Arc-Based Dynamic Discretization Discovery for Continuous-Time Service Network Design

Alexander Helber
Chair of Operations Research, RWTH Aachen, helber@or.rwth-aachen.de

January 27, 2026

Abstract: In the *continuous time service network design problem*, a freight carrier decides the path of shipments in their network as well as the dispatch times of the vehicles transporting the shipments. State-of-the-art algorithms for this problem are based on the dynamic discretization discovery framework. These algorithms solve a relaxation of the problem using a sparse discretization of time at each network node and iteratively refine the discretization. We introduce a novel relaxation for this framework, which allows independently choosing the time discretization of each arc, leading to smaller time-expanded networks than previous approaches. We also adapt existing acceleration strategies to the new relaxation. Our computational experiments demonstrate that this arc-based relaxation leads to faster solving times, especially on instances with network topologies with hub-like nodes that many commodities traverse.

Keywords: service network design, dynamic discretization discovery, arc-based

1 Introduction

The service network design problem is an important optimization problem arising in the context of consolidation-based transportation. Companies in this sector need to cost-efficiently organize the transportation of known or estimated volumes of shipments that are generally smaller than the capacity of the used vehicles (Crainic and Hewitt 2021). The companies operate consolidation terminals at which shipments can be consolidated and dispatched together in vehicles towards an intermediary or destination terminal. Consolidation enables more efficient utilization of vehicles and thus more cost-effective transport operations. Since shipments can only be consolidated if they are dispatched from the same terminal at the same time, both the routing of shipments and timing of dispatches are integral to this problem. Overall, the problem consists of deciding which path each shipment should take through the companies network and at what time each shipment should be dispatched from each terminal in its path such that overall costs are minimized.

Commonly, the service network design problem and related problems are modeled using time-expanded networks based on a discretization of time. Time-expanded networks contain nodes representing the terminals of the physical network at concrete times and arcs representing either dispatches between different terminals at a specific time or holding shipments for some time at a terminal. The problem can then be stated as finding paths for each shipment in the time-expanded network with the assumption that shipments on the same dispatch arc in the time-expanded network can be consolidated. The chosen time discretization has a large impact on the quality of obtained solution and computational tractability (Boland et al. 2019). This motivated Boland et al. (2017) to develop a dynamic discretization discovery approach that iteratively refines a very small time-expanded network and solves a corresponding relaxation until a provably optimal solution in continuous time (i.e., with arbitrary time resolution) is found. Their approach, paired with various later improvements (Hewitt 2019; Marshall et al. 2021; Van Dyk and Koenemann 2024; Shu et al. 2025), is capable of solving instances in seconds or minutes for which just constructing an integer program based on the classical time-expanded networks would already take longer and solving the program is not feasible in reasonable time.

One drawback of the previously mentioned approaches is that for each timed copy of a terminal in the time-expanded network an arc for each possible transport to another terminal needs to be created. Thus, strongly interconnected terminals with many potential outbound terminals may result in many timed arcs, especially if shipments are dispatched there at many points in time. But on each individual outbound connection, only a few relevant times would be sufficient to consider. The additional timed copies of arcs result in additional variables and constraints in the integer program, making it potentially harder to solve. This observation motivated Van Dyk and Koenemann (2024) to work towards an arc-based discretization of time, in which the dispatch times of each terminal-to-terminal connection are discretized independently of other connections leaving from the same terminal. They showed that under specific conditions it is possible to modify the network and split some nodes such that some outgoing connections of a terminal can receive individual time discretizations, leading to smaller programs that can be solved faster. Inspired by their work, we make the following contributions in our work:

- We propose a new relaxation that enables individual discretization of time for each connection and does not require the instance to fulfill any specific conditions.
- We show that the relaxation always provides lower bounds to the original problem and that these lower bounds become tight if the relaxation is sufficiently refined.
- We show how to adapt concepts introduced by Shu et al. (2025) to obtain strong initial relaxations and refine our discretization.
- We conduct computational experiments that demonstrate that an algorithm using our relaxation is computationally beneficial on instances with hub-like nodes that are traversed by many commodities. Specifically, while our approach only slightly improves on the state-of-the-art on the commonly used benchmark instances, it can solve instances with specific topologies up to 6 times faster.

The remainder of the paper is organized as follows. In Section 2, we review relevant literature. In Section 3, we present a formal description of the problem that we study. In Section 4, we introduce our new arc-based

relaxation of the problem, show how to refine it, and how our algorithm works. In Section 5, we describe the results of our computational study that shows the efficacy of our algorithm. Finally, in Section 6, we summarize our findings and discuss potential future work.

2 Related Work

The dynamic discretization discovery framework that our work is based on was first introduced by Boland et al. (2017). Their work also contains an overview of prior work that advanced similar concepts but in the context of different problems and utilizing different algorithmic approaches. We focus on algorithmic contributions to solving network design problems with dynamic discretization discovery approaches that have been published since their seminal work and relate them to our work.

Boland et al. (2017) introduce the *continuous time service network design problem* (CTSNDP) and the dynamic discretization discovery framework for solving it. They demonstrated how to construct an integer program on a specific time-expanded network which is a relaxation of the CTSNDP. In this relaxed problem, travel times of commodities are underestimated, allowing consolidations that are not physically possible. Therefore, the solution value of this integer program is a lower bound on the optimal solution value of the CTSNDP. They also propose an algorithm that, given a solution to the integer program, refines the time-expanded network in a way that this lower bound converges towards the optimal solution value. Together with a primal heuristic that repairs the solutions of the integer program, this forms the basis of an exact algorithm for solving the CTSNDP. Hewitt (2019) show that a stronger initial relaxation can be obtained by adding additional time points obtained by solving the linear programming relaxation of the integer program. They also adapt the concept of valid inequalities to the dynamic discretization discovery framework to improve the lower bounds obtained from the integer program and show that some symmetric solutions can be eliminated by a modification of the refinement algorithm proposed by Boland et al. (2017). Marshall et al. (2021) propose a novel relaxation of the CTSNDP utilizing an interval-based interpretation of time discretization. They also introduce a new refinement strategy which not only prevents the same relaxation solution from occurring again but also prevents the same impossible consolidations from occurring again, which can reduce the number of iterations needed for their algorithm to converge. Van Dyk and Koenemann (2024) propose a method by which certain nodes can be split into multiple copies with only a subset of outgoing arcs each. Specifically, if the arcs leaving a node can be partitioned so that each commodity can traverse only arcs in one part of the partition, the node can be split. With this method, not all arcs leaving a node need to receive a timed copy in the time-expanded network for each time point in the discretization for that node. This can lead to smaller relaxation problems and faster solving times. Shu et al. (2025) improve the refinement strategy of Marshall et al. (2021), show how to obtain a much stronger initial relaxation by adding so-called significant time points to the time discretization, and develop a new primal heuristic that can obtain better solutions than the heuristic initially introduced by Boland et al. (2017).

In our work we utilize the same dynamic discretization discovery framework as Boland et al. (2017) but construct a different relaxation than all prior works. Our relaxation is based on an arc-based discretization

of time which is conceptually inspired by Van Dyk and Koenemann (2024) and an interval-based interpretation of this time discretization which is conceptually inspired by Marshall et al. (2021). Despite these inspirations, our relaxation does not require the instance to fulfill the specific condition of the method of Van Dyk and Koenemann (2024) and also results in smaller networks compared to previous approaches. We also adapt the state-of-the-art methods for obtaining a strong initial relaxation and refining the relaxation of Shu et al. (2025) to our relaxation.

3 Problem Description

For notational convenience, we refer to some sets of integer numbers as follows: For $n_1, n_2 \in \mathbb{N}$, we say $[n_1] = \{1, \dots, n_1\}$, $[n_1, n_2] = \{n_1, \dots, n_2\}$ if $n_2 \geq n_1$ and $[n_1, n_2] = \emptyset$ else, and $[n_1, n_2) = [n_1, n_2 - 1]$. We study a generalization of the *(continuous time) service network design problem*, the *(continuous time) service network design problem with restricted routes* (SND-RR) as introduced by Van Dyk and Koenemann (2024). In an instance of this problem we are given a network $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, also called the *flat-network*, and a set of commodities $\mathcal{K}$. The nodes $\mathcal{V}$ represent terminals and the arcs $\mathcal{A}$ connections between terminals along which shipments can be dispatched. For each arc $a \in \mathcal{A}$ we are given a transit time $\tau_a \in \mathbb{N}$, a capacity $u_a > 0$, and a fixed cost $f_a \geq 0$ for vehicles transporting shipments dispatched on this arc. Each commodity $k \in \mathcal{K}$ has a quantity $q_k > 0$ that needs to be transported along a single path within its (sub)network $\mathcal{G}^k = (\mathcal{V}^k, A^k)$ (with $\mathcal{V}^k \subseteq \mathcal{V}, \mathcal{A}^k \subseteq \mathcal{A}$) from its source node $o_k \in \mathcal{V}^k$ to its destination node $d_k \in \mathcal{V}^k$. A commodity k becomes available at its source node at its release time $r_k \in \mathbb{N}$ and needs to arrive at its destination node no later than its deadline $\ell_k \in \mathbb{N}$. Transporting a commodity k along some arc $a \in \mathcal{A}^k$ incurs flow costs of $c_a^k \geq 0$ per unit.

The problem consists of finding a path from source node to destination node for each commodity in its network and deciding the time at which each arc in the path is taken so that the overall costs are minimized and all commodities are transported between their release time and deadline. The overall cost consists of flow costs, which depend on the chosen paths, and fixed costs, which depend on the timing of when the commodities are dispatched on arcs in their path. At any time that one or multiple commodities are dispatched on an arc a, enough capacity (in integer multiples of u_a) needs to be purchased to transport those commodities. To more formally define a solution, we adapt the notation and terminology introduced by Marshall et al. (2021) for the (continuous time) service network design problem to the SND-RR. For every commodity $k \in \mathcal{K}$, we want to find an o_k-d_k-path p^k in its flat-network $\mathcal{G}^k$, which we call a *flat-path*. We state flat-paths as a sequence of arcs, i.e., $p^k = (a_1^k, \dots, a_{n_k}^k)$ where n_k is the number of arcs in the flat-path for commodity k. Note that we assume in this work (without loss of optimality) that all flat-paths in an optimal solution are simple (i.e., repeat no vertices). Additionally, for every commodity we seek a set $t^k = \{t_a^k\}_{a \in p^k}$ that denotes the dispatch times at which the commodity is transported over each arc in its flat-path. A solution $S = (P, T)$ to SND-RR consists of a set of flat-paths $P = \{p^k\}_{k \in \mathcal{K}}$ and a set of dispatch times $T = \{t^k\}_{k \in \mathcal{K}}$ for each commodity. We introduce the following concepts to define a feasible solution:

Definition 1 (k-feasible)**.** *A flat-path p^k is k-feasible if and only if the commodity k can traverse it within*

the time window of its release time to deadline, i.e., $r_k + \sum_{a \in p^k} \tau_a \leq \ell_k$.

Definition 2 (Consistent dispatch times)**.** *Dispatch times* t^k *for a flat-path* p^k *are consistent if and only if* $t^k_{a_1} \geq r_k$, $t^k_{a_{n_k}} + \tau_{a_{n_k}} \leq \ell_k$, *and* $t^k_{a_i} + \tau_{a_i} \leq t^k_{a_{i+1}}$ *for all* $i \in [n_k - 1]$.

A solution is feasible if for all $k \in \mathcal{K}$ the dispatch times t^k are consistent for the flat-path p^k. Note that this also implies that p^k is k-feasible. To compute the cost of a solution, we define for each arc $a \in \mathcal{A}$ the set of time points at which commodities are dispatched on that arc as $\Theta(a) = \bigcup_{k \in \mathcal{K}\colon a \in p^k} \{t^k_a\}$ and for each such time point $t \in \Theta(a)$ the set of commodities that can be consolidated as

$$\kappa(a,t) = \{k \in \mathcal{K} \mid a \in p^k, t^k_a = t\}.$$

Then, the cost of a solution S is computed as

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

where the first term sums the flow costs for each flat-path and the second term the costs for purchasing transport capacity. To solve SND-RR is to find a feasible solution of minimum cost.

