

Petri Net Modeling and Deadlock-Free Scheduling of Attachable Heterogeneous AGV Systems

Boyu Li , Graduate Student Member, IEEE, Zhengchen Li , Graduate Student Member, IEEE, Weimin Wu , Senior Member, IEEE, and Mengchu Zhou , Fellow, IEEE

arXiv:2508.00724v1 [eess.SY] 1 Aug 2025

Abstract—The increasing demand for automation and flexibility drives the widespread adoption of heterogeneous automated guided vehicles (AGVs). This work intends to investigate a new scheduling problem in a material transportation system consisting of attachable heterogeneous AGVs, namely carriers and shuttles. They can flexibly attach to and detach from each other to cooperatively execute complex transportation tasks. While such collaboration enhances operational efficiency, the attachment-induced synchronization and interdependence render the scheduling coupled and susceptible to deadlock. To tackle this challenge, Petri nets are introduced to model AGV schedules, well describing the concurrent and sequential task execution and carrier-shuttle synchronization. Based on Petri net theory, a firing-driven decoding method is proposed, along with deadlock detection and prevention strategies to ensure deadlock-free schedules. Furthermore, a Petri net-based metaheuristic is developed in an adaptive large neighborhood search framework and incorporates an effective acceleration method to enhance computational efficiency. Finally, numerical experiments using real-world industrial data validate the effectiveness of the proposed algorithm against the scheduling policy applied in engineering practice, an exact solver, and four state-of-the-art metaheuristics. A sensitivity analysis is also conducted to provide managerial insights.

Note to Practitioners—Motivated by a real-world heterogeneous AGV application in manufacturing automation, this paper addresses the practical scheduling challenge of a carrier–shuttle system. Two types of AGVs are physically attachable and exhibit complementary functions, with the carrier’s material handling relying on the shuttle and the shuttle’s transfer dependent on the carrier. This interdependence leads to synchronization constraints and deadlock issues, thereby increasing scheduling complexity. Petri nets are introduced to model such interactions, upon which deadlock detection and prevention strategies are developed to ensure scheduling feasibility. Furthermore, a Petri net-based metaheuristic is developed, enhanced with an acceleration technique to meet industrial response time requirements. Experimental results demonstrate that the exact solver is applicable only to small-scale instances. In contrast, the proposed algorithm consistently yields superior scheduling outcomes and outperforms the compared methods. Additionally, a sensitivity analysis offers managerial insights to optimize the configuration of carriers and shuttles. Although developed for the carrier–shuttle system, the proposed Petri net model and deadlock-free scheduling approach

Boyu Li, Zhengchen Li, and Weimin Wu are with the State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China (e-mail: byli@zju.edu.cn; 12432031@zju.edu.cn; wmwu@iipc.zju.edu.cn).

MengChu Zhou is with the School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China, and also with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: mengchu@gmail.com).

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

can be generalized to various cooperative multi-agent scenarios.

Index Terms—Heterogeneous AGV, synchronization, Petri net, deadlock, adaptive large neighborhood search.

I. INTRODUCTION

Rapid advancements in automation and robotics drive the widespread adoption of automated guided vehicles (AGVs) in smart logistics, manufacturing, and warehousing. By automating material transportation, AGVs have been used to enhance manufacturing system productivity, reduce labor cost, and improve production safety. With the increasing demand for full automation, heterogeneous AGVs attract significant attention from both academia and industry. Inspired by the practical industrial context depicted in Fig. 1, this work aims to study a new attachable heterogeneous AGV scheduling problem (AHASP) in a battery manufacturing system.

As shown in Fig. 1, two types of heterogeneous AGVs with complementary capabilities collaborate to transport battery plates in a manufacturing system. 1) *Shuttles* with material handling capabilities navigate confined production chambers to load and unload plates and are restricted to local movement within chambers. 2) *Carriers* with substantial load capacity and transport capabilities enable the inter-chamber transfer of loaded or unloaded shuttles. The shuttle and carrier can attach to execute tasks synchronously or detach to operate asynchronously. Each task necessitates synchronized collaboration between a carrier and shuttle, with the carrier relying on the shuttle for material handling and the shuttle depending on the carrier for transfer. Thus, a triangular interdependence exists among a task, carrier, and shuttle, as illustrated in Fig. 2. Given the straightforward layout of the manufacturing system and the sensor-based AGV collision avoidance capability, this study focuses on task assignment rather than path planning. Based on the taxonomy in [1], AHASP is classified as a multi-robot task assignment (MRTA) problem involving crossed-schedule dependencies, single-task robots, and multi-robot tasks, making it a highly challenging scheduling problem.

Carriers and shuttles, with their simple structures and distinct functions, offer cost-effectiveness, stability, and improved operational efficiency. However, the attachment-induced interdependence and synchronization introduce significant complexity to their scheduling. Four key challenges are observed: 1) *Expanded Solution Space*: The cooperative execution of individual tasks by multiple AGVs in AHASP substantially increases the solution space compared with traditional vehicle



Fig. 1. Battery manufacturing system in Jiangxi, China.

routing problems (VRPs) that are limited to single-vehicle task assignments. 2) *Non-decomposability*: The physical interdependence between a carrier and shuttle prevents the decomposition of tasks in AHASP into independently assignable sub-tasks, as commonly done in solving problems involving multi-robot tasks [2]. 3) *Complex Schedule Representation*: The solutions to AHASP involve concurrent and sequential task execution as well as carrier–shuttle synchronization, complicating the encoding, decoding, and evaluation of its solution. 4) *Coupled Precedence Relations*: The scheduling of one AGV depends on or affects the schedules of other AGVs, giving rise to cascading precedence relations that may induce deadlock and render scheduling infeasible.

To the best of our knowledge, despite its theoretical importance and practical relevance, AHASP remains completely unexplored in the literature. To fill the research gap, we aim to make the following novel contributions to the field of AHASP:

- We propose AHASP for the first time and formulate it as a mixed-integer linear programming (MILP) model to minimize the travel distance and tardiness.
- We introduce Petri nets (PNs) to model coupled AGV schedules, and thus describe the concurrent and sequential task execution as well as AGV synchronization.
- We propose a firing-driven decoding (FDD) method, a deadlock-based feasibility theorem, and a bidirectional reachability search (BRS) method to prevent deadlock and ensure deadlock-free scheduling results.
- We develop a PN-based metaheuristic in an adaptive large neighborhood search (ALNS) framework, and successfully incorporate a BRS-based acceleration method to enhance its computational effectiveness.
- In addition, we conduct extensive experiments on datasets collected from a real-world industrial system to validate the effectiveness of the proposed algorithm by comparing it with the on-site scheduling policy, an exact solver, and four state-of-the-art metaheuristics. We also perform a sensitivity analysis to derive managerial insights.

The remainder of the paper is organized as follows. Section II reviews related work. Section III formulates AHASP as a mathematical model. Section IV constructs a PN model and analyzes deadlock. Section V develops a PN-based metaheuristic. Section VI presents experimental results. Section VII concludes this paper.

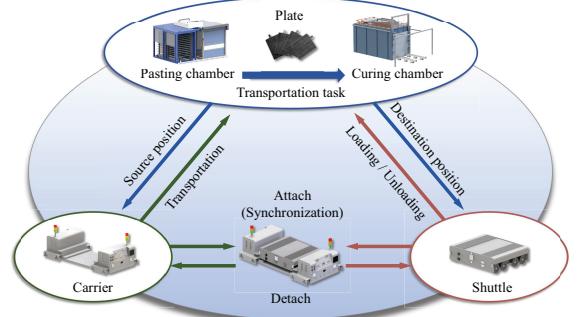


Fig. 2. Triangular interdependence among the carrier, shuttle, and task.

II. LITERATURE REVIEW

A. Heterogeneous Robot Scheduling Problem (HRSP)

Fu et al. [2] classify robot heterogeneity into two categories: structural (e.g., size, speed, and capacity) and functional (e.g., loading, transporting, and stacking) ones. A representative structural HRSP is the heterogeneous vehicle routing problem (HVRP), involving vehicles with varying costs and capacities [3], [4]. Comprehensive reviews of HVRP are available in [5]. Despite its various forms, structural heterogeneity are typically reflected through cost and capacity constraints [6], [7]. In the absence of collaboration and synchronization, HVRP tasks are scheduled independently, which resembles homogeneous MRTA. AHASP addressed in this paper involves functionally heterogeneous AGVs operating collaboratively with synchronization constraints. Functional HRSP is reviewed next.

Functionally heterogeneous robots possess distinct yet complementary capabilities and cooperate to accomplish compound tasks [1], [8], [9]. A common example is the cooperation between forklifts and transport vehicles to transport and load cargo onto warehouse storage racks [10], [11]. This cooperation synchronizes their schedules and exhibits their interdependence where the utilization of one robot directly affects or depends on the scheduling of others [12]. A widely adopted strategy for collaborative HRSP is to decompose compound tasks into elemental tasks with interrelated constraints. Each elemental task is then assigned to a robot with the required functionality [2], [13]–[16]. Specifically, Chen et al. [14] investigate the integrated scheduling of yard cranes and AGVs, decomposing it into a set of crane-specific and vehicle-specific sub-tasks. Ferreira et al. [16] examine the scheduling of heterogeneous multi-robot teams, breaking down compound tasks into fine-grained action-level components. Fu et al. [2] develop a stochastic programming framework for heterogeneous multi-agent systems that concurrently optimizes decomposition, task assignment, and scheduling.

Although AHASP involves functionally heterogeneous AGVs, it introduces a unique attachment-induced physical dependency between carriers and shuttles, which is absent in existing HRSP [1], [2]. This dependency renders all existing decomposition strategies inapplicable to AHASP, as the compound task in AHASP cannot be decomposed into carrier-specific and shuttle-specific sub-tasks for separate assignment. A similar attachment-based dependency is observed in unmanned air-ground vehicle (UAV/UGV) systems where

UAVs depend on UGVs for transportation, charging, and deployment to conduct aerial surveillance [17]. Existing research on UAV/UGV systems typically favors one side or partition cooperative scheduling into UGV and UAV-level sub-problems, solved hierarchically [18]–[20]. While two-level scheduling reduces computational complexity, it weakens UAV-UGV collaboration and sacrifices global optimality. Coordinated scheduling for a non-decomposable carrier-shuttle system remains a new and challenging problem.

B. VRP with Multiple Synchronization Constraints (VRPMS)

VRPMS is closely related to AHASP investigated in this study, as both involve multiple vehicles serving a customer synchronously. A comprehensive review of VRPMS can be found in [21], [22].

Synchronization constraints are first introduced by Everborn et al. [23] within the context of staff scheduling in healthcare systems where two caregivers are required to serve a single patient simultaneously. Bredstrø et al. [24] further extend synchronization constraints into temporal precedence and investigate synchronization issues in contexts such as forest operations and homecare staff scheduling. Dohn et al. [25] then generalize synchronization and precedence constraints as temporal dependencies. VRPMS has applications in various real-world scenarios, including the delivery of medication, appliances, and electrical equipment [26], two-echelon distribution [27], and staff scheduling [28]. To model the synchronized precedence relations in VRPMS, two common approaches are precedence graph [15], [29], [30] and precedence matrix [31], [32]. Specifically, Masson et al. [29] initially introduce the precedence graph and employ cycle detection to identify inconsistencies. Liu et al. [31] formulate dependencies using a precedence matrix and utilize it to prevent cross-synchronization. Wang et al. [32] combine the precedence graph and matrix to assess both time-related and cycle-related feasibility. However, both approaches are static, thus failing to capture the dynamic intermediate states of AGV schedules.

PNs constitute a powerful formalism for modeling discrete event systems [33]. They offer a dynamic graphical representation capable of dynamically capturing asynchronous and concurrent events, synchronization constraints, and sequential relationships, all of which are inherent in VRPMS. Moreover, PNs are extensively employed for deadlock analysis and resolution [34], which correspond to inconsistencies observed in VRPMS. Despite their strong applicability, the use of PNs in VRPMS remains unexplored. The carrier-shuttle system represents a typical event-driven discrete event system, characterized by concurrent, asynchronous, and sequential task execution, along with AGV synchronization. Accordingly, this paper introduces PNs to model and analyze the solutions to AHASP to achieve deadlock-free scheduling.

III. PROBLEM DESCRIPTION AND FORMULATION

A. Problem Description

The layout of a battery manufacturing system is shown in Fig. 3. Battery plates are pasted in pasting chambers and

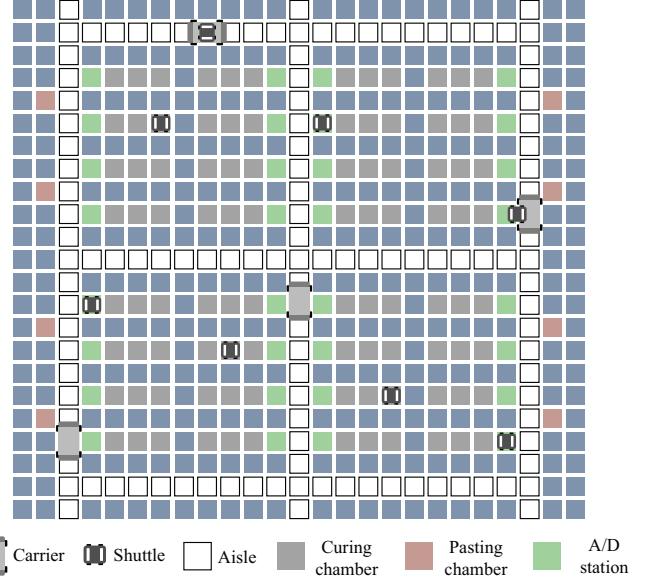


Fig. 3. Layout of the battery manufacturing system.

