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

Shengnan Shu
Department of Logistics and Maritime Studies, the Hong Kong Polytechnic University, 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, the existing DDD algorithms struggle with computational performance, particularly for instances with a high-cost ratio of vehicle-based costs over flow-based costs and high time flexibility, leaving many instances unsolved. This study enhances the DDD solution framework by introducing three new 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-based strategy for computing upper bounds; and (iii) a new discretization refinement strategy based on eliminating too-long paths in a newly defined dispatch-node graph. 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 focused 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 and Less-Than-Truckload (LTL) freight. LTL carriers support numerous supply chains, while parcel and small package carriers are essential for 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 SNDP-involved industries.

Given the markets they serve, these carriers employ freight consolidation operations as a key strategy to enhance profitability. Transportation costs are incurred both on a per-vehicle basis, dependent on distances traveled and per-unit-of-distance costs, and on a flow basis, influenced by the quantity of flow and per-unit-of-flow costs. Consolidation operations enable carriers to consolidate multiple shipments into the same vehicle dispatch, increasing vehicle utilization and reducing vehicle-based transportation costs. To achieve such consolidation, carriers in these industries utilize a network of consolidation terminals where shipments can be transferred from inbound to outbound vehicles. These consolidation terminals allow for the consolidation of multiple shipments onto a single outbound vehicle for more efficient transportation. Each shipment is scheduled to be picked up at its origin terminal at an appropriate time following its availability, routed through the terminal network following a predefined path and consolidation plan, and delivered to its destination terminal within a specified timeframe. Shipments can be temporarily held at both the origin and intermediate terminals along their designated path, allowing for efficient consolidation of shipments.

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 employed in the literature to model the SNDP is *discretization*, which involves dividing the planning horizon into an adequate number of 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 time and space, and arcs represent physical movements between locations or in time at a single location. 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 impacts the computational tractability of the model 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, in each iteration, updates the lower and upper bounds on the optimal solution cost of the problem. 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 challenges in achieving efficient solutions, particularly for instances characterized by a high-cost ratio of vehicle-based costs over flow-based costs and high time flexibility, nearly half of which still need to be solved. The limitations of current DDD algorithms in efficiently solving such instances can be attributed to several factors. The first factor is weak relaxation. A loose relaxation allows numerous impossible consolidations indicated by the available times and due times of shipments. Consequently, this can lead to weak relaxation solutions with many impossible consolidations. These weak relaxation solutions resulting from the relaxation thus pose difficulties in proving 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 by adhering to the routing plan obtained from the relaxation model but repairing the corresponding consolidation plan. However, these heuristic methods do not guarantee the computation of high-quality, feasible solutions, thus impeding the algorithm's convergence. The third factor is the inefficiency of the refinement methods. An effective refinement method should ensure the attainment of a sufficiently fine discretization within a small number of iterations. However, existing refinement methods may fail to reach a reasonably fine discretization or require excessive iterations, adversely affecting the algorithm's overall performance.

1.1. Contributions of this Paper

This paper proposes a new DDD algorithm for the CTSNDP to address the aforementioned challenges. While it shares a similar solution framework with previous DDD algorithms, it distinguishes itself through innovative DDD strategies. The major contributions of our study are summarized as follows.

- The new DDD algorithm features the following innovative strategies:

  —*A new initial relaxation strategy*: We construct a *timed-node-based time-expanded commodity network* for each shipment, enhancing the initial discretization by adding a minimal set of *significant time points*, determined through the solution of *minimum-hitting set problems*. This process effectively eliminates infeasible consolidations from the relaxation model over these networks, yielding a tighter relaxation than the one proposed by Boland et al. (2017). This strategy emphasizes identifying and adding significant time points in advance, improving the algorithm's convergence.

  —*A new upper-bound deriving strategy*: Introducing a new Mixed-Integer Programming (MIP) model, we derive the best upper-bound solution based on all feasible CTSNDP solutions that adhere to the routing

plans provided by a collection of feasible solutions of the relaxation model, significantly enhancing the accuracy and quality of upper-bound solutions. This strategy significantly diverges from the heuristic methods utilized by previous CTSNDP algorithms, aiming at identifying the best CTSNDP solution among all feasible solutions that utilize the routing plans derived by the relaxation model.

—*A more effective discretization refinement strategy*: We present a new discretization refinement strategy which focuses on eliminating newly identified structural patterns (referred to as *minimum too-long paths* in a newly defined *dispatch-node graph*) that cause the infeasibility of the solutions provided by the relaxation model. Based on this, we adopt a three-stage approach capable of eliminating all minimum too-long paths of a collection of feasible solutions of the relaxation model with a few time points. This strategy ensures the algorithm converges to the optimal solution within a few iterations and with a small final network.

- We conduct extensive computational experiments to validate the efficiency and effectiveness of the enhanced DDD algorithm and each proposed strategy. Results demonstrate that the enhanced DDD algorithm exhibited dominant computational performance, being the first to optimally solve all classic CTSNDP instances in the literature.

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, whose details are described in Sections 5-7. Specifically, we introduce the new initial relaxation (§5), illustrate an 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 considering release and due times for the commodities. 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 the arcs, aiming to maximize 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). By highlighting the relationship between SNDP and multicommodity flows over time, we establish the context and draw attention to the relevant literature that forms the basis for addressing the temporal component in the SNDP.

Similar to the approach 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* over 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 time points in the subset 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 dynamic 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 with restricted routes for each commodity, using a network with arc-dependent time discretizations. Specifically, they optimized the time-expanded network by assigning distinct departure time points to different services departing from the same terminal, in contrast to the general time-expanded network, where all services from the same terminal share a common set of departure time points. 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.

The DDD algorithm was initially introduced by Boland et al. (2017) for solving the CTSNDP. However, the concept of dynamically refining discretization had already been applied in previous research within related domains. For the Traveling Salesman Problem with Time Windows (TSPTW), Wang and Regan (2009) introduced a time window discretization method and discussed its convergence properties. They demonstrated that their time window discretization method always converges to the optimal continuous-time solution when the length of partition intervals approaches infinitesimally small values. However, their approach utilized a predetermined uniform interval size for each time window in the formulation and did not propose a dynamic solution algorithm. Dash et al. (2012) introduced a branch-and-cut algorithm for solving the TSPTW by partitioning the time windows into subwindows, referred to as *buckets*. They employed dynamic discretization discovery as a preprocessing step within the branch-and-cut framework to determine a good discretization for these buckets. Once the final discretization was established, it remained unchanged throughout the algorithm. Fischer and Helmberg (2014) designed a dynamic framework that solves a sequence of shortest path problems on valid subnetworks with a given finite time horizon to address the original shortest path problem with infinite time horizons. In each iteration, the dynamic framework recognizes whether the current network excludes the shortest paths that exceed the current time horizon and then extends the current valid subnetwork accordingly. This approach is specific to the shortest path problem, and the methods for generating and refining the networks differ from the DDD approach.

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 modeling method for the problem. 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 following a single delivery path or route. Multiple demands can be consolidated to pass through arc (i, j), reducing the total fixed cost for using the service on arc (i, j). For all practical purposes, we allow the travel times and time window restrictions to be integers in minutes and assume $\min_{k\in\mathcal{K}}\{e^k\} = 0$. 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.

A feasible solution to the SNDP must first specify each commodity's delivery path. In this paper, we formally represent a path p in a network as a sequence of arcs in the network. However, for simplicity, we will adopt an abuse of notation and use the expression $n \in p$ to indicate that node n is either the head or tail node of some arc in the path p. A *flat path* $p^k = (a^k_1, ..., a^k_{|p^k|})$ is called *k-feasible* for commodity k if it is a simple path from o^k to d^k with $e^k + \tau_{a\in p^k}\tau_a \leq l^k$, where $\tau_{a\in p^k} = \sum_{a\in p^k} \tau_a$ denotes the total travel time of the path. To identify the consolidation plan and the required resources for executing the routing and consolidation plans, a feasible solution to the SNDP must also specify a set of departure times $t^k = (t^k_{a^k_1}, ..., t^k_{a^k_{|p^k|}})$ associated with the arcs of the path p^k. These times must satisfy $t^k_{a^k_1} \geq e^k$, $t^k_{a^k_n} \geq t^k_{a^k_{n-1}} + \tau_{a^k_{n-1}}$ for each $n = 2, \ldots, |p^k|$, and $t^k_{a^k_{|p^k|}} + \tau_{a^k_{|p^k|}} \leq l^k$. We refer to such t^k as *p^k-dispatch times*. Without loss of optimality, all commodities that pass through the same arc with the same departure time can be consolidated. For each arc $a \in \mathcal{A}$ and each departure time $t \in \bigcup_{k\in\mathcal{K}, a\in p^k}\{t^k_a\}$, the consolidation set can thus be expressed as $\kappa(a, t) = \{k \in \mathcal{K} : a \in p^k \text{ and } t^k_a = t\}$. The number of vehicles required to transport each consolidation set $\kappa(a, t)$ is calculated as $\left\lceil \sum_{k\in\kappa(a,t)} q^k/u_a \right\rceil$.

A set of k-feasible flat paths, $\{p^k\}_{k\in\mathcal{K}}$ together with p^k-dispatch times, $\{t^k\}_{k\in\mathcal{K}}$, indicates a feasible solution to the SNDP with a total cost of

$$\sum_{a\in\mathcal{A}} \sum_{t\in\bigcup_{k\in\mathcal{K}, a\in p^k}\{t^k_a\}} f_a \left\lceil \frac{\sum_{k\in\kappa(a,t)} q^k}{u_a} \right\rceil + \sum_{k\in\mathcal{K}} \sum_{a\in p^k} c^k_a q^k,$$

where the two terms represent the fixed and flow costs, respectively. As defined above, the SNDP seeks a feasible solution that minimizes the total cost. Without loss of generality, we assume that there always exists a k-feasible flat path for each $k \in \mathcal{K}$, which is sufficient to ensure a feasible solution to the SNDP.

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 $\mathcal{D}^\Delta_\mathcal{T}$ includes two subsets of *timed-arcs*. (i) *Holding arcs* $\mathcal{H}^\Delta_\mathcal{T}$: 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 $\mathcal{D}_\mathcal{T}^\Delta$ with 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 $t'=\arg\max\{s\in\mathcal{T}_{d^k}:s\le d^k\}$ 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 result in large, potentially intractable integer programs, whereas coarse discretizations are more computationally amenable 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. $\mathcal{D}_\mathcal{T}^{\hat{\Delta}}$ is a *fully time-expanded network*

such that $z(\mathcal{D}_{\mathcal{T}}^{\hat{\Delta}})$ provides the optimal solution cost of the CTSNDP, and SND$(\mathcal{D}_{\mathcal{T}}^{\hat{\Delta}})$ becomes a *complete time-indexed model* for the CTSNDP. The corresponding discretization is the *complete discretization*. We thus define the CTSNDP as SND$(\mathcal{D}_{\mathcal{T}}^{\hat{\Delta}})$. As shown in Boland et al. (2017), this fully time-expanded network $\mathcal{D}_{\mathcal{T}}^{\hat{\Delta}}$ 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 and New Strategies

This section provides an overview of our newly developed enhanced DDD algorithm for the CTSNDP, denoted as EDDD, which follows the DDD framework proposed by Boland et al. (2017), but differs in key algorithm components, as summarized in Section 4.3.

