

# Online Job Scheduler for Fault-tolerant Quantum Multiprogramming

Shin Nishio<sup>1,2,\*</sup>, Ryo Wakizaka<sup>3</sup>, Daisuke Sakuma<sup>2</sup>, Yosuke Ueno<sup>4</sup>, Yasunari Suzuki<sup>5</sup>

1: University College London, 2: Keio University, 3: Kyoto University, 4: RIKEN, 5: NTT Corporation

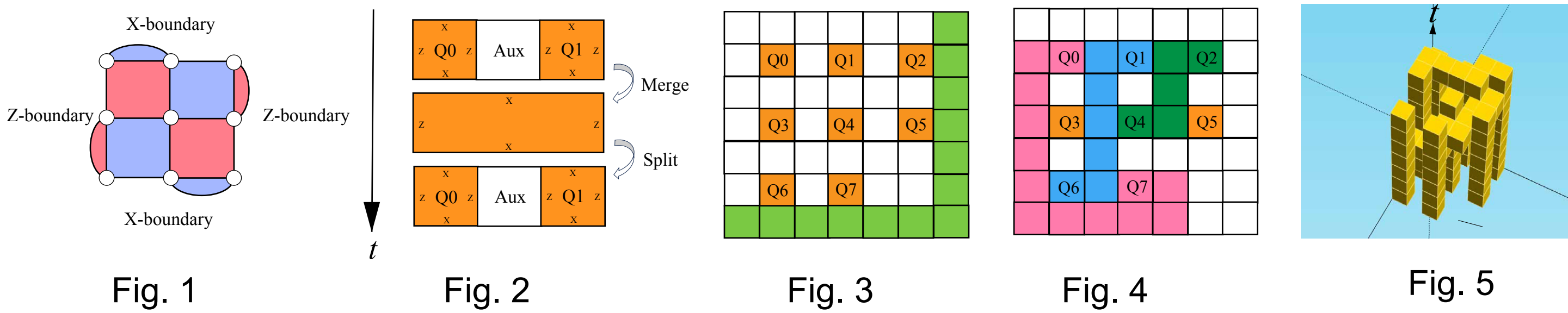


Fault-tolerant quantum computers are expected to be offered as cloud services due to their significant resource and infrastructure requirements. Quantum multiprogramming, which runs multiple quantum jobs in parallel, is a promising approach to maximize the utilization of such systems. A key challenge in this setting is the need for an online scheduler capable of handling jobs submitted dynamically while other programs are already running.

In this study, we formulate the online job scheduling problem for fault-tolerant quantum computing systems based on lattice surgery and propose an efficient scheduler to address it. To meet the responsiveness required in an online environment, our scheduler approximates lattice surgery programs, originally represented as polycubes, by using simpler cuboid representations. This approximation enables efficient scheduling while improving overall throughput. In addition, we incorporate a defragmentation mechanism into the scheduling process, demonstrating that it can further enhance QPU utilization.

## Fault-tolerant quantum computing

**Fault-tolerant quantum computing (FTQC)** with **quantum error-correcting codes** (QECCs) has been proposed to realize practical quantum applications in noisy quantum devices. (Rotated) **surface codes** [1] have attracted attention as one of the codes suitable for FTQC (Fig. 1). A method called **lattice surgery** has been proposed to implement the logic gates on surface codes (Fig. 2). To realize a quantum circuit using lattice surgery, it is necessary to arrange the qubits, and this arrangement is called a **floorplan** (Fig. 3). floorplan consists of logical data qubits as well as logical auxiliary qubits that are used as pathways for lattice surgery. (Fig. 4). Given floorplan, a quantum circuit can be compiled as a lattice surgery sequence on a two-dimensional plane. By arranging gate rows that cannot be executed in parallel, the quantum program in the end becomes a **polycube** in space-time 3D space (Fig. 5).



## Quantum Multiprogramming

**Quantum multiprogramming (QMP)**, (parallelly execute multiple jobs on a single computing system) works well with client-server model.

