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

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

April 29, 2025

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 to solve 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 arc-based relaxation for this framework and adapt acceleration strategies from the literature to it. Our computational experiments demonstrate that this arc-based relaxation leads to significantly smaller integer programming models and overall faster solving times for a set of hard instances from the literature.

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. 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. 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 our relaxation is computationally beneficial. Specifically, our relaxation leads to integer programming models that are about a factor of five smaller for many instances than those obtained by previous approaches and can be solved significantly faster, leading to about a $45\,\%$ reduction in average solving times for a challenging set of instances from the literature.

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 and show how to refine it. In Section 5, we describe the algorithm that makes use of our relaxation to solve the overall problem. In Section 6, we describe the results of our computational

study that shows the efficacy of our algorithm. Finally, in Section 7, 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) introduced 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 proposed 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) showed 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 inter-

pretation 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 significantly 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 $\mathcal{G}^k = (\mathcal{V}^k, A^k)$ ($\mathcal{V}^k \subseteq \mathcal{V}, \mathcal{A}^k \subseteq \mathcal{A}$) 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 $c_a^k \geq 0$.

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 depends 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 a 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 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 minimal 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 commodities 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 commodities' flat-network without losing any feasible solutions. We now define for each commodity $k \in \mathcal{K}$ its 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., $A^k_{\mathrm{h}} = \{((v,t),(v,t+1)) \mid \forall v \in \mathcal{V}, \forall t \in [\underline{t}_{kv}, \bar{t}_{kv})\}$. 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:

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

$$\text{s.t.} \sum_{a \in \delta^+(v)} x_a - \sum_{a \in \delta^-(v)} x_a = \begin{cases} 1 & \text{if } v = (o_k, r_k) \\ -1 & \text{if } v = (d_k, \ell_k) \\ 0 & \text{else} \end{cases} \quad \forall k \in \mathcal{K}, \forall v \in V^k \tag{2}$$

$$\sum_{k \in \mathcal{K}:\ a \in A_{\mathrm{t}}^k} q_k x_a^k \leq u_a y_a \qquad \forall a \in A_{\mathrm{t}} \tag{3}$$

$$x_a^k \in \{0, 1\} \qquad \forall k \in \mathcal{K}, \forall a \in A^k \tag{4}$$

$$y_a \in \mathbb{N}_0 \qquad \forall a \in A_{\mathrm{t}} \tag{5}$$

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 commodities flat-path p^k and the dispatch time of that arc is $t^k_{(u,v)} = t$. 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 (2) ensures flow conservation, i.e., that every commodity flows along a path in its time-expanded network. Constraint (3) 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 proposed by Boland et al. (2017). As in their framework, we propose a relaxation of SND-RR, which can be formulated as an integer program on a time-expanded network. Solving the relaxation provides a lower bound on the optimal value of SND-RR. We also use their method that tries to obtain a feasible solution to SND-RR with the same value as the relaxation solution. If this succeeds, we have found an optimal solution. If not, our algorithm determines how to modify the relaxation 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 relaxation differently and in a way that results in significantly smaller integer programs that can be solved faster. Specifically, Boland et al. (2017) use a time discretization with time points for each node and create a copy of all outgoing arcs of each node for each time point. We use commodity-specific discretization of the node times and also give each arc an individual discretization.

In the following, we will first introduce our relaxed problem and show that every solution to SND-RR has a representation in this problem with the same or lower costs, i.e., that an optimal solution of the relaxed problem gives a lower bound on the optimal solution for SND-RR. Then, we propose a new integer program to solve our relaxed problem. We adapt some concepts first introduced by Marshall et al. (2021)

and extended by Shu et al. (2025) to our relaxation to determine if our relaxation solution can be converted into a feasible solution to SND-RR and if not, how the relaxation needs to be adjusted.

4.1 Relaxed Problem

Our relaxed problem is based on a discretization that consists of a partition of time into intervals, as was proposed by Marshall et al. (2021). They partition the time at each node into intervals and then created timed copies of arcs between any interval at the tail node and interval at the head node where a commodity dispatched in the tail node interval could arrive within the head node interval. We propose a different approach: each commodity receives its own interval partition of time at each node and instead of describing the exact times at which commodities are at a node, we only keep track of the intervals in which they are at a node. Similarly, each arc in the flat-network receives its own interval partition of time, and we only track which commodities are dispatched in an interval instead of considering their exact dispatch times. We assume that a commodity can be dispatched on an arc in a specific arc interval as long as it is at the tail node in or before the last node interval that overlaps the arc interval. This means we are overly optimistic about how late a commodity can arrive at a node and still be dispatched. Similarly, we assume a commodity dispatched on an arc in a specific arc interval arrives in the earliest head node interval in which a vehicle dispatched in the arc interval could arrive. This means that we are also overly optimistic about how early a commodity can arrive at a node if it is dispatched in a given interval. Together with the assumption that all commodities dispatched on an arc in the same interval can be consolidated, these assumptions are what makes this problem a relaxation of SND-RR.