4.1. DDD Algorithm in the Literature

With the fully time-expanded network $\mathcal{D}_{\mathcal{T}}^{\hat{\Delta}}$, the optimal solution for the CTSNDP can be obtained by solving SND$(\mathcal{D}_{\mathcal{T}}^{\hat{\Delta}})$. The size of the fully time-expanded network $\mathcal{D}_{\mathcal{T}}^{\hat{\Delta}}$ can be prohibitively large for practical instances, and solving the resulting SND$(\mathcal{D}_{\mathcal{T}}^{\hat{\Delta}})$ becomes challenging. Therefore, Boland et al. (2017) proposed a DDD algorithm, which iteratively expands the size of the time-expanded network.

The DDD algorithm proposed by Boland et al. (2017) 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 $\mathcal{N}_{\mathcal{T}}^{\hat{\Delta}}$, $\underline{\mathcal{H}_{\mathcal{T}}}$ 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 Property 2, we observe that the lengths of arcs in $\underline{\mathcal{A}_{\mathcal{T}}}$ can be shorter than their true travel times; these arcs are denoted 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 $\left|\underline{\mathcal{N}_{\mathcal{T}}}\right| << \left|\mathcal{N}_{\mathcal{T}}^{\hat{\Delta}}\right|$, 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, introduced by Boland et al. (2017), 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. Simultaneously, it solves a linear program to minimize the total dispatch time difference between any two commodities in any consolidation indicated by the optimal relaxation solution. This process yields a feasible

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 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 the optimal objective value of SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$);
5 (Upper Bound Computation) (i) For each relaxation solution obtained, solve a new MIP model 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 as a valid upper bound $\hat{z}$ (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) Apply a new three-stage approach to refine the discretization $\mathcal{T}$ and update the networks in $\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$, so that all the obtained relaxation solutions that utilize infeasible routing and consolidation plans become infeasible to SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$) under the updated $\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$ (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**

solution whose total cost is an upper bound on the optimal solution cost of the CTSNDP. The algorithm, therefore, dynamically updates a lower bound (LB) and upper bound (UB). If the obtained upper bound falls within a pre-determined optimality tolerance, indicating the finding of an optimal solution (within the specified tolerance) for the CTSNDP, the DDD algorithm terminates. In cases where this condition is not met, an MIP is formulated and solved to identify short-arcs in $\underline{\mathcal{A}_{\mathcal{T}}}$ that need adjustment to their true travel times. Subsequently, 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

Algorithm 1 outlines the main steps of the EDDD algorithm.

The EDDD algorithm begins by enhancing the initial discretization $\mathcal{T}$ through the incorporation of a minimum set of significant time points, which are then utilized to construct the corresponding timed-node-based time-expanded commodity networks in $\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$ and the initial relaxation model SND($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$) (line 2). The time-expanded commodity network was initially proposed by Marshall et al. (2021) as an interval-based network. We extend it here to a timed-node-based network enriched with significant time points. These significant time points are introduced to tighten the lower bound derived from the relaxation model obtained, thereby expediting the algorithm's convergence.

Subsequently, the EDDD algorithm solves the relaxation model SND($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$) based on the time-expanded commodity networks $\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$ iteratively to compute a collection of relaxation solutions in each iteration, which includes the optimal relaxation solution with its objective value serving as a valid lower bound for the CTSNDP (line 4). For each obtained relaxation solution, an upper bound for the CTSNDP is then derived by solving a new MIP model. This MIP model aims to achieve a feasible CTSNDP solution, aligning with the routing plan outlined by the relaxation solution while minimizing the total cost (line 5). The algorithm accordingly updates the lower and upper bounds (LB and UB).

If the obtained upper bound fails to meet a predetermined optimality tolerance, a new three-stage refinement based on the obtained relaxation solutions is employed to identify new time points that need to be incorporated into the discretization $\mathcal{T}$, consequently updating the corresponding time-expanded commodity networks $\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$, so that all the obtained relaxation solutions that utilize infeasible routing and consolidation plans become infeasible to SND($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$) under the updated $\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$ (line 10). Otherwise, the EDDD algorithm terminates, finding an optimal (within the imposed tolerance) solution to the CTSNDP.

As we will explain later, our upper bound computation and discretization refinement ensure that if the optimal relaxation solution can be transformed to a feasible CTSNDP solution based on the same routing plan without changing its cost value, then both LB and UB will be equal to the optimal cost value of the CTSNDP, and otherwise, the optimal relaxation solution will not be feasible to the relaxation model after the refinement of discretization with some additional time points. This ensures the convergence of our EDDD algorithm towards an optimal CTSNDP solution in a finite number of iterations. Our EDDD algorithm utilizes a collection of relaxation solutions including the optimal relaxation solution for upper-bound computation and discretization refinement to expedite its convergence further. The relaxation solutions can be obtained by applying a general optimization solver (like Gurobi) on the relaxation model SND($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$).

4.3. Summary of New Strategies: EDDD vs. DDD

The proposed EDDD algorithm differs significantly from Boland et al. (2017)'s DDD and its extension, the DDDI algorithm Marshall et al. (2021), by employing innovative strategies for initial relaxation, upper-bound derivation and discretization refinement:

- It uses a new initial relaxation, SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$), based on timed-node-based time-expanded commodity networks for shipments, in contrast to the timed-node-based and interval-based time-expanded networks employed by Boland et al. (2017) and Marshall et al. (2021), respectively, while integrating 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 relaxation than that used by Boland et al. (2017), and are efficiently computed by solving a series of minimum hitting set problems for intervals (see §5).
- It employs a novel MIP-based approach to compute an upper bound at each iteration, leveraging a collection of relaxation solutions. Unlike the heuristic methods used by Boland et al. (2017) and Marshall et al. (2021), the EDDD algorithm utilizes an MIP model to identify the best CTSNDP solution among all feasible solutions that adhere to the routing plans provided by the relaxation solutions (see §6).
- It refines the discretization by eliminating newly identified structural patterns, referred to as *minimum too-long paths* in a newly defined *dispatch-node graph*, suggesting impractical routing and consolidation plans in the relaxation solutions. To achieve this, it adopts a novel three-stage approach that identifies additional time points to prevent a group of impractical routing and consolidation plans from emerging in the relaxation model, thereby accelerating the convergence of the EDDD algorithm (see §7).

Below, we provide a detailed description of the key improvements introduced in our EDDD algorithm.

5. Initial Relaxation Based on 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 reported by Marshall et al. (2021) and Shu et al. (2024) demonstrated that strengthening the relaxation can significantly accelerate the convergence of the DDD algorithm.

In this section, we propose a new strategy for the initial relaxation model based on timed-node-based time-expanded commodity networks and specific significant time points to strengthen the lower bound further 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($\underline{\mathcal{D}_{\mathcal{T}}}$) by reducing decision variables. 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). These significant time points are identified by solving minimum-hitting set problems.

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

$\underline{\mathcal{D}_{\mathcal{T}}^k} = (\underline{\mathcal{N}_{\mathcal{T}}}, \underline{\mathcal{A}_{\mathcal{T}}^k} \cup \underline{\mathcal{H}_{\mathcal{T}}})$ for every commodity $k \in \mathcal{K}$, where each service arc set $\underline{\mathcal{A}_{\mathcal{T}}^k}$ 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 $\underline{\mathcal{D}_{\mathcal{T}}^k}$, 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 $\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}} = \{\underline{\mathcal{D}_{\mathcal{T}}^k}\}_{k\in\mathcal{K}}$ denote the collection of the timed-node-based time-expanded commodity networks $\underline{\mathcal{D}_{\mathcal{T}}^k}$ for $k \in \mathcal{K}$. Define $\underline{\mathcal{A}_{\mathcal{T}}^{\mathcal{K}}} = \bigcup_{k\in\mathcal{K}} \underline{\mathcal{A}_{\mathcal{T}}^k}$. We can replace $\underline{\mathcal{A}_{\mathcal{T}}}$ with $\underline{\mathcal{A}_{\mathcal{T}}^{\mathcal{K}}}$ 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($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$).

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

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

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

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

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

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 k-feasible delivery path for each commodity $k \in \mathcal{K}$.

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

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

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.

5.2. Strengthening the Relaxation through Significant Time Points and the Minimum Hitting Set

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 $\mathcal{K}_{i,j} = \{k \in \mathcal{K} : d^k \neq i, o^k \neq j, \text{ and } e^k + \phi^k_{o^k,i} + \tau_{i,j} + \phi^k_{j,d^k} \leq l^k\}$ 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 $e^{k_1} + \phi^{k_1}_{o^{k_1},i} + \tau_{i,j} + \phi^{k_2}_{j,d^{k_2}} > l^{k_2}$, 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 $\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$ 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($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$) could still contain impossible consolidations, i.e., $x^{k_1 t\bar{t}}_{i,j} = x^{k_2 t\bar{t}}_{i,j} = 1$ for some $((i,t),(j,\bar{t})) \in \underline{\mathcal{A}_{\mathcal{T}}}$, $k_1, k_2 \in \mathcal{K}_{i,j}$, but $e^{k_1} + \phi^{k_1}_{o^{k_1},i} + \tau_{i,j} + \phi^{k_2}_{j,d^{k_2}} > l^{k_2}$. 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 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($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$).

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.*

Section EC.2 of the e-companion to this paper illustrates the effectiveness of Theorem 1 with an example. Below, we describe how a minimal set of significant time points can be computed efficiently to eliminate all impossible consolidations.

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 straightforward approach may be overly aggressive, potentially leading to excessive significant time points in the resulting discretization. To tackle this issue, we propose minimizing the significant time points required to eliminate impossible consolidations of two commodities 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 $m \in \{1, ..., |W_i|\}$. Assume that these time intervals $(a^i_m, b^i_m]$ are sorted in a non-decreasing order of a^i_m, with ties broken by smaller b^i_m. 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* and can be solved efficiently in polynomial time by 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 $\mathcal{T}_i$, refining the initial discretization $\mathcal{T}$ and eliminating all impossible consolidations of two commodities from 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-time 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 to this paper illustrates in detail how the initialization is performed.