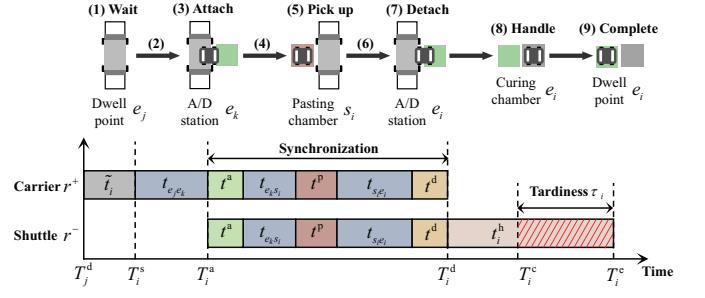


Fig. 4. Synchronized execution of task i by carrier r^+ and shuttle r^- .

then transported to curing chambers via carriers and shuttles. Each task i involves a pasting chamber as source position s_i and a curing chamber as destination position e_i , requiring the cooperation between a carrier r^+ and a shuttle r^- . Let tasks j and k be the previous tasks of carrier r^+ and shuttle r^- , respectively, before executing task i . The execution process of task i is shown in Fig. 4 and detailed as follows.

- 1) Upon completing task j at time T_j^d , carrier r^+ dwells at position e_j for a duration of \bar{t}_i ;
- 2) Carrier r^+ starts task i at time T_i^s , moving from position e_j to e_k ;
- 3) Shuttle r^- begins attaching to carrier r^+ at time T_i^a ;
- 4) Carrier r^+ transfers shuttle r^- to pasting chamber s_i ;
- 5) Shuttle r^- picks up pasted plates;
- 6) Carrier r^+ transfers shuttle r^- to curing chamber e_i ;
- 7) Shuttle r^- detaches from carrier r^+ , and carrier r^+ completes task i at time T_i^d ;
- 8) Shuttle r^- handles plates in curing chamber e_i ;
- 9) Shuttle r^- completes task i and dwells at the attach/detach (A/D) station of e_i at time T_i^e .

As shown in Fig. 4, carrier r^+ and shuttle r^- initiate and complete task i asynchronously, but operate synchronously during the attachment period. To ensure plate quality, each task i has a designated curing due time T_i^c . However, delays are

inevitable during periods of high production and task volume. Therefore, a penalty cost is incurred for tardiness, as indicated by the red slanted bar in Fig. 4. The paths of carriers can be planned by using A* algorithm, with the travel distance contributing to the overall cost. The travel time t_{s_i, e_i} between positions s_i and e_i is calculated as $t_{s_i, e_i} = d_{s_i, e_i}/v$. Given the shuttle's limited intra-chamber movement, its travel cost is considered negligible.

B. Mathematical Formulation

Parameters

\mathcal{N}	Set of tasks
\mathcal{N}_o	Set of virtual tasks
\mathcal{N}'	Set of nodes
\mathcal{V}	Set of all AGVs
\mathcal{V}^+	Set of carriers
\mathcal{V}^-	Set of shuttles
n	Number of tasks
n_r	Number of tasks assigned to AGV r
m^+	Number of carriers
m^-	Number of shuttles
o_r	Virtual task of AGV r
π_i^r	i -th task assigned to AGV r
s_i	Source position of task i
e_i	Destination position of task i
d_{s_i, e_i}	Travel distance between two positions
t_{s_i, e_i}	Travel time between two positions
v	Velocity of AGVs
t^a	Attachment duration
t^d	Detachment duration
t^p	Pick-up duration
t_i^h	Handling duration of task i
T_i^s	Start time of task i
T_i^a	Attach time of task i
T_i^c	Designated curing due time of task i
λ	Cost weight factor
L	A sufficiently large positive value

Decision variables

x_{ij}^r	$x_{ij}^r = 1$ if edge (i, j) is assigned to AGV r and 0 otherwise
\tilde{t}_i	Carrier's waiting duration before executing task i
T_i^d	Detach time of task i
T_i^e	Completion time of task i
δ_i	Travel distance of task i
τ_i	Tardiness of task i

AHASP is represented as a directed graph $\mathcal{G} = \{\mathcal{N}', \mathcal{E}\}$, where $\mathcal{N}' = \mathcal{N}_o \cup \mathcal{N}$ is the set of nodes and $\mathcal{E} = \{(i, j) \mid i, j \in \mathcal{N}', i \neq j\}$ is the set of edges. The nodes in $\mathcal{N}_o = \{o_r \mid r \in \mathcal{V}\}$ represent the virtual start and end tasks for each AGV. For AGV r , its task sequence starts and ends at task o_r , with e_{o_r} indicating its initial position. The nodes in $\mathcal{N} = \{1, 2, \dots, n\}$ correspond to transportation tasks. AHASP entails assigning each task in \mathcal{N} to a carrier in $\mathcal{V}^+ = \{1, 2, \dots, m^+\}$ and a shuttle in $\mathcal{V}^- = \{m^+ + 1, m^+ + 2, \dots, m^+ + m^-\}$, and determining the execution sequence for each AGV in $\mathcal{V} = \mathcal{V}^+ \cup \mathcal{V}^-$. The objective is to minimize the weighted sum of travel distance and tardiness. The MILP model for AHASP is formulated as follows.

$$\min F = \lambda \cdot \sum_{i \in \mathcal{N}} \delta_i + (1 - \lambda) \cdot \sum_{i \in \mathcal{N}} \tau_i \quad (1)$$

s.t.:

$$\sum_{r \in \mathcal{V}^+} \sum_{i \in \mathcal{N}'} x_{ij}^r = 1, \forall j \in \mathcal{N} \quad (2)$$

$$\sum_{r \in \mathcal{V}^-} \sum_{i \in \mathcal{N}'} x_{ij}^r = 1, \forall j \in \mathcal{N} \quad (3)$$

$$\sum_{r \in \mathcal{V}^+} \sum_{j \in \mathcal{N}'} x_{ij}^r = 1, \forall i \in \mathcal{N} \quad (4)$$

$$\sum_{r \in \mathcal{V}^-} \sum_{j \in \mathcal{N}'} x_{ij}^r = 1, \forall i \in \mathcal{N} \quad (5)$$

$$\sum_{i \in \mathcal{N}'} x_{ij}^r - \sum_{i \in \mathcal{N}'} x_{ji}^r = 0, \forall j \in \mathcal{N}, r \in \mathcal{V} \quad (6)$$

$$\sum_{i \in \mathcal{N}'} x_{io_r}^r = \sum_{i \in \mathcal{N}'} x_{o_r i}^r = 1, r \in \mathcal{V} \quad (7)$$

$$d_{e_i, e_k} + d_{e_k, s_j} + d_{s_j, e_j} - L(2 - x_{ij}^{r^-} - x_{kj}^{r^+}) \leq \delta_j, \quad \forall i, k \in \mathcal{N}', j \in \mathcal{N}, r^+ \in \mathcal{V}^+, r^- \in \mathcal{V}^- \quad (8)$$

$$T_k^e - T_i^d - t_{e_i, e_k} - L(2 - x_{ij}^{r^-} - x_{kj}^{r^+}) \leq \tilde{t}_j, \quad \forall i, k \in \mathcal{N}', j \in \mathcal{N}, r^+ \in \mathcal{V}^+, r^- \in \mathcal{V}^- \quad (9)$$

$$T_i^d + \tilde{t}_j + t_{e_i, e_k} + t^a + t_{e_k, s_j} + t^p + t_{s_j, e_j} + t^d - L(2 - x_{ij}^{r^-} - x_{kj}^{r^+}) \leq T_j^d, \forall i, k \in \mathcal{N}', j \in \mathcal{N}, r^+ \in \mathcal{V}^+, r^- \in \mathcal{V}^- \quad (10)$$

$$T_i^d + t_i^h \leq T_i^e, \forall i \in \mathcal{N} \quad (11)$$

$$T_i^e - T_i^c \leq \tau_i, \forall i \in \mathcal{N} \quad (12)$$

$$x_{ij}^r = 0, \forall i, j \in \mathcal{N}, r \in \mathcal{V}, i = j \quad (13)$$

$$\tilde{t}_i, \tau_i, T_i^d, T_i^e \geq 0, \forall i \in \mathcal{N}' \quad (14)$$

$$x_{ij}^r \in \{0, 1\}, \forall i, j \in \mathcal{N}', r \in \mathcal{V} \quad (15)$$

The objective function is defined in (1). Constraints (2)-(5) guarantee the assignment of each task to one carrier and one shuttle. Constraint (6) enforces flow conservation. Constraint (7) ensures that the task sequence of AGV r starts and ends at virtual task o_r . Constraint (8) determines the travel distance of each task. Constraint (9) defines the carrier's waiting duration prior to attachment. Constraint (10) defines the relationship between the detachment times of two consecutive tasks, expressing precedence relations in terms of temporal constraints. Constraint (11) establishes the relationship between the detachment time and completion time of each task. Constraint (12) determines the tardiness of each task. Constraint (13) eliminates self-visits. Constraints (14) and (15) define the feasible domains of decision variables.

IV. PETRI NET MODELING AND DEADLOCK ANALYSIS

A. Solution Representation

This work adopts PNs to model the solution to AHASP. For ease of understanding and analysis, a permutation-based representation of the solution is introduced in this section, which serves as an intermediate abstraction to facilitate PN modeling. Note that all metaheuristic operations are performed directly on a PN model, not on its permutation.

The permutation representation of carrier and shuttle schedules is a dual chain $\Pi = (\Pi^+, \Pi^-)$, where Π^+ and Π^- represent the carrier and shuttle chains, respectively. Specifically, $\Pi^+ = (0, \pi_1, 0, \pi_2, 0, \dots, 0, \pi_{m^+})$, and $\Pi^- =$

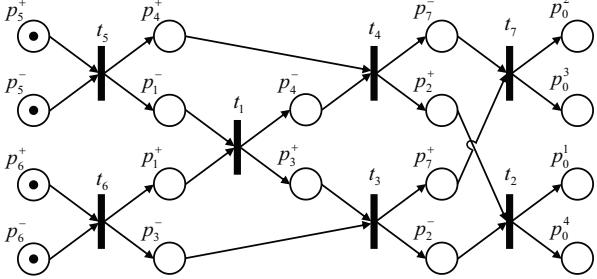


Fig. 5. PN $\Sigma(\Pi)$ modeling solution Π , where $\Pi^+ = (0, 5, 4, 2, 0, 6, 1, 3, 7)$ and $\Pi^- = (0, 5, 1, 4, 7, 0, 6, 3, 2)$.

$(0, \pi_{m^++1}, 0, \pi_{m^++2}, 0, \dots, 0, \pi_{m^++m^-})$, where 0 separates adjacent sub-vectors. The task sequence of AGV $r \in \mathcal{V}$ is denoted as $\pi_r = (\pi_1^r, \pi_2^r, \dots, \pi_{n_r}^r)$, specifying the assigned tasks and their execution order. Any task $i \in \mathcal{N}$ exists in both Π^+ and Π^- simultaneously.

For instance, consider $\Pi^+ = (0, 5, 4, 2, 0, 6, 1, 3, 7)$ and $\Pi^- = (0, 5, 1, 4, 7, 0, 6, 3, 2)$, where seven tasks $\mathcal{N} = \{1, 2, 3, 4, 5, 6, 7\}$ are assigned to two carriers $\mathcal{V}^+ = \{1, 2\}$ and two shuttles $\mathcal{V}^- = \{3, 4\}$. The sequence $\pi_1 = (5, 4, 2)$ in Π^+ specifies that carrier 1 executes tasks 5, 4, and 2 in sequence. The sequence $\pi_4 = (6, 3, 2)$ in Π^- assigns shuttle 4 to sequentially execute tasks 6, 3, and 2. In this example, we have precedence relation $5 \prec 4$ defined in π_1 . Note that $5 \prec 3$ also holds, propagating through the intermediary task 1 in both π_2 and π_3 .

B. Petri Net Modeling

A PN is defined as a five-tuple $\Sigma = \langle \mathcal{P}, \mathcal{T}, \mathcal{A}, W, M_0 \rangle$, where \mathcal{P} is the set of places and \mathcal{T} is the set of transitions. $\mathcal{A} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$ is the set of directed arcs connecting places and transitions. The mapping $W : \mathcal{A} \rightarrow \mathbb{N} \setminus \{0\}$ assigns weights to arcs, where \mathbb{N} is the set of non-negative integers. The marking of Σ is a mapping $M : \mathcal{P} \rightarrow \mathbb{N}$ that assigns tokens to places. M_0 denotes the initial marking. Let $x \in \mathcal{P} \cup \mathcal{T}$ be a node in Σ , with $\bullet x = \{y | (y, x) \in \mathcal{A}\}$ and $x^\bullet = \{y | (x, y) \in \mathcal{A}\}$ denoting the preset and postset of x , respectively. A transition $t \in \mathcal{T}$ is enabled at marking M if $\forall p \in \bullet t, M(p) \geq W(p, t)$, denoted as $M[t]$. Firing t results in a new marking M' such that $\forall p \in \mathcal{P}, M'(p) = M(p) - W(p, t) + W(t, p)$, denoted as $M[t]M'$. For a firing sequence $\sigma = (t_1, t_2, \dots, t_n)$, $M[\sigma]M'$ indicates that the transitions in σ fire sequentially, transforming M into M' . $\mathcal{R}(M_0)$ represents the set of markings reachable from M_0 . A detailed discussion on PN theory can be found in [33], [34]. The PN model for solution Π is defined in Definition 1.

Definition 1. A PN modeling solution Π is defined as $\Sigma(\Pi) = \langle \mathcal{P}, \mathcal{T}, \mathcal{A}, W, M_0 \rangle$, where:

- $\mathcal{P} = \mathcal{P}^+ \cup \mathcal{P}^- \cup \mathcal{P}_0^+ \cup \mathcal{P}_0^-$ is the set of places, with $\mathcal{P}^+ = \{p_i^+ \mid i \in \mathcal{N}\}$, $\mathcal{P}^- = \{p_i^- \mid i \in \mathcal{N}\}$, $\mathcal{P}_0^+ = \{p_0^r \mid r \in \mathcal{V}^+\}$, and $\mathcal{P}_0^- = \{p_0^r \mid r \in \mathcal{V}^-\}$.
- $\mathcal{T} = \{t_i \mid i \in \mathcal{N}\}$ is the set of transitions.
- $\mathcal{A} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$ is the set of arcs.
- W is the arc weight mapping such that $W(a) = 1$ if $a \in \mathcal{A}$; and 0 otherwise.