A common way of modeling and solving such problems is to use a time-indexed integer programming formulation over a time-expanded network that explicitly models every possible time at which a commodity could leave or arrive at a node. We provide one such formulation as an alternative problem description, identical to that of Van Dyk and Koenemann (2024) except for notation. To not introduce unnecessary variables, we first determine the first and last time point at which a commodity could be at a node in any feasible solution. To compute these time points, we denote by $\tau^k_{uv} \in \mathbb{N} \cup \{\infty\}$ the length (with respect to τ_a) of a shortest path between two nodes $u, v \in \mathcal{V}^k$ in the commodity's flat-network $\mathcal{G}^k$ if such a path exists (and $\tau^k_{uv} = \infty$ otherwise). Then we denote with $\underline{t}_{kv} = r_k + \tau^k_{o_k v}$ the earliest time that the commodity k can arrive at node $v \in \mathcal{V}^k$ and with $\bar{t}_{kv} = \ell_k - \tau^k_{v d_k}$ the latest time it needs to depart that node. Without loss of generality we assume that $\underline{t}_{kv} \leq \bar{t}_{kv}$, since otherwise we could delete node v from the commodity's flat-network without losing any feasible solutions. We now define for each commodity $k \in \mathcal{K}$ its fully time-expanded network $G^k = (V^k, A^k)$ with copies of each node in the flat-network for each time point in the time window for that node, i.e., $V^k = \{(v,t) \mid v \in \mathcal{V}^k, t \in [\underline{t}_{kv}, \bar{t}_{kv}]\}$. The set of arcs $A^k = A^k_{\mathrm{t}} \cup A^k_{\mathrm{h}}$ consists of transport arcs A^k_{t} and holding arcs A^k_{h}. The set of transport arcs contains copies of each arc in the flat-network for each time point if both the dispatch and arrival time are within the time window of the respective node, i.e., $A^k_{\mathrm{t}} = \{((u,t),(v,t+\tau_{(u,v)})) \mid \forall (u,v) \in \mathcal{A}^k, \forall t \in [\underline{t}_{ku}, \bar{t}_{ku}]\colon t + \tau_{(u,v)} \in [\underline{t}_{kv}, \bar{t}_{kv}]\}$. The set of holding arcs contains arcs representing that a commodity waits at it current location, i.e., $A^k_{\mathrm{h}} = \{((v,t),(v,t+1)) \mid \forall v \in \mathcal{V}^k, \forall t \in [\underline{t}_{kv}, \bar{t}_{kv})\}$. For ease of notation, we also define the complete set of transport arcs as $A_{\mathrm{t}} = \bigcup_{k \in \mathcal{K}} A^k_{\mathrm{t}}$. We define parameters for arcs in the time-expanded network analogously to the flat-network as $f_{((u,t),(v,t'))} = f_{(u,v)}$, $u_{((u,t),(v,t'))} = u_{(u,v)}$ for all $((u,t),(v,t')) \in A_{\mathrm{t}}$, and $c^k_{((u,t),(v,t'))} = c^k_{(u,v)}$ for all $k \in \mathcal{K}$ and $((u,t),(v,t')) \in A^k_{\mathrm{t}}$. With this, we state the following integer

program for SND-RR:

$$\min \quad \sum_{k\in\mathcal{K}} \sum_{a\in A_{\mathrm{t}}^k} c_a^k x_a^k + \sum_{a\in A_{\mathrm{t}}} f_a y_a \tag{1a}$$

$$\text{s.t.} \sum_{a\in\delta^+(v)} x_a^k - \sum_{a\in\delta^-(v)} x_a^k = \begin{cases} 1 & \text{if } v = (o_k, r_k) \\ -1 & \text{if } v = (d_k, \ell_k) \\ 0 & \text{else} \end{cases} \quad \forall k \in \mathcal{K}, \forall v \in V^k \tag{1b}$$

$$\sum_{k\in\mathcal{K}:\ a\in A_{\mathrm{t}}^k} q_k x_a^k \le u_a y_a \qquad \forall a \in A_{\mathrm{t}} \tag{1c}$$

$$x_a^k \in \{0,1\} \qquad \forall k \in \mathcal{K}, \forall a \in A^k \tag{1d}$$

$$y_a \in \mathbb{N}_0 \qquad \forall a \in A_{\mathrm{t}} \tag{1e}$$

Variable x_a^k takes value 1 if commodity k takes timed arc $a = ((u,t),(v,t'))$. If this is a transport arc, i.e., if $u \neq v$, this also means that arc (u,v) is selected to be part of the commodity's flat-path p^k and the dispatch time of that arc is $t_{(u,v)}^k = t$. Variable y_a for a timed transport arc $a = ((u,t),(v,t'))$ indicates how many units of capacity need to be purchased on the arc (u,v) to transport the commodities dispatched at time t. Constraint (1b) ensures flow conservation, i.e., that every commodity flows along a path in its time-expanded network. Constraint (1c) ensures that sufficient capacity is purchased to cover all transports that take place. As has been discussed by Boland et al. (2017), directly solving this integer program is often not a promising approach.

4 Dynamic Discretization Discovery

We propose an algorithm to solve SND-RR that fits into the dynamic discretization discovery framework introduce by Boland et al. (2017). As in their framework, we create a relaxation of SND-RR, which can be formulated as an integer program on a time-expanded network. Solving the relaxed problem provides a lower bound on the optimal value of SND-RR. We also use their primal heuristic that tries to obtain a feasible solution to SND-RR with the same value as the solution of the relaxed problem. If this succeeds, we have found an optimal solution. If not, our algorithm determines how to modify the relaxed problem such that the same solution does not appear again. This process is iterated until an optimal solution is found and is guaranteed to converge eventually. The main difference in our approach is that we construct the relaxed problem differently from previous approaches and in a way that results in smaller integer programs that can often be solved faster.

In Section 4.1 we introduce and compare our relaxed problem to those of Boland et al. (2017) and Marshall et al. (2021). We also prove that every solution to SND-RR has a corresponding solution of equivalent or lower cost in the relaxed problem (i.e, that it is indeed a relaxation) and propose an integer program to solve it. Then, in Section 4.2 we demonstrate how to determine if a solution to the relaxed problem can be converted into a feasible solution to SND-RR and if not, how the relaxed problem needs to be adjusted. We also adapt the concept of significant timepoints from Shu et al. (2025) to our relaxed problem to strengthen

the initial relaxation, this is described in Section 4.3. Lastly, we combine all ingredients in Section 4.4 into a dynamic discretization discovery algorithm for solving SND-RR.

4.1 Relaxed Problem

We first introduce the idea of our relaxation conceptually with an example and compare it to previously suggested relaxations and then introduce it more formally. Conceptually, a relaxed version of the problem can be derived by aggregating timed nodes and arcs in the fully time-expanded network in such a way that two properties hold: first, every path in the fully time-expanded network needs to have a corresponding path in the aggregated network with the same (or lower) flow costs. Second, all commodities that traveled over the same timed copy of an arc in the fully time-expanded network also need to travel over the same timed copy of an arc in the aggregated network, i.e., all consolidations that are possible in the actual problem are also possible in the relaxed problem. Then, solving program (1) on such an aggregated network will provide a lower bound on the objective value of SND-RR. With this idea in mind, we give an example for how to construct appropriate aggregated networks that correspond to the relaxations of Boland et al. (2017), Marshall et al. (2021), and our relaxation and compare their features.

4.1.1 Idea: Aggregating Time-Expanded Networks

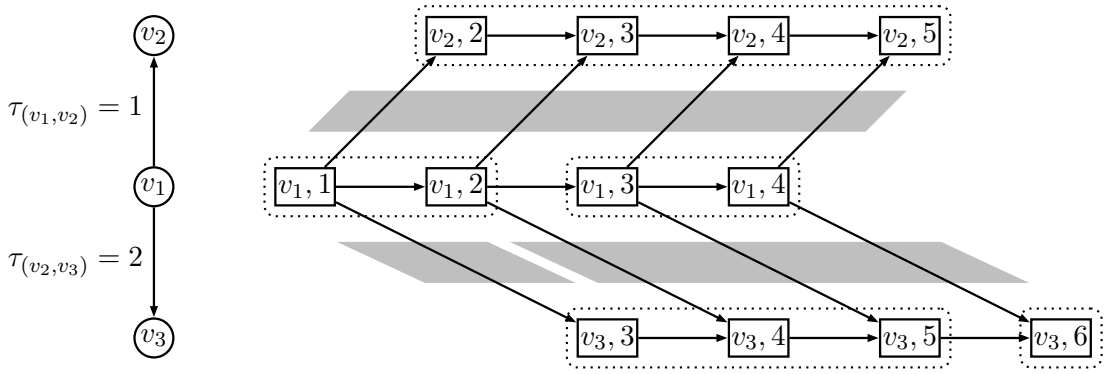
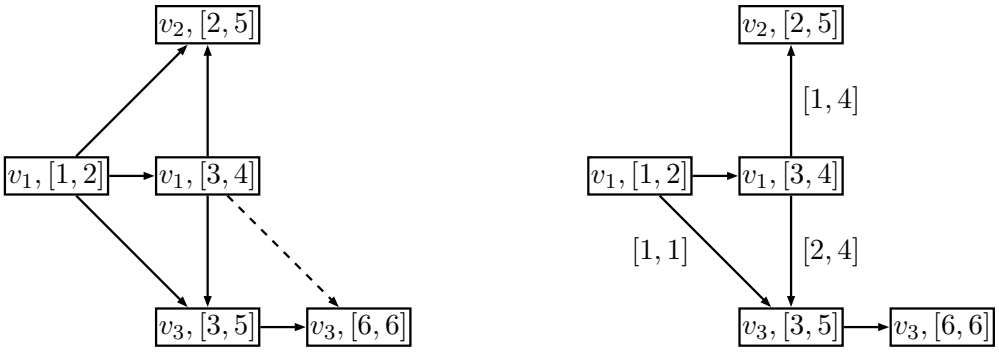


Figure 1: Excerpt of flat-network and fully time-expanded network

Consider Figure 1 which displays (an excerpt of) a flat-network and (an excerpt of) the corresponding fully time-expanded network. We start by demonstrating how to obtain the relaxation of Marshall et al. (2021), since it is perhaps the most natural one. For each node in the flat-network, we partition the time horizon into intervals and merge all timed copies of a node in the same interval. If this creates parallel arcs, these are also merged. We call a specific selection of intervals for all nodes a (node) discretization. In Figure 1, we give an example where the dotted lines represent the intervals, showing only intervals for the relevant part of the time horizon. For example, for node v_1 the time horizon is partitioned into the intervals $[1, 2]$ and $[3, 4]$ while for node v_2 we have a single interval $[2, 5]$. Figure 2a depicts the resulting aggregated network. The four timed copies of node v_1 have been aggregated into the two copies representing the two intervals while a single timed copy represents the interval for node v_2. If a commodity is at such a timed node this represents that it is at the corresponding flat-node at some time point in the interval.



(a) Time-expanded network for node-based relaxation

(b) Time-expanded network for arc-based relaxation

Figure 2: Comparison of node- and arc-based relaxation

Similarly, if a commodity traverses a copy of a flat-arc, this represents the commodity being dispatched on this flat-arc at some time point in the interval of the tail node. Each path in the fully time-expanded network can be represented in this aggregated network: for example, the path (in terms of timed nodes) $((v_1, 1), (v_1, 2), (v_1, 3), (v_2, 4))$ becomes $((v_1, [1, 2]), (v_1, [2, 3]), (v_2, [2, 5]))$. If each aggregated timed arc has the same flow cost value as the timed arcs it replaces, the two paths will have identical flow cost. Additionally, all commodities that can be consolidated in the fully time-expanded network can still be consolidated since they also traverse the same arc in the aggregated network.

As noted in previous works, these aggregated networks are overly optimistic in two regards: First, they allow consolidating commodities that can not be consolidated in the original problem. For example, two commodities dispatched on arc (v_1, v_2) at time 1 and 2 can not actually be consolidated, but in the aggregated network they can, leading potentially to lower costs. Second, travel times are underestimated, allowing commodities to be dispatched earlier than physically feasible. For example, a commodity released at node v_1 at time 2 could still arrive at node v_2 in the interval representing time $[2, 5]$ in the aggregated network. So it could also be dispatched at node v_2 at time 2, even though it could never reach the node at this time. These observations explain why solving the network design problem on such an aggregated network is a relaxation of the original problem.

The relaxation originally proposed by Boland et al. (2017) is constructed almost the same way with the following differences:

- They require that at each commodity's source (sink) node o_k (d_k) an interval has to start at the release time r_k (deadline ℓ_k). This increases the size of the aggregated network substantially.
- Each timed copy has only one timed copy of each outgoing arc, selected to be as early as possible. In our example, they would not create the arc indicated by the dashed line in Figure 2a.
- They interpret the intervals to be single points in time. In relationship to our presentation here, they would refer to an interval $[a, b]$ by the starting time point a.

Some approaches like the one by Marshall et al. (2021) use additional preprocessing rules to remove timed copies of arcs from each commodity's timed subnetwork that provably can not be part of a feasible path.

Except for such preprocessing steps, to the best of our knowledge all previous works use essentially one of these two approaches for constructing the relaxed problem.

Note that in the presented approaches, which timed copies of an arc exist in the aggregated network depend solely on the discretization of the head and tail node of the arc. This may lead to many unnecessary timed copies of some arcs that by chance emanate from the same node as one arc that requires a fine discretization of time to obtain a tight relaxation. In contrast, in this work we propose to explicitly decide on a discretization for each arc independent of the discretization of the nodes, allowing us to employ a fine discretization where necessary and a coarse discretization where possible. Therefore, we also partition the time horizon for each arc into intervals. In Figure 1, the gray areas indicate a possible discretization of time: for arc (v_1, v_2) a single interval $[1, 4]$ and for arc (v_1, v_3) into the intervals $[1, 2]$ and $[3, 4]$. All timed copies of an arc starting in the same interval are replaced by a single copy. If a commodity traverses such an interval arc this represents that it is dispatched along the corresponding flat-arc at some time point in this interval. Additionally, the timed nodes are aggregated as previously described. To ensure that each path in the fully time-expanded network can still be represented in the aggregated network, the interval arc starts from the timed node from which the last replaced timed arc started and ends at the timed node at which the first replaced timed arc ended. The resulting aggregated network for the example is depicted in Figure 2b. The path (in terms of nodes) $((v_1, 1), (v_1, 2), (v_1, 3), (v_2, 4))$ in the fully time-expanded network has a corresponding path $((v_1, [1, 2]), (v_1, [3, 4]), (v_2, [2, 5]))$ in this aggregated network. Even in this toy example we can observe that the arc-based discretization approach can lead to smaller aggregated networks than the node-based approach, as we only need a single timed copy of arc (v_1, v_2) even though the arc (v_1, v_3) leaving from the same node has a finer discretization. We also observe that we only need the information in which interval a commodity traverses an arc to determine which commodities can be consolidated. So contrary to previous approaches, we can give each commodity a different discretization of the nodes, leading potentially to even smaller networks.

Finally, we want to describe the difference between our arc-based approach to that of Van Dyk and Koenemann (2024). If no commodity would have both arc (v_1, v_2) and arc (v_1, v_3) in its subnetwork, they propose to split node v_1 into two copies with only one outgoing arc each. Then, these nodes and their outgoing arcs could receive independent discretizations even if using a node-based approach. While Van Dyk and Koenemann (2024) make convincing arguments that this a partition is possible in many practical applications, our discretization allows to always use an arc-dependent discretization without any requirements on the commodities' subnetworks.