We now formally introduce our time discretization and then use it to define the 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 commodities' 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 intervals $\mathcal{T}_{kv} = \{[t^{kv}_1, t^{kv}_2), \dots, [t^{kv}_{n_{kv}}, t^{kv}_{n_{kv}+1})\}$ that partition $[\underline{t}_{kv}, \bar{t}_{kv}]$, such that $t^{kv}_1 = \underline{t}_{kv}$ and $t^{kv}_{n_{kv}+1} = \bar{t}_{kv}+1$. These intervals represent possible times at which commodity k could be dispatched at this node. As an initial time discretization we can simply use $\mathcal{T}_{kv} = \{[\underline{t}_{kv}, \bar{t}_{kv}]\}$ 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 $\mathcal{T}_a = \{[h^a_1, h^a_2), \dots, [h^a_{n_a}, h^a_{n_a+1})\}$ a partition of $[H]$ into n_a intervals, such that $h^a_1 = 1$ and $h^a_{n_a+1} = H$. As an initial partition, we can simply 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 $p^k = (a^k_1, \dots, a^k_{n_k})$ 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 dispatch interval

indices $N^k = \{n_i^k\}_{i \in [n_k]}$ states the index of the interval $[h^{a_i}_{n_i^k}, h^{a_i}_{n_i^k+1}) \in \mathcal{T}_{a_i}$ in which a commodity is dispatched on the i-th arc in its flat-path. 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 dispatch 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 dispatch interval)**.** *For a commodity $k \in \mathcal{A}^k$ and arc $a = (u, v) \in \mathcal{A}^k$, we call a dispatch 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, a dispatch interval is feasible for a commodity if we can dispatch it at some point inside this interval and still arrive on time. We require that the dispatch interval for each arc in the flat-path for each commodity is feasible. 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. 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. We instead require that a commodity can only be dispatched in the interval in which it arrives at a node or a later interval. This begs the question in which node time interval a commodity arrives if it is dispatched in a given arc time interval. First, we determine the earliest possible arrival for a dispatch in that arc time interval. We then assume a commodity arrives at the head node in the node time interval that contains this time. As for the dispatch at the tail node, we assume a commodity can be dispatched in a given arc time interval if it is at the tail node in a node time interval that overlaps the arc time interval. More formally, we assume a commodity k can be dispatched on arc $a = (u, v) \in \mathcal{A}^k$ in time interval $[h^a_q, h^a_{q+1})$ if it is at node u in the node interval with the index $n^-_{kaq} = \max\{n \in [n_{ku}] \mid [h^a_q, h^a_{q+1}) \cap [t^{ku}_n, t^{ku}_{n+1}) \neq \emptyset\}$ and will arrive at node v in the node interval with the index $n^+_{kaq} = \min\{n \in [n_{kv}] \mid [h^a_q + \tau_a, h^a_{q+1} + \tau_a) \cap [t^{kv}_n, t^{kv}_{n+1}) \neq \emptyset\}$. We interchangeably call n^-_{kaq} and the associated interval the *relaxed dispatch interval* and n^+_{kaq} and the associated interval the *relaxed arrival 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 4 (Relaxed time consistent dispatch intervals)**.** *We call a set of dispatch intervals N^k for a flat-path p^k relaxed time consistent if the dispatch 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., $n^+_{ka_i n_i^k} \leq n^-_{ka_{i+1} n^k_{i+1}}$.*

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, where we assume that all commodities dispatched in the same time interval can be consolidated. For each arc $a \in \mathcal{A}$ and interval index $q \in [n_a]$ we collect the set of consolidated commodities $\bar{\kappa}(a, q) = \{k \in \mathcal{K} \mid \exists i \in$

$[n_k]\colon a_i^k = a, n_i^k = q\}$. With this, we can state the cost formally as

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

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.

To illustrate the concepts of our relaxed problem, we give an example. First, consider the SND-RR instance on the simple line network shown in Figure 1. The instance has three commodities for which the relevant parameters are given in Table 1a. All cost and demand parameters take on value 1 and both arcs have a capacity of 2. For each commodity, its flat-network consists of the nodes reachable from its source node. 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) and to consolidate commodities k_1 and k_3 on arc (v_2, v_3). The instance is constructed such that it is not possible to do both consolidations at once. Assume that we consolidate k_1 and k_2. Then they can not be dispatched before the release time $r_{k_2} = 2$ of k_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.

We now also give one specific time discretization $\mathcal{T}$ as an example and study the resulting relaxed problem. Table 1b and Table 1c display the intervals in this time discretization. Let us first consider which arc intervals are feasible for each commodity. For commodity k_1, all intervals on both arcs are feasible. As an example, it is possible to dispatch k_1 on arc (v_1, v_2) in the interval $[1, 2)$ since k_1 can be at v_1 at time 1 and if it is dispatched at this time it also arrives on time at v_2. The same is not true for commodity k_2, which only becomes available at $r_{k_2} = 2$ and as such can not be dispatched in the arc interval $[1, 2)$ on arc (v_1, v_2). The other arc interval $[2, 8)$ is feasible for k_2, since the commodity can be at v_1 at time 2 and if it is dispatched at this time it also arrives on time at v_2. Lastly, k_3 can only be dispatched in arc interval $[1, 4)$ on arc (v_2, v_3), as the arc interval $[4, 8)$ does not contain 3, the only time at which k_3 can be at v_2. We next give an example of the relaxed dispatch and arrival intervals. If commodity k_1 is dispatched on arc (v_2, v_3) in the arc interval $[4, 8)$, the relaxed dispatch interval would be $[4, 5]$ and the relaxed arrival time would be $[6, 8]$. The relaxed dispatch time represents that the commodity could be dispatched in the arc interval $[4, 8)$ if it arrives at node v_2 at time 5 or earlier. The relaxed arrival time represents that if the commodity is dispatched in this interval, it could certainly not arrive at node 3 before time point 6. Next, we demonstrate that this sparse discretization of time is already sufficient to reflect all the possible consolidations in the unrelaxed problem and thus solving this relaxed problem would be sufficient to obtain a tight bound on the solution value of the unrelaxed problem. