A Faster Algorithm for 0-1 Integer Programming (via Communication Complexity)

TSS - 07/12/22

Key Takeaways / Outline

> What is 0-1 IP?

> What is Orthogonal Vectors?

> (Main Focus) A 1-sided error MA protocol for an LTF

> Reduction from 0-1 IP to OV

> Background Story

0-1 IP.

0-1 IP.

> Simple gen. of CNF-SAT.

0-1 IP.

> Simple gen. of CNF-SAT.

> m "clauses" over n vars - is there an assignment satisfying all of them?

0-1 IP.

> Simple gen. of CNF-SAT.

> m "~~clauses~~ LTFs" over n vars - is there an assignment satisfying all of them?

> Linear Threshold Function: $F : \{0,1\}^n \to \{0,1\}$

$$F(x_1, \dots, x_n) = 1 \text{ iff } \sum_{i=1}^{n} w_i x_i \geq \theta$$

0-1 IP.

> Simple gen. of CNF-SAT.

> m "~~clauses~~ LTFs" over n vars – is there an assignment satisfying all of them?

> Linear Threshold Function: $F : \{0,1\}^n \to \{0,1\}$

$$F(x_1, \ldots, x_n) = 1 \text{ iff } \sum_{i=1}^{n} \underbrace{w_i}_{\text{"wts"}} x_i \geq \underbrace{\theta}_{\text{"threshold value"}}$$

More on LTFs

More on LTFs

> Note: same LTF can have multiple 'representations'

eg: $x_1 + x_2 + x_3 \leq 4 \iff x_1 + x_2 + x_3 \leq 100$

More on LTFs

> Note: same LTF can have multiple 'representations'

eg: $x_1 + x_2 + x_3 \leq 4 \iff x_1 + x_2 + x_3 \leq 100$

> In fact, any LTF on n vars can be 'represented' using $|w_i|, |\theta| \leq n^{O(n)} = 2^{O(n \log n)}$

0-1 IP (contd).

0-1 IP (contd).

- Formally, given m linear inequalities:

$$\left(\sum_{j=1}^{n} w_{1,j}\, x_j \geq \theta_1\right) \wedge \ldots \wedge \left(\sum_{j=1}^{n} w_{m,j}\, x_j \geq \theta_m\right)$$

0-1 IP (contd).

> Formally, given m linear inequalities:

$$\left(\sum_{j=1}^{n} w_{1,j}\, x_j \geq \theta_1\right) \wedge \ldots \wedge \left(\sum_{j=1}^{n} w_{m,j}\, x_j \geq \theta_m\right)$$

Is there a sat. assignment?

0-1 IP (contd).

> Formally, given m linear inequalities:

$$\left(\sum_{j=1}^{n} w_{1,j}\, x_j \geq \theta_1\right) \wedge \dots \wedge \left(\sum_{j=1}^{n} w_{m,j}\, x_j \geq \theta_m\right)$$

Is there a sat. assignment?

> Any clause for eg. $x_1 \vee \neg x_2 \vee x_3 \iff x_1 + (1 - x_2) + x_3 \geq 1$

$$\iff x_1 - x_2 + x_3 \geq 0$$

0-1 IP (contd).

> Formally, given m linear inequalities:

$$\left(\sum_{j=1}^{n} w_{1,j}\, x_j \geq \theta_1\right) \wedge \ldots \wedge \left(\sum_{j=1}^{n} w_{m,j}\, x_j \geq \theta_m\right)$$

Is there a sat. assignment?

> Any clause for eg. $x_1 \vee \neg x_2 \vee x_3 \iff x_1 + (1 - x_2) + x_3 \geq 1$

$$\iff x_1 - x_2 + x_3 \geq 0$$

> Shows NP-Completness of 0-1 IP

0-1 IP (contd).

> Formally, given m linear inequalities:

$$\left(\sum_{j=1}^{n} w_{1,j} x_j \geq \theta_1\right) \wedge \cdots \wedge \left(\sum_{j=1}^{n} w_{m,j} x_j \geq \theta_m\right)$$

Is there a sat. assignment?

> Assume all $|w_{i,j}|$, $|\theta|$ have bit-complexity $\leq M$

i.e. in the range $[-2^M, 2^M]$

0-1 IP (contd).

> Formally, given m linear inequalities:

$$\left(\sum_{j=1}^{n} w_{1,j}\, x_j \geq \theta_1\right) \wedge \dots \wedge \left(\sum_{j=1}^{n} w_{m,j}\, x_j \geq \theta_m\right)$$

Is there a sat. assignment?

> Assume all $|w_{i,j}|$, $|\theta|$ have bit-complexity $\leq M$

i.e. in the range $[-2^M, 2^M]$

> Brute-Force takes $2^n \cdot \text{poly}(n\, m\, M)$ time

0-1 IP (contd).

> Formally, given m linear inequalities:

$$\left(\sum_{j=1}^{n} w_{1,j}\, x_j \geq \theta_1\right) \wedge \cdots \wedge \left(\sum_{j=1}^{n} w_{m,j}\, x_j \geq \theta_m\right)$$

Is there a sat. assignment?

> Assume all $|w_{i,j}|$, $|\theta|$ have bit-complexity $\leq M$

i.e. in the range $[-2^M, 2^M]$

> Brute-Force takes $2^n \cdot \mathrm{poly}(n\, m\, M)$ time

> Note: only interested in $M = \mathrm{poly}(n)$.

Main Result

Main Result

> An algo to solve 0-1 IP (n vars, m clauses) in time

$$2^{n - \frac{n}{O(\log m)}}$$

Main Result

> An algo to solve 0-1 IP (n vars, m clauses) in time

$$2^{n - \frac{n}{O(\log m)}}$$ ← larger than $2^{0.99n}$.

