http://pubsonline.informs.org/journal/ijoc

INFORMS JOURNAL ON COMPUTING
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0899-1499, EISSN 1526-5528

Submitted to *INFORMS Journal on Computing*

A simheuristic for the cross-docking scheduling problem under uncertain truck arrival times

Wei Yang, Yang Wang*
School of Management, Northwestern Polytechnical University,127 Youyi West Road, 710072 Xi'an, China, weiyang1905@gmail.com, yangw@nwpu.edu.cn

Jin-Kao Hao
LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49000 Angers, France, jin-kao.hao@univ-angers.fr

Ada Che*
School of Management, Northwestern Polytechnical University,127 Youyi West Road, 710072 Xi'an, China, ache@nwpu.edu.cn

Abraham P. Punnen
Department of Mathematics, Simon Fraser University Surrey, 250-13450 102nd AV, Surrey, British Columbia, V3T 0A3, Canada, apunnen@sfu.ca

*Corresponding author

Abstract. Cross-docking is a strategic imperative for just-in-time logistics and manufacturing. This paper addresses the stochastic cross-docking scheduling problem (SCDSP) by incorporating flexible dock occupancy based on actual processing durations and uncertain truck arrivals, aiming to minimize penalties for untransferred goods and tardy departures. We propose a scheduling policy based on fixed sequences and release times to generate real-time schedules dynamically. To solve the SCDSP, we develop a nested simulation-based local search (NSBLS) framework. Specifically, a variable neighborhood descent optimizes truck sequences in the deterministic setting, while a simulation-based local search refines truck release times for promising candidates filtered by a cutoff rule under uncertainty. A key innovation is a fast evaluation technique for scenario sets, which significantly reduces the computational overhead by bypassing redundant simulations during neighborhood exploration. To further enhance solution quality, the NSBLS is integrated into a hybrid genetic search framework, resulting in a hybrid genetic simheuristic (SimHGS). Extensive computational experiments validate the efficiency of the SimHGS. Compared with baseline policies, our approach achieves superior performance across varying levels of uncertainty. Sensitivity and ablation analyses further confirm the policy's adaptability and the contribution of each algorithmic component to both search efficiency and solution robustness.

Key words: Cross-docking scheduling; simulation-based local search; simheuristic; hybrid genetic search; metaheuristic

1. Introduction

Cross-docking is a distribution strategy in which goods are transferred directly from inbound to outbound trucks with minimal intermediate storage. By reducing storage and handling costs, it has become a cornerstone of modern logistics, especially for express delivery and perishable goods (Boysen et al. 2019, Zaerpour et al. 2015). The efficiency of a cross-docking terminal depends critically on the spatiotemporal synchronization of inbound and outbound trucks (Buijs et al. 2014).

Two closely related decisions are central to cross-docking operations: determining *when* each truck should be processed and *where* it should be assigned within the dock area. The *where* decision, commonly known as the *dock-door assignment* or *truck-to-door assignment*, focuses on allocating trucks to specific doors in a manner that minimizes the internal transport distance of goods between inbound and outbound doors. This challenge is formally captured by the cross-dock door assignment problem (CDAP) (Guignard et al. 2012, Guemri et al. 2019, Sokol et al. 2020). Complementing this is the *when* decision, referred to as the *truck scheduling*, which involves determining the sequence and timing of truck operations to minimize performance metrics such as total tardiness or makespan (Boysen et al. 2013). These two decisions are inherently interdependent and jointly define the broader *cross-docking scheduling problem* (CDSP) (Buijs et al. 2014, Rijal et al. 2019). Effectively coordinating door assignments and truck schedules is critical to maximizing throughput, minimizing delays, and fully leveraging the potential of cross-docking systems.

In the context of truck scheduling, *time windows* are typically defined as the interval between a truck's arrival and its latest allowable departure time, within which the truck can be serviced (Bodnar et al. 2015, Rijal et al. 2019). However, this conventional definition focuses solely on operational feasibility and overlooks driver service preferences. In practice, cross-docking systems face significant uncertainty in truck arrival times due to traffic congestion, weather conditions, and other disruptions (Konur and Golias 2013a, Zaerpour et al. 2015, Xi et al. 2020). This variability disrupts tightly synchronized operations and induces a fundamental trade-off: delaying a truck may reduce untransferred goods by allowing more inbound consolidation, but at the cost of increased waiting time and degraded service quality. To explicitly capture this trade-off, we introduce an *expected departure time* that reflects the driver's preferred service completion. This representation, which acts as a soft target alongside the rigid departure deadline, allows for a more nuanced balance between maximizing operational efficiency and mitigating driver dissatisfaction stemming from schedule volatility under uncertainty.

To accurately position our work, we classify the existing CDSP literature into two streams based on their temporal representation. The first stream adopts discrete-time or time-indexed formulations (Ladier and Alpan 2018, 2016, Bodnar et al. 2015, Rijal et al. 2019), some of which further incorporate mixed service modes (Bodnar et al. 2015, Rijal et al. 2019). While effective for decisions over predefined time slots, these models cannot capture the precise, continuous-time coordination among unloading completion, internal transfer, and outbound loading start times required in our study. The second stream employs continuous-time representations, notably Miao et al. (2014). However, our problem diverges from theirs in three fundamental ways. First, unlike Miao et al. (2014), who assume a truck occupies a dock for its entire time window, we model actual processing times. This allows a door to be released immediately upon service completion, permitting overlapping time windows as long as the physical service intervals remain disjoint. Consequently, our model necessitates the simultaneous optimization of dock assignment, sequencing, and service start times. Second, we define internal transfer time as a function of both goods

flow and physical distance. This captures the varying number of trips required for different flow volumes, ensuring temporal feasibility that constant-time estimates often overlook. Third, we incorporate stochastic truck arrivals to account for operational uncertainty. By modeling arrival variability, we ensure the schedule's reliability under real-world fluctuations, moving beyond the deterministic assumptions of prior work. Other continuous-time studies under uncertainty (Acar et al. 2012, Konur and Golias 2013a,b, Xi et al. 2020) remain distinct, as they fail to capture the coupled inbound–outbound interactions and goods transfer constraints central to this research.

Among the various approaches for addressing large-scale optimization problems under uncertainty, simheuristics, which combine metaheuristic optimization with simulation techniques, have gained prominence as an effective "first-resort" methodology (Chica et al. 2017). The central premise underlying simheuristics is that high-quality solutions to the deterministic version of a problem are typically high quality when applied to their stochastic version. Leveraging this assumption, simheuristics employ a metaheuristic component to explore the solution space of the deterministic problem and identify promising candidates. These candidate solutions are then assessed using simulation to estimate their performance in the presence of uncertainty (Juan et al. 2015). A notable application of this approach is provided by Albareda-Sambola et al. (2017), where the authors address a facility location-allocation problem with Bernoulli demand uncertainty. However, high-quality deterministic solutions in scheduling problems characterized by temporal uncertainty are typically tightly coupled where the execution time of each task is strongly dependent on the timing of preceding and succeeding tasks. Even minor delays or disruptions can cascade throughout the schedule, significantly undermining both feasibility and overall performance. To more effectively account for uncertainty, some studies integrate simulation directly into the local search process, evaluating each neighborhood solution under a set of stochastic scenarios to identify robust solutions to the stochastic version of the problem (Deblaere et al. 2011, Goodson et al. 2013). While this integration can improve solution quality under uncertainty, the substantial computational cost associated with repeatedly simulating neighborhood solutions, even under a modest number of scenarios, can severely limit algorithmic efficiency.

In this paper, we study a stochastic cross-docking scheduling problem (SCDSP) that incorporates flexible dock occupancy based on actual processing durations and accounts for uncertainty in truck arrival times. To address the challenge caused by the new features, we propose a novel simheuristic algorithm that draws inspiration from the simulation-based descent approach (SBD) (Deblaere et al. 2011) and the hybrid genetic search (HGS) (Vidal et al. 2012). A comprehensive computational study is conducted, and the results demonstrate the algorithm's efficiency and its ability to deliver high-quality schedules. The main contributions of this work are summarized as follows:

- We study the stochastic SCDSP by integrating truck sequencing and dock assignment. Our model captures the precise, continuous-time coordination among unloading completion, internal transfer, and outbound loading start times, specifically accounting for the temporal volatility of stochastic truck

arrivals. To manage operational stability under uncertainty, we propose a fixed-sequence and release-time policy. This policy generates adaptive real-time schedules aimed at minimizing total expected costs, including penalties for untransferred goods and tardiness costs incurred by deviations from expected departure targets.

- To address the integrated optimization challenge, we develop hybrid genetic simheuristic (SimHGS) framework that combines a population-based search with a nested simulation-based local search (NSBLS). Within this framework, a simulation-based local search (SBLS) is embedded in a variable neighborhood descent (VND) procedure. Specifically, the VND explores the solution space by optimizing truck sequences in a deterministic setting, while the SBLS refines the corresponding truck release times under stochastic realizations. Unlike traditional simheuristics that simulate only deterministic local optima, SimHGS evaluates a broader range of candidate solutions under uncertainty to enhance both robustness and search diversity.
- We propose a fast evaluation technique for scenario sets to mitigate the computational burden of SBLS. By analyzing the recursive structure of the SCDSP, we establish two fundamental theorems that identify trucks and scenarios unaffected by local time adjustments. This allows for the incremental calculation of cost gains, bypassing redundant simulations and achieving speedups of 1–2 orders of magnitude (up to 700 times) for large-scale instances.
- Comprehensive experiments on 72 benchmark instances demonstrate that SimHGS outperforms state-of-the-art algorithms for 93%-97% of instances. Furthermore, we validate the fixed-sequence and release-time policy through sensitivity and ablation analyses. The results confirm the policy's robustness across varying uncertainty levels and quantify the significant performance gains contributed by the SBLS refinement and the hybrid genetic framework.

The remainder of this paper is organized as follows. Section 2 describes the stochastic cross-docking problem settings, while Section 3 details the proposed fixed-sequence and release-time policy. Section 4 presents the NSBLS and the fast evaluation technique, which form the core of our solution approach. The overall SimHGS framework is then described in Section 5. Computational experiments and comparative analyses are reported in Section 6, and Section 7 concludes the paper with final remarks.

2. Problem description

Consider a rectangular cross-docking terminal with multiple docking doors, consisting of inbound doors $\mathcal{D}^I$ and outbound doors $\mathcal{D}^O$. The set of inbound trucks is denoted by $\mathcal{I}$, and the set of outbound trucks by $\mathcal{O}$. Let $\mathcal{F} = (f_{ij})$ be an $|\mathcal{I}| \times |\mathcal{O}|$ matrix that represents the flow of goods from inbound trucks to outbound trucks, where f_{ij} indicates the quantity of goods transferred from inbound truck $i \in \mathcal{I}$ to outbound truck $j \in \mathcal{O}$. In addition, let $\mathcal{H} = (h_{kl})$ be a $|\mathcal{D}^I| \times |\mathcal{D}^O|$ matrix, where h_{kl} denotes the distance between inbound door $k \in \mathcal{D}^I$ and outbound door $l \in \mathcal{D}^O$. Each truck $i \in \mathcal{I} \cup \mathcal{O}$ is associated with three time parameters (a_i, e_i, d_i),

where a_i is the stochastic arrival time following a given distribution, e_i is the expected departure time, and d_i is the latest permissible departure time. Each truck also has a processing time g_i, which represents the time required to complete the unloading or loading of all its goods.

The transfer time of goods depends on both the distance between dock doors and the quantity of goods being transferred (Nassief et al. 2018). Forklifts, with constant speed v and maximum load capacity Q, are used to transport goods inside the terminal. Let $t_{kl} = h_{kl}/v$ denote the travel time of a forklift from inbound door k to outbound door l. For each goods flow f_{ij}, let $q_{ij} = \lceil f_{ij}/Q \rceil$ denote the number of forklift trips required to move the goods from inbound truck i to outbound truck j. Hence, the total transfer time of flow f_{ij} from door k to door l is $q_{ij}t_{kl}$.

When a truck arrives at the terminal, it may not immediately proceed to its assigned dock door. To alleviate congestion at the dock doors, arriving trucks may wait in the yard until their scheduled start times. For inbound trucks, the operational decision is to assign an inbound door and determine the unloading sequence at that door. Once unloaded, their goods are sorted and transferred to the corresponding outbound doors. For outbound trucks, the decision includes door assignment, processing sequence, and the time when loading starts.

This study makes the following assumptions: (1) Goods flows $\mathcal{F}$ are pre-allocated and are not treated as decision variables; (2) each truck is handled in a single, uninterrupted loading or unloading operation and departs immediately upon completion; (3) each dock door handles at most one truck at a time; (4) sufficient internal staging capacity is available; (5) outbound loading is modeled as a single operation: only goods that have arrived at the assigned outbound door before the start of the corresponding loading operation can be loaded. Partial or overlapping loading is prohibited. This assumption ensures that the required loading time remains a binding operational constraint, rather than being implicitly circumvented by late-arriving goods. Goods that do not meet this condition are considered untransferred; (6) a truck is considered unserved if its service cannot be completed by its latest departure time. In such cases, all goods flows associated with that truck are classified as untransferred. All untransferred goods incur a unit penalty cost p_{ij}, representing the cost of either temporary storage within the cross-dock or outsourced emergency services.

Rather than relying on a deterministic schedule, the execution of an SCDSP instance is governed by a general scheduling policy Π. In this approach, truck operations are modeled as a multi-stage decision-making process, where the scheduling policy dynamically determines which truck to process. Formally, the policy Π can be defined as a function $\Pi : \mathbb{R}^{|\mathcal{I}|+|\mathcal{O}|} \longrightarrow \{1, \ldots, |\mathcal{D}^{\mathcal{I}}| + |\mathcal{D}^{\mathcal{O}}|\}^{|\mathcal{I}|+|\mathcal{O}|} \times \mathbb{R}^{|\mathcal{I}|+|\mathcal{O}|}$. It maps a vector of stochastic truck arrival times $\boldsymbol{a} = (a_1, a_2, \ldots, a_{|\mathcal{I}|+|\mathcal{O}|})$ to an ordered pair of output vectors $(\boldsymbol{r}, \boldsymbol{s})$, where $\boldsymbol{r} = (r_1, r_2, \ldots, r_{|\mathcal{I}|+|\mathcal{O}|})$ denotes the corresponding dock door assignments, with each $r_i \in \{1, \ldots, |\mathcal{D}^{\mathcal{I}}| + |\mathcal{D}^{\mathcal{O}}|\}$, and with $\boldsymbol{s} = (s_1, s_2, \ldots, s_{|\mathcal{I}|+|\mathcal{O}|})$ representing the realized start times without overlaps.

Under a realized execution $(\boldsymbol{r}, \boldsymbol{s})$, a truck $i \in \mathcal{I}$ or truck $j \in \mathcal{O}$ fails to be served within the planning horizon if $s_i + g_i > d_i$ or $s_j + g_j > d_j$, respectively. In this case, all goods associated with that truck

incur penalty costs. Likewise, if $s_j < s_i + g_i + q_{ij}t_{r_ir_j}$, then goods flow f_{ij} cannot be transferred from inbound truck i to outbound truck j in time for the loading operation of truck j, and it is therefore treated as untransferred. Let η_{ij} denote the penalty cost associated with goods flow f_{ij}, defined by

$$\eta_{ij} = \begin{cases} p_{ij}f_{ij}, & \text{if } s_i + g_i > d_i \text{ or } s_j + g_j > d_j \text{ or } s_j < s_i + g_i + q_{ij}t_{r_ir_j}, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The *policy execution cost* $\mathcal{C}$ of a scheduling policy Π is then defined as

$$\mathcal{C}(\Pi) = \mathbb{E}\left(c\left[\sum_{i\in\mathcal{I}} \max(0, s_i + g_i - e_i) + \sum_{j\in\mathcal{O}} \max(0, s_j + g_j - e_j)\right] + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{O}} \eta_{ij} \right). \tag{2}$$

The first two terms represent the tardiness costs of trucks relative to their expected departure times and are used to capture driver satisfaction. Here, c denotes the tardiness cost per unit time. The last term represents the penalty for goods that cannot be successfully transferred and loaded within the planning horizon. The objective of SCDSP is to determine a scheduling policy Π^* that minimizes $\mathcal{C}(\Pi)$ under stochastic truck arrivals while satisfying the time window constraints and the non-overlap requirement at dock doors.

3. Fixed-sequence and release-time policy

This section introduces the fixed-sequence and release-time policy adopted for the SCDSP. Fixing truck sequences enables anticipative goods transfers: once an inbound truck is unloaded, its goods can be moved immediately to the outbound door assigned to the corresponding outbound truck, even before that outbound truck arrives. This reduces the waiting of outbound trucks at dock doors. To accommodate uncertainties in truck arrivals, release times are further employed. A release time specifies the earliest time an outbound truck can start loading. By setting release times appropriately, loading starts can be deferred until the corresponding goods are likely to have arrived, thereby improving transfer completion under uncertainty. Although less flexible than a fully dynamic scheduling policy, this approach offers a practical balance between operational implementability and decision adaptability under uncertainty. Specifically, Section 3.1 describes the policy representation, and Section 3.2 derives structural properties of scenario-based policy execution that provide the basis for the fast evaluation technique developed later.

3.1. Policy representation

We denote a scheduling policy by a pair $\Pi = (\boldsymbol{\pi}, \boldsymbol{\tau})$, where $\boldsymbol{\pi} = (\boldsymbol{\pi}_u)_{u\in\mathcal{D}^I\cup\mathcal{D}^O}$ is a family of truck sequences, with one sequence π_u assigned to each dock door u. For each door u, $\boldsymbol{\pi}_u = (\pi_{u,1}, \ldots, \pi_{u,m_u})$ lists the m_u trucks assigned to that door, where m_u may vary across doors. When distinguishing inbound from outbound doors, we use indices k and l (instead of u) for inbound and outbound doors, respectively. The vector $\boldsymbol{\tau} = \left(\tau_1, \tau_2, \ldots, \tau_{|\mathcal{I}|+|\mathcal{O}|}\right)$ denotes the release times of all trucks. Under the fixed sequences $\boldsymbol{\pi}$, each dock door processes trucks in a predetermined order. The release time vector $\boldsymbol{\tau}$ requires that the realized start time of a truck cannot be earlier than its release time. In our setting, release times are mainly used to control the loading starts of outbound trucks under uncertain arrivals, so that loading can be postponed when necessary to wait for additional inbound goods.

Definition 1 (Release Time rule, RT) *For outbound truck j_k in sequence $\boldsymbol{\pi}_l$ ($l \in \mathcal{D}^O$), its start time based on the release time is computed as:*

$$s_{j_k} = \max\{a_{j_k}, \tau_{j_k}, \min\{s_{j_{k-1}} + g_{j_{k-1}}, d_{j_{k-1}}\}\} \tag{3}$$

with boundary condition $s_{j_1} = \max\{a_{j_1}, \tau_{j_1}\}$ for the first truck j_1 in sequence.

Definition 2 (Earliest Start Time rule, EST) *For a truck ν_k in sequence $\boldsymbol{\pi}_u$ ($u \in \mathcal{D}^I \cup \mathcal{D}^O$), its earliest start time is computed as $ES_{\nu_k} = \max\left\{a_{\nu_k}, \min\{ES_{\nu_{k-1}} + g_{\nu_{k-1}}, d_{\nu_{k-1}}\}\right\}$, with boundary condition $ES_{\nu_1} = a_{\nu_1}$ for the first truck ν_1 in sequence.*

Proposition 1 *For any fixed truck sequences $\boldsymbol{\pi}$, the EST rule yields the optimal start time for inbound trucks under the given sequences.*

The proof is shown in Online Supplement OS.1.

Definition 3 (Latest Start Time rule, LST) *For outbound truck j_k in sequence $\boldsymbol{\pi}_l$ ($l \in \mathcal{D}^O$), its latest start time is computed as $LS_{j_k} = \min\left\{d_{j_k} - g_{j_k}, LS_{j_{k+1}} - g_{j_k}\right\}$, with boundary condition $LS_{j_{m_l}} = d_{j_{m_l}} - g_{j_{m_l}}$ for the last truck j_{m_l} in sequence, where m_l is the total number of trucks assigned to door l.*

The RT rule determines the realized loading starts of outbound trucks under a given release time vector. The EST rule computes the earliest feasible start times under fixed truck sequences and is used to determine inbound truck start times. The LST rule computes the latest feasible start times for loading outbound trucks. By maximizing the amount of goods transferred, the LST rule minimizes penalty costs associated with incomplete transfers, and its resulting start times are used to construct the initial release time vector.

Let $\Omega = \{\hat{\boldsymbol{a}}^1, \hat{\boldsymbol{a}}^2, \ldots, \hat{\boldsymbol{a}}^\omega, \ldots, \hat{\boldsymbol{a}}^{|\Omega|}\}$ denote the scenario set, where each $\hat{\boldsymbol{a}}^\omega$ is a possible realization of the random truck arrival time vector. The *estimated policy execution cost* of a scheduling policy Π under the scenario set Ω is defined via sample average evaluation as:

$$\mathcal{C}(\Pi, \Omega) = \frac{1}{|\Omega|}\left(c\left[\sum_{\omega\in\Omega}\sum_{i\in\mathcal{I}}\max(0, \hat{s}_{i\omega} + g_i - e_i) + \sum_{\omega\in\Omega}\sum_{j\in\mathcal{O}}\max(0, \hat{s}_{j\omega} + g_j - e_j)\right] + \sum_{\omega\in\Omega}\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{O}}\hat{\eta}_{ij\omega}\right) \tag{4}$$

where, for each scenario ω, $\hat{s}_{i\omega}$ $(i \in \mathcal{I})$ and $\hat{s}_{j\omega}$ $(j \in \mathcal{O})$ denote the realized start times of inbound and outbound trucks, respectively, obtained by applying the EST rule and the RT rule in Eq. (3) with the arrival time realization $\hat{\boldsymbol{a}}^\omega$ substituted for $\boldsymbol{a}$. The term $\hat{\eta}_{ij\omega}$ represents the penalty cost of goods flow f_{ij} under scenario ω, computed from Eq. (1) using $\hat{s}_{i\omega}$ and $\hat{s}_{j\omega}$.

Figure 1 illustrates a simple example with 2 inbound doors and 2 outbound doors, 3 inbound trucks (1, 2, 3), 3 outbound trucks (4, 5, 6) with processing sequence $\boldsymbol{\pi}_1 = (3), \boldsymbol{\pi}_2 = (1, 2), \boldsymbol{\pi}_3 = (6), \boldsymbol{\pi}_4 = (4, 5)$. The release time vector is $\boldsymbol{\tau} = \{-, -, -, 23, 34, 30\}$, where the first three entries correspond to inbound trucks; their release times are omitted since the optimal start time can be determined by the EST rule. To keep the figure concise, truck processing times and internal transfer times are omitted; only the uncertain

arrival intervals of inbound and outbound trucks, together with the induced uncertain arrival intervals of goods at the outbound doors, are shown. Specifically, τ_4 is set to the latest possible arrival time of flow f_{24}, thereby ensuring that all goods assigned to outbound truck 4 (f_{14}, f_{24}) are available before loading starts. Similarly, $\tau_5 = 34$ corresponds to the latest possible arrival time of flow f_{35}, so that all goods assigned to outbound truck 5 (f_{15}, f_{35}) can be transferred in time. In contrast, outbound truck 6 cannot wait until the latest possible arrival of all its assigned goods, because its release time is capped by the latest feasible start time, i.e., $\tau_6 \leq d_6 - g_6 = 30$. Therefore, τ_6 is set to 30, and any goods arriving during $[30, 32]$ cannot be transferred to truck 6. This example illustrates how setting release times helps control the realized start times of outbound trucks, with the goal of maximizing the expected goods transfer and driver satisfaction.

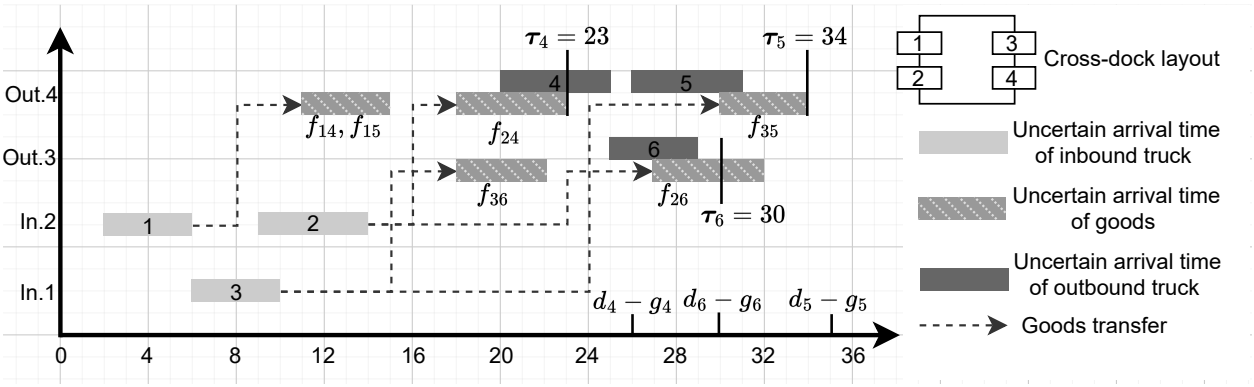


Figure 1 **Illustrative SCDSP example under the fixed-sequence and release-time policy**

3.2. Structural properties of scenario-based policy execution

This subsection studies the structural properties of scenario-based policy execution under the fixed-sequence and release-time policy. It first defines the notion of reasonable release times, then derives an equivalent expression for the realized start times of outbound trucks under reasonable release times, and shows that any release-time vector can be transformed into an equivalent reasonable one. These results provide the theoretical foundation for the fast evaluation mechanism developed later.

Definition 4 (Reasonable Release Time) *A release time $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_{|\mathcal{I}\cup\mathcal{O}|})$ is said to be* **reasonable** *if for any two consecutive outbound trucks $j' \prec j$, the condition $\tau_j \geq \min\{\tau_{j'} + g_{j'}, d_{j'}\}$ holds.*

Let Θ denote the set of all possible release time vectors $\boldsymbol{\tau}$, and let $\Theta_r \subseteq \Theta$ be the subset of reasonable release time vectors. Let ES^O be a $|\mathcal{O}| \times |\Omega|$ matrix, where each entry $ES^O_{j\omega}$ denotes the earliest start time of outbound truck j under scenario ω, obtained by applying the EST rule to the corresponding outbound truck sequence with the arrival time realization $\hat{\boldsymbol{a}}^\omega$.

Theorem 1 *For any scheduling policy $\Pi = (\boldsymbol{\pi}, \boldsymbol{\tau})$ with $\boldsymbol{\tau} \in \Theta_r$, the realized start time of outbound truck j under scenario ω computed using Eq. (3) can be equivalently computed as:*

$$\hat{s}_{j\omega} = \max\{ES^O_{j\omega}, \tau_j\}. \tag{5}$$

The proof of this theorem is given in the Online Supplement OS.1.

Theorem 1 establishes that under the reasonable release time constraints, the complex recursive relationship in Eq. (3) reduces to a form that depends only on two fundamental parameters: (1) the scenario-specific earliest start time $ES^{O}_{j\omega}$, and (2) the reasonable release time τ_j. Next, we present a method for transforming an unreasonable release time into a reasonable one and prove that they can equivalently generate the same realized start time $\hat{\boldsymbol{s}}$, as shown in Theorem 2.

Theorem 2 *For any scheduling policy* $\Pi = (\boldsymbol{\pi}, \boldsymbol{\tau})$ *with* $\boldsymbol{\tau} \in \Theta \setminus \Theta_r$*, there exists a reasonable* $\boldsymbol{\tau}' \in \Theta_r$ *such that:*

(1) for the first truck j_1 *in each outbound door:* $\tau'_{j_1} = \tau_{j_1}$ *and*

(2) for subsequent trucks j_k $(k \geq 2)$*:* $\tau'_{j_k} = \max\left\{\min\{\tau'_{j_{k-1}} + g_{j_{k-1}}, d_{j_{k-1}}\}, \tau_{j_k}\right\}$.

The resulting $\boldsymbol{\tau}'$ *and* $\boldsymbol{\tau}$ *can generate the same realized start time* $\hat{\boldsymbol{s}}$ *under* Π.

The proof is shown in Online Supplement OS.1.

For any $\boldsymbol{\tau} \in \Theta$, the realized start times of all outbound trucks can be determined analytically using an equivalent reasonable release time vector, without explicitly simulating the execution process. This observation serves as a key structural basis for the fast evaluation mechanism introduced later.

4. Nested simulation-based local search

In this section, we propose a NSBLS algorithm. NSBLS operates on two levels. At the outer level, the search alternates between improving inbound-related and outbound-related decisions. At the inner level, a VND procedure explores the corresponding neighborhood structures. Whenever a sequence improvement is sufficiently promising under the deterministic evaluation, SBLS is invoked to refine the associated release time vector under uncertainty. In what follows, Section 4.1 presents the main scheme of NSBLS. Section 4.2 describes the VND procedure for optimizing truck sequences. Section 4.3 introduces the deterministic local search for release-time improvement. Section 4.4 further extends this search to the stochastic setting and presents the fast evaluation technique.

4.1. Main scheme of NSBLS

Algorithm 1 presents the NSBLS framework. It alternates between inbound and outbound side: an outer loop manages search domain alternation via the OPPOSITE operator (Line 3-23), while an inner loop performs intensive neighborhood exploration on single side (Line 6-22, VND loop). The neighborhood structure $\mathcal{N}$ contains 12 distinct neighborhoods derived from six neighborhood operators (Section 4.2), partitioned into inbound ($\text{InSet} = \{\mathcal{N}_1, \ldots, \mathcal{N}_6\}$) and outbound ($\text{OutSet} = \{\mathcal{N}_7, \ldots, \mathcal{N}_{12}\}$) subsets. Each neighborhood $\mathcal{N}_{cur}$ undergoes iterative exploration until no improvements are detected. In each iteration, a neighborhood operator is applied to the current solution to generate a new truck sequences $\boldsymbol{\pi}'$, after which the corresponding release time $\boldsymbol{\tau}'$ is constructed using default scheduling rules (EST rule for inbound trucks and LST rule

Algorithm 1 NSBLS$(\Pi, \Omega, DetLB)$

Input: Current policy $\Pi = (\boldsymbol{\pi}, \boldsymbol{\tau})$, scenario set Ω, local optimum $DetLB$
Output: Improved policy Π^{LB} with execution cost $\mathcal{C}(\Pi^{LB}, \Omega)$

1: $\Pi^{LB} \leftarrow \Pi$, InSet $= \{\mathcal{N}_1, \ldots, \mathcal{N}_6\}$, OutSet $= \{\mathcal{N}_7, \ldots, \mathcal{N}_{12}\}$
2: $Imp \leftarrow true$, $currentSide \leftarrow$ InSet
3: **while** Imp **do** ▷ *Alternating search*
4: $\quad Imp \leftarrow false$, $sideImp \leftarrow true$
5: $\quad$ **while** $sideImp$ **do** ▷ *Single-side intensification*
6: $\quad\quad sideImp \leftarrow false$
7: $\quad\quad$ **for** $\mathcal{N}_{cur} \in currentSide$ **do** ▷ *Neighborhood exploration (Sec. 4.2)*
8: $\quad\quad\quad$ **if** $\exists$ improving $move$ in $\mathcal{N}_{cur}$ **then**
9: $\quad\quad\quad\quad \boldsymbol{\pi}' \leftarrow \boldsymbol{\pi} \oplus move$
10: $\quad\quad\quad\quad \boldsymbol{\tau}' \leftarrow$ DEFAULTRULE$(\boldsymbol{\pi}')$
11: $\quad\quad\quad\quad$ **if** $\beta \cdot DetC(\boldsymbol{\pi}', \boldsymbol{\tau}') \leq DetLB + MaxOT$ **then** ▷ *Cutoff rule for promising $\boldsymbol{\pi}'$*
12: $\quad\quad\quad\quad\quad \boldsymbol{\tau}' \leftarrow$ LOCALSEARCH$(\boldsymbol{\tau}')$ ▷ *Release time improvement (Sec. 4.3)*
13: $\quad\quad\quad\quad\quad$ **if** $DetC(\boldsymbol{\pi}', \boldsymbol{\tau}') < DetLB$ **then**
14: $\quad\quad\quad\quad\quad\quad DetLB \leftarrow DetC(\boldsymbol{\pi}', \boldsymbol{\tau}')$
15: $\quad\quad\quad\quad\quad$ **if** $DetC(\boldsymbol{\pi}', \boldsymbol{\tau}') \leq \mathcal{C}(\Pi^{LB}, \Omega)$ **then** ▷ *Promising policy $(\boldsymbol{\pi}', \boldsymbol{\tau}')$*
16: $\quad\quad\quad\quad\quad\quad \Pi \leftarrow$ SBLS$(\boldsymbol{\pi}', \boldsymbol{\tau}', \Omega)$ ▷ *SBLS refinement (Sec. 4.4)*
17: $\quad\quad\quad\quad\quad\quad$ **if** $\mathcal{C}(\Pi, \Omega) < \mathcal{C}(\Pi^{LB}, \Omega)$ **then**
18: $\quad\quad\quad\quad\quad\quad\quad \Pi^{LB} \leftarrow \Pi$
19: $\quad\quad\quad\quad sideImp \leftarrow true$, $Imp \leftarrow true$
20: $\quad\quad\quad\quad$ **break** ▷ *Switch to neighborhood N_1*
21: $\quad currentSide \leftarrow$ OPPOSITE$(currentSide)$
22: **return** $\Pi^{LB}, \mathcal{C}(\Pi^{LB}, \Omega)$

for outbound trucks). This produces a candidate neighborhood solution $(\boldsymbol{\pi}', \boldsymbol{\tau}')$ along with its associated objective value $DetC(\boldsymbol{\pi}', \boldsymbol{\tau}')$.