6. An MIP-Based Strategy for Computing an Upper Bound

The optimal solution of the relaxation model SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$) may not correspond to a feasible solution of the CTSNDP with the same cost. Since travel times are underestimated in SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$), a relaxation solution, though corresponding to a collection of k-feasible paths p^k for all commodities k as defined by inequalities (13), could still involve an infeasible consolidation that cannot be achieved by any set of p^k-dispatch times. 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($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$). For each relaxation

solution, we compute a feasible CTSNDP solution by solving a *consolidation planning problem*, which aims to optimize the adjustment of consolidations for the delivery paths p^k 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($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$). 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 $\underline{\mathcal{A}^k_{\mathcal{T}}}$ 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 associated with the timed path $\mathscr{Q}^k(\boldsymbol{x}, \boldsymbol{y})$, with $\mathcal{N}^k(\boldsymbol{x}, \boldsymbol{y}) = \{\nu^k_1 = o^k, ..., \nu^k_{|\mathcal{N}^k(\boldsymbol{x},\boldsymbol{y})|} = d^k\} \subseteq \mathcal{N}$ and $\mathcal{A}^k(\boldsymbol{x}, \boldsymbol{y}) = \{a^k_1, ..., a^k_{|\mathcal{A}^k(\boldsymbol{x},\boldsymbol{y})|}\} \subseteq \mathcal{A}$ 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 $\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$ if there exists a feasible solution $(\boldsymbol{x}, \boldsymbol{y})$ of the relaxation model SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$) 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 $((i, j), \kappa) \in \mathcal{C}, k \in \kappa, k' \in \kappa$. The first two conditions ensure p^k-dispatch times for each $k \in \mathcal{K}$ 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 the valid inequalities (13) included in the relaxation model SND($\underline{\mathcal{D}^{\mathcal{K}}_{\mathcal{T}}}$), each flat path $p^k(\boldsymbol{x}, \boldsymbol{y})$ must be k-feasible for $k \in \mathcal{K}$, 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:

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

For the flat solution $(\mathcal{P}(\boldsymbol{x},\boldsymbol{y}),\mathcal{C}(\boldsymbol{x},\boldsymbol{y}))$ of a relaxation solution $(\boldsymbol{x},\boldsymbol{y})$, whether it is implementable or not, we can obtain the best feasible CTSNDP solution adhering to the given routing plan $\mathcal{P}(\boldsymbol{x},\boldsymbol{y})$ by solving the CPP defined above to optimality. Our EDDD algorithm solves the CPP for each relaxation solution, including the best one. Among all the feasible CTSNDP solutions obtained, the one with the lowest objective value is selected to update the upper bound (UB).

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 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 an 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^k_{ijr} 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^k_i 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($\boldsymbol{x},\boldsymbol{y}$):

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

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

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

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

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

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

$$\delta^k_i \le b_{ijr} + M_1(1-z^k_{ijr}),\ \forall\,(i,j)\in\mathcal{A}(\boldsymbol{x},\boldsymbol{y}), k\in\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y}), r\in\{1,2,...,|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|\}, \tag{21}$$

$$\delta^k_i \ge b_{ijr} - M_1(1-z^k_{ijr}),\ \forall\,(i,j)\in\mathcal{A}(\boldsymbol{x},\boldsymbol{y}), k\in\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y}), r\in\{1,2,...,|\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})|\}, \tag{22}$$

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

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

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

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

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. Here, M_1 denotes a sufficiently large constant. 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, resulting in many redundant solutions that can make the model difficult to solve. To address this challenge, we propose an acceleration strategy, detailed in §EC.3.2 in the e-companion. Our computational study demonstrates that general optimization solvers can effectively solve the resulting MIP model after applying the acceleration strategy.

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. Algorithm EDDD solves model CPPMIP($\boldsymbol{x},\boldsymbol{y}$)

for each obtained relaxation solution, including the optimal relaxation solution, and selects the solution with the lowest cost value among all the feasible solutions obtained to update the best upper bound UB. 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. A Three-Stage Approach for Refining Discretization Based on Too-Long Paths

If the EDDD algorithm termination condition UB=LB is not satisfied, the flat solution of the optimal relaxation solution must be non-implementable. In such cases, it is necessary to refine the discretization $\mathcal{T}$ to "eliminate" this non-implementable flat solution so that it no longer corresponds to any feasible solution to the relaxation model under a refined discretization $\mathcal{T}'$. In Step 10 of our EDDD algorithm, we refine the discretization to "eliminate" each non-implementable flat solution derived from a given collection of relaxation solutions, which includes the optimal relaxation solution. As a result, all the given relaxation solutions become infeasible to the relaxation model under such a refined discretization. Moreover, since only a finite number of flat solutions correspond to all feasible solutions of the CTSNDP, our EDDD algorithm will eventually obtain an optimal relaxation solution with an implementable flat solution after a finite number of iterations and reaches UB=LB.

To achieve the refinement of discretization, we adopt the same refinement operation as proposed by Boland et al. (2017), which involves incorporating new time points to the discretization $\mathcal{T}$. To boost the convergence of our EDDD algorithm, we adopt a new refinement strategy as detailed in Section 7.1. It aims to eliminate some newly identified structural patterns, referred to as minimum too-long paths in a newly defined dispatch-node graph, which cause the infeasibility of flat solutions. Following this refinement strategy, we develop a three-stage approach to refine the discretization based on the minimal too-long paths, as outlined in Section 7.1 and detailed in Sections 7.2–7.4.

7.1. A New Refinement Strategy for Eliminating Minimal Too-Long Paths

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 $\mathcal{P}$ 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 $(\eta_i^k, s_{i,j}^\kappa) \in \mathscr{A}_1$, and is $\tau_{i,j}$ for all $(s_{i,j}^\kappa, \eta_j^k) \in \mathscr{A}_2$. Based on the solution graph, we introduce a *dispatch-node graph* which only contains the dispatch nodes, depicting the arrival time correlations resulting from consolidation relationships among commodity pairs.

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 $a = (i,j) \in \mathscr{A}$.*

For a given flat solution $\mathcal{S}$, the solution graph $\mathscr{D}(\mathcal{S})$ is designed to represent its routing and consolidation plan. In contrast, the dispatch-node graph $\mathscr{G}(\mathcal{S})$ keeps only the relationships between the arrival times of commodities indicated by its routing and consolidation plan. Each arc $(\eta_i^k, \eta_j^{k'})$ in the dispatch-node graph indicates that the arrival time of k' at terminal j must be no earlier than the arrival time of k at terminal i plus the transit time on service (i,j). This arrival time correlation results from consolidating commodities k and k' on arc (i,j) indicated by the given flat solution $\mathcal{S}$ (see §EC.3.3 for an example).

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 established by the following definition.

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

We can establish Theorem 2 below, which reveals the equivalence between the non-implementability of a flat and the existence of too-long paths in the corresponding dispatch-node graph.

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

Consider any non-implementable flat solution $\mathcal{S}$ that is representable in $\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$. Theorem 2 implies that the corresponding dispatch-node graph $\mathscr{G}(\mathcal{S})$ must contain at least one too-long path denoted by P. If we can refine $\mathcal{T}$ to a new discretization $\hat{\mathcal{T}}$ such that P does not exist in the dispatch-node graph of every representable solution in $\underline{\mathcal{D}_{\hat{\mathcal{T}}}^{\mathcal{K}}}$, it ensures that $\mathcal{S}$ is not representable in $\underline{\mathcal{D}_{\hat{\mathcal{T}}}^{\mathcal{K}}}$. This indicates that, to eliminate the non-implementable flat solution $\mathcal{S}$ through the refined discretization $\hat{\mathcal{T}}$, our strategy for discretization refinement can focus on removing the too-long path P using $\hat{\mathcal{T}}$.

Remark 3 *According to the definition of the solution graph, each consolidation node $s^{\kappa}_{i,j}$ with $\kappa = \{k\}$ for some $k \in \mathcal{K}$ is adjacent to a single head end, which is of arc $(\eta^k_i, s^{\kappa}_{i,j})$, and to a single tail end, which is of arc $(s^{\kappa}_{i,j}, \eta^k_j)$. Thus, as demonstrated in Marshall et al. (2021), such nodes $(s^{\kappa}_{i,j})$ can be excluded from the solution graph $\mathscr{D}(\mathcal{S})$, with arcs $(\eta^k_i, s^{\kappa}_{i,j})$ and $(s^{\kappa}_{i,j}, \eta^k_j)$ replaced by (η^k_i, η^k_j). We retain such nodes $(s^{\kappa}_{i,j})$ in $\mathscr{D}(\mathcal{S})$ for the sake of notation simplicity. The similarity and difference between the solution graph and dispatch-node graph, as well as the too-long paths over dispatch-node graphs, are further illustrated in EC.3.3 of the e-companion. It is obvious that for each flat solution $\mathcal{S}$, each path P' in the solution graph $\mathscr{D}(\mathcal{S})$, with the initial node being the origin dispatch node of a commodity in $\mathcal{K}$, and the final node being a dispatch node in $\mathcal{V}_1$, corresponds to a path P in its dispatch-node graph $\mathscr{G}(\mathcal{S})$ that follows the same sequence of dispatch nodes. We emphasize that distinct paths P' in the solution graphs of different flat solutions may correspond to the same too-long paths in their respective dispatch-node graphs. This demonstrates our rationale for eliminating flat solutions based on too-long paths in dispatch-node graphs.*

Consider any too-long path $P \in \mathscr{P}(\mathcal{S})$. P may contain another too-long path as its partial path. In this case, to eliminate P, we only need to eliminate such a too-long partial path. As different too-long paths may share the same too-long partial path, to eliminate too-long paths, we only need to eliminate those that do not contain any other too-long partial paths, which are defined as minimal too-long paths in Definition 3 below (see an example in EC.3.3 of the e-companion for further illustration).

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.*

Our refinement strategy is to eliminate minimal too-long paths obtained from non-implementable flat solutions defined by relaxation solutions. Our refinement consists of the following three stages, the details of which are illustrated in the following:

- *Stage 1*: Obtain a set of minimal too-long paths from non-implementable flat solutions that are defined by relaxation solutions;
- *Stage 2*: Develop an initial refined discretization to eliminate all the obtained minimal too-long paths;
- *Stage 3*: Enhance the initial refined discretization by reducing its size while still ensuring to eliminate all the obtained minimal too-long paths.

Remark 4 *Based on minimal too-long paths in dispatch-node graphs, our refinement strategy has some advantages over the refinement strategy adopted by Marshall et al. (2021). The refinement of discretization in Marshall et al. (2021) relies on "failed paths" and "cycles" in the solution graph. Each failed path that starts from an origin dispatch node $\eta^{k_1}_{o^k_1}$ and ends at a destination dispatch node $\eta^{k_2}_{d^k_2}$ has a total transit time that exceeds $(l^{k_2} - e^{k_1})$. Each cycle revisits the same dispatch node η^k_i exponentially along the cycle route $\tilde{P}$ starting and ending with η^k_i, indicating a path in the solution graph composed of a partial route P from $\eta^k_{o^k}$ to η^k_i and m instances of the cycle route $\tilde{P}$, and ultimately reaching η^k_i with a total transit time that exceeds*

$(l^k - e^k)$. *We emphasize that the too-long path in the dispatch-node graph is not necessarily simple. A cyclic solution graph with "cycles" is associated with a cyclic dispatch-node graph with non-elementary paths. Therefore, their approach is equivalent to refining the discretization based on too-long paths in the dispatch-node graph that follows the same sequence of dispatch nodes to these two types of paths in the solution graph. Given that these too-long paths in the dispatch-node graph may not be minimal, they could contain other too-long partial paths. Thus, eliminating these too-long paths is not as effective as our approach, which eliminates only the minimal too-long paths contained in them, requiring fewer additional time points. Moreover, the refinement of discretization in Marshall et al. (2021) solely relies on the flat solutions defined by the optimal relaxation solution to generate failed paths for elimination. In contrast, we use a collection of relaxation solutions, which includes the optimal relaxation solution, to generate minimal too-long paths. This enables the elimination of more minimal too-long paths and more non-implementable flat solutions during each refinement iteration, accelerating the convergence of our EDDD algorithm.*

7.2. Stage 1: Obtain Minimal Too-Long Paths for Refinement

When the EDDD algorithm does not terminate, the flat solution $(\mathcal{P}(\boldsymbol{x}^*, \boldsymbol{y}^*), \mathcal{C}(\boldsymbol{x}^*, \boldsymbol{y}^*))$ defined by the optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ to the relaxation model SND($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$) must be non-implementable, but it is representable in $\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$. Consider a collection of feasible solutions $(\boldsymbol{x}, \boldsymbol{y})$ to the relaxation model SND($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$), which includes $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ and is denoted by $\mathscr{X}$. This can be obtained by applying a general-purpose optimization solver, such as Gurobi (Gurobi Optimization, LLC 2024), to SND($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$). Let $\mathscr{S}$ indicate a collection of non-implementable flat solutions $(\mathcal{P}(\boldsymbol{x}, \boldsymbol{y}), \mathcal{C}(\boldsymbol{x}, \boldsymbol{y}))$ with $(\boldsymbol{x}, \boldsymbol{y}) \in \mathscr{X}$. Set $\mathscr{S}$ includes $(\mathcal{P}(\boldsymbol{x}^*, \boldsymbol{y}^*), \mathcal{C}(\boldsymbol{x}^*, \boldsymbol{y}^*))$, and each non-implementable flat solution in $\mathscr{S}$ is representable in $\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$.

The first stage of our discretization refinement is obtaining a $\mathscr{P}$ collection of all the minimal too-long paths from the non-implementable flat solutions in $\mathscr{S}$. As previously mentioned, by eliminating these minimal too-long paths, we can eliminate all the non-implementable flat solutions in $\mathscr{S}$ along with other non-implementable flat solutions whose dispatch-node graphs contain any of these minimal too-long paths.

In particular, for each non-implementable flat solution $\mathcal{S} \in \mathscr{S}$, we apply an enumeration algorithm based on label extension to obtain all minimal too-long paths over its dispatch-node graph $\mathscr{G}(\mathcal{S})$. Here, a label represents a path in the dispatch-node graph $\mathscr{G}(\mathcal{S})$ from any origin dispatch node of a certain commodity to any other dispatch node. The algorithm starts with a label set with initial labels for each origin dispatch node $\eta_{o^k}^k$ with $k \in \mathcal{K}$. For each label in the label set, towards each of its successor dispatch nodes in the dispatch-node graph $\mathscr{G}(\mathcal{S})$, it can be extended to a new label. After the extension, the old label is removed from the label set. If a new label represents a too-long path P, which starts from $\eta_{o_1^k}^{k_1}$ and reaches $\eta_i^{k_2}$ with $\rho(P) > l_{k_2} - e^{k_1} - \phi_{i,d^{k_2}}^{k_2}$, we add P to $\mathscr{P}$. Otherwise, we add the new label to the label set. We extend labels until the label set is empty, obtaining all minimal too-long paths to constitute the collection $\mathscr{P}$.

7.3. Stage 2: Develop Initial Refinement Based on Minimum Too-Long Paths

The second stage of our discretization refinement is to develop an initial refinement of discretization so that all the minimal too-long paths in $\mathscr{P}$ are eliminated. First, we establish a sufficient condition in Lemma 1 below for a discretization $\mathcal{T}$ to eliminate any given path P along with all the flat solutions that include P in their dispatch-node graphs.

Lemma 1 *Let $P=(\eta_{i_1}^{k_1},\eta_{i_2}^{k_2},\ldots,\eta_{i_{|P|}}^{k_{|P|}})$ represent a path in the dispatch-node graph $\mathscr{G}(\mathcal{S})$ of a flat solution $\mathcal{S}$. If there does not exist any timed-path in $(\underline{\mathcal{N}_{\mathcal{T}}},\underline{\mathcal{A}_{\mathcal{T}}^{\mathcal{K}}}\cup\underline{\mathcal{H}_{\mathcal{T}}})$ from a timed-node (i_1,t_1) to any timed-node $(i_{|P|},\bar{t}_{|P|})$, with its service arcs denoted by $((i_h,t_h),(i_{h+1},\bar{t}_{h+1}))\in\underline{\mathcal{A}_{\mathcal{T}}^{\mathcal{K}}}$ for $h\in\{1,2,\ldots,|P|-1\}$, such that each $((i_h,t_h),(i_{h+1},\bar{t}_{h+1}))$ is contained in $\underline{\mathcal{A}_{\mathcal{T}}^{k}}$ for every $k\in\{k_h,k_{h+1}\}$, for every flat solution that is representable in $\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$, its dispatch-node graph does not contain P.*

Consider any too-long path $P\in\mathscr{P}$, where $P=(\eta_{i_1}^{k_1},\eta_{i_2}^{k_2},\ldots,\eta_{i_{|P|}}^{k_{|P|}})$ with $\eta_{o^k}^{k}=\eta_{i_1}^{k_1}$ and $\eta_{i}^{k'}=\eta_{i_{|P|}}^{k_{|P|}}$. For each $g\in\{1,2,\ldots,|P|\}$, let $\rho(P,g)$ indicate the total transit time of the partial path from the initial node $\eta_{i_1}^{k_1}$ of P to the g-th dispatch node $\eta_{i_g}^{k_g}$ of P. Thus, $[e^{k_1}+\rho(P,g)]$ indicates the earliest arrival time at the g-th dispatch node $\eta_{i_g}^{k_g}$ of P along path P. To eliminate P, we can refine $\mathcal{T}$ by adding such time points $[e^{k_1}+\rho(P,g)]$ to $\mathcal{T}_{i_g}$ for all $g\in\{1,2,\ldots,|P|-1\}$, because as shown in the proof of Theorem 3, the resulting discretization, denoted by $\hat{\mathcal{T}}$, satisfies the sufficient condition specified in Lemma 1.

Theorem 3 (Too-Long Path Elimination I) *Consider any too-long path $P=(\eta_{i_1}^{k_1},\eta_{i_2}^{k_2},\ldots,\eta_{i_{|P|}}^{k_{|P|}})\in\mathscr{P}$. Refine $\mathcal{T}$ to $\hat{\mathcal{T}}$ with each $\hat{\mathcal{T}}_j=\mathcal{T}_j\cup\{e^{k_1}+\rho(P,g):g\in\{1,2,\ldots,|P|-1\},i_g=j\}$ for $j\in\mathcal{N}$. Then, for every flat solution that is representable in $\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$, its dispatch-node graph does not contain P.*

Remark 5 *Although each $P\in\mathscr{P}$ is a minimal too-long path by definition, Theorem 3 applies to any too-long path P contained in a dispatch-node graph of a non-implementable solution. Thus, Theorem 3 generalizes the result of Marshall et al. (2021), as the latter is established to only eliminate a non-implementable solution based on a specific too-long path P, requiring that P is a simple path and that the final node of P is a destination dispatch node of a commodity. Theorem 3 reveals that with the refined discretization, all the flat solutions with their dispatch-node graphs containing the too-long path P are eliminated.*

Accordingly, for each $P=(\eta_{i_1}^{k_1},\eta_{i_2}^{k_2},\ldots,\eta_{i_{|P|}}^{k_{|P|}})\in\mathscr{P}$, we apply Theorem 3 to obtain a patch of time points, denoted by set $\tilde{\mathcal{T}}_j(P)=\{e^{k_1}+\rho(P,g):g\in\{1,2,\ldots,|P|-1\},i_g=j\}$, for each $j\in\mathcal{N}$. Using these patches to update $\mathcal{T}$, we obtain an initial discretization refinement $\mathcal{T}^{(0)}$ with $\mathcal{T}_j^{(0)}=\mathcal{T}_j\cup[\bigcup_{P\in\mathscr{P}}\tilde{\mathcal{T}}_j(P)]$ for $j\in\mathcal{N}$, which, by Theorem 3, eliminates all the minimal too-long paths $P\in\mathscr{P}$.

7.4. Stage 3: Enhance Refinement by Reducing Time Points

In Stage 2, we obtain an initial refinement $\mathcal{T}^{(0)}$ by using time points in $\tilde{\mathcal{T}}_j(P)$ with $j\in\mathcal{N}$ and $P\in\mathscr{P}$. In Stage 3, we aim to select only a subset set of time points in $\tilde{\mathcal{T}}_j(P)$ for each $j\in\mathcal{N}$ and $P\in\mathscr{P}$, so that using only these selected time points to refine the discretization can still eliminate all the minimal too-long paths

$P \in \mathscr{P}$. Let $\hat{\mathcal{T}}$ indicate a currently refined discretization, which is initially set equal to $\mathcal{T}$. Let $\hat{\mathscr{P}}$ denote a set of minimal too-long paths that have not been eliminated by $\hat{\mathcal{T}}$, which is initially set equal to $\mathscr{P}$.

We examine each pair (j,t) with $j \in \mathcal{N}$ and $t \in \bigcup_{P\in\mathscr{P}} \tilde{\mathcal{T}}_j(P)$ in an increasing order of t (breaking ties arbitrarily), and check whether t should be selected and added to $\hat{\mathcal{T}}_j$. Let $\Lambda(j,t)$ denote a set of minimal too-long paths $\mathcal{P} \in \hat{\mathscr{P}}$, where $P = (\eta_{i_1}^{k_1}, \eta_{i_2}^{k_2}, \ldots, \eta_{i_{|P|}}^{k_{|P|}})$, such that P has an index $g \in \{2,3,\ldots,|P|-1\}$ with $i_g = j$ and $e^{k_1} + \rho(P,g) = t$. For each too-long path $P \in \Lambda(j,t)$, we can check whether P has already been eliminated by the current refined discretization $\hat{\mathcal{T}}$. If it is not, the time point t needs to be selected and added to $\hat{\mathcal{T}}_j$, and otherwise, we can remove P from $\hat{\mathscr{P}}$. Theorem 4 below enables us to check whether P has been already eliminated by the currently refined discretization $\hat{\mathcal{T}}$. Specifically, by definition, P must have an index $g \in \{2,3,\ldots,|P|-1\}$ that satisfies condition (i) in Theorem 4, and condition (ii) in Theorem 4 essentially implies that the sufficient condition in Lemma 1 is satisfied, so that P has been eliminated.

Theorem 4 (Too-Long Path Elimination II) *Given a discretization $\hat{\mathcal{T}}$, consider any too-long path $P \in \mathscr{P}$, where $P = (\eta_{i_1}^{k_1}, \eta_{i_2}^{k_2}, \ldots, \eta_{i_{|P|}}^{k_{|P|}})$. There exists no flat solution that is representable in $\underline{\mathcal{D}_{\hat{\mathcal{T}}}^{\mathcal{K}}}$ with its dispatch-node graph containing P if there exists an index $g \in \{2,3,...,|P|-1\}$ such that the conditions below are both satisfied:*

(i) $[e^{k_1} + \rho(P,h)]$ is contained in $\hat{\mathcal{T}}_{i_h}$ for all $h \in \{1,2,...,g-1\}$, and

(ii) there does not exist any timed-path in $(\underline{\mathcal{N}_{\hat{\mathcal{T}}}}, \underline{\mathcal{A}_{\hat{\mathcal{T}}}^{\mathcal{K}}} \cup \underline{\mathcal{H}_{\hat{\mathcal{T}}}})$ from $(i_{g-1}, e^{k_1} + \rho(P,g-1))$ to any timed-node $(i_{|P|}, \bar{t}_{|P|})$, with its service arcs denoted by $((i_h,t_h),(i_{h+1},\bar{t}_{h+1})) \in \underline{\mathcal{A}_{\hat{\mathcal{T}}}^{\mathcal{K}}}$ for $h \in \{g-1,g,\ldots,|P|-1\}$, such that each $((i_h,t_h),(i_{h+1},\bar{t}_{h+1}))$ is contained in $\underline{\mathcal{A}_{\hat{\mathcal{T}}}^{k}}$ for every $k \in \{k_h, k_{h+1}\}$.