4.1.2 Formal Description of Relaxed Problem

We now formally introduce our time discretization and then use it to define our relaxed problem. Without loss of generality, we assume that $\min_{k \in \mathcal{K}} r_k = 1$ and let $H = \max_{k \in \mathcal{K}} \ell_k$ denote the end of the time horizon, i.e., all transportation occurs inside the time horizon $[H] = \{1, \dots, H\}$. Formally, a time discretization $\mathcal{T} = (\{\mathcal{T}_{kv}\}_{k \in \mathcal{K}, v \in \mathcal{V}^k}, \{\mathcal{T}_a\}_{a \in \mathcal{A}})$ is a tuple of a set containing time intervals for each node in each commodity's network as well as a set of time intervals for each arc in the full network. For each commodity $k \in \mathcal{K}$ and node $v \in \mathcal{V}^k$ the discretization contains a set of $n_{kv} \in \mathbb{N}$ time inter-

vals $\mathcal{T}_{kv} = \{[t_1^{kv}, t_2^{kv}), \ldots, [t_{n_{kv}}^{kv}, t_{n_{kv}+1}^{kv})\}$ that partition $[\underline{t}_{kv}, \bar{t}_{kv}]$, such that $t_1^{kv} = \underline{t}_{kv}$ and $t_{n_{kv}+1}^{kv} = \bar{t}_{kv}+1$. These intervals represent possible times at which commodity k could be dispatched at this node. As an initial time discretization we can simply use the full time horizon, i..e, $\mathcal{T}_{kv} = \{[\underline{t}_{kv}, \bar{t}_{kv}]\}$ for each commodity and node. For the arcs, we partition the time horizon into intervals in which commodities may be dispatched. Specifically, for each arc $a \in \mathcal{A}$ we denote by $\mathcal{T}_a = \{[h_1^a, h_2^a), \ldots, [h_{n_a}^a, h_{n_a+1}^a)\}$ a partition of $[H]$ into n_a intervals, such that $h_1^a = 1$ and $h_{n_a+1}^a = H$. As an initial partition, we can again use the full time horizon, i.e., $\mathcal{T}_a = \{[1, H)\}$ for all $a \in \mathcal{A}$. Note that H does not need to be included in the possible dispatch times since we assume all transit times to be positive and H is the latest time that any commodity needs to arrive at its destination, so no commodity can be dispatched at time point H or later in any feasible solution.

The relaxed problem R-SND-RR($\mathcal{T}$) receives as input an instance to SND-RR as well as a time discretization $\mathcal{T}$. Just as in SND-RR, we want to find an o_k-d_k-path $p^k = (a_1^k, \ldots, a_{n_k}^k)$ in $\mathcal{G}^k$ for every commodity $k \in \mathcal{K}$. Instead of specifying for each arc in a flat-path the exact dispatch time, we specify the time interval in which the commodity is dispatched on that arc. Specifically, the set of arc interval indices $N^k = \{n_i^k\}_{i \in [n_k]}$ states the index of the interval $[h_{n_i^k}^{a_i}, h_{n_i^k+1}^{a_i}) \in \mathcal{T}_{a_i}$ in which a commodity is dispatched on the i-th arc in its flat-path. We assume that commodities dispatched on the same arc in the same interval can be consolidated. A solution to R-SND-RR($\mathcal{T}$) is now given by a tuple $W = (P, N)$ of flat-paths $P = \{p^k\}_{k \in \mathcal{K}}$ and arc intervals $N = \{N^k\}_{k \in \mathcal{K}}$. To describe our notion of a feasible solution to R-SND-RR($\mathcal{T}$), we introduce some additional concepts.

Definition 3 (Feasible arc interval)**.** *For commodity $k \in \mathcal{K}$ and arc $a = (u, v) \in \mathcal{A}^k$, we call an arc interval $[t', t'') \in \mathcal{T}_a$ feasible if and only if there exists a time point $t \in [t', t'')$ such that both $t \in [\underline{t}_{ku}, \bar{t}_{ku}]$ and $t + \tau_a \in [\underline{t}_{kv}, \bar{t}_{kv}]$.*

Said another way, an arc interval is feasible for a commodity if we can dispatch it at some point inside this interval and still arrive on time. Note that the feasibility of an arc interval can be checked in constant time once $[\underline{t}_{kv}, \bar{t}_{kv}]$ has been precomputed for all nodes in each commodity's subnetwork. Note that Definition 3 also ensures that a commodity can not be dispatched from its source node in an interval that lies before its release time or be dispatched on an arc to its destination node in an interval where the earliest arrival time is after its deadline. We require that the interval for each arc in the flat-path for each commodity is feasible.

Next, we relax the concept of consistent dispatch times (Definition 2) which required that a commodity can only be dispatched from a node at or after the time at which it arrives at that node. Recall the example of an aggregated network we constructed in Figure 2b. There, we said a commodity can be dispatched on an arc in an interval $[t', t'']$ from the last interval node whose interval still overlaps the arc interval. So in that example, we say a commodity can be dispatched on arc (v_1, v_3) in interval $[2, 4]$ if it is at node v_1 at interval $[3, 4]$ (or earlier, in which case it can use a holding arc first). More formally, we want the last node interval from which a commodity can dispatch in an arc interval to be the latest node interval that still overlaps the arc interval:

Definition 4 (Relaxed dispatch interval)**.** *Given commodity $k \in \mathcal{K}$, arc $a = (u, v) \in \mathcal{A}^k$, and feasible*

arc interval $[h^a_q, h^a_{q+1})$, we call the node interval index $n^-_{kaq} = \max\{n \in [n_{ku}] \mid t^{kv}_n < h^a_{q+1}\}$ and the corresponding interval interchangeably the relaxed dispatch interval of this arc interval.

Conversely, we say that a commodity dispatched on an arc a in an interval $[t', t'']$ arrives at the interval node whose interval contains the earliest time point at which the commodity could arrive, i.e., $t' + \tau_a$. In the example, a commodity dispatched on arc (v_1, v_3) in interval $[2, 4]$ arrives at node v_3 in the node interval $[3, 5]$ since $2 + 2 = 4 \in [3, 5]$. More formally, we want the first node interval at which a commodity can arrive if it is dispatched on a given arc interval be the earliest node interval that still overlaps the arc interval shifted by the travel time:

Definition 5 (Relaxed arrival interval)**.** *Given commodity $k \in \mathcal{K}$, arc $a = (u, v) \in \mathcal{A}^k$ and feasible arc interval $[h^a_q, h^a_{q+1})$, we call the node interval index $n^+_{kaq} = \min\{n \in n_{kv} \mid h^a_{q+1} + \tau_a \geq t^{kv}_n\}$ and the corresponding interval interchangeably the relaxed arrival interval of this arc interval.*

Note that if the arc intervals are feasible, such a node interval always exists. Based on these relaxed dispatch and arrival intervals we formally state our relaxed notion of every shipment arriving on time at each node along its path for the next dispatch.

Definition 6 (Relaxed time consistent arc intervals)**.** *We call a set of arc intervals N^k for a flat-path p^k relaxed time consistent if the interval for each arc is feasible and for each $i \in [n_k - 1]$ the relaxed arrival interval of arc a_i is no later than the relaxed dispatch interval of arc a_{i+1}, i.e., $n^+_{ka_i n^k_i} \leq n^-_{ka_{i+1} n^k_{i+1}}$.*

We say a solution $W = (P, N)$ to R-SND-RR($\mathcal{T}$) is feasible if for all $k \in \mathcal{K}$ the flat-path p^k is k-feasible and the dispatch time intervals N^k are relaxed time consistent for p^k. The cost of a feasible solution W is the sum of the flow costs for each commodity as well as the fixed costs for dispatched vehicles.

For each arc $a \in \mathcal{A}$ and interval index $q \in [n_a]$ we collect the set of consolidated commodities $\bar{\kappa}(a, q) = \{k \in \mathcal{K} \mid \exists i \in [n_k] \colon a^k_i = a, n^k_i = q\}$. With this, we can state the cost formally as

$$c(W) = \sum_{k \in \mathcal{K}} \sum_{a \in p^k} c^k_a + \sum_{a \in \mathcal{A}} \sum_{q=1}^{n_a} f_a \left\lceil \frac{\sum_{k \in \bar{\kappa}(a,q)} q_k}{u_a} \right\rceil.$$

Note that we can get non-simple flat-paths (i.e., that repeat vertices and even arcs) as part of our relaxation solutions. Such paths can without loss of optimality be assumed to not exist in optimal solutions to SND-RR but in optimal solutions to R-SND-RR($\mathcal{T}$) they can occur since they allow traveling back in time. It would be possible to require feasible solutions to R-SND-RR($\mathcal{T}$) to only contain simple paths, which could give stronger bounds, but preliminary experiments indicated that this made the relaxed problem computationally harder to solve.

We now claim that R-SND-RR($\mathcal{T}$) is indeed a relaxation of SND-RR, i.e., for every solution to SND-RR we can find a corresponding solution of R-SND-RR($\mathcal{T}$) of equal or lower cost.

Proposition 1. *Given an instance to SND-RR and a time discretization $\mathcal{T}$, for every feasible solution $S = (P, T)$ to SND-RR there is a corresponding feasible solution $W(S) = (P(S), N(S))$ to R-SND-RR($\mathcal{T}$) of equal or lower cost.*

Proof. We first show how to construct such a feasible solution $W(S) = (P(S), N(S))$ and then show that $c(W(S)) \leq c(S)$. The relaxed solution will utilize the same flat-paths for each commodity, i.e., $P(S) = P$. To determine the arc intervals, we select the interval in which the time point lies at which a commodity is dispatched in the solution S. More formally, for each commodity k for the i-th arc $a_i = (u, v) \in p^k$, we set n_i^k such that $t_{a_i}^k \in [h_{n_i^k}^{a_i}, h_{n_i^k+1}^{a_i})$. These arc intervals are feasible because S is feasible and therefore $t_{a_i}^k \in [\underline{t}_{kv}, \bar{t}_{kv}]$. We still need to show that they are also relaxed time consistent. Recall that they are relaxed time consistent if for any two consecutive arcs $a_i = (u, v), a_{i+1} = (v, w)$ in p^k it holds that $n^+_{ka_in_i^k} \leq n^-_{ka_{i+1}n_{i+1}^k}$. First, consider the relaxed dispatch interval for arc a_{i+1}, which is the latest node interval in $\mathcal{T}_{kv}$ that overlaps with the arc interval chosen before for a_{i+1}. Since that arc interval contains $t_{a_{i+1}}^k$, this node interval either also contains $t_{a_{i+1}}^k$ or is later than this time point. Now consider the relaxed arrival interval for arc a_i, which is the node interval in $\mathcal{T}_{kv}$ that contains $t_{a_i}^k + \tau_{a_i}$. Since the arc interval contains $t_{a_i}^k$, this node interval either contains $t_{a_i}^k + \tau_{a_i}$ or is earlier than this time point. Finally, since S is a feasible solution we know that $t_{a_i}^k + \tau_{a_i} \leq t_{a_{i+1}}^k$ and therefore the relaxed dispatch interval of a_{i+1} is the same or later than the relaxed arrival interval of a_i and therefore the dispatch intervals are relaxed time consistent and $W(S)$ is feasible for R-SND-RR($\mathcal{T}$).

We still need to show that this solution does indeed give a lower bound on the cost of S. With regard to the flow costs, both solutions are identical since the flow costs are computed the same for both problems and P and $P(S)$ are identical. With regard to the fixed costs, we note that commodities that could previously be consolidated since they were dispatched at the same time can still be consolidated in the relaxed problem since they are now dispatched in the same interval. Additionally, some commodities that can not be consolidated in SND-RR since they are dispatched at different times may be consolidated if their dispatch times lie in the same interval, leading to lower costs. More formally, let us consider the $\bar{\kappa}(a, q)$ set of commodities dispatched on arc a in interval q in solution $W(S)$. The cost contribution of this set in the relaxed problem is $f_a \left\lceil \frac{\sum_{k \in \bar{\kappa}(a,q)} q_k}{u_a} \right\rceil$ while in the original problem it would be $\sum_{t=h_q^a}^{h_{q+1}^a - 1} f_a \left\lceil \frac{\sum_{\kappa(a,t)} q_j}{u_a} \right\rceil$. Since $\bar{\kappa}(q, a) = \bigcup_{t=h_q^a}^{h_{q+1}^a - 1} \kappa(a, t)$, the first term is evidently no larger than the second. Therefore, the fixed costs for the solution $W(S)$ are either identical to or an underestimation of the fixed costs for a solution S. □

We showed that solving R-SND-RR($\mathcal{T}$) for a given time discretization indeed gives us lower bounds on the solution value of SND-RR. We still need to demonstrate that if we add sufficiently many time points to the discretization the lower bound eventually becomes tight, i.e., an optimal solution to SND-RR and R-SND-RR($\mathcal{T}$) have the same value. This can be seen if we construct a full time-discretization, i.e., using intervals of length 1 only.

Proposition 2. *Given an instance to SND-RR and a time discretization $\mathcal{T}$ with $\mathcal{T}_{kv} = \{[\underline{t}_{kv}, \underline{t}_{kv} + 1), [\underline{t}_{kv} + 1, \underline{t}_{kv} + 2), \ldots, [\bar{t}_{kv}, \bar{t}_{kv} + 1)\}$ for all $k \in \mathcal{K}$ and $v \in \mathcal{V}^k$ as well as $\mathcal{T}_a = \{[1, 2), [2, 3), \ldots, [H - 1, H)\}$ for all $a \in \mathcal{A}$, the value $c(S)$ of an optimal solution S to the instance of SND-RR is equal to the value $c(W)$ of an optimal solution W to R-SND-RR($\mathcal{T}$).*

Proof. Since for each time point a unique interval exists on each arc and at each node, the relaxed problem R-SND-RR($\mathcal{T}$) does not underestimate any travel times. Specifically, if a commodity is dispatched on an

arc a in an interval $[t, t+1)$, the relaxed dispatch interval is $[t, t+1)$ and the relaxed arrival interval is $[t+\tau_a, t+\tau_a+1)$, i.e., they are identical to the actual times. So relaxed time consistent arc intervals in the solution to R-SND-RR($\mathcal{T}$) can be mapped directly to consistent dispatch times, and we can derive a feasible solution S to SND-RR from a feasible solution W to R-SND-RR($\mathcal{T}$). These solutions also have identical costs, since only commodities dispatched at exactly the same time on an arc can be consolidated in both cases. Therefore, each solution to R-SND-RR($\mathcal{T}$) corresponds directly to a solution of SND-RR with identical cost and the relaxation is tight. □

4.1.3 Solving the Relaxed Problem

To actually solve R-SND-RR($\mathcal{T}$), we utilize an integer program formulated over a time-expanded network. We will first describe how to construct this time-expanded network and then introduce the integer program. Given a time discretization $\mathcal{T} = (\{\mathcal{T}_{kv}\}_{k\in\mathcal{K}, v\in\mathcal{V}^k}, \{\mathcal{T}_a\}_{a\in\mathcal{A}})$, we can construct what we call a *relaxed time-expanded network* $G^k_{\mathcal{T}} = (V^k_{\mathcal{T}}, A^k_{\mathcal{T}})$ for each commodity k. The set of nodes contains one copy of each node in the commodity's flat-network for each time interval in the discretization, i.e., $V^k_{\mathcal{T}} = \{(v,q) \mid \forall v \in \mathcal{V}^k, \forall q \in [n_{kv}]\}$. The set of arcs $A^k_{\mathcal{T}} = A^k_{\mathcal{T}\mathrm{t}} \cup A^k_{\mathcal{T}\mathrm{h}}$ contains transport and holding arcs. The holding arcs are constructed as usual, linking timed copies of a node, as $A^k_{\mathcal{T}\mathrm{h}} = \{((v,q),(v,q+1)) \mid \forall v \in \mathcal{V}_k, \forall q \in [n_{kv}-1]\}$. Constructing the set of transport arcs is a bit more involved but follows directly from the definition of the relaxed arrival and dispatch intervals. For each arc in the flat-network, we create a copy for each interval in its discretization. The copy leaves from the node representing the relaxed dispatch interval and arrives at the node representing the relaxed arrival interval. Formally, we state the set of transport arcs in the relaxed time-expanded network of a commodity k as

$$A^k_{\mathcal{T}\mathrm{t}} = \{((u, n^-_{kaq}), (v, n^+_{kaq})) \mid \forall a = (u,v) \in \mathcal{A}^k \; \forall q \in [n_a] \colon [h^a_q, h^a_{q+1}) \text{ is feasible for } k\}.$$

Note that this may create parallel arcs that share their head and tail node but represent dispatching in different time intervals. To tell apart parallel arcs, we denote by $q_{\hat{a}} \in [n_a]$ the index of the time interval belonging to arc a that lead to the creation of the timed arc $\hat{a} \in A^k_{\mathcal{T}\mathrm{t}}$. Again we define parameters for interval arcs in the relaxed time-expanded network analogously to the flat-network as $f_{((u,q),(v,q'))} = f_{(u,v)}, u_{((u,q),(v,q'))} = u_{(u,v)}, \tau_{((u,q),(v,q'))} = \tau_{(u,v)}$, and $c^k_{((u,q),(v,q'))} = c^k_{(u,v)}$ for all $k \in \mathcal{K}$ and $((u,q),(v,q')) \in A^k_{\mathcal{T}\mathrm{t}}$. For easier notation, we denote by $\mathcal{K}_{aq}$ the set of commodities for which $a \in \mathcal{A}^k$ and q is a feasible arc interval. For each $k \in \mathcal{K}_{aq}$ we denote by $\hat{a}_{kaq}$ the timed arc corresponding to the q-th interval of arc a in the time-expanded network of k.

An integer program to solve R-SND-RR($\mathcal{T}$) can be stated as follows.

$$\min \quad \sum_{k\in\mathcal{K}} \sum_{\hat{a}\in A^k_{\mathcal{T}_t}} c^k_{\hat{a}} x^k_{\hat{a}} + \sum_{a\in\mathcal{A}} \sum_{q=1}^{n_a} f_a y_{aq} \tag{2a}$$

$$\text{s.t.} \sum_{\hat{a}\in\delta^+(v)} x^k_{\hat{a}} - \sum_{\hat{a}\in\delta^-(v)} x^k_{\hat{a}} = \begin{cases} 1 & \text{if } v = (o_k, 1) \\ -1 & \text{if } v = (d_k, n_{kd_k}) \\ 0 & \text{else} \end{cases} \quad \forall k \in \mathcal{K}, \forall v \in V^k_{\mathcal{T}} \tag{2b}$$

$$\sum_{k\in\mathcal{K}_{aq}} q_k x^k_{\hat{a}_{kaq}} \leq u_a y_{aq} \qquad \forall a \in \mathcal{A}, \forall q \in [n_a] \tag{2c}$$

$$\sum_{\bar{a}\in A^k_{\mathcal{T}}} \tau_{\hat{a}} x^k_{\hat{a}} \leq \ell_k - r_k \qquad \forall k \in \mathcal{K} \tag{2d}$$

$$x^k_{\hat{a}} \in \{0,1\} \qquad \forall k \in \mathcal{K}, \forall \hat{a} \in A^k_{\mathcal{T}} \tag{2e}$$

$$y_{aq} \in \mathbb{N}_0 \qquad \forall a \in \mathcal{A}, \forall q \in [n_a] \tag{2f}$$

A variable $x^k_{\hat{a}}$ takes value 1 if commodity k is transported along timed arc $\hat{a}$. If that arc $\hat{a} = ((u,q),(v,q'))$ is a transport arc ($u \neq v$), it also means that the arc (u,v) should be in the flat-path p^k of the commodity and the corresponding arc interval is $q_{\hat{a}}$. Variable y_{aq} indicates how many vehicles are dispatched on arc a in the time interval $[h^a_q, h^a_{q+1})$. Constraint (2b) ensures that the selected arcs for each commodity form a path in the time-expanded network. Constraint (2c) ensures that enough vehicles are dispatched on each arc in each time interval to transport the commodities traveling over that arc in that time interval. Constraint (2d) ensures that the resulting flat-paths are k-feasible.

4.1.4 Full Example of Relaxed Problem

To illustrate the concepts of our relaxed problem and demonstrate that it can produce equally tight lower bounds on smaller time-expanded networks than previous approaches, we give a fully worked out example. The flat-network is shown in Figure 3. Each commodity's subnetwork contains the nodes and arcs that can be part of a k-feasible o_k-d_k path. The instance has four commodities for which the relevant parameters are given in Table 1a. All cost and demand parameters take on value 1 and all arcs have a capacity of 2. Observe that in this instance, all commodities have only a single flat-path available and the only question is at what time each commodity is dispatched along each arc in its path. The only possible consolidations are to consolidate commodities k_1 and k_2 on arc (v_1, v_2), commodities k_1 and k_3 on arc (v_2, v_3) and commodities k_2 and k_4 on arc (v_2, v_4). The instance is constructed such that we can either consolidate k_1 with k_2 or k_1 with k_3 but not both. Assume that we consolidate k_1 and k_2. Then they can not be dispatched before the release time $r_{k_2} = 2$ and thus will arrive at node 2 at time $r_{k_2} + \tau_{(v_1,v_2)} = 4$. This is after the last time at which commodity k_3 can be dispatched to arrive on time ($\bar{t}_{k_3 v_2} = 3$), so it is not possible to consolidate k_1 and k_3 as well. In either case, k_2 and k_4 can be consolidated (k_4 exists solely to demonstrate the smaller network size due to the arc-based approach).

We now also give one specific time discretization $\mathcal{T}$ as an example and study the resulting time-

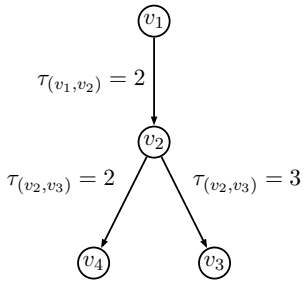


Figure 3: Flat-network $\mathcal{G}$ for the example instance

	o_k	d_k	r_k	ℓ_k
k_1	v_1	v_3	1	8
k_2	v_1	v_4	2	7
k_3	v_2	v_3	3	6
k_4	v_2	v_4	1	8

(a) Commodity parameters

	v_1	v_2	v_3	v_4
k_1	$\{[1,3]\}$	$\{[3,3],[4,5]\}$	$\{[6,8]\}$	-
k_2	$\{[2,3]\}$	$\{[4,5]\}$	-	$\{[6,7]\}$
k_3	-	$\{[3,3]\}$	$\{[6,6]\}$	-
k_4	-	$\{[3,3]\}$	$\{[6,6]\}$	$\{[3,8]\}$

(b) Node intervals $\mathcal{T}_{kv}$

a	$\mathcal{T}_a$
(v_1, v_2)	$\{[1,1],[2,7]\}$
(v_2, v_3)	$\{[1,3],[4,7]\}$
(v_2, v_4)	$\{[1,7]\}$

(c) Arc intervals $\mathcal{T}_a$

Table 1: Parameters for an example instance of R-SND-RR($\mathcal{T}$) on the network in Figure 3

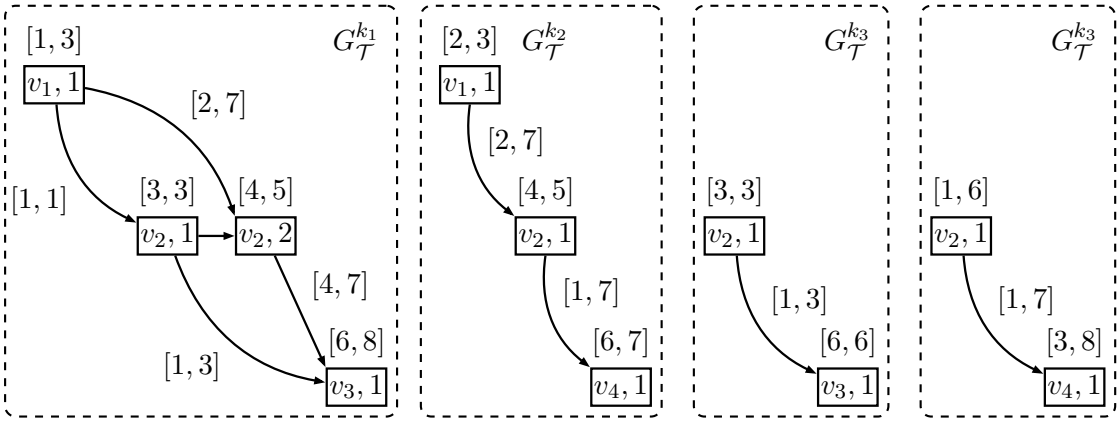


Figure 4: Example of time-expanded networks for each commodity in the relaxed problem

expanded networks of the relaxed problem. Table 1b and Table 1c display the intervals in this time discretization and Figure 4 the resulting relaxed time-expanded network of each commodity. Recall that two commodities can be consolidated on an arc if they traverse it in the same interval. So commodities k_1 and k_2 can only be consolidated if they are transported on arc (v_1, v_2) in interval $[2, 7]$. But if we chose this interval for k_1, it can not traverse the arc (v_2, v_3) in the interval $[1, 3]$ anymore and since this is the only feasible interval for k_3, we can not consolidate k_1 and k_3. So this discretization $\mathcal{T}$ already captures the trade-off to be made in the example instance and solving R-SND-RR($\mathcal{T}$) will return a tight lower bound.

With this example, we can also demonstrate that the arc-based approach can lead to smaller time-expanded networks and correspondingly smaller models to solve compared to node-based approaches. The smallest possible node-based discretization that leads to a tight lower bound achievable for this problem can be constructed using the approach of Marshall et al. (2021) by splitting the time horizon of node v_2 into the intervals $[1, 4)$ and $[4, 8)$ and using the full time horizon as the single interval for all other nodes. With the preprocessing rules they describe, they would obtain time-expanded networks of identical size for each commodity except for k_4, where they would have two timed copies of node v_2 with each of them having a timed copy of the outgoing arc. While this is not a large difference, it shows that in principle smaller time-expanded networks are possible, an effect we will later investigate empirically.

4.2 Refining the Discretization

Solving the relaxation R-SND-RR($\mathcal{T}$) only gives us a lower bound on the objective value of SND-RR. After solving the relaxation we need to answer the question if it is possible to convert the relaxation solution into a feasible solution to SND-RR with the same objective value, in which case that solution would be optimal. If that is not the case, we want to refine the discretization so that we do not receive the same relaxation solution again and repeat the process. We use some concepts introduced by Marshall et al. (2021) adapted to our relaxation for both these steps. Specifically, we make use of their notion of a *flat-solution* corresponding to a relaxed solution, which specifies flat-paths and consolidations but not dispatch times or intervals. They introduce the concept of a flat-solution being *implementable* when there exist dispatch times so that these consolidations are achievable in a feasible solution to the unrelaxed problem. The flat-paths from the flat-solution together with such dispatch times then make up a feasible solution to the unrelaxed problem with the same objective value as the relaxation solution, which also means that this solution is optimal (since it has the same objective value as the solution to the relaxation). To handle the case that the flat-solution is not implementable, they introduce the concept of a flat-solution being *representable* in the relaxed problem for a given time discretization. A flat-solution is representable when one can find a solution to the relaxed problem with the same flat-paths and consolidations. If a flat-solution is not implementable, we want to refine the discretization to make the solution not representable. Convergence relies on the fact that there are only a finite number of flat-solutions. The discretization refinement process can then conceptually be stated as follows:

1. Obtain an optimal solution to the relaxed problem and determine the corresponding flat-solution.
2. Determine if the flat-solution is implementable.

(a) If yes, determine dispatch times and return solution.

(b) If no, adjust discretization to make flat-solution not representable. Go back to step 1.

We now formally describe the adaption of these concepts to our relaxed problem. Note that for simplicity of presentation and notation, in the following we will assume that the paths in solutions obtained from the relaxation are simple, i.e., do not repeat nodes. As stated before, node and arc repetitions are possible in optimal solutions to the relaxation, but the necessary notation for the following arguments would be cumbersome. A *consolidation* (a, κ) is a tuple of an arc $a \in \mathcal{A}$ as well as a set of commodities $\kappa \subseteq \mathcal{K}$ with the interpretation that these commodities are transported together along this arc. Given a solution $W = (P, N)$ to R-SND-RR($\mathcal{T}$), we define its set of consolidations as $C = C(W) = \{(a, \bar{\kappa}(a, q)) \mid \forall a \in \mathcal{A}, \forall q \in [n_a] \colon |\bar{\kappa}(a, q)| \geq 1\}$ and call the pair (P, C) a *flat-solution*.

