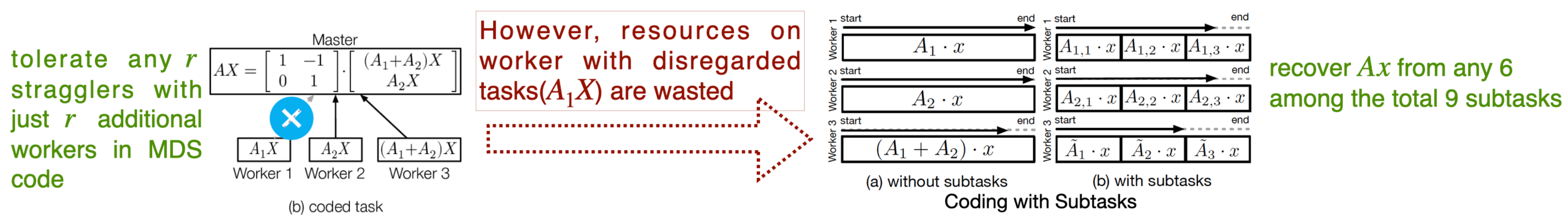
Sequence-Aware Coding for Leveraging Stragglers in Coded Matrix Multiplication

Presenter: **Yuchun Zou**[1] *Other Co-authors:* Xiaodi Fan[1] Pedro Soto[2] Xian Su[1] Jun Li[3]

[1]The Graduate Center, City University of New York [2]University of Oxford [3]Queens College and the Graduate Center, City University of New York

Background

- Modern distributed computing systems process large-scale matrix multiplication is to parallelize the computation into multiple tasks that are executed on different workers
- However, ***stragglers*** are inevitable, which lags the progress of the job significantly.
- *Leverage the partially completed tasks in the stragglers* is to dynamically adjust the workload of each worker
- **Coded tasks, Coding with Subtasks:** *t*olerate the same number of stragglers with fewer additional tasks and each worker then can upload the results of completed sub-tasks instead of the whole task.

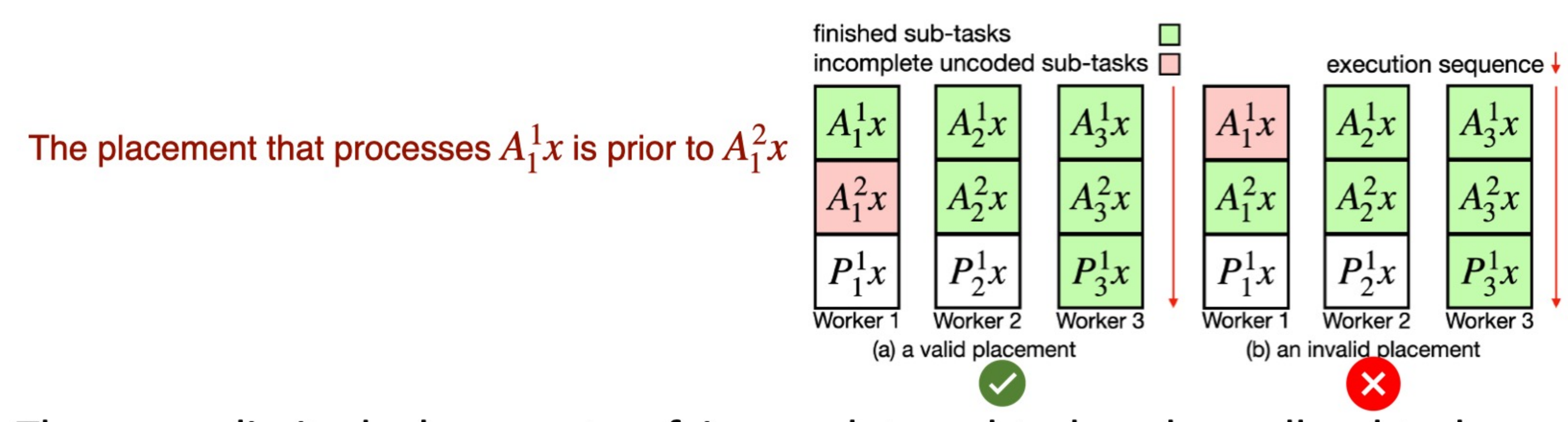


recover Ax from any 2 of the 3 tasks

Each $\tilde{A}_j$ encoded from all four submatrices of A, $j = 1,2,3$

Motivation

- The execution sequence of subtasks is typically in a fixed order, a subtask executed earlier naturally has a lower probability to be incomplete than another one executed later.



