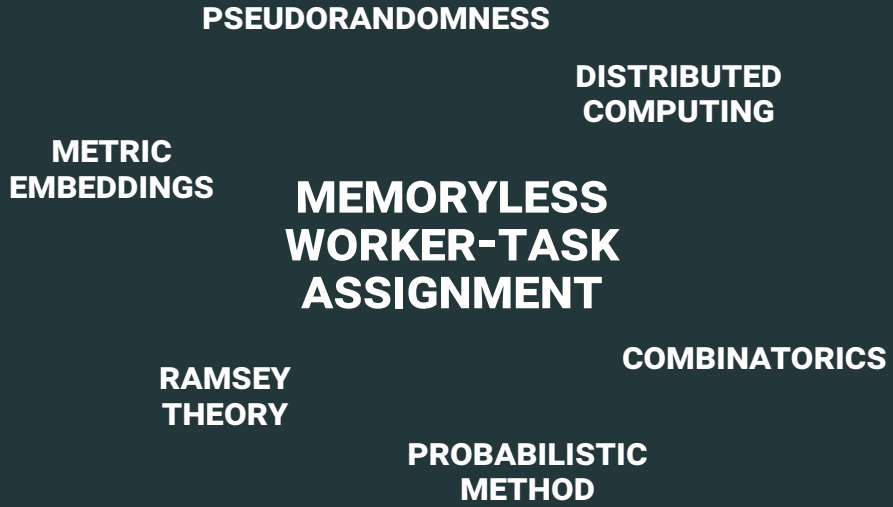
MEMORYLESS WORKER-TASK ASSIGNMENT WITH POLYLOGARITHMIC SWITCHING COST

Aaron Berger · William Kuszmaul · **Adam Polak** · Jonathan Tidor · Nicole Wein

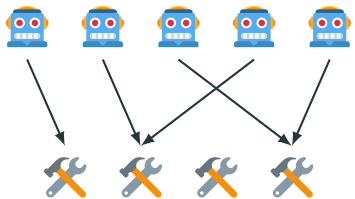

PSEUDORANDOMNESS
DISTRIBUTED COMPUTING
METRIC EMBEDDINGS
MEMORYLESS WORKER-TASK ASSIGNMENT
COMBINATORICS
RAMSEY THEORY
PROBABILISTIC METHOD

WORKER-TASK ASSIGNMENT

WORKER-TASK ASSIGNMENT

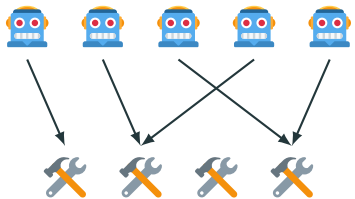
$$v = (1, \quad 2, \quad 0, \quad 2)$$

WORKER-TASK ASSIGNMENT



$v = (1, \quad 2, \quad 0, \quad 2)$

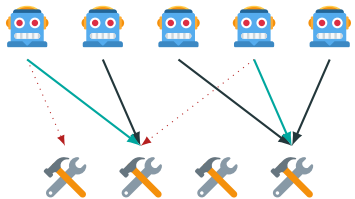
WORKER-TASK ASSIGNMENT



$v = (1, 2, 0, 2)$

$(0, 2, 0, 3)$

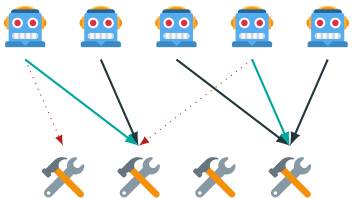
WORKER-TASK ASSIGNMENT



$v = (1, 2, 0, 2)$

$(0, 2, 0, 3)$

WORKER-TASK ASSIGNMENT

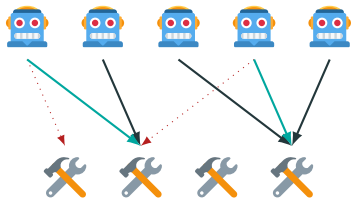


$v = (\sout{1}, \quad 2, \quad 0, \quad \sout{2})$

$(0, \quad 2, \quad 0, \quad 3)$

switching cost $= 2$

MEMORYLESS WORKER-TASK ASSIGNMENT



$v = (1, 2, 0, 2)$

$(0, 2, 0, 3)$

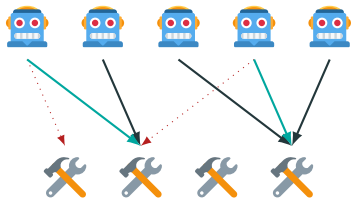
switching cost $= 2$

Memoryless workers [SSDL'17]

