Distinctness** function
  - $\mathrm{ED}_m : \{0,1\}^{n=m2\log m} \to \{0,1\}$
  - m inputs $x_1, \ldots, x_m$ each representing a number in $[m^2]$
  - $f(x_1, \ldots, x_m) = 1$ iff no two x_i, x_j are equal
- Let there be a size S branching program computing ED_n. Let S_i be the number of nodes in the BP which queries a bit from x_i (x_i is a $2\log m$ bit input).
- The number of different branching programs on S_i nodes is at most $2^{3S_i \log S_i}$
- $\mathrm{ED}_4(1,*,3,4) \not\equiv \mathrm{ED}_4(1,*,2,3)$. There are $2^{\Omega(n)}$ restrictions which give different restrictions of ED_n for each $i \in [m]$.
- For every i, $2^{3S_i \log S_i} \geq 2^{\Omega(n)}$, that is $S_i = \Omega(n/\log n)$
- $S = \sum_{i=1}^{m=n/\log n} S_i = \Omega(n^2/\log^2 n)$.

State of the art of BP size lower bounds

- For deterministic branching programs it is $n^2/\log^2 n$ by Nechiporuk **from 60's**
- Nechiporuk's method applies for many functions. We consider the **Element Distinctness** function
  - $\mathrm{ED}_m : \{0,1\}^{n=m2\log m} \to \{0,1\}$
  - m inputs $x_1, \ldots, x_m$ each representing a number in $[m^2]$
  - $f(x_1, \ldots, x_m) = 1$ iff no two x_i, x_j are equal
- Let there be a size S branching program computing ED_n. Let S_i be the number of nodes in the BP which queries a bit from x_i (x_i is a $2\log m$ bit input).
- The number of different branching programs on S_i nodes is at most $2^{3S_i \log S_i}$
- $\mathrm{ED}_4(1,*,3,4) \not\equiv \mathrm{ED}_4(1,*,2,3)$. There are $2^{\Omega(n)}$ restrictions which give different restrictions of ED_n for each $i \in [m]$.
- For every i, $2^{3S_i \log S_i} \geq 2^{\Omega(n)}$, that is $S_i = \Omega(n/\log n)$
- $S = \sum_{i=1}^{m=n/\log n} S_i = \Omega(n^2/\log^2 n)$.

State of the art of BP size lower bounds

- For deterministic branching programs it is $n^2/\log^2 n$ by Nechiporuk **from 60's**
- Nechiporuk's method applies for many functions. We consider the **Element Distinctness** function
  - $\mathrm{ED}_m : \{0,1\}^{n=m2\log m} \to \{0,1\}$
  - m inputs $x_1,\dots,x_m$ each representing a number in $[m^2]$
  - $f(x_1,\dots,x_m) = 1$ iff no two x_i, x_j are equal
- Let there be a size S branching program computing ED_n. Let S_i be the number of nodes in the BP which queries a bit from x_i (x_i is a $2\log m$ bit input).
- The number of different branching programs on S_i nodes is at most $2^{3S_i \log S_i}$
- $\mathrm{ED}_4(1,*,3,4) \not\equiv \mathrm{ED}_4(1,*,2,3)$. There are $2^{\Omega(n)}$ restrictions which give different restrictions of ED_n for each $i \in [m]$.
- For every i, $2^{3S_i \log S_i} \geq 2^{\Omega(n)}$, that is $S_i = \Omega(n/\log n)$
- $S = \sum_{i=1}^{m=n/\log n} S_i = \Omega(n^2/\log^2 n)$.

Projective Dimension

- Measure on bipartite graphs introduced by Pudlak and Rodl
- Graph $G(U, V, E)$. Assign subspaces from $\mathbb{F}^d$ to vertices so that

$$(x, y) \in E \iff \phi(x) \cap \phi(y) \neq \{0\}$$

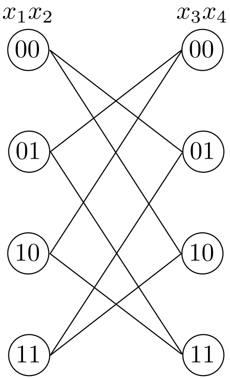
- Smallest such d : $\mathrm{pd}_{\mathbb{F}}(G)$.

Projective Dimension

- Measure on bipartite graphs introduced by Pudlak and Rodl
- Graph $G(U, V, E)$. Assign subspaces from $\mathbb{F}^d$ to vertices so that

$$(x, y) \in E \iff \phi(x) \cap \phi(y) \neq \{0\}$$

- Smallest such d : $\mathrm{pd}_{\mathbb{F}}(G)$.

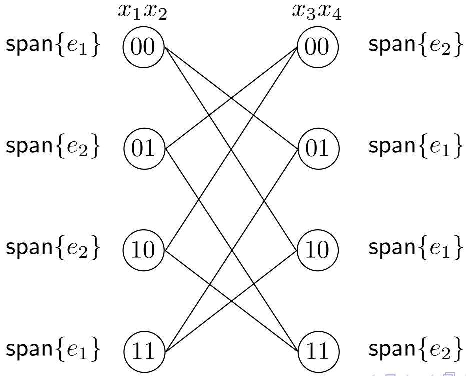


Projective Dimension

- Measure on bipartite graphs introduced by Pudlak and Rodl
- Graph $G(U, V, E)$. Assign subspaces from $\mathbb{F}^d$ to vertices so that

$$(x,y) \in E \iff \phi(x) \cap \phi(y) \neq \{0\}$$

- Smallest such d : $\mathrm{pd}_{\mathbb{F}}(G)$.



$\text{bpsize}(f) \geq \text{pd}(f)$

Theorem, (Pudlak and Rodl (1992))

Over any $\mathbb{F}$, $\text{bpsize}(f) \geq \text{pd}_{\mathbb{F}}(G_f)$.

- To define the bipartite graph G_f associated with a function f on $2n$ variables, take some natural partition of the variable set into two equal parts

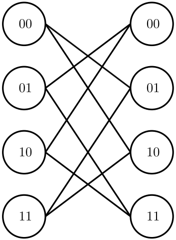
$\text{bpsize}(f) \geq \text{pd}(f)$

Theorem, (Pudlak and Rodl (1992))

Over any $\mathbb{F}$, $\text{bpsize}(f) \geq \text{pd}_{\mathbb{F}}(G_f)$.

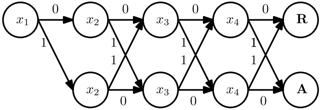
- To define the bipartite graph G_f associated with a function f on $2n$ variables, take some natural partition of the variable set into two equal parts

Proof of the Pudalk Rodl theorem



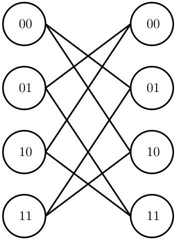
G_f

$f(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$



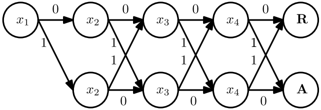
Branching program computing $f = \mathsf{PARITY}_4$

Proof of the Pudalk Rodl theorem

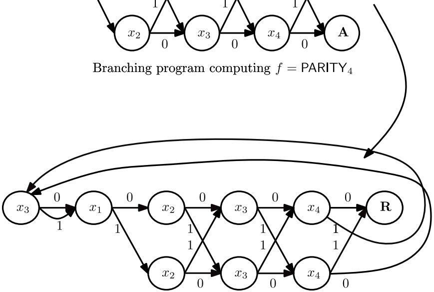


G_f

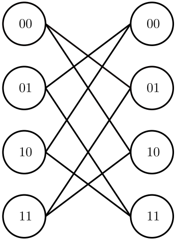
$f(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$