- M_0 is the initial marking, where $\forall i \in \{\pi_1^r \mid r \in \mathcal{V}^+\}$, $M_0(p_i^+) = 1$, $\forall i \in \{\pi_1^r \mid r \in \mathcal{V}^-\}$, $M_0(p_i^-) = 1$, and $M_0(p) = 0$ for all other $p \in \mathcal{P}$.

Lemma 1. In PN $\Sigma(\Pi)$, $\forall t \in \mathcal{T}$ can be fired at most once.

Fig. 5 presents the PN $\Sigma(\Pi)$ that models the example solution Π given in Section IV-A. In $\Sigma(\Pi)$, transitions denote tasks, places represent task states, tokens represent AGVs, and arcs indicate precedence relations. The firing of t_i corresponds to the execution of task i . According to Lemma 1, transitions are not re-fired, ensuring that each task is executed exactly once. The proofs of all proposed lemmas and theorems are provided in the supplementary file. For any $t_i \in \mathcal{T}$, $|\bullet t_i| = 2$ and $|t_i^\bullet| = 2$. Let p_i^+ and p_i^- represent the two input places of t_i . A token in place p_i^+ (resp. p_i^-) indicates that the assigned carrier (resp. shuttle) is ready to execute task i . Thus, the enabling of t_i signifies that the carrier and shuttle assigned to task i are synchronized and ready. In place p_0^r , a token indicates that AGV r completes all assigned tasks. Arcs $(t_i, p), (p, t_j) \in \mathcal{A}$ indicate that task i is the direct predecessor of task j , i.e., $\exists r \in \mathcal{V}, x_{ij}^r = 1$. Thus, if transition t_j is reachable from t_i , it follows that $i \prec j$.

C. Firing-driven Decoding

By constructing PN $\Sigma(\Pi)$, the decoding process can be formalized as transition firing. We first introduce the final marking M_F .

Definition 2. The final marking of PN $\Sigma(\Pi)$, denoted by M_F , is defined such that $\forall p_0^r \in \mathcal{P}_0^+ \cup \mathcal{P}_0^-$, $M_F(p_0^r) = 1$, $\forall p \in \mathcal{P}^+ \cup \mathcal{P}^-$, $M_F(p) = 0$.

Theorem 1. A necessary and sufficient condition for PN $\Sigma(\Pi)$ to reach M_F is that $\forall t \in \mathcal{T}$ is fired exactly once.

Theorem 1 establishes that the firing of each transition once, corresponding to the completion of all tasks, results in PN $\Sigma(\Pi)$ with M_F . Thus, based on Theorem 1, an FDD method is proposed in Algorithm 1 to assess solution Π . FDD begins with the initial marking M_0 and sequentially fires the enabled transitions until M_F is reached. The structural properties of PN $\Sigma(\Pi)$ ensure that the firing sequence adheres to the precedence relations defined by Π , and the firing condition enforces synchronization between a carrier and shuttle. The computational complexity of FDD is $O(|\mathcal{T}| + |\mathcal{P}|)$, as each transition and place is visited only once.

The FDD process for the solution Π in Fig. 5 is represented by the firing sequence $\sigma = (t_5, t_6, t_1, t_4, t_3, t_7, t_2)$, which is detailed in the supplementary file. It is noted that the firing sequence from M_0 to M_F is not unique. For example, at marking M_0 , transitions t_5 and t_6 are concurrently enabled. However, for concurrent transitions, the firing order does not affect the decoding outcome, as $\Sigma(\Pi)$ is a conflict-free PN, i.e., $\forall p \in \mathcal{P}, |p^\bullet| \leq 1$.

D. Deadlock-based Feasibility Evaluation

The carrier-shuttle interdependence and synchronization constraints may introduce circular wait in the cascading precedence relations, rendering solution Π infeasible. That is, there

Algorithm 1: Firing-driven decoding

input : Petri net $\Sigma(\Pi)$, final marking M_F ;
output: Objective value $F(\Pi)$;

- 1 Initialize current marking $M_C \leftarrow M_0$ and enabled transition set $\mathcal{T}_E \leftarrow \emptyset$;
- 2 Initialize objective value $F(\Pi) \leftarrow 0$;
- 3 $\mathcal{T}_E \leftarrow \{t_i | i = \pi_1^r, r \in \mathcal{V}^+\} \cap \{t_j | j = \pi_1^r, r \in \mathcal{V}^-\}$;
- 4 **while** $M_C \neq M_F$ **do**
- 5 **if** $\mathcal{T}_E = \emptyset$ **then**
- 6 $F(\Pi) \leftarrow \infty$;
- 7 **return** $F(\Pi)$. $\triangleright \Sigma(\Pi)$ encounters deadlock at M_C .
- 8 Randomly select and remove t_k from set \mathcal{T}_E ;
- 9 Calculate δ_k and τ_k using Eqs. (8)-(12);
- 10 $F(\Pi) \leftarrow F(\Pi) + \lambda \cdot \delta_k + (1 - \lambda) \cdot \tau_k$;
- 11 $M_C[t_k] M'_C$ and update $M_C \leftarrow M'_C$;
- 12 **for** $t \in \bigcup_{p \in t_k^\bullet} p^\bullet$ **do**
- 13 **if** $M_C[t]$ **then**
- 14 Append t into set \mathcal{T}_E ;
- 15 **return** $F(\Pi)$.

exist tasks $i, j \in \mathcal{N}$ such that both $i \prec j$ and $j \prec i$ hold, violating the acyclicity of precedence. As indicated in Line 7 of Algorithm 1, such infeasibility manifests as a deadlock in $\Sigma(\Pi)$, where the PN is blocked before reaching M_F . The deadlock in PN is defined next.

Definition 3. Deadlock is defined as a marking $M \in \mathcal{R}(M_0)$ such that $\nexists t \in \mathcal{T}, M[t]$.

Strictly speaking, the final marking M_F is a deadlock marking, at which no transition remains enabled. However, M_F is a specially designed representation of the desired final state, in which all tasks have been successfully completed. The deadlocks discussed in the following refer specifically to those caused by circular wait, where no transition is enabled while some tasks remain incomplete. A simple example of such deadlock is given in Fig. 6, where $t_i \in \bullet p_j^+$ and $t_j \in \bullet p_i^-$. The firing condition for t_i is the firing of t_j , enabling p_i^- to receive a token. Conversely, t_j is enabled by the firing of t_i , allowing p_j^+ to receive a token. Therefore, a circular wait arises between tasks i and j , where neither transition can be fired, as indicated by the red loop in Fig. 6. This leads to a deadlock that prevents $\Sigma(\Pi)$ from reaching M_F .

Theorem 2. A necessary and sufficient condition for Π to be a feasible solution is that PN $\Sigma(\Pi)$ is deadlock-free at $\forall M \in \mathcal{R}(M_0) \setminus \{M_F\}$.

To assess the feasibility of solution Π , we establish Theorem 2 based on deadlock analysis in $\Sigma(\Pi)$. By combining Theorems 1 and 2, the absence of deadlock at $\forall M \in \mathcal{R}(M_0) \setminus \{M_F\}$ implies the existence of a firing sequence σ that contains each transition in \mathcal{T} once, such that $M_0[\sigma] M_F$. Thus, the feasibility of Π can be determined by constructing the reachability graph [34] of PN $\Sigma(\Pi)$, which inherently follows the same process as FDD. Hence, the PN model unifies the decoding and feasibility verification of solution Π into a single process, i.e., the procedure in Algorithm 1.

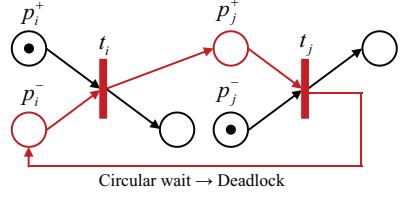


Fig. 6. An example of deadlock in PN $\Sigma(\Pi)$ caused by circular wait.

E. Deadlock Prevention

Based on Theorem 2, deadlock prevention in the PN model is equivalent to ensuring scheduling feasibility. In ALNS, infeasible solutions arise from the improper reinsertion of removed tasks. Since circular wait is a necessary condition for deadlock, a BRS method is developed to avoid circular wait during task insertion to achieve deadlock-free scheduling.

The insertion of a removed task, denoted as task k , occurs in both Π^+ and Π^- , corresponding to the addition of a transition t_k , two places p_k^+ and p_k^- , and arcs consistent with the precedence relations in the PN model. Removal is the reverse process. It is assumed that PN $\Sigma(\Pi)$ satisfies the condition stated in Theorem 2 prior to the insertion, i.e., solution Π is feasible. Clearly, inserting task k into either Π^+ or Π^- individually does not result in circular wait. Below, we discuss the detection of infeasible positions in Π^- after the insertion of task k into Π^+ based on $\Sigma(\Pi)$. The reverse process follows similarly due to the duality between Π^+ and Π^- . Let $\overleftarrow{\epsilon}^+(i)$ and $\overrightarrow{\epsilon}^+(i)$ denote the preceding and succeeding positions of task i in Π^+ , and $\overleftarrow{\epsilon}^-(i)$ and $\overrightarrow{\epsilon}^-(i)$ denote the corresponding positions in Π^- . The pseudo-code of BRS is given in Algorithm 2, where task k is inserted at position $\overrightarrow{\epsilon}^+(i) / \overleftarrow{\epsilon}^+(j)$ and introduces t_k and p_k^+ . Then, BRS identifies infeasible positions in Π^- through bidirectional searches in $\Sigma(\Pi)$. The backward search explores all transitions that can reach t_k , while the forward search identifies the transitions that are reachable from t_k . In other words, $\forall \overleftarrow{\epsilon}^-(b) \in \mathcal{L}_B, b \prec k$, and $\forall \overrightarrow{\epsilon}^-(f) \in \mathcal{L}_F, k \prec f$. The computational complexity of BRS is $O(|\mathcal{T}| + |\mathcal{P}|)$, as each transition and place is searched at most once. The soundness and completeness of BRS are proved in the supplementary file.

An illustrative example of BRS is presented in Fig. 7, where task 8 is inserted at position $\overrightarrow{\epsilon}^+(1) / \overleftarrow{\epsilon}^+(3)$ in the solution Π shown in Fig. 5. The corresponding modification in PN $\Sigma(\Pi)$ is highlighted in the red box. The backward and forward searches are depicted using arrows of two different colors in $\Sigma(\Pi)$. Note that the direction of the backward search is opposite to that of the blue arrows. The backward search identifies three infeasible positions: $\mathcal{L}_B = \{\overleftarrow{\epsilon}^-(1), \overleftarrow{\epsilon}^-(5), \overleftarrow{\epsilon}^-(6)\}$. The forward search detects three other invalid positions: $\mathcal{L}_F = \{\overrightarrow{\epsilon}^-(2), \overrightarrow{\epsilon}^-(3), \overrightarrow{\epsilon}^-(7)\}$. Only three feasible insertion positions exist: $\overleftarrow{\epsilon}^-(4)$, $\overleftarrow{\epsilon}^-(7)$, and $\overrightarrow{\epsilon}^-(6)$, reflecting an appreciable proportion of infeasible positions.

V. PETRI NET-BASED METAHEURISTIC

Building on the PN model introduced in Section IV, we develop a PN-based metaheuristic in the ALNS framework to solve AHASP. Recall that all ALNS operations are performed

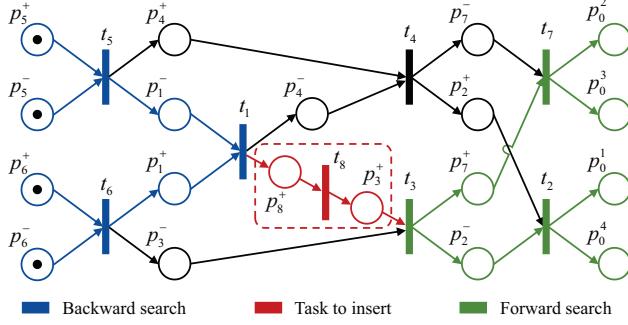


Fig. 7. BRS in PN $\Sigma(\Pi)$ with task 8 inserted at $\bar{\epsilon}^+(1) / \bar{\epsilon}^+(3)$.

Algorithm 2: Bidirectional reachability search

```

input : Solution  $\Pi$ , task  $k$ 's insertion position  $\bar{\epsilon}^+(i) / \bar{\epsilon}^+(j)$  in  $\Pi^+$ ;
output: Infeasible insertion positions for task  $k$  in  $\Pi^-$ ;
1 ▷ Backward search
2 Initialize the backward search transition set  $\mathcal{T}_B \leftarrow \emptyset$ , and
   infeasible position set  $\mathcal{L}_B \leftarrow \emptyset$ ;
3 if  $i \neq 0$  then
4   Append transition  $t_i$  into set  $\mathcal{T}_B$ ;
5 while  $\mathcal{T}_B \neq \emptyset$  do
6   Randomly select and remove  $t_b$  from set  $\mathcal{T}_B$ ;
7   Append position  $\bar{\epsilon}^-(b)$  into set  $\mathcal{L}_B$ ;
8    $\mathcal{T}_B \leftarrow [(\cup_{p \in \bullet_{t_b}} p) \setminus \{t_d | \bar{\epsilon}^-(d) \in \mathcal{L}_B\}] \cup \mathcal{T}_B$ ;
9 ▷ Forward search
10 Initialize the forward search transition set  $\mathcal{T}_F \leftarrow \emptyset$ , and
    infeasible position set  $\mathcal{L}_F \leftarrow \emptyset$ ;
11 if  $j \neq 0$  then
12   Append transition  $t_j$  into set  $\mathcal{T}_F$ ;
13 while  $\mathcal{T}_F \neq \emptyset$  do
14   Randomly select and remove  $t_f$  from set  $\mathcal{T}_F$ ;
15   Append position  $\bar{\epsilon}^-(f)$  into set  $\mathcal{L}_F$ ;
16    $\mathcal{T}_F \leftarrow [(\cup_{p \in t_f^\bullet} p) \setminus \{t_d | \bar{\epsilon}^-(d) \in \mathcal{L}_F\}] \cup \mathcal{T}_F$ ;
17 return  $\mathcal{L}_B \cup \mathcal{L}_F$ .