By repeating the above procedure for all pairs (j,t) with $j \in \mathcal{N}$ and $t \in \bigcup_{P\in\mathscr{P}} \tilde{\mathcal{T}}_j(P)$, the final discretization refinement $\hat{\mathcal{T}}$ reduces the size of the initial refinement $\mathcal{T}^{(0)}$. Furthermore, for those minimal too-long paths P removed from $\hat{\mathscr{P}}$, according to Theorem 4, they must be eliminated by $\hat{\mathcal{T}}$. For those P remained in $\hat{\mathscr{P}}$, it can be seen that all the time points in $\tilde{\mathcal{T}}_j(P)$ with $j \in \mathcal{N}$ are added to $\hat{\mathcal{T}}$, and thus, by Theorem 3, such paths P are also eliminated by $\hat{\mathcal{T}}$.

Remark 6 *Theorem 4 essentially generalizes Theorem 3. To see this, we note that Theorem 3 is a special case of Theorem 4 for $g = |P|$. In this case, condition (ii) of Theorem 4 is always satisfied since P is a too-long path. Moreover, condition (i) is easy to examine. In fact, for the P, g, and $\hat{\mathcal{T}}$ considered in the refinement procedure of Stage 3, condition (i) is always satisfied, as $P \in \Lambda(j,t)$ implies that time points $[e^{k_1} + \rho(P,h)]$ are all added to $\hat{\mathcal{T}}_{i_h}$ for all $h \in \{1,...,g-1\}$ in the past iterations. To examine condition (ii) of Theorem 4, noting that $P = (\eta_{i_1}^{k_1}, \eta_{i_2}^{k_2}, \ldots, \eta_{i_{|P|}}^{k_{|P|}})$, we can construct an acyclic network. Its nodes are (h,t_h) for $h \in \{g-1,g,\ldots,|P|\}$ and $t_h \in \hat{\mathcal{T}}_{i_h}$. Its arcs are $((h,t_h),(h+1,\bar{t}_{h+1}))$ for $((i_h,t_h),(i_{h+1},\bar{t}_{h+1})) \in \underline{\mathcal{A}_{\hat{\mathcal{T}}}^{k_h}} \cap \underline{\mathcal{A}_{\hat{\mathcal{T}}}^{k_{h+1}}}$ and $((h,t),(h,t'))$ for each two consecutive time points t and t' in $\hat{\mathcal{T}}_{i_h}$, for $h \in \{g-1,g,\ldots,|P|-1\}$. It can be seen that condition (ii) is satisfied if and only if there exists a path in this acyclic network from $(h, e^{k_1} + \rho(P,g))$ to any node $(|P|, \bar{t}_{|P|})$, which can be efficiently determined.*

8. Computational Study

This section presents an extensive computational analysis to demonstrate the effectiveness and efficiency of the newly proposed EDDD algorithm and better understand the factors contributing 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 about the effectiveness of the new MIP-based strategy for upper bounds and using a pool of relaxation solutions in the upper-bound deriving and refinement strategies.

8.1. Instances

We evaluated the performance of different DDD algorithms by solving 558 CTSNDP instances originally generated by 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 having *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 further assess the performance of EDDD, we implemented the algorithm proposed by Boland et al. (2017) for the CTSNDP, incorporating variables reduction (see §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). All experiments were conducted on an Intel(R) Core(TM) i7-8700 desktop PC with 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

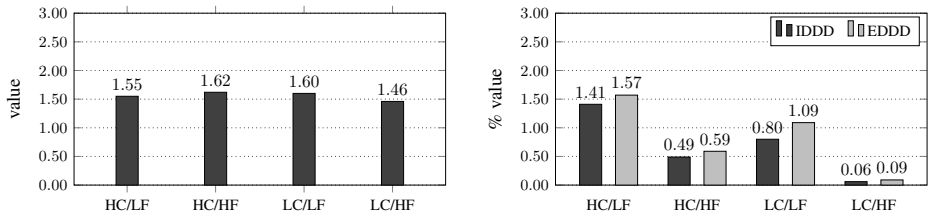
Table 1 Summary Results on the CTSNDP Instances

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	0.73	9.1	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	0.85	131.1	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	0.33	0.2	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

("**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.

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's important to note that the computational configuration for EDDD differs from that for BHMS and MBSH in terms of the machines utilized (a cluster of nodes containing 32 cores each, with speeds ranging from 2.3 to 2.8 GHz for BHMS and MBSH) and the MIP solver adopted (Cplex for BHMS and MBSH). However, comparing the total number of instances solved to proven optimality and the total iterations needed provides a clear global picture of the relative performance, even with differences in hardware and coding configurations. To further ensure a fair comparison between EDDD and MBSH, the best algorithm known so far, we also evaluate their results on identical machines with the same solver settings (see §EC.4.1). Table 1 shows that the algorithm IDDD, our implementation of the algorithm mentioned in Boland et al. (2017), compares well with 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.

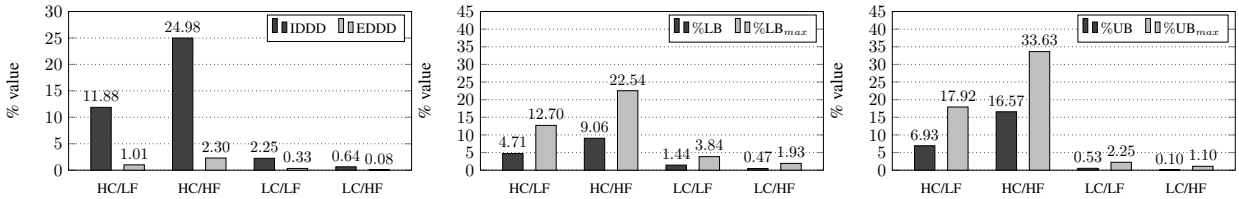
Figure 1 provides insights into the relative sizes of the initial and final discretization generated by EDDD. The algorithm creates a new initial discretization by identifying and adding significant time points based



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

Figure 1 Relative Network Size

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, capable of generating the optimal solution for the CTSNDP, remains a small portion 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, some of which 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 $|\underline{\mathcal{N}_{\mathcal{T}}}|/|\mathcal{N}_{\hat{\mathcal{T}}}^{\Delta=1}| < 4.15\%$.



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

Figure 2 Detailed Comparison of the First Iteration

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.

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 and refinement strategy is discussed in §8.3.1 and §8.3.2, respectively. Additionally, the effectiveness of the new MIP-based approach for computing upper bound is analyzed in §EC.4.2 in the e-companion to this paper. Results show that this new upper-bound deriving method can provide significantly tighter upper-bound values; however, its standalone use has a limited impact on the algorithm's convergence. Moreover, the necessity of considering bundle relaxation solutions in the UB deriving and refinement method is demonstrated in §EC.4.3 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 have demonstrated 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.

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

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 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 the occurrence of cycles in these instances. However, it is important to note that the newly proposed initial relaxation may not be able to prevent the occurrence of cycles in all situations. For instance, if all commodities' due times are unrestricted or unlimited, 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 the original IDDD approach, which adopts 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 $\mathcal{D}^{\hat{\Delta}}_{\mathcal{T}}$ ("**%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	122.0	3.4	98.9	1.53
HC/HF	IDDD	2.34	1377.7	14.8	70.1	0.49
177	IDDD_1	0.91	611.6	3.8	93.8	0.63

IDDD_1: IDDD based on the new refinement strategy.

As indicated by Table 3, the IDDD variant with the new refinement strategy can solve 23% more **HC/HF** instances and 3% 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 the substantial impact the refinement strategy can have on the convergence of DDD algorithms.

9. Conclusions and Future Research

This study proposed theoretical advancements by introducing novel Dynamic Discretization Discovery (DDD) strategies to solve the Continuous-Time Service Network Design Problem (CTSNDP). These included the construction of a timed-node-based time-expanded commodity network for each shipment. The network incorporates a minimal set of significant time points, identified by solving minimum-hitting set problems, thereby eliminating infeasible consolidations in the relaxation model. This approach yielded a tighter relaxation than the one proposed by Boland et al. (2017), laying a stronger theoretical foundation for algorithm convergence. A new Mixed-Integer Programming (MIP) model was also introduced to compute high-quality upper bounds. This approach differs significantly from heuristic methods in earlier CTSNDP algorithms, focusing on deriving optimal solutions based on feasible routing plans from relaxation models. Finally, a novel discretization refinement strategy was implemented, targeting a bundle of relaxation solutions and structural patterns referred to as minimum too-long paths in a newly defined dispatch-node graph. The approach utilized a three-stage method to eliminate these paths, ensuring rapid convergence to the optimal solution with a sparse final network.

The new theoretical advancements were used to design an enhanced DDD algorithm. 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 can potentially enhance 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.

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.

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.

Dash S, Günlük O, Lodi A, Tramontani A (2012) A time bucket formulation for the traveling salesman problem with time windows. *INFORMS Journal on Computing* 24(1):132–147.

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.

Fischer F, Helmberg C (2014) Dynamic graph generation for the shortest path problem in time expanded networks. *Mathematical Programming* 143(1-2):257–297.

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.

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.

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.

Wang X, Regan AC (2009) On the convergence of a new time window discretization method for the traveling salesman problem with time window constraints. *Computers & Industrial Engineering* 56(1):161–164.

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 exist 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 $\eta_{o^k}^k$ and final node denoted by $\eta_i^{k'}$, there must be a path P'' in $\mathscr{G}(\mathcal{S})$ that starts from node $\eta_{o^k}^k$, along path P' to node $\eta_i^{k'}$ and then along the shortest path from $\eta_i^{k'}$ to $\eta_{d^{k'}}^{k'}$ 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 $e^k + \rho(P'') > e^k + \rho(P') + \phi_{i,d^{k'}}^{k'} > l_{k'}$. 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 $P = (v_1^P, v_2^P \ldots, v_{|P|}^P) \in \mathscr{P}(\mathcal{S})$ with $v_1^P = \eta_{o^{k_1}}^{k_1}$ and $v_{|P|}^P = \eta_{i'}^{k_2}$. Let $t(P,g)$ indicate the total transit time of the partial path from the initial node $\eta_{o^{k_1}}^{k_1}$ of P to the g-th node v_g^P of P along path P, and let $T(P,g) = e^k + t(P,g)$ represent the earliest departure time at node v_g^P along P.