Main Result

> An algo to solve 0-1 IP (n vars, m clauses) in time
$$2^{n - \frac{n}{O(\log m)}}$$ ← larger than $2^{0.99n}$.

> Matches the best known algo running time for CNF-SAT! (Schuler 2005)

Main Result

- An algo to solve 0-1 IP (n vars, m clauses) in time

$$2^{n - \frac{n}{O(\log m)}}$$ ← larger than $2^{0.99n}$.

- Matches the best known algo running time for CNF-SAT! (Schuler 2005)
- Expect much better? No! (SETH).

Digression: Orthogonal Vectors (OV) Problem

Digression: Orthogonal Vectors (OV) Problem

> I gave a TSS on it 2 years ago!

The Orthogonal Vectors Problem: Definition and Hardness Conjecture

OV: Problem Description

- Two vectors $u, v \in \{0,1\}^d$ (or binary strings of length d) are orthogonal if $\sum_{i\in[d]} u_i \cdot v_i = 0$
- Sum is considered over $\mathbb{R}$ (not $\mathbb{F}_2$)
- Equivalently, they are orthogonal if $\bigvee_{i\in[d]} u_i \wedge v_i = 0$ (there is no position at which both vectors have a 1)

Problem:

- Input: Two lists A, B of n d-dimensional $0-1$ vectors
- Output: "Accept" iff there is an orthogonal pair $(u, v) \in A \times B$

What is d?

- Obvious brute-force running time of $O(n^2 \cdot d)$
- If d is sufficiently smaller than n (for e.g., $d \ll \log n$), we must have redundant vector copies in each list, so we can weed them out first and then brute-force
- In particular, it follows that if $d \leq (1-\varepsilon)\log n$ for some *constant* $\varepsilon > 0$, then there is a $O(n^{2-\varepsilon} \cdot d) = \tilde{O}(n^{2-\varepsilon})$ time algo for $OV_{n,d}$
- Natural question: What about $d = c \log n$ for *any* constant c?
- Specifically, is there a **universal** constant $\varepsilon > 0$ so that for every constant c, $OV_{n,c \log n}$ can be solved in $\tilde{O}(n^{2-\varepsilon})$ time?
- **Orthogonal Vectors Conjecture (OVC) [R. Williams, Theor. Comp. Sci. '05]: No, there is not!**

Remarks:

- Think of this regime ($d = O(\log n)$) as the *smallest* possible for which $OV_{n,d}$ becomes interesting. OVC says that *even* in this case, "truly sub-quad. time" is impossible
- Note the order of quantifiers here! Because for a *given* constant c, $\tilde{O}(n^{2-\varepsilon_c})$ is possible, for ε_c depending on c

Connection to SETH: why we believe in OVC

Strong Exponential Time Hypothesis: Introduction

- $k - CNF - SAT$:
  - Input: Boolean variables $x_1, \dots, x_n$ and a formula in the conjunctive normal form i.e. of the form $C_1 \wedge \cdots \wedge C_m$ where each C_i is the logical OR of at most k variables (or their negations)
  - Output: "Accept" iff there exists an assignment to these variables on which this formula evaluates to 1
- Obvious $O(2^n \cdot mn)$ algorithm
- SETH asserts that we can't do much better for arbitrary k. More precisely:
- **SETH**: for every $\varepsilon > 0$, there is a k such that $k - CNF - SAT$ on n variables, m clauses cannot be solved in $2^{(1-\varepsilon)n} \cdot \mathrm{poly}(m)$ time
- Equivalently, if there is a $2^{(1-\varepsilon)n} \cdot \mathrm{poly}(m)$ time algorithm for some $\varepsilon > 0$ that can solve SAT on CNF Formulas (for all k) on n variables and m clauses , then SETH is false

SETH implies OVC!

- Contrapositive: Want to show that a "fast" algo for OV yields "fast" algo for SAT
- In other words, given a SAT instance on n variables $x_1, \dots, x_n$ and m clauses $C_1, \dots, C_m$, want to construct an OV instance on which we can apply this supposed "fast" algo
- This OV instance will have lists A, B of size $N = 2^{n/2}$, consisting of binary strings (vectors) of length m
- How to define these vectors? Use "split and list". Split variable set into halves: $\{x_1, \dots, x_{n/2}\}$ and $\{x_{n/2+1}, \dots, x_n\}$. A then consists of vectors u_α, where α is a partial assignment that assigns bits to the first half of variables. B consists of the set of v_β
- $u_\alpha(i) = \begin{cases} 1, \text{ if } \alpha \text{ does not satisfy } C_i \\ 0, \text{ otherwise} \end{cases}$ $\quad v_\beta(i) = \begin{cases} 1, \text{ if } \beta \text{ does not satisfy } C_i \\ 0, \text{ otherwise} \end{cases}$
- So u_α, v_β are orthogonal iff $\alpha \cup \beta$ satisfies all the clauses
- Note that it takes $O(2^{n/2} \cdot m)$ time to go from a given SAT instance to defining these lists A, B
- If there is an algo that solves $OV_{N,d}$ in $\tilde{O}(N^{2-\varepsilon})$ time, then SAT, after above reduction, on any k can be solved in time

$$O\left(2^{n/2} \cdot m + \left(2^{n/2}\right)^{2-\varepsilon}\right) = O\left(2^{\left(1-\frac{\varepsilon}{2}\right)n}\right)$$

- This contradicts SETH!

