New Dynamic Discretization Discovery Strategies for Continuous-Time Service Network Design

Shengnan Shu
Faculty of Business for Science and Technology, School of Management, University of Science and Technology of China, shengnan.shu@connect.polyu.hk

Zhou Xu
Department of Logistics and Maritime Studies, the Hong Kong Polytechnic University, zhou.xu@polyu.edu.hk

Roberto Baldacci
College of Science and Engineering, Hamad Bin Khalifa University, Qatar Foundation, Doha, Qatar, rbaldacci@hbku.edu.qa

Abstract. Service Network Design Problems (SNDPs) are prevalent in the freight industry. While the classic SNDP is defined on a discretized planning horizon with integral time units, the Continuous-Time SNDP (CTSNDP) uses a continuous-time horizon to avoid discretization errors. Existing CTSNDP algorithms primarily rely on the Dynamic Discretization Discovery (DDD) framework, which iteratively refines discretization and constructs a partially time-expanded network based on the discretization to derive relaxation and feasible solutions. However, existing DDD algorithms struggle with computational performance, particularly for instances with a high vehicle-to-flow cost ratio and high time flexibility, leaving many instances unsolved. This study enhances the DDD solution framework by introducing three new optimization-based strategies: (i) a new initial relaxation strategy based on timed-node-based time-expanded commodity networks and significant time points identified by solving minimum hitting-set problems; (ii) a new mixed-integer-programming (MIP)-based upper-bound strategy that leverages newly defined consolidation planning problems; and (iii) a new joint minimal-infeasible-pattern-elimination strategy for refining discretization by solving a newly introduced joint refinement problem. Computational experiments demonstrate that our enhanced DDD algorithm achieves exceptional performance, optimally solving all classic CTSNDP instances within one hour. These results validate the effectiveness of the proposed strategies in addressing the limitations of existing DDD algorithms.

Key words: service network design; continuous time; dynamic discretization discovery; exact algorithm

1. Introduction

The Service Network Design Problem (SNDP) is a well-known optimization problem that focuses on planning transportation operations for carriers specializing in managing small shipments relative to vehicle capacities (Crainic 2000, Wieberneit 2008, Crainic and Hewitt 2021). This problem is particularly prevalent in industries such as parcel and small package delivery, as well as Less-Than-Truckload (LTL) freight. LTL carriers support numerous supply chains, while parcel and small-package carriers are essential to facilitating e-commerce sales. These industries collectively contribute billions of dollars to the global economy, exemplified by the LTL industry's $46 billion valuation in the United States in 2021 and UPS's reported revenue of $69.44 billion from its US and international small package operations in 2020 (Hewitt and Lehuédé 2023). These figures underscore the economic significance and scale of the industries involved in the SNDP.

The SNDP has been extensively explored in the literature, with researchers investigating numerous variants owing to its applicability across diverse industries. The classic SNDP aims to establish the delivery

path for each shipment within a terminal network and determine suitable dispatch times for terminal-to-terminal movements on this path. This must be done while adhering to delivery timeframes and minimizing the overall transportation costs. Shipments are consolidated when multiple shipments are scheduled for simultaneous dispatch from the same terminal.

A common technique in the literature for modeling the SNDP is *discretization*, which divides the planning horizon into discrete time units or points. The problem can then be formulated as an Integer Programming (IP) model on a discretized *time-expanded network*. In this network, nodes represent locations in both time and space, and arcs represent physical movements between locations or within a single location over time. The formulation identifies consolidation opportunities when multiple shipments travel on the same arc in the time-expanded network. It ensures that enough vehicles are available on the same arc to serve the consolidation by utilizing knapsack-type linking constraints. However, the granularity of the time discretization in the time-expanded network significantly affects the model's computational tractability and the quality of the solutions obtained. This complicates accurate scheduling and shipment consolidation on a continuous-time planning horizon, known as the Continuous-Time SNDP (CTSNDP).

The CTSNDP poses significant computational challenges due to the continuous nature of time and the increased complexity of the optimization problem. To address this, Boland et al. (2017) proposed a Dynamic Discretization Discovery (DDD) algorithm that iteratively refines the time discretization of the time-expanded network until the proven optimal solution is obtained. Specifically, the DDD algorithm begins with an initial discretization and, at each iteration, updates the lower and upper bounds on the optimal solution cost. The lower bound is obtained by solving a relaxation model based on the current discretization. In contrast, the upper bound is derived by finding a feasible solution based on the relaxation model's solution. If the current upper bound cannot be proven optimal based on the current lower bound, the algorithm refines the time discretization of the time-expanded network by adding new *timed nodes*. The DDD algorithm has been extended to address the CTSNDP in an *interval-based time-expanded network*, where nodes represent locations in time intervals and space (Marshall et al. 2021). It has also been applied to variants of the CTSNDP (He et al. 2022a, Shu et al. 2024).

However, existing DDD algorithms for the CTSNDP face difficulties when dealing with instances characterized by a high vehicle-to-flow cost ratio and high time flexibility, with nearly half of these instances still not solved to optimality. The limitations of current DDD algorithms in efficiently solving such instances stem from several factors. The first factor is weak relaxation. A loose relaxation allows numerous impossible consolidations, as indicated by the available and due times of shipments. Consequently, this can lead to weak relaxation solutions with many infeasible consolidations, thereby making it difficult to prove the solution's optimality within a reasonable computational time. The second factor is the ineffectiveness of heuristic methods for deriving upper-bound solutions. All existing DDD algorithms employ a heuristic approach to derive upper-bound solutions, which cannot guarantee the computation of high-quality, feasible

solutions, thus impeding the algorithm's convergence. The third factor is the inefficiency of the refinement methods. Existing refinement methods often fail to achieve a sufficiently fine discretization or require excessive iterations, thereby adversely affecting the algorithm's overall performance.

This paper proposes a new DDD algorithm for the CTSNDP to address the aforementioned challenges. While adopting a similar solution framework to previous DDD algorithms, this approach distinguishes itself by introducing several novel optimization-based strategies for initial relaxation, upper-bound derivation, and discretization refinement. Specifically, the new DDD algorithm features: (i) A new initial relaxation strategy using timed-node-based commodity time-expanded networks and significant time points identified by solving minimum hitting-set problems; (ii) A new mixed-integer-programming (MIP)-based upper-bound derivation strategy that leverages consolidation planning problems defined over a collection of relaxation solutions; and (iii) A new joint minimal-infeasible-pattern-based strategy for refining discretization that generates the final refinement time points by solving newly defined refinement problems. Computational results show that our new DDD algorithm significantly outperforms existing methods, becoming the first to optimally solve all benchmark CTSNDP instances reported in the literature within a one-hour time limit.

The remainder of the paper is organized as follows. Section 2 reviews the relevant literature. Section 3 provides a formal description of the CTSNDP. An overview of the newly developed enhanced DDD algorithm for the CTSNDP is presented in Section 4, with the key methodological results and contributions summarized in Section 4.2. The details of the enhanced DDD algorithm are described in Sections 5-7. Specifically, we introduce the new initial relaxation (§5), illustrate a MIP-based approach for upper-bound computation (§6), and detail the new discretization refinement strategy (§7). We then present the computational studies in Section 8. Finally, the paper is concluded in Section 9. Proofs of the statements and additional materials are available in the e-companion to this paper.

2. Related Work

The SNDP has been a focal point of research in the operations research community since the 1990s, given its extensive practical applications and theoretical significance (Crainic and Rousseau 1986, Farvolden and Powell 1994). The SNDP is closely associated with the *multicommodity flows over time* problem, which involves routing and scheduling multiple commodities through a network while accounting for their release and due times. The multicommodity network flow problem addresses routing multiple commodities through a network, assuming instantaneous flow travel and disregarding flow changes over time (Ford Jr and Fulkerson 1958b). To incorporate flow changes over time, the concept of dynamic flows, also known as *flows over time*, was introduced by Ford Jr and Fulkerson (1958a) for single commodities within a discrete-time framework. They considered a maximal dynamic flow problem over a network with transit times on the arcs, specifying the time it takes to pass through each arc, aiming to maximize the flow from a source to a sink within a specified time horizon. They demonstrated that a flow-over-time problem in a network with

transit times can be formulated as a linear program over a time-expanded network. Fleischer and Tardos (1998) extended this work to continuous time, addressing dynamic flows in a continuous-time setting. Furthermore, Hall et al. (2007) extended the notion of flows over time to multicommodity cases and proved the $\mathcal{NP}$-hardness of the multicommodity flows over time problem. A comprehensive overview of this research area can be found in Skutella (2009).

Similar to approaches taken in multicommodity flows over time, existing studies on the SNDP have primarily approximated the temporal component of the problem using a *time-indexed formulation* on a discretized, time-expanded network (Crainic 2000, Andersen et al. 2009, Crainic and Hewitt 2021). However, selecting an appropriate discretization level poses a significant challenge. To assist in this decision within the SNDP, Boland et al. (2019) proposed metrics to estimate the price of discretization and introduced a mechanism for selecting a suitable discretization level based on a given tolerance. Even with careful discretization, some time-expanded network models become too large to solve efficiently, prompting many studies to develop heuristics (Pedersen et al. 2009, Teypaz et al. 2010, Erera et al. 2013).

The CTSNDP has garnered significant attention in recent years, with breakthroughs in developing exact algorithms based on discretized time-indexed formulations to obtain continuous-time optimal solutions. Boland et al. (2017) introduced a DDD algorithm to solve the CTSNDP optimally. The DDD algorithm addresses the problem by solving a sequence of relaxation models defined on a subset of time points (i.e., a partial discretization), with variables indexed by the subset's time points providing lower bounds on the optimal objective function value. New times are discovered and used to refine the partial discretization at each iteration. Once the correct subset of time points is discovered, the resulting relaxation model yields the optimal continuous-time solution. Following this work, Marshall et al. (2021) proposed a similar DDD algorithm for the CTSNDP, but based on an interval-based network where nodes are attributed by time intervals and space, as opposed to the method proposed by Boland et al. (2017), which uses a network with nodes labeled with time points. This DDD algorithm has been successfully extended to several variants of the CTSNDP. For instance, He et al. (2022a) employed the DDD algorithm to address the CTSNDP variant with hub capacity constraints, while Hewitt (2022) and Shu et al. (2024) extended the DDD framework to tackle CTSNDP variants considering flexible commodity available and due times and holding costs, respectively. Additionally, Van Dyk and Koenemann (2024) adopted the DDD framework for the CTSNDP, utilizing a network with arc-dependent time discretizations and restricted routes for each commodity. Specifically, they optimized the time-expanded network by assigning distinct departure time points to different services departing from the same terminal, whereas the general time-expanded network shares a common set of departure time points for all services from the same terminal. However, similar sparsity can be achieved by safely removing certain arcs based on specific reduction rules for the general time-expanded network, as demonstrated in Marshall et al. (2021). The DDD framework exhibits versatility extending beyond the

CTSNDP, being applied to several transportation-related problems, including the Time-Dependent Traveling Salesman Problem with Time Windows (Vu et al. 2020), the Time-Dependent Shortest Path Problem (He et al. 2022b), the Continuous-time Load Plan Design Problem (Hewitt 2019), the Continuous-time Inventory Routing Problem (Lagos et al. 2022), and the Two-echelon Location Routing Problem (Escobar-Vargas and Teodor Gabriel 2024), among others. For further insights into time-dependent models and an introduction to DDD algorithms, interested readers can refer to Boland and Savelsbergh (2019). The research presented here complements and extends the work in Boland et al. (2017) and Marshall et al. (2021), providing a robust foundation to enhance DDD algorithms for CTSNDP variants and other transportation problems.

Moreover, for the CTSNDP variant with predefined commodity delivery paths, Hewitt and Lehuédé (2025) employ a consolidation-based formulation that requires enumerating all shipment combinations, and a new dynamic column-generation algorithm. Our experimental results (see §EC.4.1) show that the enhanced DDD algorithm also provides significant computational advantages for this variant.

3. Problem Description

The SNDP can be defined as follows. We are given a directed graph $\mathcal{D} = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N}$ represents the physical terminal set and set $\mathcal{A}$ indicates the arc set, and a set $\mathcal{K}$ of commodities which must be routed through the network $\mathcal{D}$. The network $\mathcal{D}$ is referred to as the *flat network* to distinguish it from the time-expanded network typically used in the problem's modeling method. In the flat network $\mathcal{D}$, each arc (i,j) is characterized by its travel time $\tau_{i,j} \in \mathbb{N}_{>0}$, a per-unit-of-flow cost $c^k_{i,j} \in \mathbb{R}_{>0}$ for each commodity $k \in \mathcal{K}$, a fixed cost $f_{i,j} \in \mathbb{R}_{>0}$ for each dispatch of a vehicle on the arc, and a capacity $u_{i,j} \in \mathbb{N}_{>0}$ for each vehicle served on the arc. Each commodity $k \in \mathcal{K}$ is defined by a single origin $o^k \in \mathcal{N}$, a single destination $d^k \in \mathcal{N}$, a transportation demand $q^k \in \mathbb{N}_{>0}$, and a time window $[e^k, l^k]$, where $e^k \in \mathbb{N}_{\geq 0}$ is the earliest available time at its origin, and $l^k \in \mathbb{N}_{>0}$ denotes the due time for arriving at its destination. Each demand q^k must be delivered from the origin to the destination along a single delivery path. Multiple demands can be consolidated to pass through arc (i,j), thereby reducing the total fixed cost associated with using the service on (i,j). For all practical purposes, we assume that the travel times and time window restrictions are integers and $\min_{k\in\mathcal{K}}\{e^k\} = 0$. We use (i,j) and a interchangeably to represent an arc in $\mathcal{A}$.

The SNDP seeks to determine the routing and consolidation plans for the commodities and the required services/resources to execute them. It aims to minimize total flow and fixed costs while ensuring compliance with time constraints on the commodities.

The SNDP is typically modeled using a time-indexed formulation based on a time-expanded network. To achieve this, we define a time-expanded network $\mathcal{D}^\Delta_\mathcal{T} = (\mathcal{N}^\Delta_\mathcal{T}, \mathcal{A}^\Delta_\mathcal{T} \cup \mathcal{H}^\Delta_\mathcal{T})$ with a given discretization parameter Δ. Here, $\mathcal{T} = \{\mathcal{T}_i\}_{i\in\mathcal{N}}$ represents the set of *time points* for all $i \in \mathcal{N}$, where $\mathcal{T}_i = \{0, \Delta, 2\Delta, ..., \Delta\lceil \max_{k\in\mathcal{K}} l^k/\Delta\rceil\}$. Time points in $\mathcal{T}_i$, $i \in \mathcal{N}$, are also represented by set $\{t^i_1, t^i_2, \ldots, t^i_{n_i}\}$ where $n_i = |\mathcal{T}_i|$. The node set $\mathcal{N}^\Delta_\mathcal{T}$ consists of a node (i,t) for each $i \in \mathcal{N}$ and $t \in \mathcal{T}_i$. The set of arcs in

$\mathcal{D}_\mathcal{T}^\Delta$ includes two subsets of *timed arcs*. (i) *Holding arcs* $\mathcal{H}_\mathcal{T}^\Delta$: For every terminal $i \in \mathcal{N}$ and every $n \in \{1, ..., n_i - 1\}$, there exists an arc $((i, t_n^i), (i, t_{n+1}^i))$ representing a holding of $(t_{n+1}^i - t_n^i)$ time units at terminal i. (ii) *Service arcs* $\mathcal{A}_\mathcal{T}^\Delta$: For every arc $(i, j) \in \mathcal{A}$ and node $(i, t) \in \mathcal{N}_\mathcal{T}^\Delta$, there exists an arc $((i, t), (j, \bar{t}))$ with $\bar{t} = t + \Delta\lceil \tau_{ij}/\Delta \rceil$, representing a dispatch from terminal i at time t arriving at time $\bar{t}$ at terminal j.

The time-indexed formulation of the SNDP can be modeled based on the graph $\mathcal{D}_\mathcal{T}^\Delta$, which features two types of decision variables. Flow variables $x_{i,j}^{kt\bar{t}}$ are binary variables equal to 1 if commodity $k \in \mathcal{K}$ is routed along arc $(i, j) \in \mathcal{A}$, departing at time t and arriving at j, and 0 otherwise. Design variables $y_{i,j}^{t\bar{t}}$ are nonnegative integer variables representing the number of vehicles on arc $(i, j) \in \mathcal{A}$ required to serve the commodities that dispatch from terminal i at time t and arrive at time $\bar{t}$ in j.

The SNDP seeks to optimize the following model SND($\mathcal{D}_\mathcal{T}^\Delta$).

$$z(\mathcal{D}_\mathcal{T}^\Delta) = \min \sum_{((i,t),(j,\bar{t}))\in\mathcal{A}_\mathcal{T}^\Delta} f_{i,j} y_{i,j}^{t\bar{t}} + \sum_{k\in\mathcal{K}} \sum_{((i,t),(j,\bar{t}))\in\mathcal{A}_\mathcal{T}^\Delta} (c_{i,j}^k q^k) x_{i,j}^{kt\bar{t}} \tag{1}$$

$$\text{s.t.} \sum_{((i,t),(j,\bar{t}))\in\mathcal{A}_\mathcal{T}^\Delta \cup \mathcal{H}_\mathcal{T}^\Delta} x_{i,j}^{kt\bar{t}} - \sum_{((j,\bar{t}),(i,t))\in\mathcal{A}_\mathcal{T}^\Delta \cup \mathcal{H}_\mathcal{T}^\Delta} x_{j,i}^{k\bar{t}t} = \begin{cases} 1 & (i,t) = (o^k, e^k), \\ -1 & (i,t) = (d^k, l^k), \\ 0 & \text{otherwise}, \end{cases} \quad \forall k \in \mathcal{K}, (i,t) \in \mathcal{N}_\mathcal{T}^\Delta, \tag{2}$$

$$\sum_{k\in\mathcal{K}} q^k x_{i,j}^{kt\bar{t}} \le u_{i,j} y_{i,j}^{t\bar{t}}, \quad \forall\, ((i,t),(j,\bar{t})) \in \mathcal{A}_\mathcal{T}^\Delta, \tag{3}$$

$$x_{i,j}^{kt\bar{t}} \in \{0,1\}, \quad \forall k \in \mathcal{K}, ((i,t),(j,\bar{t})) \in \mathcal{A}_\mathcal{T}^\Delta \cup \mathcal{H}_\mathcal{T}^\Delta, \tag{4}$$

$$y_{i,j}^{t\bar{t}} \in \mathbb{N}_{\ge 0}, \quad \forall ((i,t),(j,\bar{t})) \in \mathcal{A}_\mathcal{T}^\Delta. \tag{5}$$

The objective function (1) aims to minimize the total cost, computed as the sum of fixed and flow costs, respectively. Constraints (2) represent *flow conservation constraints*, ensuring that each commodity $k \in \mathcal{K}$ is routed along a single path starting from its origin after it becomes available and ending at its destination before its due time. Without loss of optimality, it is assumed that all commodities passing through the same service arc $((i,t),(j,\bar{t})) \in \mathcal{A}_\mathcal{T}^\Delta$ can be consolidated. Constraints (3) represent *capacity constraints*, ensuring that sufficient capacity is available to serve the flows on each service arc $((i,t),(j,\bar{t})) \in \mathcal{A}_\mathcal{T}^\Delta$. Finally, constraints (4) and (5) specify the domains of the variables. For presentational convenience, we assume that the nodes (o_k, e_k) and (d_k, l_k) are contained in $\mathcal{N}_\mathcal{T}^\Delta$ for all $k \in \mathcal{K}$. Otherwise, nodes (o^k, t) with $t = \arg\min\{s \in \mathcal{T}_{o^k} : s \ge e^k\}$ and (d^k, t') with $t' = \arg\max\{s \in \mathcal{T}_{d^k} : s \le l^k\}$ can be used instead in (2).

The discretization parameter Δ introduces deviations to travel times, available times, and due times, inevitably leading to approximations in the problem. The key to the modeling technique based on a time-expanded network lies in the discretization parameter Δ. Indeed, the quality of the approximate solution heavily depends on the choice of Δ for the time-expanded network. Fine discretizations typically provide good approximations to the continuous-time problems. Still, they may yield large, potentially intractable integer programs, whereas coarse discretizations are more computationally tractable but generally yield poorer approximations. According to Boland et al. (2017), no approximations are introduced when Δ takes

a value such that all values τ_{ij}/Δ, e^k/Δ, and l^k/Δ are naturally integers. Let $\hat{\Delta}$ be the greatest common divisor of all τ_{ij}, e^k, and l^k. In the worst case, $\hat{\Delta}$ takes a value of 1. $\mathcal{D}^{\hat{\Delta}}_{\hat{\mathcal{T}}}$ is a *fully time-expanded network* such that $z(\mathcal{D}^{\hat{\Delta}}_{\hat{\mathcal{T}}})$ provides the optimal solution cost of the CTSNDP, and SND($\mathcal{D}^{\hat{\Delta}}_{\hat{\mathcal{T}}}$) becomes a *complete time-indexed model* for the CTSNDP. The corresponding discretization $\hat{\mathcal{T}}$ is referred to as the *complete discretization*. We thus define the CTSNDP as SND($\mathcal{D}^{\hat{\Delta}}_{\hat{\mathcal{T}}}$). As shown in Boland et al. (2017), this fully time-expanded network $\mathcal{D}^{\hat{\Delta}}_{\hat{\mathcal{T}}}$ can be further reduced to include only time points e^k for some commodity k or time points of the form $e^k + \sum_{a \in p} \tau_a$ for some commodity k and some path $p \subseteq \mathcal{A}$ originating at o^k.

4. An Enhanced Dynamic Discretization Discovery Algorithm: Overview

This section presents an overview of the enhanced DDD algorithm developed for the CTSNDP, denoted as EDDD. It follows the DDD framework proposed by Boland et al. (2017), summarized in Section 4.1, but differs in key algorithmic components, as outlined in Section 4.2.

4.1. DDD Framework in the Literature

With the fully time-expanded network $\mathcal{D}^{\hat{\Delta}}_{\hat{\mathcal{T}}}$, the optimal solution for the CTSNDP can be obtained by solving SND($\mathcal{D}^{\hat{\Delta}}_{\hat{\mathcal{T}}}$). The size of the fully time-expanded network $\mathcal{D}^{\hat{\Delta}}_{\hat{\mathcal{T}}}$ can be prohibitively large for practical instances, and solving the resulting SND($\mathcal{D}^{\hat{\Delta}}_{\hat{\mathcal{T}}}$) becomes challenging.

Boland et al. (2017) proposed a DDD algorithm, which iteratively expands the size of the time-expanded network. The DDD algorithm utilizes the concept of a *partially time-expanded network*, denoted by $\underline{\mathcal{D}_{\mathcal{T}}} = (\underline{\mathcal{N}_{\mathcal{T}}}, \underline{\mathcal{A}_{\mathcal{T}}} \cup \underline{\mathcal{H}_{\mathcal{T}}})$. Here, $\underline{\mathcal{N}_{\mathcal{T}}}$ represents a subset of the set of all possible timed nodes $\mathcal{N}^{\hat{\Delta}}_{\hat{\mathcal{T}}}$, $\underline{\mathcal{H}_{\mathcal{T}}}$ comprises the timed-node pairs connecting consecutive timed nodes at the same node in the flat network, and $\underline{\mathcal{A}_{\mathcal{T}}}$ is a set of timed-node pairs $(i,t),(j,t') \in \underline{\mathcal{N}_{\mathcal{T}}}$ for which $(i,j) \in \mathcal{A}$. Additionally, the network $\underline{\mathcal{D}_{\mathcal{T}}}$ satisfies the following Properties 1-4, introduced by Boland et al. (2017).

Property 1 *For all commodities $k \in \mathcal{K}$, nodes (o^k, e^k) and (d^k, l^k) are in $\underline{\mathcal{N}_{\mathcal{T}}}$.*

Property 2 *Every arc $((i,t),(j,\bar{t})) \in \underline{\mathcal{A}_{\mathcal{T}}}$ has $\bar{t} \leq t + \tau_{i,j}$.*

Property 3 *For every $(i,j) \in \mathcal{A}$ and every $(i,t) \in \underline{\mathcal{N}_{\mathcal{T}}}$, these exists a timed-copy arc $((i,t),(j,\bar{t})) \in \underline{\mathcal{A}_{\mathcal{T}}}$.*

Property 4 *For each arc $((i,t),(j,\bar{t})) \in \underline{\mathcal{A}_{\mathcal{T}}}$, there does not exist a node $(j,t') \in \underline{\mathcal{N}_{\mathcal{T}}}$ with $\bar{t} < t' \leq t + \tau_{i,j}$.*

By Properties 2–4, for each $(i,t) \in \underline{\mathcal{N}_{\mathcal{T}}}$ and each $(i,j) \in \mathcal{A}$, there exists exactly one arc of the form $((i,t),(j,\bar{t}))$ in $\underline{\mathcal{A}_{\mathcal{T}}}$, whose length may be shorter than the true travel time of its underlying physical arc. Such arcs whose lengths are shorter than the true travel times are referred to as *short arcs*. The DDD algorithm constructs initial sets $\underline{\mathcal{N}_{\mathcal{T}}}$, $\underline{\mathcal{H}_{\mathcal{T}}}$, and $\underline{\mathcal{A}_{\mathcal{T}}}$, usually in such a way that $\left|\underline{\mathcal{N}_{\mathcal{T}}}\right| << \left|\mathcal{N}^{\hat{\Delta}}_{\hat{\mathcal{T}}}\right|$, and modifies them at each iteration of the algorithm.

Building on the properties above, Boland et al. (2017) demonstrated that SND($\underline{\mathcal{D}_{\mathcal{T}}}$) is a valid relaxation of the CTSNDP, i.e., a feasible SND($\underline{\mathcal{D}_{\mathcal{T}}}$) solution $(\boldsymbol{x}, \boldsymbol{y})$ provides a *relaxation solution* to the CTSNDP. The DDD algorithm follows an iterative approach to solve the relaxation model, SND($\underline{\mathcal{D}_{\mathcal{T}}}$), providing an

optimal relaxation solution and a valid lower bound for the CTSNDP. It then derives a feasible solution based on the optimal relaxation solution, whose total cost is an upper bound on the optimal solution cost of the CTSNDP. The algorithm, therefore, dynamically updates the lower bound (LB) and upper bound (UB). If the obtained upper bound falls within a predetermined optimality tolerance, indicating that an optimal solution (within the specified tolerance) has been found for the CTSNDP, the DDD algorithm terminates. In cases where this condition is not met, the algorithm identifies short-arcs in $\underline{\mathcal{A}_{\mathcal{T}}}$ that need adjustment to their true travel times. This travel time adjustment is then achieved by incorporating new time points to the current discretization $\mathcal{T}$, and refining the corresponding partially time-expanded network $\underline{\mathcal{D}_{\mathcal{T}}}$ through further modifications to the arcs, ensuring adherence to the four defined properties.

4.2. Overview of Our EDDD Algorithm and Key Methodological Results

Algorithm 1 outlines the main steps of the EDDD algorithm. While it follows a similar iterative framework, the proposed EDDD algorithm differs substantially from the DDD of Boland et al. (2017) and its extension, the DDDI algorithm of Marshall et al. (2021), by introducing innovative optimization-based strategies for initial relaxation, upper-bound derivation, and discretization refinement, as detailed below.

- It uses a new initial relaxation (Step 2), SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$) (see §5), which is based on the timed-node-based time-expanded commodity networks for shipments. In contrast to the initial relaxations employed by Boland et al. (2017) and Marshall et al. (2021), our approach integrates a set of significant time points into the initial discretization. These significant time points eliminate impossible consolidations from the relaxation model, resulting in a tighter initial relaxation than those used by Boland et al. (2017) and Marshall et al. (2021). Finding these significant time points is formulated as a series of ***minimum hitting-set problems*** for intervals, which can be efficiently solved in linear time.