Branching program computing $f = \mathsf{PARITY}_4$

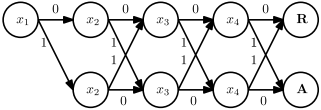


Proof of the Pudalk Rodl theorem

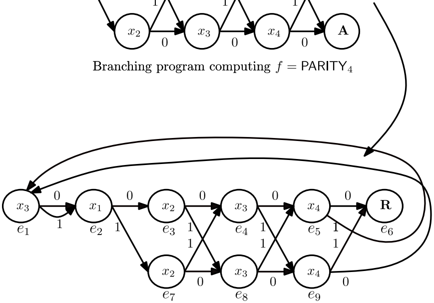


G_f

$f(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$



Branching program computing $f = \mathsf{PARITY}_4$

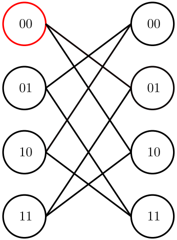


Modified graph giving subspace assignment for G_f

Proof of the Pudalk Rodl theorem

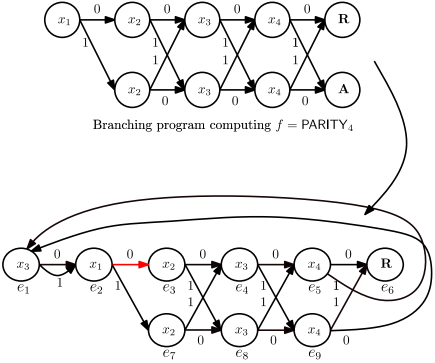
$\{e_2 - e_3,$

$\}$



G_f

$f(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$

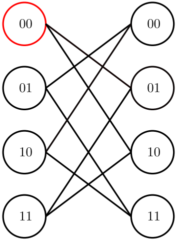


Branching program computing $f = \mathsf{PARITY}_4$

Modified graph giving subspace assignment for G_f

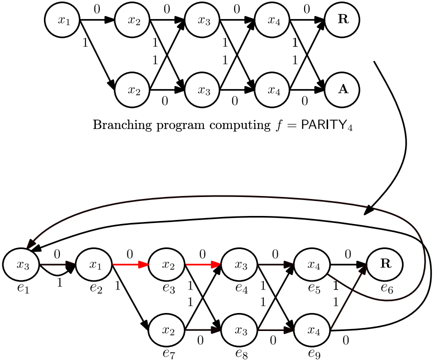
Proof of the Pudalk Rodl theorem

$\{e_2 - e_3,$
$e_3 - e_4 \quad \}$



G_f

$f(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$

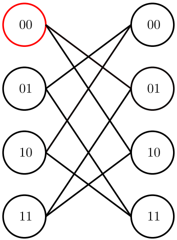


Branching program computing $f = \mathsf{PARITY}_4$

Modified graph giving subspace assignment for G_f

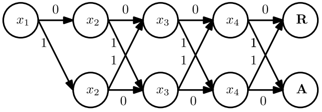
Proof of the Pudalk Rodl theorem

$\{e_2 - e_3,$
$e_3 - e_4, e_7 - e_8\}$

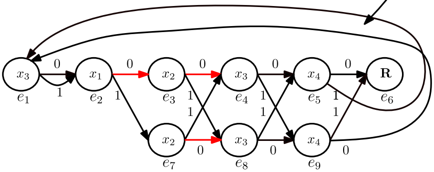


G_f

$f(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$

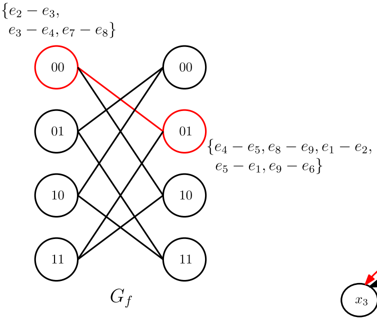


Branching program computing $f = \mathsf{PARITY}_4$

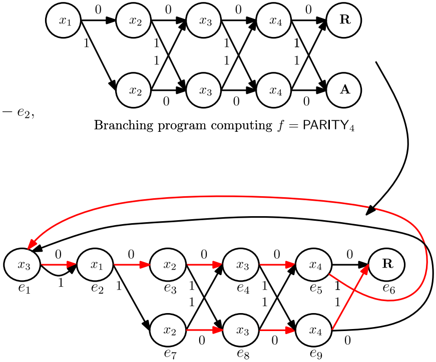


Modified graph giving subspace assignment for G_f

Proof of the Pudalk Rodl theorem



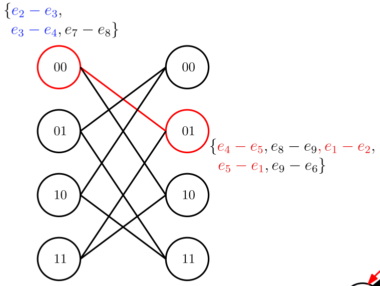
$f(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$



Branching program computing $f = \mathsf{PARITY}_4$

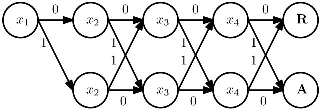
Modified graph giving subspace assignment for G_f

Proof of the Pudalk Rodl theorem

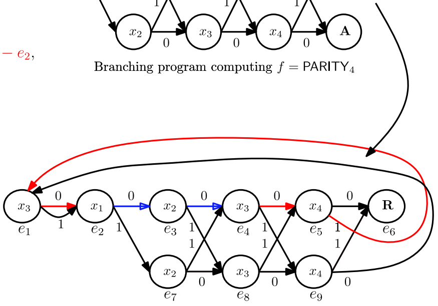


G_f

$f(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$



Branching program computing $f = \mathsf{PARITY}_4$



Modified graph giving subspace assignment for G_f

Proof contd.

- Let (x, y) be an input. And H_x be the edge-subgraph of the branching program whose edges query variables in x. Similarly define H_y.
- After the transformation $f(x, y) = 1$ iff $H_x \cup H_y$ contains a cycle
- Make sure that for any (x, y) s.t. $f(x, y) = 1$ this unique cycle has edges from both H_x and H_y.
- Any linear dependence in $span\{\phi(x), \phi(y)\}$ corresponds to a cycle in $H_x \cup H_y$

Proof contd.

- Let (x, y) be an input. And H_x be the edge-subgraph of the branching program whose edges query variables in x. Similarly define H_y.
- After the transformation $f(x, y) = 1$ iff $H_x \cup H_y$ contains a cycle
- Make sure that for any (x, y) s.t. $f(x, y) = 1$ this unique cycle has edges from both H_x and H_y.
- Any linear dependence in $span\{\phi(x), \phi(y)\}$ corresponds to a cycle in $H_x \cup H_y$

Proof contd.

- Let (x, y) be an input. And H_x be the edge-subgraph of the branching program whose edges query variables in x. Similarly define H_y.
- After the transformation $f(x, y) = 1$ iff $H_x \cup H_y$ contains a cycle
- Make sure that for any (x, y) s.t. $f(x, y) = 1$ this unique cycle has edges from both H_x and H_y.
- Any linear dependence in $span\{\phi(x), \phi(y)\}$ corresponds to a cycle in $H_x \cup H_y$

Proof contd.

- Let (x, y) be an input. And H_x be the edge-subgraph of the branching program whose edges query variables in x. Similarly define H_y.
- After the transformation $f(x, y) = 1$ iff $H_x \cup H_y$ contains a cycle
- Make sure that for any (x, y) s.t. $f(x, y) = 1$ this unique cycle has edges from both H_x and H_y.
- Any linear dependence in $span\{\phi(x), \phi(y)\}$ corresponds to a cycle in $H_x \cup H_y$

Known Bounds on $\text{pd}_{\mathbb{F}}$

- (Existential) N vertex bipartite G such that

$\text{pd}_{\mathbb{F}}(G)$	Field	Result
$\Omega\left(\sqrt{\frac{N}{\log N}}\right)$	Infinite	Babai et.al, 2002
$\Omega\left(\sqrt{N}\right)$	Finite	Pudlak and Rodl, 1992

- (Explicit) $G =$ Complement of N perfect matchings. $\text{pd}_{\mathbb{R}}(G) = \Omega(\log N)$
- (Upper bounds) Bipartite G, $\text{pd}_{\mathbb{R}}(G) = O\left(\frac{N}{\log N}\right)$
- To summarize, we only know **linear** lower bounds for projective dimension of explicit functions.