Fast Algorithm for OV

- Reminder: OVC states that there is no **universal** constant $\varepsilon > 0$ so that for every constant c, $OV_{n,c\log n}$ can be solved in $\tilde{O}(n^{2-\varepsilon})$ time
- But for a *given* c, one may still hope for $\tilde{O}(n^{2-\varepsilon_c})$ time
- And indeed, Abboud, R. Williams, and Yu (SODA '15) prove the following:
- **Theorem:** For Boolean vectors of dimension $d = c(n)\log n$, OV can be solved in $n^{\left\{2-\frac{1}{O(\log c(n))}\right\}}$ time by a randomized algorithm that is correct with high probability
- T. M. Chan and R. Williams (SODA '16) derandomize this:
- **Theorem:** There is a *deterministic* algorithm for $OV_{n,\, d=\, c(n)\; log\; n}$ that runs in $n^{\left\{2-\frac{1}{O(\log c(n))}\right\}}$ time, provided $d \le 2^{\{(\log n)^{\{o(1)\}}\}}$

All hail the polynomial method

- Checking if a pair of vectors $(x_i, y_j) \in A \times B$ is orthogonal is the formula

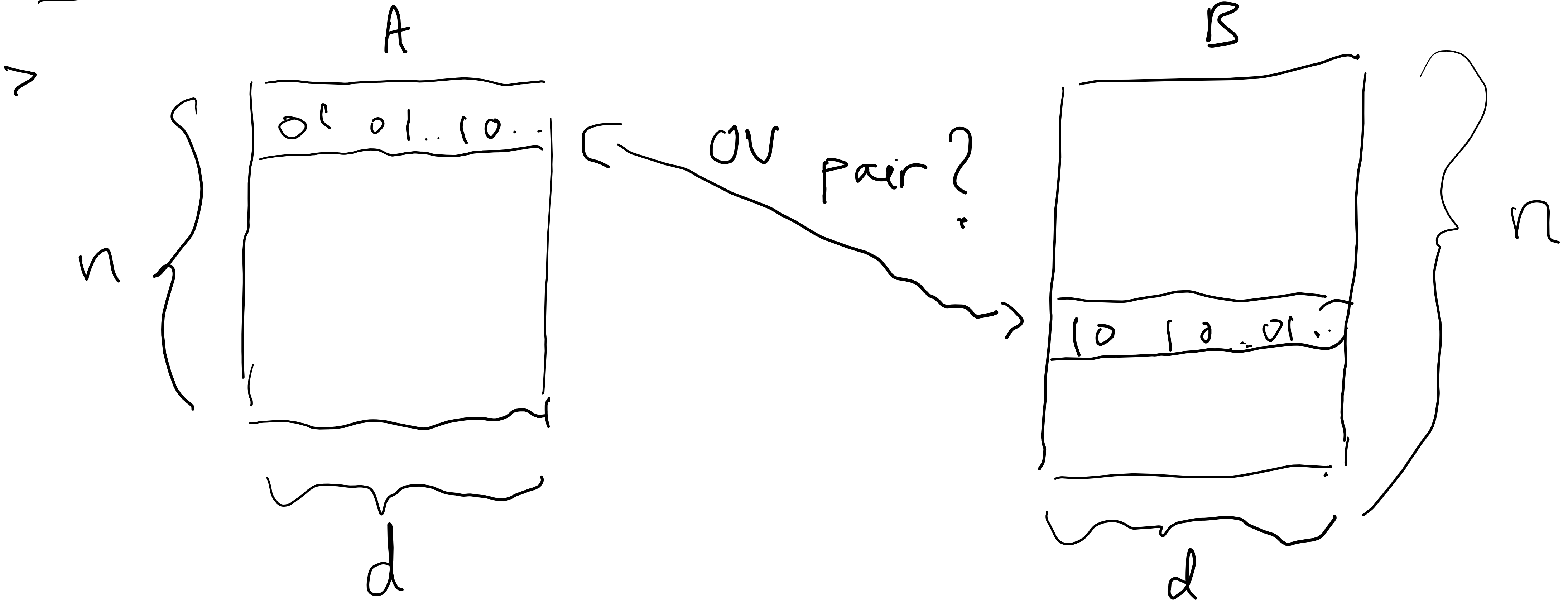
$$E(x_i, y_j) = \wedge_{k=1}^{d} (\neg x_i[k] \vee \neg y_j[k])$$

- Block them up into s parts $A_1, \ldots, A_s$ & $B_1, \ldots, B_s$, each containing n/s vectors (s tbd)
- Write down the formula that evaluates if there is an orthogonal pair in $A_i \times B_j$ (big OR of s^2 pairs of $E(\cdot,\cdot)$)
- Convert that formula into a polynomial, of not-too-large degree! How?
- Razborov & Smolensky in the 80s figured out low-degree "probabilistic" polynomials that "approximate" AND and OR functions really well
- Finally, set s accordingly to use "fast rectangular matrix multiplication" by Coppersmith