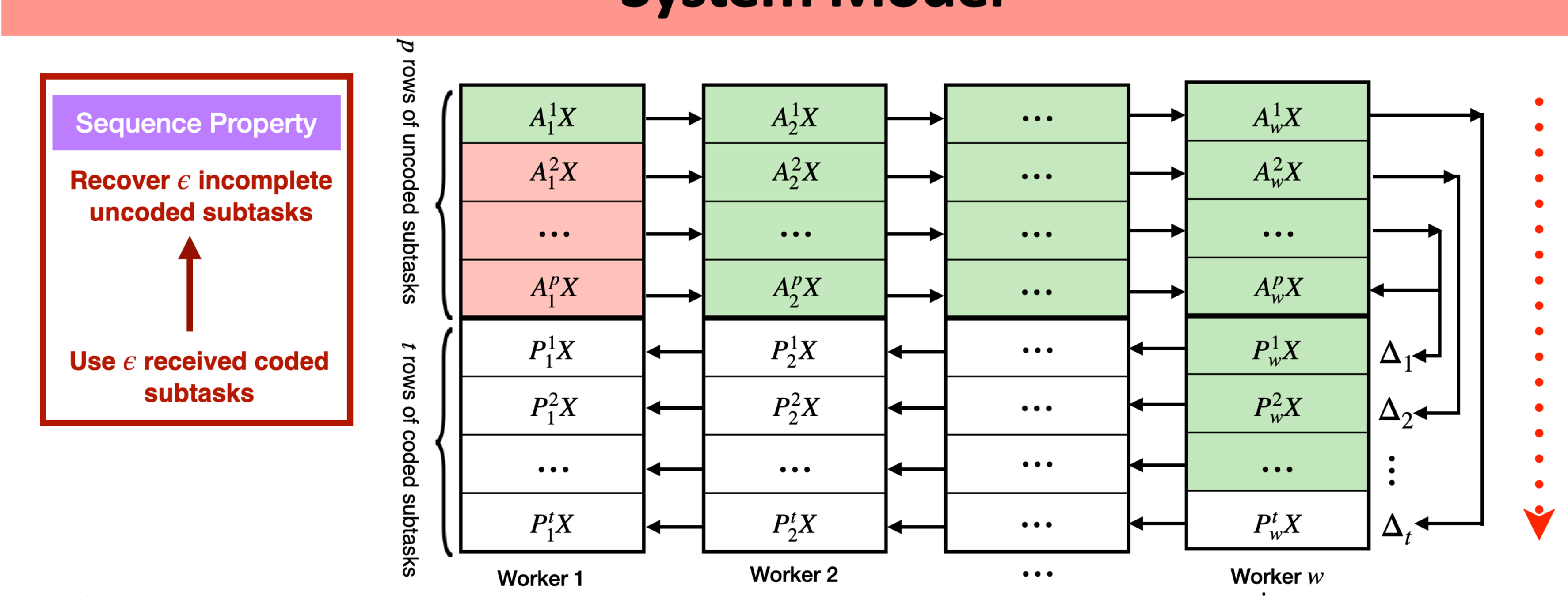
- There are limited placements of incomplete subtasks when all subtasks are executed sequentially. Subtask with a higher probability to be incomplete should be encoded into coded subtask with a higher priority.

Only possible placements $\Longrightarrow$ enabling the coded sub-tasks to recover the result from valid placements of incomplete uncoded sub-tasks only $\Longrightarrow$ reduce encoding complexity

Proposed Sequence-Aware Coding scheme

- Respect the **temporal sequence** in which sub-tasks are executed
- **Adjust the encoding locality** and use a subset of whole uncoded subtasks for encoding
- **Leverage partial results** from stragglers with lower encoding complexity