- It employs a novel MIP-based approach to compute an upper bound at each iteration (Step 5), leveraging ***consolidation planning problems*** newly defined over a collection of relaxation solutions (see §6). Unlike the heuristic methods utilized by Boland et al. (2017) and Marshall et al. (2021), it solves an explicit consolidation planning problem (see §6.1) to identify the best CTSNDP solution among all feasible solutions that adhere to the routing plans provided by the relaxation solution, representing a major advance over prior approaches. We present a MIP for this consolidation planning problem (see §6.2), which is further strengthened by symmetry-breaking techniques adapted from graph coloring problems (see §EC.3.2).

- It utilizes a new joint minimal-infeasible-pattern-based strategy for refining discretization (Step 10), which generates the final refinement time points by solving a newly defined ***refinement problem*** (see §7). Existing DDD algorithms refine the discretization $\mathcal{T}$ by adding time points to eliminate the optimal relaxation solution (Boland et al. 2017) or the non-implementable flat solution (Marshall et al. 2021) so that they no longer occur under a refined discretization $\mathcal{T}'$. While our new refinement method uses the same refinement operation of adding new time points to the discretization $\mathcal{T}$ (see §7.1 for the rationale behind our

Algorithm 1: EDDD Algorithm for the CTSNDP

Input: CTSNDP defined on a flat network $\mathcal{D} = (\mathcal{N}, \mathcal{A})$, $optimality\ tolerance$;
Output: Solution $\left(\{p^k\}_{k\in\mathcal{K}}, \{t^k\}_{k\in\mathcal{K}}\right)$ of cost UB;
begin
1 UB $\leftarrow +\infty$, LB $\leftarrow -\infty$;
2 (Initial Relaxation) Compute a minimum set of significant time points by solving minimum hitting-set problems to establish an initial discretization $\mathcal{T}$, initial time-expanded commodity networks $\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$, and an initial relaxation model SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$) (see §5);
3 **while** $(UB - LB)/UB > optimality\ tolerance$ **do**
4 (Lower Bound Computation) Solve SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$) to obtain a collection of relaxation solutions, including the optimal relaxation solution, and set LB equal to its optimal objective value;
5 (Upper Bound Computation) (i) For each relaxation solution obtained, solve a consolidation planning problem to obtain the best CTSNDP solution among all feasible solutions that utilize the routing plan given by the relaxation solution; (ii) Select the best of these best solutions as $\left(\{\hat{p}^k\}_{k\in\mathcal{K}}, \{\hat{t}^k\}_{k\in\mathcal{K}}\right)$ along with its objective value $\hat{z}$ (see §6);
6 **if** $\hat{z} < UB$ **then**
7 UB $\leftarrow \hat{z}$ and $\left(\{p^k\}_{k\in\mathcal{K}}, \{t^k\}_{k\in\mathcal{K}}\right) \leftarrow \left(\{\hat{p}^k\}_{k\in\mathcal{K}}, \{\hat{t}^k\}_{k\in\mathcal{K}}\right)$;
8 **end**
9 **if** $(UB - LB)/UB > optimality\ tolerance$ **then**
10 (Discretization Refinement) Solve a refinement problem to refine the discretization $\mathcal{T}$ and update the networks in $\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$ (see §7);
11 **end**
12 **end**
13 **return** Solution $\left(\{p^k\}_{k\in\mathcal{K}}, \{t^k\}_{k\in\mathcal{K}}\right)$ of cost UB;
14 **end**

approach in light of Marshall et al. (2021)), it focuses on a new type of structural pattern: minimal too-long paths in a newly defined dispatch-node graph (see §7.2). As the minimal structural patterns responsible for non-implementability, these paths may recur across multiple non-implementable flat solutions. However, existing methods typically refine selected too-long paths separately, despite their structural relationships. This suggests that effective refinement should target multiple minimal too-long paths that extend beyond those in the optimal relaxation solution and select refinement time points jointly rather than separately. Accordingly, we define a new refinement problem (see §7.3) that aims to jointly eliminate these minimal too-long paths derived from a set of relaxation solutions using a minimal set of time points. We then formulate this problem as a MIP model and develop a three-step matheuristic to solve it.

Our approach significantly enhances the initial relaxation, upper-bound derivation, and discretization refinement by solving well-defined optimization problems in each phase, thereby achieving the crucial milestone of optimally solving all classic CTSNDP instances documented in the literature.

5. Initial Relaxation from Commodity Networks and Significant Time Points

The quality of the relaxation directly impacts the convergence efficiency of the DDD algorithm. Existing DDD algorithms for the CTSNDP (Marshall et al. 2021) and variants of the CTSNDP, i.e., the CTSNDP with holding costs (Shu et al. 2024), have focused on generating valid inequalities to strengthen the relaxation model and consequently expedite the algorithm's convergence. The results of Marshall et al. (2021) and Shu et al. (2024) show that stronger relaxations can greatly speed up DDD convergence.

In this section, we propose a new strategy for the initial relaxation model based on timed-node-based time-expanded commodity networks that include specific significant time points to strengthen the lower bound and expedite the DDD algorithm's convergence. We first introduce timed-node-based time-expanded commodity networks in Section 5.1, to derive a tighter relaxation model SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$) of CTSNDP than SND($\underline{\mathcal{D}_{\mathcal{T}}}$). The relaxation model SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$) is further strengthened by incorporating a minimum set of significant time points to the initial discretization $\mathcal{T}$ (see Section 5.2).

5.1. Timed-Node-Based Time-Expanded Commodity Networks and Relaxation SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$)

First, following an approach similar to that in Marshall et al. (2021) for the interval-based time-expanded commodity network, we can define a timed-node-based time-expanded commodity network $\underline{\mathcal{D}^{k}_{\mathcal{T}}} = (\underline{\mathcal{N}_{\mathcal{T}}}, \underline{\mathcal{A}^{k}_{\mathcal{T}}} \cup \underline{\mathcal{H}_{\mathcal{T}}})$ for every commodity $k \in \mathcal{K}$, where each service arc set $\underline{\mathcal{A}^{k}_{\mathcal{T}}}$ consists of only arcs $((i,t),(j,\bar{t})) \in \underline{\mathcal{A}_{\mathcal{T}}}$ with $i \neq d^k$ and $j \neq o^k$ and satisfies the following two properties:

$$\begin{cases} \text{(i)} & e^k + \phi^k_{o^k,i} + \tau_{i,j} + \phi^k_{j,d^k} \leq l^k, \\ \text{(ii)} & \max\{s \in \mathcal{T}_i : s \leq e^k + \phi^k_{o^k,i}\} \leq t \leq \max\{s \in \mathcal{T}_i : s + \tau_{i,j} + \phi^k_{j,d^k} \leq l^k\}, \end{cases} \tag{6}$$

where $\phi^k_{i',j'}$, $i', j' \in \mathcal{N}$, indicate the length of the shortest-time path from terminal i' to terminal j' over the flat network $\mathcal{D}$ for commodity $k \in \mathcal{K}$. Here, property (i) ensures that there exists a path in $\mathcal{D}$ such that it uses arc (i,j), starts at o^k no earlier than e^k, and reaches d^k no later than l^k. Property (ii) ensures that in $\underline{\mathcal{D}^{k}_{\mathcal{T}}}$, the departure time t of i cannot be excessively earlier than its earliest departure time $(e^k + \phi^k_{o^k,i})$, or later than its latest departure time $(l^k - \phi^k_{j,d^k} - \tau_{i,j})$.

Let $\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}} = \{\underline{\mathcal{D}^{k}_{\mathcal{T}}}\}_{k\in\mathcal{K}}$ denote the collection of the timed-node-based time-expanded commodity networks $\underline{\mathcal{D}^{k}_{\mathcal{T}}}$ for $k \in \mathcal{K}$. Define $\underline{\mathcal{A}^{\mathcal{K}}_{\mathcal{T}}} = \bigcup_{k\in\mathcal{K}} \underline{\mathcal{A}^{k}_{\mathcal{T}}}$. We can replace $\underline{\mathcal{A}_{\mathcal{T}}}$ with $\underline{\mathcal{A}^{\mathcal{K}}_{\mathcal{T}}}$ in SND($\underline{\mathcal{D}_{\mathcal{T}}}$) to reduce the number of decision variables and obtain the following optimization model, which is referred to as SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$).

$$z(\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}) = \min \sum_{((i,t),(j,\bar{t}))\in\underline{\mathcal{A}^{\mathcal{K}}_{\mathcal{T}}}} f_{i,j} y^{t\bar{t}}_{i,j} + \sum_{k\in\mathcal{K}} \sum_{((i,t),(j,\bar{t}))\in\underline{\mathcal{A}^{k}_{\mathcal{T}}}} (c^k_{i,j} q^k) x^{kt\bar{t}}_{i,j} \tag{7}$$

$$\text{s.t.} \sum_{((i,t),(j,\bar{t}))\in\underline{\mathcal{A}^{k}_{\mathcal{T}}}\cup\underline{\mathcal{H}_{\mathcal{T}}}} x^{kt\bar{t}}_{i,j} - \sum_{((j,\bar{t}),(i,t))\in\underline{\mathcal{A}^{k}_{\mathcal{T}}}\cup\underline{\mathcal{H}_{\mathcal{T}}}} x^{k\bar{t}t}_{j,i} = \begin{cases} 1 & (i,t) = (o^k, e^k), \\ -1 & (i,t) = (d^k, l^k), \\ 0 & \text{otherwise}, \end{cases} \quad \forall k \in \mathcal{K}, (i,t) \in \underline{\mathcal{N}_{\mathcal{T}}}, \tag{8}$$

$$\sum_{k\in\mathcal{K}:((i,t),(j,\bar{t}))\in\underline{\mathcal{A}^{k}_{\mathcal{T}}}} q^k x^{kt\bar{t}}_{i,j} \leq u_{i,j} y^{t\bar{t}}_{i,j}, \quad \forall\, ((i,t),(j,\bar{t})) \in \underline{\mathcal{A}^{\mathcal{K}}_{\mathcal{T}}}, \tag{9}$$

$$x_{i,j}^{kt\bar{t}} \in \{0,1\}, \quad \forall k \in \mathcal{K}, ((i,t),(j,\bar{t})) \in \underline{\mathcal{A}_{\mathcal{T}}^k} \cup \underline{\mathcal{H}_{\mathcal{T}}}, \tag{10}$$

$$y_{i,j}^{t\bar{t}} \in \mathbb{N}_{\geq 0}, \quad \forall ((i,t),(j,\bar{t})) \in \underline{\mathcal{A}_{\mathcal{T}}^{\mathcal{K}}}. \tag{11}$$

The following proposition holds.

Proposition 1 *SND($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$) is a relaxation of the CTSNDP, and it is tighter than SND($\underline{\mathcal{D}_{\mathcal{T}}}$).*

The relaxation model SND($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$) can further be strengthened by incorporating the following two valid inequalities, proposed by Marshall et al. (2021) and Boland et al. (2017), respectively, where (12) is based on a lower bound of the number of vehicles needed, and (13) ensures the existence of a feasible delivery path for each commodity $k \in \mathcal{K}$.

$$\left\lceil q^k / u_{i,j} \right\rceil x_{i,j}^{kt\bar{t}} \leq y_{i,j}^{t\bar{t}}, \quad \forall\, ((i,t),(j,\bar{t})) \in \underline{\mathcal{A}_{\mathcal{T}}^k}, k \in \mathcal{K}, \tag{12}$$

$$\sum_{((i,t),(j,\bar{t})) \in \underline{\mathcal{A}_{\mathcal{T}}^k}} \tau_{i,j} x_{i,j}^{kt\bar{t}} \leq l^k - e^k, \quad \forall k \in \mathcal{K}. \tag{13}$$

Moreover, as illustrated in Marshall et al. (2021) and Shu et al. (2024), some decision variables could be safely removed to strengthen the relaxation further as it can be proved that there always exists an optimal SND($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$) solution with these decision variables being zero. Specifically, the variable $x_{i,j}^{kt\bar{t}}$ for $((i,t),(j,\bar{t})) \in \underline{\mathcal{A}_{\mathcal{T}}^k}$ can be removed if there exists another decision variable $x_{i,j}^{kt'\bar{t}}$ with $t' > t$ such that $\{k' \in \mathcal{K} : ((i,t),(j,\bar{t})) \in \underline{\mathcal{A}_{\mathcal{T}}^{k'}}\} \subseteq \{k' \in \mathcal{K} : ((i,t'),(j,\bar{t})) \in \underline{\mathcal{A}_{\mathcal{T}}^{k'}}\}$. The variable $x_{i,i}^{kt\bar{t}}$ for $((i,t),(i,\bar{t})) \in \underline{\mathcal{H}_{\mathcal{T}}}$ can be safely removed if there is no variables $x_{j,i}^{k\bar{t}'t'}$ associated to any $((j,\bar{t}'),(i,t')) \in \underline{\mathcal{A}_{\mathcal{T}}^k}$ with $t' \leq t < \arg\max\{s \in \mathcal{T}_i : s \leq e^k + \phi_{o^k,i}^k\}$ or if $\bar{t} > \arg\max\{s \in \mathcal{T}_i : s + \phi_{i,d^k}^k \leq l^k\}$. If all variables $x_{i,j}^{kt\bar{t}}$ associated with the arc $((i,t),(j,\bar{t})) \in \underline{\mathcal{A}_{\mathcal{T}}^{\mathcal{K}}}$ are successfully removed, then the variable $y_{i,j}^{t\bar{t}}$ can be safely eliminated.

5.2. Strengthening the Relaxation through Significant Time Points

We start by observing that consolidating two commodities on an arc may not be possible, even if both can pass through the arc and reach their destinations before the due times. For each arc $(i,j) \in \mathcal{A}$, define $\mathcal{K}_{i,j} = \{k \in \mathcal{K} : d^k \neq i, o^k \neq j, \text{ and } e^k + \phi_{o^k,i}^k + \tau_{i,j} + \phi_{j,d^k}^k \leq l^k\}$ as the set of commodities k for which there exists a path that uses arc (i,j), starting at o^k no earlier than e^k, and reaching d^k no later than l^k. The following observation highlights a sufficient condition for the occurrence of *impossible consolidations* of two commodities, determined by their available times and due times.

Observation 1 (Impossible Consolidation) *Given any arc (i,j) in $\mathcal{A}$ and two different commodities k_1 and k_2 in $\mathcal{K}_{i,j}$, if $e^{k_1} + \phi_{o^{k_1},i}^{k_1} + \tau_{i,j} + \phi_{j,d^{k_2}}^{k_2} > l^{k_2}$, then in any feasible CTSNDP solution, k_1 and k_2 cannot be consolidated on arc (i,j), and such a consolidation of k_1 and k_2 on arc (i,j) is termed impossible.*

Similar to $\underline{\mathcal{D}_{\mathcal{T}}}$, the travel times of arcs in the timed-node-based time-expanded commodity networks $\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$ are lower than or equal to their actual travel times. Thus, the relaxation solution $(\boldsymbol{x}, \boldsymbol{y})$ obtained by solving the relaxation model SND($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$) could still contain impossible consolidations, i.e., $x_{i,j}^{k_1 t\bar{t}} = x_{i,j}^{k_2 t\bar{t}} = 1$ for some $((i,t),(j,\bar{t})) \in \underline{\mathcal{A}_{\mathcal{T}}}$, $k_1, k_2 \in \mathcal{K}_{i,j}$, but $e^{k_1} + \phi_{o^{k_1},i}^{k_1} + \tau_{i,j} + \phi_{j,d^{k_2}}^{k_2} > l^{k_2}$. Below, we present a new initial

discretization $\mathcal{T}$ ensuring the absence of impossible consolidations in any feasible solution of SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$), thereby improving the quality of the relaxation.

The following Theorem 1 demonstrates that by incorporating a specific time point, a *significant time point*, to the discretization, an impossible consolidation of two commodities can be eliminated from any feasible solutions of the relaxation model SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$) (see §EC.2 for an example).

Theorem 1 (Significant Time Points) *Given any arc (i,j) in $\mathcal{A}$ and two different commodities k_1 and k_2 in $\mathcal{K}_{i,j}$ with $e^{k_2}+\phi^{k_2}_{o^{k_2},i}+\tau_{i,j}+\phi^{k_1}_{j,d^{k_1}} > l^{k_1}$, if there exists a time point $\hat{t}$ incorporated in $\mathcal{T}_i$ of discretization $\mathcal{T}$ with $\hat{t} \in (l^{k_1}-\phi^{k_1}_{j,d^{k_1}}-\tau_{i,j}, e^{k_2}+\phi^{k_2}_{o^{k_2},i}]$, which is defined as a significant time point, then commodities k_1 and k_2 cannot be consolidated on any arc $((i,t),(j,t')) \in \underline{\mathcal{A}^{\mathcal{K}}_{\mathcal{T}}}$ in any feasible solution of SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$).*

Remark 1 *By a similar argument to that in Marshall et al. (2021), incorporating the time point $e^{k_2}+\phi^{k_2}_{o^{k_2},i}$ into the discretization can eliminate the impossible consolidation of the two commodities k_1 and k_2 described in Theorem 1 above. Theorem 1 generalizes this result by showing that any point in the interval $(l^{k_1}-\phi^{k_1}_{j,d^{k_1}}-\tau_{i,j}, e^{k_2}+\phi^{k_2}_{o^{k_2},i}]$ can be used to eliminate such an impossible consolidation.*

For each terminal $i \in \mathcal{N}$, define W_i as the set of all time intervals $(l^{k_1}-\phi^{k_1}_{j,d^{k_1}}-\tau_{i,j}, e^{k_2}+\phi^{k_2}_{o^{k_2},i}]$, each associated with an impossible consolidation of two different commodities $k_1,k_2 \in \mathcal{K}$ on arc $(i,j) \in \mathcal{A}$. Each time interval in W_i is denoted by $(a^i_m, b^i_m]$ with $a^i_m < b^i_m$ for $m \in \{1,2,\ldots,|W_i|\}$. According to Theorem 1, to eliminate all impossible consolidations, a straightforward approach is to incorporate the right endpoints b^i_m of each interval $(a^i_m, b^i_m]$ to $\mathcal{T}_i$ of discretization $\mathcal{T}$, for all $m \in \{1,...,|W_i|\}$ and $i \in \mathcal{N}$. However, this approach may be overly aggressive, potentially leading to an excessive number of significant time points in the resulting discretization. To tackle this issue, we propose minimizing the significant time points required to eliminate impossible consolidations as a minimum hitting-set problem for intervals, as illustrated below.

For each $i \in \mathcal{N}$, consider the time intervals $(a^i_m, b^i_m]$ in W_i with $m \in \{1,...,|W_i|\}$. According to Theorem 1, to find a minimum number of significant time points for the elimination of all impossible consolidations of two commodities, it is equivalent to finding a minimal hitting set $\mathcal{M}_i \subseteq \bigcup_{m=1}^{|W_i|}(a^i_m, b^i_m]$ that is, with a minimal cardinality and intersects every time interval in W_i, i.e., $|\mathcal{M}_i \cap (a^i_m, b^i_m]| \geq 1$ for all $m \in \{1,...,|W_i|\}$. This problem is known as *the minimum hitting-set problem for intervals*. As established in the literature, it suffices to restrict the hitting set to a subset of intervals right endpoints (Damaschke 2017). Consequently, the problem is solvable in linear time via a greedy algorithm (see, e.g., Golumbic 2004). By solving a minimum hitting-set problem for intervals, we obtain a minimal set of significant time points for each terminal $i \in \mathcal{N}$. These significant time points are incorporated into the discretization $\mathcal{T}_i$, thereby refining the initial discretization $\mathcal{T}$ and eliminating all infeasible consolidations of two commodities in the initial relaxation.

The initial timed-node-based time-expanded commodity networks in $\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$ are defined to satisfy Properties 1, 2, and 4, and conditions (6)-(i) and (6)-(ii). Specifically, the networks consist of (i) timed-nodes (o^k, e^k) and (d^k, l^k) for all $k \in \mathcal{K}$; $(i, \min_{k\in\mathcal{K}}\{e^k+\phi_{o^k,i}\})$ for all $i \in \mathcal{N}$, where $\phi_{i,j}$ is the length of the shortest path from terminal i to terminal j over the flat network $\mathcal{D}$; $(i,\hat{t})$ for $i \in \mathcal{N}$ and $\hat{t} \in \mathcal{M}_i$, and (ii) timed arcs in $\underline{\mathcal{A}^k_{\mathcal{T}}}$ and $\underline{\mathcal{H}_{\mathcal{T}}}$. Algorithm 2 in §EC.3.1 of the e-companion illustrates in detail how the initialization is performed.

6. MIP-Based Strategy for Computing Upper Bound

The optimal solution of the relaxation model SND($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$) may not correspond to a feasible solution of the CTSNDP with the same cost. Since travel times are underestimated in SND($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$), a relaxation solution, though corresponding to a collection of feasible delivery paths for all commodities as defined by inequalities (13), could still involve an infeasible consolidation. In such cases, existing DDD algorithms in the literature typically employ heuristic methods to derive feasible CTSNDP solutions (serving as upper-bound solutions) by following the delivery routes given in the optimal relaxation solution and adjusting the consolidations. However, the quality of such upper-bound solutions is not guaranteed, limiting the convergence efficiency of these existing DDD algorithms.

In this section, we propose a new MIP-based strategy to derive a tighter upper-bound solution from a collection of relaxation solutions obtained by solving the relaxation model SND($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$). For each relaxation solution, we compute a feasible CTSNDP solution by solving a *consolidation planning problem* that optimizes the adjustment of consolidations to the delivery paths of the relaxation solution while minimizing the total cost. The best feasible solution obtained is then used to update the best upper-bound solution and the upper bound (UB) in our EDDD algorithm. The next section defines the consolidation planning problem, and Section 6.2 presents its MIP formulation to compute a feasible CTSNDP solution.

6.1. Consolidation Planning Problem (CPP)

The following notation is introduced to define the problem. Consider any relaxation solution $(\boldsymbol{x}, \boldsymbol{y})$, which is a feasible solution to the relaxation model SND($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$). Let $\mathscr{Q}^k(\boldsymbol{x}, \boldsymbol{y})$ be the timed path from (o^k, e^k) to (d^k, l^k) with timed arcs in $\underline{\mathcal{A}_{\mathcal{T}}^k}$ specified by solution $(\boldsymbol{x}, \boldsymbol{y})$ for commodity $k \in \mathcal{K}$, which can be referred to as a *timed path* for commodity k. A collection of such timed paths $\{\mathscr{Q}^k(\boldsymbol{x}, \boldsymbol{y})\}_{k\in\mathcal{K}}$ constitutes a *timed-path solution* to the CTSNDP. Let $p^k(\boldsymbol{x}, \boldsymbol{y})$ be the *flat path* for commodity k (a simple path over $\mathcal{D}$), associated with the timed path $\mathscr{Q}^k(\boldsymbol{x}, \boldsymbol{y})$, with $\mathcal{N}^k(\boldsymbol{x}, \boldsymbol{y}) = \{\nu_1^k = o^k, ..., \nu_{|\mathcal{N}^k(\boldsymbol{x},\boldsymbol{y})|}^k = d^k\} \subseteq \mathcal{N}$ and $\mathcal{A}^k(\boldsymbol{x}, \boldsymbol{y}) = \{a_1^k, ..., a_{|\mathcal{A}^k(\boldsymbol{x},\boldsymbol{y})|}^k\} \subseteq \mathcal{A}$ representing the node sequence and arc sequence along $p^k(\boldsymbol{x}, \boldsymbol{y})$, respectively. Accordingly, a collection of all such flat paths, denoted by $\mathcal{P}(\boldsymbol{x}, \boldsymbol{y}) = \{p^k(\boldsymbol{x}, \boldsymbol{y})\}_{k\in\mathcal{K}}$, represents a *routing plan of commodities* specified by the timed-path solution $\{\mathscr{Q}^k(\boldsymbol{x}, \boldsymbol{y})\}_{k\in\mathcal{K}}$. Moreover, a *consolidation* can be represented by $((i, j), \kappa)$ where (i, j) is an arc in the flat network and $\kappa \subseteq \mathcal{K}$ is a commodity set, indicating that all commodities in κ are consolidated on arc (i, j). For each timed arc $a = ((i, t), (j, \bar{t})) \in \bigcup_{k\in\mathcal{K}} \mathscr{Q}^k(\boldsymbol{x}, \boldsymbol{y})$, the set of commodities that use timed arc a in the timed-path solution $\{\mathscr{Q}^k(\boldsymbol{x}, \boldsymbol{y})\}_{k\in\mathcal{K}}$ can be denoted by $\kappa_a = \{k \in \mathcal{K} : a \in \mathscr{Q}^k(\boldsymbol{x}, \boldsymbol{y})\}$, and these commodities constitute a consolidation $((i, j), \kappa_a)$. Accordingly, a collection of all such consolidations, denoted by $\mathcal{C}(\boldsymbol{x}, \boldsymbol{y}) = \{((i, j), \kappa_a) : a = ((i, t), (j, \bar{t})) \in \cup_{k\in\mathcal{K}} \mathscr{Q}^k(\boldsymbol{x}, \boldsymbol{y})\}$, represents a *consolidation plan* specified by the timed-path solution $\{\mathscr{Q}^k(\boldsymbol{x}, \boldsymbol{y})\}_{k\in\mathcal{K}}$.

The pair $(\mathcal{P}(\boldsymbol{x}, \boldsymbol{y}), \mathcal{C}(\boldsymbol{x}, \boldsymbol{y}))$ defines a *flat solution* of the CTSNDP. A flat solution $(\mathcal{P}, \mathcal{C})$ is *representable* in $\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$ if there exists a feasible solution $(\boldsymbol{x}, \boldsymbol{y})$ of the relaxation model SND($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$) with $(\mathcal{P}(\boldsymbol{x}, \boldsymbol{y}), \mathcal{C}(\boldsymbol{x}, \boldsymbol{y})) =$

$(\mathcal{P},\mathcal{C})$. A flat solution $(\mathcal{P},\mathcal{C})$ is *implementable* if there exists a *dispatch schedule* $\mathcal{T}(\mathcal{P},\mathcal{C})$ that is a collection of departure times $t^k_{\nu^k_n}$ for $k\in\mathcal{K}$ and $n\in\{1,2,...,|p^k|\}$, such that (i) $t^k_{\nu^k_1}\geq e^k$ and $t^k_{\nu^k_{|p^k|}}+\tau_{a^k_{|p^k|}}\leq l^k$ for all $k\in\mathcal{K}$, (ii) $t^k_{\nu^k_n}\geq t^k_{\nu^k_{n-1}}+\tau_{a^k_{n-1}}$ for all $k\in\mathcal{K}$, $n=2,\ldots,|p^k|$, and (iii) $t^k_i=t^{k'}_i$ for all $((i,j),\kappa)\in\mathcal{C}, k\in\kappa,k'\in\kappa$. Otherwise, the flat solution is termed *non-implementable*. The first two conditions ensure feasible dispatch times for each p^k, denoted as p^k-dispatch time, and the last condition ensures that each consolidation $((i,j),\kappa)\in\mathcal{C}$ is achieved so that all commodities consolidated on each arc (i,j) pass through the arc at the same time. An implementable flat solution $(\mathcal{P},\mathcal{C})$ together with the dispatch schedule $\mathcal{T}(\mathcal{P},\mathcal{C})$ forms a feasible solution to the CTSNDP.