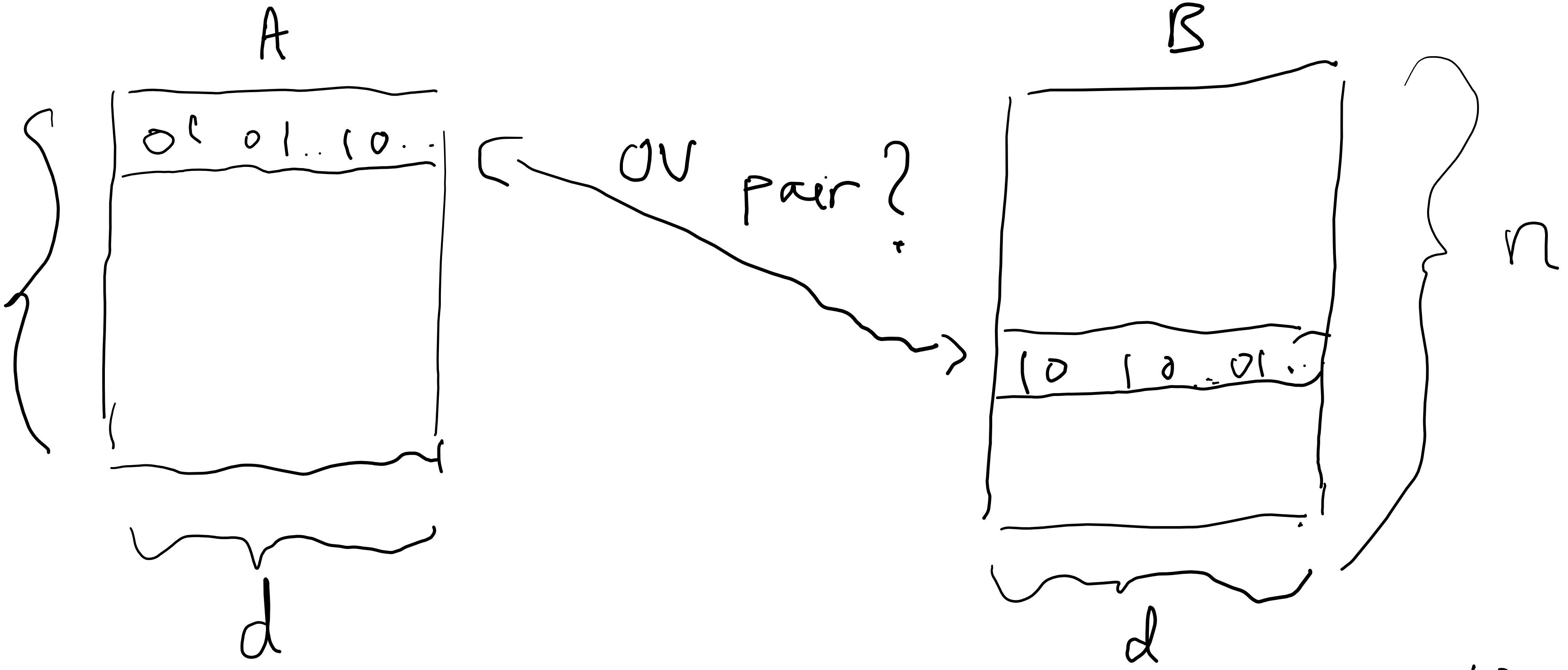
  (∃ constant $C \approx 0.172$ s.t. multiplication of an $N \times N^C$ matrix with an $N^C \times N$ matrix can be done using $\tilde{O}(N^2)$ arithmetic operations)