Specifically, our NSBLS algorithm first initializes the locally optimal scheduling policy Π^{LB} and the subsets InSet/OutSet (Line 1). An improvement flag Imp is employed to track whether an improved solution has been found in the current side, and a single-side improvement flag $sideImp$ controls the local search. When an improving move is identified, the cutoff rule $\beta \cdot DetC(\boldsymbol{\pi}', \boldsymbol{\tau}') \leq DetLB + MaxOT$ evaluates whether the truck sequences $\boldsymbol{\pi}$ are sufficiently promising, where $MaxOT = \sum_{j \in \mathcal{O}}(d_j - e_j)$ represents the maximum tardiness improvement and $DetLB$ denotes the local optimum of the deterministic problem (Line 11). This rule assumes that if the deterministic objective value of the current solution $(\boldsymbol{\pi}', \boldsymbol{\tau}')$ is worse than the local optimum plus the possible maximum improvement, a local improvement of $\boldsymbol{\tau}'$ cannot yield a solution better than $DetLB$, thus the deterministic local search on $\boldsymbol{\tau}'$ is abandoned. The scaling factor β prevents the cutoff rule from being too strict. If $\boldsymbol{\pi}'$ is identified as promising, a local search based on the time adjustment operator (Section 4.3) is used to obtain the local optimal release time $\boldsymbol{\tau}'$ of the deterministic version under the current $\boldsymbol{\pi}'$ (Line 12).

The deterministic local search improves $\boldsymbol{\tau}'$ for two purposes. First, it updates the local optimum $DetLB$ through $DetC(\boldsymbol{\pi}', \boldsymbol{\tau}')$, thereby tightening the constraint of the cutoff rule. Second, $DetC(\boldsymbol{\pi}', \boldsymbol{\tau}')$ serve as an approximate lower bound of solution $(\boldsymbol{\pi}', \boldsymbol{\tau}')$ in the stochastic context, providing a basis for evaluating its potential under uncertainty. Specifically, Line 15 checks whether the obtained solution $(\boldsymbol{\pi}', \boldsymbol{\tau}')$ is promising in the stochastic setting. If so, SBLS is invoked to find the locally optimal scheduling policy under the stochastic settings (Line 16). Lines 17 and 18 handle the update of the best-known local scheduling policy Π^{LB}. When the current VND completes its exploration, the search domain is switched to the alternative domain (Line 21). The algorithm terminates when all neighborhood explorations fail to yield further improvements, returning the local optimal scheduling policy Π^{LB}.

4.2. Variable neighborhood descent to optimize truck sequences

Under the policy representation adopted in this study, each dock door is associated with an ordered list of trucks. Therefore, sequence optimization in the present problem is structurally similar to route improvement in vehicle routing, where each route is also represented as an ordered sequence. Following Vidal et al. (2012) and Máximo et al. (2024), we adapt six classical neighborhood operators to modify the truck sequences $\boldsymbol{\pi}$.

- *relocate1(u, v)*: Relocate truck u to a position immediately after truck v.
- *relocate2(u, v)*: Relocate two consecutive trucks u and x to a position immediately after truck v.
- *relocate3(u, v)*: Relocate two consecutive trucks u and x in reverse order to a position immediately after truck v.
- *swap1(u, v)*: Swap trucks u and v.
- *swap2(u, v)*: Swap two consecutive trucks u and x with truck v.
- *swap3(u, v)*: Swap two consecutive trucks u and x with two consecutive trucks v and y.

All neighborhood operators can be applied within the same door or across different doors. To ensure feasible solutions, trucks handled by an operator must belong to the same side, yielding the inbound side neighborhoods subset $\mathrm{InSet} = \{\mathcal{N}_1, \ldots, \mathcal{N}_6\}$ and the outbound side neighborhoods subset $\mathrm{OutSet} = \{\mathcal{N}_7, \ldots, \mathcal{N}_{12}\}$. We explore their neighborhoods following the sequence of operators described above. To assess the cost improvement of these operators, the EST and LST rules are applied as default methods to determine the release times of inbound and outbound trucks, respectively. The use of LST here aims to quickly determine the release times $\boldsymbol{\tau}$ without resorting to local search, in order to improve the computational efficiency of the neighborhood evaluation mentioned above.

4.3. Local search to improve release times

The release times $\boldsymbol{\tau}$ of the outbound trucks, initially obtained using the LST rule, are designed to maximize goods transfer without considering truck tardiness costs. To address tardiness, we refine these release times

using a time adjustment operator aimed at minimizing tardiness costs in the deterministic setting. Specifically, for each outbound truck $j \in \mathcal{O}$, two neighboring solutions are generated by increasing or decreasing its release time τ_j by Δt time units, where the Δt was set to 1. As a result, the neighborhood of the current solution contains up to $2|\mathcal{O}|$ alternatives. In each iteration, the best improvement strategy is applied, and the search continues until no further improvements are possible.

4.4. Simulation-based local search to improve release times

The SBLS is used to improve the release times $\boldsymbol{\tau}$ while fixing the truck sequence $\boldsymbol{\pi}$ in the stochastic setting, using the same time adjustment operator as in the deterministic local search. The policy execution cost is decomposable into truck tardiness cost and penalty costs for untransferred goods. Under a time adjustment move, only a subset of outbound trucks experience changes in their realized start times, so only the corresponding tardiness terms and transfer-penalty terms for the affected truck–scenario combinations need to be updated (See Section 4.4.1). Building on the structural properties established in Section 3.2, we develop a fast evaluation technique for SBLS that avoids recomputing the full policy execution cost after each move in Section 4.4.2. Then Section 4.4.3 integrates this technique into the SBLS procedure.

4.4.1. Affected trucks and scenarios Given a *reasonable* release time $\boldsymbol{\tau}$ and a *reasonable* modified version $\boldsymbol{\tau}'$ that differs only by adjusting the release time of a specific truck v by Δt units, the set of trucks whose realized start times are affected can be identified as

$$\Phi = \{j | \tau'_j \neq \tau_j, j \in \mathcal{O}\}. \tag{6}$$

Define the following scenario set for each outbound truck j

$$\Psi_j(\tau_j) = \{\omega | ES^O_{j\omega} < \tau_j, \omega \in \Omega\}, \tag{7}$$

which identifies the scenarios where the realized start time $\hat{s}_{j\omega}$ must be updated to τ_j. Then the realized start time $\hat{s}_{j\omega}$ can be represented as:

$$\hat{s}_{j\omega} = \begin{cases} \tau_j, & \omega \in \Psi_j(\tau_j), \\ ES^O_{j\omega}, & \omega \in \Omega \setminus \Psi_j(\tau_j). \end{cases} \tag{8}$$

Evaluating Eq. (7) directly requires scanning all scenarios. To avoid this, we preprocess the earliest start times of each outbound truck. For truck $j \in \mathcal{O}$, let $\boldsymbol{\delta}_j$ be the ordered vector obtained by sorting $ES^O_j = \{ES^O_{j1}, \ldots, ES^O_{j|\Omega|}\}$ in non-increasing order, where ties are broken by placing smaller scenario index first. Thus, δ_{jk} denotes the k-th largest earliest start time of truck j, where $k \in 1, \ldots, |\Omega|$. Additionally, let $\boldsymbol{\xi}_j$ be the *scenario index vector* associated with $\boldsymbol{\delta}_j$, i.e., ξ_{jk} is the scenario index of earliest start time δ_{jk}. Finally, let $\boldsymbol{\sigma}_j$ be the *position mapping vector* that represents the position of the earliest starting time $ES^O_{j\omega}$ in the ordered vector $\boldsymbol{\delta}_j$. When multiple positions have the same earliest starting time, the largest position is returned, i.e., σ_{jt} is the position of the earliest start time t in $\boldsymbol{\delta}_j$ for truck $j \in \mathcal{O}$.

Based on this construction, the following Corollary 1 is derived.

Corollary 1 *For any reasonable release time vector $\boldsymbol{\tau} \in \Theta_r$, the subset of scenarios requiring updates to the realized start time for truck $j \in \mathcal{O}$ can be formally characterized by:*

$$\Psi_j(\tau_j) = \{\xi_{jk} | \sigma_{j\tau_j} + 1 \leq k \leq |\Omega|\} \tag{9}$$

Table 1 Small example illustrating Corollary 1

Numbered scenario k	1	2	3	4	5	Interpretation
Original ES_j^ω	18	21	19	23	20	$\omega = 1,2,3,4,5$
δ_{jk}	23	21	20	19	18	Sorted ES_j^ω
ξ_{jk}	4	2	5	3	1	Scenario index after sorting
$\hat{s}_j^\omega$ under $\tau_j = 20$	23	21	20	**20**	**20**	$\sigma_{j20} = 3,\ \Psi_j(20) = \{\xi_{j4}, \xi_{j5}\} = \{3,1\}$
$\hat{s}_j^\omega$ under $\tau_j = 21$	23	21	**21**	**21**	**21**	$\sigma_{j21} = 2,\ \Psi_j(21) = \{\xi_{j3}, \xi_{j4}, \xi_{j5}\} = \{5,3,1\}$

Corollary 1 avoids explicit enumeration of all scenarios by locating the critical suffix in the ordered vector $\boldsymbol{\delta}_j$. Hence, given $\boldsymbol{\delta}_j$, $\boldsymbol{\xi}_j$, and $\boldsymbol{\sigma}_j$, the affected scenarios can be extracted directly. Table 1 presents a small example for the fast update mechanism based on Corollary 1. Suppose the earliest start times of outbound truck j under five scenarios are $(ES_j^1, ES_j^2, ES_j^3, ES_j^4, ES_j^5) = (18, 21, 19, 23, 20)$. Sorting them in non-increasing order gives $\boldsymbol{\delta}_j = (23, 21, 20, 19, 18)$, with corresponding scenario-index vector $\boldsymbol{\xi}_j = (4, 2, 5, 3, 1)$. When $\tau_j = 20$, we have $\sigma_{j20} = 3$, and thus $\Psi_j(20) = \{\xi_{j4}, \xi_{j5}\} = \{3, 1\}$. Therefore, only the scenarios in $\{3, 1\}$ are truncated by the release time, and their realized start times become 20. When the release time increases to $\tau_j = 21$, we obtain $\sigma_{j21} = 2$ and $\Psi_j(21) = \{\xi_{j3}, \xi_{j4}, \xi_{j5}\} = \{5, 3, 1\}$. In this case, the realized start times of all scenarios in $\Psi_j(21)$ become 21. Hence, the local objective contributions associated with all scenarios in $\{5, 3, 1\}$ must be re-evaluated, because their realized start times under $\tau_j = 21$ differ from those under $\tau_j = 20$.

4.4.2. The fast evaluation technique

For simplicity, time unit $\Delta t^+ > 0$ indicates a plus move, and $\Delta t^- < 0$ indicates a minus move. Given an initial reasonable release time $\boldsymbol{\tau}$, adding Δt^+ time units to an outbound truck $v \in \mathcal{O}$ updates its release time to $\tau_v' = \tau_v + \Delta t^+$. According to Theorem 2, the adjusted feasible release time $\boldsymbol{\tau}'$ is generated after the move. Subsequently, the set of trucks Φ whose realized start times are affected is identified. Based on Corollary 1, for each affected truck $j \in \Phi$, the corresponding scenario set $\Psi_j(\tau_j')$ is determined. In these scenarios, the realized start time is updated to $\hat{s}_{j\omega}' = \tau_j'$, while the start times of the other scenarios remain unchanged. The original realized start time of these scenarios with release time $\boldsymbol{\tau}$ are:

$$\hat{s}_{j\omega} = \begin{cases} \tau_j, & \omega \in \Psi_j(\tau_j), \\ ES_{j\omega}^O, & \omega \in \Psi_j(\tau_j') \setminus \Psi_j(\tau_j), \end{cases} \tag{10}$$

where $\Psi_j(\tau_j') \setminus \Psi_j(\tau_j) = \{\xi_{jk} | \boldsymbol{\sigma}_{j\tau_j'} \leq k \leq \boldsymbol{\sigma}_{j\tau_j} - 1\}$.

It is worth noting that $|\Delta t| > 1$ can be achieved by plus/minus time unit with length $|\Delta t| = 1$ multiple times. We use a time unit $|\Delta t| = 1$ for each move, which can also reduce the size of the neighborhood

explored each time. Let $\Delta\mathcal{C}_{v,\pm1}$ be the cost gain resulting from the change in realized start times of these scenarios. For simplicity, let $\mathcal{R}_j(t) = c \cdot \max(0, t + g_j - e_j)$ represent the tardiness cost of outbound truck $j \in \mathcal{O}$ with start time t, $\hat{\eta}'_{ij\omega}$ and $\hat{\eta}_{ij\omega}$ be the penalty cost of untransferred goods associated with release time τ'_j and τ_j computed using Eq. (1), respectively. Let $\Psi_j = \Psi_j(\max\{\tau_j, \tau'_j\})$ denote the set of scenarios in which the realized start time changes due to a $+\Delta t$ or $-\Delta t$ adjustment to the release time τ_j. When $|\Delta t| = 1$, the corresponding cost change $\Delta\mathcal{C}_{v,\pm1}$ can be computed as follows:

$$\Delta\mathcal{C}_{v,\pm1} = \frac{1}{|\Omega|}\left(\sum_{j\in\Phi}|\Psi_j|(\mathcal{R}_j(\tau'_j) - \mathcal{R}_j(\tau_j)) + \sum_{i\in\mathcal{I}}\sum_{j\in\Phi}\sum_{\omega\in\Psi_j}(\hat{\eta}'_{ij\omega} - \hat{\eta}_{ij\omega})\right). \tag{11}$$

The above formulation enables efficient fast evaluation with a computational complexity of $O(\sum_{i\in\mathcal{I}}\sum_{j\in\Phi}|\Psi_j|)$ instead of $O(|\mathcal{I}||\mathcal{O}||\Omega|)$.

Let $Cost^+$ and $Cost^-$ be $|\mathcal{O}|$-dimensional vectors recording the cost of plus and minus $\Delta t = 1$ time unit, respectively.