```

directly on $\Sigma(\Pi)$, which explicitly captures the coupled precedence relations and facilitates decoding and deadlock resolution. The permutation-based solution Π is adopted as an intermediate representation for ease of understanding. The designed operators, acceleration method, and metaheuristic framework are introduced next.

A. Neighborhood Operators

Inspired by [35]–[37], we develop five removal operators and three insertion operators, collectively denoted as \mathcal{O}^R and \mathcal{O}^I , respectively. In each iteration of ALNS, a removal operator is selected from \mathcal{O}^R to remove $\phi \cdot n$ tasks from the current solution Π_C , placing them into a removed task set \mathcal{N}^R . The corresponding transitions, places, and arcs are removed from PN $\Sigma(\Pi_C)$. Parameter $\phi \in [0, 1]$ denotes the proportion of tasks to be removed relative to the total number of tasks. The resulting partially destroyed solution is denoted as Π_D . An insertion operator is then selected from \mathcal{O}^I to reinsert the tasks from \mathcal{N}^R into Π_D , yielding a new solution Π_N . The PN model is updated to $\Sigma(\Pi_N)$ by adding the corresponding elements, as illustrated in Section IV-E.

The operators are selected by using a roulette wheel principle based on their historical performance. Each operator $o \in \mathcal{O}^R \cup \mathcal{O}^I$ is assigned a weight ω_o , initialized to 1. The selection probability for a removal operator o^- is calculated as $\omega_{o^-} / \sum_{o \in \mathcal{O}^R} \omega_o$, with the same principle applied to insertion operators. After κ iterations, the weight of operator o is updated to $\omega_o = (1 - \rho) \cdot \omega_o + \rho \cdot \xi_o / \theta_o$, where $\rho \in [0, 1]$ denotes the reaction factor, and ξ_o and θ_o represent the accumulated score and the usage count of operator o , respectively. Details of the scoring rules are provided in [36].

1) Removal Operators:

- Random removal (RR): $\phi \cdot n$ tasks are randomly selected and removed from Π_C .
- Highest cost removal (HCR): The cost of each task $i \in \mathcal{N}$ is defined as $F_i = \lambda \cdot \delta_i + (1 - \lambda) \cdot \tau_i$. The $\phi \cdot n$ tasks with the highest costs are then removed.
- Longest distance removal (LDR): The travel distance δ_i of task $i \in \mathcal{N}$ is given by (8). The $\phi \cdot n$ tasks with the longest travel distances are removed.
- Largest tardiness removal (LTR): The tardiness τ_i of task i is given by (12). The $\phi \cdot n$ tasks with the greatest tardiness are removed.
- Shaw removal (SR) [35]: The difference between tasks i and j is evaluated as $\Omega(i, j) = \psi \cdot (d_{s_i, s_j} + d_{e_i, e_j}) + (1 - \psi) \cdot |T_i^c - T_j^c|$. SR randomly selects a reference task and removes it, followed by removing the $(\phi \cdot n - 1)$ tasks with the smallest difference from it.

2) Insertion Operators:

- Greedy insertion (GI): A task $i \in \mathcal{N}^R$ is randomly selected, and the increase in the objective value resulting from inserting task i into each feasible insertion position is evaluated. Task i is then inserted into the position that minimizes this increase, i.e., the optimal position.
- Urgency-prioritized greedy insertion (UGI): The most urgent task, defined as $i = \arg \min_{j \in \mathcal{N}^R} (T_j^c)$, is prioritized and inserted into the optimal position.
- Cost-prioritized greedy insertion (CGI): The task with the maximum cost, defined as $i = \arg \max_{j \in \mathcal{N}^R} (F_j)$, is prioritized and inserted into the optimal position.

B. BRS-based Acceleration

As a response-time-sensitive industrial problem, AHASP necessitates the optimization of computational efficiency. In ALNS, the evaluation of each insertion position, driven by the GI operation, dominates computation time. As noted in Section IV-E, the interdependence among AGVs renders a large number of positions infeasible. Evaluating such positions introduces redundant computation and yields no valid improvement. Accordingly, a BRS-based acceleration method is proposed to enhance computational efficiency. The pseudo-code of BRS-accelerated GI operation is provided in Algorithm 3. By preemptively detecting infeasible positions, BRS ensures that FDD is employed exclusively to evaluate feasible neighborhoods, thereby ensuring deadlock-free schedules. BRS's acceleration effect is analyzed theoretically as follows.

Given a destroyed solution Π_D with n tasks, m^+ carriers, and m^- shuttles, the total number of pairwise position combinations in Π^+ and Π^- is $(n + m^+) \cdot (n + m^-)$. Without BRS,

the GI operation assesses each position combination, with its computational complexity as:

$$C_O = O^\dagger \cdot (n + m^+) \cdot (n + m^-) \quad (16)$$

where $O^\dagger = O(|\mathcal{T}| + |\mathcal{P}|)$ that is the complexity of both FDD and BRS.

With the introduction of BRS, the computational complexity of the insertion procedure in Algorithm 3 is given as:

$$C_A = O^\dagger \cdot \left[\sum_{\epsilon \in \mathcal{L}^+} (n + m^- - |\mathcal{L}_I^-(\epsilon)|) + n + m^+ \right] \quad (17)$$

where \mathcal{L}^+ is the set of all positions in Π^+ with $|\mathcal{L}^+| = n + m^+$, and $\mathcal{L}_I^-(\epsilon)$ is the set of infeasible positions in Π^- identified by BRS after inserting the removed task at position ϵ .

The ratio of C_A to C_O is given as:

$$R_C = \frac{C_A}{C_O} = 1 - \frac{\sum_{\epsilon \in \mathcal{L}^+} |\mathcal{L}_I^-(\epsilon)| - (n + m^+)}{(n + m^+) \cdot (n + m^-)} \quad (18)$$

which indicates that the acceleration effect of BRS is positively correlated with the ratio of infeasible positions to the total positions. If $\forall \epsilon \in \mathcal{L}^+, |\mathcal{L}_I^-(\epsilon)| = 1$, then $R_C = 1$. Preliminary experiments demonstrate that the number of infeasible positions $|\mathcal{L}_I^-(\epsilon)|$ is typically much greater than one, leading to $C_O > C_A$. Thus, the introduction of BRS enhances computational efficiency.

Algorithm 3: BRS-accelerated greedy insertion

```

input : Removed task set  $\mathcal{N}^R$ , destroyed solution  $\Pi_D$ ,
         insertion operator  $o^+ \in \mathcal{O}^I$ ;
output: New solution  $\Pi_N$ ;
1 while  $\mathcal{N}^R \neq \emptyset$  do
2    $\Pi^\dagger \leftarrow \Pi_D$ ,  $F^\dagger \leftarrow$  large positive number;
3   Select and remove task  $i \in \mathcal{N}^R$  based on  $o^+$ ;
4   Obtain the set of all insertion positions  $\mathcal{L}^+$  in  $\Pi_D^+$  and
      the set of all insertion positions  $\mathcal{L}^-$  in  $\Pi_D^-$ ;
5   for  $\epsilon^+ \in \mathcal{L}^+$  do
6      $\mathcal{L}_I^-(\epsilon^+) \leftarrow \text{BRS}(\Pi_D, \epsilon^+)$ ;
7     for  $\epsilon^- \in \mathcal{L}^- \setminus \mathcal{L}_I^-(\epsilon^+)$  do
8        $\Pi' \leftarrow$  insert task  $i$  at  $\epsilon^+$  and  $\epsilon^-$  in  $\Pi_D$ ;
9        $F' \leftarrow \text{FDD}(\Pi')$ ;
10      if  $F' < F^\dagger$  then
11         $\Pi^\dagger \leftarrow \Pi'$ ,  $F^\dagger \leftarrow F'$ ;
12    $\Pi_D \leftarrow \Pi^\dagger$ ;
13  $\Pi_N \leftarrow \Pi_D$ ;
14 return  $\Pi_N$ .

```

C. Framework of ALNS

ALNS first constructs an initial solution by greedily inserting all tasks. In each iteration, a pair of removal and insertion operators is selected to generate a new solution Π_N . Subsequently, Π_N is evaluated by using FDD. Then, a simulated annealing-like acceptance criterion [38], [39] determines whether to accept Π_N with a fixed temperature $T_F = \mu \cdot \sum_{i \in \mathcal{N}} d_{s_i, e_i} / [n \cdot (m^+ + m^-)]$ [40]. Here, μ is the temperature coefficient requiring calibration. Following this, the weights of the selected operators are updated. Finally, when

the terminal condition is met, the best solution is returned. For a detailed description of the ALNS framework, refer to [37]. Considering the engineering background of AHASP, the terminal condition of ALNS is defined as the CPU runtime reaching ΔT . The computation duration ΔT is defined as $\zeta \cdot n^2 \cdot (m^+ + m^-)$, where ζ is a tunable parameter reflecting practical responsiveness requirements.

VI. NUMERICAL EXPERIMENT

A. Experimental Setting

Since AHASP is a new problem derived from real-world manufacturing systems, no publicly available benchmark instances exist. We collect a comprehensive dataset from the advanced battery manufacturing company shown in Fig. 1. To safeguard data privacy, the dataset is aggregated and perturbed while preserving practical relevance. The dataset is categorized into five instance scales based on task count: $n = 5, 10, 20, 30$, and 40 . Each group contains 10 instances, with a total of 50. Instances are labeled in the format $T10_I1$, denoting the first instance with 10 tasks. This manufacturing system is equipped with four carriers and eight shuttles ($m^+ = 4$ and $m^- = 8$). The AGVs operate at a velocity of $v = 1.2\text{m/s}$. The attachment, detachment, and pick-up durations are set to $t^a = 8\text{s}$, $t^d = 8\text{s}$, and $t^p = 30\text{s}$, respectively. To balance the distance and tardiness, λ is set to 0.4. Parameter ζ is set to 10 according to practical response requirements. Instances and the optimal solutions obtained in this paper are available for download at <https://github.com/Kemingjun/AHASP>.

Algorithm performance is assessed using the relative percentage deviation (RPD), i.e.,

$$R_{PD} = \frac{F - F^\dagger}{F^\dagger} \times 100\% \quad (19)$$

where F denotes the minimum objective value obtained by a specific algorithm for a given instance, and F^\dagger denotes the best (minimal) objective value achieved by all algorithms for that instance.

All algorithms are programmed in Python and run on a computer with AMD Ryzen 9 7945HX 2.5GHz CPU, 32GB of RAM, and 64-bit Windows 11.

B. Parameter Tuning

The proposed metaheuristic involves five parameters: temperature coefficient μ , removal proportion ϕ , reaction factor ρ , update interval κ , and Shaw removal weight ψ . To identify the optimal parameter configuration, we apply the Taguchi method for the design of experiments [41] by using instance T30_1. Each of the five parameters is tested at four levels, forming an orthogonal array $L_{16}(4^5)$. Each parameter configuration is independently run for 20 times, and the main effects of the five parameters are demonstrated in Fig. 8. As the Smaller-the-Better criterion is adopted, smaller AOVs in Fig. 8 imply better parameter configurations. Thus, the parameters are configured as $\mu = 20$, $\phi = 0.3$, $\rho = 0.1$, $\kappa = 15$, and $\psi = 0.3$.

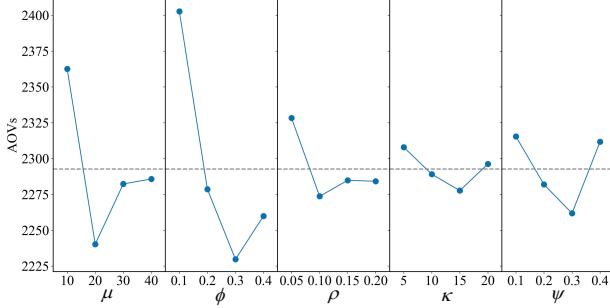


Fig. 8. Main effects plot for the five key parameters of ALNS.

C. Evaluation of BRS-based Acceleration Method

The acceleration performance of BRS reflects the computational advantage of the proposed PN-based deadlock prevention strategy. To evaluate this, we record iteration counts (χ^A and χ^O) and RPD values (R_{PD}^A and R_{PD}^O) achieved by the PN-based metaheuristic with and without BRS. As noted in Section V-B, the effectiveness of BRS positively correlates with the proportion of infeasible positions, which, as observed in preliminary experiments, depends on both the number of tasks and AGVs. Thus, we present the acceleration ratio (A_R), the gap between RPDs (G_{AP}), and the infeasible position proportion (I_{PP}) under varying task and AGV counts, i.e.,

$$A_R = \frac{\chi^A - \chi^O}{\chi^O} \times 100\% \quad (20)$$

$$G_{AP} = R_{PD}^O - R_{PD}^A \quad (21)$$

$$I_{PP} = \frac{\sum_{\epsilon \in \mathcal{L}^+} |\mathcal{L}_I^-(\epsilon)|}{[(n + m^+) \cdot (n + m^-)]} \times 100\% \quad (22)$$

CPU runtimes for FDD and BRS are also recorded. Each instance is run for 20 times, and the average metric values are reported in Table I. In this notation, C2_S4 denotes two carriers and four shuttles, with all other symbols following the same convention.

As shown in Table I, the runtimes of both FDD and BRS scales linearly with the number of tasks, consistent with their theoretical complexity. In line with (18), a positive correlation is observed between A_R and I_{PP} . As the number of tasks increases, the acceleration effect of BRS strengthens. For small-scale instances, the limited solution space and low proportion of infeasible positions result in relatively low A_R and G_{AP} values. In contrast, for large-scale instances such as C2_S4 with $n = 30$ and 40 , BRS effectively doubles the iteration count, with A_R exceeding 100%. Fewer AGVs intensify cascading effects in precedence relations, causing an increase in I_{PP} . Thus, the effectiveness of BRS is negatively correlated with the number of AGVs. Nonetheless, for C5_S10 with $n = 40$, A_R exceeds 40%, indicating a significant increase in iterations despite the higher number of AGVs. BRS yields limited improvement in objective values with additional iterations, as reflected by the relatively low G_{AP} . This is expected, as further optimization progressively becomes more challenging during the later stages of ALNS. In summary, the PN-based deadlock-free scheduling implemented by BRS significantly improves computational efficiency.

D. Comparison Experiments for PN-based Metaheuristic

Given the novelty of AHASP as an engineering-driven problem without prior research, we compare the proposed PN-based ALNS with four state-of-the-art metaheuristics for related AGV scheduling problems: variable neighborhood search (VNS) [42], iterated greedy (IG) algorithm [43], genetic algorithm (GA) [44], and discrete artificial bee colony (DABC) algorithm [45]. VNS in 2024 and IG in 2023 share a similar framework with ALNS as individual-based algorithms, differing mainly in neighborhood strategies. GA in 2023 and DABC in 2020 are two representative population-based algorithms. Their parameters are also tuned as in Section VI-B, with the results given in the supplementary file. Additionally, as a commercially available exact solver, Gurobi (version 11.0) is employed to solve and validate the MILP model. Moreover, the Earliest Due Date Bidding-based (EDDBID) scheduling policy commonly used in practice is included for comparison, as detailed in [6]. Each instance is run independently for 20 times. The average (Avg) and standard deviation (Std) of the obtained RPDs for instances with 5 and 10 tasks are presented in Tables II and III, with the best Avg highlighted in bold. Recall that the solving time of all metaheuristics is given by $\Delta T = \zeta \cdot n^2 \cdot (m^+ + m^-)$. Gurobi records the results obtained under both ΔT and a relaxed time of 60s (5 tasks) or 3600s (10 tasks), which are much larger than ΔT .

As shown in Table II, EDDBID performs well only on a few instances with 5 tasks, such as T5_I5 and T5_I9, where RPDs are below 3%. Despite its short computation time (averaging 90μs), EDDBID performs poorly and is unstable on most small-scale instances. For the MILP model, Gurobi fails to obtain optimal solutions on nine out of the ten instances within the 3-second time limit imposed for metaheuristics. Compared to the practical scheduling policy and the exact method, heuristic algorithms demonstrate a distinct advantage, with mean RPDs below 2% and mean Std below 1% across all metaheuristics. By contrast, our method outperforms its four peers, consistently obtaining the best solution in every run across all 5-task instances.

According to Table III, the performance of EDDBID deteriorates significantly when $n = 10$, with a mean RPD exceeding 60%. Gurobi demonstrates exceptionally poor performance within 12s (i.e., ΔT), failing even to find a feasible solution for instance T10_I6. Within the relaxed solving time of 3600s, Gurobi achieves the optimal solution only for instance T10_I7, with a mean RPD of 12.62%. Thus, Gurobi's excessive solving time makes it impractical to meet the real-time requirements of industrial applications. Regarding heuristic algorithms, GA and VNS exhibit a notable increase in RPD, while the mean RPDs of IG and DABC surpass 3%. In contrast, the advantage of our PN-based ALNS over other metaheuristics becomes more pronounced for instances with 10 tasks, with the lowest mean RPD of 1.34% and the lowest mean Std of 1.11%.

For instances with 20 or more tasks, EDDBID yields significantly inferior results (mean RPD > 100%). Gurobi fails to solve these large-scale instances due to memory limitations caused by excessively large models. Detailed experimental results for all instances solved by EDDBID and metaheuristics

TABLE I
AVERAGE IMPROVEMENT RATIOS IN ITERATIONS AND OBJECTIVES WITH BRS-BASED ACCELERATION METHOD.

Task Number	CPU Runtime (μs)		C2_S4			C3_S6			C4_S8			C5_S10		
	FDD	BRS	I _{PP}	A _R	G _{AP}	I _{PP}	A _R	G _{AP}	I _{PP}	A _R	G _{AP}	I _{PP}	A _R	G _{AP}
5	14.60	9.64	26.43	25.09	0.27	14.64	22.21	0.41	11.16	19.75	0.11	8.07	14.54	0.13
10	28.72	18.63	38.80	49.44	0.78	22.03	34.26	0.28	14.65	24.62	0.74	11.24	21.95	0.16
20	57.74	37.47	55.95	84.37	2.29	33.12	54.46	1.43	21.23	37.25	2.65	15.03	30.52	1.31
30	87.37	59.20	63.99	120.94	3.08	39.45	70.18	4.44	25.84	50.15	4.06	17.82	37.46	2.07
40	123.12	84.84	69.58	154.56	2.34	43.40	87.19	3.93	29.82	62.21	3.81	20.07	44.93	2.86

TABLE II
EXPERIMENTAL RESULTS FOR INSTANCES WITH 5 TASKS ($\Delta T = 3s$).

Instance	EDDBID	Gurobi		VNS		IG		GA		DABC		Ours	
		RPD (3s)	RPD (60s)	Avg	Std								
T5_I1	49.29	19.25	0.00	3.29	6.74	0.00	0.00	1.43	1.47	0.00	0.00	0.00	0.00
T5_I2	7.76	29.63	0.00	0.00	0.00	1.97	1.66	3.37	0.13	0.00	0.00	0.00	0.00
T5_I3	20.24	0.00	0.00	0.00	0.00								
T5_I4	47.25	134.34	0.00	0.39	1.76	0.00	0.00	1.16	1.63	0.00	0.00	0.00	0.00
T5_I5	2.33	52.09	0.00	0.00	0.00								
T5_I6	5.88	36.63	0.00	0.00	0.00	1.39	2.51	6.01	0.00	0.00	0.00	0.00	0.00
T5_I7	97.47	4.81	0.00	0.00	0.00	0.00	0.00	0.51	1.56	0.00	0.00	0.00	0.00
T5_I8	20.37	21.50	0.00	0.00	0.00	0.00	0.00	0.20	0.61	0.00	0.00	0.00	0.00
T5_I9	2.34	44.88	0.00	0.04	0.17	0.63	1.13	0.43	0.40	0.04	0.17	0.00	0.00
T5_I10	59.55	46.98	0.00	0.28	1.26	0.00	0.00	2.70	3.98	0.41	1.56	0.00	0.00
Mean	31.25	39.01	0.00	0.40	0.99	0.40	0.53	1.58	0.98	0.04	0.17	0.00	0.00

TABLE III
EXPERIMENTAL RESULTS FOR INSTANCES WITH 10 TASKS ($\Delta T = 12s$).

Instance	EDDBID	Gurobi		VNS		IG		GA		DABC		Ours	
		RPD (12s)	RPD (3600s)	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
T10_I1	89.85	882.77	12.20	6.68	2.41	4.85	1.99	9.01	1.13	3.08	1.92	3.04	1.42
T10_I2	69.61	343.51	12.97	5.33	5.21	2.79	2.86	8.80	4.49	5.55	2.25	0.52	0.84
T10_I3	56.81	325.59	7.79	4.28	1.00	3.71	1.51	2.92	1.09	2.91	1.12	2.20	0.96
T10_I4	27.37	819.79	9.01	4.11	4.59	2.33	2.33	10.93	2.63	4.85	2.24	0.79	1.21
T10_I5	95.81	523.29	15.98	15.68	4.27	6.45	3.17	9.76	1.99	7.63	0.69	3.82	2.30
T10_I6	59.81	—	23.82	7.74	7.53	2.69	2.53	8.93	3.20	1.54	0.68	1.40	1.03
T10_I7	74.15	896.44	0.00	1.00	2.15	0.29	0.37	0.40	0.38	0.13	0.14	0.01	0.07
T10_I8	54.02	421.28	14.56	5.72	3.51	2.71	3.75	7.50	6.60	0.57	1.39	0.65	2.00
T10_I9	50.51	512.42	22.85	7.21	5.91	2.22	2.20	18.14	3.90	2.99	2.43	0.22	0.31
T10_I10	66.45	682.43	7.14	1.79	2.56	4.19	2.77	6.57	1.68	2.00	0.75	0.75	0.96
Mean	64.44	—	12.63	5.95	3.91	3.22	2.35	8.30	2.71	3.12	1.36	1.34	1.11

(—) solution not found.

are provided in the supplementary file. To examine performance variations of metaheuristics across different instance scales, an analysis of variance is performed at a 95% confidence level using Tukey's Honest Significant Difference test. The interaction plot is presented in Fig. 9. It is observed that the performance of all heuristic algorithms declines as the instance scale increases. VNS, GA, DABC, and IG all show notable performance degradation on large-scale instances with 30 and 40 tasks, with mean RPDs exceeding 20%. Despite some fluctuations in performance, our method maintains mean RPDs around 10% with no noticeable increase when $n = 20, 30$, and 40 . The confidence intervals for our PN-based metaheuristic are significantly lower than those of other algorithms on large-scale instances, with no overlap observed. To further assess stability, RPD box-and-whisker plots are presented for instances with 10, 20, 30, and 40 tasks, as shown in Fig. 10. The boxplots of our method show smaller boxes, lower medians, and shorter whiskers, suggesting improved performance and stability over its peers.

To better understand the performance disparity between our method and other algorithms, we analyze their convergence

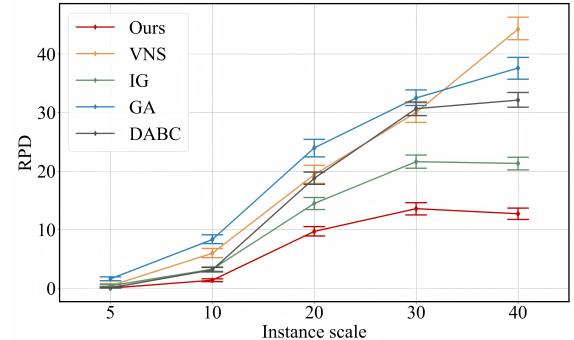


Fig. 9. Interaction plot between five metaheuristics and instance scales.

curves on two challenging instances, T30_I1 and T40_I1, to identify the underlying causes, as shown in Fig. 11. 1) *VNS*: The random shake in VNS typically generates extremely low-quality solutions to AHASP, rendering the neighborhood search ineffective and impeding convergence. 2) *IG*: The greedy local search in IG is computationally expensive, diminishing its global exploration capability and increasing the

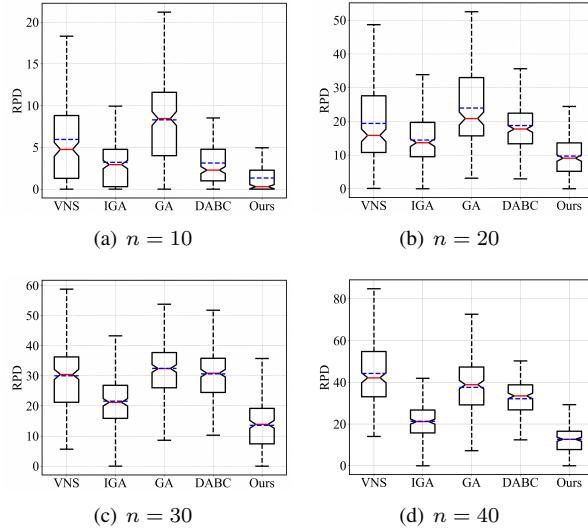


Fig. 10. Box-and-whisker plots of RPDs.

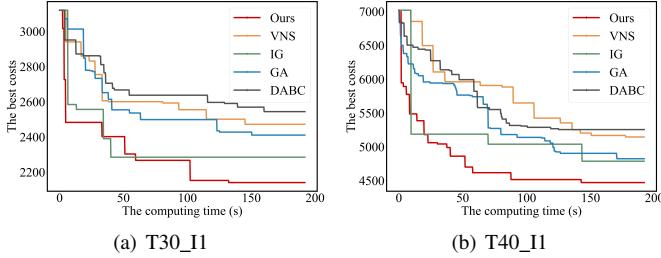


Fig. 11. Convergence curves of five metaheuristics.

risk of entrapment in local optima. 3) *GA*: The coupled dependencies in AHASP make genetic operations complex and susceptible to generating infeasible solutions, leading to suboptimal final solutions. 4) *DABC*: The framework of DABC is complex and involves multiple phases, which slows convergence. In contrast, our method incorporates a streamlined ALNS framework and efficient neighborhood strategies, ensuring fast and stable convergence to high-quality solutions.

E. Sensitivity Analysis

The configuration of carriers and shuttles critically influences transportation efficiency and is a primary concern for industry practitioners. This section presents a sensitivity analysis to provide managerial insights. The carrier count m^+ is examined at five levels: $\{2, 3, 4, 5, 6\}$, and the shuttle count m^- is examined at nine levels: $\{4, 5, 6, 7, 8, 9, 10, 11, 12\}$. Each combination is run for 20 times on instance T30_I1, and the average distance and tardiness for each combination are presented in the heatmaps shown in Fig. 12.

As shown in Fig. 12(a), the travel distance exhibits a non-monotonic relationship with the number of carriers and shuttles. Modifying AGV quantity has a limited effect on travel distance. In contrast, as depicted in Fig. 12(b), tardiness generally shows a negative correlation with the number of carriers and shuttles. Tardiness decreases significantly with an increase in the number of carriers and shuttles, especially when AGV count is low. However, a diminishing marginal effect is observed, as the reduction in tardiness becomes

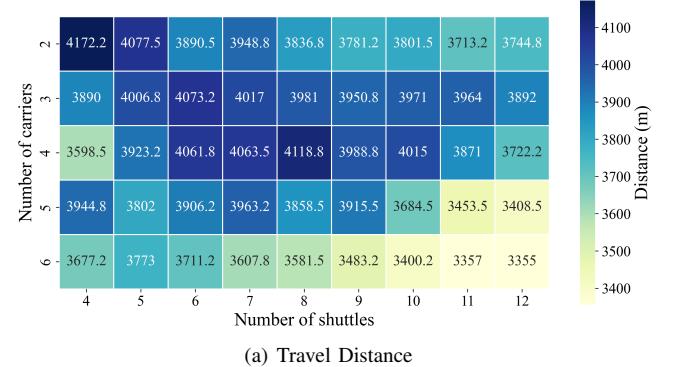


Fig. 12. Sensitivity heatmap for the configuration of carriers and shuttles.

negligible when the carrier count exceeds 10. Given the collaboration efficiency, each carrier can handle the transfer of 2 to 3 shuttles. Thus, the ratio $m^-/m^+ \in [2, 3]$ represents an optimal range. In the current fleet configuration of $m^+ = 4$ and $m^- = 8$, the travel distance is the second highest, while tardiness exceeds ten minutes. Therefore, considering the trade-off between transportation efficiency and financial investment, it is advisable to moderately increase the number of AGVs, with $m^+ = 5$, $m^- = 10$ as a preferred combination.

VII. CONCLUSION

This paper investigates a new scheduling problem involving attachable heterogeneous AGVs motivated by practical engineering applications. PNs are introduced to model the coupled AGV schedules and synchronize the operations of carriers and shuttles. Leveraging PN theory, we propose deadlock detection and prevention strategies to achieve deadlock-free scheduling. Furthermore, we develop a PN-based metaheuristic in the ALNS framework, and integrate an efficient acceleration technique into it. Extensive experiments on instances derived from real-world industrial settings validate the effectiveness of the proposed algorithm. The BRS-based acceleration method exhibits notable improvements in computational efficiency, validating the significance of PN-based deadlock-free scheduling. Comparative experiments demonstrate the poor performance of the on-site scheduling policy, while the substantial computational complexity of Gurobi renders it impractical for real-world industrial applications. In contrast, the proposed algorithm exhibits superior stability and optimization performance, outperforming four state-of-the-art metaheuristics. Furthermore, a sensitivity analysis provides managerial insights

for practitioners into the optimization of carrier and shuttle configurations.

Although this study advances the field, research on the scheduling of attachable heterogeneous AGVs remains limited. Three key research directions merit further exploration: 1) Extending the proposed PN model to more general heterogeneous agent scheduling contexts to facilitate deadlock analysis and resolution. 2) Exploring dynamic scheduling to accommodate industrial fluctuations and enhance system robustness and adaptability. 3) Developing novel solution methods, such as hybrid algorithms and deep reinforcement learning, is a key avenue for future research.

REFERENCES

- [1] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.
- [2] B. Fu, W. Smith, D. M. Rizzo, M. Castanier, M. Ghaffari, and K. Barton, "Robust task scheduling for heterogeneous robot teams under capability uncertainty," *IEEE Transactions on Robotics*, vol. 39, no. 2, pp. 1087–1105, 2022.
- [3] V. Baradaran, A. Shafaei, and A. H. Hosseini, "Stochastic vehicle routing problem with heterogeneous vehicles and multiple prioritized time windows: Mathematical modeling and solution approach," *Computers & Industrial Engineering*, vol. 131, pp. 187–199, 2019.
- [4] J. C. Molina, J. L. Salmeron, I. Eguia, and J. Racero, "The heterogeneous vehicle routing problem with time windows and a limited number of resources," *Engineering Applications of Artificial Intelligence*, vol. 94, p. 103745, 2020.
- [5] Ç. Koç, T. Bektaş, O. Jabali, and G. Laporte, "Thirty years of heterogeneous vehicle routing," *European Journal of Operational Research*, vol. 249, no. 1, pp. 1–21, 2016.
- [6] N. Singh, Q.-V. Dang, A. Akcay, I. Adan, and T. Martagan, "A matheuristic for agv scheduling with battery constraints," *European Journal of Operational Research*, vol. 298, no. 3, pp. 855–873, 2022.
- [7] Q.-V. Dang, N. Singh, I. Adan, T. Martagan, and D. van de Sande, "Scheduling heterogeneous multi-load agvs with battery constraints," *Computers & Operations Research*, vol. 136, p. 105517, 2021.
- [8] X. Bai, M. Cao, W. Yan, and S. S. Ge, "Efficient routing for precedence-constrained package delivery for heterogeneous vehicles," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 1, pp. 248–260, 2019.
- [9] R. Deng, R. Yan, P. Huang, Z. Shi, and Y. Zhong, "A distributed auction algorithm for task assignment with robot coalitions," *IEEE Transactions on Robotics*, 2024.
- [10] H. Wang and W. Chen, "Task scheduling for heterogeneous agents pickup and delivery using recurrent open shop scheduling models," *Robotics and Autonomous Systems*, vol. 172, p. 104604, 2024.
- [11] Y. Li and H. Huang, "Efficient task planning for heterogeneous agvs in warehouses," *IEEE Transactions on Intelligent Transportation Systems*, 2024.
- [12] M. C. Gombolay, R. J. Wilcox, and J. A. Shah, "Fast scheduling of robot teams performing tasks with temporospatial constraints," *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 220–239, 2018.
- [13] N. Kamra, T. S. Kumar, and N. Ayanian, "Combinatorial problems in multirobot battery exchange systems," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 852–862, 2017.
- [14] X. Chen, S. He, Y. Zhang, L. C. Tong, P. Shang, and X. Zhou, "Yard crane and agv scheduling in automated container terminal: A multi-robot task allocation framework," *Transportation Research Part C: Emerging Technologies*, vol. 114, pp. 241–271, 2020.
- [15] H. Wang, W. Chen, and J. Wang, "Coupled task scheduling for heterogeneous multi-robot system of two robot types performing complex-schedule order fulfillment tasks," *Robotics and Autonomous Systems*, vol. 131, p. 103560, 2020.
- [16] B. A. Ferreira, T. Petrović, M. Orsag, J. R. Martínez-de Dios, and S. Bogdan, "Distributed allocation and scheduling of tasks with cross-schedule dependencies for heterogeneous multi-robot teams," *IEEE Access*, 2024.
- [17] D. Wu, J. Zhang, A. Abusorrah, and M. Zhou, "A bin-packing-based method for path planning of air-ground cooperative system," *IEEE Transactions on Vehicular Technology*, vol. 74, no. 1, pp. 279–291, 2025.
- [18] J. Li, T. Sun, X. Huang, L. Ma, Q. Lin, J. Chen, and V. C. Leung, "A memetic path planning algorithm for unmanned air/ground vehicle cooperative detection systems," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 2724–2737, 2021.
- [19] P. Maini, K. Sundar, M. Singh, S. Rathinam, and P. Sujit, "Cooperative aerial-ground vehicle route planning with fuel constraints for coverage applications," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 6, pp. 3016–3028, 2019.
- [20] S. G. Manyam, K. Sundar, and D. W. Casbeer, "Cooperative routing for an air-ground vehicle team—exact algorithm, transformation method, and heuristics," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 1, pp. 537–547, 2019.
- [21] M. Drexel, "Synchronization in vehicle routing—a survey of vrps with multiple synchronization constraints," *Transportation Science*, vol. 46, no. 3, pp. 297–316, 2012.
- [22] R. Soares, A. Marques, P. Amorim, and S. N. Parragh, "Synchronization in vehicle routing: classification schema, modelling framework and literature review," *European Journal of Operational Research*, vol. 313, no. 3, pp. 817–840, 2024.
- [23] P. Eveborn, P. Flisberg, and M. Rönnqvist, "Laps care—an operational system for staff planning of home care," *European journal of operational research*, vol. 171, no. 3, pp. 962–976, 2006.
- [24] D. Bredström and M. Rönnqvist, "Combined vehicle routing and scheduling with temporal precedence and synchronization constraints," *European journal of operational research*, vol. 191, no. 1, pp. 19–31, 2008.
- [25] A. Dohn, M. S. Rasmussen, and J. Larsen, "The vehicle routing problem with time windows and temporal dependencies," *Networks*, vol. 58, no. 4, pp. 273–289, 2011.
- [26] M. S. Erdogan and R. M'Hallah, "Synchronizing delivery and installation with vehicle sharing: A hybrid adaptive large neighborhood search," *Computers & Industrial Engineering*, vol. 185, p. 109676, 2023.
- [27] J. Lehmann and M. Winkenbach, "A matheuristic for the two-echelon multi-trip vehicle routing problem with mixed pickup and delivery demand and time windows," *Transportation Research Part C: Emerging Technologies*, vol. 160, p. 104522, 2024.
- [28] C. S. Sartori, P. Smet, and G. V. Berghe, "Scheduling truck drivers with interdependent routes under european union regulations," *European Journal of Operational Research*, vol. 298, no. 1, pp. 76–88, 2022.
- [29] R. Masson, F. Lehuédé, and O. Péton, "Efficient feasibility testing for request insertion in the pickup and delivery problem with transfers," *Operations Research Letters*, vol. 41, no. 3, pp. 211–215, 2013.
- [30] P. Grangier, M. Gendreau, F. Lehuédé, and L.-M. Rousseau, "An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization," *European journal of operational research*, vol. 254, no. 1, pp. 80–91, 2016.
- [31] R. Liu, Y. Tao, and X. Xie, "An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits," *Computers & Operations Research*, vol. 101, pp. 250–262, 2019.
- [32] X. Wang, Y. Liang, X. Wei, and E. P. Chew, "An adaptive large neighborhood search algorithm for the tugboat scheduling problem," *Computers & Industrial Engineering*, vol. 177, p. 109039, 2023.
- [33] R. David and H. Alla, "Petri nets for modeling of dynamic systems: A survey," *Automatica*, vol. 30, no. 2, pp. 175–202, 1994.
- [34] Z. Li and M. Zhou, *Deadlock resolution in automated manufacturing systems: a novel Petri net approach*. Springer Science & Business Media, 2009.
- [35] P. Shaw, "Using constraint programming and local search methods to solve vehicle routing problems," in *International conference on principles and practice of constraint programming*. Springer, 1998, pp. 417–431.
- [36] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transportation science*, vol. 40, no. 4, pp. 455–472, 2006.
- [37] S. Voigt, "A review and ranking of operators in adaptive large neighborhood search for vehicle routing problems," *European Journal of Operational Research*, 2024.
- [38] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3774–3785, 2020.
- [39] Z. Zhao, Z. Bian, J. Liang, S. Liu, and M. Zhou, "Scheduling and logistics optimization for batch manufacturing processes with temperature constraints and alternative thermal devices," *IEEE Transactions on Industrial Informatics*, 2024.

- [40] R. Ruiz, Q.-K. Pan, and B. Naderi, "Iterated greedy methods for the distributed permutation flowshop scheduling problem," *Omega*, vol. 83, pp. 213–222, 2019.
- [41] D. C. Montgomery, *Design and analysis of experiments*. John wiley & sons, 2017.
- [42] X. Wang, W. Zou, L. Meng, B. Zhang, J. Li, and H. Sang, "Effective metaheuristic and rescheduling strategies for the multi-agv scheduling problem with sudden failure," *Expert Systems with Applications*, vol. 250, p. 123473, 2024.
- [43] W.-q. Zou, Q.-k. Pan, L.-l. Meng, H.-y. Sang, Y.-y. Han, and J.-q. Li, "An effective self-adaptive iterated greedy algorithm for a multi-agvs scheduling problem with charging and maintenance," *Expert Systems with Applications*, vol. 216, p. 119512, 2023.
- [44] Q. Liu, C. Wang, X. Li, and L. Gao, "An improved genetic algorithm with modified critical path-based searching for integrated process planning and scheduling problem considering automated guided vehicle transportation task," *Journal of Manufacturing Systems*, vol. 70, pp. 127–136, 2023.
- [45] W.-Q. Zou, Q.-K. Pan, T. Meng, L. Gao, and Y.-L. Wang, "An effective discrete artificial bee colony algorithm for multi-agvs dispatching problem in a matrix manufacturing workshop," *Expert Systems with Applications*, vol. 161, p. 113675, 2020.

SUPPLEMENTARY MATERIAL

S.I. PROOF FOR LEMMA 1

Proof. Assume, for the sake of contradiction, that $\exists t \in \mathcal{T}$ that can be fired more than once. Since t is enabled iff all its direct predecessor transitions in $\bigcup_{p \in \bullet t} \bullet p$ are fired, it follows that each transition in $\bigcup_{p \in \bullet t} \bullet p$ must be fired more than once. By induction, there exists a preceding transition t_i , where $t_i \preceq t$ and $i \in \{\pi_1^r \mid r \in \mathcal{V}^+ \cup \mathcal{V}^-\}$, that must also be fired more than once. However, since $\bullet p_i^+ = \emptyset$ or $\bullet p_i^- = \emptyset$, t_i cannot be re-fired. This leads to a contradiction, which disproves the assumption. Therefore, $\forall t \in \mathcal{T}$ can be fired at most once. \square

S.II. PROOF FOR THEOREM 1

A. Necessity of Theorem 1

Proof. Necessity

The necessity is proved by contradiction. Assume that when $\Sigma(\Pi)$ reaches M_F , $\exists t \in \mathcal{T}$ that remains unfired. Since t is unfired, $\forall t' \in \bigcup_{p \in \bullet t} p^\bullet$ also remains unfired. By induction, there exists a transition t^\dagger , where $t \preceq t^\dagger$, such that t^\dagger is unfired and $t^\dagger \bullet (\mathcal{P}_0^+ \cup \mathcal{P}_0^-) \neq \emptyset$. Thus, $\exists p_0^r \in \mathcal{P}_0^+ \cup \mathcal{P}_0^-$, $M_F(p_0^r) = 0$, which contradicts Definition 2. Therefore, $\forall t \in \mathcal{T}$ must be fired when $\Sigma(\Pi)$ reaches M_F . By Lemma 1, each transition $t \in \mathcal{T}$ can be fired at most once. Hence, when $\Sigma(\Pi)$ reaches M_F , $\forall t \in \mathcal{T}$ is fired exactly once. \square

B. Sufficiency of Theorem 1

Proof. Sufficiency

To prove the sufficiency of Theorem 1, it suffices to show that if $\forall t \in \mathcal{T}$ is fired exactly once, then $\Sigma(\Pi)$ reaches the final marking M_F .