Definition 7 (Implementable flat-solution (adapted from Marshall et al. 2021))**.** *A flat-solution (P, C) is implementable if and only if there exist dispatch times T such that the solution $S = (P, T)$ to SND-RR is feasible and $t_a^{k_1} = t_a^{k_2}$ for all $(a, \bar{\kappa}) \in C, k_1, k_2 \in \bar{\kappa}$.*

Definition 8 (Representable flat-solution (adapted from Marshall et al. 2021))**.** *A flat-solution (P, C) is called representable with respect to a discretization $\mathcal{T}'$ if and only if there exist dispatch time intervals N' such that $W' = (P, N')$ is a feasible solution for R-SND-RR($\mathcal{T}'$) and the resulting set of consolidations is identical, i.e., $C(W') = C$.*

To detect if a flat-solution is implementable and if not to refine the discretization we adapt the algorithm proposed by Shu et al. (2025) to our relaxation. They introduce a *dispatch-node graph* which is constructed based on the flat-solution and show that if that graph contains no *too-long paths*, the flat-solution is implementable and if it contains such paths, adding certain time points based on the path to the discretization is sufficient to ensure that the flat-solution is no longer representable. We first reproduce from their work how the dispatch-node graph is constructed, how a too-long path is defined, and how they can be found. Then we present a theorem that details which time points need to be added to our time discretization to make a nonimplementable flat-solution also not representable.

Definition 9 (Dispatch-node graph (Shu et al. 2025))**.** *Given a flat-solution (P, C), its dispatch-node graph is defined as a directed graph $\mathscr{G}(P, C) = (\mathscr{V}, \mathscr{A})$ with a set of dispatch (and arrival) nodes $\mathscr{V} = \{(k, u) \mid \forall k \in \mathcal{K}, (u, v) \in p^k\} \cup \{(k, d_k) \mid \forall k \in \mathcal{K}\}$ and an arc set $\mathscr{A} = \{((k_1, u), (k_2, v)) \mid \forall ((u, v), \bar{\kappa}) \in C, k_1, k_2 \in \bar{\kappa}\}$. Each arc $a = ((k_1, u), (k_2, v)) \in \mathscr{A}$ has a length of $\rho_a = \tau_{(u,v)}$.*

They showed that the length of a path starting from any node representing the origin of any commodity (k, o_k) to any other node (k', v) gives a lower bound on the earliest dispatch time of the commodity k' at node v. If that dispatch time plus the remaining duration of a shortest path for k' from v to its destination $d_{k'}$ is later than its deadline, the solution is not implementable. Recall that by τ_{uv}^k we denote the length of a shortest path between two nodes $u, v \in \mathcal{G}^k$ in the commodity's subnetwork. Let $\mathscr{P}$ denote the set of paths in $\mathscr{G}(P, C)$ that start at any node (k, o_k) representing the source node of some commodity $k \in \mathcal{K}$.

Definition 10 (Too-long path (Shu et al. 2025)). *Given a flat-solution (P, C), a (not necessarily simple) path $p = (((k_1, v_1), (k_2, v_2)), \ldots, ((k_{n-1}, v_{n-1}), (k_n, v_n))) \in \mathscr{P}$ with $v_1 = o_{k_1}$ is a too-long path if $r_{k_1} + \sum_{a \in p} \rho_a + \tau^{k_n}_{v_n d_{k_n}} > \ell_{k_n}$.*

Theorem 1 (Shu et al. 2025). *A flat-solution (P, C) is nonimplementable if and only if its dispatch-node graph $\mathscr{G}(P, C)$ contains a too-long path.*

Shu et al. (2025) explain that a too-long path can contain a partial path that is also too-long and that if the discretization is adjusted so that the partial path can no longer occur in the dispatch-node graphs of representable flat-solutions, the original too-long path can also no longer occur. This motivates the following definition:

Definition 11 (Minimal too-long path (Shu et al. 2025)). *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.*

To find minimal too-long paths, Shu et al. (2025) propose a label-propagating algorithm that enumerates all paths starting from each commodity's source node in the dispatch-node graph. Since all arcs have positive lengths, a path either eventually becomes too-long, at which point it is not further propagated but stored as a minimal too-long path or reaches some commodity's sink node and is not further extended. We direct the reader to Shu et al. (2025) for a detailed description of the algorithm.

Similar to Theorem 3 of Shu et al. (2025), we establish which time intervals we can split in a discretization based on a too-long path to make a nonimplementable solution also not representable. The time points we split intervals at are identical to those of their theorem, but we show that in our relaxation it is sufficient for each time new point to only split a node interval of one commodity and to only split an interval of one arc. This is in stark contrast to all previously presented approaches, where a new time point at a node would lead to additional timed nodes for each commodity and additional copies of all arcs leaving that node.

We first explain conceptually which intervals we split, give an example and then formally state the result. Note that a too-long path represents consolidations that are not possible together because at the end of the path one commodity has been delayed so much that it can not reach its destination before the deadline. So we need to modify the discretization such that all of these consolidations together are also not possible in the relaxed problem. Consider that an arc $((k_j, v_j), (k_{j+1}, v_{j+1}))$ in the too-long path indicates that commodity k_{j+1} can not be dispatched from node v_{j+1} before commodity k_j arrives at that node, either because $k_j = k_{j+1}$ or because they are consolidated together on the arc (v_j, v_{j+1}). We now modify the discretization for only these commodities to ensure that if these commodities are to be consolidated as the too-long path indicates, they need to be at each node on time. This will not be possible for the last commodity k_n that by definition of a too-long path would arrive at node v_n so late that it could not arrive its destination on time. So dispatching commodities k_{n-1} and k_n together at this time would lead to an infeasible arc time interval (see Definition 3) for commodity k_n. Thus, these consolidations would not be possible in a feasible solution to the relaxed problem with these new time points and therefore the flat-solution would not be representable.

As an example, consider again the instance described in Table 1 and Figure 3. Assume we solve R-SND-RR($\mathcal{T}$) for this instance with an initial time discretization $\mathcal{T}$ such that $\mathcal{T}_a = \{[1,7]\}$ for all $a \in \mathcal{A}$ and $\mathcal{T}_{kv} = \{[\underline{t}_{kv}, \bar{t}_{kv}]\}$ for all $k \in \mathcal{K}$ and $v \in \mathcal{V}^k$, i.e., only one interval for each node and arc. Then, we would obtain a flat solution where k_1 and k_2 are consolidated on arc (v_1, v_2) while k_1 and k_3 are consolidated on arc (v_2, v_3), which will lead k_3 to be delayed at its destination. The corresponding too-long path is $((k_2, v_1), (k_1, v_2), (k_3, v_3))$. To now reflect that these consolidations together are not possible, we can split intervals as follows: first, for arc (v_1, v_2) we split the interval $[1, 7]$ into the intervals $[1, 1]$ and $[2, 7]$. For commodity k_2, only the second of these intervals is feasible, so if we would want to still consolidate commodities k_1 and k_2, they would need to both chose this interval. Second, for commodity k_1 at node v_2 we split the interval $[3, 5]$ into the intervals $[3, 3]$ and $[4, 5]$. Since $[4, 5]$ is the relaxed arrival interval for the interval $[2, 7]$ on arc (v_1, v_2), k_1 arrives in this interval if it is consolidated with k_2. Last, for arc (v_2, v_3) we split interval $[1, 7]$ into the intervals $[1, 3]$ and $[4, 7]$. For k_1, the relaxed dispatch interval of the arc interval $[1, 3]$ is the interval $[3, 3]$ at node v_2, so we can not dispatch k_1 in the early interval $[1, 3]$ on its second arc if we dispatched it in the late interval $[2, 7]$ on its first arc. But the interval $[4, 7]$ on arc (v_2, v_3) is not feasible for k_3, so it is not possible anymore to consolidate k_1 with k_2 on one arc and k_1 with k_3 on the other. Note that the resulting discretization is exactly the one given in Table 1b and Table 1c and the corresponding time-expanded networks is depicted in Figure 4.

We formalize the discretization refinement idea in the following theorem.

Theorem 2. *Let p denote a too-long path in the dispatch-node graph $\mathscr{G}(P, C)$ of a flat-solution (P, C). The flat-solution is not representable with respect to a discretization $\mathcal{T}'$ where for all arcs $((k_j, v_j), (k_{j+1}, v_{j+1})) \in p$ it holds that for the time point $t_j = r_{k_1} + \sum_{j'=1}^{j-1} \rho_{((k_{j'}, v_{j'}), (k_{j'+1}, v_{j'+1}))}$ there exists some $[t, t') \in \mathcal{T}'_{k_j v_j}$ with $t = t_j$ and for arc $a = (v_j, v_{j+1})$ there exists some $[t, t') \in \mathcal{T}'_a$ with $t = t_j$.*

Proof. We consider the arcs of the too-long path one by one and determine feasible arc intervals for the relevant commodities. For sake of presentation, we will assume that $k_j \neq k_{j+1}$ for all $j \in [n]$, but the same arguments also holds without this assumption. For the first arc $((k_1, v_1), (k_2, v_2))$ note that by definition of a too-long path, $v_1 = o_{k_1}$. We demand that an interval starting at the time point $t_1 = r_{k_1}$ is in $\mathcal{T}'_{k_1 o_{k_1}}$, which it is by definition of a discretization. We further demand that there is an interval $[h_{q_1}^{(v_1,v_2)}, h_{q_1+1}^{(v_1,v_2)}) \in \mathcal{T}'_{(v_1,v_2)}$ with $h_{q_1}^{(v_1,v_2)} = r_{k_1}$. So by Definition 3 only arc intervals with index $q'_1 \geq q_1$ are feasible for dispatching k_1 on this arc. Consider next the second arc $((k_2, v_2), (k_3, v_3))$, which leads us to demand that an interval starting at the time point $t_2 = r_{k_1} + \tau_{(v_1,v_2)}$ is in the set $\mathcal{T}'_{k_2 v_2}$ and that there exists some interval $[h_{q_2}^{(v_2,v_3)}, h_{q_2+1}^{(v_2,v_3)} \in \mathcal{T}'_{(v_2,v_3)})$ with $h_{q_2}^{(v_2,v_3)} = t_2$. So if commodities k_1 and k_2 are to be dispatched together on the first arc (v_1, v_2), they need to be dispatched in an interval $q'_1 \geq q_1$ and the relaxed arrival interval of k_2 starts at t_2 or later (since $h_{q'_1}^{(v_1,v_2)} + \tau_{(v_1,v_2)} \geq h_{q_1}^{(v_1,v_2)} + \tau_{(v_1,v_2)} = t_2$). So for the arc intervals for commodity k_2 to be relaxed time consistent (see Definition 6), it needs to be dispatched on arc (v_2, v_3) in some interval with index $q'_2 \geq q_2$. We now repeat this argument for the remaining arcs until we reach the last arc and commodity k_n, which, if it is to be consolidated with commodity k_{n-1}, needs to be dispatched in an interval $[h_{q'_{n-1}}^{(v_{n-1},v_n)}, h_{q'_{n-1}+1}^{(v_{n-1},v_n)}) \in \mathcal{T}'_{(v_{n-1},v_n)}$ with $h_{q'_{n-1}}^{(v_{n-1},v_n)} \geq t_{n-1}$ and since by definition of a too long path $t_{n-1} + \tau_{(v_{n-1},v_n)} > \bar{t}_{k_n v_n}$, this arc interval would not be feasible for k_n. So the consolidations implied by the too-long path are not possible to achieve with feasible and relaxed

time consistent arc intervals given this discretization. Therefore, the flat-solution is not representable with respect to this discretization. □

4.3 Strengthening the Initial Relaxation

The minimal initial discretization $\mathcal{T}$ necessary for our relaxation is to choose $\mathcal{T}_{kv} = \{[\underline{t}_{kv}, \bar{t}_{kv}]\}$ for all $k \in \mathcal{K}, v \in \mathcal{V}^k$ and $\mathcal{T}_a = \{[1, H)\}$ for all $a \in \mathcal{A}$. We strengthen the initial discretization by adding additional time points as proposed by Shu et al. (2025). They note that some commodities can never be consolidated on an arc (u, v) because the earliest time at which some commodity can arrive at u plus the travel time $\tau_{(u,v)}$ might already be later than the latest time at which another commodity has to be dispatched at v to still arrive on time. They show that these consolidations can be made impossible in the relaxation by inserting what they call *significant time points*. We adapt their statement to our relaxation and notation:

Theorem 3 (Significant time points (adapted from Shu et al. 2025))**.** *Given any two different commodities $k_1, k_2 \in \mathcal{K}$ and some arc $a = (u, v) \in \mathcal{A}^{k_1} \cap \mathcal{A}^{k_2}$ with $\underline{t}_{k_2 u} + \tau_{(u,v)} > \bar{t}_{k_1 v}$, if there exists an interval $[t, t') \in \mathcal{T}_a$ with $t \in [\bar{t}_{k_1 v} - \tau_{(u,v)} + 1, \underline{t}_{k_2 u}]$, then no interval in $\mathcal{T}_a$ can be feasible for both k_1 and k_2.*

Proof. Consider that the interval $[t, t')$ can not be feasible for k_1, since the earliest possible arrival time $t + \tau_{(u,v)} + 1 \geq \bar{t}_{k_1 v} + 1$ is already after the latest possible dispatch time $\bar{t}_{k_1 v} + 1$, compare Definition 3. Evidently, all later intervals are also not feasible for k_1. Similarly, no interval before $[t, t')$ can be feasible for k_2, since the end of any such interval would need to be before the earliest dispatch time $\underline{t}_{k_2 u}$. Therefore, there exist no interval that is feasible for both commodities, and they can never be consolidated together on this arc. □

For an arc $(u, v) \in \mathcal{A}$, we can collect the set of intervals of significant time points for all impossible consolidations as

$$I_{(u,v)} = \{[\bar{t}_{k_1 v} - \tau_{(u,v)} + 1, \underline{t}_{k_2 u}] \mid \forall k_1, k_2 \in \mathcal{K} \colon (u, v) \in \mathcal{A}^{k_1} \cap \mathcal{A}^{k_2}, \underline{t}_{k_2 u} + \tau_{(u,v)} > \bar{t}_{k_1 v}\}.$$

Shu et al. (2025) propose solving a minimum hitting set problem to determine the smallest set of time points to cover all intervals. Since they can only add time points for a node and thereby to all outgoing arcs of that node, they determine the minimum hitting set for the intervals of all outgoing arcs of a node u, i.e., for $I_u = \bigcup_{v \colon (u,v) \in \mathcal{A}} I_{(u,v)}$. Our relaxation permits arc-dependent time discretization, therefore we solve the minimum hitting set problem for each arc and add the resulting time points only to the discretization for this arc. This can potentially lead to fewer intervals for each arc in the relaxed problem compared to adding the same time points for all outgoing arcs. Let the ordered set $T_{(u,v)} = \{t_1, \ldots, t_m\}$ denote a minimum hitting set for $I_{(u,v)}$. We initialize the set of intervals for arc (u, v) as $\mathcal{T}_{(u,v)} = \{[0, t_1), [t_1, t_2), \ldots, [t_m, H)\}$.