Known Bounds on $\mathrm{pd}_{\mathbb{F}}$

- (Existential) N vertex bipartite G such that

$\mathrm{pd}_{\mathbb{F}}(G)$	Field	Result
$\Omega\left(\sqrt{\frac{N}{\log N}}\right)$	Infinite	Babai et.al, 2002
$\Omega\left(\sqrt{N}\right)$	Finite	Pudlak and Rodl, 1992

- (Explicit) $G =$ Complement of N perfect matchings. $\mathrm{pd}_{\mathbb{R}}(G) = \Omega(\log N)$
- (Upper bounds) Bipartite G, $\mathrm{pd}_{\mathbb{R}}(G) = O\left(\frac{N}{\log N}\right)$
- To summarize, we only know **linear** lower bounds for projective dimension of explicit functions.

Known Bounds on $\mathrm{pd}_{\mathbb{F}}$

- (Existential) N vertex bipartite G such that

$\mathrm{pd}_{\mathbb{F}}(G)$	Field	Result
$\Omega\left(\sqrt{\frac{N}{\log N}}\right)$	Infinite	Babai et.al, 2002
$\Omega\left(\sqrt{N}\right)$	Finite	Pudlak and Rodl, 1992

- (Explicit) $G =$ Complement of N perfect matchings. $\mathrm{pd}_{\mathbb{R}}(G) = \Omega(\log N)$
- (Upper bounds) Bipartite G, $\mathrm{pd}_{\mathbb{R}}(G) = O\left(\frac{N}{\log N}\right)$
- To summarize, we only know **linear** lower bounds for projective dimension of explicit functions.

Known Bounds on $\mathrm{pd}_{\mathbb{F}}$

- (Existential) N vertex bipartite G such that

$\mathrm{pd}_{\mathbb{F}}(G)$	Field	Result
$\Omega\left(\sqrt{\frac{N}{\log N}}\right)$	Infinite	Babai et.al, 2002
$\Omega\left(\sqrt{N}\right)$	Finite	Pudlak and Rodl, 1992

- (Explicit) $G =$ Complement of N perfect matchings. $\mathrm{pd}_{\mathbb{R}}(G) = \Omega(\log N)$
- (Upper bounds) Bipartite G, $\mathrm{pd}_{\mathbb{R}}(G) = O\left(\frac{N}{\log N}\right)$
- To summarize, we only know **linear** lower bounds for projective dimension of explicit functions.

An exponential gap!

Our Result, a similar result known for Formulas and Graph Complexity by Jukna

There exists (non-explicit) function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ such that $\mathrm{pd}(f) = O(n)$, but $\mathrm{bpsize}(f) = \Omega(2^n/n)$.

An exponential gap!

Our Result, a similar result known for Formulas and Graph Complexity by Jukna

There exists (non-explicit) function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ such that $\mathrm{pd}(f) = O(n)$, but $\mathrm{bpsize}(f) = \Omega(2^n/n)$.

An exponential gap!

- Projective dimension of a bipartite graph $G(U, V, E)$ is invariant under relabeling vertices on the right side
- Move the subspace assignments of the vertices along with the vertices
- The Equality function denoted by $\mathrm{EQ}(x, y)$ checks whether two n bit strings x and y are equal. Has BP of size $O(n)$. Hence $\mathrm{pd}(G_{\mathrm{EQ}_n}) = O(n)$
- Let $\pi \in S_{2^n}$ be a permutation of the right vertices (y's). For any two different permutations the resulting bipartite graph has same projective dimension as EQ_n.
- But for any two different permutations the corresponding Boolean function is different.
- There are only $2^{O(S \log S)}$ different branching programs of size at most S

An exponential gap!

- Projective dimension of a bipartite graph $G(U, V, E)$ is invariant under relabeling vertices on the right side
- Move the subspace assignments of the vertices along with the vertices
- The Equality function denoted by $\mathrm{EQ}(x, y)$ checks whether two n bit strings x and y are equal. Has BP of size $O(n)$. Hence $\mathrm{pd}(G_{\mathrm{EQ}_n}) = O(n)$
- Let $\pi \in S_{2^n}$ be a permutation of the right vertices (y's). For any two different permutations the resulting bipartite graph has same projective dimension as EQ_n.
- But for any two different permutations the corresponding Boolean function is different.
- There are only $2^{O(S \log S)}$ different branching programs of size at most S

An exponential gap!

- Projective dimension of a bipartite graph $G(U, V, E)$ is invariant under relabeling vertices on the right side
- Move the subspace assignments of the vertices along with the vertices
- The Equality function denoted by $\mathrm{EQ}(x, y)$ checks whether two n bit strings x and y are equal. Has BP of size $O(n)$. Hence $\mathsf{pd}(G_{\mathrm{EQ}_n}) = O(n)$
- Let $\pi \in S_{2^n}$ be a permutation of the right vertices (y's). For any two different permutations the resulting bipartite graph has same projective dimension as EQ_n.
- But for any two different permutations the corresponding Boolean function is different.
- There are only $2^{O(S \log S)}$ different branching programs of size at most S

An exponential gap!

- Projective dimension of a bipartite graph $G(U, V, E)$ is invariant under relabeling vertices on the right side
- Move the subspace assignments of the vertices along with the vertices
- The Equality function denoted by $\mathrm{EQ}(x, y)$ checks whether two n bit strings x and y are equal. Has BP of size $O(n)$. Hence $\mathrm{pd}(G_{\mathrm{EQ}_n}) = O(n)$
- Let $\pi \in S_{2^n}$ be a permutation of the right vertices (y's). For any two different permutations the resulting bipartite graph has same projective dimension as EQ_n.
- But for any two different permutations the corresponding Boolean function is different.
- There are only $2^{O(S \log S)}$ different branching programs of size at most S

An exponential gap!

- Projective dimension of a bipartite graph $G(U, V, E)$ is invariant under relabeling vertices on the right side
- Move the subspace assignments of the vertices along with the vertices
- The Equality function denoted by $\mathrm{EQ}(x, y)$ checks whether two n bit strings x and y are equal. Has BP of size $O(n)$. Hence $\mathrm{pd}(G_{\mathrm{EQ}_n}) = O(n)$
- Let $\pi \in S_{2^n}$ be a permutation of the right vertices (y's). For any two different permutations the resulting bipartite graph has same projective dimension as EQ_n.
- But for any two different permutations the corresponding Boolean function is different.
- There are only $2^{O(S \log S)}$ different branching programs of size at most S

An exponential gap!

- Projective dimension of a bipartite graph $G(U, V, E)$ is invariant under relabeling vertices on the right side
- Move the subspace assignments of the vertices along with the vertices
- The Equality function denoted by $\mathrm{EQ}(x, y)$ checks whether two n bit strings x and y are equal. Has BP of size $O(n)$. Hence $\mathrm{pd}(G_{\mathrm{EQ}_n}) = O(n)$
- Let $\pi \in S_{2^n}$ be a permutation of the right vertices (y's). For any two different permutations the resulting bipartite graph has same projective dimension as EQ_n.
- But for any two different permutations the corresponding Boolean function is different.
- There are only $2^{O(S \log S)}$ different branching programs of size at most S

Bridging the gap

- The gap example gave an assignment which is of low projective dimension, but it may not be easy (read poly in n) to describe
- The assignment constructed from branching program by Pudlak and Rodl is easy to describe.
- There are $4n$ subspaces, 2 for each of the $2n$ bits whose various spans create all the subspaces assigned to the $2^n + 2^n$ vertices of the bipartite graph
- For each $i \in [2n]$ and $b \in \{0, 1\}$, look at the edges querying $x_i = b$. The span of the vectors assigned to these edges constitute these building block sub-spaces.