- 👍 Pros
  - Increase the throughput of the system
- 👎 Cons
  - Side effects of sharing resources (noise, competing for magic states, etc.)
  - (Classical) Computing resource for scheduling and resource distribution

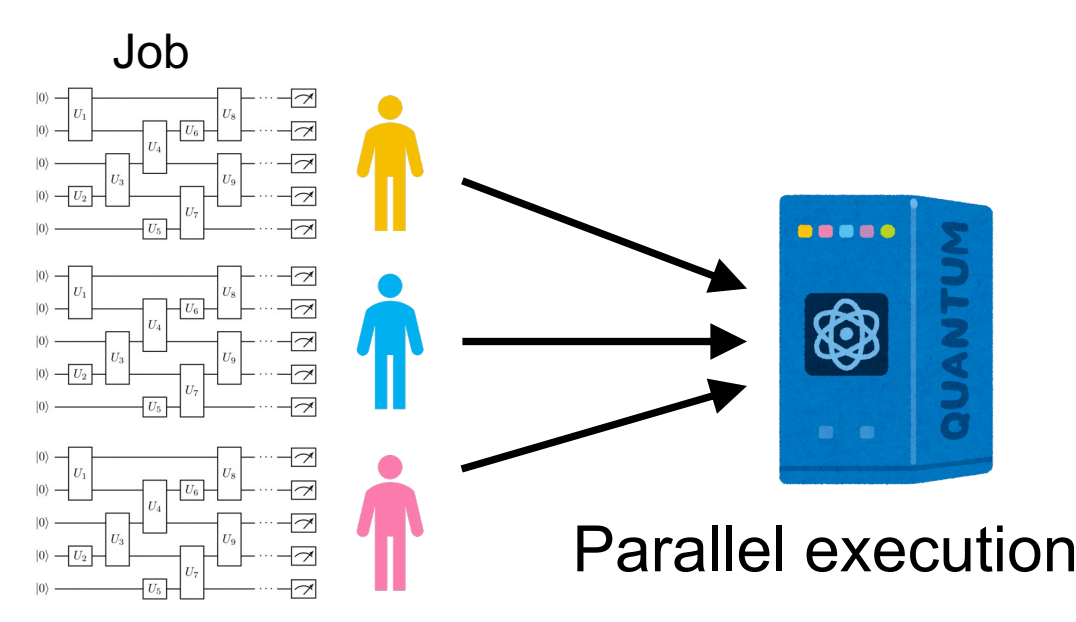


Fig. 6 Quantum multiprogramming

The quantum multiprogramming method for FTQC proposed so far [2] is not online - new jobs cannot be added while a job is running. Also, it does use polycube-based ILP which is not responsive enough

## Online Scheduling Problem

We formalized the online scheduling problem for fault-tolerant quantum multiprogramming as an **integer linear program (ILP)** problem.

**Problem 1 (Online scheduling):** Given a program sequence  $P_1, \dots, P_n$ , where  $P_i$  is the  $i$ -th request represented by a polycube. The goal is to find the schedule  $S_1, \dots, S_n$  that minimizes the total execution time  $\max_i \{t_{\text{fin}}(S_i(P_i))\}$  where each schedule  $S_i$  contains the schedule information, which consists of the 3D positions, rotation and whether the program will be flipped. We denote by  $t_{\text{fin}}(P)$  the time at which program  $P$  completes its execution.