4.4 Dynamic Discretization Discovery Algorithm

We combine the previously introduced concepts into Algorithm 1 to solve SND-RR to optimality. The algorithm first initializes the discretization as described in Section 4.3. Then it iteratively solves the relaxation

R-SND-RR($\mathcal{T}$) and uses the heuristic from Marshall et al. (2021) (see Appendix A) to try and find a primal solution of identical value, which will succeed if the flat-solution corresponding to the relaxation solution is implementable. In that case, the algorithm found an optimal solution and can terminate. Otherwise, the flat-solution is not implementable, and we add time points to the discretization to make it not representable and repeat the process.

Algorithm 1 Arc-based Dynamic Discretization Discovery Algorithm for SND-RR

Require: An instance to SND-RR
Ensure: S is an optimal solution
1: Initialize discretization $\mathcal{T}$
2: $\overline{z} \leftarrow \infty, \underline{z} \leftarrow -\infty, S \leftarrow \emptyset$ ▷ Upper bound $\overline{z}$, lower bound $\underline{z}$
3: **while** $\overline{z} > \underline{z}$ **do**
4: Find an optimal solution W to R-SND-RR($\mathcal{T}$), $\underline{z} \leftarrow c(W)$
5: Obtain solution S' to SND-RR from primal heuristic
6: **if** $\overline{z} > c(S')$ **then**
7: $\overline{z} \leftarrow c(S'), S \leftarrow S'$
8: **end if**
9: **if** $\overline{z} > \underline{z}$ **then**
10: Refine discretization according to Algorithm 2
11: **end if**
12: **end while**

Algorithm 2 describes our refinement approach, which is based on adding time points according to Theorem 2. The approach always starts with the current discretization. It then constructs the dispatch-node graph and finds a set of minimal too-long paths as described by Shu et al. (2025). In principle, we could then split arc and node intervals along each too-long path such that the condition of Theorem 2 is fulfilled. But Shu et al. (2025) showed that in some cases a too-long path can be eliminated by only refining the discretization for the first few arcs, if the remaining discretization is fine enough to ensure that the problematic consolidations do not occur again. They make use of this observation and sort all the pairs of nodes and time points at which the discretization would need to be refined by non-decreasing order of time. Then, they consider these pairs in order. If all too-long paths that would introduce the currently considered time point at a node are already eliminated, the discretization is not refined with this time point. We use the same procedure, but instead collect the time points that would be added on each arc. Then, if any of the too-long paths leading to this time point has not been eliminated, we refine the discretization. We first check if commodity k_j associated with this too-long path on this arc already has an interval starting at t_j at node v_j and if not, we replace the node interval that contains t_j by the two intervals that we obtain by splitting it at this time, see Line 10. Then, we check if arc (v_j, v_{j+1}) already has an arc interval starting at t_j and if not, we split the interval in the same manner, see Line 14.

We further remark on some details of our implementation:

- The algorithm allows early termination if the gap $\frac{\overline{z}-\underline{z}}{\overline{z}}$ reaches some pre-defined target gap value.
- By default, we solve the relaxation only to a gap of $98\,\%$ of the target gap value and then use the lower bound obtained by the solver as $\underline{z}$ instead of $c(W)$.

Algorithm 2 Refinement procedure

Require: Time discretization $\mathcal{T}$, relaxation solution W
Ensure: Time discretization $\mathcal{T}'$ such that flat-solution corresponding to W is not representable

$\mathcal{T}' \leftarrow \mathcal{T}$
Determine flat-solution (P, C) corresponding to W
Construct dispatch-node graph $\mathscr{G}(P, C)$
Find a set $\mathscr{P}'$ of too-long paths
Determine set $\Delta T \subseteq \mathbb{N} \times \mathcal{A}$ of arc time points to add, sorted by time
for all $(t, (i, j)) \in \Delta T$ **do**
    for all $p \in \mathscr{P}'$ and j with $(v_j, v_{j+1}) = (i, j)$, $t_j = t$, and p not eliminated by $\mathcal{T}$ **do**
        if $\nexists q \in [n_{k_j v_j}]$ with $t_q^{k_j v_j} = t_j$ **then**
            Let q' such that $t_j \in [t_{q'}^{k_j v_j}, t_{q'+1}^{k_j v_j})$
            $\mathcal{T}'_{k_j v_j} \leftarrow (\mathcal{T}'_{k_j v_j} \setminus \{[t_{q'}^{k_j v_j}, t_{q'+1}^{k_j v_j})\}) \cup \{[t_{q'}^{k_j v_j}, t_j), [t_j, t_{q'+1}^{k_v v_j})\}$
        end if
        if $\nexists q \in [n_{(v_j, v_{j+1})}]$ with $h_q^{(v_j, v_{j+1})} = t_j$ **then**
            Let q' such that $t_j \in [h_{q'}^{(v_j, v_{j+1})}, h_{q'+1}^{(v_j, v_{j+1})})$
            $\mathcal{T}'_{(v_j, v_{j+1})} \leftarrow (\mathcal{T}'_{(v_j, v_{j+1})} \setminus \{[h_{q'}^{(v_j, v_{j+1})}, h_{q'+1}^{(v_j, v_{j+1})})\}) \cup \{[h_{q'}^{(v_j, v_{j+1})}, t_j), [t_j, h_{q'+1}^{(v_j, v_{j+1})})\}$
        end if
    end for
end for

- Optionally, one can use the adaptive gap procedure proposed by Marshall et al. (2021) where the gap to which the relaxation is solved depends on the current value of $\frac{\overline{z} - \underline{z}}{\overline{z}}$.
- We terminate the solving process if the solver for the relaxation obtains a lower bound that is good enough to reach the target gap value.
- The relaxation model also contains the strong linking valid inequalities as proposed by Marshall et al. (2021) in their Section 5.4.
- As proposed by Shu et al. (2025), we obtain too-long paths solutions the solver finds for the relaxation and not just the best solution. We also try to repair these solutions using the primal heuristic.

5 Computational Study

We introduce the algorithms we compare against in Section 5.1 and the instance sets we use in Section 5.2. Then, our computational study has three parts: In Section 5.3 we compare our arc-based algorithm against the two state-of-the-art algorithms on a commonly used benchmark set of instances to confirm that our implementation is competitive. Since the used benchmark set is highly artificial and does not match the topology of real-life logistic networks, we then investigate the benefits of arc-based discretization for networks with specific topologies in Section 5.4. Lastly, we note that the refinement procedure described in Algorithm 2 refines the discretization at nodes only for the commodities where doing so is necessary to guarantee that the same too-long paths do not occur again. Therefore, in Section 5.5 we compare this

baseline refinement procedure with variants where the node discretization is refined also for other commodities in the same consolidation or even all commodities.

5.1 Compared Algorithms

We call our algorithm ADDD. For the comparisons against the state-of-the-art, we have implemented the algorithms by Marshall et al. (2021) (called MBSH) and Shu et al. (2025) (called SXB) ourselves, clarifying any unclear details with the corresponding authors. For MBSH we implemented the most competitive version of their algorithm called 'Fewer Time Points'/'Adaptive' (depending on if an adaptive gap was used, specified for each experiment below). The only modification we have made to SXB is that we use the same primal heuristic as for ADDD and MBSH because the heuristic of Shu et al. (2025) did not materially improve convergence in our preliminary experiments and because we want to focus the experiments on comparing the impact of different relaxations. Furthermore, we introduce a new algorithm MBSH-E based on MBSH but enhanced with the following algorithmic features proposed by Shu et al. (2025): Every solution found by the solver for the relaxation is used to refine the discretization and repaired using the primal heuristic, not just the best solution. The significant time points are added before the first iteration. The refinement procedure based on too-long paths is used. Preliminary experiments showed that all of these modifications improved the performance of the algorithm. Importantly, the only differences between the algorithms SXB, MBSH-E, and ADDD is that they use the relaxations proposed by Boland et al. (2017), Marshall et al. (2021) and us respectively. They all use the refinement strategy from Shu et al. (2025), refining the discretization at nodes for SXB and MBSH-E and on arcs for ADDD. So any difference in performance between these models must stem from the used relaxation, allowing us to isolate the benefit of our contribution.

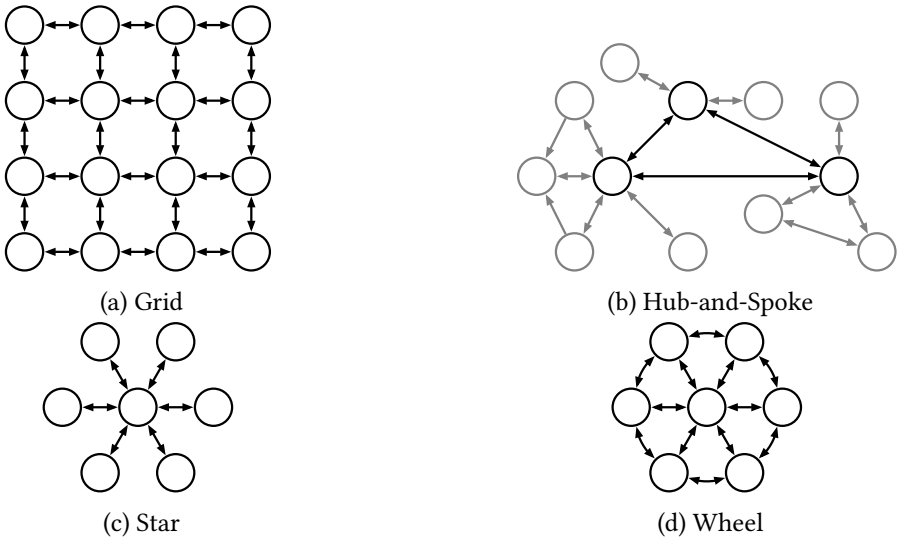
Marshall et al. (2021) also provide an open-source implementation of their algorithm at `github.com/mathgeekcoder/DDDI`, which we also use in our comparisons and call MBSH-OG. Van Dyk and Koenemann (2024) also provide an open-source implementation of their algorithm at `https://github.com/madisonvandyk/SND-RR`, but in our preliminary experiments it was not competitive against any of the other algorithms, we therefore omit the results for this algorithm. We want to mention that we are very grateful to the authors for providing their code and instances openly accessible and will follow their example upon acceptance of this manuscript.

All algorithms that we compare run in Python 3.12.2 and use Gurobi 12.0.1 (Gurobi Optimization, LLC 2025) to solve the relaxation and the linear program in the primal heuristic. All instances were solved with a time limit of one hour on machines with a Xeon L5630 Quad Core 2.13 GHz processor. Note that the relaxation models in MBSH-OG are modified during the refinement step instead of rebuilding them in every iteration as in our implementation. Except for very easy instances that solve within a few seconds, the amount of time spent outside of solving the relaxation model with Gurobi is negligible for all approaches, indicating that rebuilding the models is not a large drag on performance for harder instances.

5.2 Instance Sets

We use two instance sets from the literature as well as a new one that we propose to investigate networks with differing topologies. The first set are the 558 instances from Boland et al. (2017) with 20 to 30 nodes, 228 to 685 arcs, and 39 to 400 commodities. Note that in these instances, the commodities' networks are not restricted, so in principle they are equivalent to the full flat-network. In a preprocessing step, we construct subnetworks for each commodity k based on the nodes and arcs that can be part of a k-feasible path. The instances in this set were classified by Marshall et al. (2021) depending on the tightness of the release-deadline time windows and the relation of variable to fixed costs. They classify an instance as having low flexibility (LF) if $\min_{k \in \mathcal{K}} l_k - r_k + \tau^k_{o_k,d_k} < 227$ and high flexibility (HF) otherwise. An instance also has low cost ratio (LC) if $\frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \frac{f_a}{c_a u_a} < 0.175$ and high cost ratio (HC) otherwise. For our analysis, we report the result for each of the four resulting instance classes *LC/LF*, *LC/HF*, *HC/LF*, and *HC/HF*.

The second set are the 576 instances proposed by Van Dyk and Koenemann (2024) that were designed to showcase the benefit of arc-based discretizations of time. Specifically, in (near-)optimal solutions to these instances, some nodes will have dispatches on many outgoing arcs at many time points. So if all arcs receive timed copies for each commodity that can traverse them at all time points for this node, the relaxation should become difficult to solve quickly. Their instances have 20 or 30 nodes, 45 to 480 arcs, and 100 to 300 commodities. They divide their instances into three classes of 192 instances each, which we call *designated path*, *critical times*, and *hub-and-spoke (easy)*. In designated path instances, each commodity's subnetwork $\mathcal{G}^k$ consists only of one origin-destination path and just the dispatch times need to be decided. In critical times instances, release times and deadlines of commodities are modified such that there is just a handful of time points per node to simulate warehouse shifts. Hub-and-spoke instances represent regional networks that are connected by hub nodes such that commodities can only travel between two regions via the hubs.