If a plus move is performed, $Cost^+$ and $Cost^-$ are updated as follows,

$$Cost^+_j = \begin{cases} Cost^+_j - Cost^+_v, & j \in \Gamma_a, \\ \Delta\mathcal{C}_{j,1}, & j \in \Phi, \end{cases} \tag{12}$$

$$Cost^-_j = \Delta\mathcal{C}_{j,-1}, \quad j \in \Phi, \tag{13}$$

where $\Gamma_a = \left\{i \,\middle|\, \tau_i + 1 + \sum_{i'\in\rho_i} g_{i'} = \tau_v,\ i \prec v\right\}$ denotes trucks for which plus $\Delta t = 1$ will not affect τ_v. Here, ρ_i is the set of trucks between i and v.

If a minus move is performed, the update operates as follows,

$$Cost^+_v = -Cost^-_v, \tag{14}$$

$$Cost^-_j = \Delta\mathcal{C}_{j,-1}, \quad j \in \Gamma_b \cup \{v\}, \tag{15}$$

where $\Gamma_b = \left\{j' \,\middle|\, \tau_v + 1 + \sum_{i'\in\rho_{j'}} g_{i'} = \tau_{j'},\ v \prec j'\right\}$ denotes trucks for which minus $\Delta t = 1$ will not affect $\tau_{j'}$. Here, $\rho_{j'}$ is the set of trucks between v and j'.

4.4.3. Simulation based local search The pseudocode of SBLS using the above fast evaluation technique is shown in Algorithm 2. Line 1 computes the earliest start times ES^I and ES^O of the inbound and outbound trucks respectively. Line 2 computes the objective value of scheduling policy Π under scenario set Ω using Eq.(4), where the realized start times of outbound trucks are determined by Theorem 1. This is followed by the calculation of the auxiliary data needed for the fast evaluation (line 3), as described in Section 4.4.2. Line 4 initializes the cost gain vectors $Cost^+$ and $Cost^-$ using Eq.(11). Lines 6 to 15 outline the local search procedure. A best improvement strategy is used to select moves (line 8). Line 11 performs the time adjustment move of Δt time unit if such an improved move is identified. Then line 12 updates the objective value and line 13 follows from Theorem 2 to obtain a reasonable $\boldsymbol{\tau}$. Then the cost gain vectors $Cost^+$ and $Cost^-$ are updated using Eq.(12),(13) and Eq.(14),(15) (line 14). When no improved move is found, the search ends and the local optimal scheduling policy $\Pi = (\boldsymbol{\pi}, \boldsymbol{\tau})$ found is returned.

Algorithm 2 SBLS$(\boldsymbol{\pi}, \boldsymbol{\tau}, \Omega)$

Input: policy $(\boldsymbol{\pi}, \boldsymbol{\tau})$ and scenario set Ω
Output: the local optimal policy $\Pi = (\boldsymbol{\pi}, \boldsymbol{\tau})$ and execution cost $\mathcal{C}(\Pi, \Omega)$
1: Compute the earliest start times ES^I and ES^O by EST and LST rules respectively
2: Compute $\mathcal{C}(\Pi, \Omega)$ using Eq. (4)
3: Compute $\boldsymbol{\delta}_j, \boldsymbol{\xi}_j$ and $\boldsymbol{\sigma}_j$ for each outbound truck $j \in \mathcal{O}$ ▷ *Sec. 4.4.1*
4: Initialize cost gain vectors $Cost^+$ and $Cost^-$ using Eq. (11)
5: $improved \leftarrow true$
6: **while** $improved$ **do**
7: $\quad improved \leftarrow false$
8: $\quad$ Find the pair $(slt, \Delta t)$ that minimizes the cost gain $\Delta\mathcal{C}_{j,\Delta t}$ from the cost vectors $Cost^+$ and $Cost^-$.
9: $\quad$ **if** $\Delta\mathcal{C}_{slt,\Delta t} < 0$ **then**
10: $\quad\quad improved \leftarrow true$
11: $\quad\quad \tau_{slt} \leftarrow \tau_{slt} + \Delta t$
12: $\quad\quad \mathcal{C}(\Pi, \Omega) \leftarrow \mathcal{C}(\Pi, \Omega) + \Delta\mathcal{C}_{slt,\Delta t}$
13: $\quad\quad$ Updated to *reasonable* $\boldsymbol{\tau}$ by Theorem 2
14: $\quad\quad$ Update cost gain vectors $Cost^+$ and $Cost^-$ ▷ *Eq.(12,13) and Eq.(14,15)*
15: **return** $\Pi = (\boldsymbol{\pi}, \boldsymbol{\tau})$ and $\mathcal{C}(\Pi, \Omega)$

5. Hybrid genetic search based simheuristic

Population-based evolutionary search has proven effective across a wide range of vehicle routing problems (Vidal et al. 2012, 2014, He et al. 2025). To overcome the local optima convergence issue observed in the NSBLS algorithm, we propose a SimHGS algorithm based on the hybrid genetic search framework (Vidal et al. 2012). The main scheme, population management, and crossover operators are detailed in Sections 5.1, 5.2, and 5.3, respectively.

5.1. Main scheme of SimHGS

The general SimHGS framework is presented in Algorithm 3. The scenario set Ω remains constant throughout the search process and is large enough to accurately estimate the policy's execution cost. The population $\mathcal{P}$ is initialized in line 3, and both the local optimal policy Π^{LB} and the best policy Π^* are initially set to the best individual in this population (line 4). In each generation of the evolutionary process, two parent policies Π_A and Π_B are selected via a binary tournament strategy (line 6). A crossover operator is then applied to generate an offspring policy Π (line 7), which is subsequently refined using the NSBLS algorithm to produce an improved local optimal policy Π^{LB} (line 8). The best policy Π^* and the population are updated accordingly (lines 9-10, 13). When the population size reaches the maximum threshold $(\mu + \lambda)$, it is reduced by retaining the top μ individuals based on fitness. If Π^* shows no improvement for a given number of consecutive generations, the current population is discarded and a new initial population is generated. The algorithm terminates and returns the best policy Π^* found once the stopping criterion is met.

Algorithm 3 The framework of SimHGS

Input: deterministic problem instance and a scenario set Ω of truck arrival time, minimum population size μ, maximum population size $\mu+\lambda$, population replacement threshold $Imax$
Output: the best policy $\Pi^* = (\boldsymbol{\pi}^*, \boldsymbol{\tau}^*)$ with policy execution cost $\mathcal{C}(\Pi^*, \Omega)$

$Iter \leftarrow 0$
$DetLB \leftarrow \infty$
$\mathcal{P} = \{\Pi_1, \Pi_2, ..., \Pi_\lambda\} \leftarrow$ POPGENERATE(μ) ▷ *Population initiation, Sec. 5.2*
$\Pi^*, \Pi^{LB} \leftarrow \arg\min\{\mathcal{C}(\Pi_i, \Omega) | \Pi_i \in \mathcal{P}\}$
while Time Limit is not reached **do**
  $\Pi_A, \Pi_B \leftarrow$ PARENTSELECT$(\mathcal{P})$
  $\Pi \leftarrow$ CROSSOVER(Π_A, Π_B) ▷ *Sec. 5.3*
  $\Pi^{LB} \leftarrow$ NSBLS$(\Pi, \Omega, DetLB)$ ▷ *Local improvement, Sec. 4.1*
  if $\mathcal{C}(\Pi^{LB}, \Omega) < \mathcal{C}(\Pi^*, \Omega)$ **then**
    $\Pi^* \leftarrow \Pi^{LB}, Iter \leftarrow 0$
  else
    $Iter \leftarrow Iter + 1$
  $\mathcal{P} \leftarrow$ POPUPDATE(Π^{LB}) ▷ *Population management, Sec. 5.2*
  if $|\mathcal{P}| >= \mu + \lambda$ **then**
    $\mathcal{P} \leftarrow$ POPREDUCTION(μ) ▷ *Population management, Sec. 5.2*
  if $Iter \geq Imax$ **then**
    $\mathcal{P} \leftarrow$ POPGENERATE(μ) ▷ *Population replacement, Sec. 5.2*
return Π^*

5.2. Population management

The population management is carried out in the initialization and updating phases. Population initialization proceeds via an iterative process: (1) randomly assign inbound/outbound trucks to respective doors, (2) sort trucks at each door by ascending arrival time to form initial sequence $\boldsymbol{\pi}$, (3) determine release times using the EST rule for inbound trucks and the LST rule for outbound trucks. The NSBLS algorithm then refines initial policy $\Pi = (\boldsymbol{\pi}, \boldsymbol{\tau})$ to produce local optima Π^{LB}. Each NSBLS optimized policy expands the population until it reaches the maximum size of $\mu + \lambda$, at which point a reduction process is triggered to retain the μ fittest individuals. Fitness is evaluated based on both solution quality and diversity contribution, following the approach in Vidal et al. (2012), computed as $fit(\Pi) = f^R(\Pi) + (1 - \frac{n_{elite}}{|\mathcal{P}|}) f^{DIV}(\Pi)$, where $f^R(\Pi)$ denotes solution quality rank and $f^{DIV}(\Pi)$ represents its rank in terms of diversity contribution. The diversity contribution is evaluated using the broken-pair distance, which quantifies the number of common truck sequence shared with its $n_{closest}$ most similar individuals in the population $\mathcal{P}$. The parameter n_{elite} ensures the preservation of elite individuals during selection. Population reinitialization is triggered if no improvement is observed over $Imax$ consecutive generations.

5.3. Crossover operators

Crossover is a key component to generate promising offspring solutions. First, parent solutions Π_A and Π_B are selected through binary tournament selection based on fitness. The truck assignment and sequence $\boldsymbol{\pi}$ is

decomposed into inbound and outbound door permutations (P^I, P^O) for each parent. An ordered crossover (OX) is then independently applied to the inbound and outbound permutations: a continuous subsequence is inherited from one parent while maintaining relative order from the other parent. With the two new offspring permutations obtained by OX, the trucks are equally reassigned to the inbound and outbound doors according to the permutation order. Crucially, the release time $\boldsymbol{\tau}$ are determined by the EST/LST rules rather than being directly crossed. For the trucks outside the specified time window, their assignment and sequence are retained, while setting a negative start time indicating their unscheduled status. By retaining their position instead of removing it, this methodology accounts for the potential feasibility arising from assigning these trucks to other dock doors during subsequent local improvements.

6. Computational experiments

This section reports the computational experiments evaluating the proposed algorithm. Section 6.1 describes the benchmark instances and experimental protocols. Section 6.2 compares SimHGS with representative reference algorithms. The subsequent sections further analyze the robustness of the proposed policy as well as the contribution of key algorithmic components.

6.1. Benchmark sets and experimental protocols

We generated 72 test instances, categorized into three sets (SetA, SetB, and SetC), to evaluate our approach. These sets differ mainly in their goods flow characteristics and truck arrival patterns. SetA features a sparse goods flow matrix (density 0.3), while SetB adopts a fully dense matrix (density 1.0). SetC shares the same flow density as SetB but introduces more concentrated truck arrival times to simulate higher temporal congestion. For SetA and SetB, truck related parameters (arrival times, expected departure times, and loading/unloading durations) were generated following the methodology of Miao et al. (2014). In contrast, SetC adjusts the arrival time distributions to reflect tighter scheduling windows. Together, these three sets disentangle the effects of flow density and temporal congestion: SetA isolates sparse coupling, SetB captures dense coupling under baseline arrival patterns, and SetC adds concentrated arrivals to further increase complexity. This design enables a separate assessment of each factor's impact on algorithmic performance. Each instance is labeled in the format $|\mathcal{I}| \times |\mathcal{D}^I|c$ (e.g., 10×4A), where $|\mathcal{I}| = |\mathcal{O}|$ denotes the number of inbound/outbound trucks and $|\mathcal{D}^I| = |\mathcal{D}^O|$ the number of inbound/outbound doors. The cross-docking layout is rectangular, with inter-door distances h_{kl} computed using the Manhattan metric (Guignard et al. 2012, Miao et al. 2014). Detailed instance generation parameters are provided in Table OS.1, and all instances are available online at https://github.com/INFORMSJoC/2025.1515.

Uncertain truck arrivals are modeled by bounded perturbations around their nominal arrival times $\bar{a}_i$. In the main setting, the arrival deviation for each truck is sampled from a triangular distribution $Tri(-6, 6, 0)$. This distribution is chosen for its bounded support, which avoids unrealistic extreme deviations, and its unimodal shape to concentrate probability around zero, reflecting that small deviations are more likely than

large ones. The processing times follow a uniform distribution $U[5, 10]$, representing a neutral assumption where any duration within the range is equally likely. The maximum deviation of 6 time units is comparable to the average processing time and slightly exceeds the range of processing times. Thus, the chosen deviation magnitude represents a moderate yet non-negligible uncertainty level, capturing practically relevant early and late arrivals without introducing extreme disruptions. For each scenario $\omega \in \Omega$, the arrival times of all inbound and outbound trucks are independently perturbed according to the selected distribution, yielding one complete realization of the stochastic arrival process. The scenario set Ω is generated via Monte Carlo sampling and is used to estimate the expected objective value in simulation. In addition to main comparative experiments, we conduct sensitivity analyses on the tardiness-weight parameter and the scenario size. For brevity, these results are reported in Section OS.5 of the Online Supplement.

Our algorithms were coded in C++ and compiled using GNU gcc 10.2.0 with the -O3 flag. Experiments was conducted on a computing cluster equipped with an Intel(R) Xeon(R) Gold 6226R (2.90GHz) processor running the Linux 3.10.x86_64 system. Each task was executed using a single core. SimHGS employs several parameters, described in Table OS.2 (Section OS.2). These parameters were calibrated using the automatic tuning package Irace (López-Ibáñez et al. 2016) with a total computation budget of 5,000 runs.

6.2. Computational comparisons

To demonstrate the efficiency of the SimHGS algorithm, we compare it with three representative stochastic scheduling heuristics, namely the traditional simheuristic (TSim), simulation-based descent with first improvement (SBDF), and simulation-based descent with best improvement (SBDB).

- **TSim**: TSim is based on the general simheuristic framework proposed by Juan et al. (2015) and serves as the traditional simheuristic counterpart of our algorithm. In the implementation, each promising solution is evaluated using a simulation of 100 scenarios.
- **SBDF**: Similar to SimHGS, SBDF can be viewed as a variant of the SBD algorithm proposed by Deblaere et al. (2011), incorporating a first improvement strategy but without the fast evaluation technique. In this setting, every iteration of SBLS evaluates the entire neighborhood using a simulation of 100 scenarios.
- **SBDB**: SBDB is similar to SBDF but employs a best improvement strategy. This variant is included primarily to assess the impact of the improvement strategy on performance.

We employ three different scenario budgets to balance computational speed with the accuracy of our estimates. During the search, a small in-search budget is used to keep repeated evaluations fast, which is necessary since the reference algorithms do not have a fast evaluation technique. After the search, we re-evaluate the candidate solutions using 1,000 scenarios to reduce the random noise in our estimates and more reliably identify the best solution from each algorithm. Finally, a large budget of 20,000 scenarios is used only for the out-of-sample assessment reported in the tables to ensure the final results are precise.

Additionally, all algorithms follow the same multi-start protocol, consisting of 10 independent runs with the same time limit per run. This design ensures a fair comparison by accounting for the fact that these randomized methods can be sensitive to their starting points.

Table 2 reports the comparative performance of SimHGS and the three reference algorithms on SetA–SetC; detailed instance-level results are available online at https://github.com/INFORMSJoC/2025.1515. For each instance set, "BestObj" and "AvgObj" denote the best and average objective values over 10 independent runs, respectively. Under both metrics, "#Wins", "#Ties", and "#Losses" denote the numbers of instances on which SimHGS performs better than, equal to, or worse than the corresponding reference algorithm; these counts are based on pairwise comparisons under the respective metric. The p-values from the Wilcoxon signed-rank test are also reported to assess the statistical significance at the 0.05 level.

As shown in Table 2, SimHGS consistently outperforms the reference algorithms TSim, SBDF, and SBDB, both in terms of the best and average results. For the best results, SimHGS achieves superior performance for 67 out of 72 instances. Notably, it finds the best solutions for all instances in SetA and SetB. In SetC, SimHGS is outperformed by TSim for 2 instances and by SBDF and SBDB for 3 instances each. Regarding average performance, SimHGS delivers better average solutions for 70 out of 72 instances. The small p–values (< 0.05) from Wilcoxon signed-rank tests against each reference algorithm confirm that the performance differences are statistically significant.