- know only current v

[SSDL'17] Su, Su, Dornhaus, Lynch

MEMORYLESS WORKER-TASK ASSIGNMENT



$v = (1, 2, 0, 2)$

$(0, 2, 0, 3)$

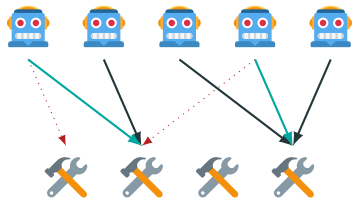
switching cost $= 2$

Memoryless workers [SSDL'17]

- know only current v
- implicitly communicationless

[SSDL'17] Su, Su, Dornhaus, Lynch

MEMORYLESS WORKER-TASK ASSIGNMENT



$v = (\cancel{1}, \quad \cancel{2}, \quad \cancel{0}, \quad \cancel{2})$

$(0, \quad 2, \quad 0, \quad 3)$

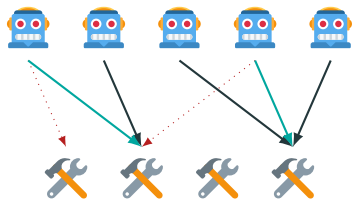
switching cost $= 2$

Memoryless workers [SSDL'17]

- know only current v
- implicitly communicationless
- e.g., internet of things, insects, . . .

[SSDL'17] Su, Su, Dornhaus, Lynch

MEMORYLESS WORKER-TASK ASSIGNMENT



$v = (\cancel{1}, \quad \cancel{2}, \quad 0, \quad \cancel{2})$

$(0, \quad 2, \quad 0, \quad 3)$

switching cost $= 2$

Memoryless workers [SSDL'17]

- know only current v
- implicitly communicationless
- e.g., internet of things, insects, . . .
- fault-tolerant, can reboot

[SSDL'17] Su, Su, Dornhaus, Lynch

FORMAL PROBLEM STATEMENT

Problem: design a *worker-task assignment function f* that

FORMAL PROBLEM STATEMENT

Problem: design a *worker-task assignment function* f that

given: demand vector $v \in \mathbb{Z}_{\geqslant 0}^{\#\text{tasks}}$ with $\sum_i v_i = \#\text{workers}$

outputs: assignment $f(v)$ of workers to tasks that satisfies v

FORMAL PROBLEM STATEMENT

Problem: design a *worker-task assignment function* f that

given: demand vector $v \in \mathbb{Z}_{\geqslant 0}^{\#\text{tasks}}$ with $\sum_i v_i = \#\text{workers}$

outputs: assignment $f(v)$ of workers to tasks that satisfies v

Goal: minimize the switching cost = $\max_{\substack{v_1, v_2 \\ |v_1 - v_2| = 1}} |f(v_1) - f(v_2)|$

FORMAL PROBLEM STATEMENT

Problem: design a *worker-task assignment function* f that

given: demand vector $v \in \mathbb{Z}_{\geqslant 0}^{\#\text{tasks}}$ with $\sum_i v_i = \#\text{workers}$

outputs: assignment $f(v)$ of workers to tasks that satisfies v

Goal: minimize the switching cost = $\max_{\substack{v_1, v_2 \\ |v_1 - v_2| = 1}} |f(v_1) - f(v_2)|$

$\ldots$ then $|f(v_1) - f(v_2)| \leqslant$ switching cost $\cdot\, |v_1 - v_2|$, for every v_1, v_2

A SIMPLE ALGORITHM

[SU, SU, DORNHAUS, LYNCH '17]

For i from 1 to #workers

worker $i \mapsto$ the first task with yet unmet demand

A SIMPLE ALGORITHM

[SU, SU, DORNHAUS, LYNCH '17]

