Team Number:	apmcmlz2501604
Problem Chosen:	Quantum Optimization

2026 APMCM summary sheet

Hybrid Classical-Quantum Logic-Based Benders Decomposition Algorithm for Large-Scale Integrated Seru Scheduling

Abstract: Motivated by the conversion of large-scale assembly lines into order-driven serus at Lenovo's laptop manufacturing factory, LCFC (Hefei, China), this paper investigates the integrated "Order-Device-Worker" scheduling in seru production systems. This problem constitutes a complex combinatorial optimization challenge within the context of large-scale discrete manufacturing. The challenges stem from nonlinear worker processing times and sequence-dependent setup times, which collectively contribute to an NP-hard structure that is intractable using conventional exact solution methods.

To tackle these computational challenges, this paper proposes a tractable **mixed-integer second-order cone programming (MISOCP) model** and an effective **hybrid quantum logic-based Benders decomposition (Q-LBBD)** algorithm. *First*, by exploiting structural properties and leveraging second-order cone (SOC) duality theory, we establish an exact MISOCP reformulation of the original mixed-integer nonlinear programming model, enabling exact solution via existing branch-and-cut algorithms. *Second*, we propose an efficient hybrid classical-quantum Benders decomposition algorithm. This approach decomposes the MISOCP model into a master assignment problem and a set of NP-hard sequencing subproblems, which are reformulated into **quantum-solvable QUBO** forms and solved by the **Coherent Ising Machine**. Our algorithm introduces two key innovations: (i) to accelerate convergence, we develop a **sequence-independent setup time decomposition method** based on linear programming, which yields strong **combinatorial and analytical Benders cuts** that are proven to dominate traditional no-good cuts; (ii) to alleviate the computational burden on the quantum processing unit, we design an **alternating cut strategy** that prunes infeasible solutions before quantum execution, and a **variable-reduction strategy** integrating reduced cost fixing and dual picking to decrease the number of 0-1 decision variables.

Numerical studies demonstrate that the proposed Q-LBBD with **CIM guarantees optimality** while achieving a **98%–99% runtime reduction** compared with Gurobi. Furthermore, the proposed **variable reduction technique** facilitates deployment by achieving average fixation rates of **14.95%** for raw Ising variables and **23.27%** after precision adaptation. Overall, these results verify the superiority of the CIM-based hybrid quantum computing paradigm for industrial scheduling.

Keywords: Seru Production System Logic-Based Benders Decomposition Coherent Ising Machine Hybrid Classical-Quantum Computing

Contents

1 Introduction

In the Industry 4.0 era, with the increasingly drastic fluctuations in production demand and the continuous shortening of product lifecycles, rapid response capability has become as crucial as production volume and variety diversity [13]. However, traditional assembly lines, built upon large-scale production and task specialization, struggle to meet the demands of a rapidly evolving market landscape [7]. Inspired by Japanese electronics manufacturers, a novel production paradigm known as *Seru* production has emerged. *Seru* production takes place within at least one *Seru* production system, which comprises one or more *Seru* units [8]. Each *Seru* is an assembly cell containing simple equipment and one or more multi-skilled workers [5]. By strategically coordinating personnel, equipment, and products, *Seru* production achieves an organic unity of efficiency, flexibility, and responsiveness. *Seru* production involves two core processes: *Seru* formation and *Seru* scheduling. *Seru* formation determines the number of parallel *Seru*s to be constructed and the worker composition within each unit; whereas *Seru* scheduling determines which *Seru* processes each product batch. In practice, these two processes are often interrelated; thus, considering their synergistic optimization is of significant practical importance.

The superior reconfigurability and rapid response capability of *Seru* production make it a widely adopted method in modern manufacturing systems. Laptop manufacturing factories employ *Seru* production lines to tackle the challenges of high-demand-fluctuation products while ensuring high quality and short delivery times. These lines specialize in producing high-urgency, customized business laptops (as shown in Figure 1).

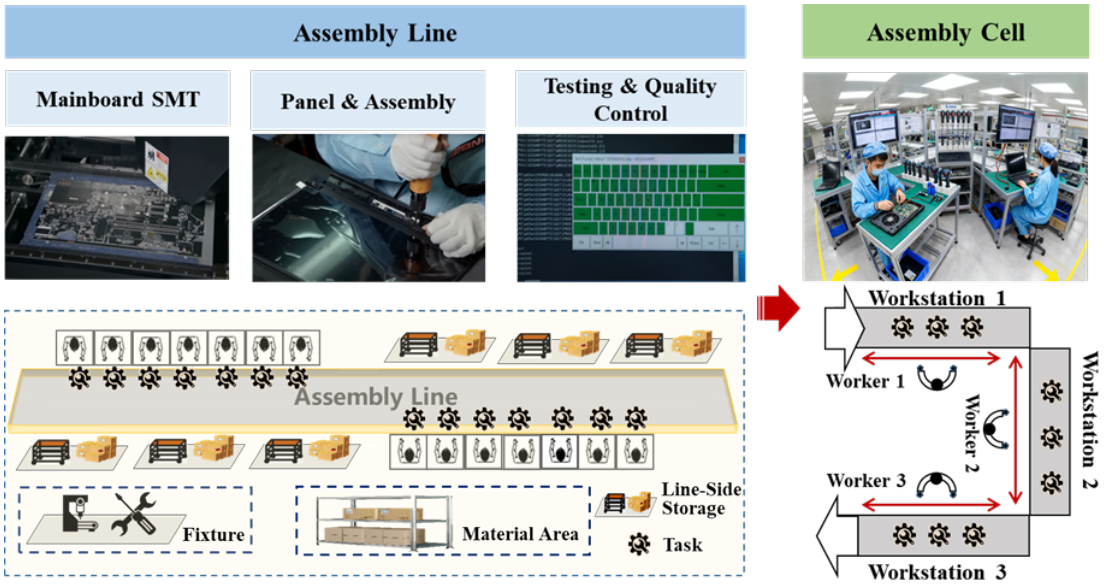


Figure 1 The integrated "Order-Device-Worker" scheduling at Lenovo's laptop manufacturing factory, LCFC (Hefei, China)

The *Seru* production line operates through four main stages: pre-assembly, assembly, testing, and packaging (as shown in Figure 2). In the *seru* production line in Lenovo's laptop manufacturing factory, upon receiving customer orders, the pre-assembly stage is responsible for the material preparation and processing of semi-finished products. These semi-finished products are integrated

and assembled into complete laptops using specialised equipment and tools in the assembly stage. The testing stage rigorously evaluates the hardware functionality, ensuring the laptops meet quality standards. Finally, the packaging stage completes the production process by securely packaging the laptops. The assembly stage is the core of the *Seru* production line, utilizing a Rotating *Seru* system. A Rotating *Seru* is a U-shaped line consisting of two or more cross-trained workers. These workers are responsible for assembling the product from start to finish, moving with the product sequentially from one workstation to the next to perform value-added operations. Upon completing tasks at the last workstation, they return to the first workstation to begin assembling the next product.

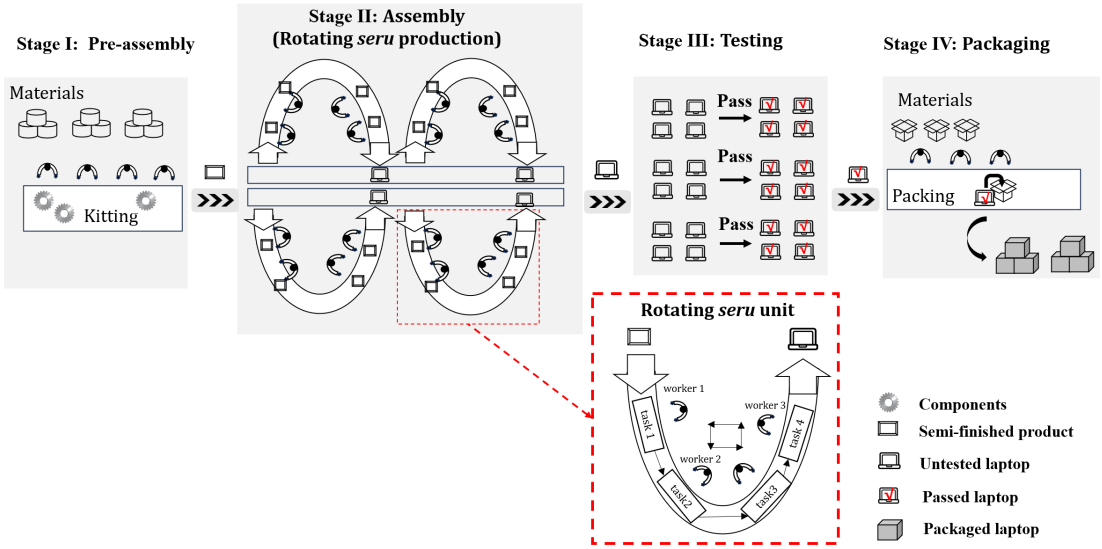


Figure 2 Schematic diagram of a Seru production line in a large laptop manufacturing factory.

A primary advantage of the Rotating *Seru* system lies in its flexibility to handle fluctuating production demands, particularly when processing customized laptop orders with short lead times. Compared to two other *Seru* forms—Yatai (booth) and Divisional—Rotating *Seru* stands out for its superior efficiency and flexibility. The Yatai system relies on a single person to complete unit tasks, whereas Rotating *Seru* involves multiple workers, enabling higher productivity to meet daily rush orders. Divisional *Seru* typically consists of workers assigned to specific workstations on the line; since each worker is responsible only for tasks within a specific area, flexibility is limited. In contrast, Rotating *Seru* allows cross-trained workers to complete all tasks without interruption. This shift has led to a drastic reduction in assembly flow time from 32 minutes to 15 minutes and reduced high work-in-process inventory from 70 units to 8 units, which is crucial for the rapid production cycles required for high-end laptop manufacturing. Additionally, the physical layout of Rotating *Seru* imposes constraints, such as prohibiting workers from overtaking one another.

Despite significant progress in the field of *Seru* production operations, existing research has not fully addressed the computational challenges in large-scale collaborative scheduling, primarily reflected in the following three aspects:

First, a lack of exact algorithms for large-scale complex Seru systems. Due to the heterogeneity of workers in Seru systems leading to nonlinear processing times (synergistic effects) and involving complex sequence-dependent setup times, existing studies mostly rely on meta-heuristic algorithms or simple enumeration. When facing industrial-scale problems, the computation time of

these methods increases exponentially with the number of workers, making it difficult to guarantee solution optimality.

Second, the NP-hard nature of subproblems constitutes a computational bottleneck. In Seru scheduling, determining the job processing sequence within each unit is essentially a Traveling Salesman Problem (TSP). As the number of batches increases, existing decomposition algorithms are inefficient in repeatedly solving these NP-hard sequencing subproblems, severely hindering algorithm scalability.

Third, emerging quantum computing technologies have not been fully utilized to solve such complex industrial scheduling problems. Although Coherent Ising Machines (CIM) show great potential in solving combinatorial optimization problems, there is currently no research applying them within the Benders decomposition framework for Seru production scheduling, and how to handle qubit scale limitations remains an unresolved issue.

To fill the aforementioned research gaps, this paper proposes a Hybrid Classical-Quantum Logic-Based Benders Decomposition (Q-LBBD) algorithm. The main contributions of this paper are summarized as follows:

1. **Proposed the first MISOCP reformulation for SPPs.** We provide equivalently tractable MISOCP reformulation for SPPs, which can be exactly solved by existing branch-and-cut methods. Existing exact algorithms require enumerating all possible *seru* formations due to the nonlinear characteristic of SPTs, resulting in an exponential computational complexity. To address this difficulty, we analyse the structural properties of robust SPTs and derive computationally tractable reformulations. To the best of our knowledge, it is the first equivalently tractable MISOCP reformulation for SPPs. Moreover, a set of symmetry-breaking constraints is introduced to speed up the search process.
2. **Proposed the first Hybrid Classical-Quantum LBBD algorithm for the Seru integrated scheduling problem.** We decompose the problem into a Master Problem responsible for worker and job assignment, and Subproblems responsible for job sequencing. Addressing the issue of weak lower bounds in existing decomposition methods, we propose a Sequence-Independent Setup Time Decomposition (SISTD) strategy. Unlike existing heuristic estimates [9], we extract the maximum possible sequence-independent components through a linear programming model, significantly tightening the subproblem lower bound. Based on this, we derive problem-specific Combinatorial Benders cuts and Analytical Benders cuts, theoretically guaranteeing the effectiveness of the cuts and their dominance over traditional no-good cuts.
3. **Constructed a subproblem solver and variable fixing technique based on the Coherent Ising Machine.** We reformulate the sequencing subproblem into a Quadratic Unconstrained Binary Optimization (QUBO) model suitable for quantum computing. To overcome quantum hardware scale limitations, we design a variable reduction method integrating Reduced Cost Fixing and Dual Picking. This method effectively reduces problem size, making it possible to solve large-scale industrial instances using limited qubits.
4. **Extensive experimental validation based on industrial data.** Numerical experiments show that the Q-LBBD algorithm reduces computation time by over 97% compared to Gurobi on small-scale instances; on larger-scale instances, it also achieves efficient solutions within a short time, validating the potential of quantum computing in smart manufacturing scheduling.

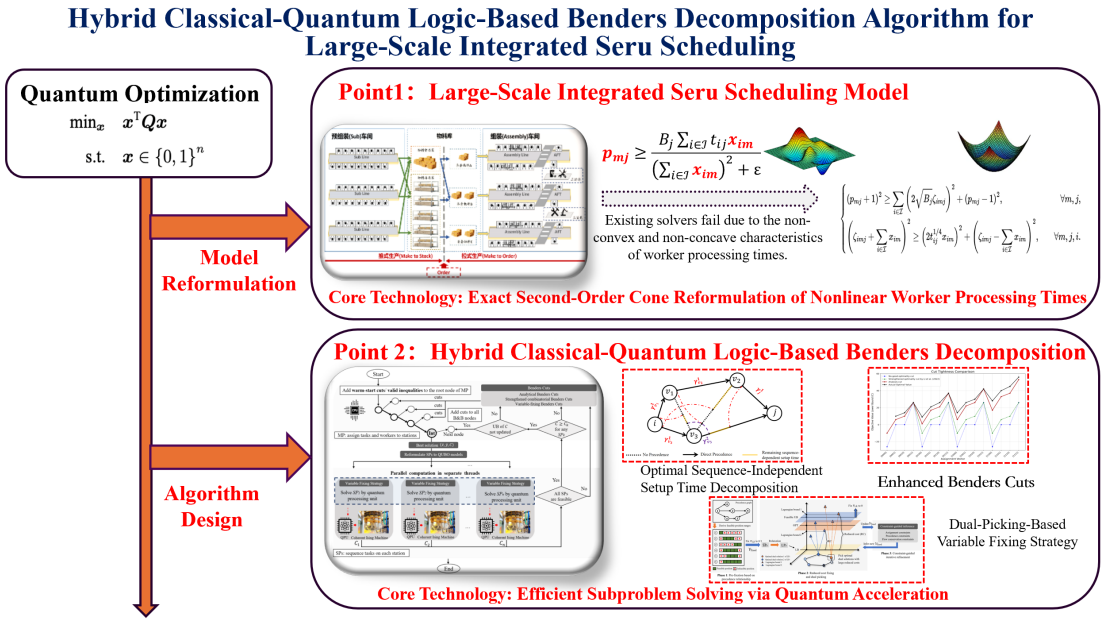


Figure 3 Framework of this paper

The remainder of this paper is organized as follows: Chapter 2 constructs the mathematical model for the Seru production problem and performs the Second-Order Cone Programming reformulation. Chapter 3 details the design of the hybrid quantum LBBD algorithm, including the SISTD strategy, cut derivation, and quantum solution scheme. Chapter 4 reports numerical experimental results and algorithm performance analysis. Finally, Chapter 5 concludes the paper.

2 Seru Production Model

This section first formally outlines the problem description of the seru production problem (SPP), defining the sets, decision variables, and specifically characterizing the nonlinear synergistic effect among multi-skilled workers. Subsequently, a mixed-integer nonlinear programming (MINLP) model is constructed to minimize the makespan. To address the computational intractability arising from the non-convex constraints and bilinear terms within the MINLP, we then derive an exact mixed-integer second-order cone programming (MISOCP) reformulation based on convex optimization theory. Finally, the complete tractable MISOCP model, augmented with linearization techniques and symmetry-breaking constraints, is presented.

2.1 Problem description

The Seru Production Problem (SPP) mainly consists of two core subproblems: Seru Formation and Seru Scheduling. The Seru Formation subproblem aims to determine the number of parallel Serus to be built and the worker composition within each Seru; the Seru Scheduling subproblem aims to determine which Seru each product batch is assigned to for processing.

Let $\mathcal{I} = \{1, \ldots, I\}$ denote the set of available workers, and $\mathcal{M} = \{1, \ldots, M\}$ denote the set of potential Serus (satisfying $M \leq I$). In the Seru formation phase, workers need to be assigned to potential Serus. Define binary decision variable x_{im}, where $x_{im} = 1$ if worker $i \in \mathcal{I}$ is assigned to Seru $m \in \mathcal{M}$, and $x_{im} = 0$ otherwise. When $\sum_{i\in\mathcal{I}} x_{im} \geq 1$, it indicates that Seru m is formally constructed. The set of feasible Seru formation plans X is defined as follows:

$$X = \left\{ x \;\middle|\; \sum_{m\in\mathcal{M}} x_{im} = 1,\ x_{im} \in \{0,1\},\ \forall i \in \mathcal{I},\ m \in \mathcal{M} \right\}. \tag{1}$$

This constraint ensures that each worker must be assigned to exactly one Seru.

Let $\mathcal{J} = \{1, \ldots, J\}$ denote the set of product batches, with each batch j containing B_j products of the same type. In the Seru scheduling phase, batches need to be assigned to the constructed Serus. Define binary decision variable z_{mj}, where $z_{mj} = 1$ if batch $j \in \mathcal{J}$ is assigned to Seru $m \in \mathcal{M}$, and $z_{mj} = 0$ otherwise. The set of feasible Seru scheduling plans Z is defined as follows:

$$Z = \left\{ z \;\middle|\; \sum_{m\in\mathcal{M}} z_{mj} = 1,\ z_{mj} \in \{0,1\},\ \forall m \in \mathcal{M},\ j \in \mathcal{J} \right\}. \tag{2}$$

In the Seru system, worker heterogeneity results in varied processing efficiencies for different products. Let t_{ij} denote the Worker Processing Time (WPT) for worker i to process a single product in batch j. Due to differing skill levels, WPT can be expressed as the product of the standard processing time s_j and the worker's skill level f_{ij}, i.e., $t_{ij} = f_{ij}s_j$. According to [11] on Rotating Seru, a single worker possesses the capability to complete the entire product assembly. Given a Seru formation plan x, the average processing time e_{mj} for batch j on Seru m can be expressed as:

$$e_{mj} = \frac{B_j \sum_{i\in\mathcal{I}} t_{ij} x_{im}}{\sum_{i\in\mathcal{I}} x_{im}}. \tag{3}$$

A distinguishing feature of the Seru system is the synergistic effect among multi-skilled workers. Collaboration often yields efficiency gains beyond simple linear addition [10]. Considering this

synergistic effect, the actual processing time (Seru Processing Time, SPT) p_{mj} for batch j on Seru m is defined as [4]:

$$p_{mj} = \frac{e_{mj}}{\sum_{i\in\mathcal{I}} x_{im}} = \frac{B_j \sum_{i\in\mathcal{I}} t_{ij} x_{im}}{\left(\sum_{i\in\mathcal{I}} x_{im}\right)^2}. \tag{4}$$

The squared denominator $\left(\sum_{i\in\mathcal{I}} x_{im}\right)^2$ in Equation (4) clearly characterizes this nonlinear synergistic property, which is the core reason making this problem difficult to solve using traditional linear programming.

The sets, parameters, decision variables, and auxiliary variables involved in this paper are detailed in Table 1.

Table 1 Parameters and Variables Notation

Sets	
$\mathcal{I}$	Set of workers, indexed by $i \in \{1, \ldots, \lvert\mathcal{I}\rvert\}$
$\mathcal{M}$	Set of potential Serus, indexed by $m \in \{1, \ldots, \lvert\mathcal{M}\rvert\}$
$\mathcal{J}$	Set of product batches, indexed by $j \in \{1, \ldots, \lvert\mathcal{J}\rvert\}$
$\mathcal{J}_0$	Extended set of batches (including dummy node 0), $j \in \{0, 1, \ldots, \lvert\mathcal{J}\rvert\}$
Parameters	
B_j	Number of products in batch j
t_{ij}	Unit time for worker i to process a single product in batch j
s_{jk}	Setup time required to switch from batch j to batch k ($j, k \in \mathcal{J}_0, j \neq k$)
ε	A very small positive number to prevent division by zero
$\widehat{p}_{mj}$	Big-M parameter, defined as $B_j \max_{i\in\mathcal{I}} t_{ij}$
Decision Variables	
x_{im}	Binary variable: 1 if worker i is assigned to Seru m
z_{mj}	Binary variable: 1 if batch j is assigned to Seru m
u_m	Binary variable: 1 if Seru m is used (non-empty)
y_{jkm}	Binary variable: 1 if batch k is processed immediately after batch j in Seru m
p_{mj}	Continuous variable: Total processing time of batch j on Seru m
l_{mj}	Continuous variable: Linearized actual processing time (0 if $z_{mj} = 0$)
ω_{mj}	Continuous variable: Auxiliary variable for subtour elimination (MTZ)
$C_{\max}$	Continuous variable: System makespan

2.2 Mixed-Integer Nonlinear Programming Model (MINLP)

Based on the above definitions, the goal of the deterministic Seru Production Problem (SPP) is to find the optimal production plan (x, z) to minimize the makespan. This problem can be modeled as the following MINLP:

$$
\begin{aligned}
(\textbf{SPP}) \quad \min \quad & C_{\max} && && (5)\\
\text{s.t.} \quad & \sum_{m\in\mathcal{M}} x_{im} = 1, && \forall i \in \mathcal{I}, && (6)\\
& \sum_{m\in\mathcal{M}} z_{mj} = 1, && \forall j \in \mathcal{J}, && (7)\\
& \sum_{j\in\mathcal{J}} z_{mj} \le |\mathcal{J}| \sum_{i\in\mathcal{I}} x_{im}, && \forall m \in \mathcal{M}, && (8)\\
& p_{mj} \ge \frac{B_j \sum_{i\in\mathcal{I}} t_{ij} x_{im}}{\left(\sum_{i\in\mathcal{I}} x_{im}\right)^2 + \varepsilon}, && \forall m \in \mathcal{M}, j \in \mathcal{J}, && (9)\\
& C_{\max} \ge \sum_{j\in\mathcal{J}} z_{mj} p_{mj}, && \forall m \in \mathcal{M}, && (10)\\
& x_{im}, z_{mj} \in \{0,1\}, && \forall i, m, j. && (11)
\end{aligned}
$$

2.3 MISOCP-Based Reformulation

The aforementioned MINLP model contains non-convex constraints (9) and bilinear terms $z_{mj}p_{mj}$ (10), which cannot be directly solved by existing branch-and-bound algorithms. Therefore, we reformulate it into a Mixed-Integer Second-Order Cone Programming (MISOCP) model.

2.3.1 *SOC Representation of Nonlinear Constraints*

Lemma 1 *By introducing auxiliary variables ζ_{imj}, the nonlinear constraint (9) is equivalent to the following set of SOC constraints:*

$$
\begin{cases}
(p_{mj}+1)^2 \ge \sum_{i\in\mathcal{I}} \left(2\sqrt{B_j}\zeta_{imj}\right)^2 + (p_{mj}-1)^2, & \forall m, j,\\
\left(\zeta_{imj} + \sum_{i\in\mathcal{I}} x_{im}\right)^2 \ge \left(2t_{ij}^{1/4} x_{im}\right)^2 + \left(\zeta_{imj} - \sum_{i\in\mathcal{I}} x_{im}\right)^2, & \forall m, j, i.
\end{cases}
\tag{12}
$$

Proof 1 *Let $\zeta_{imj} = \sqrt{t_{ij}x_{im}}/\sum_{i\in\mathcal{I}} x_{im}$, then the original constraint is equivalent to $p_{mj} \ge B_j \sum \zeta_{imj}^2$. Using the algebraic identity $4xy = (x+y)^2 - (x-y)^2$, the above inequality can be transformed into a standard rotated second-order cone form, as stated in the lemma.*

2.3.2 *Linearization of Bilinear Terms*

For the bilinear term $z_{mj}p_{mj}$ in constraint (10), introduce variable $l_{mj} = z_{mj}p_{mj}$ and use the Big-M method (Glover's linearization) for linearization:

$$\begin{aligned} & l_{mj} \le \widehat{p}_{mj} z_{mj}, \\ & l_{mj} \le p_{mj}, \\ & l_{mj} \ge p_{mj} - \widehat{p}_{mj}(1 - z_{mj}), \\ & l_{mj} \ge 0. \end{aligned} \tag{13}$$

Where $\widehat{p}_{mj} = B_j \max_i t_{ij}$ is a sufficiently large upper bound parameter.

2.4 Equivalent MISOCP Model

Based on Lemma 1 and Glover's linearization method, we reformulate the deterministic SPP into the following Mixed-Integer Second-Order Cone Programming (MISOCP) model. This model eliminates the fractional non-convex constraints and bilinear terms from the original MINLP and can be directly solved exactly by modern commercial solvers such as Gurobi or CPLEX.

Proposition 1 *The deterministic Seru Production Problem is equivalently reformulated as the following MISOCP model:*

$$\begin{aligned}
(\textbf{\textit{MISOCP}}) \quad \min \quad & C_{\max} && && (14)\\
\textit{s.t.} \quad & (12)-(13), && && (15)\\
& \sum_{m\in\mathcal{M}} x_{im} = 1, && \forall i \in \mathcal{I}, && (16)\\
& \sum_{m\in\mathcal{M}} z_{mj} = 1, && \forall j \in \mathcal{J}, && (17)\\
& \sum_{j\in\mathcal{J}} z_{mj} \le |\mathcal{J}| \sum_{i\in\mathcal{I}} x_{im}, && \forall m \in \mathcal{M}, && (18)\\
& C_{\max} \ge \sum_{j\in\mathcal{J}} l_{mj}, && \forall m \in \mathcal{M}, && (19)\\
& \sum_{i\in\mathcal{I}} x_{im} \ge u_m, && \forall m \in \mathcal{M}, && (20)\\
& \sum_{j\in\mathcal{J}} z_{mj} \ge u_m, && \forall m \in \mathcal{M}, && (21)\\
& \sum_{i\in\mathcal{I}} x_{im} \le |\mathcal{I}| u_m, && \forall m \in \mathcal{M}, && (22)\\
& u_m \ge u_{m+1}, && \forall m < |\mathcal{M}|, && (23)\\
& \sum_{k=1}^{i-1} x_{k,m} \ge x_{i,m+1}, && \forall m < |\mathcal{M}|, i > 1, && (24)\\
& x_{im}, z_{mj}, u_m \in \{0,1\}, && \forall i, m, j, && (25)\\
& p_{mj}, l_{mj}, \zeta_{imj}, C_{\max} \ge 0. && && (26)
\end{aligned}$$

Constraints (16)-(18) form the basic resource assignment logic, ensuring unique assignment of workers and batches, and preventing batches from being assigned to idle Serus. Constraints (19) handle the bilinear term $z_{mj}p_{mj}$ in the objective function. By introducing the auxiliary variable l_{mj}, we convert the logical relationship if $z_{mj} = 0$, the processing time does not count towards the total makespan" into linear constraints.

3 Hybrid Classical-Quantum Logic-Based Benders Decomposition Algorithm

Addressing the challenges of tight variable coupling, significant nonlinear effects, and the NP-hard nature of subproblems (TSP) in Seru production scheduling, this chapter proposes a Hybrid Classical-Quantum Logic-Based Benders Decomposition (Q-LBBD) algorithm. Instead of stopping at the traditional Benders decomposition framework, this algorithm designs a two-level acceleration mechanism for large-scale collaborative scheduling problems: tightening lower bounds and reducing subproblem calls at the classical level via Sequence-Independent Setup Time Decomposition (SISTD) and alternating cut strategies; and breaking the computational bottleneck of sequencing subproblems at the quantum level utilizing a Coherent Ising Machine.

As shown in Figure 4, Q-LBBD decouples the original problem into a Master Problem (MP) responsible for worker and job assignments, and multiple Subproblems (SPs) responsible for job sequencing. The overall interaction flow and core improvement strategies of the algorithm are as follows:

1. **Decoupling of Master and Subproblems**: The MP determines the assignment plan for workers and jobs, passing the parameterized allocation scheme $\bar{x}$, $\bar{z}$ to the SPs; given the allocation, the SPs solve single-machine sequencing problems aiming to minimize the makespan.
2. **Classical Acceleration Layer based on SISTD**: To overcome the poor quality of relaxed lower bounds in traditional LBBD, we first introduce the SISTD strategy before solving the NP-hard job sequencing SPs. By constructing a linear programming model to extract the maximum possible sequence-independent time components, we derive enhanced Combinatorial and Analytical Benders cuts that dominate traditional no-good cuts. Meanwhile, adopting an Alternating Cuts mechanism, we use the computationally cheap SISTD lower bound as a ”filter” to directly intercept and prune obviously infeasible master problem solutions, thereby avoiding ineffective calls to NP-hard job sequencing SPs.
3. **Quantum Acceleration Layer based on Ising Machine**: For difficult SPs that cannot be filtered by SISTD, we reformulate them into Quadratic Unconstrained Binary Optimization (QUBO) models. To handle industrial-grade instances within limited qubit scales, we introduce variable reduction techniques integrating Reduced Cost Fixing and Dual Picking, finally efficiently searching for the optimal permutation of the subproblem using the Coherent Ising Machine and feeding the generated Benders cuts back to the MP.

The Master Problem and Subproblems are solved via Branch-and-Check until the gap between the upper and lower bounds of the problem closes. The specific implementation details are elaborated in the following sections.

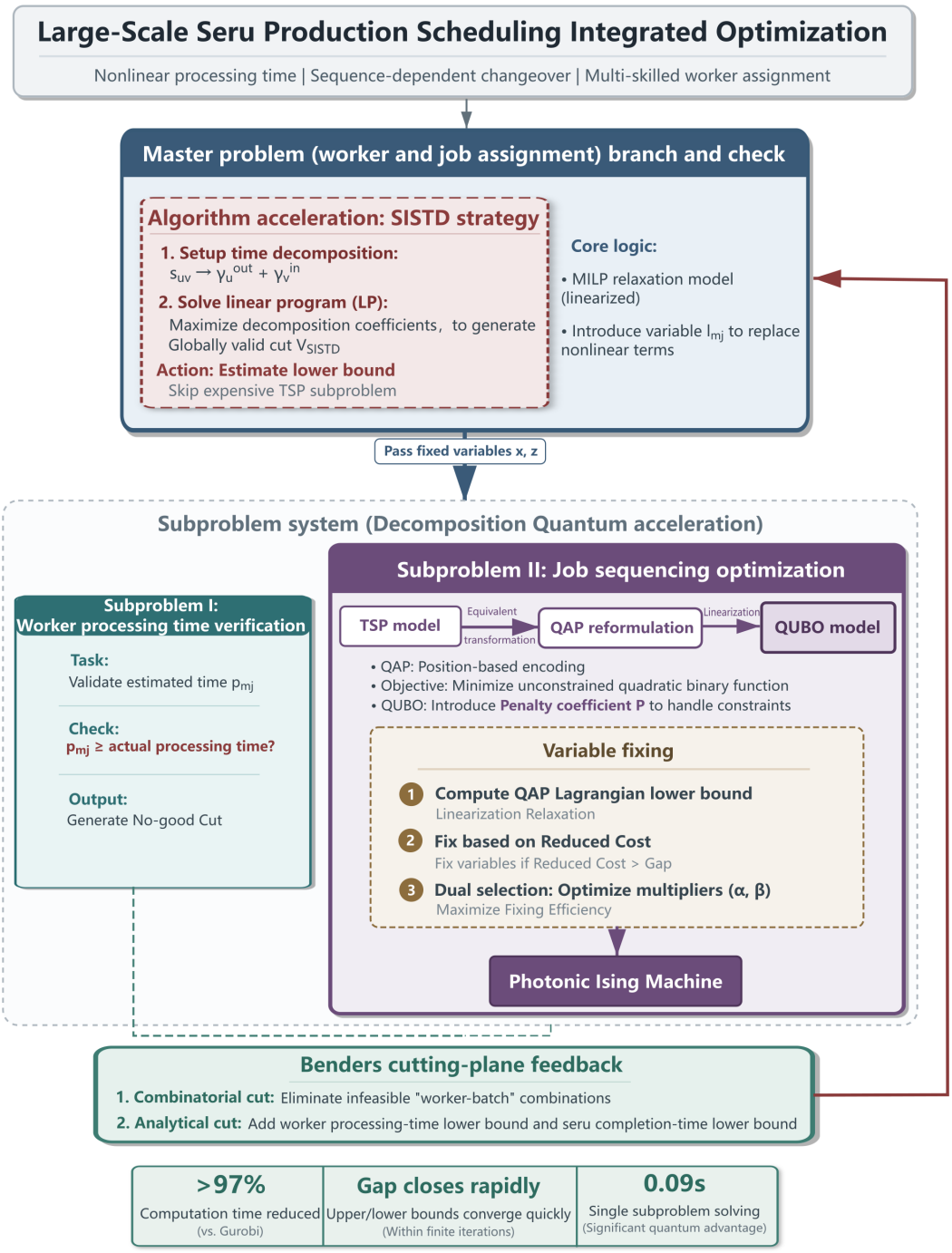


Figure 4 Overall framework schematic of the Hybrid Classical-Quantum LBBD algorithm.

3.1 Master Problem Model (MP)

The Master Problem is responsible for determining worker assignments x_{im}, batch allocations z_{mj}, and the relaxed values of processing time variables p_{mj}. To handle logical relationships, we introduce auxiliary variables l_{mj} for linearization. Let $\mathcal{R}_p(m,j)$ be the set of cuts for processing time p_{mj}, and $\mathcal{R}_s$ be the set of cuts for setup times.

$$(\textbf{MP}) \quad \min \quad C_{\max} \tag{27}$$
$$\text{s.t.} \quad \text{Constraints } (16) - (19),$$

$$l_{mj} \le \widehat{p}_{mj} z_{mj}, \qquad \forall m \in \mathcal{M}, j \in \mathcal{J}, \tag{28}$$

$$l_{mj} \le p_{mj}, \qquad \forall m \in \mathcal{M}, j \in \mathcal{J}, \tag{29}$$

$$l_{mj} \ge p_{mj} - \widehat{p}_{mj}\left(1 - z_{mj}\right), \qquad \forall m \in \mathcal{M}, j \in \mathcal{J}, \tag{30}$$

$$p_{mj} \ge \Gamma^r_{mj}(x), \qquad \forall m \in \mathcal{M}, j \in \mathcal{J}, \forall r \in \mathcal{R}_p(m,j), \tag{31}$$

$$C_{\max} \ge \sum_{j \in \mathcal{J}} l_{mj} + \Psi^k_m(x,z), \qquad \forall m \in \mathcal{M}, \forall k \in \mathcal{R}_s, \tag{32}$$

$$x_{im}, z_{mj}, u_m \in \{0,1\}, \quad p_{mj}, l_{mj} \ge 0. \tag{33}$$

Where $\Gamma^r_{mj}(x)$ is the function generated by Subproblem I for the r-th correction of the processing time lower bound (see Section 3.2). $\Psi^k_m(x,z)$ is the function for the k-th Benders cut generated by Subproblem II to constrain setup times (see Section 3.3).

3.2 Subproblem I: Nonlinear Processing Time Verification

3.2.1 *Subproblem I Mathematical Model*

Subproblem I aims to verify whether the processing time $\bar{p}_{mj}$ estimated by the Master Problem is correct. For a given MP solution $(\bar{x}, \bar{z})$, we check if there exists $\bar{p}_{mj} < p_{mj}$. Although this is essentially a computational verification process, its implied optimization model is as follows:

$$(\text{SP}_{\text{mj}}) \quad \min_{p_{mj}} \quad p_{mj} \tag{34}$$

$$\text{s.t.} \quad p_{mj} \ge \frac{B_j \sum_{i \in \mathcal{I}} t_{ij} \bar{x}_{im}}{\left(\sum_{i \in \mathcal{I}} \bar{x}_{im}\right)^2 + \varepsilon}, \tag{35}$$

$$p_{mj} \ge 0. \tag{36}$$

When the MP solution $\bar{p}_{mj}$ violates constraint (35), we add cuts. We provide two forms of cuts.

3.2.2 *No-good Benders Optimality Cut*

To ensure the nonlinear constraint (35) holds for any $\bar{x} \in X$, we introduce the following No-good cut:

$$p \ge \sum_{i \in \bar{S}} \frac{a_i}{m^2} \left(\sum_{i \in \bar{S}} x_i - m - \sum_{i \notin \bar{S}} x_i \right) + \sum_{i \in \bar{S}} \frac{a_i}{m^2}. \tag{37}$$

The following proposition proves the validity of the No-good cut, i.e., this No-good cut guarantees the exact value of batch processing time at point $\bar{x} \in X$, while providing a lower bound for processing time at other points.

Proposition 2 *For any $\bar{x} \in X$, let $\bar{S} = \{i : \bar{x}_i = 1\}$ and $m = |\bar{S}|$. Define the function*

$$\bar{l}(x) = \sum_{i \in \bar{S}} \frac{a_i}{m^2} \left(\sum_{i \in \bar{S}} x_i - m - \sum_{i \notin \bar{S}} x_i \right) + \sum_{i \in \bar{S}} \frac{a_i}{m^2}.$$

Then

$$\bar{l}(\bar{x}) = f(\bar{x}), \quad \textit{and} \quad \bar{l}(x) \le f(x), \quad \forall x \in X.$$

Proof 2 *For the current solution $\bar{x}$, it is evident that $\bar{l}(\bar{x}) = \sum_{i \in \bar{S}} \frac{a_i}{m^2} = f(\bar{x})$. For any other solution $x \in X$, i.e., for any $S \subseteq \mathcal{I}$ ($|S| \ge 1$), we have $\sum_{i \in S} x_i - m - \sum_{i \notin S} x_i \le 0$. Therefore, $\bar{l}(x) \le \sum_{i \in S} \frac{a_i}{m^2} = f(x)$.*

3.2.3 *Basic Analytical Benders Cut*

We derive a basic analytical cut that exploits the structure of the recourse function $f(x)$. For any incumbent $\bar{x} \in X$, we construct an affine underestimator $\bar{l}(x) = \alpha^\top x$ that is tight at $\bar{x}$ and valid for all $x \in X$. The following propositions establish its validity and show that it dominates the no-good cut.

Proposition 3 *For any $\bar{x} \in X$, let $\bar{S} = \{i : \bar{x}_i = 1\}$ and $m = |\bar{S}|$, and define $\bar{l}(x) = \alpha^\top x$. Set*

$$\alpha_i = \begin{cases} \dfrac{a_i}{m^2}, & i \in \bar{S}, \\ \dfrac{a_i}{|\mathcal{I}|^2} + \left(\dfrac{1}{(m+1)^2} - \dfrac{1}{m^2}\right) \displaystyle\sum_{j \in \bar{S}} a_j, & i \notin \bar{S}. \end{cases}$$

Then $\bar{l}(\bar{x}) = f(\bar{x})$ and $\bar{l}(x) \le f(x)$ for all $x \in X$.

Proof 3 *For detailed proof, please refer to* Appendix 1.1.

Proposition 4 *For any $\bar{x} \in X$, let $\bar{S} = \{i : \bar{x}_i = 1\}$ and $m = |\bar{S}|$. The analytical cut $\bar{l}(\bar{x})$ dominates the No-good cut $\hat{l}(x)$, i.e.,:*

$$\bar{l}(\bar{x}) \ge \hat{l}(x), \quad \forall x \in X.$$

Proof 4 *For detailed proof, please refer to* Appendix 1.2.

3.3 Subproblem Solving Based on Coherent Ising Machine

Coherent Ising Machines (CIMs) are a type of optical quantum computer designed to solve combinatorial optimization problems by finding the ground state of an Ising model. This architecture has demonstrated advantages in handling large-scale problems. In this paper, we employ a CIM to efficiently solve complex scheduling subproblems within our hybrid quantum-classical optimization framework.

3.3.1 *QUBO Model Reformulation of Batch Sequencing Subproblem*

To utilize quantum computing for acceleration, we reformulate the aforementioned job sequencing subproblem into a Quadratic Unconstrained Binary Optimization (QUBO) model. For Seru m and its corresponding batch set $\bar{\mathcal{J}}_m$ (let $N = |\bar{\mathcal{J}}_m|$), a position-based encoding is adopted. Given that the TSP model contains numerous inequality constraints, direct conversion to QUBO form usually requires introducing extra auxiliary variables for equality handling. To avoid this, we first equivalently transform the TSP into a Quadratic Assignment Problem (QAP), thereby enabling direct conversion to the QUBO model while maintaining model equivalence without introducing extra auxiliary variables.

$$(\text{QAP}_m) \quad \min \quad \sum_{k=1}^{N-1} \sum_{j \in \bar{\mathcal{J}}_m} \sum_{l \in \bar{\mathcal{J}}_m, l \ne j} s_{j,l} w_{j,k} w_{l,k+1} \tag{38}$$

$$\text{s.t.} \quad \sum_{j \in \bar{\mathcal{J}}_m} w_{j,k} = 1, \quad \forall k = 1, \cdots N, \tag{39}$$

$$\sum_{k=1}^{N} w_{j,k} = 1, \quad \forall j \in \bar{\mathcal{J}}_m, \tag{40}$$

$$w_{j,k} \in \{0,1\}, \quad \forall j \in \bar{\mathcal{J}}_m, k = 1, \cdots N, \tag{41}$$

where $w_{j,k}$ is a binary decision variable indicating whether batch j is placed at the k-th position. We relax the constraints in the QAP model into the objective function $H(w)$ via penalty coefficients P:

$$\begin{aligned} H(w) = & \sum_{k=1}^{N-1} \sum_{j \in \bar{\mathcal{J}}_m} \sum_{l \in \bar{\mathcal{J}}_m, l \neq j} s_{jl} w_{j,k} w_{l,k+1} \\ & + P_1 \sum_{k=1}^{N} \left(\sum_{j \in \bar{\mathcal{J}}_m} w_{j,k} - 1 \right)^2 + P_2 \sum_{j \in \bar{\mathcal{J}}_m} \left(\sum_{k=1}^{N} w_{j,k} - 1 \right)^2 \end{aligned} \tag{42}$$

The last two penalty terms ensure: exactly one batch per position, and each batch is processed exactly once. This model can be directly mapped to the Coherent Ising Machine for solution.

3.3.2 *Variable Reduction Technique: Reduced Cost Fixing and Dual Picking*

Penalty terms in the QUBO objective often densify the QUBO graph, weakening preprocessing such as Roof Duality [2, 6] (Figure 5). To better leverage the Coherent Ising Machine, we propose a variable reduction technique: build the linearized subproblem ($\overline{\text{QAP}}_m$), compute reduced costs from the dual of its LP relaxation with dual picking, and iteratively fix variables until no further fixings occur. The resulting reduced instance is then converted to QUBO and solved.

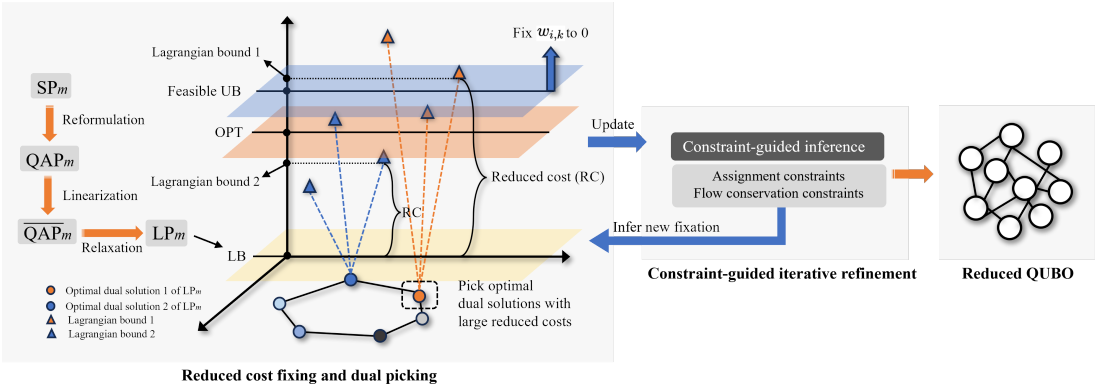


Figure 5 Variable Reduction Technique: Flowchart of Reduced Cost Fixing and Dual Picking

Reduced Cost Fixing (RCF) removes variables that would make the objective exceed a known upper bound. To obtain a tight lower bound, we introduce auxiliary binaries $v_{j,k,l}$, where $v_{j,k,l} = 1$ iff batch j is assigned to position k and batch l to position $k+1$. This yields the following equivalent linearized model:

$$(\overline{\text{QAP}}_m) \quad \min_{w,v} \sum_{k=1}^{N-1} \sum_{j \in \bar{\mathcal{J}}_m} \sum_{l \in \bar{\mathcal{J}}_m, l \neq j} s_{j,l}\, v_{j,k,l} \tag{43}$$

$$\text{s.t.} \sum_{l \in \bar{\mathcal{J}}_m, l \neq j} v_{j,k,l} = w_{j,k}, \qquad \forall j \in \bar{\mathcal{J}}_m,\ k = 1, \ldots, N-1, \tag{44}$$

$$\sum_{j \in \bar{\mathcal{J}}_m, j \neq l} v_{j,k,l} = w_{l,k+1}, \qquad \forall l \in \bar{\mathcal{J}}_m,\ k = 1, \ldots, N-1, \tag{45}$$

$$\sum_{j\in\bar{\mathcal{J}}_m} w_{j,k} = 1, \qquad \forall k = 1,\dots,N, \tag{46}$$

$$\sum_{k=1}^{N} w_{j,k} = 1, \qquad \forall j \in \bar{\mathcal{J}}_m, \tag{47}$$

$$w_{j,k} \in \{0,1\}, \quad v_{j,k,l} \in \{0,1\}. \tag{48}$$

Constraints (44)–(45) ensure flow conservation between adjacent positions, i.e., if batch j is assigned to position k, it must uniquely correspond to some batch assigned to position $k+1$, and vice versa. This structure ensures that auxiliary variables $v_{j,k,l}$ accurately capture the adjacency relationships in the original quadratic objective function.

Let (LP_m) denote the continuous relaxation of model $(\overline{\mathrm{QAP}}_m)$, and its optimal objective value be denoted as $\underline{h}$, serving as a valid lower bound for the Quadratic Assignment Problem. To further enhance fixing capability, a Dual Picking strategy [12] is introduced, i.e., selecting a set of Lagrange multipliers from all optimal dual solutions such that the reduced costs associated with adjacent variables are maximized. Let α_k and β_j be the dual variables corresponding to position constraints and batch assignment constraints respectively, and $\gamma_{j,k}$ and $\delta_{l,k}$ correspond to constraints (44) and (45) respectively. Based on these definitions, the dual picking problem can be formulated as:

$$(\widehat{\mathrm{DP}}_m) \quad \max_{\alpha,\beta,\gamma,\delta} \quad \sum_{k=1}^{N-1} \sum_{j\in\bar{\mathcal{J}}_m} \sum_{l\in\bar{\mathcal{J}}_m,\, l\neq j} s_{j,k,l}(\gamma,\delta) \tag{49}$$

$$\text{s.t.} \quad s_{j,k,l}(\gamma,\delta) \geq 0, \qquad \forall j,l \in \bar{\mathcal{J}}_m,\ j\neq l,\ k = 1,\dots,N-1, \tag{50}$$

$$c_{j,k}(\alpha,\beta,\gamma,\delta) \geq 0, \qquad \forall j \in \bar{\mathcal{J}}_m,\ k = 1,\dots,N, \tag{51}$$

$$-\sum_{k=1}^{N} \alpha_k - \sum_{j\in\bar{\mathcal{J}}_m} \beta_j = \underline{h}, \tag{52}$$

$$\alpha_k, \beta_j, \gamma_{j,k}, \delta_{l,k} \in \mathbb{R}, \tag{53}$$

Where the reduced cost of auxiliary variable $v_{j,k,l}$ is defined as $s_{j,k,l}(\gamma,\delta) \triangleq s_{j,l} + \gamma_{j,k} + \delta_{l,k}$, and the reduced cost of position variable $w_{j,k}$ is defined as $c_{j,k}(\alpha,\beta,\gamma,\delta) \triangleq \alpha_k + \beta_j - \gamma_{j,k}\mathbf{1}_{\{k\leq N-1\}} - \delta_{j,k-1}\mathbf{1}_{\{k\geq 2\}}$.

Proposition 5 (Reduced Cost Fixing based on Dual Picking) *Let $\underline{h}$ be the optimal value of the linear relaxation (LP_m) of the quadratic assignment problem, and $\overline{h}$ be a feasible upper bound. Let $(\hat{\alpha},\hat{\beta},\hat{\gamma},\hat{\delta})$ be an optimal solution to the dual picking problem $(\widehat{\mathrm{DP}}_m)$, and (w^*,v^*) be the optimal solution to $(\overline{\mathrm{QAP}}_m)$. Then for any $j \in \bar{\mathcal{J}}_m$ and $k \in \{1,\dots,N\}$, the following conclusions hold:*

1. *If $\underline{h} + c_{j,k}(\hat{\alpha},\hat{\beta},\hat{\gamma},\hat{\delta}) > \overline{h}$, then $w^*_{j,k} = 0$.*
2. *If there exists $l \in \bar{\mathcal{J}}_m$ such that $\underline{h} + s_{j,k,l}(\hat{\gamma},\hat{\delta}) > \overline{h}$, then $v^*_{j,k,l} = 0$, and $w^*_{j,k} + w^*_{l,k+1} \leq 1$.*

Proof 5 *For detailed proof, please refer to* Appendix 1.3.

To apply Proposition 5, we first compute a feasible upper bound $\overline{h}$ for $(\overline{\mathrm{QAP}}_m)$ via a fast heuristic. The heuristic quickly constructs a feasible sequence (thus a feasible (w,v)) and evaluates its objective value as $\overline{h}$.

3.4 Benders Cuts

Once the optimal solution $T_m^{set}(\bar{z}_m)$ of Subproblem II is obtained, if the makespan estimated by the Master Problem is too low, the following cuts are added.

1. Combinatorial Benders Cut If the subproblem solution indicates that the current assignment plan $(\bar{\mathcal{I}}_m, \bar{\mathcal{J}}_m)$ causes the makespan to exceed the known upper bound U, implying this specific "worker-batch" combination is infeasible. To exclude this specific combination (and cases containing more batches or exactly the same workers), we add the following cut:

$$\underbrace{\sum_{j\in\bar{\mathcal{J}}_m} z_{mj}}_{\text{Batch Part}} + \underbrace{\sum_{i\in\bar{\mathcal{I}}_m} x_{im} - \sum_{i\in\mathcal{I}\setminus\bar{\mathcal{I}}_m} x_{im}}_{\text{Worker Combination Part}} \leq |\bar{\mathcal{J}}_m| + |\bar{\mathcal{I}}_m| - 1 \tag{54}$$

Proposition 6 *Given the currently infeasible assignment plan $(\bar{\mathcal{I}}_m, \bar{\mathcal{J}}_m)$, the Combinatorial Benders Cut (54) can exactly cut off the current assignment plan $(\bar{\mathcal{I}}_m, \bar{\mathcal{J}}_m)$, and holds for any plan deviating from this assignment (reducing batches, reducing workers, or adding workers).*

Proof 6 *For detailed proof, please refer to* Appendix 1.4.

2. Analytical Benders Cut Let $T_m^{set}(\bar{z}_m)$ be the optimal objective value of Subproblem II, i.e., the minimum sequence-dependent setup time required for Seru m to complete all assigned tasks under the current batch assignment $\bar{z}_m$.

If the current makespan $\bar{C}$ of the Master Problem is less than the actual required total time, add the following cut:

$$C \geq \sum_{j\in\mathcal{J}} l_{mj} + T_m^{set}(\bar{z}_m)\left(\sum_{j\in\bar{\mathcal{J}}_m} z_{mj} - |\bar{\mathcal{J}}_m| + 1\right) \tag{55}$$

Proposition 7 *Let $T_m^{set}(\bar{z}_m)$ be the optimal sequence-dependent setup time for Subproblem II given batch set $\bar{\mathcal{J}}_m$. For any feasible solution $(x, z, p, C_{\max})$ of the Master Problem, the above Analytical Benders Cut always holds.*

Proof 7 *For detailed proof, please refer to* Appendix 1.5.

3.5 Algorithm Acceleration Strategy: Enhanced Sequence-Independent Setup Time Decomposition (SISTD)

To obtain high-quality lower bounds while avoiding repeated solutions of expensive TSP subproblems (i.e., subtour elimination constraint related problems), we propose an Enhanced Sequence-Independent Setup Time Decomposition (SISTD) strategy. Unlike existing heuristic decomposition methods, this strategy constructs a linear programming model to obtain theoretically tightest decomposition parameters, utilizes its dual properties to analyze algorithmic complexity, and reinforces the Master Model through pre-calculated global cuts.

3.5.1 *Linear Programming Model for Optimal Decomposition Parameters*

Definition 1 *For any pair of tasks $u, v \in \mathcal{J}$, decompose the sequence-dependent setup time s_{uv} into two sequence-independent components: intrinsic inbound penalty γ_v^{in} for task v and intrinsic outbound penalty γ_u^{out} for task u. This decomposition must satisfy the relaxed constraint of triangle inequality properties: $\gamma_u^{out} + \gamma_v^{in} \leq s_{uv}$.*

Existing decomposition methods (e.g., max-sum heuristic in [9] or sequential decomposition in [14]) typically rely on fixed calculation orders (e.g., determining inbound components first, then calculating outbound components). This sequential calculation process is often limited by the order of computation, failing to fully extract potential sequence-independent times. To overcome this limitation, we propose using Linear Programming (LP) to optimize all parameters simultaneously, with the objective of "maximizing the sum of sequence-independent components," thereby obtaining a globally optimal decomposition scheme.

In the algorithm initialization phase, for a given batch set $\mathcal{J}$, we solve the following LP model:

$$\max \quad \sum_{v \in \mathcal{J}} (\gamma_v^{in} + \gamma_v^{out}) \tag{56}$$

$$\text{s.t.} \quad \gamma_u^{out} + \gamma_v^{in} \leq s_{uv}, \quad \forall u, v \in \mathcal{J}, u \neq v \tag{57}$$

$$\gamma_v^{in} \geq 0, \gamma_v^{out} \geq 0, \quad \forall v \in \mathcal{J} \tag{58}$$

The above linear programming model contains $2|\mathcal{J}|$ variables and $O(|\mathcal{J}|^2)$ constraints. Notably, its dual problem is equivalent to the Minimum Cost Flow (MCF) problem on a bipartite graph. This special mathematical structure implies that although we use a general LP solver for pre-calculation, in large-scale problems, this parameter can be efficiently solved via polynomial-time combinatorial optimization algorithms [1].

3.5.2 *Valid Inequality Generation Based on Subset Lower Bounds*

Based on the optimal parameters γ_v^{in} and γ_v^{out} solved by the above LP, we can quickly estimate the lower bound of minimum setup cost for any batch subset on a single machine.

For any given batch subset $\bar{\mathcal{J}}_k \subseteq \mathcal{J}$, its minimum setup time lower bound $V_{SISTD}(\bar{\mathcal{J}}_k)$ can be expressed as the sum of intrinsic inbound and outbound penalties of all tasks in the set. However, in actual sequencing, the first task of the path does not need to bear inbound penalty, and the last task does not need to bear outbound penalty. To ensure validity (Valid Lower Bound), we conservatively subtract the possible maximum inbound penalty and maximum outbound penalty in that set from the total sum:

$$V_{SISTD}(\bar{\mathcal{J}}_k) = \sum_{j \in \bar{\mathcal{J}}_k} (\gamma_j^{in} + \gamma_j^{out}) - \left(\max_{j \in \bar{\mathcal{J}}_k} \gamma_j^{in} + \max_{j \in \bar{\mathcal{J}}_k} \gamma_j^{out} \right) \tag{59}$$

Utilizing this property, we randomly generate K groups of batch subsets in the preprocessing phase and add global valid inequalities of the following form to the model. This constraint uses Big-M logic to enforce that when machine m carries all tasks in subset $\bar{\mathcal{J}}_k$ (i.e., $\sum z_{mj} = |\bar{\mathcal{J}}_k|$), its makespan $C_{\max}$ must include the intrinsic minimum setup cost V_{SISTD} of that subset:

$$C_{\max} \geq \sum_{j \in \mathcal{J}} l_{mj} + V_{SISTD}(\bar{\mathcal{J}}_k) \cdot \left(\sum_{j \in \bar{\mathcal{J}}_k} z_{mj} - |\bar{\mathcal{J}}_k| + 1 \right), \quad \forall m \in \mathcal{M}, k \in \{1, \ldots, K\} \tag{60}$$

This series of inequalities utilizes pre-calculated optimal decomposition parameters, significantly tightening the linear relaxation lower bound of $C_{\max}$.

3.5.3 *SISTD-Based Alternating Cut Strategy*

To further reduce the computational burden of the algorithm in the branch-and-bound search tree, we adopt an Alternating Cuts strategy. The core idea is to use the computationally cheaper relaxed lower bound (i.e., the aforementioned SISTD) as a proxy for the expensive exact subproblem (i.e., TSP), thereby reducing the frequency of calling the exact TSP solver.

In the Logic-Based Benders Decomposition (LBBD) framework, candidate solutions $(\hat{\mathbf{z}}, \hat{\theta})$ provided by the Master Problem need to be verified by subproblems. Typically, we need to solve the exact TSP subproblem to obtain the true cost $Q(\hat{\mathbf{z}})$. If $\hat{\theta} < Q(\hat{\mathbf{z}})$, an expensive combinatorial Optimality Cut is added. However, TSP belongs to NP-hard problems, and frequent calls are not only time-consuming but often unnecessary in the early stages of search.

Based on the property $Q_{SISTD}(\hat{\mathbf{z}}) \leq Q_{TSP}(\hat{\mathbf{z}})$, the lower bound Q_{SISTD} provided by SISTD is a convex relaxation of the true cost. This alternating strategy effectively uses Q_{SISTD} as a filter for feasibility checks, preventing unnecessary and expensive exact evaluation of obviously inferior solutions. The specific flow is shown in Algorithm 1.

Algorithm 1 SISTD-Based Alternating Cut Generation

Require: Solution $(\hat{\mathbf{z}}, \hat{\theta})$, Sets V_{LP}, V_{Exact}
Ensure: Feasibility (True/False) and Cuts

if $\hat{\mathbf{z}} \in V_{Exact}$ **then**
  return True
end if
Step 1: SISTD Screening
if $\hat{\mathbf{z}} \notin V_{LP}$ **then**
  $LB \leftarrow$ ComputeSISTD($\hat{\mathbf{z}}$); $V_{LP} \leftarrow V_{LP} \cup \{\hat{\mathbf{z}}\}$
  if $\hat{\theta} < LB$ **then**
    Add SISTD Cut (60);
    return False
  end if
end if
Step 2: Exact TSP Verification
$Cost \leftarrow$ SolveTSP($\hat{\mathbf{z}}$); $V_{Exact} \leftarrow V_{Exact} \cup \{\hat{\mathbf{z}}\}$
if $\hat{\theta} < Cost$ **then**
  Add Optimality Cut;
  return False
end if
return True

4 Experiments and Results Analysis

4.1 Experimental Setup

To comprehensively evaluate the performance of the proposed hybrid classical-quantum Q-LBBD algorithm in solving the ”Order-Device-Worker” collaborative scheduling problem in Seru systems, we designed benchmark experiments based on real industrial scenarios.

4.1.1 *Experimental Environment and Parameters*

All classical computing parts of the experiments were run on a computer equipped with an Intel Core i7-8700 CPU (3.19 GHz) and 8.00 GB RAM. The classical Master Problem and the comparative Mixed-Integer Programming models were solved using the Gurobi 10.0 solver, with the MIP gap tolerance set to 10^{-6} and a global time limit of 3600 seconds.

For the quantum computing part, sequencing subproblems were solved by invoking the simulation interface of the Coherent Ising Machine. The experimental code was written in Python, utilizing the Ising Intelligence SDK interface for hybrid architecture data interaction.

4.1.2 *Benchmark Instance Generation*

Experimental instances were adapted from the classic Seru production benchmark dataset provided by [11], based on actual production data from a large laptop manufacturing factory. Considering the significant sequence-dependent setup time characteristics in the Rotating Seru system studied in this paper, we generated enhanced test instances based on the original dataset combined with actual distribution rules from the industrial site, using the following rules:

- **Problem Scale:** To test algorithm scalability, we set different combinations of scales.
  - Number of workers (available resources) $W = |\mathcal{I}| \in \{5, 6, 7\}$;
  - Number of order batches (task scale) $J = |\mathcal{J}| \in \{8, 9, 10, 15, 20\}$;
  - Number of potential Serus $M = |\mathcal{M}| \in \{3, 4\}$.
- **Worker Processing Time (T_{ij}):** Reflects worker skill heterogeneity. According to [3], different workers have varying proficiency f_{ij}. We generate unit processing times using a uniform distribution within the benchmark range:

$$T_{ij} \sim \mathcal{U}[1.0, 3.0] \tag{61}$$

- **Sequence-Dependent Setup Time (S_{jk}):** This is a key parameter affecting sequencing decisions. Simulate fixture change or material preparation time required to switch between different batches, generated as follows:

$$S_{jk} \sim \mathcal{U}\{2, 8\} \tag{62}$$

- **Batch Size (B_j):** The number of products in each batch follows a discrete uniform distribution to simulate ”high-variety, low-volume” order characteristics:

$$B_j \sim \mathcal{U}\{20, 60\} \tag{63}$$

4.2 Performance Comparison

4.2.1 *Performance of Q-LBBD with Kaiwu SDK*

As shown in Table 2, the SDK-based Q-LBBD exhibits clear strengths over GUROBI in runtime and robustness, but may sacrifice optimality in a few cases:

- **Moderate Efficiency**: For most instances where both methods finish, Q-LBBD is faster, with **time reductions ranging from 4.97% to 95.46%**.
- **Robustness on hard cases**: When GUROBI hits the 3600-second limit (e.g., $W = 7$ and some $W = 5, 6$ cases), **Q-LBBD still returns solutions quickly (41.88–2315.58 s), achieving 96.54%–98.84%** improvement under the cutoff.
- **Quality trade-off**: Many solvable cases have identical objectives (Gap$= 0$), but several instances show **nonzero gaps (0.22%–3.18%)**, indicating a small loss of optimality.

Table 2 Performance Comparison and Improvement Rate: GUROBI vs. Q-LBBD (Kaiwu SDK)

W	J	$M=3$						$M=4$					
		GUROBI		Q-LBBD		**Imp.**	Gap	GUROBI		Q-LBBD		**Imp.**	Gap
		t(s)	obj	t(s)	obj	**(%)**	(%)	t(s)	obj	t(s)	obj	**(%)**	(%)
5	8	35.48	93.62	32.17	93.62	**9.33**	**0.00**	189.4	89.26	12.87	89.26	**93.20**	**0.00**
	9	39.53	114.16	43.40	114.16	**-**	**0.00**	322.27	111.96	58.58	111.96	**81.82**	**0.00**
	10	67.42	127.44	64.07	129.83	**4.97**	**1.87**	215.71	118.33	52.49	118.33	**75.67**	**0.00**
	15	3139.08	187.56	440.12	187.97	**85.98**	0.22	>3600	-	169.99	174.82	**95.28**	-
	20	>3600	-	2315.58	258.68	**35.68**	-	>3600	-	380.53	228.83	**89.43**	-
6	8	181.44	81.04	34.06	83.62	**81.23**	**3.18**	591.68	77.31	26.86	77.31	**95.46**	**0.00**
	9	140.79	97.52	54.50	97.52	**61.29**	**0.00**	783.95	90.58	55.75	90.58	**92.89**	**0.00**
	10	334.01	106.70	73.44	106.93	**78.01**	**0.22**	>3600	-	62.27	105.30	**98.27**	-
	15	>3600	-	329.89	157.81	**90.84**	-	>3600	-	262.32	149.24	**92.71**	-
7	8	>3600	-	46.95	70.92	**98.70**	-	>3600	-	41.88	70.06	**98.84**	-
	9	>3600	-	62.31	86.19	**98.27**	-	>3600	-	50.05	80.43	**98.61**	-
	10	>3600	-	124.63	94.25	**96.54**	-	>3600	-	99.70	89.39	**97.23**	-

4.2.2 *Performance of Q-LBBD with CIM*

Table 3 shows that Q-LBBD on **CIM** is much faster than GUROBI (queuing time excluded) while keeping high solution quality.

- **Extreme Speed Advantage**: For $W = 5$, $J = 8$–10, $M = 3$, Q-LBBD cuts runtime by about **98%–99%**.
- **Solvability of Larger Scale**: For larger J, GUROBI often times out, whereas Q-LBBD still returns solutions quickly .
- **Optimality Guarantee**: Whenever GUROBI reaches optimality, Q-LBBD matches the objective (Gap = **0.00%**).
- **Advantage over SDK**: Relative to Table 2, **CIM** provides an additional **one to two orders of magnitude** acceleration, and sometimes yields **better objectives** than the SDK results.

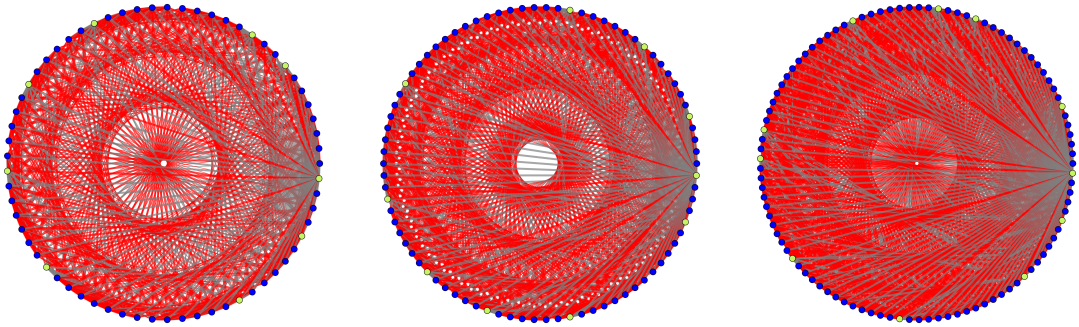
Table 3 Performance Comparison and Improvement Rate: GUROBI vs. Q-LBBD (CIM)

W	J	$M=3$						$M=4$					
		GUROBI		Q-LBBD		**Imp.**	Gap	GUROBI		Q-LBBD		**Imp.**	Gap
		t(s)	obj	t(s)	obj	**(%)**	(%)	t(s)	obj	t(s)	obj	**(%)**	(%)
5	8	35.48	93.62	0.31	93.62	**99.11**	**0.00**	189.40	89.26	0.65	89.26	**99.66**	**0.00**
	9	39.53	114.16	0.73	114.16	**98.14**	**0.00**	322.27	111.96	1.12	111.96	**99.65**	**0.00**
	10	67.42	127.44	1.01	127.44	**98.50**	**0.00**	215.71	118.33	1.24	118.33	**99.43**	**0.00**
	15	3139.08	187.56	9.79	187.56*	**99.69**	**0.00**	>3600	-	8.88	174.82	**99.75**	-
	20	>3600	-	25.75	245.67	**99.28**	-	>3600	-	16.81	226.86*	**99.53**	-
6	8	181.44	81.04	1.73	81.04	**99.05**	**0.00**	591.68	77.31	1.38	77.31	**99.77**	**0.00**
	9	140.79	97.52	1.40	97.52	**99.01**	**0.00**	783.95	90.58	2.37	90.58	**99.70**	**0.00**
	10	334.01	106.70	2.18	106.70	**99.35**	**0.00**	>3600	-	2.39	105.00*	**99.93**	-
	15	>3600	-	17.44	157.81	**99.52**	-	>3600	-	14.64	149.24	**99.59**	-
7	8	>3600	-	1.70	70.92	**99.95**	-	>3600	-	2.90	70.06	**99.92**	-
	9	>3600	-	2.95	85.62*	**99.92**	-	>3600	-	2.75	80.43	**99.92**	-
	10	>3600	-	10.02	94.25	**99.72**	-	>3600	-	9.79	89.39	**99.73**	-

* denotes an objective value better than that obtained using Kaiwu SDK.

4.2.3 *Scalability analysis of Sequencing Subproblem*

To illustrate sequencing subproblem at different scales, we visualize its optimal solutions using the equivalent **Max-cut** form in Figure 6. This is natural since the subproblem maps to an Ising model. In the plots, node colors indicate spins (blue: $+1$, green: -1); gray edges are within a set and red edges are across sets. Figures 6(a)–(c) correspond to $J = 8, 9, 10$. As J increases, the **number of spins and couplings grows rapidly**, leading to more **complex cut patterns and higher computational difficulty**.



(a) $J = 8$ **(65-bit)** **(b)** $J = 9$ **(82-bit)** **(c)** $J = 10$ **(101-bit)**

Figure 6 Solution visualization of sequencing subproblem instances at different scales

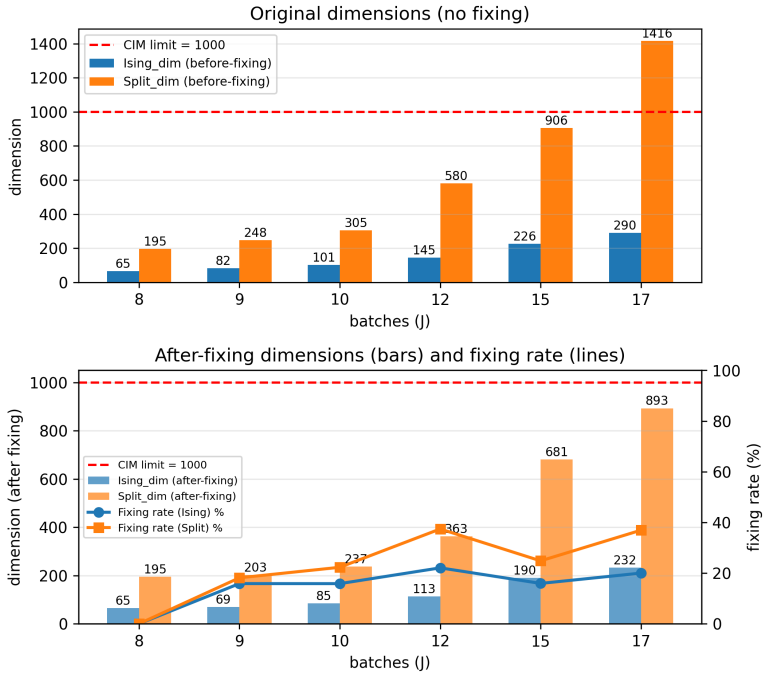


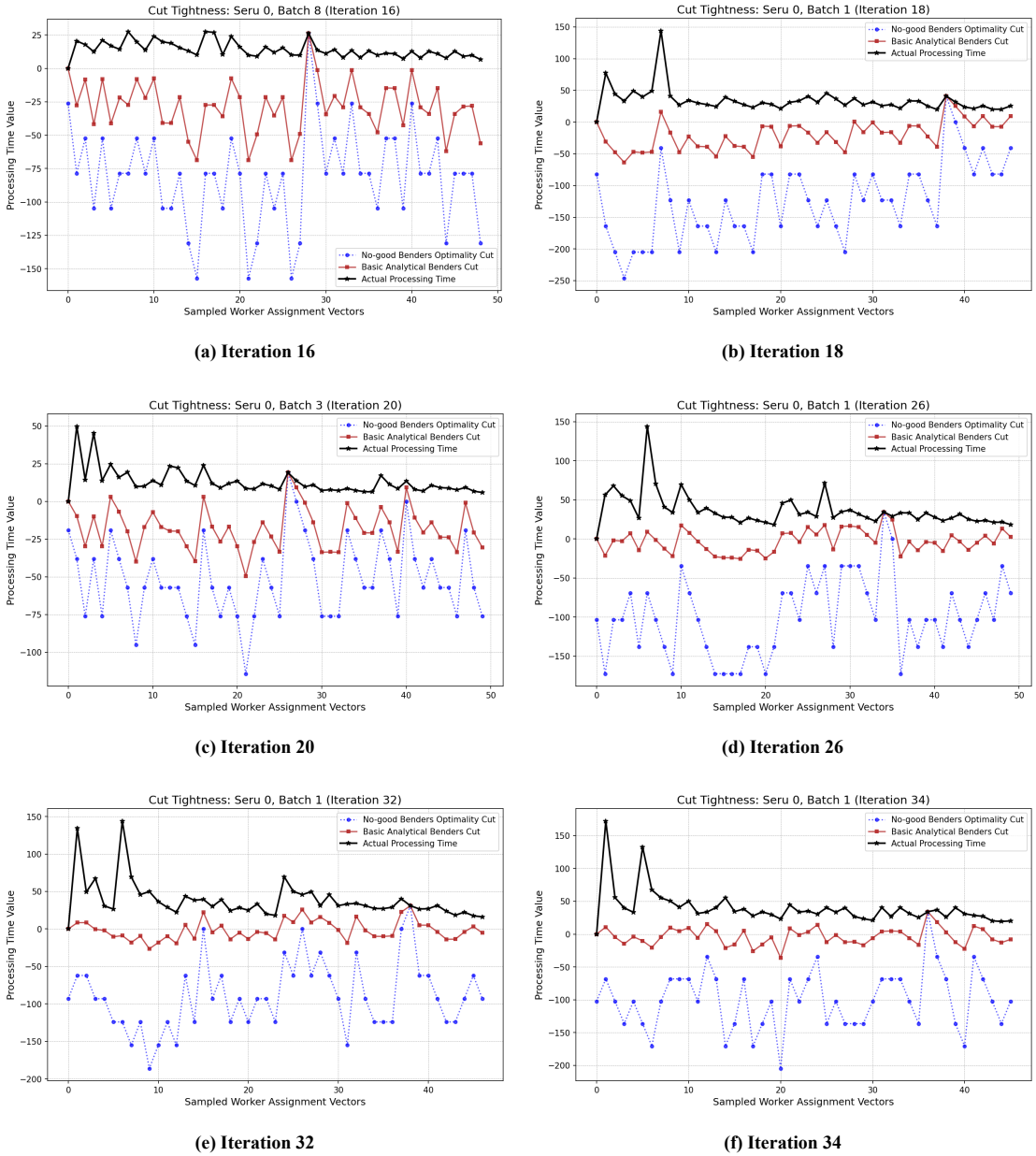
Figure 7 Ising dimension before/after precision adaptation and variable fixing.

Figure 7 shows that CIM scalability is mainly limited by precision adaptation. While the raw Ising dimension stays modest as J grows (e.g., 290 at $J = 17$), coefficient-splitting for 8-bit precision introduces many auxiliary variables, inflating the precision-adapted size to 1416 **at** $J = 17$, which exceeds the CIM limit (dimension $= 1000$). Applying the proposed **variable reduction technique** to $(\mathrm{QAP_m})$ before conversion fixes non-competitive variables while **preserving optimality**, and significantly curbs this blow-up: for $J = 17$, the precision-adapted dimension drops from 1416 **to** 893, bringing the instance back under the CIM capacity.

4.2.4 *Comparison of Cut Tightness*

To verify the effectiveness of the proposed Basic Analytical Benders Cut, we compared the tightness of the lower bounds generated by it against the classic No-good Benders Optimality Cut during the iterative solution process of the Q-LBBD algorithm. As shown in Figure 8, we selected

several key sample points (or specific iteration steps) during the algorithm iteration to plot the comparison between the lower bounds provided by different cuts and the true values. In the figure, the black solid line represents the true optimal objective value of the subproblem corresponding to the current Master Problem solution, serving as the comparison baseline; the blue dotted line depicts the lower bound approximation provided by the No-good Cut; the red solid line corresponds to the lower bound approximation of the Analytical Cut proposed in this study.



(a) Iteration 16 **(b) Iteration 18**

(c) Iteration 20 **(d) Iteration 26**

(e) Iteration 32 **(f) Iteration 34**

Figure 8 Tightness comparison between the proposed Analytical cut and No-good cut

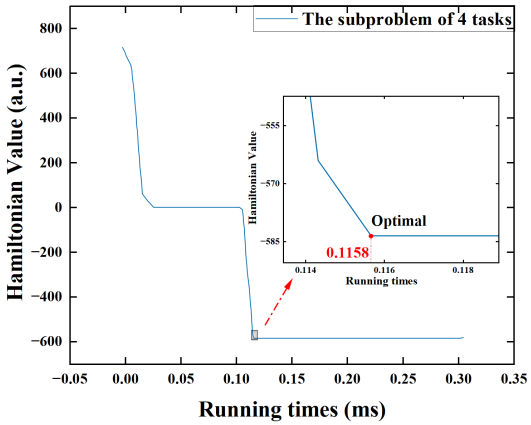
Observations show that the red curve is strictly located above the blue curve and approximates the black baseline more significantly. This phenomenon intuitively reveals that the **Analytical Cut possesses stronger tightness in the solution space**. Specifically, since the No-good Cut is exact only at the current solution (i.e., lower bound value equals true value), while its lower bound value drops rapidly in regions slightly deviating from the current solution, leading to weak pruning capa-

bility on the solution space. In contrast, the Analytical Cut utilizes key information on how worker processing time changes with the number of people, constructing a tighter supporting hyperplane. This superior tightness means the Analytical Cut can provide higher quality lower bound information, thereby more effectively **reducing the search space of the Master Problem and accelerating the algorithm's convergence process**.