Let $\Omega(\boldsymbol{x},\boldsymbol{y})$ be the set of all possible consolidation plans $\mathcal{C}$ such that the flat solution $(\mathcal{P}(\boldsymbol{x},\boldsymbol{y}),\mathcal{C})$ is implementable. Due to (13) included in SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$), each flat path $p^k(\boldsymbol{x},\boldsymbol{y})$ must be feasible for commodity k with $e^k+\sum_{a\in p^k(\boldsymbol{x},\boldsymbol{y})}\tau_a\leq l^k$, ensuring that $\Omega(\boldsymbol{x},\boldsymbol{y})$ is nonempty. For each $\mathcal{C}\in\Omega(\boldsymbol{x},\boldsymbol{y})$, since $(\mathcal{P}(\boldsymbol{x},\boldsymbol{y}),\mathcal{C})$ is implementable, there must exist a dispatch schedule, denoted by $\mathcal{T}(\mathcal{P}(\boldsymbol{x},\boldsymbol{y}),\mathcal{C}))$, such that $(\mathcal{P}(\boldsymbol{x},\boldsymbol{y}),\mathcal{T}(\mathcal{P}(\boldsymbol{x},\boldsymbol{y}),\mathcal{C}))$ forms a feasible solution to the CTSNDP. Accordingly, given any relaxation solution $(\boldsymbol{x},\boldsymbol{y})$, the *consolidation planning problem* (or CPP in short) aims to optimize the consolidation plan $\mathcal{C}\in\Omega(\boldsymbol{x},\boldsymbol{y})$, so that the total cost of such a feasible CTSNDP solution $(\mathcal{P}(\boldsymbol{x},\boldsymbol{y}),\mathcal{T}(\mathcal{P}(\boldsymbol{x},\boldsymbol{y}),\mathcal{C}))$, denoted by $f(\mathcal{P}(\boldsymbol{x},\boldsymbol{y}),\mathcal{C})$, is minimized. Thus, the CPP can be formulated as follows:

$$\min_{\mathcal{C}\in\Omega(\boldsymbol{x},\boldsymbol{y})} f(\mathcal{P}(\boldsymbol{x},\boldsymbol{y}),\mathcal{C}). \tag{14}$$

Remark 2 *The above approach differs from existing DDD algorithms in the literature, such as those proposed by Boland et al. (2017), Marshall et al. (2021), Shu et al. (2024), which derive feasible CTSNDP solutions solely based on the optimal relaxation solution $(\boldsymbol{x}^*,\boldsymbol{y}^*)$. Furthermore, these methods only slightly adjust the consolidations of the flat solution $(\mathcal{P}(\boldsymbol{x}^*,\boldsymbol{y}^*),\mathcal{C}(\boldsymbol{x}^*,\boldsymbol{y}^*))$ when it is non-implementable, generating heuristic solutions to the CPP defined by $(\boldsymbol{x}^*,\boldsymbol{y}^*)$. Consequently, the upper bound derived from the optimal solutions to the CPP for a collection of relaxation solutions, including the best relaxation solution, is tighter than those obtained using existing methods.*

6.2. Solving the MIP Formulation of CPP

To compute an upper bound solution for the CTSNDP, we need to solve the CPP defined in (14) for each given relaxation solution $(\boldsymbol{x},\boldsymbol{y})$, which, as illustrated below, can be formulated as a MIP. Let $\mathcal{A}(\boldsymbol{x},\boldsymbol{y})=\bigcup_{k\in\mathcal{K}}\mathcal{A}^k(\boldsymbol{x},\boldsymbol{y})$ indicate the set of the arcs used in the routing plan $\mathcal{P}(\boldsymbol{x},\boldsymbol{y})$. For each $(i,j)\in\mathcal{A}(\boldsymbol{x},\boldsymbol{y})$, let $\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})=\{k\in\mathcal{K}:(i,j)\in\mathcal{A}^k(\boldsymbol{x},\boldsymbol{y})\}$ be the set of commodities that pass through arc (i,j), indicating that $|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|$ is a valid upper bound on the number of consolidations on arc (i,j) for any feasible CTSNDP solution that utilizes the routing plan $\mathcal{P}(\boldsymbol{x},\boldsymbol{y})$. To formulate the MIP, we assume without loss of generality that there are exactly $|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|$ consolidations on each arc $(i,j)\in\mathcal{A}(\boldsymbol{x},\boldsymbol{y})$, denoted by $((i,j),\kappa_n)$, $n=1,\ldots,|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|$, allowing $\kappa_n=\emptyset$ to indicate an empty consolidation.

To represent a consolidation plan, we introduce a binary variable z_{ijr}^k for each $k \in \mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})$, $(i,j) \in \mathcal{A}^k(\boldsymbol{x},\boldsymbol{y})$, and $r \in \{1,2,\cdots,|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|\}$, indicating whether commodity k is included in the r-th consolidation on arc (i,j). To represent the corresponding dispatch schedule, we introduce a non-negative continuous variable δ_i^k for each $k \in \mathcal{K}$, $i \in \mathcal{N}^k(\boldsymbol{x},\boldsymbol{y})$, indicating the time when commodity k departs from terminal i. We also introduce a non-negative continuous variable b_{ijr} for each $(i,j) \in \mathcal{A}(\boldsymbol{x},\boldsymbol{y})$ and $r \in \{1,2,...,|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|\}$ to indicate the time when the consolidated shipments of the r-th consolidation on arc (i,j) departs from terminal i. Moreover, we introduce a non-negative integer variable γ_{ijr} for each $(i,j) \in \mathcal{A}(\boldsymbol{x},\boldsymbol{y})$ and $r \in \{1,...,|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|\}$ to indicate the number of vehicles that must be used to serve the r-th consolidation from terminal i to terminal j.

The CPP defined in (14) can be formulated as the following MIP, which is denoted as CPPMIP($\boldsymbol{x},\boldsymbol{y}$):

$$\min \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}^k(\boldsymbol{x},\boldsymbol{y})} c_{i,j}^k\, q^k + \sum_{(i,j)\in\mathcal{A}(\boldsymbol{x},\boldsymbol{y})} \sum_{r=1}^{|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|} f_{i,j}\,\gamma_{ijr} \tag{15}$$

$$\text{s.t.} \sum_{r=1}^{|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|} z_{ijr}^k = 1,\ \forall\, k \in \mathcal{K}, (i,j) \in \mathcal{A}^k(\boldsymbol{x},\boldsymbol{y}), \tag{16}$$

$$\sum_{k\in\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})} q^k z_{ijr}^k \le u_{i,j}\gamma_{ijr},\ \forall\,(i,j) \in \mathcal{A}(\boldsymbol{x},\boldsymbol{y}), r \in \{1,2,...,|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|\}, \tag{17}$$

$$\delta_i^k + \tau_{i,j} \le \delta_j^k,\ \forall\, k \in \mathcal{K}, (i,j) \in \mathcal{A}^k(\boldsymbol{x},\boldsymbol{y}), \tag{18}$$

$$\delta_{o^k}^k \ge e^k, \forall\, k \in \mathcal{K}, \tag{19}$$

$$\delta_{d^k}^k \le l^k,\ \forall\, k \in \mathcal{K}, \tag{20}$$

$$\delta_i^k \le b_{ijr} + M_{ijr}^{k+}(1 - z_{ijr}^k),\ \forall\,(i,j) \in \mathcal{A}(\boldsymbol{x},\boldsymbol{y}), k \in \mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y}), r \in \{1,2,...,|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|\}, \tag{21}$$

$$\delta_i^k \ge b_{ijr} - M_{ijr}^{k-}(1 - z_{ijr}^k),\ \forall\,(i,j) \in \mathcal{A}(\boldsymbol{x},\boldsymbol{y}), k \in \mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y}), r \in \{1,2,...,|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|\}, \tag{22}$$

$$\gamma_{ijr} \in \mathbb{N}_{\ge 0},\ \forall\,(i,j) \in \mathcal{A}(\boldsymbol{x},\boldsymbol{y}), r \in \{1,2,...,|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|\}, \tag{23}$$

$$z_{ijr}^k \in \{0,1\},\ \forall\, k \in \mathcal{K}, (i,j) \in \mathcal{A}^k(\boldsymbol{x},\boldsymbol{y}), r \in \{1,2,...,|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|\}, \tag{24}$$

$$\delta_i^k \ge 0,\ \forall\, k \in \mathcal{K}, i \in \mathcal{N}^k(\boldsymbol{x},\boldsymbol{y}), \tag{25}$$

$$b_{ijr} \ge 0,\ \forall\,(i,j) \in \mathcal{A}(\boldsymbol{x},\boldsymbol{y}), r \in \{1,2,...,|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|\}. \tag{26}$$

In model CPPMIP($\boldsymbol{x},\boldsymbol{y}$), the objective function (15) indicates the total cost to be minimized. Constraints (16) ensure that for every arc $(i,j) \in \mathcal{A}^k(\boldsymbol{x},\boldsymbol{y})$, commodity k must be included in exactly one consolidation on arc (i,j). Constraints (17) ensure that sufficient capacity is available to serve the consolidated shipments of each consolidation. Constraints (18)–(20) are imposed on commodities' departure times concerning the travel time of each arc, the earliest available time of each commodity, and the due time of each commodity. Constraints (21) and (22) ensure that for each arc $(i,j) \in \mathcal{A}(\boldsymbol{x},\boldsymbol{y})$, the commodities consolidated to be shipped together through arc (i,j) have the same departure time from terminal i, with M_{ijr}^{k+} and M_{ijr}^{k-} being sufficiently large constants. Finally, constraints (23) -(26) state the domains of the decision variables.

The above MIP model CPPMIP($\boldsymbol{x}, \boldsymbol{y}$) exhibits a high degree of symmetry due to permutations of the consolidation index r on each arc, leading to numerous redundant solutions, which can make the model difficult to solve. To address this challenge, we propose an acceleration strategy adapted from graph coloring problems (Campêlo et al. 2008, Malaguti et al. 2009), detailed in §EC.3.2 in the e-companion. This acceleration strategy associates a representative commodity with each consolidation on the arc, ensuring that the r-th consolidation can be used to transport other commodities only if its representative commodity has already been assigned to that consolidation.

By solving model CPPMIP($\boldsymbol{x}, \boldsymbol{y}$), the routing plan $\mathcal{P}(\boldsymbol{x}, \boldsymbol{y})$, together with the obtained optimal p^k-dispatch times, $\{\boldsymbol{\delta}^k\}_{k\in\mathcal{K}}$, represent a feasible solution to the CTSNDP that adheres to the given routing plan $\mathcal{P}(\boldsymbol{x}, \boldsymbol{y})$ with the total cost calculated in (15) minimized. We note that when solving model CPPMIP($\boldsymbol{x}, \boldsymbol{y}$), acceleration can be achieved by incorporating an initial feasible solution, which is easily obtained by revising the relaxation solution $(\boldsymbol{x}, \boldsymbol{y})$ and by terminating the search if the model's best lower bound is proven to be no smaller than the current upper bound UB or once a prescribed time limit is reached. If the flat solution $(\mathcal{P}(\boldsymbol{x}^*, \boldsymbol{y}^*), \mathcal{C}(\boldsymbol{x}^*, \boldsymbol{y}^*))$ of the optimal relaxation solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is implementable, implying that $\mathcal{C}(\boldsymbol{x}, \boldsymbol{y}) \in \Omega(\boldsymbol{x}, \boldsymbol{y})$, then the optimal objective value of model CPPMIP($\boldsymbol{x}, \boldsymbol{y}$), which is an upper bound for CTSNDP, must be equal to the objective value of the optimal relaxation solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, which is a lower bound for CTSNDP. In this case, the algorithm reaches the termination condition UB=LB.

7. Joint Minimal-Infeasible-Pattern-Elimination Strategy for Refining Discretization

For the DDD algorithm, if its termination condition UB=LB is violated, the optimal relaxation solution must be infeasible to the original CTSNDP, and its flat solution must be non-implementable. In such cases, it is necessary to refine the discretization $\mathcal{T}$ by adding new time points to eliminate the optimal relaxation solution so that it no longer appears under a refined discretization $\mathcal{T}'$. Since the complete time-expanded network contains a finite number of time points, the DDD algorithm will eventually obtain an optimal relaxation solution and a flat solution implementable at UB=LB after a finite number of iterations.

7.1. Motivation and New Refinement Strategy

To refine the model, Boland et al. (2017) select a subset of timed arcs in the current optimal relaxation solution with underestimated lengths, for which correcting the lengths would be sufficient to make the current solution infeasible. They then increase the lengths of these arcs to their true values. Later, Marshall et al. (2021) showed that every relaxation solution is associated with a flat solution, and that different relaxation solutions infeasible to the original CTSNDP may share the same non-implementable flat solution. Accordingly, their refinement shifts the focus from eliminating a single infeasible optimal relaxation solution to eliminating its associated non-implementable flat solution, ensuring that this flat solution is no longer associated with any feasible relaxation solution. To facilitate this refinement, Marshall et al. (2021) introduced the *solution graph*, a compact representation of a relaxation solution.

As proposed by Marshall et al. (2021), a flat solution $\mathcal{S} = (\mathcal{P}, \mathcal{C})$ can be represented by a solution graph, which is defined as a directed graph $\mathscr{D}(\mathcal{S}) = (\mathscr{N}, \mathscr{H})$ with its node set $\mathscr{N}$ and arc set $\mathscr{H}$ derived from the routing plan $\mathcal{P} = \{p^k\}_{k \in \mathcal{K}}$ and consolidation plan $\mathcal{C}$ of $\mathcal{S}$. In particular, the node set $\mathscr{N} = \mathscr{N}_1 \cup \mathscr{N}_2$ comprises a subset of dispatch nodes, $\mathscr{N}_1 = \{\eta_i^k : k \in \mathcal{K}, i \in p^k\}$, and a subset of consolidation nodes, $\mathscr{N}_2 = \{s_{i,j}^\kappa : ((i,j), \kappa) \in \mathcal{C}\}$. For each $k \in \mathcal{K}$, nodes $\eta_{o^k}^k$ and $\eta_{d^k}^k$ are referred to as the origin and the destination dispatch nodes of commodity k, respectively. The arc set $\mathscr{H} = \mathscr{H}_1 \cup \mathscr{H}_2$ comprises a subset of arcs $\mathscr{H}_1 = \{(\eta_i^k, s_{i,j}^\kappa) : \eta_i^k \in \mathscr{N}_1, s_{i,j}^\kappa \in \mathscr{N}_2, k \in \kappa\}$ from dispatch nodes to consolidation nodes, and a subset of arcs $\mathscr{H}_2 = \{(s_{i,j}^\kappa, \eta_j^k) : \eta_j^k \in \mathscr{N}_1, s_{i,j}^\kappa \in \mathscr{N}_2, k \in \kappa\}$ from consolidation nodes to dispatch nodes. For each arc $a \in \mathcal{A}$, the edge length, referred to as its transit time, is 0 for all $(\eta_i^k, s_{i,j}^\kappa) \in \mathscr{H}_1$, and is $\tau_{i,j}$ for all $(s_{i,j}^\kappa, \eta_j^k) \in \mathscr{H}_2$. We note that, each consolidation node $s_{i,j}^\kappa$ with $\kappa = \{k\}$ is redundant (being incident to only two arcs, $(\eta_i^k, s_{i,j}^\kappa)$ and $(s_{i,j}^\kappa, \eta_j^k)$) and could be eliminated by replacing the two arcs with (η_i^k, η_j^k). Here, we retain these nodes $s_{i,j}^\kappa$ in $\mathscr{D}(\mathcal{S})$ for the sake of notation simplicity.

Marshall et al. (2021) showed that every non-implementable flat solution contains certain infeasible patterns in its solution graph, such as those *cycles* or *failed paths* from $\eta_{o^k}^k$ to $\eta_{d^{k'}}^{k'}$ whose transit time exceeds $l^{k'} - e^k$. Their refinement, therefore, proceeds to eliminate the non-implementable flat solution associated with the optimal relaxation solution by eliminating one or more of these patterns in its solution graph. They further showed, computationally, that while eliminating a single infeasible pattern suffices to eliminate a non-implementable flat solution, removing multiple infeasible patterns per refinement step can accelerate convergence. However, since their method generates refinement time points separately for each selected pattern, it may introduce many redundant time points. As a result, although eliminating multiple patterns allows more instances to be solved than eliminating a single pattern, this separate refinement strategy faces challenges in balancing convergence acceleration with control over the relaxation model size.

From the observations above, we identify a key design insight for refinement: *to accelerate convergence, refinement should target multiple infeasible patterns, extending beyond the optimal relaxation solution, with refinement time points selected jointly rather than separately.* Accordingly, we shift the focus from eliminating a single non-implementable flat solution to jointly eliminating as many relevant infeasible patterns as possible while adding as few new time points as possible, and propose a *joint minimal-infeasible-pattern-elimination strategy* for refinement of discretization:

1. To avoid redundant eliminations, we eliminate only minimal infeasible patterns, which are defined as infeasible patterns that do not contain any other infeasible pattern.
2. To target multiple infeasible patterns for elimination, we collect minimal infeasible patterns not only from the non-implementable flat solution associated with the optimal relaxation solution, but also from those associated with other obtained relaxation solutions.
3. To jointly eliminate all the collected minimal infeasible patterns, we introduce a new refinement problem that aims to minimize the total number of new time points. To solve this problem, we formulate it as a MIP model and develop a three-step matheuristic based on this MIP model.

In the following, we first introduce the newly defined minimal infeasible pattern, referred to as a minimal too-long path, in Section 7.2. We then present the refinement problem and its MIP formulation in Section 7.3, and develop the three-step matheuristic for solving it in Section 7.4.

7.2. Minimal Too-Long Paths as Minimal Infeasible Patterns

In Marshall et al. (2021), infeasible patterns are characterized as failed paths and cycles in the solution graph involving both consolidation nodes and dispatch nodes. We observe that these infeasible patterns in the solution graph can be characterized solely by the dependencies among the arrival times of commodities at dispatch nodes. Thus, infeasible patterns, including failed paths and cycles, can be simply represented by their implied sequence of dispatch-node dependencies without including the consolidation nodes. To formalize this concise representation of infeasible patterns, we define a dispatch-node graph $\mathscr{G}(\mathcal{S})$ for a flat solution $\mathcal{S}$ as follows, by removing the consolidation nodes from the solution graph $\mathscr{D}(\mathcal{S})$.

Definition 1 (Dispatch-node graph) *Given a flat solution $\mathcal{S}$, its dispatch-node graph is defined as a directed graph $\mathscr{G}(\mathcal{S}) = (\mathscr{V}, \mathscr{A})$. In particular, the node set comprises only the set of dispatch nodes $\mathscr{V} = \{\eta_i^k : k \in \mathcal{K}, i \in p^k\}$ and the arc set is defined by $\mathscr{A} = \{(\eta_i^k, \eta_j^{k'}) : ((i,j),\kappa) \in \mathcal{C}, k \in \kappa, k' \in \kappa\}$. For each arc $a \in \mathscr{A}$, it has an edge length $\rho(a)$ (referred to as its transit time), which equals $\tau_{i,j}$ for all $a = (i,j) \in \mathscr{A}$.*

We now define infeasible patterns in the dispatch-node graph $\mathscr{G}(\mathcal{S}) = (\mathscr{V}, \mathscr{A})$, namely, too-long paths. Specifically, let $\mathscr{P}(\mathcal{S})$ indicate a set of paths in $\mathscr{G}(\mathcal{S})$ with the initial node being the origin dispatch node of a commodity in $\mathcal{K}$. Consider any path $P \in \mathscr{P}(\mathcal{S})$, where its initial node and final node are denoted by $\eta_{o^{k_1}}^{k_1}$ and $\eta_i^{k_2}$, respectively. It is worth noting that P may not be a simple path. Let $\rho(P)$ indicate the total transit time of path P. Thus, $[e^{k_1} + \rho(P)]$ indicates the earliest arrival time at the final node $\eta_i^{k_2}$ along path P. Since $\phi_{i,d^{k_2}}^{k_2}$ denotes the length of the shortest-time path from node i to the destination node d^{k_2} of commodity k_2, commodity k_2 cannot arrive at d^{k_2} earlier than $[e^{k_1} + \rho(P) + \phi_{i,d^{k_2}}^{k_2}]$. Accordingly, if $[e^{k_1} + \rho(P) + \phi_{i,d^{k_2}}^{k_2}]$ exceeds the due time l^{k_2} of k_2, or equivalently, the total transit time $\rho(P)$ exceeds $(l^{k_2} - e^{k_1} - \phi_{i,d^{k_2}}^{k_2})$, we define such a path P as a *too-long path*, as stated in Definition 2 below.

Definition 2 (Too-long Path) *Given a flat solution $\mathcal{S}$, a (not necessarily simple) path $P \in \mathscr{P}(\mathcal{S})$, starting at node $\eta_{o^{k_1}}^{k_1}$ and ending at node $\eta_i^{k_2}$, is a too-long path if $\rho(P) > l^{k_2} - e^{k_1} - \phi_{i,d^{k_2}}^{k_2}$.*

Theorem 2 below extends the result of Marshall et al. (2021) from the solution graph to the dispatch-node graph. It shows that a flat solution is non-implementable if and only if its dispatch-node graph contains a too-long path. Accordingly, if we refine $\mathcal{T}$ to a new discretization $\mathcal{T}'$ that eliminates a too-long path P in $\mathscr{G}(\mathcal{S})$, in the sense that P does not appear in the dispatch-node graph of any solution representable in $\underline{\mathcal{D}_{\mathcal{T}'}^{\mathcal{K}}}$, then $\mathcal{S}$ is also eliminated, in the sense that it is no longer representable in $\underline{\mathcal{D}_{\hat{\mathcal{T}}}^{\mathcal{K}}}$. Therefore, too-long paths in the dispatch-node graph provide a more concise and unified characterization of infeasible structural patterns using only dispatch nodes. This representation eliminates the need to treat failed paths and cycles as separate infeasible patterns, and it also avoids unnecessary distinctions among paths that traverse the same sequence of dispatch nodes but differ only in their consolidation nodes.

Theorem 2 *A flat solution $\mathcal{S}$ is non-implementable if and only if its dispatch-node graph $\mathscr{G}(\mathcal{S})$ contains a too-long path.*

Next, consider any too-long path $P \in \mathscr{P}(\mathcal{S})$, which may contain another too-long path as its partial path. To eliminate P, it suffices to eliminate such a too-long partial path. Since different too-long paths may share the same too-long partial path, it is sufficient to eliminate only those too-long paths that contain no other too-long partial path. We define such paths as minimal too-long paths in Definition 3 below.

Definition 3 (Minimal Too-Long Path) *A too-long path P is minimal if every partial path of P that starts from the initial node of P, except P itself, is not a too-long path.*

7.3. Refinement Problem and MIP Formulation

Consider a collection of feasible solutions $(\boldsymbol{x}, \boldsymbol{y})$ to the relaxation model SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$), which includes $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ and is denoted by $\mathcal{X}$. Let $\mathscr{S}$ indicate a collection of flat solutions $(\mathcal{P}(\boldsymbol{x}, \boldsymbol{y}), \mathcal{C}(\boldsymbol{x}, \boldsymbol{y}))$ with $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X}$, which includes $(\mathcal{P}(\boldsymbol{x}^*, \boldsymbol{y}^*), \mathcal{C}(\boldsymbol{x}^*, \boldsymbol{y}^*))$. We collect all distinct minimal too-long paths from the non-implementable flat solutions in $\mathscr{S}$ and denote this collection by $\mathscr{P}$.

Consider any discretization $\mathcal{T}'$ and any minimal too-long path $P \in \mathscr{P}$. We say that P is eliminated under $\mathcal{T}'$ if it does not appear in the dispatch-node graph of any flat solution representable in $\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}'}}$. We now state Lemma 1, which extends Observations 1 and 2 in Marshall et al. (2021) and provides a sufficient condition for eliminating P under $\mathcal{T}'$ from the dispatch-node graph.

Lemma 1 (Condition for Elimination) *Consider any discretization $\mathcal{T}'$ and any minimum too-long path $P = (\eta^{k^P_h}_{i^P_h})^{|P|}_{h=1} \in \mathscr{P}$. Then P is eliminated under $\mathcal{T}'$ if there does not exist a timed path $R(P, \mathcal{T}')$ in $\underline{\mathcal{D}_{\mathcal{T}'}} = (\underline{\mathcal{N}_{\mathcal{T}'}}, \underline{\mathcal{A}_{\mathcal{T}'}} \cup \underline{\mathcal{H}_{\mathcal{T}'}})$ from some timed node (i^P_1, t^P_1) to some timed node $(i^P_{|P|}, \bar{t}^P_{|P|})$ such that each of its service arcs $((i^P_h, t^P_h), (i^P_{h+1}, \bar{t}^P_{h+1})) \in \underline{\mathcal{A}_{\mathcal{T}'}}$ belongs to $\underline{\mathcal{A}^{k^P_h}_{\mathcal{T}'}} \cap \underline{\mathcal{A}^{k^P_{h+1}}_{\mathcal{T}'}}$ for all $h = 1, \ldots, |P| - 1$.*

Based on Lemma 1, we define a new refinement problem as follows. Given the current discretization $\mathcal{T}$ and the complete discretization $\hat{\mathcal{T}}$, consider each minimal too-long path $P \in \mathscr{P}$. Let $\mathrm{ELIM}(P, \mathcal{T}, \hat{\mathcal{T}})$ denote the set of all refined discretizations $\mathcal{T}'$ that satisfy the condition in Lemma 1 for eliminating P under $\mathcal{T}'$. The *refinement problem* seeks to eliminate all minimal too-long paths in $\mathscr{P}$ by satisfying the condition in Lemma 1, while minimizing the total number of new time points introduced in refining the current discretization $\mathcal{T}$. Accordingly, the refinement problem can be formulated as follows:

$$\min_{\mathcal{T} \subset \mathcal{T}' \subseteq \hat{\mathcal{T}}} |\mathcal{T}' \setminus \mathcal{T}| \;\text{ subject to } \mathcal{T}' \in \mathrm{ELIM}(P, \mathcal{T}, \hat{\mathcal{T}}) \text{ for all } P \in \mathscr{P}, \tag{27}$$

where $|\mathcal{T}' \setminus \mathcal{T}|$ equals the total number of new time points introduced, and $\mathcal{T}' \in \mathrm{ELIM}(P, \mathcal{T}, \hat{\mathcal{T}})$ for all $P \in \mathscr{P}$ ensures all the minimal too-long paths in $\mathscr{P}$ are eliminated under $\mathcal{T}'$ by satisfying the condition in Lemma 1. We note that the refinement method of Marshall et al. (2021) can be extended to eliminate the minimal too-long paths in $\mathscr{P}$. However, it essentially consists of solving, heuristically and separately for each path $P \in \mathscr{P}$, the following problem:

$$\min_{\mathcal{T} \subseteq \mathcal{T}^*_P \subseteq \hat{\mathcal{T}}} |\mathcal{T}^*_P \setminus \mathcal{T}| \;\text{ subject to } \mathcal{T}^*_P \in \mathrm{ELIM}(P, \mathcal{T}, \hat{\mathcal{T}}),$$

thereby obtaining a discretization $\mathcal{T}_P^*$ for each path P, and then combininh all such discretizations into a single refined discretization. This separate treatment can be inefficient, as a path can often be eliminated by adding time points at different locations along the path, and choices made independently for different paths may fail to exploit shared useful time points, thereby introducing redundancy. In contrast, our refinement problem jointly selects new time points across all paths and their possible elimination plans, seeking a single refinement that eliminates all paths with as few added time points as possible.

Next, we show that the above refinement problem can be formulated as a MIP model. Consider any discretization $\mathcal{T}'$ and any minimum too-long path $P \in \mathscr{P}$. Suppose there exists such a timed path $R(P,\mathcal{T}')$ as specified in Lemma 1. We have that each service arc $((i_h^P, t_h^P), (i_{h+1}^P, \bar{t}_{h+1}^P))$ of $R(P,\mathcal{T}')$ belongs to $\underline{\mathcal{A}_{\mathcal{T}'}^{k_h^P}} \cap \underline{\mathcal{A}_{\mathcal{T}'}^{k_{h+1}^P}}$ for all $h = 1,\ldots,|P|-1$, which, together with (6), implies that

$$\underline{\psi}_h^P(\mathcal{T}') \le t_h^P \le \pi_h^P, \tag{28}$$

where $\pi_h^P = \min_{k\in\{k_h^P,k_{h+1}^P\}}\{l^k - \tau_{i_h^P,i_{h+1}^P} - \phi^k_{i_{h+1}^P,d^k}\}$, and $\underline{\psi}_h^P(\mathcal{T}') = \max\{t \in \mathcal{T}'_{i_h^P} : t \le \psi_h^P\}$ with $\psi_h^P = \max_{k\in\{k_h^P,k_{h+1}^P\}}\{e^k + \phi^k_{o^k,i_h^P}\}$.