Table 2 Summarized comparison results of SimHGS with reference algorithms on three instance sets.

Instances	Pair algorithms	*BestObj*				*AvgObj*			
		#Wins	#Ties	#Losses	p-value	#Wins	#Ties	#Losses	p-value
SetA	SimHGS vs. TSim	24	0	0	1.19E-07	24	0	0	1.19E-07
	SimHGS vs. SBDF	23	1	0	2.38E-07	23	1	0	1.19E-07
	SimHGS vs. SBDB	23	1	0	2.38E-07	23	1	0	2.38E-07
	SimHGS vs. All	23	1	0	-	23	1	0	-
SetB	SimHGS vs. TSim	24	0	0	1.19E-07	24	0	0	1.19E-07
	SimHGS vs. SBDF	24	0	0	1.19E-07	24	0	0	1.19E-07
	SimHGS vs. SBDB	24	0	0	1.19E-07	24	0	0	1.19E-07
	SimHGS vs. All	24	0	0	-	24	0	0	-
SetC	SimHGS vs. TSim	22	0	2	2.98E-06	23	0	1	5.96E-07
	SimHGS vs. SBDF	20	1	3	3.02E-05	23	0	1	5.96E-07
	SimHGS vs. SBDB	21	0	3	2.47E-05	23	0	1	5.96E-07
	SimHGS vs. All	19	0	5	-	22	0	2	-

Figures OS.1a to OS.1b (Section OS.3) show the deviations of each algorithm from the best known result, reporting both best and average objective values across all instances. Compared with the summary in Table 2, Figures OS.1a to OS.1b offer a complementary view. TSim deviations decrease with problem size for both best and average results. SBDF and SBDB deviations increase with problem size, indicating weaker performance on larger instances. SimHGS deviations remain stable across all sets, showing strong performance in both best and average outcomes.

These observations highlight search performance bottlenecks in both the deterministic and stochastic components of the algorithm. For small instances, neither VND nor SBLS presents a bottleneck, enabling SimHGS, SBDF, and SBDB to outperform TSim, which lacks local search improvement under stochastic scenarios. However, as problem size increases, SBLS without the fast evaluation (limited to 100 scenarios) becomes a performance bottleneck, significantly increasing search time. As a result, the best solutions of SBDF and SBDB progressively deteriorate relative to TSim, particularly for SetB and SetC instances. In contrast, SimHGS leverages the fast evaluation mechanism to overcome this bottleneck, maintaining efficiency even when using 1,000 scenarios. These results demonstrate that our proposed algorithm framework effectively integrates metaheuristics with SBLS, while also underscoring the importance of balancing deterministic and stochastic search components in simheuristic algorithm design.

6.3. Comparison with baseline scheduling policies

To validate the advantages of SimHGS, we benchmark its performance against three baseline policies that expose the inherent limitations of deterministic optimization and myopic dispatching rules. These baselines include: (1) a Deterministic Policy (Det), which solves the deterministic SCDSP using the SimHGS but omits the SBLS refinement; (2) Earliest Departure Time (EDT), a temporal-priority online rule that sequences outbound trucks based on their expected departure times e_j; and (3) Maximum Transferred Goods (MTG), a throughput-oriented dynamic rule where an available dock selects the waiting truck j with the highest volume of currently transferable goods ($\sum_{i\in\mathcal{I}:\hat{s}_i+q_{ij}t_{kl}<t} f_{ij}$ at decision epoch t). To ensure a fair comparison, both EDT and MTG incorporate a dynamic departure mechanism where inbound trucks depart immediately upon unloading, whereas assigned outbound trucks remain at the dock to consolidate incoming goods until their latest permissible departure time. Across 20,000 out-of-sample scenarios, Figure 2 illustrates the percentage deviation of these baseline objective values relative to the SimHGS solutions.

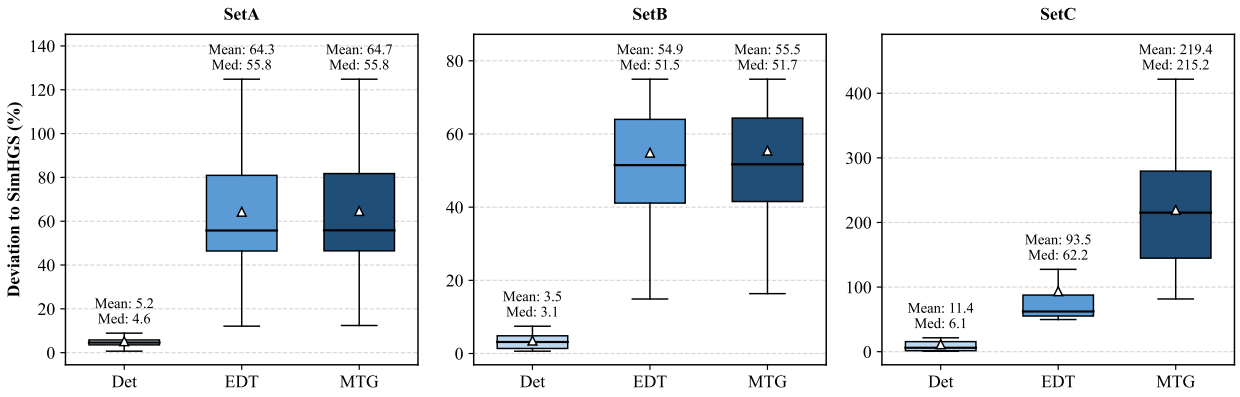


Figure 2 The performance deviation of baseline policies compared to SimHGS across all instance sets.

As illustrated in Figure 2, SimHGS consistently outperforms all baselines, emphasizing the necessity of balancing temporal constraints with flow synchronization under uncertainty. The Det policy yields mean

deviations of 5.2%, 3.5%, and 11.4% across SetA, SetB, and SetC, respectively; this confirms that executing deterministic sequences without stochastic release-time adjustments leads to sub-optimal real-time performance. In contrast, simple online rules suffer from their myopic objectives. EDT ignores inbound readiness, causing premature outbound loading and substantial untransferred goods penalties (54.9%–93.5% mean deviation). Conversely, MTG induces severe resource contention: by monopolizing dock doors to maximize immediate consolidation, it obstructs subsequent trucks, triggering a cascade of service failures and massive tardiness costs. This instability is particularly evident in the congested SetC, where MTG's deviation surges to 219.4%. Overall, the fixed-sequence and release-time policy effectively reconciles competing objectives, demonstrating superior robustness in stochastic environments. (See Section OS.5.1 of Online Supplement for a detailed sensitivity analysis under varying uncertainty levels.)

6.4. Scheduling policy robustness analysis

This section presents a robustness assessment of the scheduling policy identified by SimHGS on 20,000 out-of-sample scenarios. To comprehensively evaluate performance under varying risk conditions, we adopt the risk measure *Value-at-Risk* (VaR) (Embrechts et al. 2022), which estimates the worst-case cost at a specified confidence level (CL). Formally, VaR at CL α is defined as: $\mathrm{VaR}_\alpha = \inf\{x \in \mathbb{R} : P(\mathrm{Cost} \leq x) \geq \alpha\}$.

Based on VaR, we define *Performance Deviation* (Dev) as the percentage by which VaR at confidence level α exceeds the estimated policy execution cost $\mathcal{C}(\Pi, \Omega)$ of the scheduling policy: $Dev = \frac{\mathrm{VaR}_\alpha - \mathcal{C}(\Pi,\Omega)}{\mathcal{C}(\Pi,\Omega)} \times 100\%$. VaR is an absolute measure, whereas Dev captures relative robustness deterioration and is therefore more suitable for comparing instances with different cost magnitudes.

We evaluate Dev at confidence levels $\alpha \in \{1.0, 0.99, 0.95, 0.90\}$ using 20,000 out-of-sample scenarios. Figures 3 illustrate, for each instance, whether Dev at each CL remains below a predefined threshold. For SetA instances, all 24 instances satisfy $Dev \leq 5\%$ at $\alpha = 0.9$, while 23 instances meet this criterion at $\alpha = 0.99$ and $\alpha = 0.95$. For SetB, all 24 instances remain within $Dev \leq 7\%$ at both $\alpha = 0.95$ and $\alpha = 0.9$. In contrast, SetC achieves $Dev \leq 7\%$ for only 12 instances at $\alpha = 0.95$. Compared to SetA and SetB, the scheduling policy identified by SimHGS for SetC are less robust, suggesting that high truck concentration leads to increased sensitivity to uncertainty. Overall, SimHGS produces stable and robust scheduling policies for SetA and SetB, while robustness is comparatively weaker for SetC.

The weaker robustness of SetC can be attributed to its compressed arrival time distribution. Compared with SetA and SetB, SetC combines dense flow interactions with more temporally concentrated truck arrivals, which reduces the available temporal flexibility. Under such compression, local disruptions tend to propagate more easily to subsequent operations, consistent with the higher VaR values observed for SetC. As a result, under arrival uncertainty, a delay in one truck is more likely to increase both truck tardiness and the penalty for untransferred goods, leading to higher VaR values. From a managerial perspective, these results suggest that, in highly congested environments, cross-docking performance could be further

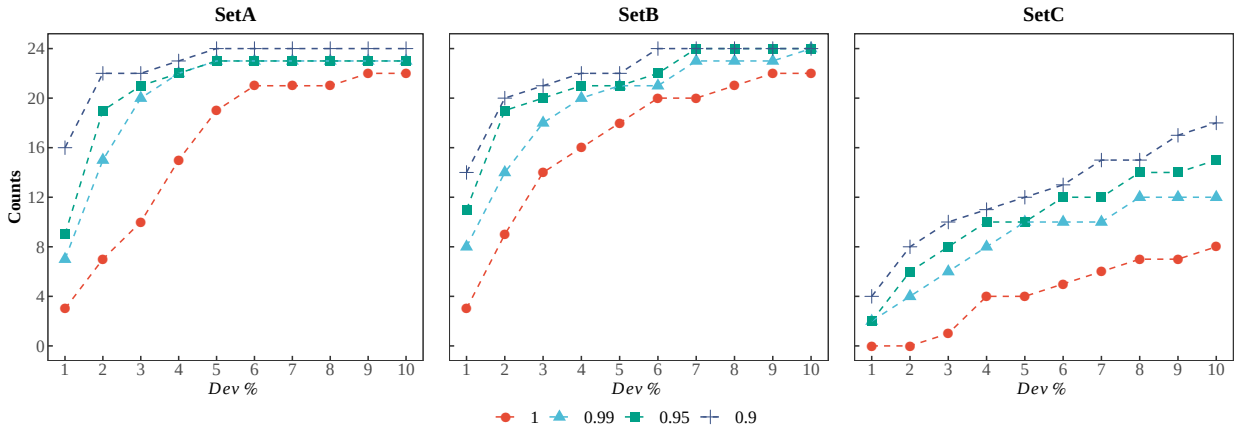


Figure 3 Counts of performance guarantee instances for worst-case scenarios across confidence levels

improved through appointment scheduling, i.e., assigning coordinated arrival time windows to inbound and outbound trucks in advance. This better synchronizes unloading and loading activities, reduces dock congestion, and enhances operational robustness.

6.5. Effectiveness of the fast evaluation technique

To evaluate the effectiveness of the proposed fast evaluation technique, we assess its impact from two perspectives: (i) the computational efficiency of the local search and (ii) the overall algorithmic performance within a constrained time budget.

We first isolate the technique's impact on SBLS using a multi-start procedure focused on optimizing release times τ. Each iteration generates an initial τ based on a given sequence π (using the EST/LST rule), optimizes τ via a specific search variant, and perturbs π to restart. We compare five variants:

- **SBLS**: Employs the best improvement strategy empowered by the proposed fast evaluation technique.
- **FI& BI**: Apply first-improvement and best-improvement strategies, respectively, without the fast evaluation technique.
- **Sim & SimN**: Serve as baseline controls that evaluate policies using standard simulation (Sim) and the modified simulation proposed in this study (SimN), both without local search improvement.

All variants were tested using identical random seeds to ensure consistent initial conditions. Figure 4 reports the average runtime for 10 runs with 1,000 restarts each. The results indicate that on SetA, SBLS introduces negligible overhead compared to the no-search baselines. For the largest instance (100x15A), SBLS requires only 34.2 seconds, whereas FI and BI exhibit near-exponential runtime growth, exceeding 13,200 seconds and 24,033 seconds, respectively. Similar trends are observed in SetB and SetC. These results confirm that the fast evaluation technique reduces neighborhood evaluation time by one to two orders of magnitude, rendering SBLS computationally tractable for large-scale instances.

To demonstrate how this computational efficiency translates into solution quality, we compare SimHGS against an ablated variant (NoFast) which disables the fast evaluation technique. Both algorithms were executed with a time limit of 1,800 seconds. As summarized in Table 3, SimHGS consistently and significantly

outperforms NoFast across all instance sets. In terms of the best objective value, SimHGS achieves 66 wins out of 72 instances; for the average objective value, the superiority is even more pronounced with 70 wins.

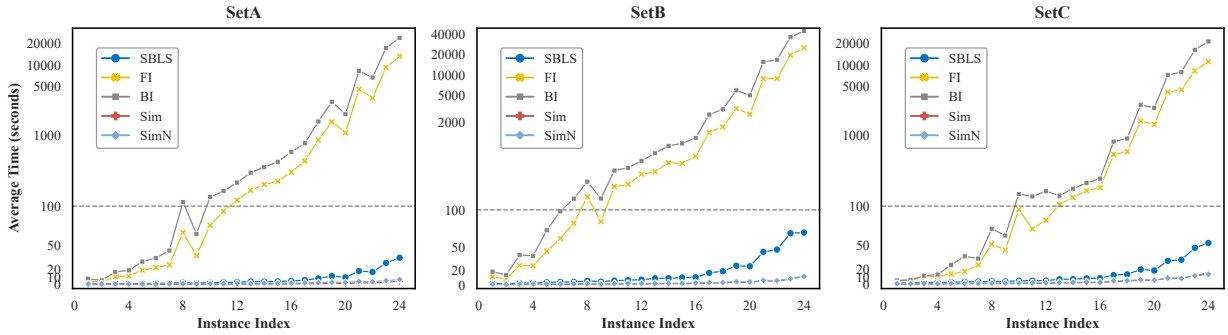


Figure 4 Average running time to execute multi-start on SetA, SetB and SetC respectively

These findings confirm that the fast evaluation technique is an indispensable component of our methodology. By drastically lowering the per-move computational cost, it enables the algorithm to conduct a substantially deeper search within the same time limit, leading to more robust optimization under uncertainty. A comprehensive ablation analysis, including runtime breakdowns, the impact of the cutoff rule, and the HGS framework, is provided in Section OS.6 of Online Supplement.

Table 3 Summary of comparison results between SimHGS and NoFast

Instances	*BestObj*			*AvgObj*		
	#Wins	#Ties	#Losses	#Wins	#Ties	#Losses
SetA	21	3	0	23	1	0
SetB	24	0	0	24	0	0
SetC	21	1	2	23	0	1
Total	66	4	2	70	1	1
p-value	5.85E-12			5.11E-13		

7. Conclusions

This study addressed the stochastic cross-docking truck scheduling problem by integrating flexible dock occupancy with a robust fixed-sequence and release-time policy. Our modeling approach captured the continuous-time dynamics of cross-docking operations, providing a practical framework to mitigate the impact of uncertain truck arrivals. To solve this problem, we developed SimHGS, a hybrid genetic simheuristic that integrates population-based search with a NSBLS. Within this architecture, a VND was employed to optimize truck sequences in deterministic settings, while a SBLS was applied to refine release times under stochastic realizations–an integrated search strategy that expands the search landscape beyond deterministic local optima to ensure both solution diversity and robustness. To mitigate the computational burden, we introduced a fast scenario evaluation technique that effectively eliminates simulation as a bottleneck. Extensive computational experiments demonstrated the superior performance of SimHGS compared to state-of-the-art simheuristics, while an out-of-sample analysis of 20,000 scenarios confirmed the

policy's effectiveness. Future research could further extend this framework by adopting Markov Decision Process formulations to refine online scheduling through rollout algorithms or reinforcement learning. For simheuristics, extending the fast evaluation technique to other stochastic scheduling problems with uncertain job start times offers a promising direction for developing efficient solution methods.