Bridging the gap

- The gap example gave an assignment which is of low projective dimension, but it may not be easy (read poly in n) to describe
- The assignment constructed from branching program by Pudlak and Rodl is easy to describe.
- There are $4n$ subspaces, 2 for each of the $2n$ bits whose various spans create all the subspaces assigned to the $2^n + 2^n$ vertices of the bipartite graph
- For each $i \in [2n]$ and $b \in \{0, 1\}$, look at the edges querying $x_i = b$. The span of the vectors assigned to these edges constitute these building block sub-spaces.

Bridging the gap

- The gap example gave an assignment which is of low projective dimension, but it may not be easy (read poly in n) to describe
- The assignment constructed from branching program by Pudlak and Rodl is easy to describe.
- There are $4n$ subspaces, 2 for each of the $2n$ bits whose various spans create all the subspaces assigned to the $2^n + 2^n$ vertices of the bipartite graph
- For each $i \in [2n]$ and $b \in \{0, 1\}$, look at the edges querying $x_i = b$. The span of the vectors assigned to these edges constitute these building block sub-spaces.

Bridging the gap

- The gap example gave an assignment which is of low projective dimension, but it may not be easy (read poly in n) to describe
- The assignment constructed from branching program by Pudlak and Rodl is easy to describe.
- There are $4n$ subspaces, 2 for each of the $2n$ bits whose various spans create all the subspaces assigned to the $2^n + 2^n$ vertices of the bipartite graph
- For each $i \in [2n]$ and $b \in \{0,1\}$, look at the edges querying $x_i = b$. The span of the vectors assigned to these edges constitute these building block sub-spaces.

Bitwise Decomposable Projective Dimension

Definition

For $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, $bpdim(f) \leq d$ if there exists $\mathscr{C} = \{U_i^a \mid a \in \{0,1\}, i \in [n]\}$, $\mathscr{D} = \{V_i^a \mid a \in \{0,1\}, i \in [n]\}$, such that

- $\phi(x) = span_{i \in [n]} \{U_i^{x_i}\}$,
- Each U_i^a is a span of difference of standard basis vectors. Similarly each V_a^i
- If $e_i - e_j \in span U_k^0 \cup U_k^1$ then for any e_l, $e_i - e_l$ and $e_j - e_l$ are not in $span_{m \neq k, a\{0,1\}} U_m^a$. Similar condition for $\mathscr{D}$.
- $\mathscr{C}, \mathscr{D}$ subspaces from $\mathbb{F}_2^d$

Main Result

$$\text{bitpdim}(f) = \Omega(\text{bpsize}(f)^{1/6})$$

Bitwise Decomposable Projective Dimension

Definition

For $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, $bpdim(f) \leq d$ if there exists $\mathscr{C} = \{U_i^a \mid a \in \{0,1\}, i \in [n]\}$, $\mathscr{D} = \{V_i^a \mid a \in \{0,1\}, i \in [n]\}$, such that

- $\phi(x) = span_{i \in [n]} \{U_i^{x_i}\}$,
- Each U_i^a is a span of difference of standard basis vectors. Similarly each V_a^i
- If $e_i - e_j \in span U_k^0 \cup U_k^1$ then for any e_l, $e_i - e_l$ and $e_j - e_l$ are not in $span_{m \neq k, a\{0,1\}} U_m^a$. Similar condition for $\mathscr{D}$.
- $\mathscr{C}, \mathscr{D}$ subspaces from $\mathbb{F}_2^d$

Main Result

$$\text{bitpdim}(f) = \Omega(\text{bpsize}(f)^{1/6})$$

Bitwise Decomposable Projective Dimension

Definition

For $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, $bpdim(f) \leq d$ if there exists $\mathscr{C} = \{U_i^a \mid a \in \{0,1\}, i \in [n]\}$, $\mathscr{D} = \{V_i^a \mid a \in \{0,1\}, i \in [n]\}$, such that

- $\phi(x) = span_{i\in[n]}\{U_i^{x_i}\}$,
- Each U_i^a is a span of difference of standard basis vectors. Similarly each V_a^i
- If $e_i - e_j \in span U_k^0 \cup U_k^1$ then for any e_l, $e_i - e_l$ and $e_j - e_l$ are not in $span_{m\neq k, a\{0,1\}} U_m^a$. Similar condition for $\mathscr{D}$.
- $\mathscr{C}, \mathscr{D}$ subspaces from $\mathbb{F}_2^d$

Main Result

$$\text{bitpdim}(f) = \Omega(\text{bpsize}(f)^{1/6})$$

Bitwise Decomposable Projective Dimension

Definition

For $f : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$, $bpdim(f) \leq d$ if there exists $\mathscr{C} = \{U_i^a \mid a \in \{0,1\}, i \in [n]\}$, $\mathscr{D} = \{V_i^a \mid a \in \{0,1\}, i \in [n]\}$, such that

- $\phi(x) = span_{i \in [n]} \{U_i^{x_i}\}$,
- Each U_i^a is a span of difference of standard basis vectors. Similarly each V_a^i
- If $e_i - e_j \in span U_k^0 \cup U_k^1$ then for any e_l, $e_i - e_l$ and $e_j - e_l$ are not in $span_{m \neq k, a\{0,1\}} U_m^a$. Similar condition for $\mathscr{D}$.
- $\mathscr{C}, \mathscr{D}$ subspaces from $\mathbb{F}_2^d$

Main Result

$$\text{bitpdim}(f) = \Omega(\text{bpsize}(f)^{1/6})$$

Bitwise Decomposable Projective Dimension

Definition

For $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, $bpdim(f) \leq d$ if there exists $\mathscr{C} = \{U_i^a \mid a \in \{0,1\}, i \in [n]\}$, $\mathscr{D} = \{V_i^a \mid a \in \{0,1\}, i \in [n]\}$, such that

- $\phi(x) = span_{i \in [n]} \{U_i^{x_i}\}$,
- Each U_i^a is a span of difference of standard basis vectors. Similarly each V_a^i
- If $e_i - e_j \in span U_k^0 \cup U_k^1$ then for any e_l, $e_i - e_l$ and $e_j - e_l$ are not in $span_{m \neq k, a\{0,1\}} U_m^a$. Similar condition for $\mathscr{D}$.
- $\mathscr{C}, \mathscr{D}$ subspaces from $\mathbb{F}_2^d$

Main Result

$$\text{bitpdim}(f) = \Omega(\text{bpsize}(f)^{1/6})$$

bitpdim assignment from Branching Programs

- Excpet for, If $e_i - e_j \in span U_k^0 \cup U_k^1$ then for any e_l, $e_i - e_l$ and $e_j - e_l$ are not in $span_{m \neq k, a\{0,1\}} U_m^a$, all the other conditions are satisfied by Pudlak Rodl Construction
- Modify the branching program so that no two edges which share an end vertex query variables from the same partition
- This can be done by blowing up the size of the given branching program by a factor of at most 4.