For i from 1 to #workers

worker $i \mapsto$ the first task with yet unmet demand

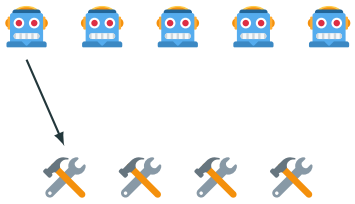
$v_1 = (2, \quad 1, \quad 2, \quad 0)$

A SIMPLE ALGORITHM

[SU, SU, DORNHAUS, LYNCH '17]

For i from 1 to #workers

worker $i \mapsto$ the first task with yet unmet demand



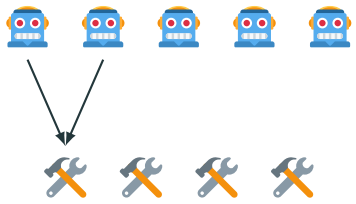
$v_1 = (2, 1, 2, 0)$

A SIMPLE ALGORITHM

[SU, SU, DORNHAUS, LYNCH '17]

For i from 1 to #workers

worker $i \mapsto$ the first task with yet unmet demand



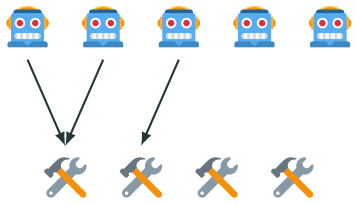
$v_1 = (2, 1, 2, 0)$

A SIMPLE ALGORITHM

[SU, SU, DORNHAUS, LYNCH '17]

For i from 1 to #workers

worker $i \mapsto$ the first task with yet unmet demand



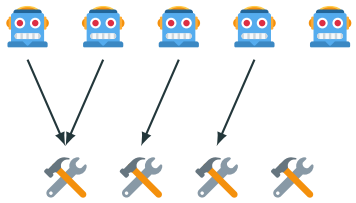
$v_1 = (2, 1, 2, 0)$

A SIMPLE ALGORITHM

[SU, SU, DORNHAUS, LYNCH '17]

For i from 1 to #workers

worker $i \mapsto$ the first task with yet unmet demand



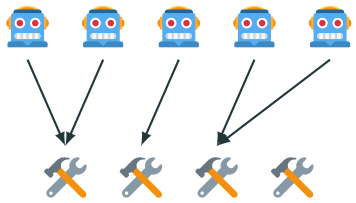
$v_1 = (2, 1, 2, 0)$

A SIMPLE ALGORITHM

[SU, SU, DORNHAUS, LYNCH '17]

For i from 1 to #workers

worker $i \mapsto$ the first task with yet unmet demand



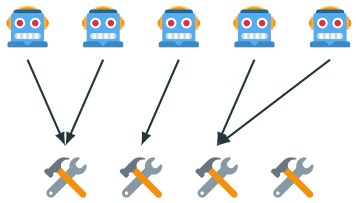
$v_1 = (2, 1, 2, 0)$

A SIMPLE ALGORITHM

[SU, SU, DORNHAUS, LYNCH '17]

For i from 1 to #workers

worker $i \mapsto$ the first task with yet unmet demand



When demands change…

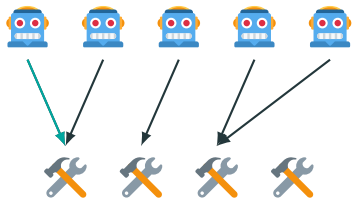
$v_1 = (2, 1, 2, 0)$

$v_2 = (1, 1, 2, 1)$

A SIMPLE ALGORITHM

[SU, SU, DORNHAUS, LYNCH '17]

For i from 1 to #workers
    worker $i \mapsto$ the first task with yet unmet demand



When demands change…

$v_1 = (2, 1, 2, 0)$

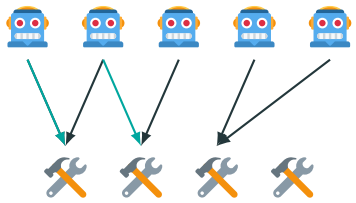
$v_2 = (1, 1, 2, 1)$

A SIMPLE ALGORITHM

[SU, SU, DORNHAUS, LYNCH '17]