By Lemma 1, for $\forall t_i \in \mathcal{T}$, if t_i is fired at marking $M_i \in \mathcal{R}(M_0)$ such that $M_i[t_i] M'_i$, then $\forall M_i^* \in \mathcal{R}(M'_i)$, $M_i^*(p_i^+) = M_i^*(p_i^-) = 0$, where $p_i^+ \in \mathcal{P}^+$, $p_i^- \in \mathcal{P}^-$. For any place $p_0^r \in \mathcal{P}_0^+ \cup \mathcal{P}_0^-$, if its unique input transition $t_j \in \bullet p_0^r$ (with $|\bullet p_0^r| = 1$) is fired at M_j such that $M_j[t_j] M'_j$, then $\forall M_j^* \in \mathcal{R}(M'_j)$, $M_j^*(p_0^r) = 1$. Therefore, if all transitions in \mathcal{T} are fired once, then $\forall p_0^r \in \mathcal{P}_0^+ \cup \mathcal{P}_0^-$, $M_F(p_0^r) = 1$, and $\forall p \in \mathcal{P}^+ \cup \mathcal{P}^-$, $M_F(p) = 0$. By Definition 2, $\Sigma(\Pi)$ reaches final marking M_F . \square

S.III. PROOF FOR THEOREM 2

A. Necessity of Theorem 2

Lemma 2. In PN $\Sigma(\Pi)$, $\forall M \in \mathcal{R}(M_0)$, the following holds:

$$\sum_{p \in \mathcal{P}^+ \cup \mathcal{P}_0^+} M(p) = m^+ \quad (\text{S.1})$$

$$\sum_{p \in \mathcal{P}^- \cup \mathcal{P}_0^-} M(p) = m^- \quad (\text{S.2})$$

Proof. In $\Sigma(\Pi)$, $\forall t_i \in \mathcal{T}$, $|\bullet t_i| = |t_i^\bullet| = 2$. The input places of t_i , denoted by p_i^+ and p_i^- , satisfy $p_i^+ \in \mathcal{P}^+$ and $p_i^- \in \mathcal{P}^-$. The output places of t_i , denoted by p_j^+ and p_k^- , satisfy $p_j^+ \in \mathcal{P}^+ \cup \mathcal{P}_0^+$ and $p_k^- \in \mathcal{P}^- \cup \mathcal{P}_0^-$. Since $M[t_i] M'$ removes one

token from both p_i^+ and p_i^- , and adds one token to both p_j^+ and p_k^- , it follows that:

$$\sum_{p \in \mathcal{P}^+ \cup \mathcal{P}_0^+} M(p) = \sum_{p \in \mathcal{P}^+ \cup \mathcal{P}_0^+} M'(p) \quad (\text{S.3})$$

$$\sum_{p \in \mathcal{P}^- \cup \mathcal{P}_0^-} M(p) = \sum_{p \in \mathcal{P}^- \cup \mathcal{P}_0^-} M'(p) \quad (\text{S.4})$$

Hence, the token sums are invariant under any firing sequence. As a result, $\forall M \in \mathcal{R}(M_0)$, the following invariants hold:

$$\sum_{p \in \mathcal{P}^+ \cup \mathcal{P}_0^+} M(p) = \sum_{p \in \mathcal{P}^+ \cup \mathcal{P}_0^+} M_0(p) = m^+ \quad (\text{S.5})$$

$$\sum_{p \in \mathcal{P}^- \cup \mathcal{P}_0^-} M(p) = \sum_{p \in \mathcal{P}^- \cup \mathcal{P}_0^-} M_0(p) = m^- \quad (\text{S.6})$$

\square

Lemma 3. In PN $\Sigma(\Pi)$, $\forall M \in \mathcal{R}(M_0) \setminus M_F$, $\sum_{p \in \mathcal{P}^+} M(p) > 0$ and $\sum_{p \in \mathcal{P}^-} M(p) > 0$.

Proof. By Definition 2, final marking M_F satisfies $\sum_{p \in \mathcal{P}_0^+} M_F(p) = m^+$ and $\sum_{p \in \mathcal{P}_0^-} M_F(p) = m^-$. According to Theorem 1, $\forall M \in \mathcal{R}(M_0) \setminus M_F$, there exists at least one unfired transition. Thus, $\forall M \in \mathcal{R}(M_0) \setminus M_F$, $\sum_{p \in \mathcal{P}_0^+} M(p) < m^+$, $\sum_{p \in \mathcal{P}_0^-} M(p) < m^-$. By Lemma 2, the total number of tokens in $\mathcal{P}^+ \cup \mathcal{P}_0^+$ and $\mathcal{P}^- \cup \mathcal{P}_0^-$ is invariant. Therefore, $\forall M \in \mathcal{R}(M_0) \setminus M_F$, it holds that $\sum_{p \in \mathcal{P}^+} M(p) > 0$ and $\sum_{p \in \mathcal{P}^-} M(p) > 0$. \square

Proof. Necessity

The necessity of Theorem 2 is proved by contradiction. Assuming a deadlock occurs at a reachable marking $M_D \in \mathcal{R}(M_0) \setminus \{M_F\}$ in PN $\Sigma(\Pi)$, we demonstrate that solution Π is infeasible.

Let $\mathcal{P}_d^+ = \{p \in \mathcal{P}^+ \mid M_D(p) = 1\}$ and $\mathcal{T}_d^+ = \bigcup_{p \in \mathcal{P}_d^+} p^\bullet$. Since $M_D \neq M_F$, it follows from Lemma 3 that $\sum_{p \in \mathcal{P}^+} M_D(p) > 0$. Thus, $\mathcal{P}_d^+ \neq \emptyset$ and $\mathcal{T}_d^+ \neq \emptyset$. Let $Post^+(i)$ denote the set of successors of task i in Π^+ , and define $\Psi^+(t_i) = \{t_j \mid j \in Post^+(i)\}$. Similarly, let $Pre^-(i)$ denote the set of predecessors of task i in Π^- , and define $\Phi^-(t_i) = \{t_j \mid j \in Pre^-(i)\}$. Assume, for contradiction, that $\forall t \in \mathcal{T}_d^+$, $\Psi^+(t) = \emptyset$. Under this assumption, at M_D , only the transitions in \mathcal{T}_d^+ remain unfired, i.e., all transitions in $\mathcal{T} \setminus \mathcal{T}_d^+$ are fired. Therefore, $\forall t_i \in \mathcal{T} \setminus \mathcal{T}_d^+$, its input place p_i^- satisfies $M_D(p_i^-) = 0$. By Lemma 3, given that $\sum_{p \in \mathcal{P}^-} M_D(p) > 0$, there exists $t_j \in \mathcal{T}_d^+$ such that its input place p_j^- satisfies $M_D(p_j^-) = 1$. Because $t_j \in \mathcal{T}_d^+$, its input place p_j^+ also satisfies $M_D(p_j^+) = 1$. Thus, $M_D[t_j]$, which contradicts the occurrence of deadlock at M_D . Therefore, the assumption that $\forall t_i \in \mathcal{T}_d^+$, $\Psi^+(t_i) = \emptyset$ is invalid. Hence, there exists a non-empty subset, denoted as $\tilde{\mathcal{T}}_d^+ \subseteq \mathcal{T}_d^+$, such that $\forall t \in \tilde{\mathcal{T}}_d^+$, $\Psi^+(t) \neq \emptyset$. It follows that the set of all unfired transitions at M_D , denoted as $\mathcal{U}(M_D)$, is given by

$$\mathcal{U}(M_D) = \left(\bigcup_{t \in \tilde{\mathcal{T}}_d^+} \Psi^+(t) \right) \cup \mathcal{T}_d^+ \quad (\text{S.7})$$

For any $t_i \in \tilde{\mathcal{T}}_d^+$, since M_D leads to deadlock and $M_D(p_i^+) = 1$, it holds that $M_D(p_i^-) = 0$. Therefore, t_i has

at least one unfired predecessor at M_D . Let $\tilde{\Phi}^-(t_i) \subseteq \Phi^-(t_i)$ denote the set of unfired predecessor transitions of t_i at M_D . From (S.7), the set of all such unfired predecessors satisfies:

$$\tilde{\Phi}^-(t) \subset \mathcal{U}(M_D), \forall t \in \tilde{\mathcal{T}}_d^+ \quad (\text{S.8})$$

From (10), the temporal expression of the precedence relations yields the following inequalities:

$$T_i^d < \min_{t_j \in \Psi^+(t_i)} (T_j^d), \forall t_i \in \tilde{\mathcal{T}}_d^+ \quad (\text{S.9})$$

$$T_i^d > \max_{t_j \in \Phi^-(t_i)} (T_j^d), \forall t_i \in \tilde{\mathcal{T}}_d^+ \quad (\text{S.10})$$

For any $t_i \in \tilde{\mathcal{T}}_d^+$, let $t_k = \arg \max_{t_j \in \Phi^-(t_i)} (T_j^d)$. By (S.8), there exists $t_{i^\dagger} \in \tilde{\mathcal{T}}_d^+$ such that $t_k \in \Psi^+(t_{i^\dagger}) \cup \{t_{i^\dagger}\}$. Therefore, combining inequalities (S.9) and (S.10), for any $t_i \in \tilde{\mathcal{T}}_d^+$, there exists $t_{i^\dagger} \in \tilde{\mathcal{T}}_d^+ \setminus \{t_i\}$ such that

$$T_i^d > \max_{t_j \in \Phi^-(t_i)} (T_j^d) \geq \min_{t_j \in \Psi^+(t_{i^\dagger}) \cup \{t_{i^\dagger}\}} (T_j^d) \geq T_{i^\dagger}^d \quad (\text{S.11})$$

Let $t_q = \arg \min_{t_i \in \tilde{\mathcal{T}}_d^+} (T_i^d)$. According to inequality (S.11), since $t_q \in \tilde{\mathcal{T}}_d^+$, there exists $t_{q^\dagger} \in \tilde{\mathcal{T}}_d^+ \setminus \{t_q\}$ such that $T_q^d > T_{q^\dagger}^d$. However, by the definition of t_q , we have $T_q^d < T_{q^\dagger}^d$, which leads to a contradiction.

In conclusion, if $\Sigma(\Pi)$ reaches a deadlock marking $M_D \in \mathcal{R}(M_0) \setminus \{M_F\}$, then the precedence relations imposed by solution Π contain a contradiction, rendering Π infeasible. \square

B. Sufficiency of Theorem 2

Proof. Sufficiency

The sufficiency of Theorem 2 is proved by contradiction. Assuming solution Π is infeasible, we demonstrate that $\exists M_D \in \mathcal{R}(M_0) \setminus \{M_F\}$ that leads to deadlock.

Since Π is infeasible, there exist tasks $i, j \in \mathcal{N}, i \neq j$, such that both $i \prec j$ and $j \prec i$ hold, forming a circular precedence relation. By the structure of $\Sigma(\Pi)$, $i \prec j$ implies the existence of a directed path from transition t_i to transition t_j , denoted as \mathcal{D}_{ij} . Similarly, $j \prec i$ implies the existence of a directed path from t_j to t_i , denoted as \mathcal{D}_{ji} . Thus, \mathcal{D}_{ij} and \mathcal{D}_{ji} form a directed cycle, denoted as \mathcal{C}^\dagger . Let \mathcal{P}^\dagger and \mathcal{T}^\dagger be the sets of places and transitions contained in \mathcal{C}^\dagger , respectively. For any $p \in \mathcal{P}^\dagger$, it holds that $|{}^\bullet p| = |p^\bullet| = 1$ and $M_0(p) = 0$. Therefore, for any $t \in \mathcal{T}^\dagger$, $\#M \in \mathcal{R}(M_0)$ such that $M[t]$. By Theorem 1, as transitions in \mathcal{T}^\dagger cannot be fired, M_F is unreachable. Thus, there exists $M_D \in \mathcal{R}(M_0) \setminus \{M_F\}$ that leads to deadlock. \square

S.IV. PROOF FOR ALGORITHM 2

Proof. Soundness

To prove the soundness of Algorithm 2, it suffices to show that every insertion position identified by BRS is infeasible.

For any $\overleftarrow{\epsilon}^-(b) \in \mathcal{L}_B$, according to Algorithm 2, transition t_k is reachable from t_b in $\Sigma(\Pi)$, implying $b \prec k$. Inserting task k at position $\overleftarrow{\epsilon}^-(b)$ imposes an additional precedence relation $k \prec b$, which contradicts $b \prec k$. Hence, position $\overleftarrow{\epsilon}^-(b)$ is infeasible.

Similarly, for any $\overrightarrow{\epsilon}^-(f) \in \mathcal{L}_F$, since t_f is reachable from t_k , we have $k \prec f$. Inserting task k at position $\overrightarrow{\epsilon}^-(f)$ imposes

$f \prec k$, which contradicts $k \prec f$. Thus, position $\overrightarrow{\epsilon}^-(f)$ is infeasible.

In conclusion, all positions in $\mathcal{L}_B \cup \mathcal{L}_F$ are infeasible, establishing the soundness of Algorithm 2. \square

Proof. Completeness

To prove the completeness of Algorithm 2, it suffices to show that if position $\overrightarrow{\epsilon}^-(f) / \overleftarrow{\epsilon}^-(b)$ is infeasible, it will be identified by BRS.

Inserting task k at position $\overrightarrow{\epsilon}^-(f) / \overleftarrow{\epsilon}^-(b)$ introduces the precedence relations $k \prec b$ and $f \prec k$. If the position is infeasible, at least one of the following precedence relations must already hold before the insertion: $b \prec k$ or $k \prec f$. If $b \prec k$ holds, transition t_k is reachable from t_b , and thus position $\overleftarrow{\epsilon}^-(b)$ will be detected by the backward search of BRS. If $k \prec f$ holds, transition t_f is reachable from t_k , and thus position $\overrightarrow{\epsilon}^-(f)$ will be detected by the forward search of BRS. Therefore, all infeasible positions are identified by BRS. \square

S.V. FIRING PROCESS

The FDD process for the example solution Π given in Section IV-A is represented by the firing sequence $M_0[t_5]M_1[t_6]M_2[t_1]M_3[t_4]M_4[t_3]M_5[t_7]M_6[t_2]M_F$. The firing process of PN $\Sigma(\Pi)$ is specified in Table S.1 and visualized in Fig. S.1.