System Model

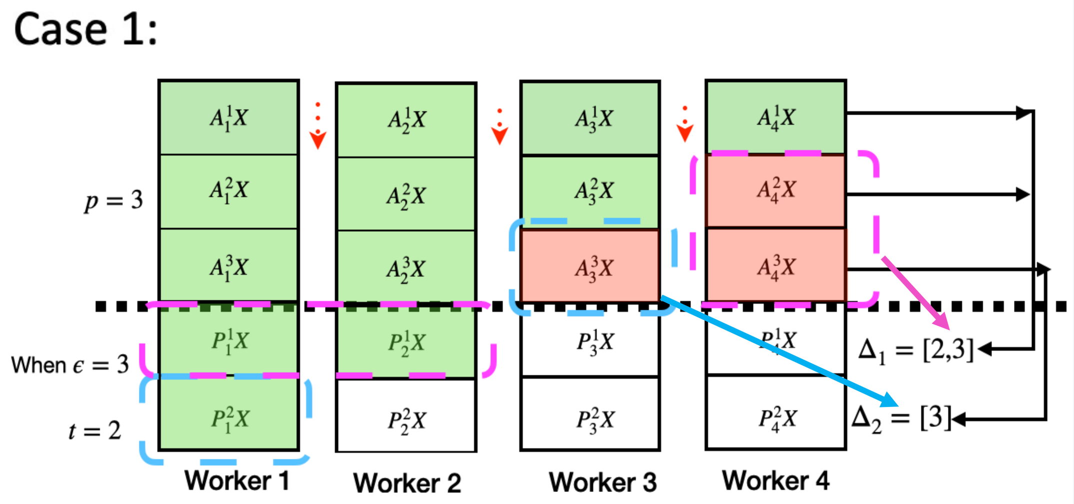


- $\Delta = (\Delta_1, \Delta_2, \cdots, \Delta_t)$, where Δ_i is the set of rows of uncoded subtasks where $P_j^i, (j-1,\cdots,w)$ are encoded from
- w: nums of worker p: nums of uncoded subtasks t: nums of coded subtasks ϵ: nums of received coded subtasks/ nums of uncompleted uncoded subtasks (w,p,t,ϵ) $\rightarrow$ **Sequence-Aware Coding(SAC)** $\rightarrow$ **Construction of Δ**

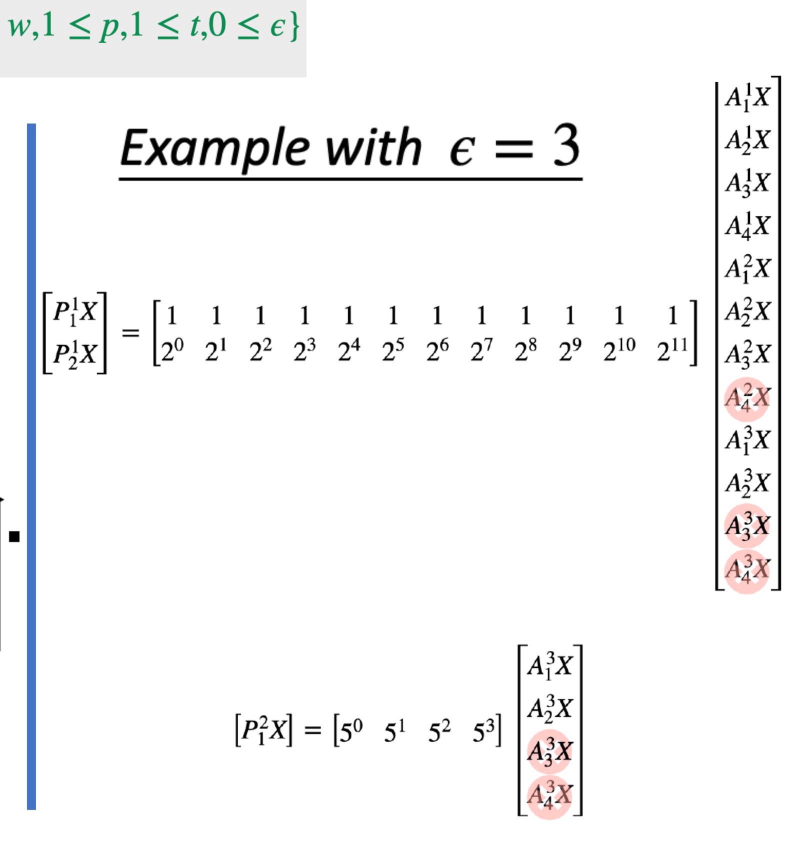
$$\{(w,p,t,\epsilon) \mid \lceil \frac{\epsilon}{t} \rceil + \lceil \frac{\epsilon}{p} \rceil \le w, 1 \le w, 1 \le p, 1 \le t, 0 \le \epsilon\}$$

Example with (w,p,t,ϵ)=(4,3,2,3)