For i from 1 to #workers

worker $i \mapsto$ the first task with yet unmet demand



When demands change…

$v_1 = (2, 1, 2, 0)$

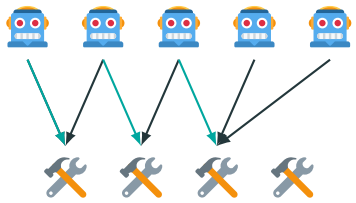
$v_2 = (1, 1, 2, 1)$

A SIMPLE ALGORITHM

[SU, SU, DORNHAUS, LYNCH '17]

For i from 1 to #workers

worker $i \mapsto$ the first task with yet unmet demand



When demands change…

$v_1 = (2, 1, 2, 0)$

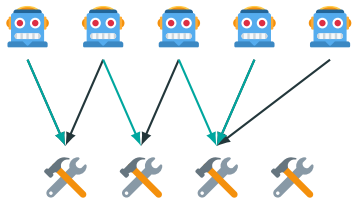
$v_2 = (1, 1, 2, 1)$

A SIMPLE ALGORITHM

[SU, SU, DORNHAUS, LYNCH '17]

For i from 1 to #workers

worker $i \mapsto$ the first task with yet unmet demand



When demands change…

$v_1 = (2, 1, 2, 0)$

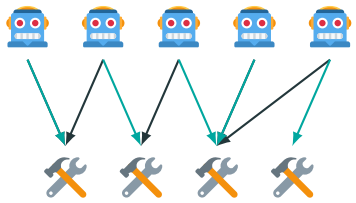
$v_2 = (1, 1, 2, 1)$

A SIMPLE ALGORITHM

[SU, SU, DORNHAUS, LYNCH '17]

For i from 1 to #workers

worker $i \mapsto$ the first task with yet unmet demand



When demands change…

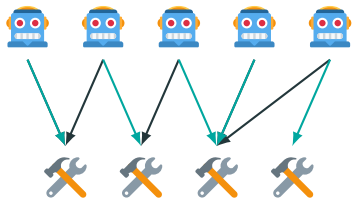
$v_1 = (2, 1, 2, 0)$

$v_2 = (1, 1, 2, 1)$

A SIMPLE ALGORITHM

[SU, SU, DORNHAUS, LYNCH '17]

For i from 1 to #workers
    worker $i \mapsto$ the first task with yet unmet demand



$v_1 = (2, 1, 2, 0)$

$v_2 = (1, 1, 2, 1)$

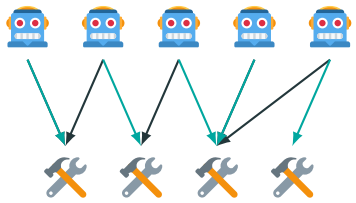
When demands change…

at most one worker moves
between tasks j and $j+1$,
for every $j = 1, 2, \ldots, \#\text{tasks} - 1$

A SIMPLE ALGORITHM

[SU, SU, DORNHAUS, LYNCH '17]

For i from 1 to #workers
    worker $i \mapsto$ the first task with yet unmet demand



$v_1 = (2, 1, 2, 0)$

$v_2 = (1, 1, 2, 1)$

When demands change…

at most one worker moves
between tasks j and $j+1$,
for every $j = 1, 2, \ldots, \#\text{tasks} - 1$

$\Longrightarrow$ switching cost $\leqslant \#\text{tasks} - 1$

SWITCHING COST BOUNDS

$t = \#\text{tasks} \quad w = \#\text{workers}$

$2 \leqslant \text{switching cost} \leqslant t - 1$

[Su, Su, Dornhaus, Lynch '17]

SWITCHING COST BOUNDS

$t = \#\text{tasks} \quad w = \#\text{workers}$

$2 \leqslant$ switching cost $\leqslant t - 1$		[Su, Su, Dornhaus, Lynch '17]
switching cost $\leqslant 2$	when $t \leqslant 4$ and $w \leqslant 6$	[Su, Su, Dornhaus, Lynch '17]
$3 \leqslant$ switching cost	when $t \geqslant 5$ and $w \geqslant 3$	[Su, Wein '20]
$4 \leqslant$ switching cost	when $t \geqslant 2 \uparrow\uparrow (w - 1)$	[Su, Wein '20]