TABLE S.1
FIRING-DRIVEN DECODING FOR THE EXAMPLE SOLUTION Π .

Marking	Places with one token	Fired transition
M_0	p_5^+ p_5^- p_6^+ p_6^-	-
M_1	p_4^+ p_1^- p_6^+ p_6^-	t_5
M_2	p_4^+ p_1^- p_1^+ p_3^-	t_6
M_3	p_4^+ p_4^- p_3^+ p_3^-	t_1
M_4	p_2^+ p_7^- p_3^+ p_3^-	t_4
M_5	p_2^+ p_7^- p_7^+ p_2^-	t_3
M_6	p_2^+ p_2^- p_3^0 p_2^-	t_7
M_F	p_1^0 p_2^0 p_3^0 p_4^0	t_2

Solution $\Pi = (\Pi^+, \Pi^-)$, where $\Pi^+ = (0, 5, 4, 2, 0, 6, 1, 3, 7)$ and $\Pi^- = (0, 5, 1, 4, 7, 0, 6, 3, 2)$.

S.VI. SUPPLEMENTARY EXPERIMENTAL RESULTS

A. Evaluation of Removal and Insertion Operators

Removal operators are divided into two categories: the random removal strategy (RRS), which includes RR, and the heuristic removal strategy (HRS), which comprises HCR, LDR, LTR, and SR. The three developed insertion operators, GI, UGI, and CGI, all follow the greedy insertion strategy (GIS). Two commonly used alternative insertion methods are the random insertion strategy (RIS) and the perturbed-greedy insertion strategy (PIS). The RPD gaps between the base algorithm (RRS and HRS for removal, GIS for insertion) and its variants are compared, as shown in Table S.2. Columns 2–3 present the results obtained using only RRS or HRS for removal; columns 4–5 use only RIS or PIS for insertion; and columns 6–7 show the impact of incorporating RIS or PIS into GIS.

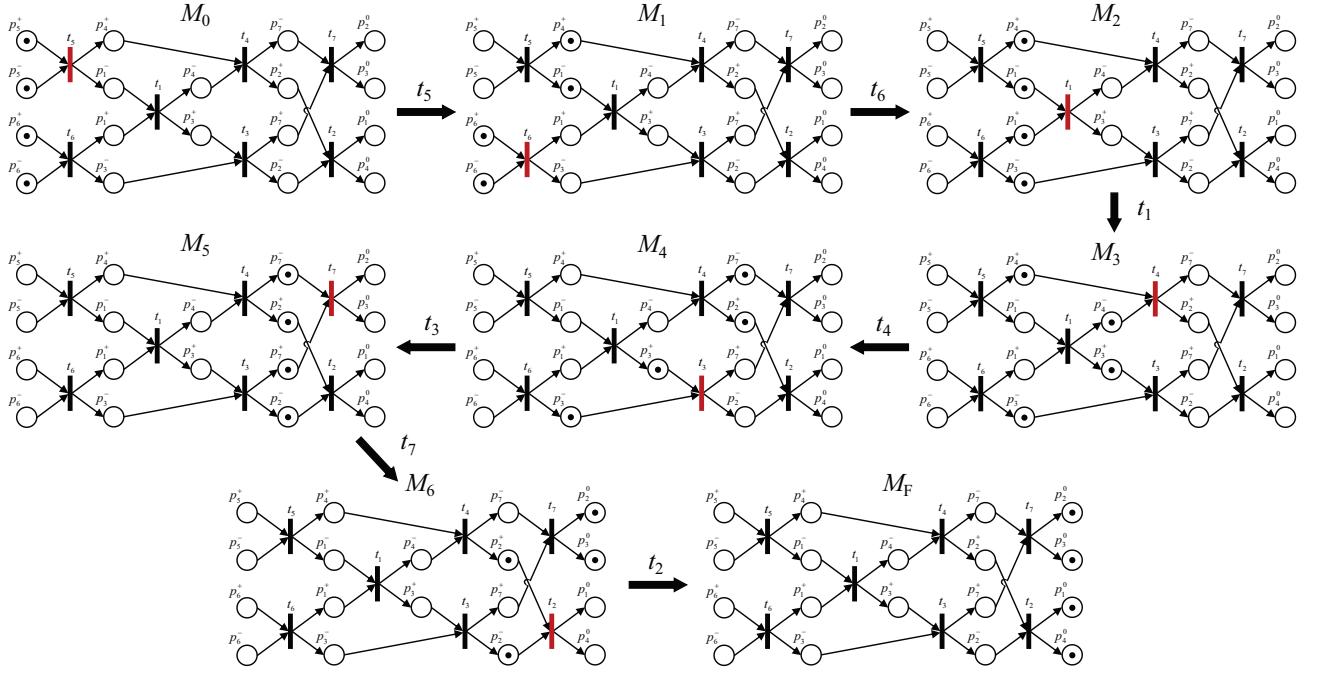


Fig. S.1. Firing process of PN $\Sigma(\Pi)$.

As shown in Table S.2, RRS matches the base algorithm only on the small-scale instance T5_I1 and performs poorly on larger ones. Excluding RRS hinders ALNS from achieving optimality on T5_I1, with HRS exhibiting a 2.51% gap. HRS slightly improves T30_I1 results but offers limited overall optimization. Regarding insertion, RIS alone yields poor-quality solutions with a large optimality gap. Hence, incorporating RIS into GIS deteriorates optimization performance. Similarly, the perturbation introduced by PIS fails to enhance search diversity, and its integration into GIS negatively impacts optimization. Therefore, all three insertion operators are designed based on GIS. The combination of RRS and HRS for removal, along with GIS for insertion, yields the best performance.

TABLE S.2
RPD GAPS BETWEEN THE BASE ALGORITHM AND ITS VARIANTS.

Instance	Removal		Insertion			
	RRS	HRS	RIS	PIS	GIS_RIS	GIS_PIS
T5_I1	0.00	2.51	13.52	0.00	0.00	0.00
T10_I1	1.20	0.25	60.88	9.27	0.29	1.51
T20_I1	6.66	0.87	93.40	23.24	3.28	1.91
T30_I1	10.22	-0.32	159.56	37.22	2.15	1.29
T40_I1	5.44	2.21	131.74	40.32	2.47	2.17
Average	4.70	1.52	91.82	22.01	1.64	1.38

B. Settings of Compared Metaheuristic

The parameters of the compared metaheuristic algorithms are tuned using the same method as ALNS. The results are summarized in Table S.3, which also specifies the operators used by each algorithm to facilitate reproducibility.

C. Comparison Results of Metaheuristic Algorithms

The experimental results of the developed ALNS and four comparative metaheuristics across all instances are presented in Table S.4.

TABLE S.3
PARAMETERS AND OPERATIONS FOR COMPARISON METAHEURISTICS.

Algorithm	Parameter	Operation
VNS	—	Insertion, Task-Swap, Route-Swap
IG	$d = 0.3 \cdot n$	Random Destruction, Greedy Construction, Referenced Local Search
GA	$PS = 100, P_c = 0.9, P_m = 0.1$	Single-Point Crossover, Insertion / Swap Mutation
DABC	$PS = 100, LIMIT = 1000, Rep = 5$	Insertion, Task-Swap, Route-Swap

(—) no parameters involved.

TABLE S.4
EXPERIMENTAL RESULTS FOR ALL INSTANCES.

Instance	EDDBID	VNS		IG		GA		DABC		Ours	
		Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
T5_I1	49.29	3.29	6.74	0.00	0.00	1.43	1.47	0.00	0.00	0.00	0.00
T5_I2	7.76	0.00	0.00	1.97	1.66	3.37	0.13	0.00	0.00	0.00	0.00
T5_I3	20.24	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
T5_I4	47.25	0.39	1.76	0.00	0.00	1.16	1.63	0.00	0.00	0.00	0.00
T5_I5	2.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
T5_I6	5.88	0.00	0.00	1.39	2.51	6.01	0.00	0.00	0.00	0.00	0.00
T5_I7	97.47	0.00	0.00	0.00	0.00	0.51	1.56	0.00	0.00	0.00	0.00
T5_I8	20.37	0.00	0.00	0.00	0.00	0.20	0.61	0.00	0.00	0.00	0.00
T5_I9	2.34	0.04	0.17	0.63	1.13	0.43	0.40	0.04	0.17	0.00	0.00
T5_I10	59.55	0.28	1.26	0.00	0.00	2.70	3.98	0.41	1.56	0.00	0.00
Mean	31.25	0.40	0.99	0.40	0.53	1.58	0.98	0.04	0.17	0.00	0.00
T10_I1	89.85	6.68	2.41	4.85	1.99	9.01	1.13	3.08	1.92	3.04	1.42
T10_I2	69.61	5.33	5.21	2.79	2.86	8.80	4.49	5.55	2.25	0.52	0.84
T10_I3	56.81	4.28	1.00	3.71	1.51	2.92	1.09	2.91	1.12	2.20	0.96
T10_I4	27.37	4.11	4.59	2.33	2.33	10.93	2.63	4.85	2.24	0.79	1.21
T10_I5	95.81	15.68	4.27	6.45	3.17	9.76	1.99	7.63	0.69	3.82	2.30
T10_I6	59.81	7.74	7.53	2.69	2.53	8.93	3.20	1.54	0.68	1.40	1.03
T10_I7	74.15	1.00	2.15	0.29	0.37	0.40	0.38	0.13	0.14	0.01	0.07
T10_I8	54.02	5.72	3.51	2.71	3.75	7.50	6.60	0.57	1.39	0.65	2.00
T10_I9	50.51	7.21	5.91	2.22	2.20	18.14	3.90	2.99	2.43	0.22	0.31
T10_I10	66.45	1.79	2.56	4.19	2.77	6.57	1.68	2.00	0.75	0.75	0.96
Mean	64.44	5.95	3.91	3.22	2.35	8.30	2.71	3.12	1.36	1.34	1.11
T20_I1	95.70	12.48	3.48	17.85	5.59	18.88	2.30	17.22	1.87	11.07	4.17
T20_I2	134.00	38.45	8.67	22.12	7.77	42.31	6.41	33.95	5.00	14.12	3.96
T20_I3	77.52	8.35	3.03	10.71	4.80	10.91	3.16	9.90	2.72	6.81	3.51
T20_I4	112.78	22.45	6.80	9.62	5.86	35.01	7.03	18.67	3.43	6.11	3.90
T20_I5	73.62	10.76	4.27	11.88	4.46	24.55	5.05	16.12	2.52	7.10	4.58
T20_I6	199.03	10.61	4.11	7.90	4.73	11.46	2.19	10.64	2.83	4.62	3.65
T20_I7	106.45	20.43	6.21	15.79	5.37	19.54	2.67	20.55	2.86	13.53	7.37
T20_I8	99.70	12.19	3.49	11.30	6.27	16.68	2.49	14.63	2.79	8.85	3.97
T20_I9	89.77	32.41	8.67	24.56	4.47	32.96	7.52	25.88	5.06	12.87	6.91
T20_I10	143.64	25.71	6.73	12.82	5.72	27.23	6.13	20.05	4.99	11.81	5.28
Mean	113.22	19.38	5.55	14.45	5.51	23.95	4.49	18.76	3.41	9.69	4.73
T30_I1	155.32	34.42	11.73	21.86	8.34	42.35	8.60	42.90	7.15	15.87	7.77
T30_I2	116.98	33.58	7.75	24.62	7.61	37.56	5.79	36.10	4.12	13.43	6.02
T30_I3	142.73	17.10	3.89	19.96	7.27	23.80	4.50	24.48	4.13	9.92	5.84
T30_I4	153.13	17.11	7.50	13.62	6.87	23.73	5.44	25.09	3.30	9.29	5.74
T30_I5	156.34	42.50	11.75	20.32	9.60	33.95	4.49	26.66	4.69	9.26	7.09
T30_I6	158.92	26.57	6.10	25.09	6.77	28.51	5.36	29.66	3.46	17.67	7.98
T30_I7	210.80	34.73	10.25	21.98	8.67	40.59	9.31	35.16	5.78	11.43	5.25
T30_I8	110.59	23.96	5.73	16.60	5.11	22.86	8.59	18.00	3.10	10.81	5.68
T30_I9	119.51	37.68	7.44	25.27	5.47	40.96	6.37	35.73	4.39	18.65	7.62
T30_I10	137.40	32.44	3.33	26.60	7.55	30.51	3.39	32.77	3.14	19.34	8.07
Mean	146.17	30.01	7.55	21.59	7.33	32.48	6.19	30.65	4.33	13.57	6.71
T40_I1	144.38	43.77	8.80	25.53	7.90	35.53	6.65	41.54	4.69	16.27	8.34
T40_I2	82.38	30.94	5.97	18.52	7.22	16.04	5.05	18.77	3.11	12.01	4.80
T40_I3	136.85	40.02	10.75	24.18	6.51	46.66	6.63	40.19	7.86	13.56	7.71
T40_I4	87.34	51.97	8.84	15.61	6.99	43.34	7.63	35.16	3.55	12.84	6.05
T40_I5	138.43	44.99	10.18	18.87	8.34	46.93	9.09	30.64	6.07	11.98	6.20
T40_I6	109.06	25.51	5.08	15.26	5.51	21.28	7.43	21.10	4.51	10.25	5.77
T40_I7	169.87	35.16	7.86	23.70	8.87	31.62	6.78	32.78	3.42	12.49	5.69
T40_I8	96.97	45.63	7.82	23.91	6.41	32.02	5.41	34.39	6.16	11.75	5.62
T40_I9	182.92	64.79	12.52	25.31	9.81	53.71	11.41	30.49	8.66	10.11	5.54
T40_I10	175.35	59.53	10.93	22.08	5.71	48.79	5.69	36.04	6.80	15.73	10.72
Mean	132.36	44.23	8.88	21.30	7.33	37.59	7.18	32.11	5.48	12.70	6.65