Acknowledgments

The authors thank the editors and all anonymous reviewers for their insightful comments and valuable suggestions, which have significantly improved the quality of this manuscript.

References

Acar K, Yalcin A, Yankov D (2012) Robust door assignment in less-than-truckload terminals. *Computers & Industrial Engineering* 63(4):729–738.

Albareda-Sambola M, Fernández E, Saldanha-da Gama F (2017) Heuristic solutions to the facility location problem with general bernoulli demands. *INFORMS Journal on Computing* 29(4):737–753.

Bodnar P, de Koster R, Azadeh K (2015) Scheduling trucks in a cross-dock with mixed service mode dock doors. *Transportation Science* 51(1):112–131.

Boysen N, Briskorn D, Tschöke M (2013) Truck scheduling in cross-docking terminals with fixed outbound departures. *OR Spectrum* 35:479–504.

Boysen N, de Koster R, Weidinger F (2019) Warehousing in the e-commerce era: A survey. *European Journal of Operational Research* 277(2):396–411.

Buijs P, Vis IF, Carlo HJ (2014) Synchronization in cross-docking networks: A research classification and framework. *European Journal of Operational Research* 239(3):593–608.

Chica M, Juan Pérez AA, Cordon O, Kelton D (2017) Why simheuristics? benefits, limitations, and best practices when combining metaheuristics with simulation. *Benefits, Limitations, and Best Practices When Combining Metaheuristics with Simulation (January 1, 2017)* .

Deblaere F, Demeulemeester E, Herroelen W (2011) Proactive policies for the stochastic resource-constrained project scheduling problem. *European Journal of Operational Research* 214(2):308–316.

Embrechts P, Schied A, Wang R (2022) Robustness in the optimization of risk measures. *Operations Research* 70(1):95–110.

Goodson JC, Ohlmann JW, Thomas BW (2013) Rollout policies for dynamic solutions to the multivehicle routing problem with stochastic demand and duration limits. *Operations Research* 61(1):138–154.

Guemri O, Nduwayo P, Todosijević R, Hanafi S, Glover F (2019) Probabilistic tabu search for the cross-docking assignment problem. *European Journal of Operational Research* 277(3):875–885.

Guignard M, Hahn PM, Pessoa AA, da Silva DC (2012) Algorithms for the cross-dock door assignment problem. *Proceedings of the Fourth International Workshop on Model-based Metaheuristics*, 145–162.

He P, Hao JK, Wu Q (2025) A hybrid genetic algorithm with multi-population for capacitated location routing. *INFORMS Journal on Computing*, Published Online:11 Jul 2025, https://doi.org/10.1287/ijoc.2023.0416.

Juan AA, Faulin J, Grasman SE, Rabe M, Figueira G (2015) A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives* 2:62–72.

Konur D, Golias MM (2013a) Analysis of different approaches to cross-dock truck scheduling with truck arrival time uncertainty. *Computers & Industrial Engineering* 65(4):663–672.

Konur D, Golias MM (2013b) Cost-stable truck scheduling at a cross-dock facility with unknown truck arrivals: A meta-heuristic approach. *Transportation Research Part E: Logistics and Transportation Review* 49(1):71–91.

Ladier AL, Alpan G (2016) Robust cross-dock scheduling with time windows. *Computers & Industrial Engineering* 99:16–28.

Ladier AL, Alpan G (2018) Crossdock truck scheduling with time windows: earliness, tardiness and storage policies. *Journal of Intelligent Manufacturing* 29(3):569–583.

López-Ibáñez M, Dubois-Lacoste J, Cáceres LP, Birattari M, Stützle T (2016) The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives* 3:43–58.

Máximo VR, Cordeau JF, Nascimento MC (2024) AILS-II: An adaptive iterated local search heuristic for the large-scale capacitated vehicle routing problem. *INFORMS Journal on Computing* 36(4):974–986.

Miao Z, Cai S, Xu D (2014) Applying an adaptive tabu search algorithm to optimize truck-dock assignment in the crossdock management system. *Expert Systems with Applications* 41(1):16–22.

Miao Z, Lim A, Ma H (2009) Truck dock assignment problem with operational time constraint within crossdocks. *European Journal of Operational Research* 192(1):105–115.

Nassief W, Contreras I, Jaumard B (2018) A comparison of formulations and relaxations for cross-dock door assignment problems. *Computers & Operations Research* 94:76–88.

Rijal A, Bijvank M, de Koster R (2019) Integrated scheduling and assignment of trucks at unit-load cross-dock terminals with mixed service mode dock doors. *European Journal of Operational Research* 278(3):752–771.

Sokol V, Custic A, Punnen AP, Bhattacharya B (2020) The bilinear assignment problem: Large neighborhoods and experimental analysis of algorithms. *INFORMS Journal on Computing* 32:730–746.

Vidal T, Crainic TG, Gendreau M, Lahrichi N, Rei W (2012) A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research* 60(3):611–624.

Vidal T, Crainic TG, Gendreau M, Prins C (2014) A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research* 234(3):658–673.

Xi X, Changchun L, Yuan W, Hay LL (2020) Two-stage conflict robust optimization models for cross-dock truck scheduling problem under uncertainty. *Transportation Research Part E: Logistics and Transportation Review* 144:102123.

Zaerpour N, Yu Y, de Koster RB (2015) Storing fresh produce for fast retrieval in an automated compact cross-dock system. *Production and Operations Management* 24(8):1266–1284.

Online Supplement for "A simheuristic for the cross-docking scheduling problem under uncertain truck arrival times"

This supplement provides mathematical proofs (Section OS.1), experimental configurations (Section OS.2), detailed comparison results (Section OS.3), the benchmark MILP model (Section OS.4), sensitivity analyses (Section OS.5), and ablation studies (Section OS.6).

OS.1. Proofs

The proof of Proposition 1

Proof. Let $\boldsymbol{s}^*$ denote the optimal start times under the given sequence $\boldsymbol{\pi}$, and define $\bar{\boldsymbol{s}}$ such that:

$$\bar{s}_i = \begin{cases} ES_i, & i \in \mathcal{I}, \\ s_i^*, & i \in \mathcal{O}, \end{cases}$$

Let the deterministic objective value under sequence $\boldsymbol{\pi}$ and schedule $\boldsymbol{s}$ be:

$$DetC(\boldsymbol{\pi}, \boldsymbol{s}) = c\left[\sum_{i\in\mathcal{I}} \max(0, s_i + g_i - e_i) + \sum_{j\in\mathcal{O}} \max(0, s_j + g_j - e_j)\right] + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{O}} \eta_{ij},$$

where e_i denotes the expected departure time for truck i, and η_{ij} is the penalty cost component involving goods f_{ij}. Since the EST rule always produces $\bar{s}_i \le s_i^*$ for all $i \in \mathcal{I}$, and $\bar{s}_j = s_j^*$ for all $j \in \mathcal{O}$, it follows that: $DetC(\boldsymbol{\pi}, \bar{\boldsymbol{s}}) \le DetC(\boldsymbol{\pi}, \boldsymbol{s}^*)$.

Therefore, $\bar{\boldsymbol{s}}$ is also an optimal start time under the fixed sequence $\boldsymbol{\pi}$. □

The proof of Theorem 1

Proof. We prove the theorem by induction on the position of trucks in the sequence $\boldsymbol{\pi}_l$ at a given outbound door $l \in \mathcal{D}^O$. Consider the first truck $j_1 \in \boldsymbol{\pi}_l$, the realized starting time of truck j_1 is

$$\hat{s}_{j_1\omega} = \max\{\hat{a}^\omega_{j_1}, \tau_{j_1}\} = \max\{ES^O_{j_1\omega}, \tau_{j_1}\},$$

which satisfies Eq. (5). Assume the Eq. (5) holds for the $(k-1)$-th truck j_{k-1} in the sequence. We need to show that this holds for the k-th truck j_k. By definition, $\hat{s}_{j_k\omega}$ is given by

$$\begin{aligned}
\hat{s}_{j_k\omega} &= \max\{\hat{a}^\omega_{j_k}, \tau_{j_k}, \min\{\hat{s}_{j_{k-1}\omega} + g_{j_{k-1}}, d_{j_{k-1}}\}\} \\
&= \max\{\hat{a}^\omega_{j_k}, \tau_{j_k}, \min\{\max\{ES^O_{j_{k-1}\omega}, \tau_{j_{k-1}}\} + g_{j_{k-1}}, d_{j_{k-1}}\}\} && \text{(OS.1)} \\
&= \max\{\hat{a}^\omega_{j_k}, \tau_{j_k}, \min\{ES^O_{j_{k-1}\omega} + g_{j_{k-1}}, d_{j_{k-1}}\}, \min\{\tau_{j_{k-1}} + g_{j_{k-1}}, d_{j_{k-1}}\}\} && \text{(OS.2)} \\
&= \max\{ES^O_{j_k\omega}, \tau_{j_k}\} && \text{(OS.3)}
\end{aligned}$$

Eq. (OS.1) follows from induction hypothesis. The equality in Eq. (OS.2) follows from the maximization. Eq. (OS.3) follows from definitions that $ES^O_{j_k\omega} = \max\{\hat{a}^\omega_{j_k}, \min\{ES^O_{j_{k-1}\omega} + g_{j_{k-1}}, d_{j_{k-1}}\}\}$ and $\tau_{j_k} \ge \min\{\tau_{j_{k-1}} + g_{j_{k-1}}, d_{j_{k-1}}\}$. By induction, the realized start time $\hat{s}_{j_k} = \max\left\{ES^O_{j_k\omega}, \tau_{j_k}\right\}$ holds for all trucks $j_k \in \boldsymbol{\pi}_l$. □

The proof of Theorem 2

Proof. Let $\hat{s}_{j\omega}(\tau_j)$ be the realized start time under τ_j, we show that $\hat{s}_{j\omega}(\tau_j) = \hat{s}_{j\omega}(\tau'_j)$ holds for all $j \in \boldsymbol{\pi}_l$. Proofs are by induction. Consider the first truck $j_1 \in \tau_l$, let $\tau'_{j_1} = \tau_{j_1}$, the result holds. Assume $\hat{s}_{j_{k-1}\omega}(\tau_{j_{k-1}}) = \hat{s}_{j_{k-1}\omega}(\tau'_{j_{k-1}})$ holds for $(k-1)$-th truck j_{k-1}. Then for k-th truck j_k, by definition,

$$\begin{aligned}
\hat{s}_{j_k\omega}(\tau_{j_k}) &= \max\{\hat{a}^{\omega}_{j_k}, \tau_{j_k}, \min\{\hat{s}_{j_{k-1}\omega}(\tau'_{j_{k-1}}) + g_{j_{k-1}\omega}, d_{j_{k-1}}\}\} \\
&= \max\{\hat{a}^{\omega}_{j_k}, \tau_{j_k}, \min\{ES^{O}_{j_{k-1}\omega} + g_{j_{k-1}}, d_{j_{k-1}}\}, \min\{\tau'_{j_{k-1}} + g_{j_{k-1}}, d_{j_{k-1}}\}\} && \text{(OS.4)} \\
&= \max\{ES^{O}_{j_k\omega}, \tau'_{j_k}\} && \text{(OS.5)} \\
&= \hat{s}_{j_k\omega}(\tau'_{j_k}) && \text{(OS.6)}
\end{aligned}$$

Eq. (OS.4) follows from Theorem 1. Eq. (OS.5) is derived from the definition of $ES^{O}_{j_k\omega}$ and $\tau'_{j_k} = \max\{\min\{\tau'_{j_{k-1}} + g_{j_{k-1}}, d_{j_{k-1}}\}, \tau_{j_k}\}$. □

OS.2. Instance generation and algorithm parameters configuration

Table OS.1 Instance Generation Parameters

Parameter	Generation Method / Value	SetA	SetB	SetC
Non-zero flow value ($f_{ij} > 0$)	$U[6, 60]$ (Miao et al. 2014)	✓	✓	✓
Density	Fraction of non-zero flows	0.3	1.0	1.0
Penalty coefficient (p_{ij})	$U[15, 25]$	✓	✓	✓
Inbound arrival ($a_i, i \in \mathcal{I}$)	$U\left[1, \frac{60 \cdot \lvert\mathcal{I}\rvert}{\lvert\mathcal{D}^I\rvert}\right]$ (Miao et al. 2014)	✓	✓	✓*
Outbound arrival ($a_j, j \in \mathcal{O}$)	$U\left[15, 15 + \frac{60 \cdot \lvert\mathcal{O}\rvert}{\lvert\mathcal{D}^O\rvert}\right]$ (Miao et al. 2014)	✓	✓	✓*
Expected departure ($e_j, j \in \mathcal{I} \cup \mathcal{O}$)	$e_j = a_j + U[25, 40]$ (Miao et al. 2014)	✓	✓	✓
Latest departure ($d_j, j \in \mathcal{I} \cup \mathcal{O}$)	$d_j = e_j + U[10, 15]$	✓	✓	✓
Processing time ($g_j, j \in \mathcal{I} \cup \mathcal{O}$)	$U[5, 10]$	✓	✓	✓
Speed (v)	1	✓	✓	✓
Max load capacity (Q)	20	✓	✓	✓
Distance (h_{kl})	Manhattan distance between inbound door $k \in \mathcal{D}^I$ and outbound door $l \in \mathcal{D}^O$. Rectangular cross-dock layout with opposite doors fixed 3 units apart and adjacent doors on the same side 1 unit apart (Miao et al. 2009, 2014, Guignard et al. 2012).			

* *SetC uses modified arrival times:* $a_i \sim U[1, 2 \cdot |\mathcal{I}|]$, $a_j \sim U[15, 15 + 2 \cdot |\mathcal{O}|]$

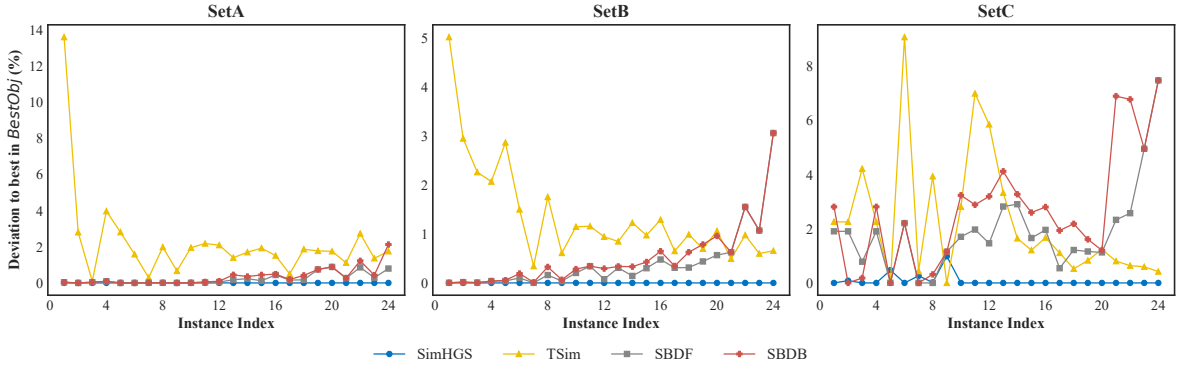
SimHGS uses the same parameter configuration for SetA and SetB instances and a different parameter configuration for SetC instances.