bitpdim assignment from Branching Programs

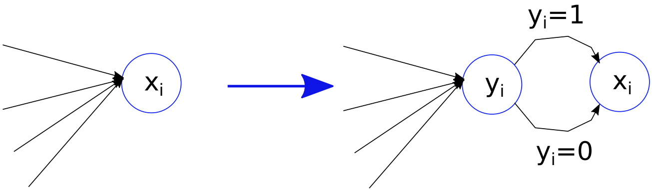
- Excpet for, If $e_i - e_j \in span U_k^0 \cup U_k^1$ then for any e_l, $e_i - e_l$ and $e_j - e_l$ are not in $span_{m \neq k, a\{0,1\}} U_m^a$, all the other conditions are satisfied by Pudlak Rodl Construction
- Modify the branching program so that no two edges which share an end vertex query variables from the same partition
- This can be done by blowing up the size of the given branching program by a factor of at most 4.

bitpdim assignment from Branching Programs

- Excpet for, If $e_i - e_j \in span U_k^0 \cup U_k^1$ then for any e_l, $e_i - e_l$ and $e_j - e_l$ are not in $span_{m \neq k, a\{0,1\}} U_m^a$, all the other conditions are satisfied by Pudlak Rodl Construction
- Modify the branching program so that no two edges which share an end vertex query variables from the same partition
- This can be done by blowing up the size of the given branching program by a factor of at most 4.

bitpdim assignment from Branching Programs

- Excpet for, If $e_i - e_j \in span U_k^0 \cup U_k^1$ then for any e_l, $e_i - e_l$ and $e_j - e_l$ are not in $span_{m \neq k, a\{0,1\}} U_m^a$, all the other conditions are satisfied by Pudlak Rodl Construction
- Modify the branching program so that no two edges which share an end vertex query variables from the same partition
- This can be done by blowing up the size of the given branching program by a factor of at most 4.



Branching programs from bitpdim

Theorem

$\text{bitpdim}(f) \leq d(n) \implies \text{bpsize}(f) \leq (d(n))^6$

Proof.

(Sketch)

- We describe a space bounded algorithm which given the bitpdim assignment as an advice, and two inputs (x, y) computes whether $f(x, y) = 1$.
- implicit G, vertices – standard basis vectors in ϕ, $(u, v) \in E(G^*)$ iff $e_u - e_v \in U_i^{x_i}$ or $V_j^{y_j}$.
- Argue that any linear dependence in $span\{\phi(x) \cup \phi(y)\}$ is a cycle in G^*.
- Coordinate-wise disjointedness of the basis vectors constituting $U_i^{x_i}$ and $U_j^{x_j}$ ensure that there is no cycle involving just edges from H_x

Branching programs from bitpdim

Theorem

$\text{bitpdim}(f) \leq d(n) \implies \text{bpsize}(f) \leq (d(n))^6$

Proof.

(Sketch)

- We describe a space bounded algorithm which given the bitpdim assignment as an advice, and two inputs (x, y) computes whether $f(x, y) = 1$.
- implicit G, vertices – standard basis vectors in ϕ, $(u, v) \in E(G^*)$ iff $e_u - e_v \in U_i^{x_i}$ or $V_j^{y_j}$.
- Argue that any linear dependence in $span\{\phi(x) \cup \phi(y)\}$ is a cycle in G^*.
- Coordinate-wise disjointedness of the basis vectors constituting $U_i^{x_i}$ and $U_j^{x_j}$ ensure that there is no cycle involving just edges from H_x

Branching programs from bitpdim

Theorem

$\text{bitpdim}(f) \leq d(n) \implies \text{bpsize}(f) \leq (d(n))^6$

Proof.

(Sketch)

- We describe a space bounded algorithm which given the bitpdim assignment as an advice, and two inputs (x, y) computes whether $f(x, y) = 1$.
- implicit G, vertices – standard basis vectors in ϕ, $(u, v) \in E(G^*)$ iff $e_u - e_v \in U_i^{x_i}$ or $V_j^{y_j}$.
- Argue that any linear dependence in $span\{\phi(x) \cup \phi(y)\}$ is a cycle in G^*.
- Coordinate-wise disjointedness of the basis vectors constituting $U_i^{x_i}$ and $U_j^{x_j}$ ensure that there is no cycle involving just edges from H_x

Branching programs from bitpdim

Theorem

$\text{bitpdim}(f) \leq d(n) \implies \text{bpsize}(f) \leq (d(n))^6$

Proof.

(Sketch)

- We describe a space bounded algorithm which given the bitpdim assignment as an advice, and two inputs (x, y) computes whether $f(x, y) = 1$.
- implicit G, vertices – standard basis vectors in ϕ, $(u, v) \in E(G^*)$ iff $e_u - e_v \in U_i^{x_i}$ or $V_j^{y_j}$.
- Argue that any linear dependence in $span\{\phi(x) \cup \phi(y)\}$ is a cycle in G^*.
- Coordinate-wise disjointedness of the basis vectors constituting $U_i^{x_i}$ and $U_j^{x_j}$ ensure that there is no cycle involving just edges from H_x

Branching programs from bitpdim

Theorem

$\text{bitpdim}(f) \leq d(n) \implies \text{bpsize}(f) \leq (d(n))^6$

Proof.

(Sketch) Given (x, y)

- implicit G, vertices – standard basis vectors in ϕ, $(u, v) \in E(G^*)$ iff $e_u - e_v \in U_i^{x_i}$ or $V_j^{y_j}$.
- $f(x, y) = 1$ iff there is a cycle in G^*
- check for a cycle in G^*. Can be done in space $5 \log |G^*|$
- $|G^*| = \text{bitpdim}(f)$.

□

Branching programs from bitpdim

Theorem

$\text{bitpdim}(f) \leq d(n) \implies \text{bpsize}(f) \leq (d(n))^6$

Proof.

(Sketch) Given (x, y)

- implicit G, vertices – standard basis vectors in ϕ, $(u, v) \in E(G^*)$ iff $e_u - e_v \in U_i^{x_i}$ or $V_j^{y_j}$.
- $f(x, y) = 1$ iff there is a cycle in G^*
- check for a cycle in G^*. Can be done in space $5 \log |G^*|$
- $|G^*| = \text{bitpdim}(f)$.

□

Branching programs from bitpdim

Theorem

$\text{bitpdim}(f) \leq d(n) \implies \text{bpsize}(f) \leq (d(n))^6$

Proof.

(Sketch) Given (x, y)

- implicit G, vertices – standard basis vectors in ϕ, $(u, v) \in E(G^*)$ iff $e_u - e_v \in U_i^{x_i}$ or $V_j^{y_j}$.
- $f(x, y) = 1$ iff there is a cycle in G^*
- check for a cycle in G^*. Can be done in space $5 \log |G^*|$
- $|G^*| = \text{bitpdim}(f)$.

□

Branching programs from bitpdim

Theorem

$\text{bitpdim}(f) \leq d(n) \implies \text{bpsize}(f) \leq (d(n))^6$

Proof.

(Sketch) Given (x, y)

- implicit G, vertices – standard basis vectors in ϕ, $(u, v) \in E(G^*)$ iff $e_u - e_v \in U_i^{x_i}$ or $V_j^{y_j}$.
- $f(x, y) = 1$ iff there is a cycle in G^*
- check for a cycle in G^*. Can be done in space $5 \log |G^*|$
- $|G^*| = \text{bitpdim}(f)$.