For any minimum too-long path $P \in \mathscr{P}$, we can construct another timed path $\underline{R}(P,\mathcal{T}')$ in $\underline{\mathcal{D}_{\mathcal{T}'}}$ with its service arcs $((i_h^P, t_h^P), (i_{h+1}^P, \bar{t}_{h+1}^P)) \in \underline{\mathcal{A}_{\mathcal{T}'}}$ for $h = 1,\ldots,|P|-1$ defined as follows. We set

$$t_1^P = \underline{\psi}_1^P(\mathcal{T}'). \tag{29}$$

By Properties 3 and 4 of $\underline{\mathcal{D}_{\mathcal{T}'}}$, there exists a unique timed service arc $((i_1^P, t_1^P), (i_2^P, \bar{t}_2^P))$ in $\underline{\mathcal{D}_{\mathcal{T}'}}$. Then, for each $h = 2,\ldots,|P|-1$, we set $t_h^P = \max\{\underline{\psi}_h^P(\mathcal{T}'), \bar{t}_h^P\}$, which, by definition of $\underline{\psi}_h^P(\mathcal{T}')$ and $\bar{t}_h^P$, implies that

$$t_h^P = \max\{t \in \mathcal{T}'_{i_h^P} : t \le \max\{\psi_h^P(\mathcal{T}'), t_{h-1}^P + \tau_{i_{h-1}^P,i_h^P}\}\}, \tag{30}$$

and by Properties 3 and 4 of $\underline{\mathcal{D}_{\mathcal{T}'}}$, there exists a unique timed service arc $((i_h^P, t_h^P), (i_{h+1}^P, \bar{t}_{h+1}^P))$ in $\underline{\mathcal{D}_{\mathcal{T}'}}$. By (30), we have $t_h^P \ge \bar{t}_h^P$. Thus, $(i_h^P, \bar{t}_h^P)$ and (i_h^P, t_h^P) are connected by unique holding arcs in $\underline{\mathcal{H}_{\mathcal{T}'}}$.

We refer to the unique timed path $\underline{R}(P,\mathcal{T}')$ constructed above as the *earliest timed path associated with* P. If $\underline{R}(P,\mathcal{T}')$ contains a service arc $((i_h^P, t_h^P), (i_{h+1}^P, \bar{t}_{h+1}^P))$ such that $t_h^P > \pi_h^P$ for some $h = 1,\ldots,|P|-1$, then $\underline{R}(P,\mathcal{T}')$ violates (28), implying that there exists no timed path $R(P,\mathcal{T}')$ satisfying the condition in Lemma 1. This observation yields Lemma 2 below, which states a sufficient and necessary condition for $\mathcal{T}' \in \mathrm{ELIM}(P,\mathcal{T},\hat{\mathcal{T}})$ in a form that will be used to derive the MIP formulation for the refinement problem.

Lemma 2 *Consider any discretization $\mathcal{T}'$ and any minimal too-long path $P \in \mathscr{P}$. Then $\mathcal{T}' \in \mathrm{ELIM}(P,\mathcal{T},\hat{\mathcal{T}})$ if and only if, along the earliest timed path $\underline{R}(P,\mathcal{T}')$ with service arcs $((i_h^P, t_h^P), (i_{h+1}^P, \bar{t}_{h+1}^P)) \in \underline{\mathcal{A}_{\mathcal{T}'}}$ for $h = 1,\ldots,|P|-1$, there exists $h \in \{1,\ldots,|P|-1\}$ such that $t_h^P > \pi_h^P$.*

Based on Lemma 2, we can formulate a MIP model for the refinement problem as follows. Given the current discretization $\mathcal{T}$ and the candidate time-point set $\bar{\mathcal{T}} = \hat{\mathcal{T}} \setminus \mathcal{T}$, let $\Omega(\bar{\mathcal{T}}) = \{(j,t) : j \in \mathcal{N},\, t \in \bar{\mathcal{T}}_j\}$. For each $(j,t) \in \Omega(\bar{\mathcal{T}})$, let $\beta_{j,t}(\mathcal{T}) = \max\{t' \in \mathcal{T}_j : t' < t\}$ be the latest discretization point in $\mathcal{T}$ preceding t, and let $\mathcal{S}_{j,t}(\bar{\mathcal{T}}) = \{t' \in \bar{\mathcal{T}}_j : \beta_{j,t}(\mathcal{T}) < t' < t\}$ be the set of candidate time points in the interval $(\beta_{j,t}(\mathcal{T}), t)$.

For a discretization $\mathcal{T}'$, define $\Phi_h^P(\mathcal{T}') = \{t \in \mathcal{T}'_{i_h^P} : t > \psi_h^P\}$ and $\underline{\Phi}_h^P(\mathcal{T}') = \{t \in \mathcal{T}'_{i_h^P} : \underline{\psi}_h^P(\mathcal{T}) < t \leq \psi_h^P\}$ for each P and h. Let M and T be sufficiently large constants greater than $\max\{t : t \in \hat{\mathcal{T}}_j, j \in \mathcal{N}\}$.

We introduce binary variables $\bar{\gamma}_{j,t}$ to indicate whether candidate time point $t \in \bar{\mathcal{T}}_j$ is added to $\mathcal{T}_j$, binary variables $\bar{\varphi}_h^P$ to indicate whether the threshold π_h^P is exceeded, and binary variables $\bar{\delta}_{h,t}^P$ to indicate that when $\bar{\gamma}_{j,t} = 1$, whether $t \leq \max\{\psi_h^P(\mathcal{T}'), \tilde{t}_{h-1}^P + \tau_{i_{h-1}^P, i_h^P}\}$. For notational convenience, we also define $\bar{\gamma}_{j,t} \equiv 1$ for all $(j,t) \in \Omega(\mathcal{T})$ and $\bar{\delta}_{h,T}^P \equiv 0$ for all $P \in \mathscr{P}$ and $h \in \{2, \ldots, |P| - 1\}$. Let $\mathcal{T}'$ denote the revised discretization obtained by adding new time points specified by $\bar{\gamma}$. We then introduce continuous variables $\tilde{t}_h^P$ to denote the departure time of each earliest timed path $\underline{R}(P, \mathcal{T}')$ from node i_h^P under the revised discretization $\mathcal{T}'$. The following MIP, denoted by $\mathrm{R}(\mathcal{T}, \bar{\mathcal{T}}, \mathscr{P})$, can then be obtained for the refinement problem, aiming to minimize the total number of time points selected from $\bar{\mathcal{T}}$ and added to the current discretization $\mathcal{T}$ while eliminating all the minimal too-long paths in $\mathscr{P}$.

$$\mathrm{R}(\mathcal{T}, \bar{\mathcal{T}}, \mathscr{P}): \min \sum_{(j,t) \in \Omega(\bar{\mathcal{T}})} \bar{\gamma}_{j,t} \tag{31}$$

$$\text{s.t. } \tilde{t}_h^P \geq \underline{\psi}_h^P(\mathcal{T}), \quad \forall P \in \mathscr{P}, h \in \{1, 2, ..., |P| - 1\}, \tag{32}$$

$$\tilde{t}_h^P \geq t \bar{\gamma}_{i_h^P, t}, \quad \forall P \in \mathscr{P}, h \in \{1, 2, ..., |P| - 1\}, t \in \underline{\Phi}_h^P(\bar{\mathcal{T}}), \tag{33}$$

$$\tilde{t}_h^P \leq t + M(1 - \bar{\gamma}_{i_h^P, t}) + M \sum_{t'' \in \underline{\Phi}_h^P(\bar{\mathcal{T}}): t'' > t} \bar{\gamma}_{i_h^P, t''}, \ \forall P \in \mathscr{P}, h = 1, t \in \underline{\Phi}_h^P(\bar{\mathcal{T}}), \tag{34}$$

$$\tilde{t}_h^P \leq \underline{\psi}_h^P(\mathcal{T}) + M \sum_{t'' \in \underline{\Phi}_h^P(\bar{\mathcal{T}})} \bar{\gamma}_{i_h^P, t''}, \ \forall P \in \mathscr{P}, h = 1, \tag{35}$$

$$\tilde{t}_{h-1}^P + \tau_{i_{h-1}^P, i_h^P} \geq t \bar{\gamma}_{i_h^P, t} - M(1 - \bar{\delta}_{h,t}^P), \forall P \in \mathscr{P}, h \in \{2, ..., |P| - 1\}, t \in \Phi_h^P(\mathcal{T} \cup \bar{\mathcal{T}}), \tag{36}$$

$$\tilde{t}_{h-1}^P + \tau_{i_{h-1}^P, i_h^P} \leq (t-1) \bar{\gamma}_{i_h^P, t} + M \bar{\delta}_{h,t}^P, \forall P \in \mathscr{P}, h \in \{2, ..., |P| - 1\}, t \in \Phi_h^P(\mathcal{T} \cup \bar{\mathcal{T}}), \tag{37}$$

$$\tilde{t}_h^P \geq t \cdot \bar{\gamma}_{i_h^P, t} - M(1 - \bar{\delta}_{h,t}^P), \quad \forall P \in \mathscr{P}, h \in \{2, ..., |P| - 1\}, t \in \Phi_h^P(\mathcal{T} \cup \bar{\mathcal{T}}), \tag{38}$$

$$\begin{aligned} \tilde{t}_h^P \leq t' + M(1 - \bar{\gamma}_{i_h^P, t'}) + M \sum_{t'' \in \mathcal{S}_{i_h^P, t}(\bar{\mathcal{T}}): t'' > t'} \bar{\gamma}_{i_h^P, t''} + M \bar{\delta}_{h,t}^P, \\ \forall P \in \mathscr{P}, h \in \{2, ..., |P| - 1\}, t \in \Phi_h^P(\mathcal{T} \cup \bar{\mathcal{T}}) \cup \{T\}, t' \in \mathcal{S}_{i_h^P, t}(\bar{\mathcal{T}}), \end{aligned} \tag{39}$$

$$\begin{aligned} \tilde{t}_h^P \leq \beta_{i_h^P, t}(\mathcal{T}) + M \textstyle\sum_{t'' \in \mathcal{S}_{i_h^P, t}(\bar{\mathcal{T}})} \bar{\gamma}_{i_h^P, t''} + M \bar{\delta}_{h,t}^P, \\ \forall P \in \mathscr{P}, h \in \{2, ..., |P| - 1\}, t \in \Phi_h^P(\mathcal{T} \cup \bar{\mathcal{T}}) \cup \{T\}, \end{aligned} \tag{40}$$

$$\tilde{t}_h^P \geq \pi_h^P + 1 - M(1 - \bar{\varphi}_h^P), \quad \forall P \in \mathscr{P}, h \in \{1, 2, ..., |P| - 1\}, \tag{41}$$

$$\sum_{h=1}^{|P|-1} \bar{\varphi}_h^P \geq 1, \quad \forall P \in \mathscr{P}, \tag{42}$$

$$\bar{\gamma}_{j,t} \in \{0, 1\}, \quad \forall (j,t) \in \Omega(\bar{\mathcal{T}}), \tag{43}$$

$$\bar{\delta}_{h,t}^P \in \{0, 1\}, \quad \forall P \in \mathscr{P}, h \in \{2, ..., |P| - 1\}, t \in \Phi_h^P(\mathcal{T} \cup \bar{\mathcal{T}}), \tag{44}$$

$$\bar{\varphi}_h^P \in \{0, 1\}, \quad \forall P \in \mathscr{P}, h \in \{1, 2, ..., |P| - 1\}, \tag{45}$$

$$\tilde{t}_h^P \geq 0, \quad \forall P \in \mathscr{P}, h \in \{1, 2, ..., |P| - 1\}. \tag{46}$$

The objective (31) minimizes the number of additional time points in the discretization. For each minimal too-long path $P \in \mathscr{P}$, constraints (32)–(40) are imposed to determine the earliest timed path $\underline{R}(P, \mathcal{T}')$ under the revised discretization $\mathcal{T}'$ specified by $\bar{\gamma}$, where for each service arc $((i_h^P, \tilde{t}_h^P), (i_{h+1}^P, \bar{t}_{h+1}^P)) \in \underline{\mathcal{A}_{\mathcal{T}'}}$, the value of $\tilde{t}_h^P$ must be determined according to the recursive relations in (29) and (30). Specifically, constraints (32)–(33) impose $\tilde{t}_h^P \geq \underline{\psi}_h^P(\mathcal{T}')$ for each h, whereas constraints (34)–(35) impose $\tilde{t}_1^P \leq \underline{\psi}_1^P(\mathcal{T}')$, ensuring that $\tilde{t}_1^P = \underline{\psi}_1^P(\mathcal{T}')$ as in (29). When t is selected to be added to $\mathcal{T}'$, that is, when $\bar{\gamma}_{j,t} = 1$, constraints (36)–(37) ensure that $\bar{\delta}_{h,t}^P = 0$ whenever $t > \max\{\psi_h^P(\mathcal{T}'), \tilde{t}_{h-1}^P + \tau_{i_{h-1}^P, i_h^P}\}$, and $\bar{\delta}_{h,t}^P = 1$ otherwise, for $h \in \{2, ..., |P| - 1\}$. Together with constraints (32)–(33), constraint (38) then enforces $\tilde{t}_h^P \geq \max\{t \in \mathcal{T}'_{i_h^P} : t \leq \max\{\psi_h^P(\mathcal{T}'), \tilde{t}_{h-1}^P + \tau_{i_{h-1}^P, i_h^P}\}\}$. Constraints (39)–(40) ensure that $\tilde{t}_h^P \leq \max\{t \in \mathcal{T}'_{i_h^P} : t < t'\}$ for all $t' \in \mathcal{T}'_{i_h^P}$ such that $t' > \max\{\psi_h^P(\mathcal{T}'), \tilde{t}_{h-1}^P + \tau_{i_{h-1}^P, i_h^P}\}$. As a result, it is ensured that $\tilde{t}_h^P \leq \max\{t \in \mathcal{T}'_{i_h^P} : t \leq \max\{\psi_h^P(\mathcal{T}'), \tilde{t}_{h-1}^P + \tau_{i_{h-1}^P, i_h^P}\}\}$. Therefore, $\tilde{t}_h^P = \max\{t \in \mathcal{T}'_{i_h^P} : t \leq \max\{\psi_h^P(\mathcal{T}'), \tilde{t}_{h-1}^P + \tau_{i_{h-1}^P, i_h^P}\}\}$ is ensured for all $h = 2, \ldots, |P| - 1$ as in (30). Moreover, constraints (41) determine whether the departure time $\tilde{t}_h^P$ is greater than the threshold π_h^P, while constraints (42) enforce that, for each minimal too-long path, this occurs for at least one h, thereby ensuring $\mathcal{T}' \in \mathrm{ELIM}(P, \mathcal{T}, \hat{\mathcal{T}})$ by Lemma 2. The remaining constraints (43)–(46) specify the variable domains.

Accordingly, the refined discretization $\mathcal{T}'$ can be obtained by solving the MIP model $\mathrm{R}(\mathcal{T}, \bar{\mathcal{T}}, \mathscr{P})$ and updating each node-specific set as $\mathcal{T}'_j = \mathcal{T}_j \cup \{t \in \bar{\mathcal{T}}_j : \bar{\gamma}_{j,t} = 1\}$ for all $j \in \mathcal{N}$.

7.4. Three-Step Matheuristic for Refinement

The MIP model $\mathrm{R}(\mathcal{T}, \bar{\mathcal{T}}, \mathscr{P})$ derived in Section 7.3 becomes computationally challenging to solve when $|\bar{\mathcal{T}}|$ is large and $\mathscr{P}$ contains too many paths. To address this issue, we develop a three-step matheuristic. It first generates all minimal too-long paths, denoted by $\mathscr{P}$, then extends the approach of Marshall et al. (2021) to construct a reduced set of candidate time points, denoted by $\bar{\mathcal{T}}^*$, and finally solves the MIP model $\mathrm{R}(\mathcal{T}, \bar{\mathcal{T}}^*, \mathscr{P})$ using a sequential batch heuristic based on a MIP solver.

Step 1: Generate a Set of Minimal Too-Long Paths Given the collection $\mathscr{S}$ of flat solutions $(\mathcal{P}(\boldsymbol{x}, \boldsymbol{y}), \mathcal{C}(\boldsymbol{x}, \boldsymbol{y}))$ induced by $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X}$, the first step of our matheuristic constructs the collection $\mathscr{P}$ of minimal too-long paths derived from the flat solutions in $\mathscr{S}$.

In particular, for each flat solution $\mathcal{S} \in \mathscr{S}$, we apply a label extension algorithm to generate minimal too-long paths over its dispatch-node graph $\mathscr{G}(\mathcal{S})$. Here, a label represents a path in $\mathscr{G}(\mathcal{S})$ from an origin dispatch node of some commodity to another dispatch node. The algorithm begins with a set of initial labels, one for each origin dispatch node $\eta_{o^k}^k$ with $k \in \mathcal{K}$. For each label currently in the set, the algorithm extends it to each successor dispatch node in $\mathscr{G}(\mathcal{S})$, thereby creating new labels, after which the original label is removed from the set. If a new label represents a too-long path P that starts at $\eta_{o^{k_1}}^{k_1}$ and reaches $\eta_i^{k_2}$ with $\rho(P) > l_{k_2} - e^{k_1} - \phi_{i,d^{k_2}}^{k_2}$, then P, which must be a minimal too-long path, is added to $\mathscr{P}$, and the new label is discarded. Otherwise, the new label is added to the label set. This label-extension process continues until the label set is empty, obtaining all minimal too-long paths to constitute the collection $\mathscr{P}$.

Step 2: Construct a Reduced Set of Candidate Time Points The second step of our refinement procedure identifies a reduced candidate set $\bar{\mathcal{T}}^*$. The set $\bar{\mathcal{T}}^*$ needs to be sufficiently small to make the resulting MIP model $\mathrm{R}(\mathcal{T}, \bar{\mathcal{T}}^*, \mathscr{P})$ more computationally tractable, while still providing enough flexibility to eliminate all minimal too-long paths in $\mathscr{P}$. We achieve this by extending the approach of Marshall et al. (2021) to too-long paths in dispatch-node graphs.

Consider any too-long path $P \in \mathscr{P}$. For each $g \in \{1, 2, \ldots, |P|\}$, let $\rho(P, g)$ denote the total transit time along the partial path from the initial dispatch node of P to the g-th dispatch node of P. Thus, $e^{k_1^P} + \rho(P, g)$ is the earliest arrival time at the g-th dispatch node of P along path P. As established in Theorem 3 below, which extends the result of Marshall et al. (2021) to too-long paths in dispatch-node graphs, path P can be eliminated by a refinement of $\mathcal{T}$ that adds the time points $e^{k_1^P} + \rho(P, g)$ to $\mathcal{T}_{i_g^P}$ for all $g \in \{1, 2, \ldots, |P| - 1\}$.

Theorem 3 (Too-Long Path Elimination) *Consider any too-long path $P = (\eta_{i_g^P}^{k_g^P})_{g=1}^{|P|} \in \mathscr{P}$. Refine $\mathcal{T}$ to $\mathcal{T}'$ with each $\mathcal{T}'_j = \mathcal{T}_j \cup \{e^{k_1^P} + \rho(P, g) : g \in \{1, 2, \ldots, |P| - 1\}, i_g = j\}$ for $j \in \mathcal{N}$. Then $\mathcal{T}' \in \mathrm{ELIM}(P, \mathcal{T}, \hat{\mathcal{T}})$.*

Accordingly, for each minimal too-long paths $P \in \mathscr{P}$, we apply Theorem 3 to obtain a patch of time points, denoted by set $\tilde{\mathcal{T}}_j(P) = \{e^{k_1^P} + \rho(P, g) : g \in \{1, 2, \ldots, |P| - 1\}, i_g = j\}$ for each $j \in \mathcal{N}$. Thus, we obtain the reduced candidate set $\bar{\mathcal{T}}^*$ with $\bar{\mathcal{T}}_j^* = \left(\bigcup_{P \in \mathscr{P}} \tilde{\mathcal{T}}_j(P) \right) \setminus \mathcal{T}_j$ for all $j \in \mathcal{N}$, implying that $|\bar{\mathcal{T}}^*| \leq |\bar{\mathcal{T}}|$. By Theorem 3, the discretization $\mathcal{T}'$ with $\mathcal{T}'_j = \mathcal{T}_j \cup \bar{\mathcal{T}}_j^*$ for $j \in \mathcal{N}$ is sufficient to eliminate all minimal too-long paths $P \in \mathscr{P}$, thereby ensuring the feasibility of the MIP model $\mathrm{R}(\mathcal{T}, \bar{\mathcal{T}}^*, \mathscr{P})$.

Step 3: Solve $\mathrm{R}(\mathcal{T}, \bar{\mathcal{T}}^*, \mathscr{P}^*)$ to Generate Final Refinement In Step 3, we generate the final refinement by solving the model $\mathrm{R}(\mathcal{T}, \bar{\mathcal{T}}^*, \mathscr{P})$. Although this model can be solved directly with a general-purpose MIP solver, it may still be computationally challenging, even with the reduction in size achieved by $\bar{\mathcal{T}}^*$. To address this challenge, we propose a sequential-batch heuristic for solving $\mathrm{R}(\mathcal{T}, \bar{\mathcal{T}}^*, \mathscr{P})$. Specifically, let $\mathscr{P}_{\mathrm{rem}}^{(0)} = \mathscr{P}$ denote the initial set of remaining minimal too-long paths, that is, those not yet eliminated, and let $\mathcal{T}^{(0)} = \mathcal{T}$ denote the initial discretization. At each iteration r of the sequential batch heuristic, it first selects a subset $\mathscr{P}^{(r)} \subseteq \mathscr{P}_{\mathrm{rem}}^{(r)}$ with at most K_0 paths, where K_0 is a batch-size parameter. It then constructs a restricted candidate set $\bar{\mathcal{T}}^{(r)}$ by collecting the candidate time points in $\tilde{\mathcal{T}}_j(P)$, generated in Step 2, for all $P \in \mathscr{P}^{(r)}$ and all relevant nodes j. It then solves the restricted refinement problem $\mathrm{R}(\mathcal{T}^{(r)}, \bar{\mathcal{T}}^{(r)}, \mathscr{P}^{(r)})$ and updates the discretization to $\mathcal{T}^{(r+1)}$ by adding the time points selected in the obtained solution. The paths in $\mathscr{P}^{(r)}$ are then removed from the remaining set by setting $\mathscr{P}_{\mathrm{rem}}^{(r+1)} = \mathscr{P}_{\mathrm{rem}}^{(r)} \setminus \mathscr{P}^{(r)}$, which is used in the next iteration $r + 1$ to select the subset $\mathscr{P}^{(r+1)}$. The heuristic repeats the iteration until $\mathscr{P}_{\mathrm{rem}}^{(r)}$ becomes empty, and it then returns the final refined discretization $\mathcal{T}^{(r)}$.

8. Computational Study

This section presents an extensive computational analysis to demonstrate the effectiveness and efficiency of the newly proposed EDDD algorithm and to better understand the factors that contribute to its performance. Section 8.2 provides a comparison of the newly proposed EDDD algorithm with the state-of-the-art

DDD algorithms proposed by Boland et al. (2017) and Marshall et al. (2021). Section 8.3 delves into the effectiveness of the new initial relaxation and refinement strategy. Section EC.4 in the electronic companion provides additional results on the effectiveness of the new MIP-based strategy for upper bounds and on the use of a pool of relaxation solutions in the upper-bound derivation and refinement strategies.

8.1. Instances

We evaluated the performance of various DDD algorithms on 558 CTSNDP instances originally generated in Boland et al. (2017). These instances, also used in Boland et al. (2019) and Marshall et al. (2021), were derived from 31 classes of the "C" instances presented in Crainic et al. (2001) for the Capacitated Fixed Charge Network Design Problem. Travel times for each arc and time windows for each commodity were generated following nominal distributions. According to Boland et al. (2019) and Marshall et al. (2021), instances were further categorized based on time flexibility and cost ratio.

Instances were classified as *Low Flexibility (LF)* if $\min_{k\in\mathcal{K}} l^k - (e^k + \phi_{o^k,d^k}) < 227$; otherwise, they were considered *High Flexibility (HF)*. For *Low Cost Ratio (LC)*, an instance was identified if $\frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}} \frac{f_a}{c_a u_a} < 0.175$; otherwise, it fell under *High Cost Ratio (HC)*. Consequently, instances were grouped into four distinct categories: "**HC/LF**", "**HC/HF**", "**LC/LF**" and "**LC/HF**." The instances are characterized by a maximum number of 30 terminals, 683 arcs, and 400 commodities, i.e., $|\mathcal{N}| \leq 30$, $|\mathcal{A}| \leq 683$, and $|\mathcal{K}| \leq 400$.

8.2. Performance of the Enhanced DDD Algorithm

Our enhanced DDD algorithm for the CTSNDP is denoted as EDDD. In comparison, the approaches presented by Boland et al. (2017) and Marshall et al. (2021) are referred to as BHMS and MBSH, respectively. Furthermore, to assess the performance of EDDD, we implemented the algorithm proposed in Boland et al. (2017) for the CTSNDP, removing variables that violate (6) and incorporating the variable-reduction methods in §5. This variant is referred to as IDDD. Both EDDD and IDDD were implemented in Java, and we utilized the Gurobi solver (v.10.0.2) for LP/MIP computations (Gurobi Optimization, LLC 2024). In the following experiments, we solved each CTSNDP instance to an optimality tolerance of 1% with a time limit of one hour for both EDDD and IDDD, as done for BHMS and MBSH in Marshall et al. (2021). We note that our implementation of EDDD harvested all solutions available in the solver's solution pool at each iteration to derive upper bounds and refine the discretization, although the number of harvested solutions can be adjusted. The batch size in the refinement matheuristic was set to $K_0 = 100$. All experiments were conducted on an Intel(R) Core(TM) i7-8700 desktop PC with a 3.20 GHz clock speed and 64 GB RAM.

Table 1 summarizes the obtained results. For each group of instances, the table displays the number of instances in the group and, for each algorithm, the average gap between the final upper bound (UB) and the final lower bound (LB) ("**%Gap**"), i.e., $100.0 \times \frac{(UB-LB)}{UB}$, the average computing time in seconds ("**Time**(s)"), the average number of iterations of the DDD algorithm ("**#Iterations**"), and the percentage of instances solved to optimality ("**%Optimal**") within the given optimality tolerance and time limit.