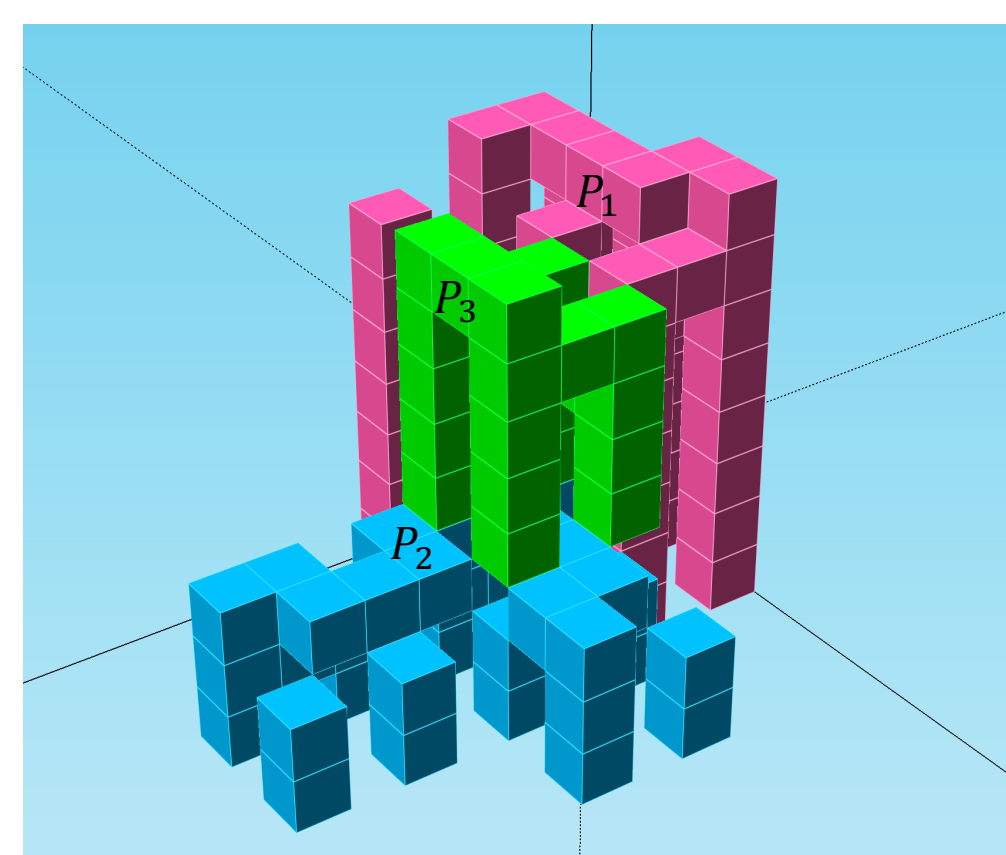


Fig. 7 An example of schedule for  $P_1, P_2, P_3$

**Problem 2 (ILP problem based on cuboids):**

$$\begin{aligned}
 & \text{minimize } v & (1) \\
 & \text{subject to} & (2) \\
 & \forall i, j \in \{1, \dots, n\}. \\
 & a_{ij} + a_{ji} + b_{ij} + b_{ji} + c_{ij} + c_{ji} \geq 1 & (3) \\
 & w_i + W(a_{ij} - 1) \leq x_j - x_i & (4) \\
 & h_i + H(b_{ij} - 1) \leq y_j - y_i & (5) \\
 & l_i + L(c_{ij} - 1) \leq z_j - z_i & (6) \\
 & \forall i \in \{1, \dots, n\}. \\
 & x_i + w_i \leq W & (7) \\
 & y_i + h_i \leq H & (8) \\
 & z_i + l_i \leq L & (9) \\
 & z_i + l_i \leq v & (10)
 \end{aligned}$$

- $W/H$ : width / height of processor
- $X_i, Y_i, Z_i$ : size of a program  $P_i$
- $x_i, y_i, z_i$ : coordinate of  $P_i$ 's schedule (positive int)
- Termination time for all programs
 
$$T := \sum_i Z_i$$
- (decision variable)  $a_{ij}, b_{ij}, c_{ij}$ : Collision between program  $P_i$  and  $P_j$

## Corner-Greedy Scheduler & Defragmentation

We propose a flow for online scheduling (Fig. 8). It hires a suboptimal scheduler which we call **corner-greedy** (Fig. 9) and **defragmentation** for quantum jobs (Fig. 10) inspired by classical memory defragmentation [3].