□

Super linear lower bounds

- The best bitpdim lower bound we get from the best known branching programs lower bounds is only **sub-linear**
- The best known pd lower bound is linear
- Can we get a super-linear lower bound ?
- Yes, but the proof we could come up with relies on using Nechiporuk's method

Super linear lower bounds

- The best bitpdim lower bound we get from the best known branching programs lower bounds is only **sub-linear**
- The best known pd lower bound is linear
- Can we get a super-linear lower bound ?
- Yes, but the proof we could come up with relies on using Nechiporuk's method

Super linear lower bounds

- The best bitpdim lower bound we get from the best known branching programs lower bounds is only **sub-linear**
- The best known pd lower bound is linear
- Can we get a super-linear lower bound ?
- Yes, but the proof we could come up with relies on using Nechiporuk's method

Super linear lower bounds

- The best bitpdim lower bound we get from the best known branching programs lower bounds is only **sub-linear**
- The best known pd lower bound is linear
- Can we get a super-linear lower bound ?
- Yes, but the proof we could come up with relies on using Nechiporuk's method

Super linear lower bounds - proof sketch

- Recall the function ED.
  - $\mathrm{ED}_m : \{0,1\}^{n=m2\log m} \to \{0,1\}$
  - m inputs $x_1, \ldots, x_m$ each representing a number in $[m^2]$
  - $f(x_1, \ldots, x_m) = 1$ iff no two x_i, x_j are equal
- Let $U_1^0, U_1^1, \ldots, U_{m/2\times 2\log m}^0, U_{m/2\times 2\log m}^1$ and $V_1^0, V_1^1, \ldots, V_{m/2\times 2\log m}^0, V_{m/2\times 2\log m}^1$ be a bitwise assignment for ED_m.
- For $1 \leq i \leq m/2$ let $d_i = \dim span\left\{U_j^b\right\}_{j \text{is a bit of } x_i, b\in\{0,1\}}$
- We will show that $d_i = \Omega(n/\log n)$, thus $d = \sum_{i=1}^{m/2} d_i = \Omega(\frac{n^2}{2\log n})$. as the subspace constituting the left are disjoint.
- Let $\rho : \{0,1\}^{n=m2\log m} \to \{0,1,*\}$ be a restriction that fixes all the bit except the $2\log m$ bits representing x_i. Also $\mathrm{ED}_m\mid_\rho$ is not a constant function.

Super linear lower bounds - proof sketch

- Recall the function ED.
  - $\mathrm{ED}_m : \{0,1\}^{n=m2\log m} \to \{0,1\}$
  - m inputs $x_1,\ldots,x_m$ each representing a number in $[m^2]$
  - $f(x_1,\ldots,x_m) = 1$ iff no two x_i, x_j are equal
- Let $U_1^0, U_1^1, \ldots, U_{m/2\times 2\log m}^0, U_{m/2\times 2\log m}^1$ and $V_1^0, V_1^1, \ldots, V_{m/2\times 2\log m}^0, V_{m/2\times 2\log m}^1$ be a bitwise assignment for ED_m.
- For $1 \le i \le m/2$ let $d_i = \dim span \left\{ U_j^b \right\}_{j \text{is a bit of } x_i, b\in\{0,1\}}$
- We will show that $d_i = \Omega(n/\log n)$, thus $d = \sum_{i=1}^{m/2} d_i = \Omega(\frac{n^2}{2\log n})$. as the subspace constituting the left are disjoint.
- Let $\rho : \{0,1\}^{n=m2\log m} \to \{0,1,*\}$ be a restriction that fixes all the bit except the $2\log m$ bits representing x_i. Also $\mathrm{ED}_m\,|_\rho$ is not a constant function.

Super linear lower bounds - proof sketch

- Recall the function ED.
  - $\mathrm{ED}_m : \{0,1\}^{n=m2\log m} \to \{0,1\}$
  - m inputs $x_1, \ldots, x_m$ each representing a number in $[m^2]$
  - $f(x_1, \ldots, x_m) = 1$ iff no two x_i, x_j are equal
- Let $U_1^0, U_1^1, \ldots, U_{m/2\times 2\log m}^0, U_{m/2\times 2\log m}^1$ and $V_1^0, V_1^1, \ldots, V_{m/2\times 2\log m}^0, V_{m/2\times 2\log m}^1$ be a bitwise assignment for ED_m.
- For $1 \leq i \leq m/2$ let $d_i = \dim span \left\{ U_j^b \right\}_{j \text{is a bit of } x_i, b\in\{0,1\}}$
- We will show that $d_i = \Omega(n/\log n)$, thus $d = \sum_{i=1}^{m/2} d_i = \Omega(\frac{n^2}{2\log n})$. as the subspace constituting the left are disjoint.
- Let $\rho : \{0,1\}^{n=m2\log m} \to \{0,1,*\}$ be a restriction that fixes all the bit except the $2\log m$ bits representing x_i. Also $\mathrm{ED}_m \mid_\rho$ is not a constant function.

Super linear lower bounds - proof sketch

- Recall the function ED.
  - $\mathrm{ED}_m : \{0,1\}^{n=m2\log m} \to \{0,1\}$
  - m inputs $x_1, \ldots, x_m$ each representing a number in $[m^2]$
  - $f(x_1, \ldots, x_m) = 1$ iff no two x_i, x_j are equal
- Let $U_1^0, U_1^1, \ldots, U_{m/2\times 2\log m}^0, U_{m/2\times 2\log m}^1$ and $V_1^0, V_1^1, \ldots, V_{m/2\times 2\log m}^0, V_{m/2\times 2\log m}^1$ be a bitwise assignment for ED_m.
- For $1 \leq i \leq m/2$ let $d_i = \dim span\left\{U_j^b\right\}_{j\text{is a bit of } x_i, b\in\{0,1\}}$
- We will show that $d_i = \Omega(n/\log n)$, thus $d = \sum_{i=1}^{m/2} d_i = \Omega(\frac{n^2}{2\log n})$. as the subspace constituting the left are disjoint.
- Let $\rho : \{0,1\}^{n=m2\log m} \to \{0,1,*\}$ be a restriction that fixes all the bit except the $2\log m$ bits representing x_i. Also $\mathrm{ED}_m\mid_\rho$ is not a constant function.

Super linear lower bounds - proof sketch

- Recall the function ED.
  - $\mathrm{ED}_m : \{0,1\}^{n=m2\log m} \to \{0,1\}$
  - m inputs $x_1,\ldots,x_m$ each representing a number in $[m^2]$
  - $f(x_1,\ldots,x_m) = 1$ iff no two x_i, x_j are equal
- Let $U_1^0, U_1^1, \ldots, U^0_{m/2\times 2\log m}, U^1_{m/2\times 2\log m}$ and $V_1^0, V_1^1, \ldots, V^0_{m/2\times 2\log m}, V^1_{m/2\times 2\log m}$ be a bitwise assignment for ED_m.
- For $1 \le i \le m/2$ let $d_i = \dim span \left\{ U_j^b \right\}_{j \text{is a bit of } x_i, b\in\{0,1\}}$
- We will show that $d_i = \Omega(n/\log n)$, thus $d = \sum_{i=1}^{m/2} d_i = \Omega(\frac{n^2}{2\log n})$. as the subspace constituting the left are disjoint.
- Let $\rho : \{0,1\}^{n=m2\log m} \to \{0,1,*\}$ be a restriction that fixes all the bit except the $2\log m$ bits representing x_i. Also $\mathrm{ED}_m\mid_\rho$ is not a constant function.

Super linear lower bounds - proof sketch

- Recall the function ED.
  - $\mathrm{ED}_m : \{0,1\}^{n=m2\log m} \to \{0,1\}$
  - m inputs $x_1,\ldots,x_m$ each representing a number in $[m^2]$
  - $f(x_1,\ldots,x_m) = 1$ iff no two x_i, x_j are equal
- Let $U_1^0, U_1^1, \ldots, U_{m/2\times 2\log m}^0, U_{m/2\times 2\log m}^1$ and $V_1^0, V_1^1, \ldots, V_{m/2\times 2\log m}^0, V_{m/2\times 2\log m}^1$ be a bitwise assignment for ED_m.
- For $1 \le i \le m/2$ let $d_i = \dim span\left\{U_j^b\right\}_{j\text{is a bit of } x_i, b\in\{0,1\}}$
- We will show that $d_i = \Omega(n/\log n)$, thus $d = \sum_{i=1}^{m/2} d_i = \Omega(\frac{n^2}{2\log n})$. as the subspace constituting the left are disjoint.
- Let $\rho : \{0,1\}^{n=m2\log m} \to \{0,1,*\}$ be a restriction that fixes all the bit except the $2\log m$ bits representing x_i. Also $\mathrm{ED}_m\mid_\rho$ is not a constant function.