Table OS.2 Parameter configuration via Irace (López-Ibáñez et al. 2016)

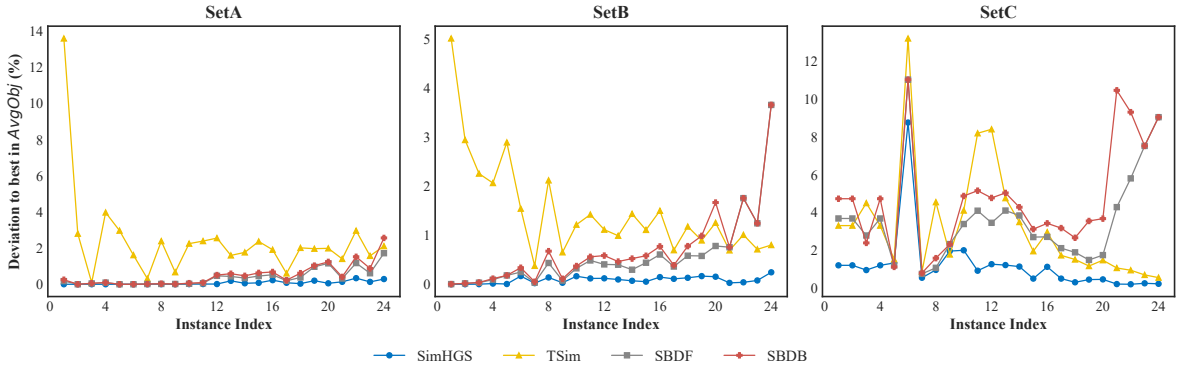
Param.	Section	Description	Tuning range	SetA & SetB	SetC
μ	3.6	Minimum population size	$\{5, 10, 15\}$	5	5
λ	3.6	Maximum increase in population size	$\{5, 10, 15, 20\}$	10	10
$Imax$	3.1	Maximum iteration of consecutive unimproved	$\{5000, 10000, 20000\}$	10000	5000
β	3.1	Tolerance to the best value for deterministic version	$[0.6, 1.0]$	1.0	0.9
n_{close}	3.4	Number of most similar individuals	$\{3, 4, 5, 6\}$	3	5
n_{elite}	3.4	Number of best elite individuals	$\{3, 4, 5, 6\}$	3	5

OS.3. Detailed comparison results of SimHGS against the reference algorithms

Figures OS.1a to OS.1b illustrate the deviations of each algorithm from the best known value, reporting both the best and average objective values across all instances. For three SetC instances (10x3C, 10x4C, 15x4C), TSim produced extreme deviations ($> 330\%$). To improve visualization, these outliers were replaced by the average deviation of the remaining instances. All detailed results are available at https://github.com/INFORMSJoC/2025.1515.



(a) Best objective value



(b) Average objective value

Figure OS.1 Deviations of the best and average objective values from the reference results

OS.4. The deterministic MILP model for SCDSP

Table OS.3 Variables for Cross-Dock Scheduling Problem

Notation	Description
x_{ik}	1 if inbound truck $i \in \mathcal{I}$ is assigned to unloading door $k \in \mathcal{D}^{\mathcal{I}}$, 0 otherwise
y_{jl}	1 if outbound truck $j \in \mathcal{O}$ is assigned to loading door $l \in \mathcal{D}^{\mathcal{O}}$, 0 otherwise
z_{ijkl}	1 if goods flow f_{ij} is successfully transferred via inbound door $k \in \mathcal{D}^{\mathcal{I}}$ to outbound door $l \in \mathcal{D}^{\mathcal{O}}$, 0 otherwise
s_i	Processing start time of truck $i \in \mathcal{I} \cup \mathcal{O}$
$\zeta_{ii'}$	1 if truck i, i' are assigned to the same door and i is processed before i', 0 otherwise
t_i	Departure tardiness of truck $i \in \mathcal{I} \cup \mathcal{O}$

To further evaluate the effectiveness of the HGS algorithm, we consider the deterministic version of the SCDSP and formulate it as a mixed-integer linear programming (MILP) model. This mathematical formula-

tion serves as an exact benchmark for assessing both the solution quality and computational efficiency of our heuristic. The notation and decision variables employed in the MILP model are summarized in Table OS.3.

$$\text{MILP min} \quad c \sum_{i \in \mathcal{I} \cup \mathcal{O}} t_i + \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{O}} p_{ij} f_{ij} \Big(1 - \sum_{k \in \mathcal{D}^{\mathcal{I}}} \sum_{l \in \mathcal{D}^{\mathcal{O}}} z_{ijkl}\Big) \tag{OS.7}$$

$$\text{s.t.} \quad \sum_{k \in \mathcal{D}^{\mathcal{I}}} x_{ik} \leq 1, \qquad \forall i \in \mathcal{I} \tag{OS.8}$$

$$\sum_{l \in \mathcal{D}^{\mathcal{O}}} y_{jl} \leq 1, \qquad \forall j \in \mathcal{O} \tag{OS.9}$$

$$\zeta_{ii'} + \zeta_{i'i} \geq x_{ik} + x_{i'k} - 1, \qquad \forall (i,i') \in \mathcal{I}^2, i < i', \forall k \in \mathcal{D}^{\mathcal{I}} \tag{OS.10}$$

$$\zeta_{jj'} + \zeta_{j'j} \geq y_{jl} + y_{j'l} - 1, \qquad \forall (j,j') \in \mathcal{O}^2, j < j', \forall l \in \mathcal{D}^{\mathcal{O}} \tag{OS.11}$$

$$\zeta_{ii'} + \zeta_{i'i} \leq 1, \qquad \forall (i,i') \in \mathcal{I}^2 \cup \mathcal{O}^2, i < i' \tag{OS.12}$$

$$s_{i'} - s_i - (M + g_i)\zeta_{ii'} \geq -M, \qquad \forall (i,i') \in \mathcal{I}^2 \cup \mathcal{O}^2, i \neq i' \tag{OS.13}$$

$$z_{ijkl} \leq x_{ik}, \qquad \forall i \in \mathcal{I}, j \in \mathcal{O}, k \in \mathcal{D}^{\mathcal{I}}, l \in \mathcal{D}^{\mathcal{O}} \tag{OS.14}$$

$$z_{ijkl} \leq y_{jl}, \qquad \forall i \in \mathcal{I}, j \in \mathcal{O}, k \in \mathcal{D}^{\mathcal{I}}, l \in \mathcal{D}^{\mathcal{O}} \tag{OS.15}$$

$$s_j - s_i - (M + q_{ij} h_{kl} + g_i) z_{ijkl} \geq -M, \qquad \forall i \in \mathcal{I}, j \in \mathcal{O}, k \in \mathcal{D}^{\mathcal{I}}, l \in \mathcal{D}^{\mathcal{O}} \tag{OS.16}$$

$$t_i \geq s_i + g_i - e_i, \qquad \forall i \in \mathcal{I} \cup \mathcal{O} \tag{OS.17}$$

$$s_i \geq a_i, \qquad \forall i \in \mathcal{I} \cup \mathcal{O} \tag{OS.18}$$

$$s_i \leq d_i - g_i, \qquad \forall i \in \mathcal{I} \cup \mathcal{O} \tag{OS.19}$$

$$x_{ik} \in \{0,1\}, \qquad \forall i \in \mathcal{I}, k \in \mathcal{D}^{\mathcal{I}} \tag{OS.20}$$

$$y_{jl} \in \{0,1\}, \qquad \forall j \in \mathcal{O}, l \in \mathcal{D}^{\mathcal{O}} \tag{OS.21}$$

$$\zeta_{ii'} \in \{0,1\}, \qquad \forall (i,i') \in \mathcal{I}^2 \cup \mathcal{O}^2, i \neq i' \tag{OS.22}$$

$$z_{ijkl} \in \{0,1\}, \qquad \forall i \in \mathcal{I}, j \in \mathcal{O}, k \in \mathcal{D}^{\mathcal{I}}, l \in \mathcal{D}^{\mathcal{O}} \tag{OS.23}$$

$$t_i \geq 0, \qquad \forall i \in \mathcal{I} \cup \mathcal{O} \tag{OS.24}$$

The objective function (OS.7) minimizes the total cost comprising truck tardiness and penalties for untransferred goods. Constraints (OS.8) and (OS.9) assign each inbound and outbound truck to at most one door. Constraints (OS.10)–(OS.13) govern the sequencing of trucks assigned to the same door and ensure nonoverlapping processing. Constraints (OS.14)–(OS.16) link goods transfer decisions to door assignments and timing feasibility. Constraints (OS.17)–(OS.19) define tardiness and enforce arrival time and deadline restrictions. Constraints (OS.20)–(OS.24) specify the variable domains.

Table OS.4 presents a comparative analysis of the proposed HGS algorithm and the MILP model across three benchmark sets (SetA, SetB, and SetC). For HGS, we report the best and average objective values from 10 independent runs, alongside the average runtime (limited to 600 seconds per run). The MILP model was solved using CPLEX 22.10 with a 3,600 seconds time limit. Bold values indicate the better result and optimal solutions obtained by the MILP model are highlighted with an asterisk "$*$". The results demonstrate that the HGS algorithm consistently outperforms the MILP model across all benchmark sets in both efficiency and solution quality. For smaller instances where the MILP model achieves proven optimality, HGS identifies the same optimal solutions in significantly less computational time. In more complex cases

Table OS.4 Detailed comparison results between HGS algorithm and MILP model across all instances

Inst.	Set A					Set B					Set C				
	HGS			MILP		HGS			MILP		HGS			MILP	
	Best	*AvgObj*	*AvgT*	*Best*	*Time*	*Best*	*AvgObj*	*AvgT*	*Best*	*Time*	*Best*	*AvgObj*	*AvgT*	*Best*	*Time*
10x3	**5506**	5506	0.1	**5506***	0.9	**17388**	17388	0.3	**17388***	69	**4**	5.4	246	**4***	2861
10x4	**7323**	7323	0.0	**7323***	109	**23637**	23637	0.3	**23637***	584	**0**	0	82	**0***	65
15x3	**10014**	10014	0.3	**10014***	4.9	**37643**	37643	1.1	**37643**	3590	**5004**	5004	67	8305	3590
15x4	**13198**	13198	0.4	**13198***	209	**50171**	50171	1.1	**50171**	3584	**67**	70.2	277	3057	3591
20x3	**17683**	17683	1.5	**17683***	18	**62015**	62015	73	**62015**	3585	**23316**	23494	207	39982	3580
20x4	**17132**	17132	2.1	**17132***	1260	**66644**	66644	6.1	66648	3591	**1129**	1232	102	17639	3591
25x3	**70383**	70383	27	**70383***	150	**241587**	241587	122	241588	3590	**69462**	70075	238	94796	3589
25x5	**35517**	35517	5.0	**35517**	3566	**115114**	115116	418	115142	3589	**38012**	38214	482	60446	3590
28x3	**64508**	64508	18	**64508***	67	**240743**	240743	84	**240743**	3586	**105510**	106402	375	147167	3590
28x5	**62791**	62791	12	**62791**	3580	**195792**	195792	144	195808	3591	**36884**	36896	499	125709	3591
30x6	**52850**	52850	92	52866	3586	**193411**	193412	336	195768	3584	**30898**	30901	321	56571	3590
30x8	**61223**	61223	70	62123	3590	**198136**	198138	274	208295	3591	**30874**	30881	452	397378	3590
35x6	**88426**	88427	149	88427	3589	**304140**	304142	360	308808	3591	**103916**	104440	466	184767	3585
35x8	**83009**	83009	226	83024	3590	**263681**	263684	340	274114	3585	**99385**	99759	452	523792	3591
40x6	**106645**	106646	425	107496	3590	**345278**	345285	416	352098	3586	**165861**	166771	445	254085	3590
40x8	**118276**	118282	449	118911	3590	**359056**	359059	536	376610	3590	**163852**	164889	318	248570	3586
50x6	**259785**	259787	381	259805	3591	**830089**	830098	446	836103	3592	**399607**	400781	494	1487770	3592
50x8	**167080**	167084	483	170956	3591	**522623**	522632	504	767383	3590	**389848**	391233	361	1441470	3592
60x8	**255983**	255999	492	265100	3587	**934177**	934202	394	1161790	3607	**504997**	505783	326	2179280	3591
60x10	**255826**	255838	523	266034	3591	**845184**	845213	395	2359840	3619	**504312**	506394	398	2359850	3612
80x8	**551678**	551707	527	722316	3593	**1808850**	1809117	368	4221240	3621	**1053370**	1056179	328	4221250	3623
80x10	**369103**	369140	424	392307	3593	**1319790**	1320270	424	4221250	3624	**1055420**	1057383	382	4221250	3599
100x10	**766271**	766329	371	1941110	3619	**2595660**	2597305	400	6608060	3628	**2168350**	2171574	280	6608070	3602
100x15	**739248**	739312	342	1941130	3608	**2490820**	2493656	369	6608060	3654	**2175300**	2179649	368	6608070	3639

where the MILP model fails to converge within the time limit, HGS consistently produces superior objective values. These findings highlight that HGS consistently outperforms the MILP model, delivering superior solution quality with significantly higher computational efficiency for large-scale instances.

OS.5. Sensitivity analysis

This section examines the robustness of the proposed SimHGS and the impact of key parameters. Section OS.5.1 analyzes the performance of SimHGS under different uncertainty levels. Section OS.5.2 investigates the trade-off between truck tardiness and untransferred goods by varying the unit tardiness cost, and Section OS.5.3 studies the effect of scenario size on neighborhood evaluation accuracy.

OS.5.1. Sensitivity analysis of uncertainty levels

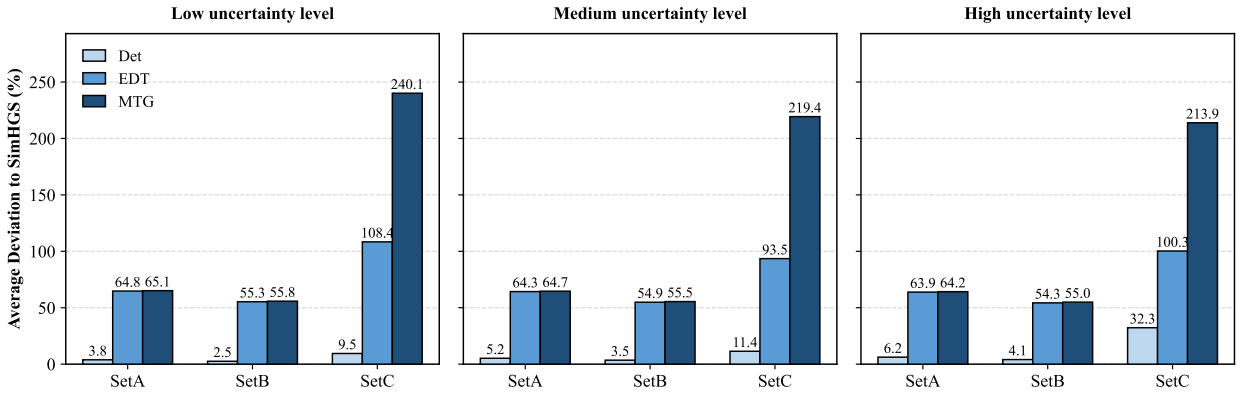


Figure OS.2 Average deviations of baseline policies from SimHGS under different uncertainty levels.

To ensure that our findings are not contingent on a specific arrival-time distribution, we evaluate the baseline policies across three distinct uncertainty levels: low ($Tri(-3,3,0)$), medium ($Tri(-6,6,0)$), and high ($Tri(-9,9,0)$). Figure OS.2 illustrates the average deviations of Det, EDT, and MTG relative to SimHGS across SetA, SetB, and SetC. The results indicate that the relative performance ranking of all methods remains remarkably stable across all tested uncertainty levels. Specifically, SimHGS consistently outperforms Det, EDT, and MTG in every scenario. This superiority is most pronounced in SetC, where EDT and MTG exhibit substantial performance gaps compared to SimHGS. Notably, the performance of the Det policy deteriorates significantly as uncertainty increases, whereas SimHGS maintains its lead. These observations confirm that the advantages of SimHGS are robust to variations in arrival-time distribution parameterization, reinforcing its reliability in diverse uncertain environments.

OS.5.2. Sensitivity analysis of objective weights

To evaluate the trade-off between service quality (truck tardiness) and throughput (untransferred goods), we conduct a sensitivity analysis on the unit tardiness cost c. Using the SimHGS algorithm, we vary $c \in \{1, 10, 20, 30, 40, 50\}$ across three representative instances (30x6A, 30x6B, 30x6C). As shown in Figure OS.3, the left y-axis represents the raw average total tardiness in time units rather than the weighted cost ($c \times$ tardiness). This choice provides a direct and intuitive assessment of operational performance. By excluding the scaling effect of c, we avoid the mathematical inflation of objective values that occurs as the weight increases, thereby providing a clearer visualization of the actual reduction in physical delays.