Case 1:



Example with $\epsilon = 3$

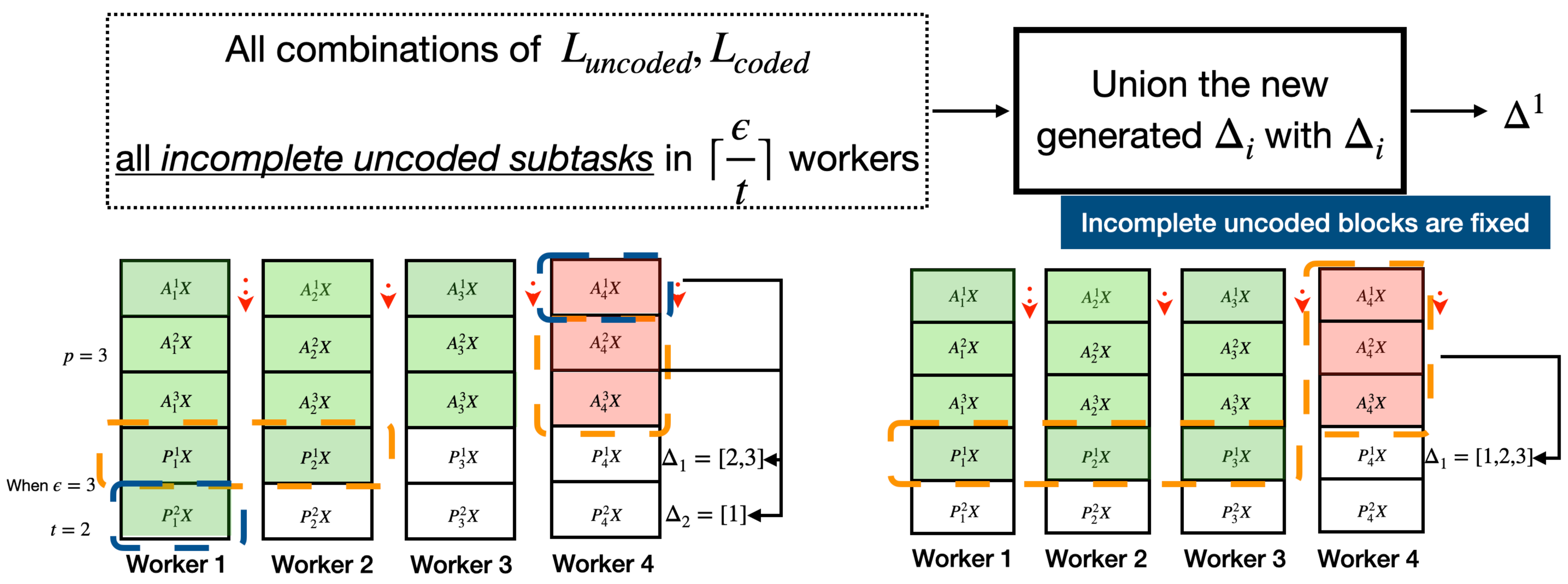


Follow-up Works and Further Direction

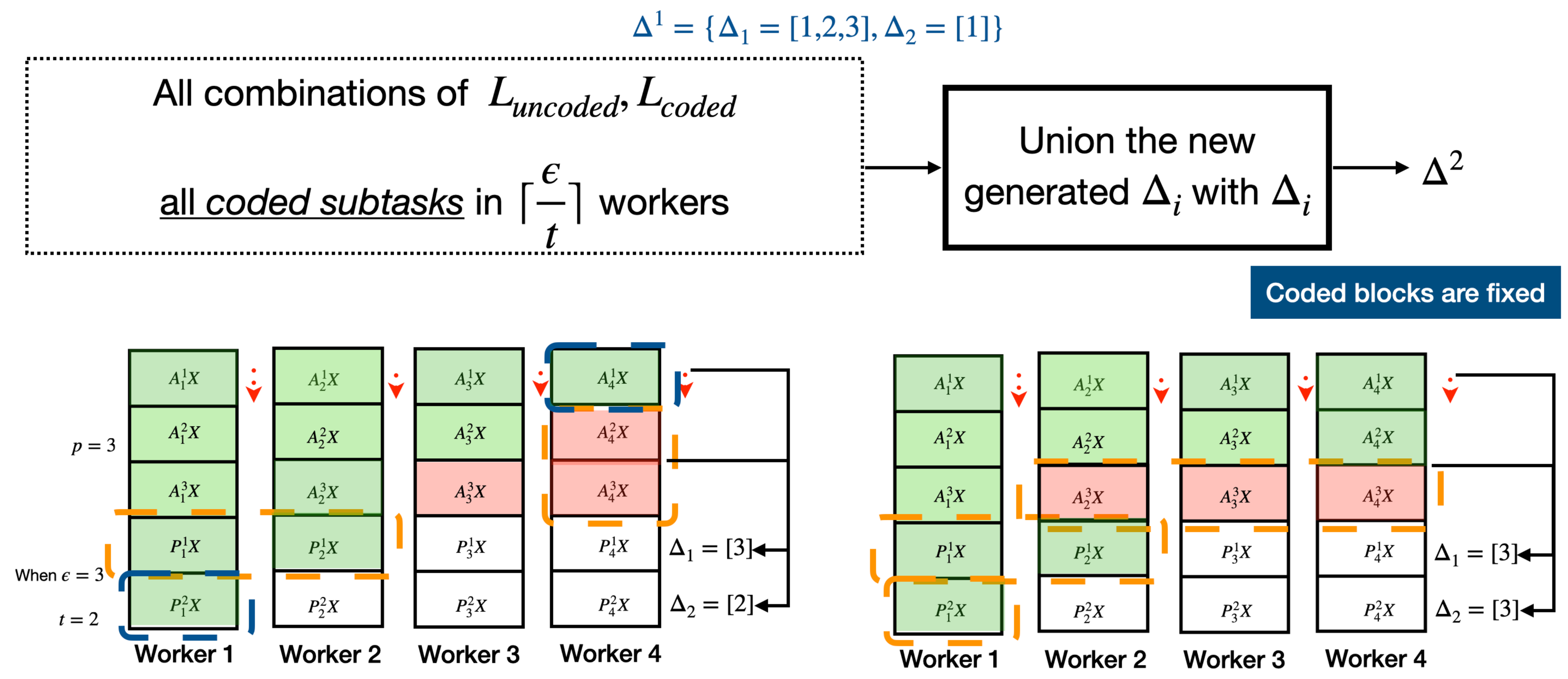
- In ongoing project based on SAC, we further have allowed a flexible parameter to control how likely a coded subtask can be used to recover the overall result at θ—arbitrary recoverability and extended it to matrix-matrix multiplication
- Next step: Speed up and improve SAC-I, a brute-force searching algorithm

Construction of Δ with SAC-I: Search Algorithm

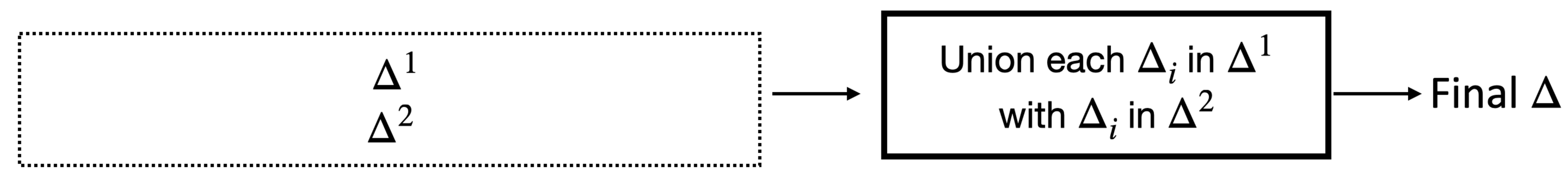
- Idea: received **coded sub-tasks** with **the highest row index** $\Longrightarrow$**incomplete uncoded sub-tasks** with **the lowest row index**