Let $E(v) = \max\{T(P,g) : v = v_g^P, P \in \mathscr{P}(\mathcal{S}), g \in \{1,2,\ldots,|P|\}\}$ for all $v \in \mathscr{V}$. We can show that $\boldsymbol{\lambda}$, defined by $\lambda_v = \{\max_{k' \in \kappa} E(\eta_i^{k'}) : v = \eta_i^k, ((i,j),\kappa) \in \mathcal{C} \text{ and } k \in \kappa\}$ 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 $\eta_{o^k}^k$, $k \in \mathcal{K}$, 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 $\eta_{o^k}^k$ for any $k \in \mathcal{K}$, we have that $v_1^P = \eta_{o^k}^k$ and $T(P,1) = e^k$, implying that $E(\eta_{o^k}^k) = e^k$. It follows that $\lambda_{\eta_{o^k}^k} \geq E(\eta_{o^k}^k) = e^k$. Therefore, we conclude that $\lambda_{\eta_{o^k}^k} \geq e^k$ 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 $\lambda(\eta_i^k) = E(\eta_i^{k^*})$ with $k^* = \arg\max\{E(\eta_i^{k'}) : ((i,j),\kappa) \in \mathcal{C}, k \in \kappa \text{ and } k' \in \kappa\}$, and there must exist an arc $(\eta_i^{k^*}, \eta_j^k) \in \mathscr{A}$ according to the definition of the dispatch-node graph $\mathscr{G}(\mathcal{S}) = (\mathscr{V}, \mathscr{A})$. Let $(P^*, g^*) = \arg\max\{T(P,g) : \eta_i^{k^*} = v_g^P, P \in \mathscr{P}(\mathcal{S}), g \in \{1,2,\ldots,|P|\}\}$ and let $\underline{P}$ represent the partial path of P^* from its origin node to $\eta_i^{k^*}$. There must exist a path $P' \in \mathscr{P}(\mathcal{S})$ such that $\underline{P} \subseteq P'$ and $v_{g^*+1}^{P'} = \eta_j^k$. This implies that $\lambda(\eta_j^k) \geq E(\eta_j^k) \geq T(P', g^*+1) = T(P', g^*) + \tau_{i,j} = E(\eta_i^{k^*}) + \tau_{i,j} = \lambda(\eta_i^k) + \tau_{i,j}$. Therefore, we conclude that $\lambda_{\eta_j^k} \geq \lambda_{\eta_i^k} + \tau_{i,j}$ 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 $\eta_{d^k}^k$ with $v_{\hat{g}}^P = \eta_{d^k}^k$, 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 $E(\eta_{d^k}^k) \leq l^k$. We note that $\lambda_{\eta_{d^k}^k} = E(\eta_{d^k}^k)$ as there dose not exists any arc $(\eta_{d^k}^k, v) \in \mathscr{A}$. This implies that $\lambda_{\eta_{d^k}^k} \leq d^k$ for all $k \in \mathcal{K}$, indicating that inequalities (EC.4) are satisfied.

(iv) Moreover, $\lambda_{\eta_i^{k_1}} = \lambda_{\eta_i^{k_2}} = \max_{k' \in \kappa} E(\eta_i^{k'})$ 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 nonimplementable 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 $\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$ when there exists a feasible solution $(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})$ of SND($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$) 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 $P=(\eta_{i_1}^{k_1},\eta_{i_2}^{k_2},\ldots,\eta_{i_{|P|}}^{k_{|P|}})$ 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) $\hat{x}_{i_h,i_{h+1}}^{kt_h\bar{t}_{h+1}}=1$ for some arc $((i_h,t_h),(i_{h+1},\bar{t}_{h+1}))\in\underline{\mathcal{A}_{\mathcal{T}}^{\mathcal{K}}}$ for all $k\in\{k_h,k_{h+1}\}$ 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($\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$), and the first condition implies that $((i_h,t_h),(i_{h+1},\bar{t}_{h+1}))\in\underline{\mathcal{A}_{\mathcal{T}}^{k}}$ for all $k\in\{k_h,k_{h+1}\}$ and $h=1,...,|P|-1$. By connecting these arcs $((i_h,t_h),(i_{h+1},\bar{t}_{h+1}))\in\underline{\mathcal{A}_{\mathcal{T}}^{\mathcal{K}}}$, $h=1,...,|P|-1$, with appropriate holding arcs, we obtain a path $\hat{P}$ in $(\underline{\mathcal{N}_{\mathcal{T}}},\underline{\mathcal{A}_{\mathcal{T}}^{\mathcal{K}}}\cup\underline{\mathcal{H}_{\mathcal{T}}})$.

We thus conclude that for a flat solution $\mathcal{S}$ that is representable in $\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$, for each path P contained in its dispatch-node graph $\mathscr{G}(\mathcal{S})$, where $P=(\eta_{i_1}^{k_1},\eta_{i_2}^{k_2},\ldots,\eta_{i_{|P|}}^{k_{|P|}})$, there exists an timed-path in $(\underline{\mathcal{N}_{\mathcal{T}}},\underline{\mathcal{A}_{\mathcal{T}}^{\mathcal{K}}}\cup\underline{\mathcal{H}_{\mathcal{T}}})$ from a timed-node (i_1,t_1) to any timed-node $(i_{|P|},\bar{t}_{|P|})$, with its service arcs denoted by $((i_h,t_h),(i_{h+1},\bar{t}_{h+1}))\in\underline{\mathcal{A}_{\mathcal{T}}^{\mathcal{K}}}$ for $h\in\{1,2,\ldots,|P|-1\}$, such that each $((i_h,t_h),(i_{h+1},\bar{t}_{h+1}))$ is contained in $\underline{\mathcal{A}_{\mathcal{T}}^{k}}$ for every $k\in\{k_h,k_{h+1}\}$. If such timed-path does not exist, it is evidence that path P is not contained in the dispatch-node graph of any flat solution that is representable in $\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$, thus proving the lemma. $\square$

EC.1.5. Proof of Theorem 3

Lemma 1 establishes a sufficient condition such that for every flat solution that is representable in $\underline{\mathcal{D}_{\mathcal{T}}^{\mathcal{K}}}$, its dispatch-node graph does not contain a specific too-long path P. 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 2 which gives a lower bound on the dispatch times from terminals when the discretization $\mathcal{T}$ contains some specific time points.

Lemma 2 *Let $\hat{P}$ be a timed-path in $(\underline{\mathcal{N}_{\mathcal{T}}},\underline{\mathcal{A}_{\mathcal{T}}^{\mathcal{K}}}\cup\underline{\mathcal{H}_{\mathcal{T}}})$ 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 $((i_h,t),(i_{h+1},\bar{t}))\in\underline{\mathcal{A}_{\mathcal{T}}^{\mathcal{K}}}$ 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 $(\underline{\mathcal{N}_{\mathcal{T}}},\underline{\mathcal{A}_{\mathcal{T}}^{\mathcal{K}}}\cup\underline{\mathcal{H}_{\mathcal{T}}})$. 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 $h=1,...,n$. $\square$

Based on Lemma 2, 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 $P = (\eta^{k_1}_{i_1}, \eta^{k_2}_{i_2}, \ldots, \eta^{k_{|P|}}_{i_{|P|}})$. Refine $\mathcal{T}$ to $\hat{\mathcal{T}}$ with each $\hat{\mathcal{T}}_j = \mathcal{T}_j \cup \{e^{k_1} + \rho(P,g) : g \in \{1,2,\ldots,|P|-1\}, i_g = j\}$ for $j \in \mathcal{N}$. According to Lemma 1, if there does not exist any timed-path P' in $(\underline{\mathcal{N}_{\hat{\mathcal{T}}}}, \underline{\mathcal{A}^{\mathcal{K}}_{\hat{\mathcal{T}}}} \cup \underline{\mathcal{H}_{\hat{\mathcal{T}}}})$ from a timed-node (i_1, t_1) to any timed-node $(i_{|P|}, \bar{t}_{|P|})$, with its service arcs denoted by $((i_h,t_h),(i_{h+1},\bar{t}_{h+1})) \in \underline{\mathcal{A}^{\mathcal{K}}_{\hat{\mathcal{T}}}}$ for $h \in \{1,2,\ldots,|P|-1\}$, such that each $((i_h,t_h),(i_{h+1},\bar{t}_{h+1}))$ is contained in $\underline{\mathcal{A}^{k}_{\hat{\mathcal{T}}}}$ for every $k \in \{k_h, k_{h+1}\}$, it is evidence that path P is not contained in the dispatch-node graph of any flat solution that is representable in $\underline{\mathcal{D}^{\mathcal{K}}_{\hat{\mathcal{T}}}}$.

Assume that there exists such timed-path P' in $(\underline{\mathcal{N}_{\hat{\mathcal{T}}}}, \underline{\mathcal{A}^{\mathcal{K}}_{\hat{\mathcal{T}}}} \cup \underline{\mathcal{H}_{\hat{\mathcal{T}}}})$. As $((i_1,t_1),(i_2,\bar{t}_2))$ must be contained in $\underline{\mathcal{A}^{k_1}_{\hat{\mathcal{T}}}}$ and $e^{k_1} \in \hat{\mathcal{T}}_{i_1}$, based on condition (6)-(ii) on $\underline{\mathcal{A}^{k_1}_{\hat{\mathcal{T}}}}$, we have that $t_1 \geq e^{k_1}$. According to Lemma 2, with $e^{k_1} + \rho(P,g) \in \hat{\mathcal{T}}_{i_g}$, for all $g = 1, ..., |P|-1$, it follows that $t_{|P|-1} \geq e^{k_1} + \rho(P, |P|-1)$. Let $k^* = k_{|P|}$. The existence of timed-path P' implies the presence of an arc $((i_{|P|-1}, t_{|P|-1}),(i_{|P|},\bar{t}_{|P|}))$ in $\underline{\mathcal{A}^{k^*}_{\hat{\mathcal{T}}}}$ with $t_{|P|-1} + \tau_{i_{|P|-1},i_{|P|}} + \phi^{k^*}_{i_{|P|},d^{k^*}} > l^{k^*}$. This violates the condition (6)-(ii) on $\underline{\mathcal{A}^{k^*}_{\hat{\mathcal{T}}}}$, and thus implies that there does not exist such timed-path P'. Thereby, according to Lemma 1, path P is not contained in the dispatch-node graph of any flat solution that is representable in $\underline{\mathcal{D}^{\mathcal{K}}_{\hat{\mathcal{T}}}}$. □

EC.1.6. Proof of Theorem 4

Proof: Given a discretization $\hat{\mathcal{T}}$, consider any too-long path $P \in \mathscr{P}$, where $P = (\eta^{k_1}_{i_1}, \eta^{k_2}_{i_2}, \ldots, \eta^{k_{|P|}}_{i_{|P|}})$. Consider any index $g^* \in \{2,3,...,|P|-1\}$ with both conditions defined in Theorem 4 satisfied. As indicated by Lemma 1, if there does not exist any timed-path P' in $(\underline{\mathcal{N}_{\hat{\mathcal{T}}}}, \underline{\mathcal{A}^{\mathcal{K}}_{\hat{\mathcal{T}}}} \cup \underline{\mathcal{H}_{\hat{\mathcal{T}}}})$ from a timed-node (i_1,t_1) to any timed-node $(i_{|P|},\bar{t}_{|P|})$, with its service arcs denoted by $((i_h,t_h),(i_{h+1},\bar{t}_{h+1})) \in \underline{\mathcal{A}^{\mathcal{K}}_{\hat{\mathcal{T}}}}$ for $h \in \{1,2,\ldots,|P|-1\}$, such that each $((i_h,t_h),(i_{h+1},\bar{t}_{h+1}))$ is contained in $\underline{\mathcal{A}^{k}_{\hat{\mathcal{T}}}}$ for every $k \in \{k_h, k_{h+1}\}$, there exists no flat solution that is representable in $\underline{\mathcal{D}^{\mathcal{K}}_{\hat{\mathcal{T}}}}$ with its dispatch-node graph containing P.