(a) Grid (b) Hub-and-Spoke (c) Star (d) Wheel

Figure 5: Examples for the four topology classes in the new instance set

We propose a new third set that is designed to showcase how the different algorithms perform on

networks with varying topologies. As in Van Dyk and Koenemann (2024), our assumption is that arc-based discretization has a benefit versus node-based discretization if the network contains some hub-like nodes which have a high outdegree and are traversed by many commodities in near-optimal solutions. Therefore, we create 50 instances each based on four different topologies with a varying degree of how important a few hub-like nodes are. For a detailed description of the generation process for the instances see Appendix B. The instances have 12 to 30 nodes, 40 to 134 arcs, and 50 to 100 commodities. So they are rather small compared to the instances in the other sets, but they are still reasonably hard due to high fixed costs for dispatches and flexible time windows for the commodities, making smart consolidation imperative for high-quality solutions. We next describe the underlying topologies.

In the class *grid* the network is a four by four grid where all arcs also have the same travel time, see Figure 5a for an example. Therefore, there are no candidates for hub-like nodes: no node is significantly better connected than any other. We would expect the arc-based approach to have very little or no benefit on these instances.

The class *hub-and-spoke (hard)* contains instances with the same topology as those in hub-and-spoke (easy). Some nodes are designated hubs and are interconnected, all other nodes are assigned to a hub and only connected to the hub and possibly other nodes assigned to the same hub, see Figure 5b for an example. Due to the choice of the commodity parameters, these instances are significantly harder to solve than the ones in the easy set. By design, a few nodes will serve as hubs over which most commodities need to traverse, therefore we expect the arc-based approaches to perform better on this set.

The classes *wheel* and *star* are similar in that they feature one dominant node to which all other nodes are connected, so it has a very high out-degree and many commodities will traverse this node due to the excellent consolidation opportunities. Therefore, we expect the benefit of the arc-based discretization to be the most pronounced on these sets of instances.

5.3 Comparison with State-of-the-Art Algorithms

We compare our algorithm to the state-of-the-art algorithms on the instance set by Boland et al. (2017), which was also used in the papers in which these algorithms were introduced. The goal of this experiment is to evaluate if the arc-based approach can match the state-of-the-art on the classic dataset and if our implementation is competitive with the available implementation MBSH-OG. All algorithms are configured to solve the overall problem to a target gap of $1\,\%$ and each relaxation to a gap of $0.98\,\%$. Table 2 displays the average gap between upper and lower bound at termination, solving time, number of iterations, number of variables in the relaxation model in the last iteration, and the percentage of solved instances by algorithm and instance class. Note that all results were computed on identical machines using the same solver and programming language.

We can observe that as reported by Shu et al. (2025), SXB can solve more instances and faster than MBSH-OG. Furthermore, our implementation MBSH beats MBSH-OG by a small margin. We suspect that this might be due to minor differences in the implementation of the refinement strategy, that was not fully specified by Marshall et al. (2021). The enhanced version MBSH-E outperforms MBSH significantly, confirming the observation by Shu et al. (2025) that their algorithm enhancements significantly improve

Class	Algorithm	Gap (%)	Time (s)	# Iterations	Variables	Solved (%)
HC/HF	ADDD	0.89	355.34	4.8	11,530	96.0
	MBSH-E	0.88	384.56	3.4	23,716	96.6
	MBSH-OG	1.44	953.97	17.7	17,259	83.1
	MBSH	1.39	885.85	16.2	14,039	84.7
	SXB	0.90	405.77	3.2	78,665	94.9
HC/LF	ADDD	0.76	33.49	3.3	12,047	100.0
	MBSH-E	0.71	33.99	2.3	25,725	100.0
	MBSH-OG	0.85	147.56	12.3	17,523	98.4
	MBSH	0.82	116.04	11.3	14,845	98.9
	SXB	0.73	62.59	2.3	106,339	100.0
LC/HF	ADDD	0.11	0.13	1.0	634	100.0
	MBSH-E	0.08	0.17	1.0	761	100.0
	MBSH-OG	0.58	0.12	1.4	543	100.0
	MBSH	0.52	0.15	1.4	343	100.0
	SXB	0.18	0.28	1.0	3,546	100.0
LC/LF	ADDD	0.39	0.35	1.1	1,979	100.0
	MBSH-E	0.27	0.53	1.1	3,210	100.0
	MBSH-OG	0.71	1.22	3.9	1,712	100.0
	MBSH	0.66	0.81	3.4	1,500	100.0
	SXB	0.32	1.04	1.0	13,763	100.0

Table 2: Results on instances from Boland et al. (2017), 1% gap limit

performance. Our algorithm ADDD, which also uses the same enhancements, performs equally well as MBSH-E, both of which slightly outperform SXB. We suspect that in general interval-based discretizations perform better than the node-time based ones used by SXB due to the smaller resulting relaxation models. Note that MBSH and MBSH-OG produce even smaller relaxation models than ADDD on the LC/LF and LC/HF instances. This is because they do not use the significant time points, meaning they start with very small initial discretizations compared to the other models.

Overall, we observe that on these instances the arc-based discretization approach does not yield significant benefits but can match the state-of-the-art achieved by node-based discretizations. Additionally, we note that our algorithm ADDD needs more iterations to solve the instances in the two harder classes HC/HF and HC/LF, but the smaller and easier to solve relaxation models make up for this drawback. Since MBSH-E beats both MBSH and MBSH-OG by a significant margin, we only consider the first of these three algorithms from now on.

5.4 Benefit of Arc-Based Discretization on Different Topologies

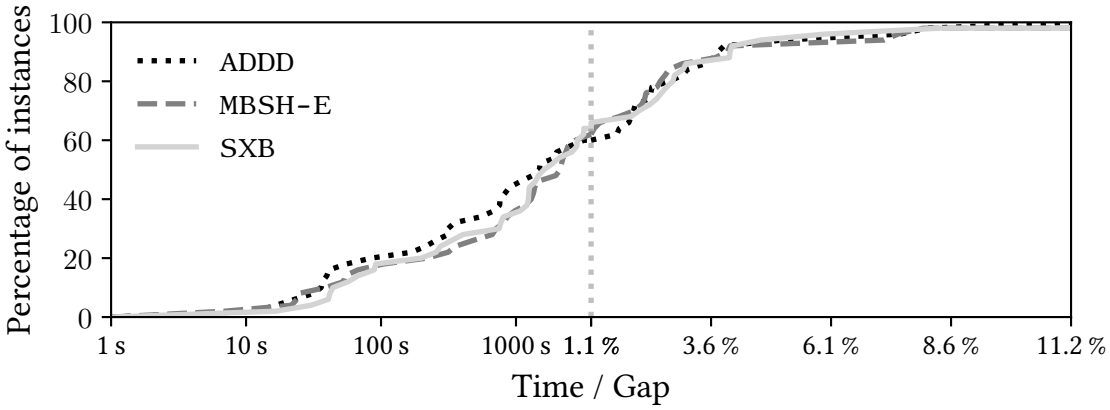
The instances in the previous section are based on randomly generated networks. Thus, they do not have any specific topology that real-life logistics networks might have. We therefore want to investigate next if an arc-based discretization can yield a computational advantage on instances with specific topologies. For this purpose we compare the performance of ADDD, MBSH-E, and SXB on the dataset by Van Dyk and Koenemann (2024) and our newly generated instances. Recall that these algorithms are identical except for the utilized relaxation approach, allowing us to investigate the impact of node-based versus arc-based discretization for varying topologies.

On multiple of these instances, the refinement algorithm of Shu et al. (2025) (that is used by all three methods) finds such a large number of too-long paths that the solving process terminates due to memory overflow. We therefore slightly modify the algorithm that finds the too-long paths as follows: at each node in the dispatch-node graph we store the highest time value seen so far, where the time value is the release time of the commodity at a start of a too-long path plus the length of the partial path to the node. Now whenever a label would be extended to a dispatch-node, if its time value is not higher than the value stored at the node, the label is not extended. Effectively, this prunes some too-long paths that cause 'less delay' than some other too-long path. We use this modification with all three algorithms and observe no more memory overflows. Additionally, we use the adaptive gap procedure (with a target gap of $1\,\%$) as described by Marshall et al. (2021) for these experiments since without it even the first relaxation can not be solved for some instances. The results are reported in Table 3.

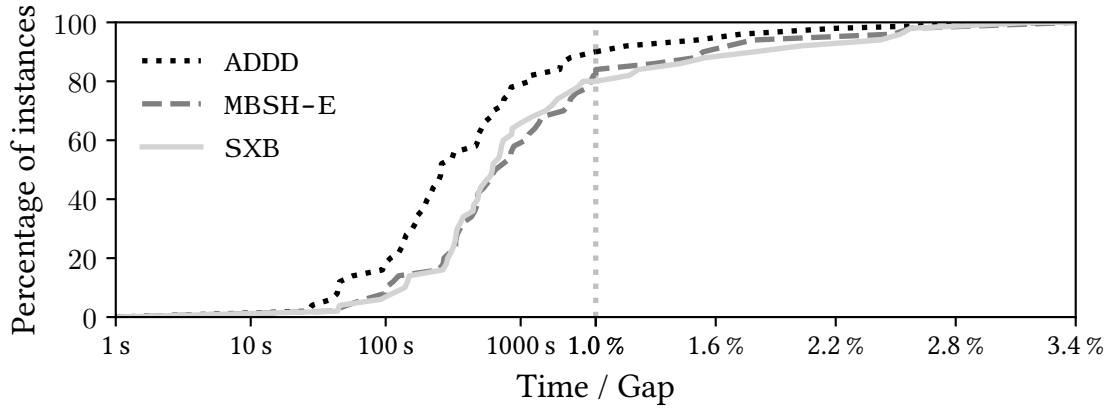
First, we observe that the instances by Van Dyk and Koenemann (2024) are no match for state-of-the-art algorithms, as all methods solve them in a few seconds on average. Our newly generated instances are significantly harder to solve and no algorithm can solve all of them, except for the star class. As we expected, ADDD is not better on the grid instances which have no hub-like nodes that can profit from the arc-based discretization. In fact, SXB solves the most instances in this class. The picture looks a bit better on the hub-and-spoke instances, where ADDD can solve $8\,\%$ more of the instances than MBSH-E and also terminates faster on average. On the wheel and star instances that have only one hub-like node,

Class	Algorithm	Gap (%)	Time (s)	# Iterations	Variables	Solved (%)
Critical time	ADDD	0.70	8.90	2.9	5,493	100.0
	MBSH-E	0.73	7.92	2.2	7,679	100.0
	SXB	0.68	10.06	2.2	19,711	100.0
Designated path	ADDD	0.71	7.24	4.2	4,411	100.0
	MBSH-E	0.69	5.55	2.7	7,494	100.0
	SXB	0.69	11.07	2.7	37,750	100.0
Grid	ADDD	1.89	1837.56	3.3	6,110	60.0
	MBSH-E	1.81	1970.69	2.7	8,458	62.0
	SXB	1.78	1917.91	2.6	17,296	66.0
Hub-and-spoke (easy)	ADDD	0.64	6.79	3.6	2,441	100.0
	MBSH-E	0.59	5.62	2.5	3,682	100.0
	SXB	0.61	7.32	2.6	10,319	100.0
Hub-and-spoke (hard)	ADDD	1.06	826.55	4.2	20,780	90.0
	MBSH-E	1.13	1385.93	2.8	32,831	82.0
	SXB	1.17	1288.26	2.8	63,247	80.0
Star	ADDD	0.87	22.21	3.6	22,601	100.0
	MBSH-E	0.95	120.92	2.0	32,752	100.0
	SXB	0.94	180.09	2.0	71,738	100.0
Wheel	ADDD	1.10	1918.39	4.9	4,458	82.0
	MBSH-E	1.49	2851.93	3.1	8,625	48.0
	SXB	1.57	2905.35	2.9	25,048	36.0

Table 3: Results on instances from Van Dyk and Koenemann (2024) and our new instances, 1% gap limit



(a) Grid instances



(b) Hub-and-spoke instances

Figure 6: Percentage of instances each approach could solve within a given amount of time or to a given gap if not solved within one hour

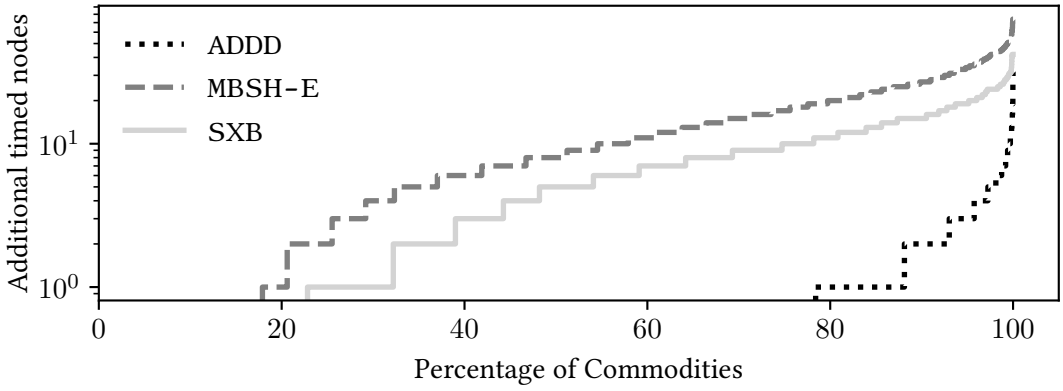
ADDD significantly outperforms the other two algorithms, solving $34\,\%$ more of the wheel instances than MBSH-E and solving the star instances about six times faster. We can also see that the decrease in model size when using arc-based discretization is more pronounced for the wheel instances (almost $-50\,\%$ fewer variables compared to MBSH-E) than for the grid instances (about $-28\,\%$).

Figure 6 additionally shows the percentage of grid and hub-and-spoke (hard) instances each algorithm could solve within a given time limit or to a certain gap. Note that for the grid instances the algorithms also perform very similarly over time, none of them being able to solve significantly more or less instances at a given time or to a smaller gap. In contrast, on the hub-and-spoke instances ADDD solves a significantly more instances for any amount of solving time and also achieves smaller gaps at the time limit.