Orthogonal Vectors: The Upshot

Orthogonal Vectors: The Upshot

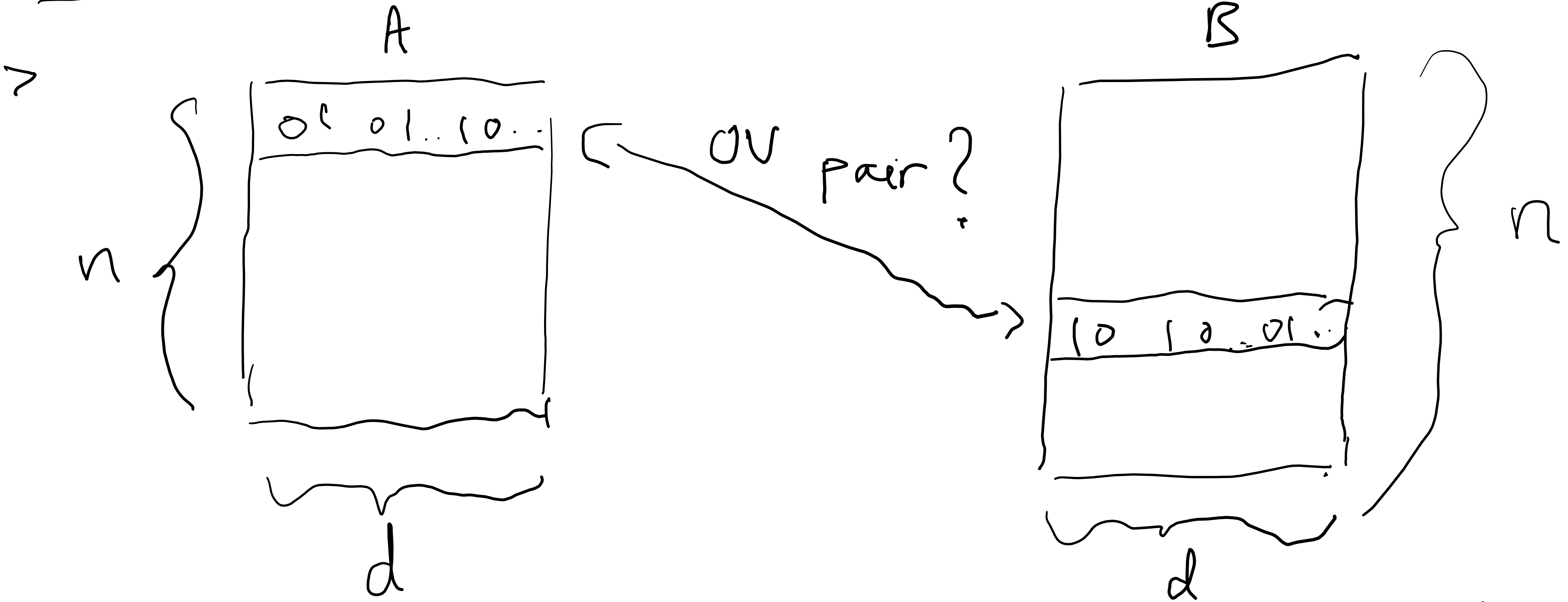


Orthogonal Vectors: The Upshot



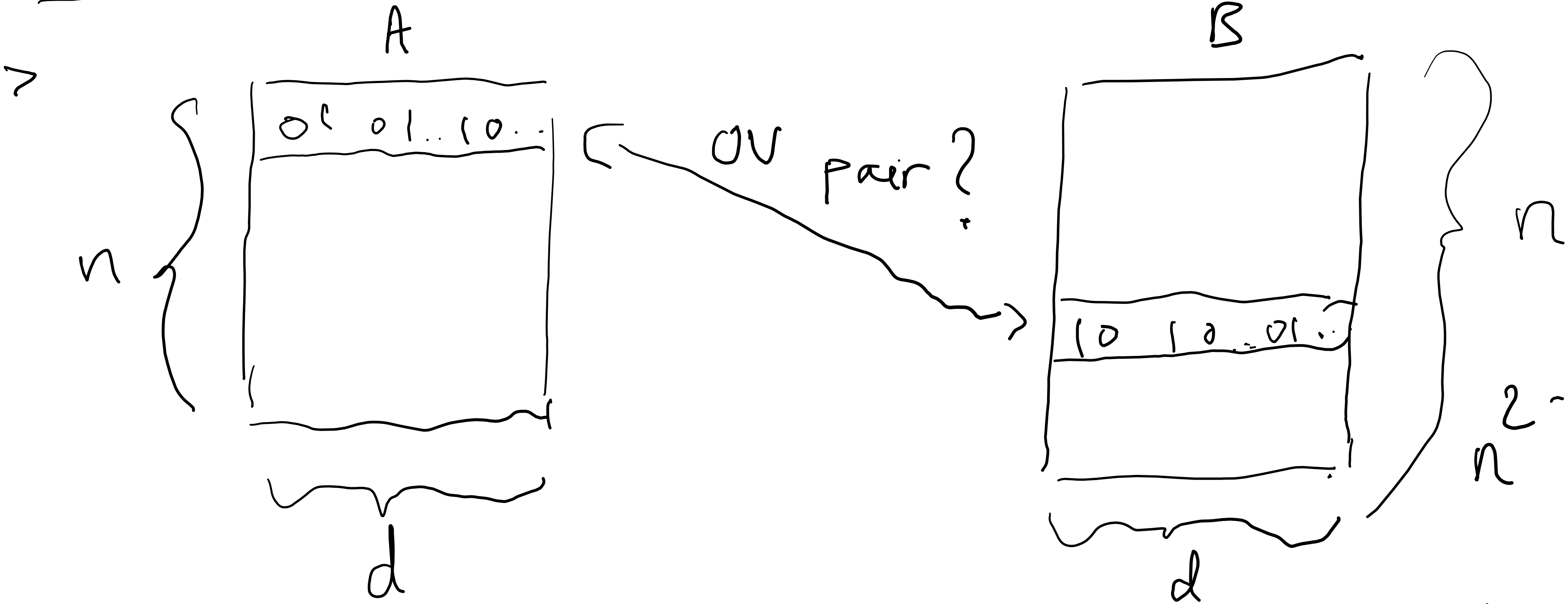
> Belief: very hard to do in truly sub-quad. time

Orthogonal Vectors: The Upshot



- Belief: very hard to do in truly sub-quad. time
- Pretty cool that a faster-than-brute-force exists

Orthogonal Vectors: The Upshot



$$n^{2 - O\left(\frac{d}{\log n}\right)}$$

> Belief: very hard to do in truly sub-quad. time

> Pretty cool that a faster-than-brute-force exists

Let's talk about Communication

Let's talk about Communication

> want to compute $F : \{0,1\}^{n/2} \times \{0,1\}^{n/2} \rightarrow \{0,1\}$.

Let's talk about Communication

> want to compute $F : \{0,1\}^{n/2} \times \{0,1\}^{n/2} \to \{0,1\}$.

given to Alice

given to Bob.

Let's talk about Communication

> want to compute $F : \{0,1\}^{n/2} \times \{0,1\}^{n/2} \to \{0,1\}$.

given to Alice

given to Bob.

Use as few bits of communication as possible to compute F.

Let's talk about Communication

> want to compute $F : \{0,1\}^{n/2} \times \{0,1\}^{n/2} \to \{0,1\}$.

given to Alice

given to Bob.

Use as few bits of communication as possible to compute F.

> Allowed to use public coins (ie, shared randomness) to compute F whp.

Simple Example: Equality.

Simple Example: Equality.

> Alice & Bob are given $n/2$-bit strings each. Determine if they are equal.

Simple Example: Equality.

> Alice & Bob are given $n/2$-bit strings each. Determine if they are equal.

> Takes $\Theta(n)$ bits deterministically.

Simple Example: Equality.

- Alice & Bob are given $n/2$-bit strings each. Determine if they are equal.
- Takes $\Theta(n)$ bits deterministically.
- Can we do better using randomness?

Simple Example: Equality

> Alice & Bob are given $n/2$-bit strings each. Determine if they are equal.
> Takes $\Theta(n)$ bits deterministically.
> Can we do better using randomness?