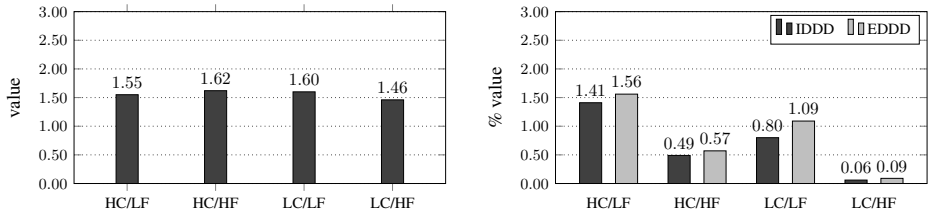
Table 1 Summary Results on the CTSNDP Instances

Group	Algorithm	%Gap	Times(s)	#Iterations	%Optimal
HC/LF	BHMS	0.08	1391.1	5.3	77.1
183	MBSH	0.12	677.8	14.8	85.8
	IDDD	0.99	285.5	13.6	95.6
	EDDD	0.69	8.6	1.5	100.0
HC/HF	BHMS	0.56	1966.7	6.0	53.7
177	MBSH	0.84	1693.8	17.5	56.5
	IDDD	2.34	1377.7	14.8	70.1
	EDDD	0.83	127.2	2.7	100.0
LC/LF	BHMS	0.00	28.6	3.7	100.0
94	MBSH	0.00	0.6	6.5	100.0
	IDDD	0.71	0.4	3.8	100.0
	EDDD	0.34	0.3	1.0	100.0
LC/HF	BHMS	0.00	1.5	2.5	100.0
104	MBSH	0.00	0.1	3.2	100.0
	IDDD	0.51	0.1	1.3	100.0
	EDDD	0.08	0.1	1.0	100.0

The results in Table 1 demonstrate that EDDD successfully solves all 558 instances within one hour. The other three methods fail to solve all instances in groups "**HC/LF**" and "**HC/HF**". Moreover, EDDD achieves this with significantly fewer iterations and shorter computational times. Notably, all 558 CTSNDP instances are solved optimally within five iterations, and for groups "**HC/LF**", "**LC/LF**", and "**LC/HF**", the majority of instances (nearly 80%) are solved optimally within a single iteration. These highlight the superior performance of EDDD compared to other existing DDD algorithms for the CTSNDP, emphasizing the effectiveness of the new initial discretization approach and the new MIP for the upper bound. It is worth noting that the hardware and software configurations used for BHMS and MBSH differed from those used for EDDD, particularly in the computing platform (a cluster of 32-core nodes running at 2.3–2.8 GHz) and the MIP solver (CPLEX). However, even when hardware and software configurations differ, comparing the total number of instances solved to proven optimality and the total number of iterations required can still provide an overall indication of relative performance.

Detailed results on the computational effort of the key EDDD components and on the effectiveness of the sequential batch heuristic in Step 3 of the refinement matheuristic are presented in EC.4.2 and EC.4.3, respectively. The results show that most of the computational effort is devoted to solving the relaxation models, whereas solving CPPMIP$(\boldsymbol{x}, \boldsymbol{y})$ to obtain upper bounds and running the refinement matheuristic are typically in seconds. The results also show that the sequential batch heuristic in Step 3 of the refinement matheuristic obtains a near-optimal number of time points efficiently for refinement, with respect to the MIP model R$(\mathcal{T}, \bar{\mathcal{T}}^*, \mathscr{P})$. This significantly reduces the redundant time points in $\bar{\mathcal{T}}^*$ generated in Step 2.

Table 1 also shows that our implementation of IDDD (which is proposed by Boland et al. 2017) performs better than both BHMS and MBSH and also demonstrates similar performance on the different groups of instances. Thus, in the following, we take the algorithm IDDD as a baseline algorithm to further analyze the detailed performance of the algorithm EDDD.

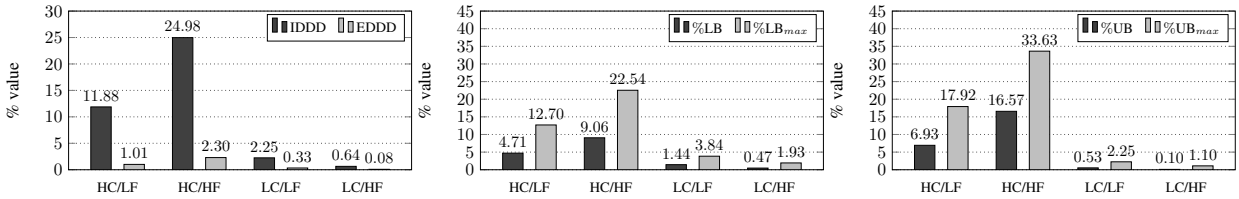


(a) Size of the Initial Discretization (b) Size of the Final Discretization

Figure 1 Relative Network Size

Figure 1 provides insights into the relative sizes of the initial and final discretization generated by EDDD. The algorithm creates a new initial discretization by identifying and adding significant time points based on impossible consolidations. Figure 1-(a) illustrates the relative cardinality of the time points in the new initial discretization compared to the initial discretization used in IDDD, which does not include these identified significant time points. Even after incorporating significant time points, the new initial discretization remains relatively small, with only a nearly 60% increase in time points. Figure 1-(b) shows the relative number of time points in the final discretization compared to the complete discretization $\hat{\mathcal{T}}$, where $\hat{\mathcal{T}}_i = \{0, \Delta, 2\Delta, ..., \Delta\lceil \max_{k\in\mathcal{K}} l^k/\Delta \rceil\}$ for all $i \in \mathcal{N}$ and $\Delta = 1$. Despite EDDD generating more time points in the initial discretization and employing a more aggressive refinement strategy, the final discretization, which can generate the optimal solution for the CTSNDP, still represents only a small fraction of the time points in the complete discretization. The corresponding node-set cardinality of the final time-expanded network is comparable to that of IDDD, but some of these networks fail to generate the optimal solution. In addition, the results from EDDD demonstrate that all 558 CTSNDP instances can be solved to proven optimality over time-expanded commodity networks, where $|\mathcal{N}_{\mathcal{T}}|/|\mathcal{N}_{\hat{\mathcal{T}}}^{\Delta=1}| < 4.1\%$.

Figure 2 highlights the rapid convergence of the EDDD algorithm by comparing the gap between the upper bound and the lower bound in the first iteration achieved by EDDD and IDDD (Figure 2-(a)), as well as by showing the improvement in the upper bound and the lower bound achieved by EDDD compared to IDDD (Figures 2-(b) and 2-(c)). Let LB_1 (LB_2) and UB_1 (UB_2) represent the lower bound and upper bound obtained by EDDD (IDDD) in the first iteration, respectively. The gaps between the upper bound and the lower bound in the first iteration are calculated as $100.0 \times (\text{UB}_1 - \text{LB}_1)/\text{UB}_1$ and $100.0 \times (\text{UB}_2 - \text{LB}_2)/\text{UB}_2$. The improvements in the lower and upper bounds are calculated as $100.0 \times (\text{LB}_1 - \text{LB}_2)/\text{LB}_2$ and $100.0 \times (\text{UB}_2 - \text{UB}_1)/\text{UB}_2$. Let %LB (%UB) and $\%\text{LB}_{max}$ ($\%\text{UB}_{max}$) denote the average and maximum improvement values in lower bound (upper bound) over the instance group. Figure 2-(a) demonstrates that the gap between the upper bound and the lower bound reaches close to 2% for all groups after the first iteration in EDDD, while IDDD only achieves a gap of 11.88% and 24.98% for the "**HC/LF**" and "**HC/HF**" groups, respectively. Figures 2-(b) and 2-(c) illustrate that, based on the new initial relaxation and the new MIP for the upper-bound solution, EDDD generates significantly better lower-bound and upper-bound values. Particularly for the most challenging instance group, "**HC/HF**", the maximum improvements in the lower and upper bounds reach 22.54% and 33.63%, respectively. These results thus demonstrate the significantly fewer iterations and shorter computation times achieved by algorithm EDDD.



(a) %Gap Over the First Iteration (b) Improvement on the LB (c) Improvement on the UB

Figure 2 Detailed Comparison of the First Iteration

8.3. Effectiveness of the Algorithm Components

In this section, we conduct a more in-depth evaluation of the effectiveness of each proposed component of EDDD. Through preliminary experiments, we noticed that the impact of the new algorithm components is negligible in the **LC** class of instances, as all instances in the **LC** class are relatively easy to solve. As a result, our assessment primarily focuses on the effectiveness of the components of the newly proposed algorithm EDDD, particularly within the **HC** class of instances. The effectiveness of the new initial relaxation, refinement strategy, and upper-bound derivation method is discussed in Sections 8.3.1, 8.3.2, and 8.3.3, respectively. Additionally, the necessity of considering bundle relaxation solutions in the upper-bound derivation and refinement is demonstrated in §EC.4.4 in the e-companion to this paper. Some hard instances benefit from this bundling strategy, without which the EDDD algorithm fails to solve all CTSNDP instances.

8.3.1. Effectiveness of the New Initial Relaxation. Figures 1 and 2 demonstrate how the new initial relaxation affects network size and lower-bound values. We now assess the effects of adding significant time points to the discretization to further evaluate the impact on convergence. We, therefore, conducted additional experiments with various variants of IDDD: 1) with inequalities (12), "IDDD+(12)"; 2) with the addition of the significant time points, "IDDD+STP"; 3) with the addition of the significant time points and inequalities (12), "IDDD+(12)+STP". Table 2 presents the results of these variants on the **HC** class of instances, utilizing the same notations introduced in Table 1 and includes the percentage of relaxation solutions whose dispatch-node graphs are cyclic with non-elementary too-long paths ("**%Cyclesol**").

The results presented in Table 2 demonstrate the effectiveness of the inequalities (12) proposed by Marshall et al. (2021). The IDDD variant with the additional inequalities (12) solves more instances and achieves this with reduced computational time. The results also show that the variant of IDDD with the newly proposed initial relaxation exhibits comparable computational performance to the variant with inequalities (12) in terms of optimality rate. Additionally, it demonstrates fewer iterations and a smaller optimality gap for unsolved instances. The results also indicate that with the newly proposed initial relaxation, the IDDD variants can successfully solve nearly 15% additional **HC/HF** instances. Moreover, it achieves this with fewer iterations, shorter computational times, and smaller optimality gaps for unsolved instances when compared to both IDDD and the variant with inequalities (12). These findings support the effectiveness of the newly proposed initial relaxation, regardless of whether inequalities (12) are utilized in the relaxation model.

Furthermore, in all the considered CTSNDP instances, we observed that the relaxation solutions obtained by the algorithms with the newly proposed initial relaxation are without non-elementary too-long paths in

Table 2 Detailed Results for IDDD Variants With/Without the New Initial Relaxation

Group	Algorithm	%Gap	Times(s)	#Iterations	%Optimal	%Cyclesol
HC/LF	IDDD	0.99	285.5	13.6	95.6	1.4
183	IDDD+ (12)	0.87	124.0	10.2	97.8	1.6
	IDDD+ STP	0.88	235.0	6.3	96.2	0.0
	IDDD+ (12) + STP	0.75	37.7	3.9	100.0	0.0
HC/HF	IDDD	2.34	1377.7	14.8	70.1	3.2
177	IDDD+ (12)	1.71	972.9	16.2	82.5	2.8
	IDDD+ STP	1.08	1048.8	8.8	79.1	0.0
	IDDD+ (12) + STP	0.92	472.9	9.2	94.6	0.0

their dispatch-node graphs, while some relaxation solutions obtained by the algorithms without the newly proposed initial relaxation are not. Moreover, in algorithm EDDD, all feasible solutions obtained by the Gurobi solver for the relaxation model are with acyclic dispatch-node graphs containing no non-elementary too-long paths for all 558 CTSNDP instances. These observations indicate that the newly proposed initial relaxation can effectively prevent cycles in these instances. However, it is important to note that the newly proposed initial relaxation may not prevent cycles in all situations. For instance, if all commodities' due times are unrestricted, there are no impossible consolidations and thus no corresponding significant time points. Consequently, cyclic dispatch-node graphs may still arise in the relaxation solutions.

8.3.2. Effectiveness of the New Refinement Strategy. To verify the effectiveness of the new refinement strategy, we executed algorithm IDDD with the new refinement strategy, denoted as variant IDDD_1. We compared its results with those of the original IDDD approach, which employs the refinement strategy proposed by Boland et al. (2017). The comparison results on the **HC** class of instances are presented in Table 3, using the same notations introduced in Table 1, and also displaying the relative number of time points in the final time-expanded network compared to the fully time-expanded network $\mathcal{D}^{\hat{\Lambda}}_{\hat{T}}$ ("**%nSize**").

Table 3 Detailed Results for IDDD Variants with Different Refinement Strategies

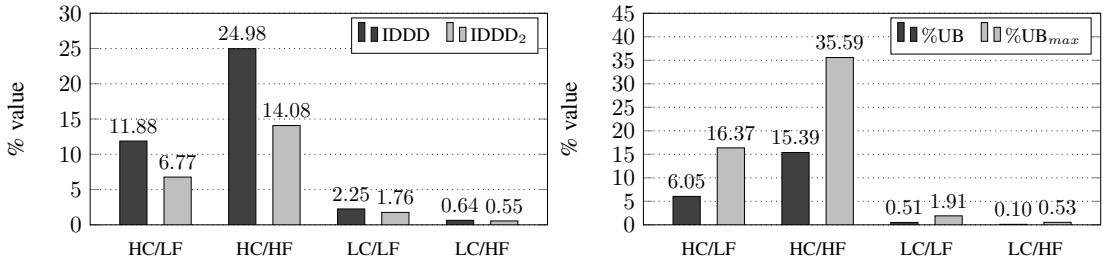
Group	Algorithm	%Gap	Times(s)	#Iterations	%Optimal	%nSize
HC/LF	IDDD	0.99	285.5	13.6	95.6	1.41
183	IDDD_1	0.84	72.3	3.6	100.0	1.30
HC/HF	IDDD	2.34	1377.7	14.8	70.1	0.49
177	IDDD_1	0.91	510.7	4.1	95.5	0.53

IDDD_1: IDDD based on the new refinement strategy.

As indicated by Table 3, the IDDD variant with the new refinement strategy can solve 25% more **HC/HF** instances and 4% more **HC/LF** instances. The algorithm can converge within fewer iterations and shorter computational times. We also observed that the cardinality of the node set of the partially time-expanded network in the last iteration of IDDD_1 remains small and comparable to that of the original IDDD algorithm. These results highlight the effectiveness of the new refinement strategy and demonstrate its substantial impact on the convergence of DDD algorithms.

8.3.3. Effectiveness of the New Upper-Bound Derivation Method To assess the effectiveness of the newly proposed method for deriving upper-bound solutions, we executed algorithm IDDD with the newly proposed upper-bound derivation method, denoted as variant IDDD_2, and compared the results with those of the original IDDD algorithm.

First, we compare the upper-bound solutions obtained from the newly proposed upper-bound derivation method with those obtained from the upper-bound derivation method proposed by Boland et al. (2017). Note that IDDD and IDDD_2 share the same best relaxation solution in the first iteration. This comparison is achieved by comparing the upper-bound solutions of IDDD_2, denoted as UB_2, with the upper-bound solutions of IDDD denoted as UB_1, obtained in the first iteration. Figure 3-(a) illustrates the resulting gaps between these upper-bound solutions and the relaxation solutions in the first iteration achieved by IDDD and IDDD_2, and Figure 3-(b) highlights the improvement in the upper-bound value achieved by the newly proposed upper-bound derivation method compared to the method proposed by Boland et al. (2017), calculated as $\%\text{UB} = 100.0 \times (\text{UB}_1 - \text{UB}_2)/\text{UB}_1$. The comparison depicted in Figure 3-(b) reveals a notable improvement achieved by the newly proposed upper-bound derivation method in contrast to the method proposed by Boland et al. (2017). The maximum improvement ("$\%\text{UB}_{max}$") amounts to as much as 35.59%, substantially reducing the gap between the upper bounds and the given lower bounds.



(a) %Gap Over the First Iteration (b) Improvement on the UB in the First Iteration

Figure 3 Effect of the New MIP for Upper-Bound Solutions (IDDD_2: algorithm IDDD with the newly proposed upper-bound derivation method).

Second, we assess the effectiveness of the newly proposed upper-bound derivation method on the algorithm's convergence. Table 4 presents the results of algorithms IDDD and IDDD_2 on the **HC** class of instances with the same notations introduced in Table 1. Unlike the results shown in Tables 2 and 3, which demonstrate that the new initial relaxation and new refinement strategy significantly affect the algorithm's convergence, the results displayed in Table 4 reveal the limited contribution of the newly proposed upper-bound derivation method alone to the convergence of the algorithm. This is primarily due to its heavy reliance on the routing plan derived from the relaxation solutions. Consequently, if the quality of the relaxation solutions improves slowly, the benefit of a better upper bound becomes increasingly limited. However, the results in Table 1 and Figure 2 demonstrate that an improved method for obtaining upper-bound solutions can still yield certain advantages in optimizing the overall solution. While its impact may be limited

due to poor performance during relaxation and refinement, a more effective upper-bound derivation method can reduce the number of iterations and computational time.

Table 4 Detailed Results for IDDD Variants with Different Methods for Deriving the Upper Bound

Group	Algorithm	%Gap	Times(s)	#Iterations	%Optimal
HC/LF	IDDD	0.99	285.5	13.6	95.6
183	$IDDD_2$	0.95	256.1	8.7	96.7
HC/HF	IDDD	2.34	1377.7	14.8	70.1
177	$IDDD_2$	1.55	1353.1	12.0	70.6

$IDDD_2$: IDDD based on the newly proposed upper bound deriving method.

9. Conclusions and Future Research

This study proposed theoretical advances by introducing novel optimization-based Dynamic Discretization Discovery (DDD) strategies for solving the Continuous-Time Service Network Design Problem (CTSNDP). Specifically, the enhanced DDD algorithm features: i) a new initial relaxation strategy based on timed-node-based time-expanded commodity networks and significant time points identified by solving minimum hitting-set problems; ii) a new MIP-based upper-bound derivation strategy that leverages consolidation planning problems defined over a collection of relaxation solutions; and iii) a new joint minimal-infeasible-pattern-based strategy for refining discretization that generates the final refinement time points by solving a newly defined refinement problem. The enhanced algorithm demonstrated dominant computational performance, becoming the first to optimally solve all classic CTSNDP instances in the literature. These results underscored the algorithm's efficiency and robustness in addressing CTSNDP problems.

The proposed study opens up several promising avenues for future research. From an algorithmic perspective, the enhanced DDD algorithm can be adapted to handle interval-based networks and extended to address other variants of the CTSNDP, such as those involving holding costs and hub capacity constraints. The innovative components of the newly proposed enhanced DDD algorithm could improve solution quality and computational efficiency in these contexts. From a modeling standpoint, the new initial relaxation for the CTSNDP, based on a newly generated initial discretization, could provide bounds that are, on average, only around 2% away from the optimal solution to the CTSNDP. These bounds could be further tightened by identifying additional sets of significant time points. Therefore, exploring a small and sufficient discretization over which the relaxation model can directly produce the optimal continuous-time solution within a predefined optimality gap for the CTSNDP and other transportation problems is intriguing. Moreover, for the refinement of time discretization, one may investigate new and stronger sufficient conditions for eliminating too-long paths. The definition of too-long paths may also be generalized by relaxing the requirement that such paths start at the commodities' origin nodes and by modifying the condition on their lengths. These potential enhancements and extensions would give rise to new refinement problems and, consequently, require new refinement algorithms in future research.

References

Andersen J, Crainic TG, Christiansen M (2009) Service network design with asset management: Formulations and comparative analyses. *Transportation Research Part C: Emerging Technologies* 17(2):197–207.

Boland N, Hewitt M, Marshall L, Savelsbergh M (2017) The continuous-time service network design problem. *Operations Research* 65(5):1303–1321.

Boland N, Hewitt M, Marshall L, Savelsbergh M (2019) The price of discretizing time: a study in service network design. *EURO Journal on Transportation and Logistics* 8(2):195–216.

Boland NL, Savelsbergh MW (2019) Perspectives on integer programming for time-dependent models. *TOP* 27(2):147–173.

Campêlo M, Campos VA, Corrêa RC (2008) On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Applied Mathematics* 156(7):1097–1111.

Crainic TG (2000) Service network design in freight transportation. *European Journal of Operational Research* 122(2):272–288.

Crainic TG, Frangioni A, Gendron B (2001) Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics* 112(1-3):73–99.

Crainic TG, Hewitt M (2021) Service network design. Crainic TG, Gendreau M, Gendron B, eds., *Network Design with Applications to Transportation and Logistics*, 347–382 (Springer).

Crainic TG, Rousseau JM (1986) Multicommodity, multimode freight transportation: A general modeling and algorithmic framework for the service network design problem. *Transportation Research Part B: Methodological* 20(3):225–242.

Damaschke P (2017) Refined algorithms for hitting many intervals. *Information Processing Letters* 118:117–122.

Erera A, Hewitt M, Savelsbergh M, Zhang Y (2013) Improved load plan design through integer programming based local search. *Transportation Science* 47(3):412–427.

Escobar-Vargas D, Teodor Gabriel C (2024) Multi-attribute two-echelon location routing: Formulation and dynamic discretization discovery approach. *European Journal of Operational Research* 314(1):66–78.

Farvolden JM, Powell WB (1994) Subgradient methods for the service network design problem. *Transportation Science* 28(3):256–272.

Fleischer L, Tardos É (1998) Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters* 23(3-5):71–80.

Ford Jr LR, Fulkerson DR (1958a) Constructing maximal dynamic flows from static flows. *Operations Research* 6(3):419–433.

Ford Jr LR, Fulkerson DR (1958b) A suggested computation for maximal multi-commodity network flows. *Management Science* 5(1):97–101.

Garey MR, Johnson DS (1979) *Computers and intractability: A guide to the theory of NP-completeness* (W. H. Freeman and Company, San Francisco).

Golumbic MC (2004) *Algorithmic graph theory and perfect graphs* (Elsevier).

Gurobi Optimization, LLC (2024) Gurobi Optimizer Reference Manual. URL https://www.gurobi.com.

Hall A, Hippler S, Skutella M (2007) Multicommodity flows over time: Efficient algorithms and complexity. *Theoretical Computer Science* 379(3):387–404.

He E, Boland N, Nemhauser G, Savelsbergh M (2022a) An exact algorithm for the service network design problem with hub capacity constraints. *Networks* 80(4):572–596.

He EY, Boland N, Nemhauser G, Savelsbergh M (2022b) Dynamic discretization discovery algorithms for time-dependent shortest path problems. *INFORMS Journal on Computing* 34(2):1086–1114.

Hewitt M (2019) Enhanced dynamic discretization discovery for the continuous time load plan design problem. *Transportation Science* 53(6):1731–1750.

Hewitt M (2022) The flexible scheduled service network design problem. *Transportation Science* 56(4):1000–1021.

Hewitt M, Lehuédé F (2023) New formulations for the scheduled service network design problem. *Transportation Research Part B: Methodological* 172:117–133.

Hewitt M, Lehuédé F (2025) The freight transportation network scheduling problem: An integer programming-based column generation algorithm. *INFORMS Journal on Computing* 38(3):710–728.

Lagos F, Boland N, Savelsbergh M (2022) Dynamic discretization discovery for solving the continuous time inventory routing problem with out-and-back routes. *Computers & Operations Research* 141:105686.

Malaguti E, Monaci M, Toth P (2009) Models and heuristic algorithms for a weighted vertex coloring problem. *Journal of Heuristics* 15(5):503–526.

Marshall L, Boland N, Savelsbergh M, Hewitt M (2021) Interval-based dynamic discretization discovery for solving the continuous-time service network design problem. *Transportation Science* 55(1):29–51.

Pedersen MB, Crainic TG, Madsen OB (2009) Models and Tabu search metaheuristics for service network design with asset-balance requirements. *Transportation Science* 43(2):158–177.

Shu S, Xu Z, Baldacci R (2024) Incorporating holding costs in continuous-time service network design: New model, relaxation, and exact algorithm. *Transportation Science* 58(2):412–433.

Skutella M (2009) An introduction to network flows over time. Cook W, Lovász L, Vygen J, eds., *Research Trends in Combinatorial Optimization: Bonn 2008*, 451–482 (Springer).

Teypaz N, Schrenk S, Cung VD (2010) A decomposition scheme for large-scale service network design with asset management. *Transportation Research Part E: Logistics and Transportation Review* 46(1):156–170.

Van Dyk M, Koenemann J (2024) Sparse dynamic discretization discovery via arc-dependent time discretizations. *Computers & Operations Research* 169:106715.

Vu DM, Hewitt M, Boland N, Savelsbergh M (2020) Dynamic discretization discovery for solving the time-dependent traveling salesman problem with time windows. *Transportation Science* 54(3):703–720.

Wieberneit N (2008) Service network design for freight transportation: a review. *OR Spectrum* 30(1):77–112.

E-companion to the paper titled "New Dynamic Discretization Discovery Strategies for Continuous-Time Service Network Design"

EC.1. Proof of Statements

This section provides the proofs for the various statements presented in the paper.

EC.1.1. Proof of Proposition 1

Proof: Let $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ be an optimal solution of formulation SND($\mathcal{D}^{\hat{\Delta}}_{\hat{\mathcal{T}}}$) (i.e., an optimal CTSNDP solution) of cost $\overline{z}$, where $\hat{\mathcal{T}}$ represents the complete discretization under $\hat{\Delta}$. Let $\overline{\mathcal{A}} = \{((i,t),(j,t+\tau_{ij})) \in \mathcal{A}^{\hat{\Delta}}_{\hat{\mathcal{T}}} : \bar{y}_{ij}^{t,t+\tau_{ij}} > 0\}$ be the set of service arcs traversed by the commodities, and let $\overline{\mathcal{K}}_{((i,t),(j,t+\tau_{ij}))} = \{k \in \mathcal{K} : \bar{x}_{i,j}^{kt\bar{t}} > 0\}$ represent the set of commodities dispatched on each arc $((i,t),(j,t+\tau_{ij})) \in \overline{\mathcal{A}}$, indicated by solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$.

Below, we show that solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ corresponds to a feasible, but not necessarily optimal, solution $(\boldsymbol{x}, \boldsymbol{y})$ of formulation SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$) of cost $z = \overline{z}$ under any given discretization $\mathcal{T}$.

For a given discretization $\mathcal{T}$, according to Boland et al. (2017), solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ corresponds to a feasible, but not necessarily optimal, solution $(\boldsymbol{x}, \boldsymbol{y})$ of formulation SND($\underline{\mathcal{D}_{\mathcal{T}}}$) of cost $z = \overline{z}$. Under the given discretization $\mathcal{T}$, for any arc $a = ((i,t),(j,t+\tau_{ij})) \in \overline{\mathcal{A}}$, there exists an unique arc $\mu(a) = ((i,\rho_i(t)),(j,\sigma(a)))$ in $\underline{\mathcal{A}_{\mathcal{T}}}$ with $\rho_i(t) = \max\{s \in \mathcal{T}_i : s \le t\}$. The existence and uniqueness of arc $\mu(a) = ((i,\rho_i(t)),(j,\sigma(a)))$ in $\underline{\mathcal{A}_{\mathcal{T}}}$ is ensured by Properties 1-4 of partially time-expanded network $\underline{\mathcal{D}_{\mathcal{T}}}$. The mapping of solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ into a solution $(\boldsymbol{x}, \boldsymbol{y})$ of formulation SND($\mathcal{D}_{\mathcal{T}}$) defined by $\mu : \overline{\mathcal{A}} \to \mathcal{A}_{\mathcal{T}}$ with $\mu(a) = ((i,\rho_i(t)),(j,\sigma(a)))$ and computed by the following expressions for each $\tilde{a} = ((i,\tilde{t}),(j,\tilde{t}')) \in \mathcal{A}_{\mathcal{T}}$:

$$x_{ij}^{k\tilde{t}\tilde{t}'} = \sum_{\substack{a=((i,t),(j,t+\tau_{ij}))\in\overline{\mathcal{A}}:\\ \mu(a)=\tilde{a}}} \overline{x}_{ij}^{kt,t+\tau_{ij}} \quad \text{and} \quad y_{ij}^{\tilde{t}\tilde{t}'} = \sum_{\substack{a=((i,t),(j,t+\tau_{ij}))\in\overline{\mathcal{A}}:\\ \mu(a)=\tilde{a}}} \overline{y}_{ij}^{t,t+\tau_{ij}}, \tag{EC.1}$$

corresponds to a feasible solution of formulation SND($\mathcal{D}_{\mathcal{T}}$) of the same cost of solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$.

We now prove that this solution $(\boldsymbol{x}, \boldsymbol{y})$ is also a feasible solution of formulation SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$) by showing that, for each arc $a \in \overline{\mathcal{A}}$, $\mu(a) \in \underline{\mathcal{A}_{\mathcal{T}}}$ is contained in $\underline{\mathcal{A}^k_{\mathcal{T}}}$ for all $k \in \overline{\mathcal{K}}_a$.