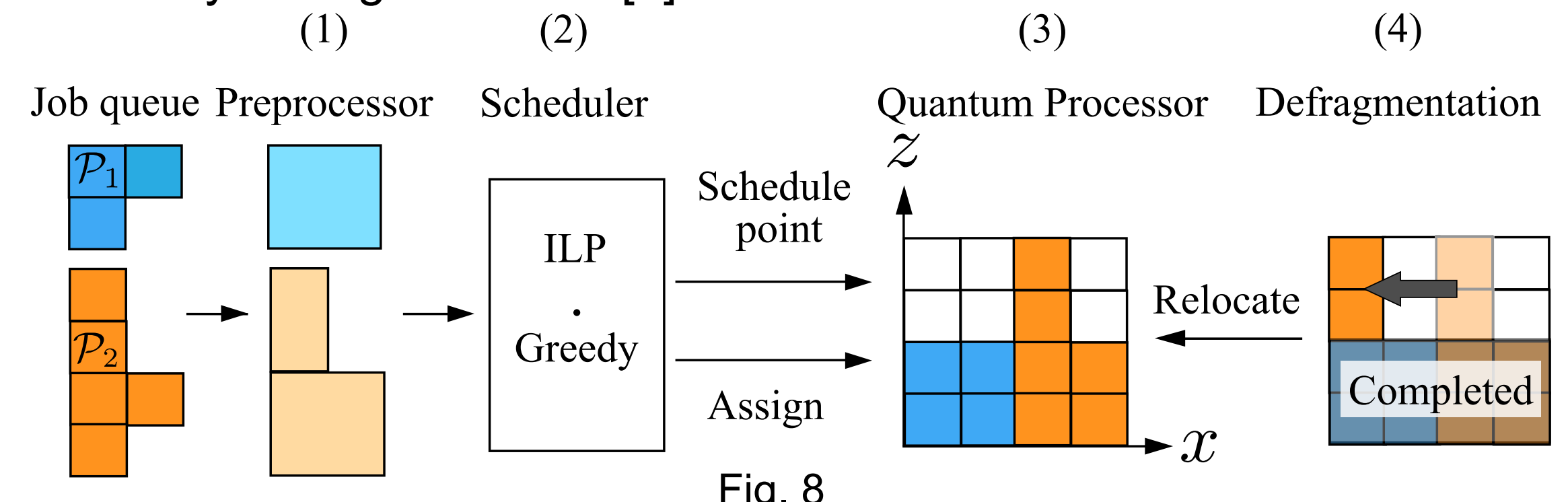


Fig. 8

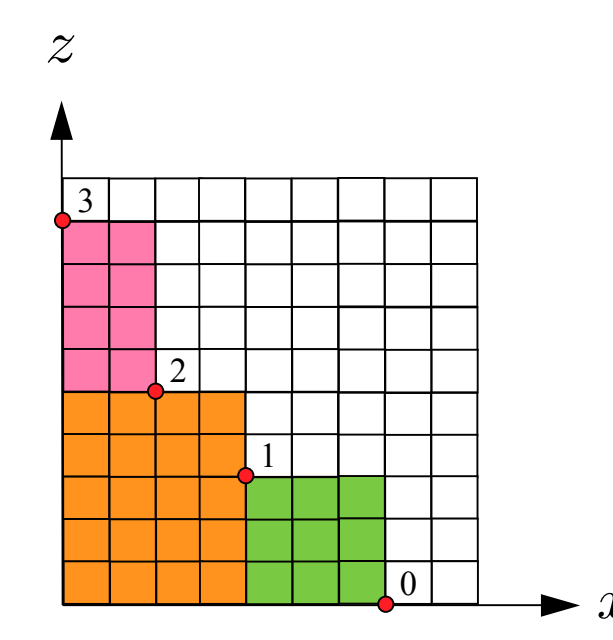


Fig. 9 Corner-Greedy Scheduler

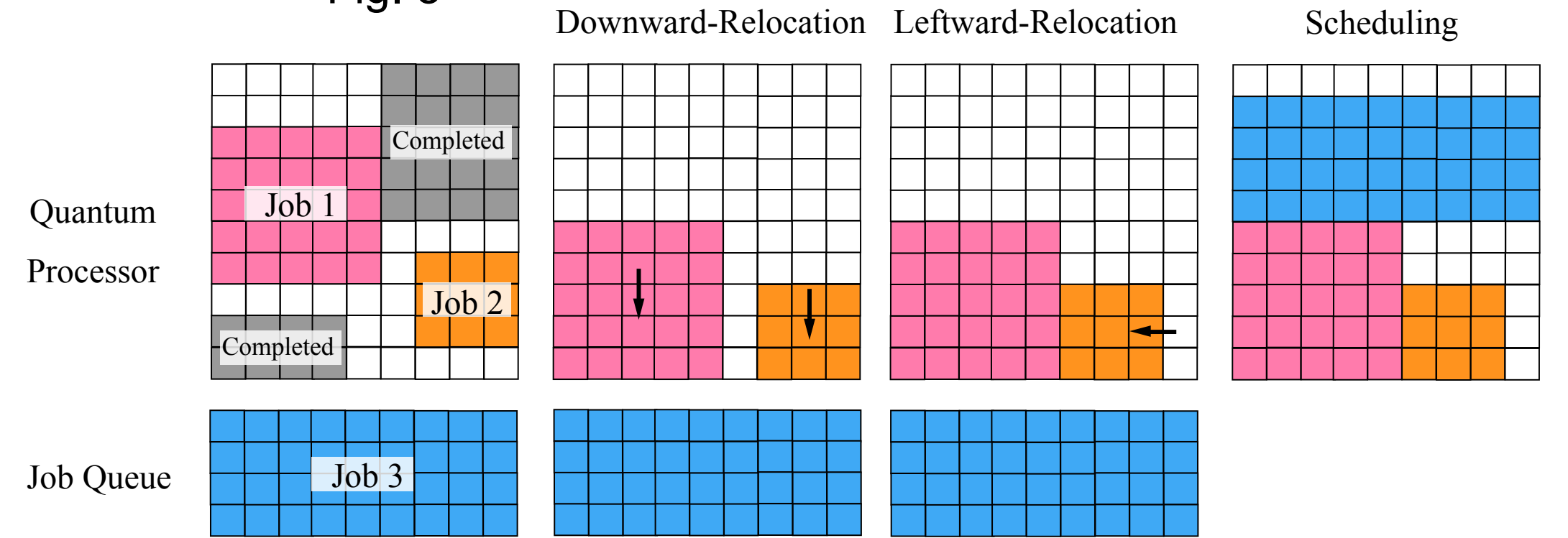


Fig. 10 Defragmentation

## Evaluation

| Parameter              | Value                            |
|------------------------|----------------------------------|
| Code cycle             | 1 $\mu$ s                        |
| Code distance $d$      | 31                               |
| Chip size $W \times H$ | 20 $\times$ 20                   |
| Batch size $B$         | 5                                |
| Defrag interval $I$    | $2 \times 10^4 \times d$ $\mu$ s |

Experimental settings

| Type | $w, h$   | $l$                              | Class | #Request | Ratio of each data type                     |
|------|----------|----------------------------------|-------|----------|---|
| I    | [5, 10]  | $[1 \times 10^4, 2 \times 10^4]$ | A     | 300      | I: 50%, others: 10% each                    |
| II   | [5, 10]  | $[4 \times 10^4, 6 \times 10^4]$ | B     |          | II: 50%, others: 10% each                   |
| III  | [5, 10]  | $[8 \times 10^4, 1 \times 10^5]$ | C     |          | III: 50%, others: 10% each                  |
| IV   | [10, 20] | $[1 \times 10^4, 2 \times 10^4]$ | D     |          | IV: 50%, others: 10% each                   |
| V    | [10, 20] | $[4 \times 10^4, 6 \times 10^4]$ | E     |          | V: 50%, others: 10% each                    |
| VI   | [10, 20] | $[8 \times 10^4, 1 \times 10^5]$ | F     |          | VI: 50%, others: 10% each                   |
|      |          |                                  | G     |          | All: 16.6% each (uniform ratio)             |
|      |          |                                  | H     |          | I, II, III: 3.33% each, IV, V, VI: 30% each |
|      |          |                                  | I     |          |   |