SWITCHING COST BOUNDS

$t = \#\text{tasks} \quad w = \#\text{workers}$

$2 \leqslant$ switching cost $\leqslant t - 1$ [Su, Su, Dornhaus, Lynch '17]

switching cost $\leqslant 2$ when $t \leqslant 4$ and $w \leqslant 6$ [Su, Su, Dornhaus, Lynch '17]

$3 \leqslant$ switching cost when $t \geqslant 5$ and $w \geqslant 3$ [Su, Wein '20]

$4 \leqslant$ switching cost when $t \geqslant 2 \uparrow\uparrow (w - 1)$ [Su, Wein '20]

switching cost $\leqslant O(\log^2(wt))$ [Berger, Kuszmaul, **P.**, Tidor, Wein '22]

SWITCHING COST BOUNDS

$t = \#\text{tasks} \quad w = \#\text{workers}$

$2 \leqslant$ switching cost $\leqslant t - 1$ [Su, Su, Dornhaus, Lynch '17]

switching cost $\leqslant 2$ when $t \leqslant 4$ and $w \leqslant 6$ [Su, Su, Dornhaus, Lynch '17]

$3 \leqslant$ switching cost when $t \geqslant 5$ and $w \geqslant 3$ [Su, Wein '20]

$4 \leqslant$ switching cost when $t \geqslant 2 \uparrow\uparrow (w-1)$ [Su, Wein '20]

$w \leqslant$ switching cost $\leqslant O(\log^2(wt))$ [Berger, Kuszmaul, **P.**, Tidor, Wein '22]

↳ when $t \geqslant 2 \uparrow\uparrow \Omega(w)$

Remaining gap: $\Omega(\log^*(wt)) \leqslant$ switching cost $\leqslant O(\log^2(wt))$

OUR ALGORITHM

WITH POLYLOGARITHMIC SWITCHING COST

OUR ALGORITHM

Baby case: $v_i \leqslant 1$ (reduce to it via dummy w copies of each task)

OUR ALGORITHM

Baby case: $v_i \leqslant 1$ (reduce to it via dummy w copies of each task)

Each round assigns $1/100$ of yet unassigned workers $\implies$ $O(\log w)$ rounds

OUR ALGORITHM

Baby case: $v_i \leqslant 1$ (reduce to it via dummy w copies of each task)

Each round assigns $1/100$ of yet unassigned workers $\implies$ $O(\log w)$ rounds

Each round needs two *hash* functions: $h_w : [w] \to [b]$ and $h_t : [t] \to [b]$

$\uparrow$ $b = \#$unassigned workers

OUR ALGORITHM

Baby case: $v_i \leqslant 1$ (reduce to it via dummy w copies of each task)

Each round assigns $1/100$ of yet unassigned workers $\implies$ $O(\log w)$ rounds

Each round needs two *hash* functions: $h_w : [w] \to [b]$ and $h_t : [t] \to [b]$

$\hookleftarrow b = \#\text{unassigned workers}$

We use random functions h_w, h_t

$\implies$ probabilistic construction of a deterministic assignment function

OUR ALGORITHM

Baby case: $v_i \leqslant 1$ (reduce to it via dummy w copies of each task)

Each round assigns $1/100$ of yet unassigned workers $\implies$ $O(\log w)$ rounds

Each round needs two *hash* functions: $h_w : [w] \to [b]$ and $h_t : [t] \to [b]$