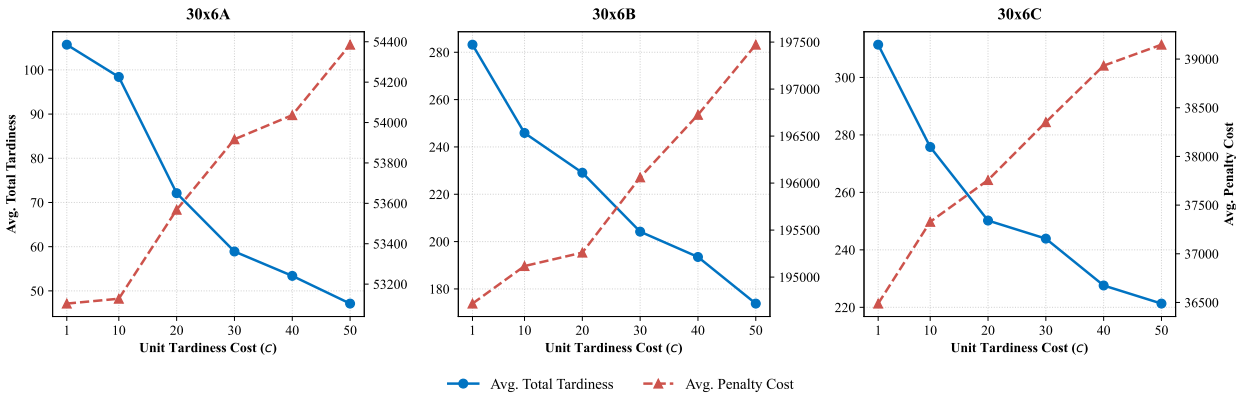


Figure OS.3 The unit tardiness cost analysis for different c

As illustrated in Figure OS.3, a distinct trade-off emerges between the two competing objectives. As the unit tardiness cost c increases, the objective function prioritizes adherence to scheduled departure times, resulting in a monotonic reduction in average tardiness (represented by solid blue lines). However, this stricter enforcement of schedule punctuality comes at the expense of throughput, as evidenced by the concomitant rise in penalties for untransferred goods (dashed red lines). These results demonstrate that SimHGS is highly sensitive to management priorities, allowing for a seamless calibration between strict schedule adherence and maximized throughput. This adaptability confirms the model's effectiveness in supporting decision-making under diverse cost structures and operational strategies.

OS.5.3. Sensitivity analysis of scenario size

To determine the scenario size used in SBLS, we conducted a sensitivity analysis on neighborhood evaluation with $N_s \in \{100, 500, 1000, 1500\}$. For each N_s, the best move identified during a neighborhood search iteration was compared with a benchmark move obtained under a reference evaluation with $N_{ref} = 2000$ scenarios. A misclassification was recorded whenever the two evaluations yielded different best moves. The resulting loss was quantified by the regret (the difference in objective values under the reference evaluation) and its corresponding percentage deviation. For each instance, these metrics were averaged over all neighborhood search iterations during the descent process across 10 independent runs. As illustrated in Figure OS.4, the percentage deviation decreases as N_s increases across all benchmark sets. The reduction is pronounced when N_s increases from 100 to 1000; however, the marginal improvement from 1000 to 1500 is notably smaller. These observations indicate diminishing marginal returns when the scenario size exceeds 1000, justifying the selection of $N_s = 1000$ as an effective balance between evaluation accuracy and computational overhead.

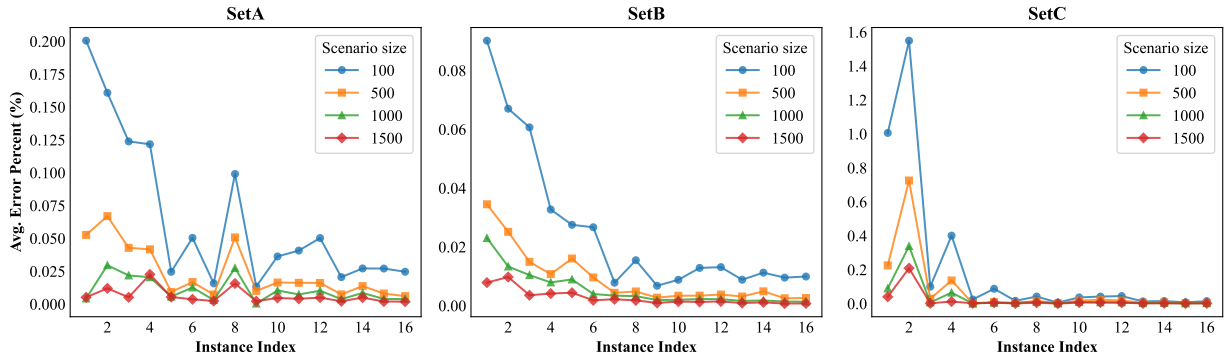
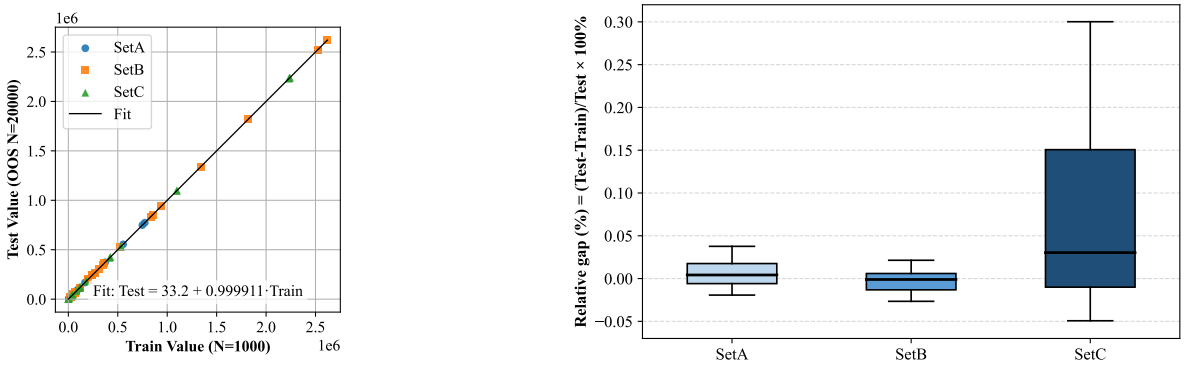


Figure OS.4 The average error percentage for different scenario size across all instances



(a) Consistency between in-sample and out-of-sample objective values

(b) Distribution of relative gaps across instance sets (SetA–SetC)

Figure OS.5 Out-of-sample ($N = 20000$) validation of the scenario-based evaluation ($N = 1000$)

To assess whether $N_s = 1000$ ensures robust out-of-sample performance, we evaluated the resulting scheduling policies against an independent test pool of 20,000 scenarios. Figure OS.5a reveals a near-perfect

linear correlation (Test $\approx 0.9999 \times$ Train), suggesting that the in-sample evaluations are unbiased estimators of the true objective values. Furthermore, Figure OS.5b further shows a close alignment between the two values within a remarkably tight margin of $\pm 0.05\%$. Collectively, these results justify the use of $N_s = 1000$ in SBLS as an optimal equilibrium between statistical precision and computational tractability.

OS.6. Component effectiveness of SimHGS

This section analyzes the contribution of the main components of SimHGS. Section OS.6.1 breaks down the runtime and evaluates the efficiency of the fast evaluation technique. The effectiveness of the cutoff rule is then assessed in Section OS.6.2. Section OS.6.3 quantifies the contribution of the simulation-based local search, while Section OS.6.4 evaluates the role of the hybrid genetic search framework.

OS.6.1. Runtime breakdown and efficiency analysis

To provide quantitative evidence for the efficacy of the fast evaluation technique, Figure OS.6 illustrates the runtime decomposition of SimHGS across all 72 instances, categorized into VND, deterministic LS, and SBLS. Additionally, it reports the average percentage of affected scenarios (Aff.Scen.) requiring updates during the fast evaluation process.

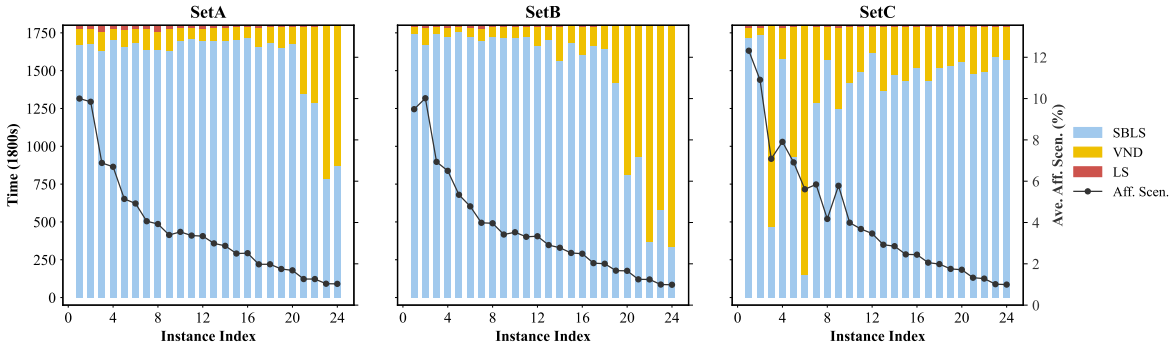


Figure OS.6 SimHGS runtime breakdown and average percentage of affected scenarios across all instances.

As shown in Figure OS.6, the proportion of affected scenarios remains consistently low across all instances, typically ranging from 1% to 12%. This confirms that only a small fraction of the scenario set necessitates re-evaluation after each timing move. This empirical observation aligns with the results presented in Section 4.4.2 and explains why the proposed technique achieves a speedup of 1–2 orders of magnitude over full evaluation, as reported in Section 6.5.

Figure OS.6 further elucidates the computational bottlenecks of SimHGS. For SetA and SetB, the computational effort dedicated to VND generally scales with problem size, suggesting that sequence optimization becomes the primary bottleneck for larger instances. In contrast, for SetC, the deterministic LS accounts for a disproportionately larger share of the total runtime (averaging 12 seconds for 100-truck instances in SetC, versus 4 seconds and 2 seconds for SetA and SetB). This indicates that the cutoff rule is less restrictive on SetC, allowing more candidate sequences to bypass the initial filter and enter the local improvement

phase. The resulting increase in LS and SBLS invocations elevates the overall computational burden for this set. Collectively, these results complement the theoretical and empirical analyses in Sections 4.4.2 and 6.5, providing a granular understanding of how the fast evaluation technique mitigates the simulation-based computational overhead.

OS.6.2. Effectiveness of the cutoff rule

In the SimHGS algorithm, only solutions that satisfy the cutoff rule (promising deterministic solutions $\boldsymbol{\pi}$) trigger the optimization of the corresponding release times $\boldsymbol{\tau}$. This selective strategy enhances the overall search performance. To verify the effectiveness of the cutoff rule, we compare SimHGS with NoCutoff, a variant that disables the cutoff rule.

Table OS.5 Summary of comparison results between SimHGS and NoCutoff

Instances	$BestObj$			$AvgObj$		
	#Wins	#Ties	#Losses	#Wins	#Ties	#Losses
SetA	13	8	3	18	5	1
SetB	20	3	1	19	2	3
SetC	14	2	8	21	0	3
Total	47	13	12	58	7	7
p-value	2.02E-06			5.14E-10		

Table OS.5 summarizes the comparison between the SimHGS and NoCutoff variants in terms of best and average objective values. The analysis shows that SimHGS outperforms NoCutoff for 47 out of 72 instances for the best objective value, while performing worse for 12 instances. For the average objective value, SimHGS achieves better results for 58 instances and worse for 7 instances. Interestingly, most of the instances where SimHGS underperforms are small instances with no more than 50 trucks. This suggests that although the cutoff rule may occasionally exclude some promising solutions, its advantages in improving search efficiency for large instances clearly outweigh these limitations. The benefits are especially pronounced in the more challenging SetB and SetC, where SimHGS consistently delivers stronger performance. Overall, the cutoff rule proves effective in filtering out unpromising truck sequences $\boldsymbol{\pi}$, thereby boosting the algorithm's overall performance across all instance types.

OS.6.3. Effectiveness of the SBLS

To quantify the contribution of SBLS beyond deterministic scheduling, we conducted a paired comparison using the final best policies identified by SimHGS. Let $\Pi^* = (\boldsymbol{\pi}, \boldsymbol{\tau}^*)$ denote the best policy after SBLS refinement, and let $\Pi = (\boldsymbol{\pi}, \boldsymbol{\tau})$ represent its deterministic predecessor, i.e., the policy immediately prior to the application of SBLS. By construction, both policies share an identical truck sequence $\boldsymbol{\pi}$ and differ exclusively in their timing decisions. Both Π^* and Π were evaluated on the same training set of 1000 scenarios. The relative improvement attributable to SBLS over the deterministic local search (LS) is calculated as $gap = \frac{LS-SBLS}{LS} \times 100\%$. As illustrated in Figure OS.7, SBLS consistently yields positive improvements

across all benchmark sets, with average gains of 4.47% on SetA, 2.75% on SetB, and 15.05% on SetC. Since the sequence π is unchanged, these performance gains are entirely driven by the stochastic refinement of timing decisions. This demonstrates that SBLS enhances the quality of the final SimHGS solutions by effectively adapting deterministic schedules to accommodate stochastic arrival time uncertainties.

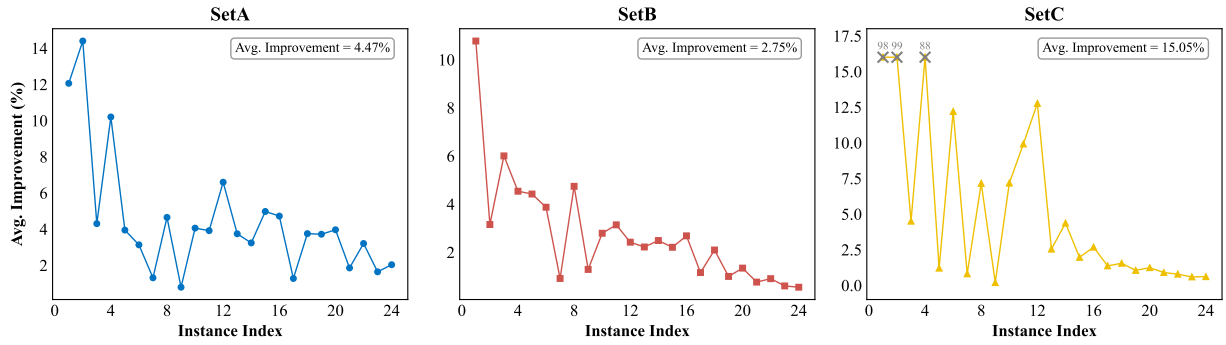


Figure OS.7 The average improvement of SBLS over LS across all instances

OS.6.4. Effectiveness of the HGS

To evaluate the contribution of the hybrid genetic search (HGS), we compare SimHGS with MS-NSBLS, a multi-start variant that repeatedly applies the nested simulation-based local search (NSBLS) from diverse initial solutions. Both algorithms are executed under a strict time limit of 1,800 seconds. This comparison is designed to distinguish the benefits of the population-based evolutionary mechanism from those achieved purely through independent local search iterations.

The summarized results in Table OS.6 reveal a consistent and statistically significant superiority of SimHGS across all benchmarks. For the best objective value, SimHGS outperforms MS-NSBLS for 57 instances, ties for 12 instances, and loses for 3 instances, yielding a highly significant p-value of 1.16E-10. The performance gap is similarly pronounced for the average objective value, where SimHGS wins for 60 instances, ties for 8 instances, and loses for 4 instances (p-value = 4.00E-11). These findings demonstrate that SimHGS's superior performance is not merely due to multiple restarts. Instead, the population-based mechanism effectively maintains diversity and facilitates information exchange between solutions. This synergy allows the algorithm to explore promising regions of the solution space that remain inaccessible to isolated local searches, thereby confirming HGS as an essential component of our methodology.

Table OS.6 Summary of comparison results between SimHGS and MS-NSBLS

Instances	*BestObj*			*AvgObj*		
	#Wins	#Ties	#Losses	#Wins	#Ties	#Losses
SetA	15	6	3	16	5	3
SetB	22	2	0	21	2	1
SetC	20	4	0	23	1	0
Total	57	12	3	60	8	4
p-value	1.16E-10			4.00E-11		