> Yes, hashing!
> Pick a random $z \in \{0,1\}^{n/2}$. They answer yes iff $\langle x, z \rangle = \langle y, z \rangle$ (over $\mathbb{F}_2$).

Simple Example: Equality.

> Alice & Bob are given $n/2$-bit strings each. Determine if they are equal.
> Takes $\Theta(n)$ bits deterministically.
> Can we do better using randomness?

> Yes, hashing!
> Pick a random $z \in \{0,1\}^{n/2}$. They answer yes iff $\langle x, z \rangle = \langle y, z \rangle$ (over $\mathbb{F}_2$).
> If $x = y$, succeed w.p. 1. If $x \neq y$, succeed w.p. $\geq 1/2$.

Simple Example: Equality.

> Alice & Bob are given $n/2$-bit strings each. Determine if they are equal.

> Takes $\Theta(n)$ bits deterministically.

> Can we do better using randomness?

> Yes, hashing!

> Pick a random $z \in \{0,1\}^{n/2}$. They answer yes iff $\langle x, z \rangle = \langle y, z \rangle$ (over $\mathbb{F}_2$).

> If $x = y$, succeed w.p. 1. If $x \neq y$, succeed w.p. $\geq 1/2$.

> Example of a '1-sided error randomized communication protocol' for Eq.

MA Communication Protocol.

MA Communication Protocol.

→ Now in addition to receiving i/ps $x, y \in \{0,1\}^n$ resp., they receive a 'proof string' $w \in \{0,1\}^k$.

MA Communication Protocol.

> Now in addition to receiving i/ps $x, y \in \{0,1\}^n$ resp., they receive a 'proof string' $w \in \{0,1\}^k$.

> They execute a randomized protocol Π (defd as a dist. over det. protocols) s.t:

MA Communication Protocol.

> Now in addition to receiving i/ps $x, y \in \{0,1\}^n$ resp., they receive a 'proof string' $w \in \{0,1\}^k$.

> they execute a randomized protocol π (defd as a dist. over det. protocols) s.t:

1. (Completeness) $F(x,y) = 1 \Rightarrow \exists w$ s.t. $\Pr_{\pi}[\pi(x,y,w) = 1] \geq 1-\varepsilon$

MA Communication Protocol.

> Now in addition to receiving i/ps $x, y \in \{0,1\}^n$ resp.,
they receive a 'proof string' $w \in \{0,1\}^k$.

> they execute a randomized protocol Π (defd as a dist. over det. protocols) s.t:

1. (Completeness) $F(x,y) = 1 \Rightarrow \exists w$ s.t. $\Pr_{\Pi}[\Pi(x,y,w) = 1] \geq 1-\varepsilon$
2. (Soundness) $F(x,y) = 0 \Rightarrow \forall w \; \Pr_{\Pi}[\Pi(x,y,w) = 1] \leq \varepsilon$

MA Communication Protocol.

> Now in addition to receiving i/ps $x, y \in \{0,1\}^n$ resp.,
they receive a 'proof string' $w \in \{0,1\}^k$.

> they execute a randomized protocol Π (defd as a dist. over det. protocols) s.t:

1. (Completeness) $F(x,y) = 1 \Rightarrow \exists w$ s.t. $\Pr_{\Pi}[\Pi(x,y,w) = 1] \geq 1-\varepsilon$
2. (Soundness) $F(x,y) = 0 \Rightarrow \forall w \ \Pr_{\Pi}[\Pi(x,y,w) = 1] \leq \varepsilon$

> We say an MA protocol has perfect completeness if 1. holds w.p. 1.

MA Communication Protocol.

> Now in addition to receiving i/ps $x, y \in \{0,1\}^n$ resp., they receive a 'proof string' $w \in \{0,1\}^k$.

> they execute a randomized protocol Π (defd as a dist. over det. protocols) s.t:

1. (Completeness) $F(x,y) = 1 \Rightarrow \exists w$ s.t. $\Pr_{\Pi}[\Pi(x,y,w)=1] \geq 1-\varepsilon$
2. (Soundness) $F(x,y) = 0 \Rightarrow \forall w \; \Pr_{\Pi}[\Pi(x,y,w)=1] \leq \varepsilon$

> We say an MA protocol has perfect completeness if 1. holds w.p. 1.

> Complexity := k + # of bits of communication (in worst case)

Task: Design a 1-sided error MA protocol for an LTF

Task: Design a 1-sided error MA protocol for an LTF

Recall: $f(x_1, \ldots, x_n) = 1 \iff \sum_{i=1}^{n} w_i x_i \geq \theta$

Task: Design a 1-sided error MA protocol for an LTF

Recall: $f(x_1, \ldots, x_n) = 1 \iff \sum_{i=1}^{n} w_i x_i \geq \theta$

> Alice receives $x_1, \ldots, x_{n/2}$, Bob receives $x_{n/2+1}, \ldots, x_n$.

Task: Design a 1-sided error MA protocol for an LTF

Recall: $f(x_1, \ldots, x_n) = 1 \iff \sum_{i=1}^{n} w_i x_i \geq \theta$

> Alice receives $x_1, \ldots, x_{n/2}$, Bob receives $x_{n/2+1}, \ldots, x_n$.

> Alice receives $\sigma : [n/2] \to \{0,1\}^{n/2}$, Bob $\tau : [n/2] \to \{0,1\}^{n/2}$

Task: Design a 1-sided error MA protocol for an LTF

Recall: $f(x_1, \dots, x_n) = 1 \iff \sum_{i=1}^{n} w_i x_i \geq \theta$

> Alice receives $\overbrace{x_1, \dots, x_{n/2}}^{X}$, Bob receives $\overbrace{x_{n/2+1}, \dots, x_n}^{Y}$.

> Alice receives $\sigma : [n/2] \to \{0,1\}^{n/2}$, Bob $\tau : [n/2] \to \{0,1\}^{n/2}$

> Alice computes $\sum_{x_j \in X} w_j \sigma(x_j)$, Bob $\sum_{x_j \in Y} w_j \tau(x_j)$

Task: Design a 1-sided error MA protocol for an LTF

Recall: $f(x_1, \ldots, x_n) = 1 \iff \sum_{i}^{n} w_i x_i \geq \theta$

> Alice receives $X = x_1, \ldots, x_{n/2}$, Bob receives $Y = x_{n/2+1}, \ldots, x_n$.

> Alice receives $\sigma : [n/2] \to \{0,1\}^{n/2}$, Bob $\tau : [n/2] \to \{0,1\}^{n/2}$

> Alice computes $\sum_{x_j \in X} w_j \sigma(x_j)$, Bob $\sum_{x_j \in Y} w_j \tau(x_j)$

> $F_\sigma := \theta - \sum_X w_i \sigma(x_i) + n \cdot 2^M$, $f_\tau = \sum_Y w_i \tau(x_i) + n \cdot 2^M$

Task: Design a 1-sided error MA protocol for an LTF

Recall: $f(x_1, \ldots, x_n) = 1 \iff \sum_{i=1}^{n} w_i x_i \geq \theta$

> Alice receives $\overbrace{x_1, \ldots, x_{n/2}}^{X}$, Bob receives $\overbrace{x_{n/2+1}, \ldots, x_n}^{Y}$.

> Alice receives $\sigma : [n/2] \to \{0,1\}^{n/2}$, Bob $\tau : [n/2] \to \{0,1\}^{n/2}$

> Alice computes $\sum_{x_j \in X} w_j \sigma(x_j)$, Bob $\sum_{x_j \in Y} w_j \tau(x_j)$

> $F_\sigma := \theta - \sum_X w_j \sigma(x_j) + n \cdot 2^M$, $F_\tau = \sum_Y w_j \tau(x_j) + n \cdot 2^M$

> Then, $F(\sigma \cup \tau) = 1 \iff F_\sigma \leq F_\tau$.

New Task: Design an MA Protocol for INequality.

New Task: Design an MA Protocol for INequality.

Stupid question: Given two numbers, how do you tell that one is larger than the other?

New Task: Design an MA Protocol for INequality.

Stupid question: Given two numbers, how do you tell that one is larger than the other?

No, really!: Given

1,087,352,600,501

& 1,087,352,613,009

New Task: Design an MA Protocol for INequality.

Stupid question: Given two numbers, how do you tell that one is larger than the other?

No, really!: Given

1,087,352,600,501

& 1,087,352,613,009

which one is larger? why? Give a 'short' proof.

New Task: Design an MA Protocol for INequality.

Stupid question: Given two numbers, how do you tell that one is larger than the other?

Everything to the left is 'equal'

No, really!: Given

1, 087, 352, 600, 501

& 1, 087, 352, 613, 009

which one is larger? why? Give a 'short' proof.

<u>Recall</u>: Alice knows F_0, Bob knows F_1.

<u>Recall</u>: Alice knows F_a, Bob knows F_i.

Both are R-bit numbers where $R = \lceil \log(n \cdot 2^{M+1}) \rceil$.

Recall: Alice knows F_σ, Bob knows F_τ.

Both are R-bit numbers where $R = \lceil \log(n \cdot 2^{M+1}) \rceil$.

Protocol (given randomly generated hashes $z_1, \ldots, z_t \in \{0,1\}^R$)

1. Prover supplies an index $i \in [R+1]$.

Recall: Alice knows F_σ, Bob knows F_T.

Both are R-bit numbers where $R = \lceil \log(n \cdot 2^{M+1}) \rceil$.

Protocol (given randomly generated hashes $z_1, \ldots, z_t \in \{0,1\}^R$)

1. Prover supplies an index $i \in [R+1]$.
2. If $i \in [R]$, Alice & Bob check if:
   (i) $F_\sigma(i) < F_T(i)$
   (ii) $\langle F_\sigma(<i), z_j(<i) \rangle = \langle F_T(<i), z_j(<i) \rangle \; \forall j \in [t]$.

Recall: Alice knows F_σ, Bob knows F_τ.

Both are R-bit numbers where $R = \lceil \log(n \cdot 2^{M+1}) \rceil$.

Protocol (given randomly generated hashes $z_1, \dots, z_t \in \{0,1\}^R$)

1. Prover supplies an index $i \in [R+1]$.
2. If $i \in [R]$, Alice & Bob check if:

(i) $F_\sigma(i) < F_\tau(i)$

(ii) $\langle F_\sigma(<i), z_j(<i) \rangle = \langle F_\tau(<i), z_j(<i) \rangle \ \forall j \in [t]$.

If both hold, output 'yes' (i.e. $\sigma \cup \tau$ sat. F). O/w, "no".

Recall: Alice knows F_σ, Bob knows F_I.

Both are R-bit numbers where $R = \lceil \log(n \cdot 2^{M+1}) \rceil$.

Protocol (given randomly generated hashes $z_1, \ldots, z_t \in \{0,1\}^R$)

1. Prover supplies an index $i \in [R+1]$.
2. If $i \in [R]$, Alice & Bob check if:

   (i) $F_\sigma(i) < F_I(i)$

   (ii) $\langle F_\sigma(<i), z_j(<i) \rangle = \langle F_I(<i), z_j(<i) \rangle \; \forall j \in [t]$.