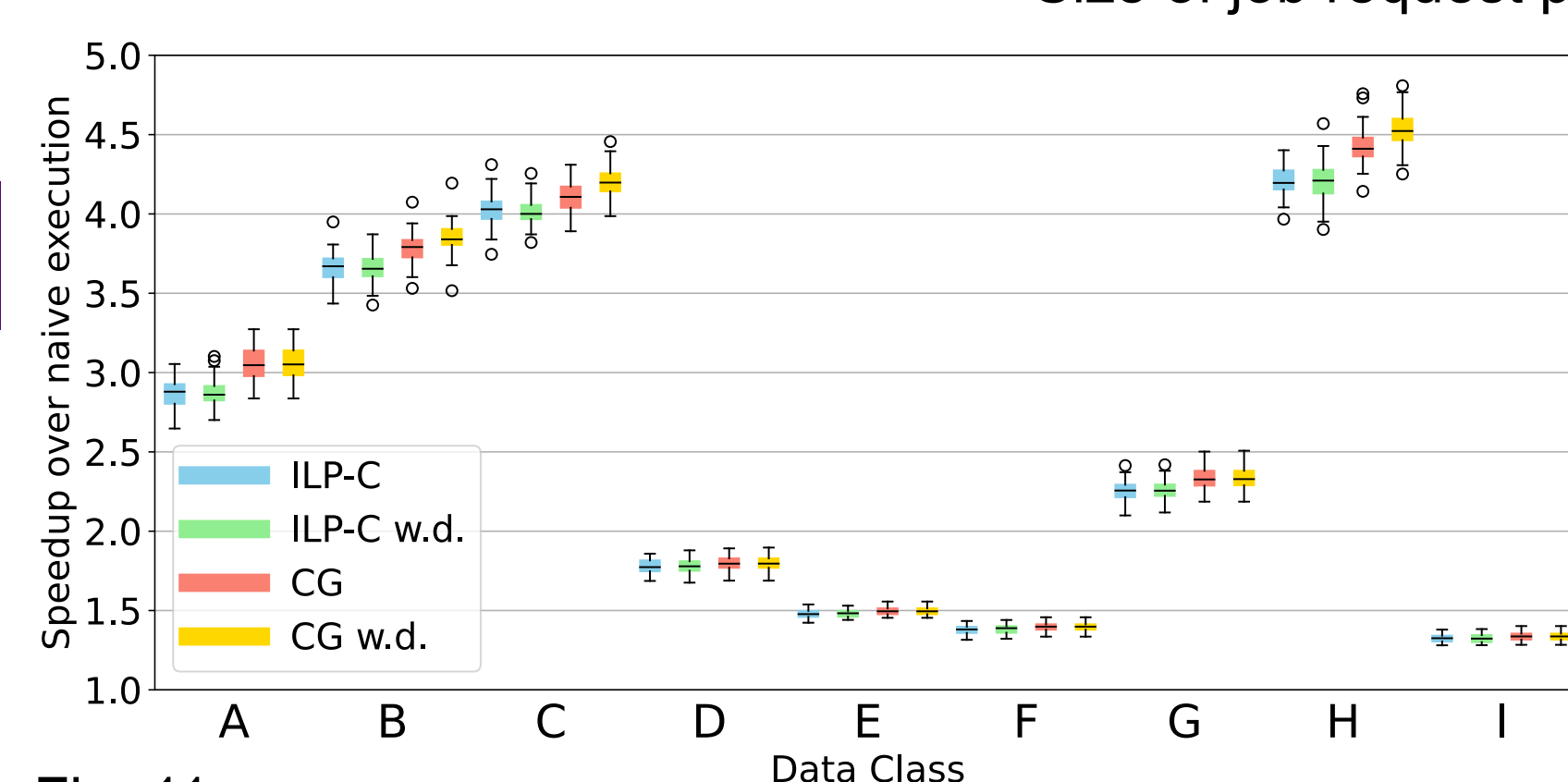


Fig. 11

Speedup relative to naive total job execution time ( $= \sum_i l_i$ )

| Type  | $B$ | Mean     | Min     | Max      | StdDev  |
|-------|-----|----------|---------|----------|---------|
| ILP-C | 5   | 244918   | 4962    | 14000017 | 831126  |
| ILP-C | 10  | 1646150  | 70221   | 21186320 | 1823412 |
| ILP-C | 15  | 5778808  | 553281  | 30102046 | 6098884 |
| ILP-C | 20  | 24384941 | 3230509 | 30086498 | 9427292 |
| CG    | 5   | 2652     | 13      | 15989    | 2652    |
| CG    | 10  | 5529     | 24      | 17444    | 4990    |
| CG    | 15  | 7948     | 50      | 25567    | 7190    |
| CG    | 20  | 10900    | 88      | 31623    | 9789    |