↑ $b = \#$unassigned workers

We use random functions h_w, h_t

$\implies$ probabilistic construction of a deterministic assignment function

↑ suceeds w.h.p., can be derandomzied with switching cost $\log^{O(1)}(wt)$

↑ using *strong dispersers*

SINGLE ROUND OF OUR ALGORITHM



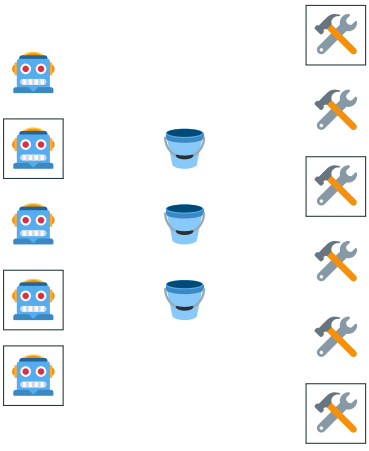




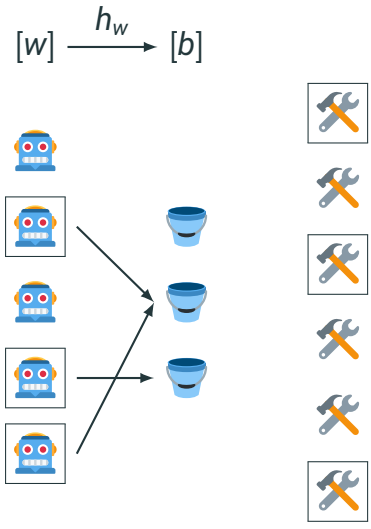




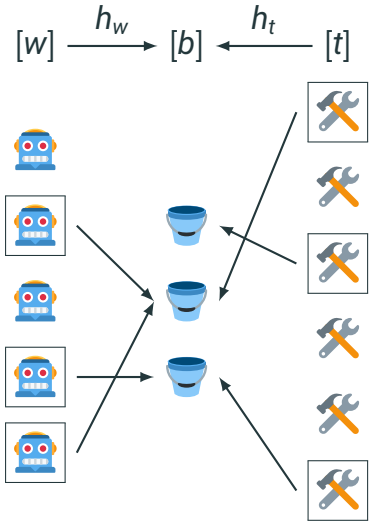
SINGLE ROUND OF OUR ALGORITHM



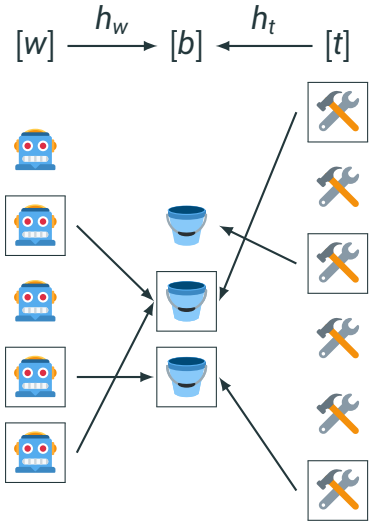
SINGLE ROUND OF OUR ALGORITHM



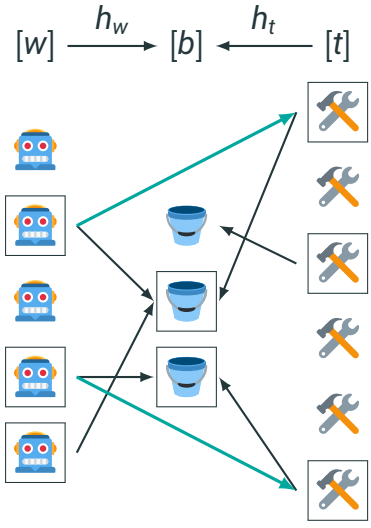
SINGLE ROUND OF OUR ALGORITHM



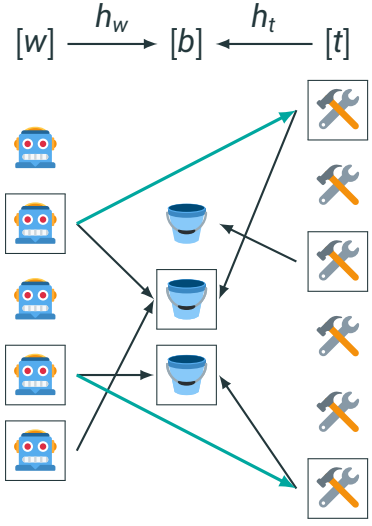
SINGLE ROUND OF OUR ALGORITHM



SINGLE ROUND OF OUR ALGORITHM



SINGLE ROUND OF OUR ALGORITHM

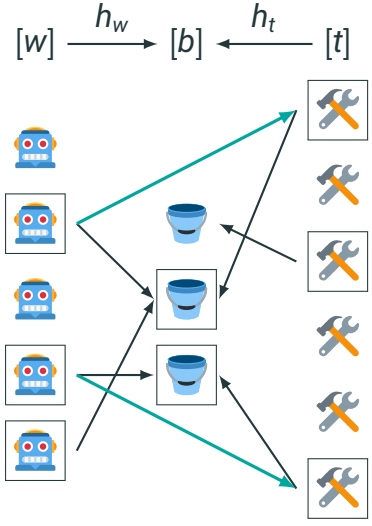


$\mathbb{P}(\text{const fraction of buckets used}) \geqslant e^{-b}$

repeat $O(\log wt)$ times to amplify

$\implies$ can union bound over $\binom{w}{b}\binom{t}{b}$ inputs

SINGLE ROUND OF OUR ALGORITHM



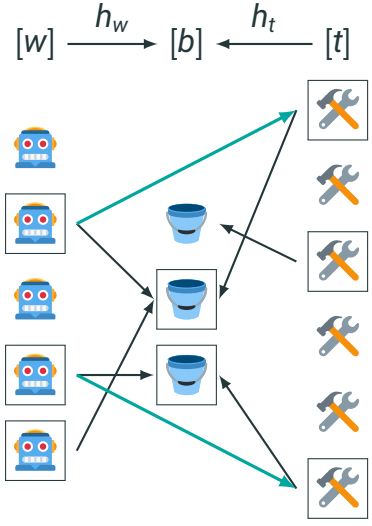
$\mathbb{P}(\text{const fraction of buckets used}) \geqslant e^{-b}$

repeat $O(\log wt)$ times to amplify

$\implies$ can union bound over $\binom{w}{b}\binom{t}{b}$ inputs

switching cost of a round $\leqslant 4$

SINGLE ROUND OF OUR ALGORITHM



$\mathbb{P}(\text{const fraction of buckets used}) \geqslant e^{-b}$

repeat $O(\log wt)$ times to amplify

$\implies$ can union bound over $\binom{w}{b}\binom{t}{b}$ inputs

switching cost of a round $\leqslant 4$

total switching cost $\leqslant O(\#\text{rounds})$