Assume that there exists such timed-path P' in $(\underline{\mathcal{N}_{\hat{\mathcal{T}}}}, \underline{\mathcal{A}^{\mathcal{K}}_{\hat{\mathcal{T}}}} \cup \underline{\mathcal{H}_{\hat{\mathcal{T}}}})$. As $((i_1,t_1),(i_2,\bar{t}_2))$ must be contained in $\underline{\mathcal{A}^{k_1}_{\hat{\mathcal{T}}}}$, and $e^{k_1} \in \hat{\mathcal{T}}_{i_1}$ due to condition (i) defined in Theorem 4, based on condition (i) on $\underline{\mathcal{A}^{k_1}_{\hat{\mathcal{T}}}}$, we have that $t_1 \geq e^{k_1}$. According to Lemma 2, with $[e^{k_1} + \rho(P,h)]$ is contained in $\hat{\mathcal{T}}_{i_h}$ for all $h \in \{1,2,...,g-1\}$, it follows that $t_{g-1} \geq e^{k_1} + \rho(P, g-1)$. The existence of timed-path P' indicates that there exists a timed-path starting from $(i_{g-1}, e^{k_1} + \rho(P,g-1))$, passing through (i_{g-1}, t_{g-1}), and ending at $(i_{|P|},\bar{t}_{|P|})$, with its service arcs denoted by $((i_h,t_h),(i_{h+1},\bar{t}_{h+1})) \in \underline{\mathcal{A}^{\mathcal{K}}_{\hat{\mathcal{T}}}}$ for $h \in \{g-1, g, \ldots, |P|-1\}$, such that each $((i_h,t_h),(i_{h+1},\bar{t}_{h+1}))$ is contained in $\underline{\mathcal{A}^{k}_{\hat{\mathcal{T}}}}$ for every $k \in \{k_h, k_{h+1}\}$. This conflicts with condition (ii) defined in Theorem 4, leading to the conclusion that P' cannot exist. By Lemma 1, we thus can conclude that there is no flat solution that is representable in $\underline{\mathcal{D}^{\mathcal{K}}_{\hat{\mathcal{T}}}}$ with its dispatch-node graph containing P. □

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 $t_3 < t_5 = \arg\max\{s \in \mathcal{T}_i : s \le e^{k_2} + \phi^{k_2}_{o^{k_2},i}\}$, 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), solving model CPPMIP($\boldsymbol{x},\boldsymbol{y}$) (§EC.3.2) and the refinement strategy (§EC.3.4).

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

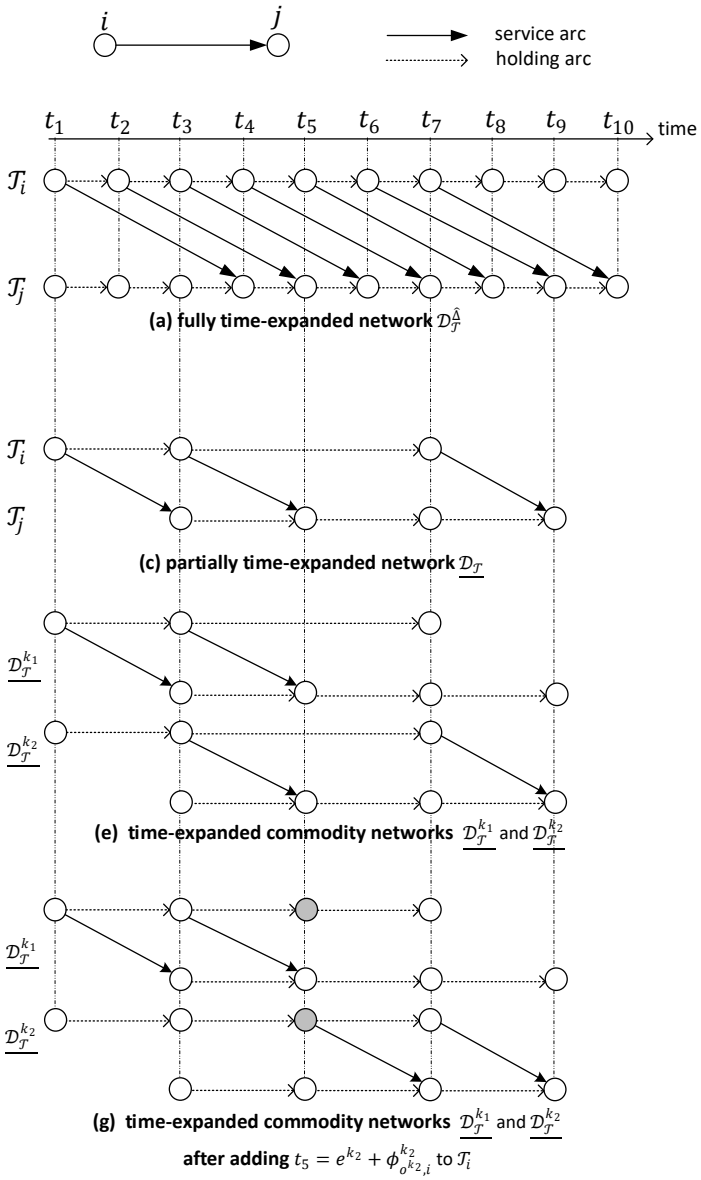
Details on identifying significant time points and constructing the initial time-expanded commodity networks are provided in Algorithm 2.

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 many redundant solutions that can complicate its resolution. This section outlines an acceleration strategy to enhance the model's solvability.

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 propose the following acceleration strategy for each $(i,j) \in \mathcal{A}(\boldsymbol{x},\boldsymbol{y})$.



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

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 $z_{ijr}^{k_h}$ with $r > h$;
   (b) *Step 2*: for each $r \in \{1, ..., |\mathcal{K}_{i,j}(\boldsymbol{x}, \boldsymbol{y})|\}$, remove all variables $z_{ijr}^{k_h}$ 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 $z_{ijh}^{k_h} = 1$, 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

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

required value of 1 or directly removing them. Additionally, Step 4 signifies that certain constraints in the CPPMIP($\boldsymbol{x},\boldsymbol{y}$) model can be reduced as a consequence of variable removal.

2. Add the following inequalities with the reduced $\boldsymbol{z}$ variables:

$$z_{ijh}^{k_{h'}}\le z_{ijh}^{k_h},\quad \forall h\in\{n+1,...,|\mathcal{F}_{i,j}(\boldsymbol{x},\boldsymbol{y})|\},h'\in\{h,...,|\mathcal{F}_{i,j}(\boldsymbol{x},\boldsymbol{y})|\}, \tag{EC.9}$$

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 commodity k_h is included in the h-th consolidation.

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, eliminating most of the 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.9). Figure EC.2-(h) displays all inequalities (EC.9) identified with the reduced $\boldsymbol{z}$ variables.

To minimize the number of $\boldsymbol{z}$ variables in CPPMIP$(\boldsymbol{x},\boldsymbol{y})$, it is recommended to sort $\mathcal{K}_{i,j}(\boldsymbol{x},\boldsymbol{y})$ according to its maximum-size conflict subset for each $(i,j) \in \mathcal{A}(\boldsymbol{x},\boldsymbol{y})$. 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.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.

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

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

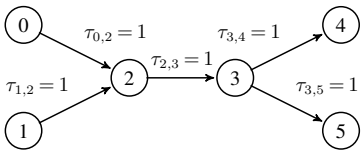
Figure EC.3 illustrates the dispatch-node graph and the too-long paths over it. Figures EC.3-(a) and EC.3-(b) introduce the flat network $\mathcal{D}$ and the commodity delivery requirements. Figures EC.3-(c) and EC.3-(d) present two different flat solutions $\mathcal{S}_1$ and $\mathcal{S}_2$. Their 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. It can be seen that the dispatch-node graph keeps all dispatch nodes and their connectivity relationships in the respective solution graph but is without any 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 between the arrival times of commodities 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 $e^{k_1} + \rho(P_1) = 1 + 3 > l^{k_4} = 3$ and $e^{k_1} + \rho(P_2) = 1 + 2 > l^{k_4} - \phi^{k_4}_{3,4} = 3 - 1 = 2$. 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 eliminating flat solutions based on too-long paths in dispatch-node graphs.

EC.3.4. Details on the Refinement Strategy

Algorithm 3 provides the main steps of the refinement strategy. The algorithm begins by initializing the new discretization $\mathcal{T}'$ with the current one (line 1) and consists of three stages. In the first stage, the algorithm obtain a set of minimal too-long paths from non-implementable flat solutions that are defined by relaxation solutions in the given solution pool $\mathcal{X}$, by applying a label-setting algorithm (lines 2-4). In the second stage, the algorithm develop an initial refined discretization to eliminate all the obtained minimal too-long paths based on Theorem 3 (lines 5-7). In the third stage, the algorithm enhance the initial refined discretization by reducing its size while still ensuring to eliminate all the obtained minimal too-long paths based on Theorem 4 (lines 8-21). A final refined discretization $\mathcal{T}'$ is then returned.

EC.4. Additional Experiments Results

This section provides additional experimental results regarding the comparison of EDDD and MBSH using the same computational environment (§EC.4.1), the bundle strategy used by the refinement strategy (§EC.4.3) and the effectiveness of the new MIP for computing upper bounds (§EC.4.2).