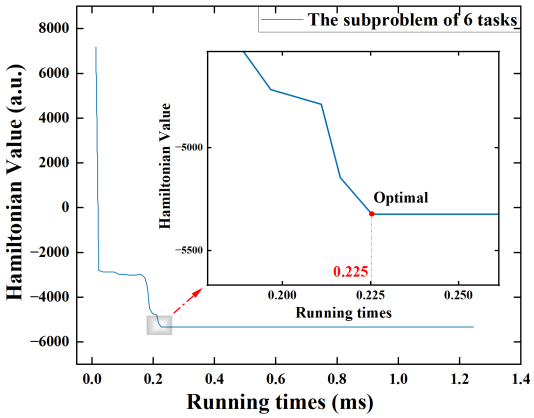
4.3 Quantum Computing Performance Analysis

4.3.1 *Ising Machine Real Hardware Solving Efficiency*

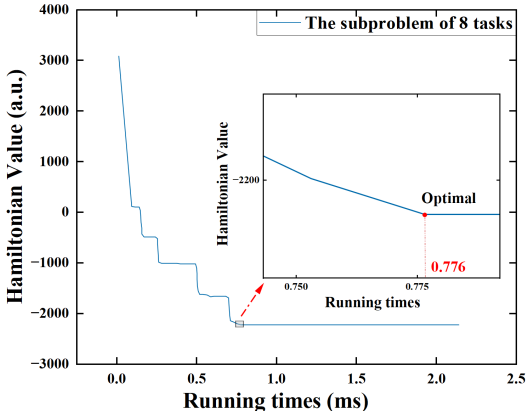
To evaluate the actual performance of quantum hardware in subproblem solving, we analyzed the distribution of running times of the Ising machine under multiple calls, as shown in Figure 9. Results indicate that the source of this acceleration is the rapid solving of the sequencing subproblems within the proposed Q-LBBD algorithm. Figure 9 shows the Hamiltonian energy evolution for subproblems of different sizes on the CIM. The plots show that the solver consistently finds a low-energy solution, corresponding to an optimal task sequence, on **a millisecond timescale** (e.g., 0.826ms for 10 tasks). This efficiency at the subproblem level directly contributes to the substantial reduction in the overall algorithm's runtime.



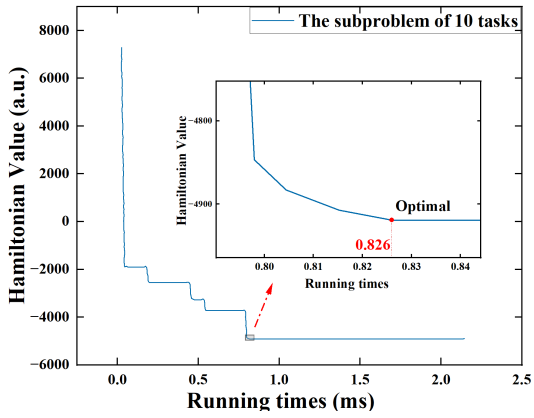
(a) 4 batches



(b) 6 batches



(c) 8 batches

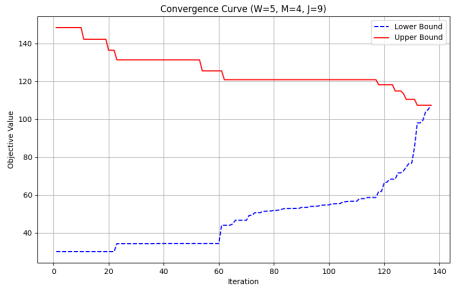


(d) 10 batches

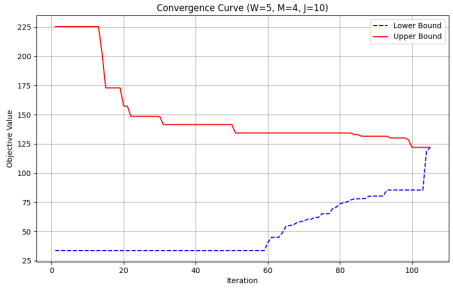
Figure 9 Hamiltonian time evolution for different subproblem sizes.

4.3.2 *Algorithm Convergence Analysis*

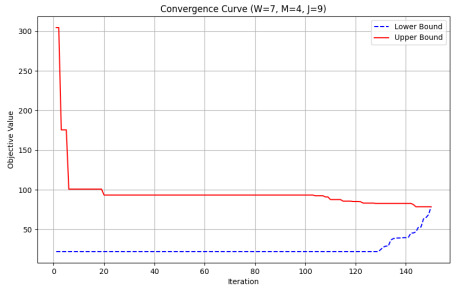
To examine convergence, Figure 10 plots the upper and lower bounds for instances of different scales. In all tested settings, the upper bound decreases quickly at the beginning, while the lower bound increases steadily as branching and cuts are added. As a result, the gap keeps shrinking and becomes tight. This consistent pattern across different (W, M, J) indicates stable and repeatable convergence of Q-LBBD with the Ising machine as the subproblem solver.



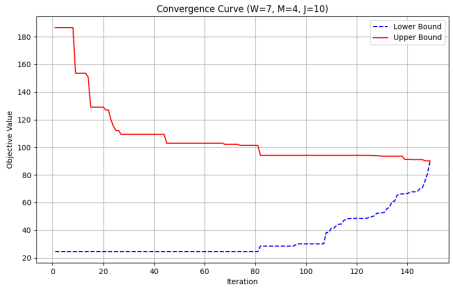
(a) $(W = 5,\ M = 4,\ J = 9)$



(b) $(W = 5,\ M = 4,\ J = 10)$



(c) $(W = 7,\ M = 4,\ J = 9)$



(d) $(W = 7,\ M = 4,\ J = 10)$

Figure 10 Q-LBBD Upper and Lower Bound Convergence Curves

5 Conclusion

This paper addresses the integrated ”Order-Device-Worker” scheduling in Seru production, a complex NP-hard problem involving nonlinear and sequence-dependent setup times. To overcome computational bottlenecks in large-scale scenarios, we propose a novel **Hybrid Classical-Quantum Logic-Based Benders Decomposition** (Q-LBBD) algorithm.

First, to accelerate convergence, a **Sequence-Independent Setup Time Decomposition (SISTD)** strategy is developed. It uses linear programming to extract sequence-independent time components, providing a tighter lower bound and generating efficient Benders cuts, thereby reducing costly subproblem calls.

Second, to solve **NP-hard subproblems** efficiently, we reformulate them as Quadratic Unconstrained Binary Optimization (QUBO) models and solve them using a Coherent Ising Machine, creating a hybrid computing paradigm. A **variable reduction technique** is designed to overcome qubit limitations, enabling the solution of industrial-scale problems with limited quantum resources.

Numerical studies demonstrate that the proposed Q-LBBD algorithm, implemented by integrating a coherent Ising machine (CIM) to solve the sequencing subproblems, **guarantees global optimality and reduces computation time by over 97%** relative to Gurobi on large-scale instances, verifying the superiority of the CIM-enabled hybrid quantum computing paradigm in industrial scheduling.

Future work may extend the model to handle uncertainties like order fluctuations and worker efficiency, and explore other quantum computing paradigms such as QAOA for further performance comparison and optimization potential.

References

[1] Li Chen, Rasmus Kyng, Yang Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. Journal of the ACM, 72(3):1–103, 2025. https://doi.org/10.1145/3728631.

[2] Thinh Q. Dinh, Son Hoang Dau, Eva Lagunas, and Symeon Chatzinotas. Efficient hamiltonian reduction for quantum annealing on satcom beam placement problem. In ICC 2023 - IEEE International Conference on Communications, pages 2668–2673, 2023. https://doi.org/10.1109/ICC45041.2023.10279813.

[3] Xiaolong Li, Yang Yu, Wei Sun, and Jiafu Tang. Reducing tardy batches by seru production: Model, exact solution, cooperative coevolution solution, and insights. Computers & Operations Research, 160:106048, 2023. https://doi.org/10.1016/cor.2023.109126.

[4] Wei Sun, Yang Yu, Qi Lou, Junwei Wang, and Yuechao Guan. Reducing the total tardiness by seru production: model, exact and cooperative coevolution solutions. International Journal of Production Research, 58(21):6441–6452, 2020. doi:10.1080/00207543.2019.1680898.

[5] Liangyan Tao, Rui Tao, and Naiming Xie. Optimization of the seru production system with

demand fluctuation: A mean-cvar model. Computers Industrial Engineering, 199:110760, 2025.

[6] Phuc Thai, My T. Thai, Tam Vu, and Thang N. Dinh. Fasthare: Fast hamiltonian reduction for large-scale quantum annealing. 2022 IEEE International Conference on Quantum Computing and Engineering, 1(1):114–124, 2022. https://doi.org/10.1109/QCE53715.2022.00028.

[7] Ziqing Wang, Wenzhu Liao, and Yaping Zhang. Rescheduling optimisation of sustainable multi-objective fuzzy flexible job shop under uncertain environment. International Journal of Production Research, pages 1–17, 2024. doi:10.1080/00207543.2024.2354830.

[8] Yuting Wu, Ling Wang, Jing-fang Chen, Jie Zheng, and Zixiao Pan. A reinforcement learning driven two-stage evolutionary optimisation for hybrid seru system scheduling with worker transfer. International Journal of Production Research, pages 1–20, 2023. doi:10.1080/00207543.2023.2252523.

[9] Wucheng Yang and Wenming Cheng. Modelling and solving mixed-model two-sided assembly line balancing problem with sequence-dependent setup time. International Journal of Production Research, 58(21):6638–6659, 2020. https://doi.org/10.1080/00207543.2019.1683255.

[10] Yong Yin, Kathryn E Stecke, Morgan Swink, and Ikou Kaku. Lessons from seru production on manufacturing competitively in a high cost environment. Journal of Operations Management, 49:67–76, 2017. doi:10.1016/j.jom.2017.01.003.

[11] Yang Yu, Jiafu Tang, Jun Gong, Yong Yin, and Ikou Kaku. Mathematical analysis and solutions for multi-objective line-cell conversion problem. European Journal of Operational Research, 236(2):774–786, 2014. doi:10.1016/j.ejor.2014.01.029.

[12] Zixuan Yu, Wei Sun, Jianwen Zhou, Yang Liu, Jiao Yu, and Xiaolong Li. Exact method based on solution space cut for bi-objective seru production. Journal of Industrial and Management Optimization, 19(12):8709–8730, 2023. https://doi.org/10.3934/jimo.2023058.

[13] Zhe Zhang, Xue Gong, Xiaoling Song, Yong Yin, Benjamin Lev, and Jie Chen. A column generation-based exact solution method for seru scheduling problems. Omega, 108:102581, 2022. https://doi.org/108:102581.

[14] Hassan Zohali, Bahman Naderi, and Vahid Roshanaei. Solving the type-2 assembly line balancing with setups using logic-based benders decomposition. INFORMS Journal on Computing, 34(1):315–332, 2022. https://doi.org/10.1287/ijoc.2021.1097.

Appendix

Appendix 1.1 Proof of Proposition 3

Proof 3 *At point $\bar{x}$, $\bar{l}(\bar{x}) = \sum_{i\in\bar{S}} \frac{a_i}{m^2} = f(\bar{x})$. It suffices to prove that for any $x \in X$, i.e., for any $S \subseteq \mathcal{I}$ ($|S| \geq 1$), the following holds:*

$$f(x) - \bar{l}(x) = \sum_{i\in S} \left(\frac{a_i}{|S|^2} - \alpha_i \right) \geq 0.$$

(1) When $1 \leq |S| \leq m$, for any $i = 1, \cdots, |\mathcal{I}|$: If $i \in \bar{S}$, then

$$\alpha_i = \frac{a_i}{m^2} \leq \frac{a_i}{|S|^2},$$

If $i \notin \bar{S}$, then

$$\alpha_i = \frac{a_i}{|\mathcal{I}|^2} + \left(\frac{1}{(m+1)^2} - \frac{1}{m^2} \right) \sum_{i\in\bar{S}} a_i \leq \frac{a_i}{|\mathcal{I}|^2} \leq \frac{a_i}{|S|^2}.$$

Therefore, when $1 \leq |S| \leq m$, $f(x) - \bar{l}(x) \geq 0$.

(2) When $m + 1 \leq |S| \leq |\mathcal{I}|$:

$$\sum_{i\in S} \left(\frac{a_i}{|S|^2} - \alpha_i \right) = \sum_{i\in S\cap\bar{S}} \left(\frac{a_i}{|S|^2} - \frac{a_i}{m^2} \right) + \sum_{i\in S\setminus\bar{S}} \left(\frac{a_i}{|S|^2} - \frac{a_i}{|\mathcal{I}|^2} + \Delta_m \sum_{i\in\bar{S}} a_i \right)$$

Where $\Delta_m = \frac{1}{m^2} - \frac{1}{(m+1)^2} = \frac{1}{m(m+1)} \left(\frac{1}{m} + \frac{1}{m+1} \right)$. Let $|S \setminus \bar{S}| = l \geq 1$, so $|S| \leq m + l$.