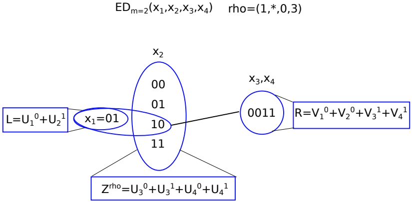
Super linear lower bounds - proof sketch

- Recall the function ED.
  - $\mathrm{ED}_m : \{0,1\}^{n=m2\log m} \to \{0,1\}$
  - m inputs $x_1, \ldots, x_m$ each representing a number in $[m^2]$
  - $f(x_1, \ldots, x_m) = 1$ iff no two x_i, x_j are equal
- Let $U_1^0, U_1^1, \ldots, U_{m/2\times 2\log m}^0, U_{m/2\times 2\log m}^1$ and $V_1^0, V_1^1, \ldots, V_{m/2\times 2\log m}^0, V_{m/2\times 2\log m}^1$ be a bitwise assignment for ED_m.
- For $1 \le i \le m/2$ let $d_i = \dim span \left\{ U_j^b \right\}_{j\text{is a bit of } x_i, b\in\{0,1\}}$
- We will show that $d_i = \Omega(n/\log n)$, thus $d = \sum_{i=1}^{m/2} d_i = \Omega(\frac{n^2}{2\log n})$. as the subspace constituting the left are disjoint.
- Let $\rho : \{0,1\}^{n=m2\log m} \to \{0,1,*\}$ be a restriction that fixes all the bit except the $2\log m$ bits representing x_i. Also $\mathrm{ED}_m|_\rho$ is not a constant function.

Super linear lower bounds - proof sketch

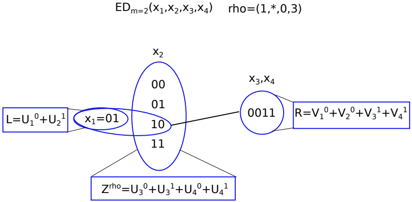
- Recall the function ED.
  - $\mathrm{ED}_m : \{0,1\}^{n=m2\log m} \to \{0,1\}$
  - m inputs $x_1,\ldots,x_m$ each representing a number in $[m^2]$
  - $f(x_1,\ldots,x_m) = 1$ iff no two x_i, x_j are equal
- Let $U_1^0, U_1^1, \ldots, U_{m/2\times 2\log m}^0, U_{m/2\times 2\log m}^1$ and $V_1^0, V_1^1, \ldots, V_{m/2\times 2\log m}^0, V_{m/2\times 2\log m}^1$ be a bitwise assignment for ED_m.
- For $1 \le i \le m/2$ let $d_i = \dim span \left\{ U_j^b \right\}_{j\text{is a bit of } x_i, b\in\{0,1\}}$
- We will show that $d_i = \Omega(n/\log n)$, thus $d = \sum_{i=1}^{m/2} d_i = \Omega(\frac{n^2}{2\log n})$. as the subspace constituting the left are disjoint.
- Let $\rho : \{0,1\}^{n=m2\log m} \to \{0,1,*\}$ be a restriction that fixes all the bit except the $2\log m$ bits representing x_i. Also $\mathrm{ED}_m\,|_\rho$ is not a constant function.

Super linear lower bounds - proof sketch



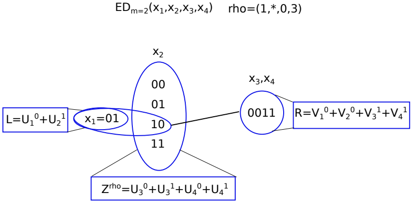
- Since ρ doesn't make the function constant $L \cap R = \{0\}$.
- Replace R with $\Pi_{Z^\rho}(R)$, that is project away L from R
- On the left side consider only vectors from Z^ρ
- For two different restrictions say ρ_1 and ρ_2 both of which fixes everything but bits of x_i, $Z^{\rho_1} = Z^{\rho_2}$ and the assignment on the left is the same.
- Thus the only thing that changes is $\Pi_{Z^\rho}(R)$.
- Let $S = \{e_u - e_v | e_u - e_v \in Z^\rho\}$. We show that there exist $S' \subseteq S$ s.t. $\Pi_{Z^\rho}(R) = span\{S'\}$.

Super linear lower bounds - proof sketch



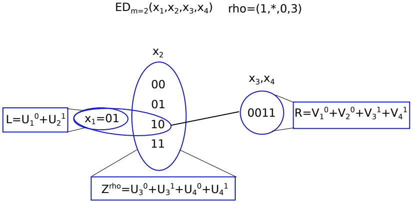
- Since ρ doesn't make the function constant $L \cap R = \{0\}$.
- Replace R with $\Pi_{Z^\rho}(R)$, that is project away L from R
- On the left side consider only vectors from Z^ρ
- For two different restrictions say ρ_1 and ρ_2 both of which fixes everything but bits of x_i, $Z^{\rho_1} = Z^{\rho_2}$ and the assignment on the left is the same.
- Thus the only thing that changes is $\Pi_{Z^\rho}(R)$.
- Let $S = \{e_u - e_v | e_u - e_v \in Z^\rho\}$. We show that there exist $S' \subseteq S$ s.t. $\Pi_{Z^\rho}(R) = span\{S'\}$.

Super linear lower bounds - proof sketch



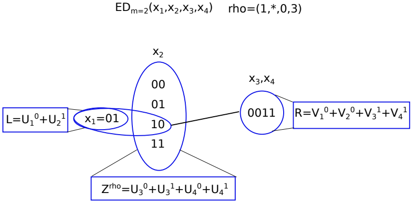
- Since ρ doesn't make the function constant $L \cap R = \{0\}$.
- Replace R with $\Pi_{Z^\rho}(R)$, that is project away L from R
- On the left side consider only vectors from Z^ρ
- For two different restrictions say ρ_1 and ρ_2 both of which fixes everything but bits of x_i, $Z^{\rho_1} = Z^{\rho_2}$ and the assignment on the left is the same.
- Thus the only thing that changes is $\Pi_{Z^\rho}(R)$.
- Let $S = \{e_u - e_v | e_u - e_v \in Z^\rho\}$. We show that there exist $S' \subseteq S$ s.t. $\Pi_{Z^\rho}(R) = span\{S'\}$.

Super linear lower bounds - proof sketch



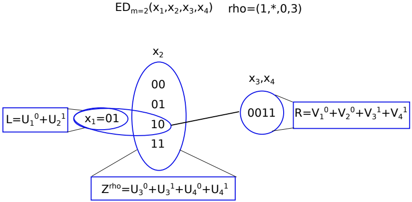
- Since ρ doesn't make the function constant $L \cap R = \{0\}$.
- Replace R with $\Pi_{Z^\rho}(R)$, that is project away L from R
- On the left side consider only vectors from Z^ρ
- For two different restrictions say ρ_1 and ρ_2 both of which fixes everything but bits of x_i, $Z^{\rho_1} = Z^{\rho_2}$ and the assignment on the left is the same.
- Thus the only thing that changes is $\Pi_{Z^\rho}(R)$.
- Let $S = \{e_u - e_v | e_u - e_v \in Z^\rho\}$. We show that there exist $S' \subseteq S$ s.t. $\Pi_{Z^\rho}(R) = span\{S'\}$.