(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

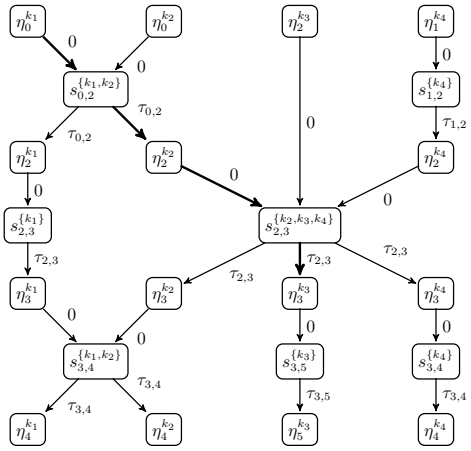
(b) Commodity information

$\mathcal{P}$		$\mathcal{C}$	
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)	$(2,3)$	$\{k_2,k_3,k_4\}$
		$(3,4)$	$\{k_1,k_2\}$
		$(3,4)$	$\{k_4\}$
		$(3,5)$	$\{k_3\}$

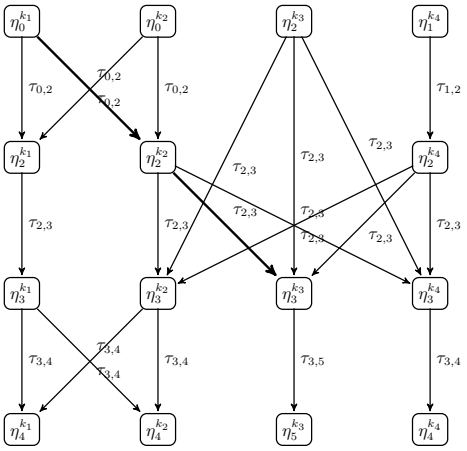
(c) Flat solution $\mathcal{S}_1$

$\mathcal{P}$		$\mathcal{C}$	
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)	$(2,3)$	$\{k_2,k_3\}$
		$(2,3)$	$\{k_4\}$
		$(3,4)$	$\{k_1,k_2\}$
		$(3,4)$	$\{k_4\}$
		$(3,5)$	$\{k_3\}$

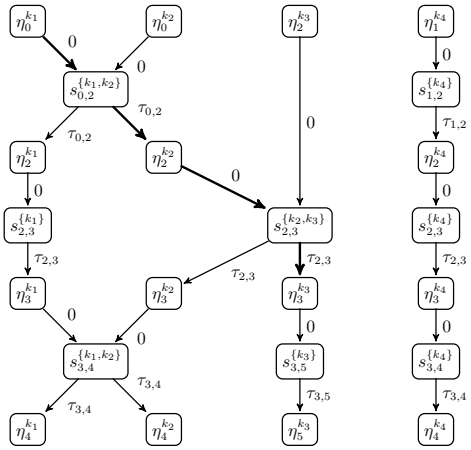
(d) Flat solution $\mathcal{S}_2$



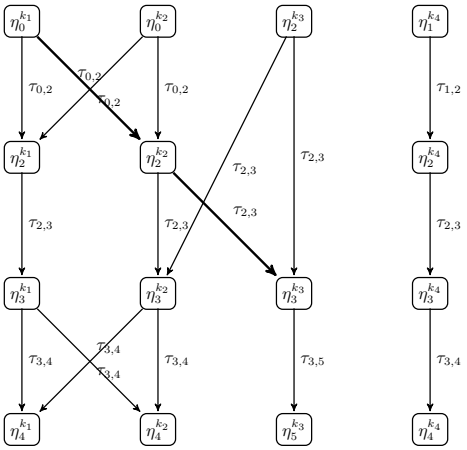
(e) Solution graph $\mathscr{D}(\mathcal{S}_1)$



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

(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

Algorithm 3: Refinement Strategy

Input: Current discretization $\mathcal{T}$, relaxation solution pool $\mathcal{X}$;
Output: Refined discretization $\mathcal{T}'$;
begin
  $\mathcal{T}' \leftarrow \mathcal{T}$ and $\mathscr{P} = \emptyset$;
  // Stage 1: Obtain Minimal Too-Long Paths for Refinement
  for *relaxation solutions* $(\boldsymbol{x}, \boldsymbol{y})$ *in the solution pool* $\mathcal{X}$ **do**
    Apply label-setting algorithm to identify minimal too-long paths and store these paths in $\mathscr{P}$.
  end
  // Stage 2: Develop Initial Refinement Based on Minimum Too-Long Paths
  for *minimal too-long path* P *in* $\mathscr{P}$ **do**
    Identify new time points $\tilde{\mathcal{T}}_j(P)$ for each $j \in \mathcal{N}$ based on Theorem 3;
  end
  // Stage 3: Enhance Refinement by Reducing Time Points
  $\hat{\mathscr{P}} \leftarrow \mathscr{P}$ and $T = \{(j,t) : j \in \mathcal{N}, t \in \bigcup_{P \in \mathscr{P}} \tilde{\mathcal{T}}_j(P)\}$;
  Sort the elements in T in increasing order of time t (breaking ties arbitrarily);
  for $(j,t) \in T$ **do**
    $\Lambda(j,t) = \{P : P \in \hat{\mathscr{P}},$ where $P = (\eta_{i_1}^{k_1}, \eta_{i_2}^{k_2}, \ldots, \eta_{i_{|P|}}^{k_{|P|}})$, such that there exists $g \in \{1,2,\ldots,|P|-1\}$ with $i_g = j$ and $e^{k_1} + \rho(P,g) = t\}$;
    for $P \in \Lambda(j,t)$ **do**
      Let g denote the index $\in \{2,\ldots,|P|-1\}$ of P with $i_g = j$ and $e^{k_1} + \rho(P,g) = t\}$;
      if *both conditions (i) and (ii) in Theorem 4 are satisfied by* P *and* g **then**
        Remove P from $\hat{\mathscr{P}}$;
      end
      else
        $\mathcal{T}'_j = \mathcal{T}'_j \cup \{t\}$;
      end
    end
  end
  return $\mathcal{T}'$;
end

EC.4.1. Comparison of EDDD and MBSH in the Same Computational Environment

To ensure a fair comparison between MBSH, the best-known algorithm for the CTSNDP, and EDDD, we evaluated their results on identical machines with consistent solver settings. A co-author of Marshall et al. (2021) kindly shared their results for MBSH with us, which were obtained by conducting experiments on an AMD EPYC 7V12 processor using the Gurobi solver (v.10.0.3) with a single thread. Accordingly, we conducted our experiments for EDDD on an AMD EPYC 7V12 processor, utilizing the same version of the Gurobi solver with default settings, except for specifying the usage of a single thread. In addition, we adjusted the solver's optimality tolerance setting for EDDD to match the methodology described for MBSH. Specifically, in MBSH, the solver's optimality tolerance is with a value of 0.04 for the first iteration, and $\max(gap \times 0.25, tol \times 0.98)$ for subsequent iterations, where gap is the optimality gap obtained in the previous iteration, and tol is the overall algorithm's optimality tolerance. Similarly, in EDDD, we set the solver's optimality tolerance to a value of 0.04 for the first iteration, 0.02 for the second iteration, and $\max(\min(0.015, gap \times 0.5), tol \times 0.98)$ for subsequent iterations. Both experiments for MBSH and EDDD

were run with a time limit of one hour. The results for EDDD and MBSH with an optimality rate of 1% and 0.001% are shown in Table EC.1 and Table EC.2, respectively. These results highlight the superior performance of EDDD compared to MBSH, which was previously considered the best algorithm for the CTSNDP. The results displayed in Table EC.2 further evidence the effectiveness of the EDDD algorithm even under a tighter optimality tolerance of 0.001%, as it achieves optimality for nearly 90% of the 558 CTSNDP instances with an optimality tolerance of 0.001% and a time limit of one hour.

Table EC.1 Results for EDDD and MBSH with an optimality rate of 1%

Group	Algorithm	%Gap	Times(s)	#Iterations	%Optimal
HC/LF	MBSH	0.82	142.3	11.7	96.7
183	EDDD	0.66	14.8	1.8	100.0
HC/HF	MBSH	1.07	1013.4	17.8	81.4
177	EDDD	0.81	242.3	3.1	100.0
LC/LF	MBSH	0.71	0.2	3.9	100.0
94	EDDD	0.33	0.3	1.3	100.0
LC/HF	MBSH	0.54	0.1	1.5	100.0
104	EDDD	0.08	0.1	1.3	100.0

Table EC.2 Results for EDDD and MBSH with an optimality rate of 0.001%

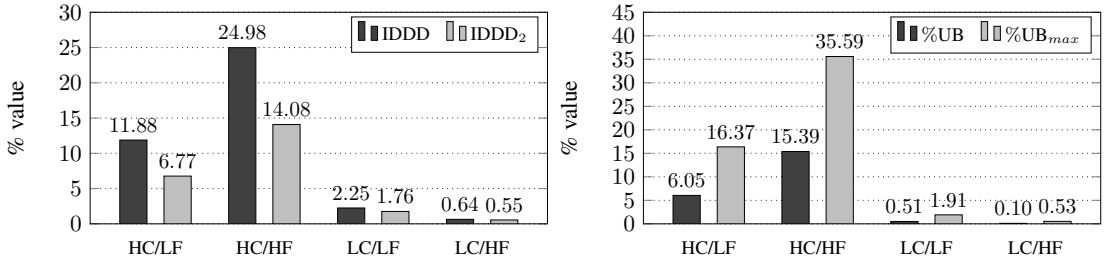
Group	Algorithm	%Gap	Times(s)	#Iterations	%Optimal
HC/LF	MBSH	0.08	555.2	15.9	89.6
183	EDDD	0.02	417.5	6.9	91.3
HC/HF	MBSH	0.56	1555.2	18.5	61.6
177	EDDD	0.12	1071.6	7.7	75.1
LC/LF	MBSH	0.00	0.4	7.4	100.0
94	EDDD	0.00	0.6	3.0	100.0
LC/HF	MBSH	0.00	0.1	4.1	100.0
104	EDDD	0.00	0.1	1.8	100.0

EC.4.2. Effectiveness of the New MIP-based Approach for Upper-Bound Solutions

To assess the effectiveness of the newly proposed method for deriving upper-bound solutions (referred to as the upper-bound deriving method), we conducted a comparison with the method proposed by Boland et al. (2017), which involves solving a corresponding LP and was also utilized by Marshall et al. (2021). To achieve this, we executed algorithm IDDD with the newly proposed upper-bound deriving method, denoted as variant IDDD_2, and compared the results with the original IDDD algorithm.

First, we compare the upper-bound solutions obtained from the newly proposed upper-bound deriving method with those obtained from the upper-bound deriving 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 EC.4-(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 EC.4-(b) highlights the improvement in the upper-bound value achieved by the newly proposed upper-bound deriving 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 EC.4-(b) reveals a notable improvement achieved by the newly proposed upper-bound deriving 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 based on the

Figure EC.4 Effect of the New MIP for Upper-Bound Solutions (IDDD$_2$: algorithm IDDD with the newly proposed upper-bound deriving method).

Second, we assess the effectiveness of the newly proposed upper-bound deriving method on the algorithm's convergence. Table EC.3 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 EC.3 reveals the limited contribution of the newly proposed upper-bound deriving 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 influence of a better upper bound becomes restricted. 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 the poor performance of the relaxation and refinement, a better upper-bound deriving method can save iterations and computational time.

EC.4.3. The Effectiveness of Considering a Pool of relaxation solutions in the UB Deriving Method and Refinement Strategy

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

Table EC.3 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.

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 indicates that the bundle strategy adopted in the newly proposed refinement method is effective for some hard instances. Experiments show that, on average, step 3 of the refinement approach can reduce the number of identified time points by approximately 30% in algorithm EDDD. This also contributes to the small size of the final network, even with the aggressive bundle strategy for refinement.

Table EC.4 Results for EDDD Variants without Bundle Strategy

Group	Algorithm	%Gap	Times(s)	#Iterations	%Optimal
HC/HF	$EDDD_1$	0.86	240.8	3.4	97.7
177	EDDD	0.85	131.1	2.7	100.0

$EDDD_1$: EDDD without the bundle strategy for relaxation solutions.