Fig. 13 Responsiveness

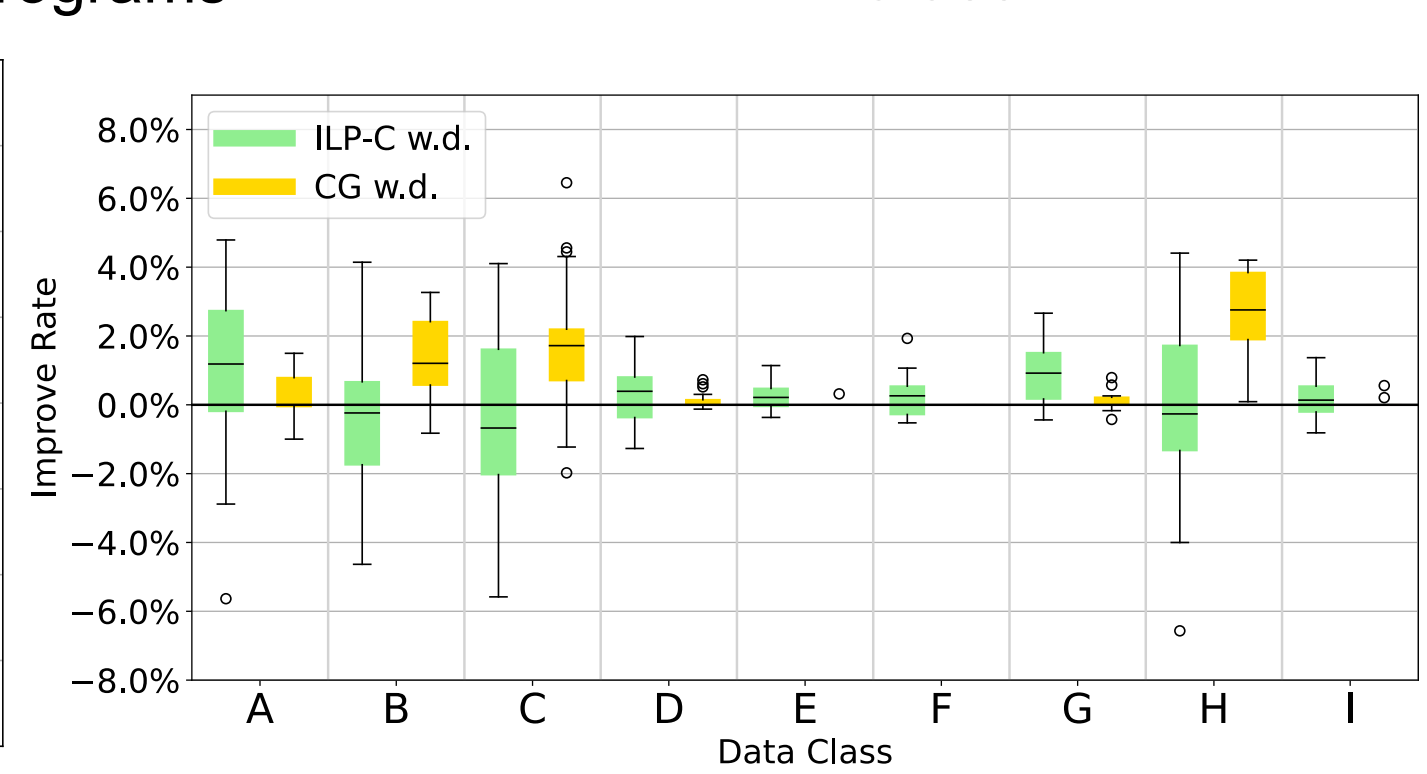


Fig. 12 Improvement rate (%) through defragmentation against the scheduler without defragmentation

Our evaluation demonstrates that the proposed scheduler, together with the defragmentation technique, improves the throughput by 2.3 to 2.4 times on average, with a maximum improvement of 4.53 times.

## Conclusion

In this work, we formulated the online scheduling problem for fault-tolerant quantum multiprogramming. Although we assumed that the quantum processor employs lattice surgery on surface codes, the scheduling is based on 3D space-time job requests and can therefore be straightforwardly generalized to any quantum programs on 2D quantum processors using different QECCs or logical gate implementations. We proposed two schedulers: an ILP-based scheduler and a scheduler referred to as corner greedy. Both approaches rely on a preprocessing step that transforms polycube representations into cuboids, enabling significantly higher responsiveness compared to schedulers based directly on polycube formulations. Furthermore, by introducing the concept of defragmentation, we experimentally demonstrated that average throughput can be improved in some datasets.

[1] Y. Tomita and K. M. Svore, "Low-distance surface codes under realistic quantum noise", Physical Review A 90, (2014) arXiv:1404.3747  
 [2] Akimoto Nakayama, Yasunari Suzuki, and Yuuki Tokunaga. マルチプロ グラミングによる FTQC のスループット向上 (in Japanese). Technical Reports for SIG on Quantum Software, Information Processing Society of Japan, 18(2023-QS-9):1-7, June 2023.  
 [3] Dror G Feitelson. Job scheduling in multiprogrammed parallel systems. 1997.

\* s.nishio@ucl.ac.uk

This work is supported by New Energy and Industrial Technology Development Organization (NEDO).