For each arc $a = ((i,t),(j,t+\tau_{ij})) \in \overline{\mathcal{A}}$ and each $k \in \overline{\mathcal{K}}_a$, it must be $e^k + \phi^k_{o^k,i} \le t \le l^k - \phi^k_{j,d^k} - \tau_{i,j}$. Therefore, $\mu(a) \in \underline{\mathcal{A}_{\mathcal{T}}}$ must be contained in $\underline{\mathcal{A}^k_{\mathcal{T}}}$ as it satisfies conditions (6)-(i) and (6)-(ii) on $\underline{\mathcal{A}^k_{\mathcal{T}}}$. Therefore, $(\boldsymbol{x}, \boldsymbol{y})$ is also a feasible solution of formulation SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$) of the same cost of solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$. This indicates that SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$) is a relaxation of CTSNDP. Additionally, SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$) is tighter than SND($\underline{\mathcal{D}_{\mathcal{T}}}$) since $\underline{\mathcal{A}^{\mathcal{K}}_{\mathcal{T}}}$ is a subset of $\underline{\mathcal{A}_{\mathcal{T}}}$. Therefore, each feasible solution of SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$) is a feasible solution of SND($\underline{\mathcal{D}_{\mathcal{T}}}$). □

EC.1.2. Proof of Theorem 1

Proof: If there exists a time point $\hat{t} \in \mathcal{T}_i$ such that $\hat{t} \in (l^{k_1} - \phi^{k_1}_{j,d^{k_1}} - \tau_{i,j}, e^{k_2} + \phi^{k_2}_{o^{k_2},i}]$, then according to the definition of each set $\underline{\mathcal{A}^k_{\mathcal{T}}}$, conditions (6), every arc $((i,t),(j,\bar{t})) \in \underline{\mathcal{A}^{k_2}_{\mathcal{T}}}$ must satisfy $t \ge \max\{s \in \mathcal{T}_i :$

$s \le e^{k_2} + \phi^{k_2}_{o^{k_2},i}\} \ge \hat{t}$, and every arc $((i,t'),(j,\bar{t}')) \in \underline{\mathcal{A}^{k_1}_{\mathcal{T}}}$ must satisfy $t' \le \max\{s \in \mathcal{T}_i : s + \tau_{i,j} + \phi^{k_1}_{j,d^{k_1}} \le l^{k_1}\} < \hat{t}$. This implies that $\underline{\mathcal{A}^{k_1}_{\mathcal{T}}}$ and $\underline{\mathcal{A}^{k_2}_{\mathcal{T}}}$ do not have any timed arcs in common associated with the arc (i,j). Then, in model SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$), commodities k_1 and k_2 do not share any x variables with same time indices. This indicates that commodities k_1 and k_2 cannot be consolidated on any arc $((i,t),(j,\bar{t})) \in \underline{\mathcal{A}^{\mathcal{K}}_{\mathcal{T}}}$ in any feasible solution of the relaxation model SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$). $\square$

EC.1.3. Proof of Theorem 2

Proof: Consider a flat solution $\mathcal{S} = (\mathcal{P},\mathcal{C})$, with $\mathcal{P} = \{p^k\}_{k\in\mathcal{K}}$ where each $p^k = (a^k_1,\ldots,a^k_{|p^k|})$ represents the flat path for commodity $k \in \mathcal{K}$ with each $a^k_n \in \mathcal{A}$. Let $(\nu^k_1,...,\nu^k_{|p^k|+1})$ represent the node sequence of path p^k with $\nu^k_1 = o^k$ and $\nu^k_{|p^k|+1} = d^k$. Based on the definition of implementability, the flat solution $\mathcal{S}$ is implementable only if there exist a dispatch scheduling $\mathcal{T}(\mathcal{P},\mathcal{C})$ consisting of a collection of departure times $t^k_{\nu^k_n}$ for $k \in \mathcal{K}$ and $n \in \{1,2,...,|p^k|\}$, such that

$$
\begin{aligned}
&t^k_{\nu^k_1} \ge e^k, \quad \forall k \in \mathcal{K},\\
&t^k_{\nu^k_n} \ge t^k_{\nu^k_{n-1}} + \tau_{a^k_{n-1}}, \quad \forall k \in \mathcal{K}, n = 2,\ldots,|p^k|,\\
&t^k_{\nu^k_{|p^k|}} + \tau_{a^k_{|p^k|}} \le l^k, \quad \forall k \in \mathcal{K},\\
&t^k_i = t^{k'}_i, \quad \forall ((i,j),\kappa) \in \mathcal{C}, k \in \kappa, k' \in \kappa.
\end{aligned}
$$

Equivalently, given its dispatch-node graph $\mathscr{G}(\mathcal{S}) = (\mathscr{V},\mathscr{A})$, the flat solution $\mathcal{S}$ is implementable only if there exists a collection of departure times λ_v for $v \in \mathscr{V}$ over $\mathscr{G}(\mathcal{S})$, such that

$$\lambda_{\eta^k_{o^k}} \ge e^k, \quad \forall k \in \mathcal{K}, \tag{EC.2}$$

$$\lambda_{v'} \ge \lambda_v + \rho(a), \quad \forall a = (v,v') \in \mathscr{A}, \tag{EC.3}$$

$$\lambda_{\eta^k_{d^k}} \le l^k, \quad \forall k \in \mathcal{K}, \tag{EC.4}$$

$$\lambda_{\eta^k_i} = \lambda_{\eta^{k'}_i}, \quad \forall ((i,j),\kappa) \in \mathcal{C}, k \in \kappa, k' \in \kappa. \tag{EC.5}$$

It is evident that $t^k_i = \lambda_{\eta^k_i}$ for all $(i,j) \in p^k$ and $k \in \mathcal{K}$, form a feasible dispatch scheduling $\mathcal{T}(\mathcal{P},\mathcal{C})$, thus establishing the equivalence. We have the following two cases.

(a) The dispatch-node graph $\mathscr{G}(\mathcal{S})$ contains any too-long path $P' \in \mathscr{P}(\mathcal{S})$ with its initial node denoted by $\eta^k_{o^k}$ and final node denoted by $\eta^{k'}_i$ with $\rho(P') > l_{k'} - e^k - \phi^{k'}_{i,d^{k'}}$. We now show that, in this case, there does not exist departure times $\boldsymbol{\lambda}$ that satisfy (EC.2)-(EC.5), implying the flat solution $\mathcal{S}$ can not be implementable.

Consider any path $P = (v^P_1, v^P_2 \ldots, v^P_{|P|}) \in \mathscr{P}(\mathcal{S})$ in the dispatch-node graph $\mathscr{G}(\mathcal{S})$ with $v^P_1 = \eta^{k_1}_{o^{k_1}}$ and $v^P_{|P|} = \eta^{k_2}_{d^{k_2}}$. Feasible departure times $\boldsymbol{\lambda}$ that satisfy (EC.2)-(EC.5) must adhere to the conditions:

$$\lambda_{v^P_1} \ge e^{k_1}, \tag{EC.6}$$

$$\lambda_{v^P_{n+1}} \ge \lambda_{v^P_n} + \rho(a = (v^P_n, v^P_{n+1})), \quad \forall n = 1,...,|P|-1, \tag{EC.7}$$

$$\lambda_{v^P_{|P|}} \le l^{k_2}. \tag{EC.8}$$

Here inequalities (EC.6), (EC.7), and (EC.8), stem from (EC.2), (EC.3), and (EC.4), respectively.

For the too-long path $P' \in \mathscr{P}(\mathcal{S})$ with its initial node denoted by $\eta^k_{o^k}$ and final node denoted by $\eta^{k'}_i$, there must be a path P'' in $\mathscr{G}(\mathcal{S})$ that starts from node $\eta^k_{o^k}$, along path P' to node $\eta^{k'}_i$ and then along the shortest path from $\eta^{k'}_i$ to $\eta^{k'}_{d^{k'}}$ over $\mathscr{G}(\mathcal{S})$. If path P'' is not simple, then inequalities (EC.7) can not be fully satisfied. If path P'' is a simple path, (EC.8) is violated due to $e^k + \rho(P'') > e^k + \rho(P') + \phi^{k'}_{i,d^{k'}} > l_{k'}$. This indicates that there does not exist departure times $\boldsymbol{\lambda}$ that satisfy (EC.2)-(EC.5), implying the flat solution $\mathcal{S}$ can not be implementable.

(b) The dispatch-node graph $\mathscr{G}(\mathcal{S}) = (\mathscr{V}, \mathscr{A})$ does not contain any too-long path. We can show that, in this case, there always exists departure times $\boldsymbol{\lambda}$ that satisfy (EC.2)-(EC.5), implying that the flat solution $\mathcal{S}$ is implementable.

Consider any path $P = (v^P_1, v^P_2 \ldots, v^P_{|P|}) \in \mathscr{P}(\mathcal{S})$ with $v^P_1 = \eta^{k_1}_{o^{k_1}}$ and $v^P_{|P|} = \eta^{k_2}_{i'}$. Let $t(P,g)$ indicate the total transit time of the partial path from the initial node $\eta^{k_1}_{o^{k_1}}$ of P to the g-th node v^P_g of P along path P, and let $T(P,g) = e^k + t(P,g)$ represent the earliest departure time at node v^P_g along P.

Let $E(v) = \max\{T(P,g) : v = v^P_g, P \in \mathscr{P}(\mathcal{S}), g \in \{1,2,\ldots,|P|\}\}$ for all $v \in \mathscr{V}$. We can show that $\boldsymbol{\lambda}$, defined by $\lambda_v = \{\max_{k' \in \kappa} E(\eta^{k'}_i) : v = \eta^k_i, ((i,j),\kappa) \in \mathcal{C} \text{ and } k \in \kappa\}$ if $\exists (v,v') \in \mathscr{A}$, otherwise $\lambda_v = E(v)$, for all $v \in \mathscr{V}$, satisfy inequalities (EC.2)-(EC.5):

(i) As each $\eta^k_{o^k}$, $k \in \mathcal{K}$, lacks incoming arcs, it can only serve as the initial node of some $P \in \mathscr{P}(\mathcal{S})$. For any path $P \in \mathscr{P}(\mathcal{S})$ that passes through node $\eta^k_{o^k}$ for any $k \in \mathcal{K}$, we have that $v^P_1 = \eta^k_{o^k}$ and $T(P,1) = e^k$, implying that $E(\eta^k_{o^k}) = e^k$. It follows that $\lambda_{\eta^k_{o^k}} \geq E(\eta^k_{o^k}) = e^k$. Therefore, we conclude that $\lambda_{\eta^k_{o^k}} \geq e^k$ for all $k \in \mathcal{K}$, indicating that inequalities (EC.2) are satisfied.

(ii) For each $(i,j) \in p^k$, $k \in \mathcal{K}$, we have that $\lambda(\eta^k_i) = E(\eta^{k^*}_i)$ with $k^* = \arg\max\{E(\eta^{k'}_i) : ((i,j),\kappa) \in \mathcal{C}, k \in \kappa \text{ and } k' \in \kappa\}$, and there must exist an arc $(\eta^{k^*}_i, \eta^k_j) \in \mathscr{A}$ according to the definition of the dispatch-node graph $\mathscr{G}(\mathcal{S}) = (\mathscr{V}, \mathscr{A})$. Let $(P^*, g^*) = \arg\max\{T(P,g) : \eta^{k^*}_i = v^P_g, P \in \mathscr{P}(\mathcal{S}), g \in \{1,2,\ldots,|P|\}\}$ and let $\underline{P}$ represent the partial path of P^* from its origin node to $\eta^{k^*}_i$. There must exist a path $P' \in \mathscr{P}(\mathcal{S})$ such that $\underline{P} \subseteq P'$ and $v^{P'}_{g^*+1} = \eta^k_j$. This implies that $\lambda(\eta^k_j) \geq E(\eta^k_j) \geq T(P', g^*+1) = T(P', g^*) + \tau_{i,j} = E(\eta^{k^*}_i) + \tau_{i,j} = \lambda(\eta^k_i) + \tau_{i,j}$. Therefore, we conclude that $\lambda_{\eta^k_j} \geq \lambda_{\eta^k_i} + \tau_{i,j}$ for all $k \in \mathcal{K}$ and $(i,j) \in p^k$, indicating that inequalities (EC.3) are satisfied.

(iii) For each $k \in \mathcal{K}$, for every path $P \in \mathscr{P}(\mathcal{S})$ that pass through node $\eta^k_{d^k}$ with $v^P_{\hat{g}} = \eta^k_{d^k}$, we have that $T(P,\hat{g}) \leq l^k$ as the dispatch-node graph $\mathscr{G}(\mathcal{S})$ does not contain any too-long path, indicating $E(\eta^k_{d^k}) \leq l^k$. We note that $\lambda_{\eta^k_{d^k}} = E(\eta^k_{d^k})$ as there dose not exists any arc $(\eta^k_{d^k}, v) \in \mathscr{A}$. This implies that $\lambda_{\eta^k_{d^k}} \leq d^k$ for all $k \in \mathcal{K}$, indicating that inequalities (EC.4) are satisfied.

(iv) Moreover, $\lambda_{\eta^{k_1}_i} = \lambda_{\eta^{k_2}_i} = \max_{k' \in \kappa} E(\eta^{k'}_i)$ holds for all $((i,j),\kappa) \in \mathcal{C}$, $k_1 \in \kappa$, and $k_2 \in \kappa$, indicating that inequalities (EC.5) are satisfied.

Therefore, the flat solution $\mathcal{S} = (\mathcal{P}, \mathcal{C})$ is implementable.

To sum up the above, we conclude that a flat solution $\mathcal{S}$ is non-implementable if and only if its dispatch-node graph $\mathscr{G}(\mathcal{S})$ contains a too-long path. $\square$

EC.1.4. Proof of Lemma 1

Proof. By the definition of representabiltiy, a flat solution $\mathcal{S}=(\mathcal{P},\mathcal{C})$ is representable in $\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}'}}$ when there exists a feasible solution $(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})$ of SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}'}}$) that utilize this flat solution, i.e., $(\mathcal{P}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}),\mathcal{C}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}))=(\mathcal{P},\mathcal{C})$. Consider a path $P=(\eta^{k^P_1}_{i^P_1},\eta^{k^P_2}_{i^P_2},\ldots,\eta^{k^P_{|P|}}_{i^P_{|P|}})$ in the dispatch-node graph $\mathscr{G}(\mathcal{S})$ of this flat solution $\mathcal{S}$. It follows that such solution $(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})$ satisfies 1) $\hat{x}^{kt_h\bar{t}_{h+1}}_{i^P_h,i_{h+1}}=1$ for some arc $((i^P_h,t_h),(i^P_{h+1},\bar{t}_{h+1}))\in\underline{\mathcal{A}^{\mathcal{K}}_{\mathcal{T}'}}$ for all $k\in\{k^P_h,k^P_{h+1}\}$ and $h=1,...,|P|-1$, and 2) $t_h\geq\bar{t}_h$ for all $h=2,...,|P|-1$. These two conditions stem from the constraints (8)-(9) of SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}'}}$), and the first condition implies that $((i^P_h,t_h),(i^P_{h+1},\bar{t}_{h+1}))\in\underline{\mathcal{A}^{k}_{\mathcal{T}'}}$ for all $k\in\{k^P_h,k^P_{h+1}\}$ and $h=1,...,|P|-1$. By connecting these arcs $((i^P_h,t_h),(i^P_{h+1},\bar{t}_{h+1}))\in\underline{\mathcal{A}^{\mathcal{K}}_{\mathcal{T}'}}$, $h=1,...,|P|-1$, with appropriate holding arcs, we obtain a path in $\underline{\mathcal{D}_{\mathcal{T}'}}=(\underline{\mathcal{N}_{\mathcal{T}'}},\underline{\mathcal{A}_{\mathcal{T}'}}\cup\underline{\mathcal{H}_{\mathcal{T}'}})$ as $\underline{\mathcal{A}^{\mathcal{K}}_{\mathcal{T}'}}\subseteq\underline{\mathcal{A}_{\mathcal{T}'}}$.

We thus conclude that if a flat solution $\mathcal{S}$ is representable in $\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}'}}$, then for each path $P=(\eta i^P_1{}^{k^P_1},\eta^{k^P_2}_{i^P_2},\ldots,\eta^{k^P_{|P|}}_{i^P_{|P|}})$ contained in its dispatch-node graph $\mathscr{G}_(\mathcal{S})$, there must exist a timed path, denoted by $R(P,\mathcal{T}')$, in $\underline{\mathcal{D}_{\mathcal{T}'}}$ from some timed node (i^P_1,t_1) to some timed node $(i^P|P|,\bar{t}|P|)$, whose service arcs $((i^P_h,t_h),(i^P_{h+1},\bar{t}_{h+1}))\in\underline{\mathcal{A}_{\mathcal{T}'}}$ for $h=1,\ldots,|P|-1$ belong to $\underline{\mathcal{A}^{k^P_h}_{\mathcal{T}'}}\cap\underline{\mathcal{A}^{k^P_{h+1}}_{\mathcal{T}'}}$.

Taking the contrapositive, for a minimal too-long path P, if no such timed path $R(P,\mathcal{T}')$ exists, then no flat solution representable in $\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}'}}$ can have a dispatch-node graph containing P. Hence, P is eliminated under $\mathcal{T}'$. The lemma is thus proved. $\square$

EC.1.5. Proof of Lemma 2

Proof. Consider any discretization $\mathcal{T}'$ and any minimal too-long path $P\in\mathscr{P}$. By Lemma 1, $\mathcal{T}'\in\text{ELIM}(P,\mathcal{T},\hat{\mathcal{T}})$ if there does not exist a timed path $R(P,\mathcal{T}')$ in $\underline{\mathcal{D}_{\mathcal{T}'}}=(\underline{\mathcal{N}_{\mathcal{T}'}},\underline{\mathcal{A}_{\mathcal{T}'}}\cup\underline{\mathcal{H}_{\mathcal{T}'}})$ from some timed node (i^P_1,ℓ^P_1) to some timed node $(i^P_{|P|},\bar{\ell}^P_{|P|})$ such that each of its service arcs its service arcs $((i^P_h,\ell^P_h),(i^P_{h+1},\bar{\ell}^P_{h+1}))\in\underline{\mathcal{A}_{\mathcal{T}'}}$ belongs to $\underline{\mathcal{A}^{k^P_h}_{\mathcal{T}'}}\cap\underline{\mathcal{A}^{k^P_{h+1}}_{\mathcal{T}'}}$ for all $h=1,\ldots,|P|-1$. By condition (ii) of (6), any such timed path must satisfy

$$\underline{\psi}^P_h(\mathcal{T}')\leq\ell^P_h\leq\pi^P_h,\qquad h=1,\ldots,|P|-1. \tag{EC.9}$$

We then construct the earliest timed path $\underline{R}(P,\mathcal{T}')$ for P. By construction, its departure times are defined recursively by

$$t^P_1=\underline{\psi}^P_1(\mathcal{T}'),$$

Properties 3 and 4 of $\underline{\mathcal{D}_{\mathcal{T}'}}$ ensure that the initial departure time t^P_1 uniquely determines the corresponding timed service arc $((i^P_1,t^P_1),(i^P_2,\bar{t}^P_2))$. Recursively, for each $h=2,\ldots,|P|-1$,

$$t^P_h=\min\left\{t\in\mathcal{T}'_{i^P_h}:t\geq\max\{\underline{\psi}^P_h(\mathcal{T}'),\bar{t}^P_h\}\right\}. \tag{EC.10}$$

where the departure time t^P_h is uniquely determined by (EC.10), and Properties 3 and 4 of $\underline{\mathcal{D}_{\mathcal{T}'}}$ then ensure that t^P_h uniquely determines the corresponding timed service arc $((i^P_h,t^P_h),(i^P_{h+1},\bar{t}^P_{h+1}))$. Moreover, since

$t_h^P \geq \bar{t}_h^P$ for $h = 2, \ldots, |P| - 1$, consecutive service arcs are uniquely connected by the appropriate holding arcs in $\underline{\mathcal{H}_{\mathcal{T}'}}$. Hence, $\underline{R}(P, \mathcal{T}')$ is the unique earliest timed path associated with P.

We next compare any feasible timed path $R(P, \mathcal{T}')$, if one exists, with $\underline{R}(P, \mathcal{T}')$. By induction on h, the departure time ℓ_h^P of any timed path satisfying the conditions of Lemma 1 must satisfy

$$\ell_h^P \geq t_h^P, \qquad h = 1, \ldots, |P| - 1.$$

Indeed, this is immediate for $h = 1$ from (EC.9), since $t_1^P = \underline{\psi}_1^P(\mathcal{T}')$. If $\ell_h^P \geq t_h^P$, then the arrival time $\bar{\ell}_{h+1}^P$ of $R(P, \mathcal{T}')$ at node i_{h+1}^P is no earlier than $\bar{t}_{h+1}^P$; therefore, its next departure must also be no earlier than the recursive choice defining t_{h+1}^P. This proves the induction.

It follows that if there exists $h \in \{1, \ldots, |P| - 1\}$ such that $t_h^P > \pi_h^P$, then every timed path $R(P, \mathcal{T}')$ has $\ell_h^P \geq t_h^P > \pi_h^P$, which violates (EC.9). Hence, no such timed path $R(P, \mathcal{T}')$ exists, and by Lemma 1, $\mathcal{T}' \in \text{ELIM}(P, \mathcal{T}, \hat{\mathcal{T}})$.

Conversely, suppose that $t_h^P \leq \pi_h^P$ for all $h = 1, \ldots, |P| - 1$. Since $\underline{R}(P, \mathcal{T}')$ satisfies $t_h^P \geq \underline{\psi}_h^P(\mathcal{T}')$ by construction and $t_h^P \leq \pi_h^P$ by assumption, all of its service arcs satisfy (EC.9). Thus, $\underline{R}(P, \mathcal{T}')$ itself is a timed path satisfying the condition in Lemma 1. Therefore, $\mathcal{T}' \notin \text{ELIM}(P, \mathcal{T}, \hat{\mathcal{T}})$. Equivalently, $\mathcal{T}' \in \text{ELIM}(P, \mathcal{T}, \hat{\mathcal{T}})$ holds if and only if there exists at least one h such that $t_h^P > \pi_h^P$.

This proves the lemma. □

EC.1.6. Proof of Theorem 3

Given a discretization $\mathcal{T}'$, Lemma 2 establishes a sufficient condition such that $\mathcal{T}' \in \text{ELIM}(P, \mathcal{T}, \hat{\mathcal{T}})$. To prove Theorem 3, we demonstrate that this condition is met by incorporating the time points defined in Theorem 3 to the discretization. Theorem 3 uses the following Lemma 3, which gives a lower bound on the dispatch times from terminals when the discretization $\mathcal{T}$ contains some specific time points.

Lemma 3 *Let $\hat{P}$ be a timed path in $\underline{\mathcal{D}_{\mathcal{T}}} = (\underline{\mathcal{N}_{\mathcal{T}}}, \underline{\mathcal{A}_{\mathcal{T}}} \cup \underline{\mathcal{H}_{\mathcal{T}}})$ with its service arcs denoted by $((i_h, t_h), (i_{h+1}, \bar{t}_{h+1}))$ for $h \in \{1, 2, \ldots, n\}$, where $i_1 = o^k$ and $t_1 \geq e^k$ for some $k \in \mathcal{K}$. Suppose $e^k + \tau(\hat{P}, h) \in \mathcal{T}_{i_h}$, for all $h = 1, ..., n$, where $\tau(\hat{P}, 1) = 0$ and $\tau(\hat{P}, h) = \sum_{m=1}^{h-1} \tau_{i_m, i_{m+1}}$ for $h = 2, ..., n$. We have that $t_h \geq e^k + \tau(\hat{P}, h)$ for all $h = 1, ..., n$.*

Proof: Suppose $e^k + \tau(\hat{P}, h) \in \mathcal{T}_{i_h}$, for all $h = 1, ..., n$. For any arc $((i_h, t), (i_{h+1}, \bar{t})) \in \underline{\mathcal{A}_{\mathcal{T}}}$ with $h = 1, ..., n$, if $t \geq e^k + \tau(\hat{P}, h)$, we must have $\bar{t} \geq e^k + \tau(\hat{P}, h + 1)$ as indicated by Properties 2-4 of the partially time-expanded network. Consider the timed path $\hat{P}$ in $\underline{\mathcal{D}_{\mathcal{T}}} = (\underline{\mathcal{N}_{\mathcal{T}}}, \underline{\mathcal{A}_{\mathcal{T}}} \cup \underline{\mathcal{H}_{\mathcal{T}}})$. If $t_h \geq e^k + \tau(\hat{P}, h)$ holds for some $h \in \{1, ..., n-1\}$, we must have $t_{h+1} \geq \bar{t}_{h+1} \geq e^k + \tau(\hat{P}, h+1)$. When $h = 1$, $t_h \geq e^k + \tau(\hat{P}, h)$ holds as $\tau(\hat{P}, 1) = 0$, $e^k \in \mathcal{T}_{i_1}$ and $t_1 \geq e^k$. This indicates that $t_h \geq e^k + \tau(\hat{P}, h)$ also holds for all $h = 2, ..., n$. We thus conclude that $t_h \geq e^k + \tau(\hat{P}, h)$ for all $h = 1, ..., n$. □

Based on Lemma 3, Theorem 3 can be proved as follows.