$\Delta^1 = \{\Delta_1 = [1,2,3], \Delta_2 = [1]\}$



$\Delta^2 = \{\Delta_1 = [3], \Delta_2 = [2,3]\}$

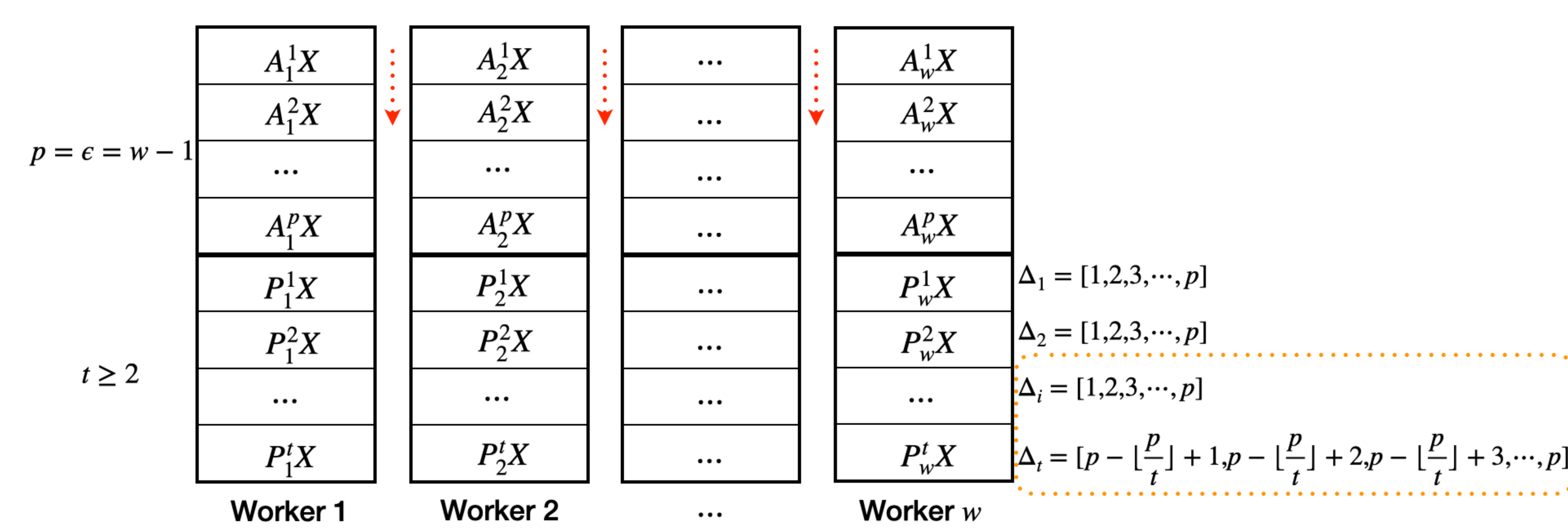


$\Delta_1 = [2,3] \cup \Delta_1 = [3] \cup \Delta_1 = [3] \cup \Delta_1 = [1,2,3] \implies \Delta^1 = [1,2,3]$

$\Delta_2 = [1] \cup \Delta_2 = [2] \cup \Delta_2 = [3] \implies \Delta^2 = [1,2,3]$

For $(w,p,t,\epsilon) = (4,3,2,3)$, $\Delta = \{\Delta^1 = [1,2,3], \Delta^2 = [1,2,3]\}$

Construction of Δ with SAC-II (Math Formulas)



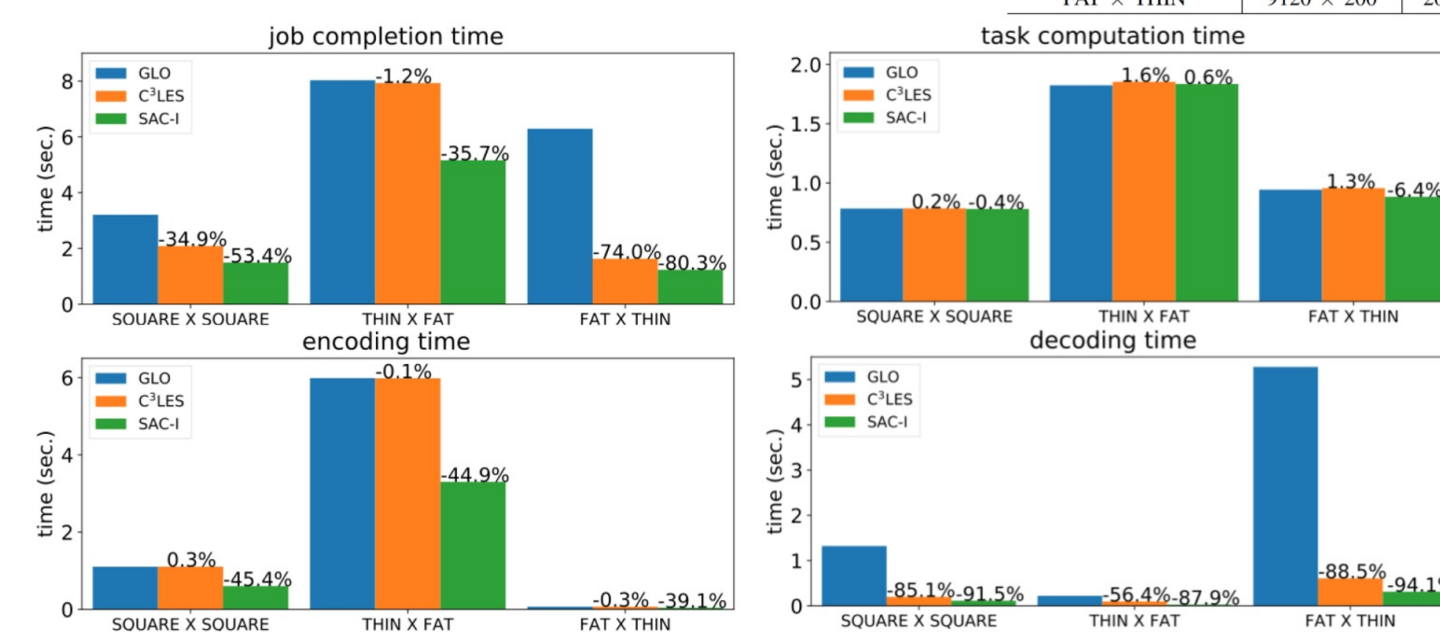
System model of SAC-II with **math formulas** for when $p = \epsilon = w-1, t \ge 2$, and $w \ge 1$