Since:

$$\left(\frac{1}{|S|^2} - \frac{1}{m^2} \right) \sum_{i\in S\cap\bar{S}} a_i \geq \left(\frac{1}{|S|^2} - \frac{1}{m^2} \right) \sum_{i\in\bar{S}} a_i \geq \left(\frac{1}{(m+l)^2} - \frac{1}{m^2} \right) \sum_{i\in\bar{S}} a_i,$$

And $\frac{a_i}{|S|^2} - \frac{a_i}{|\mathcal{I}|^2} \geq 0$,

$$\sum_{i\in S\setminus\bar{S}} \left(\frac{a_i}{|S|^2} - \frac{a_i}{|\mathcal{I}|^2} + \Delta_m \sum_{i\in\bar{S}} a_i \right) \geq \sum_{i\in S\setminus\bar{S}} \Delta_m \sum_{i\in\bar{S}} a_i = l\Delta_m \sum_{i\in\bar{S}} a_i.$$

Therefore:

$$\sum_{i\in S} \left(\frac{a_i}{|S|^2} - \alpha_i \right) \geq \sum_{i\in\bar{S}} a_i \left(\frac{1}{(m+l)^2} - \frac{1}{m^2} + l\Delta_m \right).$$

Because $\Delta_m \geq \Delta_{m+1} \geq \cdots \geq \Delta_{m+l-1}$, we have:

$$l\Delta_m \geq \Delta_m + \Delta_{m+1} + \cdots + \Delta_{m+l-1} \geq \frac{1}{m^2} - \frac{1}{(m+l)^2}.$$

Appendix 1.2 Proof of Proposition 4

Proof 4 *For any $x \in X$, let $S = \{i : x_i = 1\}$, then:*

$$\begin{aligned}
\bar{l}(\bar{x}) - \hat{l}(x) = & \sum_{i \in S \cap \bar{S}} \left(\frac{a_i}{m^2} - A \right) + \sum_{i \in S \setminus \bar{S}} \left(\frac{a_i}{|\mathcal{I}|^2} - \Delta_m \sum_{i \in \bar{S}} a_i + A \right) + (m-1)A \\
\geq & \sum_{i \in \bar{S}} \left(\frac{a_i}{m^2} - A \right) + \sum_{i \in S \setminus \bar{S}} \left(\frac{a_i}{|\mathcal{I}|^2} - \Delta_m \sum_{i \in \bar{S}} a_i + A \right) + (m-1)A \\
= & \sum_{i \in S \setminus \bar{S}} \left(\frac{a_i}{|\mathcal{I}|^2} + \frac{1}{(m+1)^2} \sum_{i \in \bar{S}} a_i \right) \geq 0,
\end{aligned}$$

Where $A = \sum_{i \in \bar{S}} \frac{a_i}{m^2}$.

Appendix 1.3 Proof of Proposition 5

Proof 5 *We prove the two statements by contradiction using the reduced costs induced by an optimal dual-picked solution $(\hat{\alpha}, \hat{\beta}, \hat{\gamma}, \hat{\delta})$ of $(\widehat{\mathrm{DP}}_m)$.*

***1. Fixing of** $w_{j,k}$. Fix any $j_0 \in \bar{\mathcal{J}}_m$ and $k_0 \in \{1, \ldots, N\}$. Assume for contradiction that $w^*_{j_0,k_0} = 1$ in an optimal integer solution (w^*, v^*) of $(\overline{\mathrm{QAP}}_m)$. Let $z^w(j_0, k_0)$ be the optimal value of the LP relaxation (LP_m) with the additional constraint $w_{j_0,k_0} = 1$. Since (LP_m) is a relaxation of $(\overline{\mathrm{QAP}}_m)$ and $\overline{h}$ is a feasible upper bound,*

$$z^w(j_0, k_0) \;\leq\; \mathrm{val}(\overline{\mathrm{QAP}}_m) \;\leq\; \overline{h}.$$

Consider the Lagrangian associated with the constraints of (LP_m) using multipliers $(\alpha, \beta, \gamma, \delta)$. After regrouping terms, it can be written as

$$L(w, v, \alpha, \beta, \gamma, \delta) = -\sum_{k=1}^{N} \alpha_k - \sum_{j \in \bar{\mathcal{J}}_m} \beta_j + \sum_{j \in \bar{\mathcal{J}}_m} \sum_{k=1}^{N} c_{j,k}(\alpha, \beta, \gamma, \delta)\, w_{j,k} + \sum_{k=1}^{N-1} \sum_{j \in \bar{\mathcal{J}}_m} \sum_{\substack{l \in \bar{\mathcal{J}}_m \\ l \neq j}} s_{j,k,l}(\gamma, \delta)\, v_{j,k,l}.$$

For $(\hat{\alpha}, \hat{\beta}, \hat{\gamma}, \hat{\delta})$ feasible to $(\widehat{\mathrm{DP}}_m)$, we have $c_{j,k}(\hat{\alpha}, \hat{\beta}, \hat{\gamma}, \hat{\delta}) \geq 0$ and $s_{j,k,l}(\hat{\gamma}, \hat{\delta}) \geq 0$, and moreover $-\sum_{k=1}^{N} \hat{\alpha}_k - \sum_{j \in \bar{\mathcal{J}}_m} \hat{\beta}_j = \underline{h}$. Hence,

$$\min_{\substack{w \geq 0,\ v \geq 0 \\ w_{j_0,k_0} = 1}} L(w, v, \hat{\alpha}, \hat{\beta}, \hat{\gamma}, \hat{\delta}) = \underline{h} + c_{j_0,k_0}(\hat{\alpha}, \hat{\beta}, \hat{\gamma}, \hat{\delta}),$$

because all other coefficients are nonnegative and are minimized by setting the corresponding variables to 0. By weak duality (Lagrangian relaxation provides a lower bound),

$$z^w(j_0, k_0) \;\geq\; \underline{h} + c_{j_0,k_0}(\hat{\alpha}, \hat{\beta}, \hat{\gamma}, \hat{\delta}).$$

If $\underline{h} + c_{j_0,k_0}(\hat{\alpha}, \hat{\beta}, \hat{\gamma}, \hat{\delta}) > \overline{h}$, this contradicts $z^w(j_0, k_0) \leq \overline{h}$. Therefore $w^*_{j_0,k_0} = 0$.

***2. Fixing of** $v_{j,k,l}$ **and adjacency exclusion.** Fix any $j_0, l_0 \in \bar{\mathcal{J}}_m$ with $j_0 \neq l_0$ and any $k_0 \in \{1, \ldots, N-1\}$. By the same argument as in (1), if $\underline{h} + s_{j_0,k_0,l_0}(\hat{\gamma}, \hat{\delta}) > \overline{h}$, then we must have $v^*_{j_0,k_0,l_0} = 0$.*

Moreover, since $v_{j,k,l}$ represents the adjacency of (j, k) and $(l, k+1)$ enforced by the flow conservation constraints (44)–(45), $v^*_{j_0,k_0,l_0} = 0$ implies that batches j_0 and l_0 cannot be assigned consecutively at positions k_0 and $k_0 + 1$, i.e., $w^*_{j_0,k_0} + w^*_{l_0,k_0+1} \leq 1$.

Appendix 1.4 Proof of Proposition 6

Proof 6 *We discuss the value of the LHS expression* $LHS(x,z) = \sum_{j\in\bar{\mathcal{J}}_m} z_{mj} + \sum_{i\in\bar{\mathcal{I}}_m} x_{im} - \sum_{i\in\mathcal{I}\setminus\bar{\mathcal{I}}_m} x_{im}$ *by cases.*

1. ***Case 1: Current worker and batch assignment plan recurs*** *If* $z_{mj} = 1, \forall j \in \bar{\mathcal{J}}_m$ *(retain all batches), and* $x_{im} = 1, \forall i \in \bar{\mathcal{I}}_m$ *(retain all original workers), and* $x_{im} = 0, \forall i \notin \bar{\mathcal{I}}_m$ *(no new workers added). Then:*

$$LHS = |\bar{\mathcal{J}}_m| + |\bar{\mathcal{I}}_m| - 0 = |\bar{\mathcal{J}}_m| + |\bar{\mathcal{I}}_m|$$

Substitute into inequality:

$$|\bar{\mathcal{J}}_m| + |\bar{\mathcal{I}}_m| \leq |\bar{\mathcal{J}}_m| + |\bar{\mathcal{I}}_m| - 1 \implies 0 \leq -1 \quad (\textit{Contradiction})$$

Thus, the current infeasible solution is successfully cut.

2. ***Case 2: Changing batch allocation*** *If at least one batch* $j^* \in \bar{\mathcal{J}}_m$ *is removed (i.e.,* $z_{mj^*} = 0$*), then the first sum is at most* $|\bar{\mathcal{J}}_m| - 1$*. The maximum possible value of* LHS *is:*

$$LHS \leq (|\bar{\mathcal{J}}_m| - 1) + |\bar{\mathcal{I}}_m| - 0 = |\bar{\mathcal{J}}_m| + |\bar{\mathcal{I}}_m| - 1$$

The inequality holds.

3. ***Case 3: Changing worker assignment***
   - ***Reducing workers:*** *If at least one original worker* $i^* \in \bar{\mathcal{I}}_m$ *is removed (i.e.,* $x_{mi^*} = 0$*), the second sum is at most* $|\bar{\mathcal{I}}_m| - 1$*.* LHS *decreases by 1, inequality holds.*
   - ***Adding workers:*** *If at least one new worker* $k \in \mathcal{I}\setminus\bar{\mathcal{I}}_m$ *is added (i.e.,* $x_{mk} = 1$*), the third term (penalty term) is at least 1, i.e.,* $-\sum x_{im} \leq -1$*.*
   - *Then the maximum value of* LHS *is:*

$$LHS \leq |\bar{\mathcal{J}}_m| + |\bar{\mathcal{I}}_m| - 1$$

The inequality holds.

In summary, this cut only cuts the specific infeasible state of "current batch set processed by current pure worker set (or its subset)", while allowing any form of relaxation (reducing batches) or enhancement (adding new workers), satisfying validity requirements.

Appendix 1.5 Proof of Proposition 7

Proof 7 *Let* $\mathcal{J}_m = \{j \in \mathcal{J} \mid z_{mj} = 1\}$ *denote the actual set of batches assigned to Seru* m *in the Master Problem. According to the definition of the Master Problem model,* $C_{\max}$ *must be greater than or equal to the sum of total processing time and total setup time in that unit, i.e.:*

$$C_{\max} \geq \sum_{j\in\mathcal{J}} l_{mj} + T^{set}(\mathcal{J}_m)$$

Where $T^{set}(\mathcal{J}_m)$ *denotes the minimum sequence-dependent setup time corresponding to set* $\mathcal{J}_m$*.*

Let the RHS of the cut inequality be RHS*. We discuss two cases based on the inclusion relationship between batch set* $\bar{\mathcal{J}}_m$ *and the current actual assignment set* $\mathcal{J}_m$*:*

1. ***Case 1: Current batch assignment does not fully contain*** $\bar{\mathcal{J}}_m$

   If $\bar{\mathcal{J}}_m \not\subseteq \mathcal{J}_m$*, it means there exists at least one batch* $k \in \bar{\mathcal{J}}_m$ *not assigned to Seru* m *(i.e.,* $z_{mk} = 0$*). Then, the summation term* $\sum_{j\in\bar{\mathcal{J}}_m} z_{mj} \leq |\bar{\mathcal{J}}_m| - 1$.

   Consider the logic term on the RHS of the inequality:

$$\left(\sum_{j\in\bar{\mathcal{J}}_m} z_{mj} - |\bar{\mathcal{J}}_m| + 1\right) \leq (|\bar{\mathcal{J}}_m| - 1) - |\bar{\mathcal{J}}_m| + 1 = 0$$

   Since setup time $T_m^{set}(\bar{z}_m) \geq 0$*, the product term is non-positive. Thus:*

$$RHS \leq \sum_{j\in\mathcal{J}} l_{mj}$$

   It is evident that the makespan must be greater than or equal to pure processing time, i.e., $C_{\max} \geq \sum_{j\in\mathcal{J}} l_{mj}$ *always holds. Therefore, in this case,* $C_{\max} \geq RHS$ *holds.*

2. ***Case 2: Current batch assignment contains all elements of*** $\bar{\mathcal{J}}_m$

   If $\bar{\mathcal{J}}_m \subseteq \mathcal{J}_m$*, meaning for all* $j \in \bar{\mathcal{J}}_m$*,* $z_{mj} = 1$*. Then, the summation term* $\sum_{j\in\bar{\mathcal{J}}_m} z_{mj} = |\bar{\mathcal{J}}_m|$.

   The value of the logic term is:

$$\left(\sum_{j\in\bar{\mathcal{J}}_m} z_{mj} - |\bar{\mathcal{J}}_m| + 1\right) = |\bar{\mathcal{J}}_m| - |\bar{\mathcal{J}}_m| + 1 = 1$$

   Thus, $RHS = \sum_{j\in\mathcal{J}} l_{mj} + T_m^{set}(\bar{z}_m)$.

   Since $\bar{\mathcal{J}}_m \subseteq \mathcal{J}_m$*, i.e.,* $\mathcal{J}_m$ *is a superset (or equal) of* $\bar{\mathcal{J}}_m$*. According to the properties of the Traveling Salesman Problem (TSP), satisfying triangle inequality, the path cost of visiting more nodes will not be lower than the path cost of visiting its subset, i.e.:*

$$T^{set}(\mathcal{J}_m) \geq T^{set}(\bar{\mathcal{J}}_m) = T_m^{set}(\bar{z}_m)$$

   Substituting into the definition of $C_{\max}$*:*

$$C_{\max} \geq \sum_{j\in\mathcal{J}} l_{mj} + T^{set}(\mathcal{J}_m) \geq \sum_{j\in\mathcal{J}} l_{mj} + T_m^{set}(\bar{z}_m) = RHS$$

   Therefore, the inequality holds in this case as well.

In conclusion, regardless of how the Master Problem assigns batches, this Analytical Benders Cut always provides a valid lower bound for $C_{\max}$.