Proof. Consider any too-long path $P \in \mathscr{P}(\mathcal{S})$ in the dispatch-node graph $\mathscr{G}(\mathcal{S})$ of a flat solution $\mathcal{S}$ that is representable in $\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$, where $P = (\eta^{k_1^P}_{i_1^P}, \eta^{k_2^P}_{i_2^P}, \ldots, \eta^{k_{|P|}^P}_{i_{|P|}^P})$. Refine $\mathcal{T}$ to $\mathcal{T}'$ by setting

$$\mathcal{T}'_j = \mathcal{T}_j \cup \{e^{k_1^P} + \rho(P,g) : g \in \{1,2,\ldots,|P|-1\},\ i_g^P = j\}, \quad j \in \mathcal{N}.$$

We then construct the unique earliest timed path $\underline{R}(P,\mathcal{T}')$ in $\underline{\mathcal{D}_{\mathcal{T}'}}$ for P, whose service arcs are denoted by $((i_h^P, t_h^P), (i_{h+1}^P, \bar{t}_{h+1}^P)) \in \underline{\mathcal{A}_{\mathcal{T}'}}$, with departure times defined recursively by

$$t_1^P = \underline{\psi}_1^P(\mathcal{T}'),$$
$$t_h^P = \min\left\{t \in \mathcal{T}'_{i_h^P} : t \geq \max\{\underline{\psi}_h^P(\mathcal{T}'), \bar{t}_h^P\}\right\}, \quad h = 2,\ldots,|P|-1.$$

By Lemma 3, since $e^{k_1^P} + \rho(P,g) \in \mathcal{T}'_{i_g^P}$ for all $g = 1,\ldots,|P|-1$, it follows that $t^P_{|P|-1} \geq e^{k_1^P} + \rho(P,|P|-1)$. Let $k^* = k^P_{|P|}$. Since P is a too-long path, $e^{k_1^P} + \rho(P) + \phi^{k^*}_{i^P_{|P|},d^{k^*}} > l^{k^*}$. Consequently,

$$t^P_{|P|-1} \geq e^{k_1^P} + \rho(P,|P|-1) > l^{k^*} - \left(\tau_{i^P_{|P|-1},i^P_{|P|}} + \phi^{k^*}_{i^P_{|P|},d^{k^*}}\right) \geq \pi^P_{|P|-1}.$$

Therefore, by Lemma 2, we have $\mathcal{T}' \in \mathrm{ELIM}(P,\mathcal{T},\hat{\mathcal{T}})$. □

EC.2. Example Illustrating Theorem 1

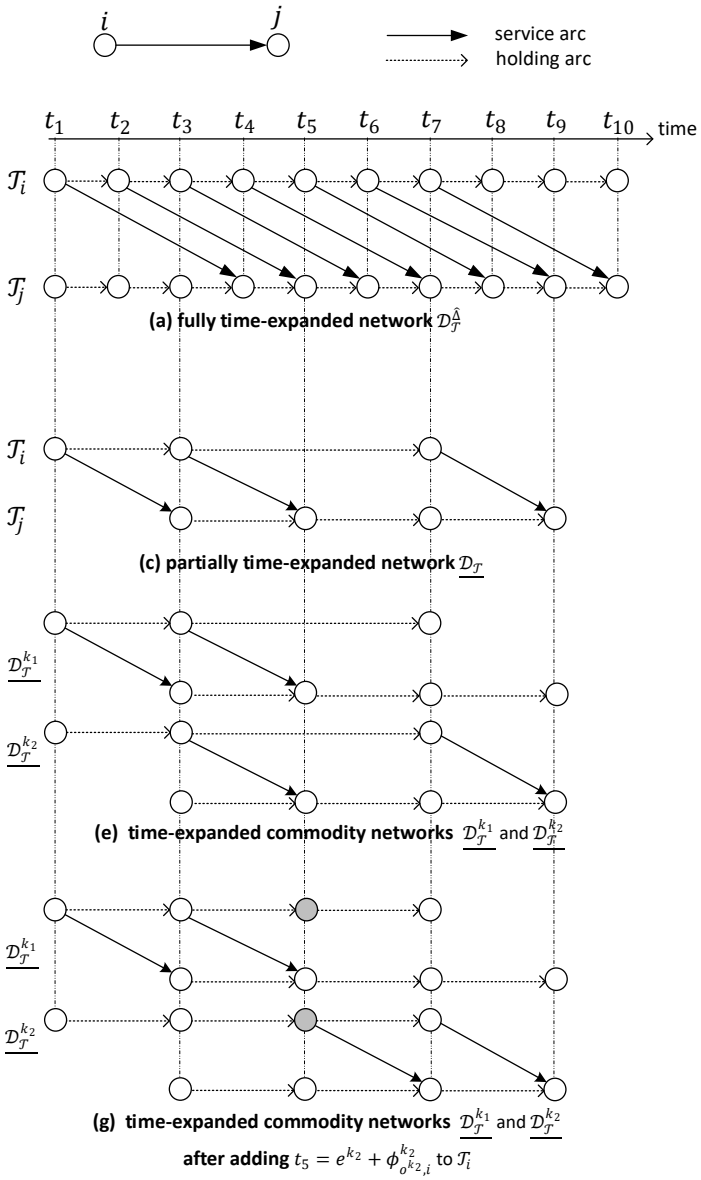
Figure EC.1 illustrates Theorem 1 using an example. According to Figure EC.1-(b), it is evident that no feasible solution of the CTSNDP can consolidate commodities k_1 and k_2 on arc (i,j) as $e^{k_2} + \phi^{k_2}_{o^{k_2},i} + \tau_{i,j} + \phi^{k_1}_{j,d^{k_1}} > l^{k_1}$. However, based on the time-expanded commodity networks shown in Figure EC.1-(e), SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$) does contain a feasible solution where commodities k_1 and k_2 are consolidated on arc $((i,t_3),(j,t_5))$ and represent an impossible consolidation. Figure EC.1-(f) clearly indicates that both commodities k_1 and k_2 have an x variable related to arc $((i,t_3),(j,t_5))$. By including $t_5 \in (l^{k_1} - \phi^{k_1}_{j,d^{k_1}} - \tau_{i,j}, e^{k_2} + \phi^{k_2}_{o^{k_2},i}]$ into $\mathcal{T}_i$, we obtain new time-expanded commodity networks as shown in Figure EC.1-(g), along with the corresponding $\boldsymbol{x}$ variables as shown in Figure EC.1-(h). Notably, the two new time-expanded commodity networks do not share any service arcs as arc $((i,t_3),(j,t_5))$ is removed in $\underline{\mathcal{D}^{k_2}_{\mathcal{T}}}$ due to $t_3 < t_5 = \arg\max\{s \in \mathcal{T}_i : s \leq e^{k_2} + \phi^{k_2}_{o^{k_2},i}\}$, implying that the two commodities do not have any x variable associated with the same service arc. Therefore, based on the time-expanded commodity networks depicted in Figure EC.1-(g), the updated SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$) does not contain any feasible solution where commodities k_1 and k_2 are consolidated on arc (i,j).

EC.3. Additional Details of the EDDD Algorithm

This section gives additional details about the algorithms used to create the initial network (§EC.3.1) and solving model CPPMIP($\boldsymbol{x},\boldsymbol{y}$) (§EC.3.2), as well as illustrative examples of the dispatch-node graph, (minimal) too-long paths, and joint refinement (§EC.3.3 and §EC.3.4).

EC.3.1. Creating the Initial Timed-Node-Based Time-Expanded Commodity Networks

Details on identifying significant time points and constructing the initial time-expanded commodity networks are provided in Algorithm 2.



commodity	$e^k + \phi_{o^k,i}^k$	$l^k - \phi_{j,d^k}^k - \tau_{i,j}$	$l^k - \phi_{j,d^k}^k$
k_1	t_1	t_4	t_7
k_2	t_5	t_7	t_{10}

(b) Delivery information

commodity	x variables for service arcs
k_1	$x_{i,j}^{k_1 t_1 t_3}, x_{i,j}^{k_1 t_3 t_5}, x_{i,j}^{k_1 t_7 t_9}$
k_2	$x_{i,j}^{k_2 t_1 t_3}, x_{i,j}^{k_2 t_3 t_5}, x_{i,j}^{k_2 t_7 t_9}$

(d) Variables based on partially time-expanded network

commodity	x variables for service arcs
k_1	$x_{i,j}^{k_1 t_1 t_3}$, $\boldsymbol{x_{i,j}^{k_1 t_3 t_5}}$
k_2	$\boldsymbol{x_{i,j}^{k_2 t_3 t_5}}$, $x_{i,j}^{k_2 t_7 t_9}$

(f) Variables based on time-expanded commodity networks

commodity	x variables for service arcs
k_1	$x_{i,j}^{k_1 t_1 t_3}, x_{i,j}^{k_1 t_3 t_5}$
k_2	$x_{i,j}^{k_2 t_5 t_7}, x_{i,j}^{k_2 t_7 t_9}$

(h) Variables based on new time-expanded commodity networks

Figure EC.1 Example Illustrating Theorem 1

The algorithm initializes the discretization $\mathcal{T}$ (lines 1-7). Subsequently, significant time points are added to the discretization (lines 8-18). Based on the discretization, the timed-node-based time-expanded network is initialized (lines 19-27), and the corresponding timed-node-based time-expanded commodity networks are created (lines 28-30).

EC.3.2. Symmetry Breaking and Variable Reductions for Model CPPMIP(x, y)

The model CPPMIP$(\boldsymbol{x}, \boldsymbol{y})$ exhibits a high degree of symmetry, leading to numerous redundant solutions that can complicate its solution. We observe that, for each $(i,j) \in \mathcal{A}(\boldsymbol{x}, \boldsymbol{y})$, any feasible solution $\{z_{ijr}^k\}_{k \in \mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y}), r \in \{1,2,\ldots,|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|\}}$ is a feasible solution of the graph coloring problem with given $|\mathcal{K}_{i,j}(\boldsymbol{x}, \boldsymbol{y})|$ colors, where the available consolidations on the arc are defined as colors, commodities act as

Algorithm 2: Generate the initial Timed-Node-Based Time-Expanded Commodity Networks

Input: Flat network $\mathcal{D}=(\mathcal{N},\mathcal{A})$, commodity set $\mathcal{K}$;
Output: Timed-Node-Based Time-Expanded Commodity Networks $\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$;
begin
// Generate the initial discretization by adding the fundamental time points based on network properties
Set $\mathcal{T}_i=\emptyset$ and $W_i=\emptyset$ for all $i\in\mathcal{N}$;
for $k\in\mathcal{K}$ **do**
Set $\mathcal{T}_{o^k}=\mathcal{T}_{o^k}\cup\{e^k\}$ and $\mathcal{T}_{d^k}=\mathcal{T}_{d^k}\cup\{d^k\}$
end
for $i\in\mathcal{N}$ **do**
Set $\mathcal{T}_i=\mathcal{T}_i\cup\{\min_{k\in\mathcal{K}}\{e^k+\phi_{o^k,i}\}\}$
end
// Generate the initial discretization by identifying and adding significant time points to address impossible consolidations
for $i\in\mathcal{N}$ **do**
for $(i,j)\in\mathcal{A}$ **do**
for $(k_1,k_2)\in\mathcal{K}_{i,j}\times\mathcal{K}_{i,j}$ *with* $e^{k_1}+\phi^{k_1}_{o^{k_1},i}+\tau_{i,j}+\phi^{k_2}_{j,d^{k_2}}>l^{k_2}$ **do**
Identify time window $\omega_{i,j}^{k_1,k_2}=(\omega_{ij,o}^{k_1,k_2},\omega_{ij,d}^{k_1,k_2}]$;
if *there does not exist any time point* $t\in\mathcal{T}_i$ *with* $t\in\omega_{i,j}^{k_1,k_2}$ **then**
Set $W_i=W_i\cup\{\omega_{i,j}^{k_1,k_2}\}$;
end
end
end
Solve the minimum hitting-set problem for intervals in W_i to find the minimum hitting set $\mathcal{M}_i$ for W_i and Set $\mathcal{T}_i=\mathcal{T}_i\cup\mathcal{M}_i$;
end
// Construct the timed-node-based time-expanded commodity networks
for $i\in\mathcal{N}$ *and* $t\in\mathcal{T}_i$ **do**
Add node (i,t) to $\underline{\mathcal{N}_{\mathcal{T}}}$;
end
for $(i,t)\in\underline{\mathcal{N}_{\mathcal{T}}}$ **do**
for $(i,j)\in\mathcal{A}$ **do**
Add arc $((i,t),(j,t'))$ where $t'=\arg\max\{s\in\mathcal{T}_j:s\le t+\tau_{i,j}\}$ to $\underline{\mathcal{A}_{\mathcal{T}}}$;
end
Find the smallest t' such that $(i,t')\in\mathcal{N}_{\mathcal{T}}$ and $t'>t$ and add $((i,t),(j,t'))$ to $\underline{\mathcal{H}_{\mathcal{T}}}$;
end
for $k\in\mathcal{K}$ **do**
Construct $\underline{\mathcal{D}_{\mathcal{T}}^{k}}=(\underline{\mathcal{N}_{\mathcal{T}}},\underline{\mathcal{A}_{\mathcal{T}}^{k}}\cup\underline{\mathcal{H}_{\mathcal{T}}})$ based on conditions (6) for $\underline{\mathcal{A}_{\mathcal{T}}^{k}}$;
end
return $\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}=\{\underline{\mathcal{D}_{\mathcal{T}}^{k}}\}_{k\in\mathcal{K}}$;
end

the graph's vertices, and the pairs (k,k') that cannot be consolidated on arc (i,j) if they follow the delivery paths $p^k(\boldsymbol{x},\boldsymbol{y})$ and $p^{k'}(\boldsymbol{x},\boldsymbol{y})$ define the edges. Any feasible solution requires that each vertex (commodity) be assigned a color (consolidation), and that any two endpoints of an edge (two conflicting commodities) be assigned different colors. The asymmetric representatives strategy, widely adopted in the graph coloring literature (Campêlo et al. 2008, Malaguti et al. 2009), can thus be utilized to effectively break the solution symmetry and enhance the model's solvability.

The core of the asymmetric representatives strategy is to associate a representative commodity with each consolidation (indexed by r), enforcing that the r-th consolidation can be used to transport other commodities only if its representative commodity has already been assigned to that consolidation. For each $(i,j) \in \mathcal{A}(\boldsymbol{x},\boldsymbol{y})$, we define the conflict set $\mathcal{F}_{i,j}(\boldsymbol{x},\boldsymbol{y}) = \{(k,k') : e^k + \tilde{\phi}^k_{o^k i} + \tau_{i,j} + \tilde{\phi}^k_{j,d^{k'}} > l^{k'}$ or $e^{k'} + \tilde{\phi}^k_{o^{k'} i} + \tau_{i,j} + \tilde{\phi}^k_{j,d^k} > l^k, k,k' \in \mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y}), k < k'\}$, where $\tilde{\phi}^k_{i,j}$ represents the length of the partial path from node i to node j in $p^k(\boldsymbol{x},\boldsymbol{y})$. Each pair $(k,k') \in \mathcal{F}_{i,j}(\boldsymbol{x},\boldsymbol{y})$ means that commodities k and k' cannot be consolidated on arc (i,j) if they follow the delivery paths $p^k(\boldsymbol{x},\boldsymbol{y})$ and $p^{k'}(\boldsymbol{x},\boldsymbol{y})$, respectively. A subset of $\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})$ is considered a *conflict subset* if each pair of commodities in the subset can form an element in $\mathcal{F}_{i,j}(\boldsymbol{x},\boldsymbol{y})$. We then sort $\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})$ as $(\tilde{k}_1,...,\tilde{k}_n,\bar{k}_1,...,\bar{k}_m)$ where set $\{\tilde{k}_1,...,\tilde{k}_n\}$ forms a conflict subset of $\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})$ containing n commodities, i.e., $\forall k,k' \in \{\tilde{k}_1,...,\tilde{k}_n\}$, $k<k'$, we have $(k,k') \in \mathcal{F}_{i,j}(\boldsymbol{x},\boldsymbol{y})$, and where $\bar{k}_1,...,\bar{k}_m$ are the remaining commodities. For presentational convenience, we re-index the elements in $\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})$ and obtain $\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y}) = \{k_1,k_2,...,k_{n+m}\}$. We thus define k_r in $\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})$ as the representative of r-th consolidation, enforcing that any commodity $k_{r'} \in \mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})$ can be included in the r-th consolidation only if $r \le r'$ and the representative k_r is also assigned to it. Consequently, we ensure that if $(k,k') \in \mathcal{F}_{i,j}(\boldsymbol{x},\boldsymbol{y})$, commodity k will never be included in the consolidation with representative k'.

We then implement this acceleration strategy for each $(i,j) \in \mathcal{A}(\boldsymbol{x},\boldsymbol{y})$ through the following two stages:

1. Apply the following reduction steps:
   (a) *Step 1*: for each k_h with $h \in \{1,...,|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|\}$, remove all variables $z^{k_h}_{ijr}$ with $r > h$;
   (b) *Step 2*: for each $r \in \{1,...,|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|\}$, remove all variables $z^{k_h}_{ijr}$ with $(k_r,k_h) \in \mathcal{F}_{i,j}(\boldsymbol{x},\boldsymbol{y})$ or $(k_h,k_r) \in \mathcal{F}_{i,j}(\boldsymbol{x},\boldsymbol{y})$;
   (c) *Step 3*: for each k_h with $h \in \{1,...,n\}$, let $z^{k_h}_{ijh} = 1$, which also means that the variable can also be removed from the model;
   (d) *Step 4*: All redundant constraints relevant to these removed redundant variables can be accordingly removed.

   In Step 1, symmetries are eliminated, and in Step 2, infeasible consolidations are excluded based on the conflict set $\mathcal{F}_{i,j}(\boldsymbol{x},\boldsymbol{y})$. Step 3 further streamlines the model by fixing binary variables with a required value of 1 or directly removing them. Additionally, Step 4 indicates that certain constraints in the CPPMIP($\boldsymbol{x},\boldsymbol{y}$) model can be relaxed due to variable removal.
2. Add the following inequalities with the reduced $\boldsymbol{z}$ variables:
$$z^{k_{h'}}_{ijh} \le z^{k_h}_{ijh}, \quad \forall h \in \{n+1,...,|\mathcal{F}_{i,j}(\boldsymbol{x},\boldsymbol{y})|\}, h' \in \{h,...,|\mathcal{F}_{i,j}(\boldsymbol{x},\boldsymbol{y})|\}, \tag{EC.11}$$
   that further break the symmetric structure of the solutions of the remaining model by restricting that a commodity $k_{h'}$ can be included in the h-th consolidation on arc (i,j) only if the representative commodity k_h is included in the h-th consolidation.

Sorted $\mathcal{K}_{ij}(\boldsymbol{x},\boldsymbol{y})$	$(k_1,k_2,k_3,k_4,k_5,k_6)$
Maximal independent (conflict) subset	(k_1,k_2,k_3)
$\mathcal{F}_{ij}(\boldsymbol{x},\boldsymbol{y})$	(k_1,k_2) (k_1,k_3) (k_1,k_4) (k_2,k_3) (k_2,k_5) (k_3,k_6) (k_5,k_6)

(a) Sets $\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})$ and $\mathcal{F}_{i,j}(\boldsymbol{x},\boldsymbol{y})$

r	1	2	3	4	5	6
$\boldsymbol{z}$	$z_{ij1}^{k_1}$	$z_{ij2}^{k_1}$	$z_{ij3}^{k_1}$	$z_{ij4}^{k_1}$	$z_{ij5}^{k_1}$	$z_{ij6}^{k_1}$
	$z_{ij1}^{k_2}$	$z_{ij2}^{k_2}$	$z_{ij3}^{k_2}$	$z_{ij4}^{k_2}$	$z_{ij5}^{k_2}$	$z_{ij6}^{k_2}$
	$z_{ij1}^{k_3}$	$z_{ij2}^{k_3}$	$z_{ij3}^{k_3}$	$z_{ij4}^{k_3}$	$z_{ij5}^{k_3}$	$z_{ij6}^{k_3}$
	$z_{ij1}^{k_4}$	$z_{ij2}^{k_4}$	$z_{ij3}^{k_4}$	$z_{ij4}^{k_4}$	$z_{ij5}^{k_4}$	$z_{ij6}^{k_4}$
	$z_{ij1}^{k_5}$	$z_{ij2}^{k_5}$	$z_{ij3}^{k_5}$	$z_{ij4}^{k_5}$	$z_{ij5}^{k_5}$	$z_{ij6}^{k_5}$
	$z_{ij1}^{k_6}$	$z_{ij2}^{k_6}$	$z_{ij3}^{k_6}$	$z_{ij4}^{k_6}$	$z_{ij5}^{k_6}$	$z_{ij6}^{k_6}$

(b) Original Variables

r	1	2	3	4	5	6
$\boldsymbol{z}$	$z_{ijr_1}^{k_1}$	$\boldsymbol{z_{ijr_2}^{k_1}}$	$\boldsymbol{z_{ijr_3}^{k_1}}$	$\boldsymbol{z_{ijr_4}^{k_1}}$	$\boldsymbol{z_{ijr_5}^{k_1}}$	$\boldsymbol{z_{ijr_6}^{k_1}}$
	$z_{ijr_1}^{k_2}$	$z_{ijr_2}^{k_2}$	$\boldsymbol{z_{ijr_3}^{k_2}}$	$\boldsymbol{z_{ijr_4}^{k_2}}$	$\boldsymbol{z_{ijr_5}^{k_2}}$	$\boldsymbol{z_{ijr_6}^{k_2}}$
	$z_{ijr_1}^{k_3}$	$z_{ijr_2}^{k_3}$	$z_{ijr_3}^{k_3}$	$\boldsymbol{z_{ijr_4}^{k_3}}$	$\boldsymbol{z_{ijr_5}^{k_3}}$	$\boldsymbol{z_{ijr_6}^{k_3}}$
	$z_{ijr_1}^{k_4}$	$z_{ijr_2}^{k_4}$	$z_{ijr_3}^{k_4}$	$z_{ijr_4}^{k_4}$	$\boldsymbol{z_{ijr_5}^{k_4}}$	$\boldsymbol{z_{ijr_6}^{k_4}}$
	$z_{ijr_1}^{k_5}$	$z_{ijr_2}^{k_5}$	$z_{ijr_3}^{k_5}$	$z_{ijr_4}^{k_5}$	$z_{ijr_5}^{k_5}$	$\boldsymbol{z_{ijr_6}^{k_5}}$
	$z_{ijr_1}^{k_6}$	$z_{ijr_2}^{k_6}$	$z_{ijr_3}^{k_6}$	$z_{ijr_4}^{k_6}$	$z_{ijr_5}^{k_6}$	$z_{ijr_6}^{k_6}$

(c) Reduction Step 1

r	1	2	3	4	5	6
$\boldsymbol{z}$	$z_{ij1}^{k_1}$					
	$\boldsymbol{z_{ij1}^{k_2}}$	$z_{ij2}^{k_2}$				
	$\boldsymbol{z_{ij1}^{k_3}}$	$\boldsymbol{z_{ij2}^{k_3}}$	$z_{ij3}^{k_3}$			
	$\boldsymbol{z_{ij1}^{k_4}}$	$z_{ij2}^{k_4}$	$z_{ij3}^{k_4}$	$z_{ij4}^{k_4}$		
	$z_{ij1}^{k_5}$	$\boldsymbol{z_{ij2}^{k_5}}$	$z_{ij3}^{k_5}$	$z_{ij4}^{k_5}$	$z_{ij5}^{k_5}$	
	$z_{ij1}^{k_6}$	$z_{ij2}^{k_6}$	$\boldsymbol{z_{ij3}^{k_6}}$	$z_{ij4}^{k_6}$	$\boldsymbol{z_{ij5}^{k_6}}$	$z_{ij6}^{k_6}$

(d) Reduction Step 2

r	1	2	3	4	5	6
$\boldsymbol{z}$	$\boldsymbol{z_{ij1}^{k_1}}$					
		$\boldsymbol{z_{ij2}^{k_2}}$				
			$\boldsymbol{z_{ij3}^{k_3}}$			
		$z_{ij2}^{k_4}$	$z_{ij3}^{k_4}$	$z_{ij4}^{k_4}$		
	$z_{ij1}^{k_5}$		$z_{ij3}^{k_5}$	$z_{ij4}^{k_5}$	$z_{ij5}^{k_5}$	
	$z_{ij1}^{k_6}$	$z_{ij2}^{k_6}$		$z_{ij4}^{k_6}$		$z_{ij6}^{k_6}$

(e) Reduction Step 3

r	1	2	3	4	5	6
$\boldsymbol{z}$		$z_{ij2}^{k_4}$	$z_{ij3}^{k_4}$	$z_{ij4}^{k_4}$		
	$z_{ij1}^{k_5}$		$z_{ij3}^{k_5}$	$z_{ij4}^{k_5}$	$z_{ij5}^{k_5}$	
	$z_{ij1}^{k_6}$	$z_{ij2}^{k_6}$		$z_{ij4}^{k_6}$		$z_{ij6}^{k_6}$

(f) Reduced Variables

r	1	2	3	4	5	6
$\boldsymbol{z}$		$z_{ij2}^{k_4}=1$	$z_{ij3}^{k_4}$	$z_{ij4}^{k_4}$		
	$z_{ij1}^{k_5}$		$z_{ij3}^{k_5}$	$z_{ij4}^{k_5}=1$	$z_{ij5}^{k_5}$	
	$z_{ij1}^{k_6}=1$	$z_{ij2}^{k_6}$		$z_{ij4}^{k_6}$		$z_{ij6}^{k_6}$
consolidations	(k_1,k_6)	(k_2,k_4)	(k_3)	(k_5)		

r	1	2	3	4	5	6
$\boldsymbol{z}$		$z_{ij2}^{k_4}=1$	$z_{ij3}^{k_4}$	$z_{ij4}^{k_4}$		
	$z_{ij1}^{k_5}$		$z_{ij3}^{k_5}$	$z_{ij4}^{k_5}$	$z_{ij5}^{k_5}=1$	
	$z_{ij1}^{k_6}=1$	$z_{ij2}^{k_6}$		$z_{ij4}^{k_6}$		$z_{ij6}^{k_6}$
consolidations	(k_1,k_6)	(k_2,k_4)	(k_3)		(k_5)	

(g) Two Equivalent Solutions

r	1	2	3	4	5	6
$\boldsymbol{z}$		$z_{ij2}^{k_4}$	$z_{ij3}^{k_4}$	$z_{ij4}^{k_4}$		
	$z_{ij1}^{k_5}$		$z_{ij3}^{k_5}$	$z_{ij4}^{k_5}$	$z_{ij5}^{k_5}$	
	$z_{ij1}^{k_6}$	$z_{ij2}^{k_6}$		$z_{ij4}^{k_6}$		$z_{ij6}^{k_6}$
valid inequalities	$z_{ij4}^{k_5} \le z_{ij4}^{k_4}$					
	$z_{ij4}^{k_6} \le z_{ij4}^{k_4}$					

(h) Valid Inequalities

Figure EC.2 Example of the Acceleration Strategy

Figure EC.2 illustrates an example that explains the abovementioned acceleration strategy. Let's consider an arc $(i,j) \in \mathcal{A}(\boldsymbol{x},\boldsymbol{y})$. Figure EC.2-(a) provides information about $\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})$ and $\mathcal{F}_{i,j}(\boldsymbol{x},\boldsymbol{y})$, and Figure EC.2-(b) lists the original $\boldsymbol{z}$ variables associated with arc (i,j). Solutions of CPPMIP$(\boldsymbol{x},\boldsymbol{y})$ exhibit a high degree of symmetry with many redundant $\boldsymbol{z}$ variables. Figures EC.2-(c), EC.2-(d), and EC.2-(e) show the redundant $\boldsymbol{z}$ variables that can be removed in each reduction step, respectively. As illustrated in Figure EC.2-(f), we finally obtain the reduced set of $\boldsymbol{z}$ variables, retaining only 31% of the original $\boldsymbol{z}$ variables. After applying these reduction steps, the proposed CPPMIP$(\boldsymbol{x},\boldsymbol{y})$ model can be significantly streamlined, thereby eliminating most redundant asymmetric solutions. However, as shown in Figure EC.2-(g), even with the reduced $\boldsymbol{z}$ variables, there may still be equivalent solutions that have the same consolidation plan on arc (i,j) but have different $\boldsymbol{z}$ variable solutions. The symmetric structure of the solutions of the reduced CPPMIP$(\boldsymbol{x},\boldsymbol{y})$ can be fully eliminated by adding inequalities (EC.11). Figure EC.2-(h) displays all inequalities (EC.11) identified with the reduced $\boldsymbol{z}$ variables.

By employing the two-stage approach, we ensure that every feasible consolidation on an arc (i,j) is uniquely associated with a single consolidation index r. This result implies that the number of feasible solutions $\boldsymbol{z}$, as well as the number of $\boldsymbol{z}$ variables, is independent of the internal ordering of commodities within the sets $\{\tilde{k}_1,...,\tilde{k}_n\}$ and $\{\bar{k}_1,...,\bar{k}_m\}$. However, the number of $\boldsymbol{z}$ variables remains dependent on the size of the set $\{\tilde{k}_1,...,\tilde{k}_n\}$ due to the reduction step 3 of stage 1. Consistent with practices in graph coloring (Campêlo et al. 2008), it is recommended to sort $\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})$ according to its maximum-size conflict subset for each $(i,j) \in \mathcal{A}(\boldsymbol{x},\boldsymbol{y})$ to minimize the number of $\boldsymbol{z}$ variables in CPPMIP$(\boldsymbol{x},\boldsymbol{y})$. A maximum-size conflict subset of $\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})$ can be obtained by solving a maximum independent set problem, a well-know $\mathcal{NP}$-hard problem (Garey and Johnson 1979), over an undirected graph $\mathcal{G}_{i,j}$ that is generated according to $\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})$ and $\mathcal{F}_{i,j}(\boldsymbol{x},\boldsymbol{y})$. The undirected graph $\mathcal{G}_{i,j}$ can be constructed as follows. Let $\mathcal{V}^{\mathcal{G}}_{i,j}$ and $\mathcal{A}^{\mathcal{G}}_{i,j}$ denote the vertex set and arc set in $\mathcal{G}_{i,j}$, respectively. For each k in $\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})$, there is a vertex ν_k in $\mathcal{V}^{\mathcal{G}}_{i,j}$. For each pair of vertexes $\nu_k \in \mathcal{V}^{\mathcal{G}}_{i,j}$ and $\nu_{k'} \in \mathcal{V}^{\mathcal{G}}_{i,j}$ with $k < k'$, if $(k,k') \notin \mathcal{F}_{i,j}(\boldsymbol{x},\boldsymbol{y})$, then there is an undirected arc between ν_k and $\nu_{k'}$ in $\mathcal{A}^{\mathcal{G}}_{i,j}$, and nodes ν_k and $\nu_{k'}$ are called adjacent. Finding a maximum-size conflict subset of $\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})$ is thus equivalent to finding a maximum independent set of graph $\mathcal{G}_{i,j} = (\mathcal{V}^{\mathcal{G}}_{i,j}, \mathcal{A}^{\mathcal{G}}_{i,j})$, which is a vertex set with pairwise non-adjacent elements and with maximum cardinality.