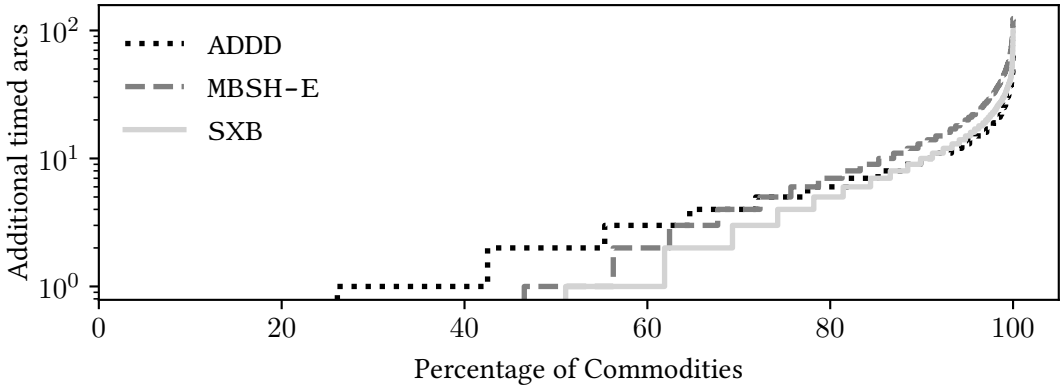
These experiments confirm our assumption that arc-based discretization becomes increasingly more competitive if the network topology features hub-like nodes that many commodities traverse. Since many real-life logistic networks follow a hub-and-spoke or star structure, this highlights the practical relevance of our contribution.

We further analyze the growth of the commodities' relaxed time-expanded networks $G^k_{\mathcal{T}}$ for each al-

gorithm and topology class. Figure 7 shows the cumulative distribution of the growth of each commodity's subnetwork from the first to the last iteration. Specifically, each line shows the proportion of commodities over all of our topology instances whose increase in the number of timed transport arc copies or timed node copies is at most the value on the (logarithmic) y-axis. We observe that the node discretizations need to rarely be refined for most commodities in ADDD compared to the other two methods, with almost $80\,\%$ of commodities not receiving a single additional timed node. This is not true for the arc discretizations: observe that in fact more commodities gain additional timed arc copies (about $70\,\%$) in ADDD than in the other two algorithms (about $50\,\%$). This is probably because the initial discretization of ADDD with the arc-wise significant time points is sparser than the initial discretization of the other approaches, which may sometimes by pure chance have added a helpful time point. Yet we also observe that for the commodities with the largest increase in the number of timed arc copies, the growth in ADDD is comparable to that of SXB (which starts with the largest initial discretizations) and less than that of MSB. This shows that while ADDD starts with the smallest relaxed time-expanded networks, it also keeps them small, especially with regard to the timed node copies. We note that the same analysis on a per-instance-class basis reveals qualitatively the same results, we therefore omit the figures.



(a) Growth of $V^k_{\mathcal{T}}$



(b) Growth of $A^k_{\mathcal{T}_\mathrm{t}}$

Figure 7: Cumulative distributions of relaxed time-expanded network growth

5.5 Impact of Different Policies for Node Interval Refinement

During refinement, the algorithm ADDD only refines the node time for one commodity for each arc in a too-long path, which we showed is sufficient for the too-long path to not reoccur. But in principle, we could also refine the node time for other commodities so that they also can not traverse the arc in the newly split interval unless they are sufficiently early at the dispatch node. On the one hand this might unnecessarily increase the model size but on the other hand the additional time points might prevent too-long paths from occurring in the future. We therefore compare our base algorithm to six variants that refine the node time for additional commodities to investigate this trade-off. Specifically, whenever we refine the node discretization as in Line 10 of Algorithm 2, we not only apply this procedure to the commodity k_j but also some others. Let (a, κ) be the consolidation in the flat solution such that $a = (v_j, v_{j+1})$ and $k_j \in \kappa$. Then, strategy CONS-DISPATCH also splits the intervals at time point t_j for all commodities $k \in \kappa$, i.e., all commodities that were consolidated together with the problematic commodity k_j at the node from which they were dispatched. Strategy CONS-ARRIVAL instead splits the intervals at the arrival node v_{j+1} for all these commodities at time point t_{j+1} while strategy CONS-BOTH does both. Similarly, ALL-DISPATCH, ALL-ARRIVAL and ALL-BOTH split the intervals at the dispatch, arrival or both nodes for all commodities that have either node in their subnetwork.

The results for all strategies on all instances are reported in Table 4. We can observe that they all perform very similarly, but the some trends match our expectations. Specifically, we see that ADDD needs the most iterations on average while the strategy adding the most timepoints (ALL-BOTH) needs the least. With regards to overall runtime, ALL-ARRIVAL and ALL-DISPATCH perform about $10\,\%$ better than ADDD, indicating that sharing node discretizations can be mildly beneficial.

Algorithm	Gap (%)	Time (s)	# Iterations	Variables	Solved (%)
ADDD	0.74	232.22	3.7	6,966	96.9
ALL-ARRIVAL	0.72	210.01	3.4	7,597	97.7
ALL-BOTH	0.72	214.84	3.4	7,965	97.5
ALL-DISPATCH	0.72	209.95	3.4	7,432	97.6
CONS-ARRIVAL	0.73	216.00	3.5	7,183	97.2
CONS-BOTH	0.73	220.82	3.4	7,341	97.3
CONS-DISPATCH	0.71	220.59	3.5	7,145	97.2

Table 4: Results on all instances, 1% gap limit

6 Conclusions and Future Work

We presented a novel arc-based relaxation for the continuous time service network design problem and adapted various recent algorithmic improvements for node-based relaxation approaches to it. Our experiments showed that while the arc-based relaxation is weaker than the node-based one, the resulting integer programming models are significantly smaller and can be solved faster, leading to overall faster solving

speeds on some challenging instance sets. In principle, we can interpret the stronger relaxations of the node-based approaches as arc-based approaches that 'accidentally' also refine the discretization on arcs where this is not necessary, preventing illegal consolidations in future iterations. This begs the question if we can somehow predict where illegal consolidations will occur in future iterations and preemptively adjust the discretization to prevent them.

Another potential improvement of interest to us is the question of how to solve each relaxation faster, since we observed very poor convergence of the MIP solver for some relaxation models. Since smaller discretizations usually can be solved faster, one could ask how to determine the smallest possible discretization that prevents all too-long paths seen in previous iterations of the algorithm. If we could determine such a discretization, it would be possible to not only add but also remove time points from the discretization while still ensuring no backsliding with respect to nonimplementable solutions. We are currently investigating this question and preliminary experiments seem promising that smaller discretizations can be determined systematically.

We note that with recent advances (especially those of Shu et al. 2025) the instances proposed by Boland et al. (2017) and Van Dyk and Koenemann (2024) do not pose significant challenges anymore. It would be helpful for further algorithmic developments to acquire challenging data sets from real-life service network design applications. Specifically applications with highly-connected networks (large node-degrees) could be amenable to arc-based approaches.

Code and Data for our implementation and the used instance sets will be published on GitHub if accepted.

A Primal Heuristic

To obtain primal feasible solutions before we find an implementable flat-solution, we utilize the heuristic first proposed by Boland et al. (2017) with the modified objective function of Marshall et al. (2021). As an input, it receives a relaxation solution $W = (P, N)$ and solves a linear program to determine consistent dispatch times for the flat-paths P while minimizing the deviation of dispatch times between commodities

that are consolidated in the relaxed solution. In our notation, the linear program is as follows:

$$\min \quad \sum_{a\in\mathcal{A}}\sum_{q=1}^{n_a}\sum_{\{k_1,k_2\}\subseteq\bar{\kappa}(a,q)} f_a\delta_a^{\{k_1,k_2\}} \tag{3}$$

$$\text{s.t.} \quad \gamma_{a_i}^k + \tau_{a_i} \le \gamma_{a_{i+1}}^k \qquad \forall k\in\mathcal{K}, \forall i\in[n_k-1] \tag{4}$$

$$\gamma_{a_1}^k \ge r_k \qquad \forall k\in\mathcal{K} \tag{5}$$

$$\gamma_{a_{n_k}}^k + \tau_{a_{n_k}} \le \ell_k \qquad \forall k\in\mathcal{K} \tag{6}$$

$$\delta_a^{\{k_1,k_2\}} \ge \gamma_a^{k_1} - \gamma_a^{k_2} \quad \forall a\in\mathcal{A}, \forall q\in[n_a], \forall\{k_1,k_2\}\subseteq\bar{\kappa}(a,q) \tag{7}$$

$$\delta_a^{\{k_1,k_2\}} \ge \gamma_a^{k_2} - \gamma_a^{k_1} \quad \forall a\in\mathcal{A}, \forall q\in[n_a], \forall\{k_1,k_2\}\subseteq\bar{\kappa}(a,q) \tag{8}$$

$$\gamma_a^k \ge 0 \qquad \forall k\in\mathcal{K}, \forall a\in p^k \tag{9}$$

$$\delta_a^{\{k_1,k_2\}} \ge 0 \qquad \forall a\in\mathcal{A}, \forall q\in[n_a], \forall\{k_1,k_2\}\subseteq\bar{\kappa}(a,q) \tag{10}$$

The variables γ_a^k represent the dispatch time t_a^k in the resulting solution $S = (P,T)$ to SND-RR. The variables $\delta_a^{\{k_1,k_2\}}$ represent the difference in dispatch times between commodities that are consolidated in the relaxed solution. Constraints (4)–(6) ensure that the dispatch times are consistent and the remaining constraints enforce the described interpretation of the δ-variables. Since the flat-paths in the relaxation solution are k-feasible, this linear program is always feasible. Note that if the flat-solution corresponding to W is implementable, this primal heuristic will find dispatch times T so that $S = (P,T)$ is optimal for SND-RR and the objective value of the linear program will be zero.

B Details on Newly Generated Instances

For all instances we set the vehicle capacity on all arcs to $c_a = 1$. All commodities have randomly drawn source and sink nodes and their quantity q_k is drawn uniformly from $[0.01, 0.5]$ and rounded to 2 decimal places. This ensures that the vehicle capacities are relatively large compared to most commodities' quantities, making consolidation attractive. The time windows of each commodity were determined as described in Boland et al. (2017), choosing parameter values that lead to more flexible time windows. For each commodity k, we determine the length of the shortest path with regards to the travel time (described below for each class) from its origin to its destination $\tau^k_{o_k d_k}$. Let $L = \max\{\tau^k_{o_k d_k} \mid k\in\mathcal{K}\}$ be the longest minimal travel time. Then, we draw the release time r_k for each commodity k from a normal distribution with mean L and standard deviation $\frac{L}{3}$ (repeating the processing if a negative number is drawn). Then we draw its flexibility F_k from a normal distribution with mean $\frac{L}{2}$ and standard deviation $\frac{L}{12}$. Its deadline is set to $d_k = r_k + \tau^k_{o_k d_k} + F_k$. The fixed costs f_a for dispatching on each arc are determined proportional to the travel time on that arc, also described below. For the variable costs, we set $c_a^k = f_a \cdot q_k$, which ensures that a fully loaded vehicle produces exactly the same fixed and variable costs (due to the unit capacity), also contributing to the attractiveness of consolidating.

For each class, the network size and number of commodities was chosen based on the order of magnitude of other instances in the literature and adjusted until not all instances in the class were either trivial

to solve or unsolvable within the one hour time limit.

B.1 Grid

We generated networks with a four by four grid of nodes and 48 arcs. Each arc has travel time and fixed cost $\tau_a = f_a = 10$. All instances in this class have 50 commodities.

B.2 Hub-and-Spoke (hard)

We generated networks with 5 hubs and 25 satellites. For each node we uniformly generated coordinates in $[0, 100]^2$ and assigned each node to its closest hub by euclidean distance, forming this hubs region. Each node has arcs to and from its hub and we further generated arcs randomly between nodes of the same hub until $75\,\%$ of the possible arcs in this region were created. The travel time of each arc is the euclidean distance between the two nodes positions rounded up plus one. The fixed cost of each arc is its length plus 10 to reflect some constant amount of effort for each dispatch. All instances in this class have 75 commodities.

B.3 Star

We generated networks with one hub and 20 spoke nodes, each of which is connected with the hub. The two arcs between the hub and each spoke have the same travel time, uniformly drawn from $[10, 100]$ and rounded to the nearest integer. Again the fixed cost of each arc is its length plus 10. All instances in this class have 100 commodities.

B.4 Wheel

We generated networks with one hub and 11 spokes that are distributed evenly around the hub. Each arc to or from the hub has a travel time of $\tau_a = 100$, while the arcs between the spokes have a travel time of $\tau_a = \frac{2\pi 100}{11}$, i.e., the time for traveling along a circular path around the hub from one spoke to the next. Again the fixed cost of each arc is its length plus 10. All instances in this class have 100 commodities.

References

Boland, Natashia, Mike Hewitt, Luke Marshall, and Martin Savelsbergh (2017). "The Continuous-Time Service Network Design Problem". In: *Operations Research* 65.5, pp. 1303–1321.

— (2019). "The price of discretizing time: a study in service network design". In: *EURO Journal on Transportation and Logistics*. Special Issue: Advances in vehicle routing and logistics optimization: exact methods 8.2, pp. 195–216.

Crainic, Teodor Gabriel and Mike Hewitt (2021). "Service Network Design". In: *Network Design with Applications to Transportation and Logistics*. Ed. by Teodor Gabriel Crainic, Michel Gendreau, and Bernard Gendron. Cham: Springer International Publishing, pp. 347–382.

Gurobi Optimization, LLC (2025). *Gurobi Optimizer Reference Manual*.

Hewitt, Mike (2019). “Enhanced Dynamic Discretization Discovery for the Continuous Time Load Plan Design Problem”. In: *Transportation Science* 53.6. Publisher: INFORMS, pp. 1731–1750.

Marshall, Luke, Natashia Boland, Martin Savelsbergh, and Mike Hewitt (2021). “Interval-Based Dynamic Discretization Discovery for Solving the Continuous-Time Service Network Design Problem”. In: *Transportation Science* 55.1. Publisher: INFORMS, pp. 29–51.

Shu, Shengnan, Zhou Xu, and Roberto Baldacci (2025). “New Dynamic Discretization Discovery Strategies for Continuous-Time Service Network Design”. In: *Optimiziation Online*, Preprint.

Van Dyk, Madison and Jochen Koenemann (2024). “Sparse dynamic discretization discovery via arc-dependent time discretizations”. In: *Computers & Operations Research* 169, p. 106715.