Super linear lower bounds - proof sketch



- Since ρ doesn't make the function constant $L \cap R = \{0\}$.
- Replace R with $\Pi_{Z^\rho}(R)$, that is project away L from R
- On the left side consider only vectors from Z^ρ
- For two different restrictions say ρ_1 and ρ_2 both of which fixes everything but bits of x_i, $Z^{\rho_1} = Z^{\rho_2}$ and the assignment on the left is the same.
- Thus the only thing that changes is $\Pi_{Z^\rho}(R)$.
- Let $S = \{e_u - e_v | e_u - e_v \in Z^\rho\}$. We show that there exist $S' \subseteq S$ s.t. $\Pi_{Z^\rho}(R) = span\{S'\}$.

Super linear lower bounds - proof sketch



- Since ρ doesn't make the function constant $L \cap R = \{0\}$.
- Replace R with $\Pi_{Z^\rho}(R)$, that is project away L from R
- On the left side consider only vectors from Z^ρ
- For two different restrictions say ρ_1 and ρ_2 both of which fixes everything but bits of x_i, $Z^{\rho_1} = Z^{\rho_2}$ and the assignment on the left is the same.
- Thus the only thing that changes is $\Pi_{Z^\rho}(R)$.
- Let $S = \{e_u - e_v | e_u - e_v \in Z^\rho\}$. We show that there exist $S' \subseteq S$ s.t. $\Pi_{Z^\rho}(R) = span\{S'\}$.

- Can we come up with a super-linear lower bound which doesn't use Nechiporuk's method
  - Nechiporuk's method cannot prove better than n^2.
  - Sub-function count bottleneck : Let ρ fix $n-k$ bits of the n bits of a function. The number of different sub functions is $\min 2^{n-k}, 2^{2^k}$.
  - Element Distinctness has an n^2 sized branching program
- A candidate function : Given two $d \times d$ matrices A, B, $f(A,B) = 1$ iff and only $\mathrm{rowspace}(A) \cap \mathrm{rowspace}(B) \neq \{0\}$
- Not believed to be in **L**, but is in **P**
- Projective dimension of this function is just d, which is sub-linear in input size
- One can prove super linear bitpdim lower bounds for this function using Nechiporuk's method. But we would like to prove a super linear lower bound using a purely algebraic method.

- Can we come up with a super-linear lower bound which doesn't use Nechiporuk's method
  - Nechiporuk's method cannot prove better than n^2.
  - Sub-function count bottleneck : Let ρ fix $n-k$ bits of the n bits of a function. The number of different sub functions is $\min 2^{n-k}, 2^{2^k}$.
  - Element Distinctness has an n^2 sized branching program
- A candidate function : Given two $d \times d$ matrices A, B, $f(A,B) = 1$ iff and only $\mathrm{rowspace}(A) \cap \mathrm{rowspace}(B) \neq \{0\}$
- Not believed to be in **L**, but is in **P**
- Projective dimension of this function is just d, which is sub-linear in input size
- One can prove super linear bitpdim lower bounds for this function using Nechiporuk's method. But we would like to prove a super linear lower bound using a purely algebraic method.

- Can we come up with a super-linear lower bound which doesn't use Nechiporuk's method
  - Nechiporuk's method cannot prove better than n^2.
  - Sub-function count bottleneck : Let ρ fix $n-k$ bits of the n bits of a function. The number of different sub functions is $\min 2^{n-k}, 2^{2^k}$.
  - Element Distinctness has an n^2 sized branching program
- A candidate function : Given two $d \times d$ matrices A, B, $f(A,B) = 1$ iff and only $\mathrm{rowspace}(A) \cap \mathrm{rowspace}(B) \neq \{0\}$
- Not believed to be in **L**, but is in **P**
- Projective dimension of this function is just d, which is sub-linear in input size
- One can prove super linear bitpdim lower bounds for this function using Nechiporuk's method. But we would like to prove a super linear lower bound using a purely algebraic method.

- Can we come up with a super-linear lower bound which doesn't use Nechiporuk's method
  - Nechiporuk's method cannot prove better than n^2.
  - Sub-function count bottleneck : Let ρ fix $n-k$ bits of the n bits of a function. The number of different sub functions is $\min 2^{n-k}, 2^{2^k}$.
  - Element Distinctness has an n^2 sized branching program
- A candidate function : Given two $d \times d$ matrices A, B, $f(A,B) = 1$ iff and only $\mathrm{rowspace}(A) \cap \mathrm{rowspace}(B) \neq \{0\}$
- Not believed to be in **L**, but is in **P**
- Projective dimension of this function is just d, which is sub-linear in input size
- One can prove super linear bitpdim lower bounds for this function using Nechiporuk's method. But we would like to prove a super linear lower bound using a purely algebraic method.

- Can we come up with a super-linear lower bound which doesn't use Nechiporuk's method
  - Nechiporuk's method cannot prove better than n^2.
  - Sub-function count bottleneck : Let ρ fix $n-k$ bits of the n bits of a function. The number of different sub functions is $\min 2^{n-k}, 2^{2^k}$.
  - Element Distinctness has an n^2 sized branching program
- A candidate function : Given two $d \times d$ matrices A, B, $f(A,B) = 1$ iff and only $\mathrm{rowspace}(A) \cap \mathrm{rowspace}(B) \neq \{0\}$
- Not believed to be in **L**, but is in **P**
- Projective dimension of this function is just d, which is sub-linear in input size
- One can prove super linear bitpdim lower bounds for this function using Nechiporuk's method. But we would like to prove a super linear lower bound using a purely algebraic method.

- Can we come up with a super-linear lower bound which doesn't use Nechiporuk's method
  - Nechiporuk's method cannot prove better than n^2.
  - Sub-function count bottleneck : Let ρ fix $n-k$ bits of the n bits of a function. The number of different sub functions is $\min 2^{n-k}, 2^{2^k}$.
  - Element Distinctness has an n^2 sized branching program
- A candidate function : Given two $d \times d$ matrices A, B, $f(A,B) = 1$ iff and only $\mathrm{rowspace}(A) \cap \mathrm{rowspace}(B) \neq \{0\}$
- Not believed to be in **L**, but is in **P**
- Projective dimension of this function is just d, which is sub-linear in input size
- One can prove super linear bitpdim lower bounds for this function using Nechiporuk's method. But we would like to prove a super linear lower bound using a purely algebraic method.

- Can we come up with a super-linear lower bound which doesn't use Nechiporuk's method
    - Nechiporuk's method cannot prove better than n^2.
    - Sub-function count bottleneck : Let ρ fix $n-k$ bits of the n bits of a function. The number of different sub functions is $\min 2^{n-k}, 2^{2^k}$.
    - Element Distinctness has an n^2 sized branching program
- A candidate function : Given two $d \times d$ matrices A, B, $f(A, B) = 1$ iff and only $\mathrm{rowspace}(A) \cap \mathrm{rowspace}(B) \neq \{0\}$
- Not believed to be in **L**, but is in **P**
- Projective dimension of this function is just d, which is sub-linear in input size
- One can prove super linear bitpdim lower bounds for this function using Nechiporuk's method. But we would like to prove a super linear lower bound using a purely algebraic method.

- Can we come up with a super-linear lower bound which doesn't use Nechiporuk's method
  - Nechiporuk's method cannot prove better than n^2.
  - Sub-function count bottleneck : Let ρ fix $n-k$ bits of the n bits of a function. The number of different sub functions is $\min 2^{n-k}, 2^{2^k}$.
  - Element Distinctness has an n^2 sized branching program
- A candidate function : Given two $d \times d$ matrices A, B, $f(A,B) = 1$ iff and only $\mathrm{rowspace}(A) \cap \mathrm{rowspace}(B) \neq \{0\}$
- Not believed to be in **L**, but is in **P**
- Projective dimension of this function is just d, which is sub-linear in input size
- One can prove super linear bitpdim lower bounds for this function using Nechiporuk's method. But we would like to prove a super linear lower bound using a purely algebraic method.

Thank You

Q Questions ?