Δ generation is:

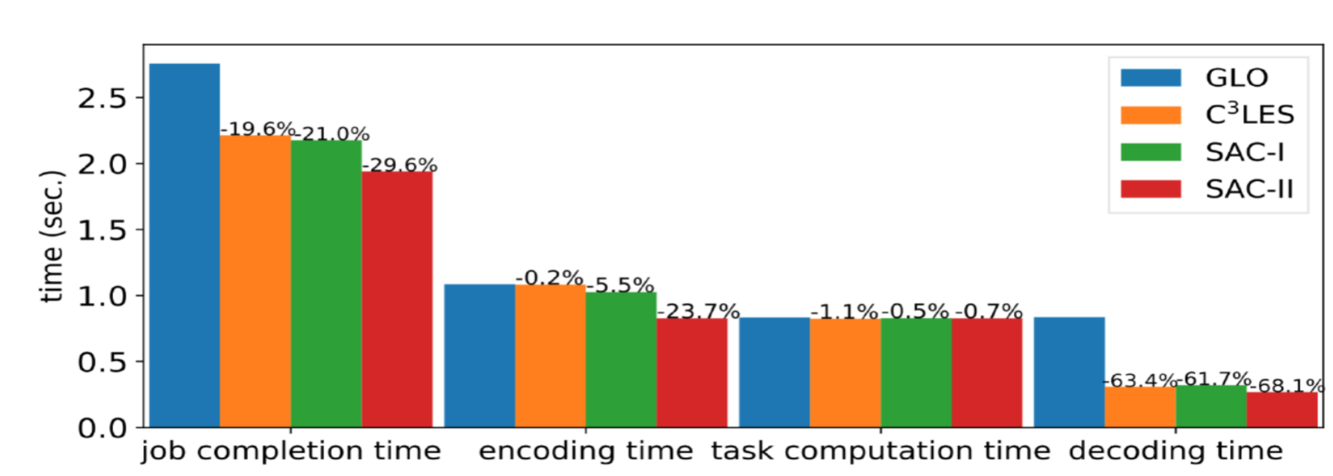
- optimal only when $t = 2, p = \epsilon = w-1, w \ge 3$.
- feasible for all $t > 2, p = \epsilon = w-1, w \ge 1$.

Simulation Results

Configurations of job	A	X
SQUARE × SQUARE	4800 × 4800	4800 × 4800
THIN × FAT	240 × 500000	500000× 240
FAT × THIN	9120 × 200	200 × 9120



Job completion time, encoding time, task computation time, and decoding time of SAC-I with $(w,p,t,\epsilon) = (16,15,2,7)$ and GLO, and C^3LES with $(w,p,t) = (16,15,2)$.



Job completion time, encoding time, task computation time, and decoding time comparison of SAC-I, SAC-II, GLO, and C^3LES, where $(w,p,t) = (16,15,2)$ and $\epsilon = 15$ for SAC-I and SAC-II.