Our computational study reveals that the size of each $\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})$ is typically small for a given relaxation solution $(\boldsymbol{x},\boldsymbol{y})$. Consequently, each graph $\mathcal{G}_{i,j}$ is sufficiently compact, allowing for the direct derivation of its maximum independent set using general-purpose MIP solvers. Furthermore, our computational study demonstrates that, following the application of the acceleration strategy, the resulting CPPMIP$(\boldsymbol{x},\boldsymbol{y})$ models can be effectively solved by general optimization solvers (where constants M^{k+}_{ijr} and M^{k-}_{ijr} can be practically specified by the latest departure time for commodity k on arc (i,j)) and the latest departure time for all

commodities in $\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})$ on arc (i,j)), respectively.) We also note that the model can be further enhanced by incorporating the following valid inequalities:

$$\gamma_{ijr} \geq \left\lceil \frac{q^{k_r}}{u_{i,j}} \right\rceil z^{k_r}_{ijr}, \quad \forall (i,j) \in \mathcal{A}, r \in \{1,2,...,|\mathcal{K}_{i,j}|\}, \tag{EC.12}$$

$$\sum_{r=1}^{|\mathcal{K}_{i,j}|} \gamma_{ijr} \geq \underline{\gamma}_{i,j}, \quad \forall (i,j) \in \mathcal{A}, \tag{EC.13}$$

We know that $\underline{\gamma}_{i,j} \geq \lceil \sum_{k\in\mathcal{K}_{i,j}} q^k/u_{i,j} \rceil$, and the value of this lower bound $\underline{\gamma}_{i,j}$ can be obtained by solving a graph coloring problem for each arc (i,j).

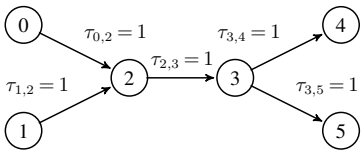
The routing plan $\mathcal{P}(\boldsymbol{x},\boldsymbol{y})$, combined with the consolidation plan z^k_{ijr} and dispatch times δ^k_i for all $k \in \mathcal{K}$ and $i \in \mathcal{N}^k(\boldsymbol{x},\boldsymbol{y})$ prescribed by the optimal solution of the CPPMIP$(\boldsymbol{x},\boldsymbol{y})$ model, constitutes a feasible solution to the CTSNDP with a total cost equal to the objective value of the CPPMIP$(\boldsymbol{x},\boldsymbol{y})$ model.

EC.3.3. Examples illustrating dispatch-node graph and (minimal) too-long paths

Figure EC.3 illustrates the dispatch-node graph and the too-long paths over it. Specifically, Figures EC.3-(a) and EC.3-(b) introduce the flat network $\mathcal{D}$ and the commodity delivery requirements. Figures EC.3-(c) and EC.3-(d) present two different flat solutions $\mathcal{S}_1$ and $\mathcal{S}_2$. Their corresponding solution graphs $\mathscr{D}(\mathcal{S}_1)$ and $\mathscr{D}(\mathcal{S}_2)$ are depicted in Figures EC.3-(e) and EC.3-(g), respectively, while their dispatch-node graphs $\mathscr{G}(\mathcal{S}_1)$ and $\mathscr{G}(\mathcal{S}_2)$ are shown in Figures EC.3-(f) and EC.3-(h), respectively. We note that, in the solution graph, each consolidation node $s^\kappa_{i,j}$ with $\kappa = \{k\}$ is redundant (being incident to only two arcs, $(\eta^k_i, s^\kappa_{i,j})$ and $(s^\kappa_{i,j}, \eta^k_j)$) and could be eliminated by replacing the two arcs with (η^k_i, η^k_j). In this paper, we retain these nodes $s^\kappa_{i,j}$ in $\mathscr{D}(\mathcal{S})$ for the sake of notation simplicity. It can be seen that the dispatch-node graph retains all dispatch nodes and their connectivity relationships in the respective solution graph, but contains no consolidation nodes. For a given flat solution, the solution graph represents its routing and consolidation plan. In contrast, the dispatch-node graph emphasizes the relationships among the arrival times of commodities, as indicated by its routing and consolidation plan. From both dispatch-node graphs $\mathscr{G}(\mathcal{S}_1)$ and $\mathscr{G}(\mathcal{S}_2)$, two too-long paths P_1 and P_2 can be identified as illustrated in Figure EC.3-(i), due to $e^{k_1} + \rho(P_1) = 1 + 3 > l^{k_3} = 3$ and $e^{k_1} + \rho(P_2) = 1 + 2 > l^{k_3} - \phi^{k_4}_{3,4} = 3 - 1 = 2$. Since P_2 is a partial too-long path of P_1, and any partial path of P_2, except P_2 itself, is not a too-long path, P_2 is identified as a minimal too-long path, while P_1 is not. Figure EC.3-(j) shows the paths in solution graphs $\mathscr{D}(\mathcal{S}_1)$ and $\mathscr{D}(\mathcal{S}_2)$ associated with the minimal too-long path P_2. More specifically, paths P and P' share the same sequence of dispatch nodes as path P_2. It is evident that paths P and P' are different but are associated with the same too-long path P_2 in their respective dispatch-node graphs $\mathscr{G}(\mathcal{S}_1)$ and $\mathscr{G}(\mathcal{S}_2)$. This demonstrates our rationale for refining based on too-long paths in dispatch-node graphs rather than on infeasible patterns in solution graphs.

EC.3.4. Illustrative Examples for Joint Refinement of Too-long Paths

To illustrate how our joint refinement approach proposed in Step 3 can generate a refinement plan with fewer time points than the separate approach in Marshall et al. (2021), consider a minimal too-long path



(a) Flat network $\mathcal{D}$

k	o^k	d^k	e^k	l^k
k_1	0	4	1	4
k_2	0	4	0	4
k_3	2	5	1	3
k_4	1	4	0	4

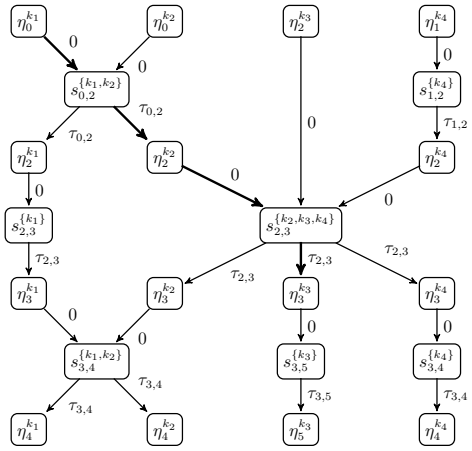
(b) Commodity information

$\mathcal{P}$		$\mathcal{C}$	
k	p^k	(i,j)	κ
k_1	(0,2,3,4)	$(0,2)$	$\{k_1,k_2\}$
k_2	(0,2,3,4)	$(1,2)$	$\{k_4\}$
k_3	(2,3,5)	$(2,3)$	$\{k_1\}$
k_4	(1,2,3,4)	$(2,3)$	$\{k_2,k_3,k_4\}$
		$(3,4)$	$\{k_1,k_2\}$
		$(3,4)$	$\{k_4\}$
		$(3,5)$	$\{k_3\}$

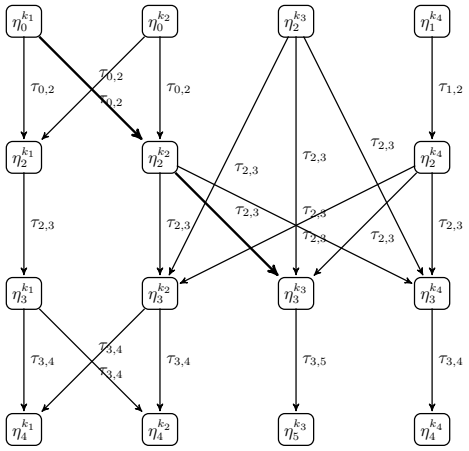
(c) Flat solution $\mathcal{S}_1$

$\mathcal{P}$		$\mathcal{C}$	
k	p^k	(i,j)	κ
k_1	(0,2,3,4)	$(0,2)$	$\{k_1,k_2\}$
k_2	(0,2,3,4)	$(1,2)$	$\{k_4\}$
k_3	(2,3,5)	$(2,3)$	$\{k_1\}$
k_4	(1,2,3,4)	$(2,3)$	$\{k_2,k_3\}$
		$(2,3)$	$\{k_4\}$
		$(3,4)$	$\{k_1,k_2\}$
		$(3,4)$	$\{k_4\}$
		$(3,5)$	$\{k_3\}$

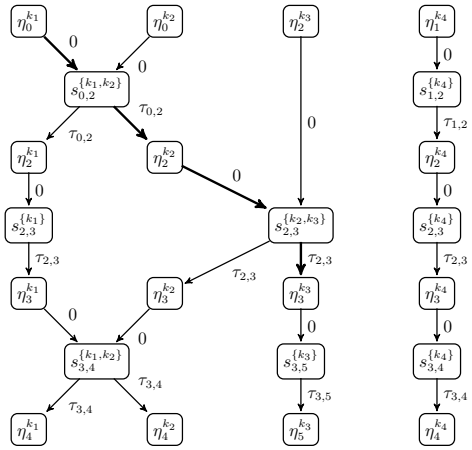
(d) Flat solution $\mathcal{S}_2$



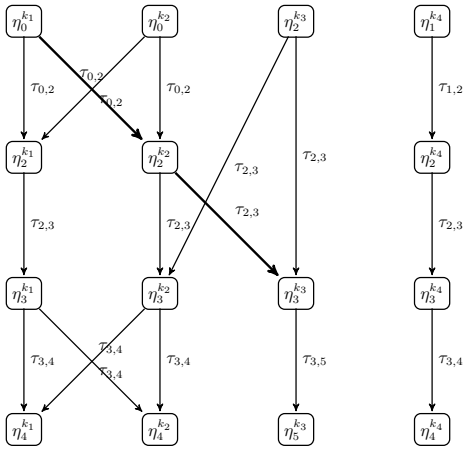
(e) Solution graph $\mathscr{D}(\mathcal{S}_1)$



(f) Dispatch-node graph $\mathscr{G}(\mathcal{S}_1)$



(g) Solution graph $\mathscr{D}(\mathcal{S}_2)$



(h) Dispatch-node graph $\mathscr{G}(\mathcal{S}_2)$

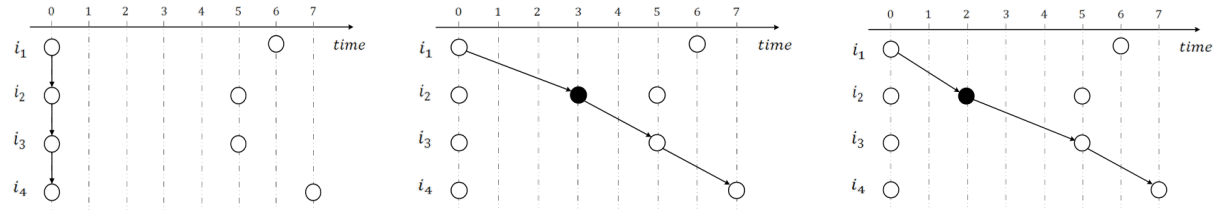
P_1	$(\eta_0^{k_1},\eta_2^{k_2},\eta_3^{k_3},\eta_5^{k_3})$
P_2	$(\eta_0^{k_1},\eta_2^{k_2},\eta_3^{k_3})$

(i) Too-long paths over $\mathscr{G}(\mathcal{S}_1)$ and $\mathscr{G}(\mathcal{S}_2)$

P in $\mathscr{D}(\mathcal{S}_1)$	$(\eta_0^{k_1},s_{0,2}^{\{k_1,k_2\}},\eta_2^{k_2},s_{2,3}^{\{k_2,k_3,k_4\}},\eta_3^{k_3})$
P' in $\mathscr{D}(\mathcal{S}_2)$	$(\eta_0^{k_1},s_{0,2}^{\{k_1,k_2\}},\eta_2^{k_2},s_{2,3}^{\{k_2,k_3\}},\eta_3^{k_3})$

(j) Paths in $\mathscr{D}(\mathcal{S}_1)$, $\mathscr{D}(\mathcal{S}_2)$ associated with P_2

Figure EC.3 Examples of flat solutions, solution graphs, dispatch-node graphs, and (minimal) too-long paths



(a) The shortest timed path $\Phi_{\mathcal{T}}(P)$ under discretization $\mathcal{T}$ (b) The shortest timed path $\underline{\Phi}_{\mathcal{T}'}(P)$ under discretization $\mathcal{T}'$ (c) The shortest timed path $\underline{\Phi}_{\mathcal{T}''}(P)$ under discretization $\mathcal{T}''$

Figure EC.4 Examples Illustrating Time Point Reduction in Step 3

$P=(\eta_{i_1}^{k_1},\eta_{i_2}^{k_2},\eta_{i_3}^{k_3},\eta_{i_4}^{k_4})$ over $\mathscr{G}(\mathcal{S})$, with each edge of length 3, where $e^{k_1}=0$, $l^{k_3}-\phi^{k}_{i_3,d^{k_3}}\geq 6$, and $\pi_3^P=4$. The current (partial) discretization $\mathcal{T}$ and the corresponding earliest timed path $\underline{R}(P,\mathcal{T})$ over $\underline{\mathcal{D}_{\mathcal{T}}}$ are shown in Figure EC.4(a). To eliminate P, by applying Theorem 3, Step 2 of our refinement matheuristic needs to add time point 0 to $\mathcal{T}_{i_1}$, time point 3 to $\mathcal{T}_{i_2}$, and time point 6 to $\mathcal{T}_{i_3}$. The minimal too-long path P corresponds to a specific structure P' in the solution graph $\mathscr{D}(\mathcal{S})$, where $\mathscr{D}(\mathcal{S})$ includes both dispatch and consolidation nodes. We note that the set of time points identified in Step 2 of our refinement matheuristic for eliminating P is identical to that obtained by Marshall et al. (2021) for eliminating the structure P'.

First, we examine the case of eliminating a single minimal too-long path P. Here, Step 3 of our refinement matheuristic solves the problem $\mathrm{R}(\mathcal{T},\bar{\mathcal{T}}^*,\mathscr{P})$ to generate a final refinement based on the candidate points $\bar{\mathcal{T}}^*_{i_1}=\{0\}$, $\bar{\mathcal{T}}^*_{i_2}=\{3\}$, and $\bar{\mathcal{T}}^*_{i_3}=\{6\}$. Notably, there exists a plan to eliminate path P using fewer time points by adding only 0 and 3 to $\mathcal{T}_{i_1}$ and $\mathcal{T}_{i_2}$, respectively. The resulting refined discretization $\mathcal{T}'$ and the corresponding earliest timed path $\underline{R}(P,\mathcal{T}')$ over $\underline{\mathcal{D}_{\mathcal{T}'}}$, are shown in Figure EC.4(b). As illustrated, $\underline{R}(P,\mathcal{T}')$ passes through timed node $(i_2,3)$ and leaves i_3 at time 5 via arc $((i_3,5),(i_4,7))$, which exceeds $\pi_3^P=4$. Therefore, the condition in Lemma 2 is satisfied, and path P is effectively eliminated by the current discretization $\mathcal{T}'$. Consequently, adding time point 6 to $\mathcal{T}_{i_3}$ is unnecessary. This refined discretization $\mathcal{T}'$ thus constitutes a feasible solution to $\mathrm{R}(\mathcal{T},\bar{\mathcal{T}}^*,\mathscr{P})$, already reducing the number of time points compared to the method in Marshall et al. (2021).

Next, consider the case where multiple minimal too-long paths must be eliminated. Suppose there is another minimal too-long path $\bar{P}=((i_5,k_5),(i_2,k_6),(i_3,k_7),(i_4,k_8))$ that is independent of path P (i.e., they originate from disconnected subgraphs of $\mathscr{G}(\mathcal{S})$). According to Theorem 3, eliminating $\bar{P}$ requires adding time point 2 to $\mathcal{T}_{i_2}$. In this scenario, Step 3 solves $\mathrm{R}(\mathcal{T},\bar{\mathcal{T}}^*,\mathscr{P})$ to generate a final refinement plan using the candidate sets $\bar{\mathcal{T}}^*_{i_1}=\{0\}$, $\bar{\mathcal{T}}^*_{i_2}=\{2,3\}$, $\{6\}\subseteq\bar{\mathcal{T}}^*_{i_3}$, and specific points in $\bar{\mathcal{T}}^*_{i_5}$. Interestingly, path P can now be eliminated efficiently by leveraging the points added for $\bar{P}$. Specifically, after adding time points 0 to $\mathcal{T}_{i_1}$ and 2 to $\mathcal{T}_{i_2}$, we obtain the refined discretization $\mathcal{T}''$. As shown in Figure EC.4(c), the corresponding earliest timed path $\underline{R}(P,\mathcal{T}'')$ starts at $(i_1,0)$ and reaches i_4 at time 7 via arc $((i_3,5),(i_4,7))$. Since the departure time from node i_3 exceeds $\pi_3^P=4$, the condition in Lemma 2 is satisfied, and path P is effectively eliminated by $\mathcal{T}''$. Consequently, it becomes unnecessary to add time point 3 to $\mathcal{T}_{i_2}$ or time point 6 to $\mathcal{T}_{i_3}$ specifically for P. In other words, the time point 2 for i_2 that is required to eliminate $\bar{P}$ also helps eliminate P, thereby reducing the total number of time points needed to eliminate both paths.

EC.4. Additional Experiments Results

EC.4.1. Computational Performance on CTSNDP Variant of Hewitt and Lehuédé (2023, 2025)

Recently, Hewitt and Lehuédé (2023, 2025) introduced a time-index–free, consolidation-based formulation for the Freight Transportation Network Scheduling Problem (FTNSP), a variant of the CTSNDP with predefined commodity delivery paths. However, enumerating all feasible shipment combinations results in an exponential number of variables, severely limiting scalability. To mitigate this issue, Hewitt and Lehuédé (2025) proposed a dynamic column-generation algorithm built upon the consolidation-based formulation. We therefore also conducted additional experiments to evaluate the performance of the EDDD algorithm against the consolidation-based formulation (denoted as *Cons-FINSP*($\mathcal{C}$)) and its column generation algorithm (denoted as *IP-ColGen*) proposed by Hewitt and Lehuédé (2025) over their instances of the FTNSP. These FTNSP instances comprise 25 terminals and 530 physical service arcs, with 100 to 750 commodities.

Given that the FTNSP incorporates predefined commodity delivery paths, we implement a refined EDDD procedure comprising the following key modifications. First, for each commodity k, the commodity time-expanded network only contains the timed arcs associated with the service arcs in its predefined delivery path. Second, all relevant shortest travel times are defined strictly in accordance with the specified delivery paths. Third, Property 1 of the partially time-expanded network is relaxed: rather than including all nodes (o^k, e^k) and (d^k, l^k) for every commodity $k \in \mathcal{K}$ during Steps 2–4 of Algorithm 2, we incorporate only the timed nodes $(i, \min\{e^k \mid o^k = i, k \in \mathcal{K}\})$ and $(i, \min\{l^k \mid d^k = i, k \in \mathcal{K}\})$ for each $i \in \mathcal{N}$. Consequently, the nodes $(o^k, \arg\max_{t \in \mathcal{T}_i}\{t \mid t \le e^k\})$ and $(d^k, \max\{t \in \mathcal{T}_i \mid t \le l^k\})$ are utilized in constraints (8). Fourth, a set of tightening inequalities is incorporated:

$$\sum_{((i,t),(j,\bar{t}))\in \underline{\mathcal{A}}^{\mathcal{K}}_{\mathcal{T}}} y^{t\bar{t}}_{i,j} \ge \underline{y}_{i,j}, \quad \forall (i,j) \in \mathcal{A}, \tag{EC.14}$$

where $\underline{y}_{i,j}$ is the lower bound on the number of vehicles required to serve all commodities in $\mathcal{K}_{i,j}$, which is redefined as the set of commodities k whose delivery path uses arc (i,j). We know that $\underline{y}_{i,j} \ge \lceil \sum_{k\in\mathcal{K}_{i,j}} q^k/u_{ij} \rceil$, and the value of this lower bound $\underline{y}_{i,j}$ can be obtained by solving a graph coloring problem for each arc (i,j) with commodities in $\mathcal{K}_{i,j}$. Moreover, since each CPP shares the same routing plan, we adapt the CPP subproblem to serve as a feasibility check for the CTSNDP. Specifically, we execute the CPP using two initial warm-start solutions: one derived from the consolidation plan for the current relaxation and another that serves as a baseline, in which each commodity is delivered independently. Each CPP terminates as soon as a feasible solution is identified.

Following the experimental settings in Hewitt and Lehuédé (2025), we solve EDDD on an Intel(R) Core(TM) i7-8700 desktop PC (3.20 GHz, 64 GB RAM) using Gurobi (v.10.0.2, single thread) with a 1% optimality tolerance and a two-hour time limit. Table EC.1 presents the computational results, reporting the average percentage of instances solved to optimality, and the average and maximum running time across

all instances. The results for *Cons-FINSP*($\mathcal{C}$) and *IP-ColGen* are taken directly from Hewitt and Lehuédé (2025), conducted on a machine equipped with 200 GB of RAM and 64 Intel Xeon Gold 6130 CPU processors (2.10 GHz) with CPLEX 22.10.

Table EC.1 Computational results on FTNSP Instances of Hewitt and Lehuédé (2025).

Method	Percentage solved (%)	Time (s)	Max Time (s)
Cons-FINSP($\mathcal{C}$)	89.29	354.27	7200.00
IP-ColGen	91.96	189.37	7200.00
EDDD	100.00	6.75	143.7

The results in Table EC.1 show that our EDDD solves all the FTNSP instances to optimality, with an average runtime of less than 7 seconds and a maximum of 143.7 seconds. Although the results of EDDD and those of *Cons-FINSP*($\mathcal{C}$) and *IP-ColGen* were obtained using different hardware and software configurations, the substantial reduction in both average and maximum computational times demonstrates that the enhanced DDD algorithm proposed in this study is sufficiently flexible to be adapted to CTSNDP variants, while also providing substantial computational advantages.

EC.4.2. Computational Effort Required for Key Components of EDDD

Below Table EC.2 details the computational effort required for key components of EDDD: solving the relaxation models ("**MIP time**(s)"); deriving the Upper Bound (UB) by solving the CPPMIPs ("**CPP time**(s)"); and executing the refinement steps ("**Refinement time**(s)"). This shows that the majority of the computational effort is spent solving the relaxation models ("**MIP time**(s)"), while the CPPMIPs and refinement steps are completed efficiently, often within seconds (on average).

Table EC.2 Computational Efforts for Key Components of EDDD

	HC/LF	**HC/HF**
MIP time(s)	6.3	119.0
CPP time(s)	1.5	6.7
Refinement time(s)	0.1	0.8

We note that the MIP time generally increases as iterations progress due to the expanding size of the time-expanded network, consistent with previous DDD implementations. In contrast, the CPP and Refinement times do not follow a strictly monotonic trend; their durations depend on the structural changes in the discretization and the solution investigated at each step.

EC.4.3. Effectiveness of Sequential Batch Heuristic in Step 3 of Refinement Matheuristic

To evaluate the effectiveness of the sequential batch heuristic used in Step 3 of our refinement matheuristic, we examine the number of time points it adds in the first iteration of the EDDD algorithm. Compared with the optimal solution of the refinement MIP model $\mathrm{R}(\mathcal{T}, \bar{\mathcal{T}}^*, \mathscr{P})$, the sequential batch heuristic adds only 3.54% more time points on average, indicating its near-optimal performance. Compared with the candidate set $\bar{\mathcal{T}}^*$ generated in Step 2 of the refinement matheuristic, the sequential batch heuristic reduces the number of added time points by 50.03%, showing its strong effectiveness in removing redundant candidate time points for joint refinement.

To further assess the effect of the solution method used in the third refinement step, we compared our EDDD approach, which employs the sequential batch heuristic, with a variant denoted by EDDD_R, which solves each MIP model $\mathrm{R}(\mathcal{T}, \bar{\mathcal{T}}^*, \mathscr{P})$ exactly using Gurobi. The results are summarized in Table EC.3. In addition to the metrics defined according to the conventions of Table 1, column "%Fsize" reports the relative size of the final discretization, column "Refinement time(s)" presents the average and maximum computational time required for the refinement procedure, and column "%Rini" indicates the relative increase in the number of time points added by our EDDD in the first iteration compared with EDDD_R.

Table EC.3 Detailed Results for EDDD Variants using Heuristic or Exact Solver in Step 3 of Refinement

Group	Algorithm	%Gap	Times(s)	#Iterations	%Optimal	%Rini	%Fsize	Refinement time(s)	
								avg	max
HCLF	EDDD	0.69	8.6	1.5	100.0	2.42	1.56	0.1	1.1
183	EDDD_R	0.69	9.3	1.5	100.0		1.56	0.3	4.3
HCHF	EDDD	0.83	127.2	2.7	100.0	4.10	0.57	0.8	11.6
177	EDDD_R	0.82	141.8	2.8	100.0		0.57	9.5	811.5

The results in Table EC.3 show that when integrated with the proposed initial relaxation and upper-bound derivation method, EDDD_R achieves performance comparable to the EDDD. This is because the proposed initial relaxation effectively eliminates many infeasible scenarios, thereby significantly reducing the size of the identified set $\mathscr{P}$. We observe that, in the absence of the new initial relaxation (e.g., IDDD_1 defined in Section 8.3.2), the solver frequently encounters out-of-memory errors when attempting to solve the MIP model $\mathrm{R}(\mathcal{T}, \bar{\mathcal{T}}^*, \mathscr{P})$. While each $\mathrm{R}(\mathcal{T}, \bar{\mathcal{T}}^*, \mathscr{P})$ can be solved to optimality within a reasonable time when the new initial relaxation is used, doing so still requires substantially more computational effort than the sequential batch heuristic. Especially for the HCHF instance group, where the maximum refinement time reaches 811.45 seconds. In contrast, the sequential batch heuristic consistently achieves the final refinement more quickly.

The results in Table EC.3 also show that our sequential batch heuristic achieves a final discretization comparable to that of the exact approach, but with significantly lower computational time. In particular,

although column "%Rini" indicates that EDDD generates more time points for refinement in the first iteration than EDDD_R, columns "%Fsize" and "#Iterations" show that the final discretization sizes and total numbers of iterations for the two approaches remain comparable.

EC.4.4. Effectiveness of Using Relaxation Solution Pool in UB Derivation and Discretization Refinement

Algorithm EDDD derives the upper bounds and refines the discretization based on a pool of relaxation solutions, referred to as the bundle strategy. To further confirm the necessity of the bundle strategy adopted in the newly proposed EDDD for solving the CTSNDP, we conducted algorithm EDDD without the bundle strategy, referred to as EDDD_1. The results on **HC/HF** instances are shown in Table EC.4 using the same notations introduced in Table 1. Table EC.4 shows that the bundle strategy improves EDDD on the hard HC/HF instances, reducing both the average number of iterations and running time while increasing the optimality rate to 100.0%.

Table EC.4 Results for EDDD Variants With/Without the Bundle Strategy

Group	Algorithm	%Gap	Times(s)	#Iterations	%Optimal
HC/HF	EDDD_1	0.85	227.4	3.4	98.3
177	EDDD	0.83	127.2	2.7	100.0

EDDD_1: EDDD without the bundle strategy for relaxation solutions.