$\leqslant O(\log(w)\log(wt))$

APPLICATION TO

METRIC EMBEDDINGS

METRIC EMBEDDINGS

Worker-task assigment function f **embeds** $t \gg w$

***w*-sparse binary** vectors from
high-dimensional L1 space

into

low-dimensional Hamming space

METRIC EMBEDDINGS

Worker-task assigment function f **embeds** $t \gg w$

***w*-sparse binary** vectors from
high-dimensional L1 space $v = (0,1,0,1,1) \in \{0,1\}^t, ||v||_0 = w$

into

low-dimensional Hamming space

METRIC EMBEDDINGS

Worker-task assigment function f **embeds**

$t \gg w$

***w*-sparse binary** vectors from

high-dimensional L1 space

$$v = (0, 1, 0, 1, 1) \in \{0, 1\}^t, ||v||_0 = w$$

into

low-dimensional Hamming space

$$f(v) = (5, 2, 4) \in [t]^w$$

METRIC EMBEDDINGS

Worker-task assigment function f **embeds** $t \gg w$

***w*-sparse binary** vectors from

high-dimensional L1 space $v = (0, 1, 0, 1, 1) \in \{0, 1\}^t, ||v||_0 = w$

into

low-dimensional Hamming space $f(v) = (5, 2, 4) \in [t]^w$

$$||x - y||_1 \leqslant \text{Ham}(f(x), f(y)) \leqslant \underbrace{\text{switching cost}}_{= \textbf{distortion} \text{ of the embedding}} \cdot ||x - y||_1$$

METRIC EMBEDDINGS

Worker-task assigment function f **embeds** $t \gg w$

***w*-sparse binary** vectors from

high-dimensional L1 space $v = (0,1,0,1,1) \in \{0,1\}^t, ||v||_0 = w$

into

low-dimensional Hamming space $f(v) = (5,2,4) \in [t]^w$

$$||x-y||_1 \leqslant \mathrm{Ham}(f(x), f(y)) \leqslant \underbrace{\text{switching cost}}_{= \textbf{distortion} \text{ of the embedding}} \cdot ||x-y||_1$$

THANK YOU!