   If both hold, output 'yes' (i.e. $\sigma \cup I$ sat. F). O/w, "no".
3. If $i = R+1$, only check (ii)

Recall: Alice knows F_σ, Bob knows F_T.

Both are R-bit numbers where $R = \lceil \log(n \cdot 2^{M+1}) \rceil$.

Protocol (given randomly generated hashes $z_1, \ldots, z_t \in \{0,1\}^R$)

1. Prover supplies an index $i \in [R+1]$.
2. If $i \in [R]$, Alice & Bob check if:

   (i) $F_\sigma(i) < F_T(i)$

   (ii) $\langle F_\sigma(<i), z_j(<i) \rangle = \langle F_T(<i), z_j(<i) \rangle \ \forall j \in [t]$.

   If both hold, output 'yes' (i.e. $\sigma \cup T$ sat. F). O/w, "no".
3. If $i = R+1$, only check (ii). Same

Note: This protocol has perfect completeness.

Note: This protocol has perfect completeness.

& By setting $t = \log(1/\varepsilon)$, we can 'boost' the error to ε.

__Note__: This protocol has perfect completeness.

& By setting $t = \log(1/\varepsilon)$, we can 'boost' the error to ε.

__Next question__: How to use this single protocol to __reduce__ 0-1 IP to OV?

Reduction to OV: A Sketch

Reduction to OV: A Sketch

Given a 0-1 IP instance: $\left(\sum_{j=1}^{n} w_{ij} x_j \geq \theta_i\right)_{i=1,\dots,m}$

Reduction to OV: A Sketch

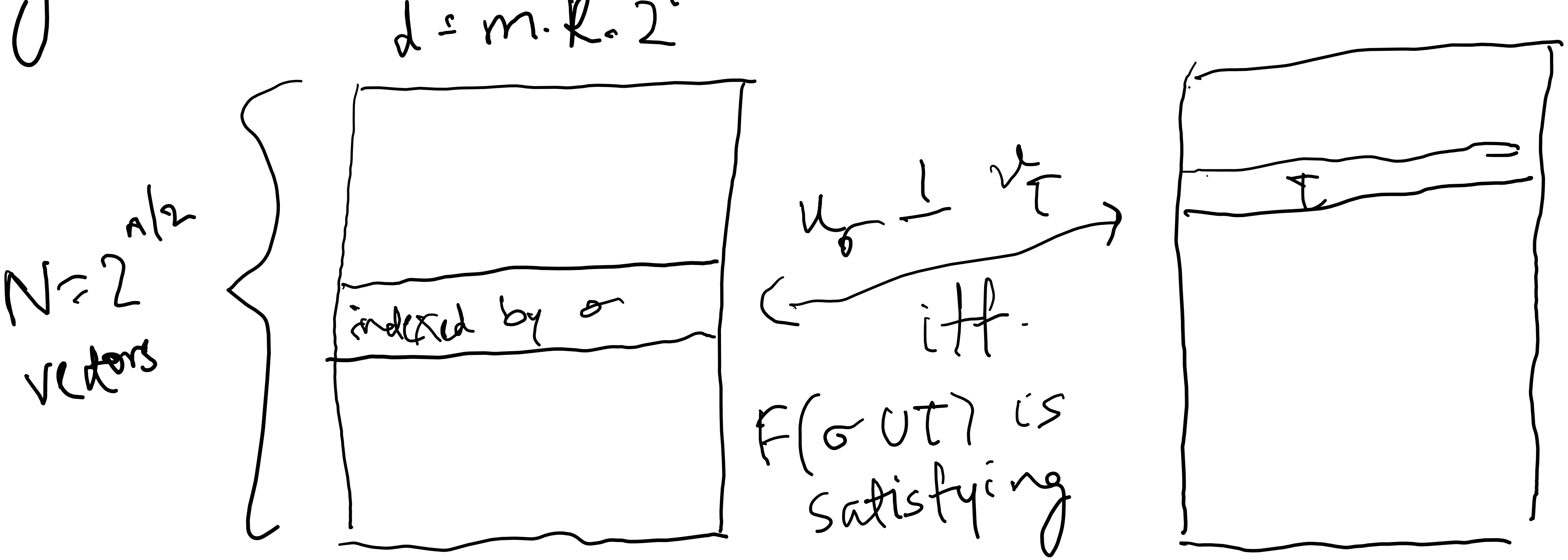
Given a 0-1 IP instance: $\left(\sum_{j=1}^{n} w_{ij} x_j \geq \theta_i\right)_{i=1,\dots,m}$

Generate an OV instance as follows:

Reduction to OV: A Sketch

Given a 0-1 IP instance: $\left(\sum_{j=1}^{n} w_{ij} x_j \geq \theta_i \right)_{i=1,\dots,m}$

Generate an OV instance as follows:



What are the $\mathcal{A} = m \cdot R \cdot 2^{t}$ coordinates?

What are the $d = m \cdot R \cdot 2^t$ coordinates?

> m clauses

> R diff. choices of index i (i.e. 'proof string')

> 2^t diff. choices of 'hash values'.

What are the $d = m \cdot R \cdot 2^t$ coordinates?

- m clauses
- R diff. choices of index i (i.e. 'proof string')
- 2^t diff. choices of 'hash values'.

- Then for $u_\sigma \in A$, $v_i \in B$ – they have a 1 in a common location iff $\sigma \cup T$ is NOT satisfying.

Finally, running the Chan-Williams fast algo for OV on this instance gives us the claimed

$2^{n - \frac{n}{O(\log m)}}$ run-time for 0-1 IP.

Thank You!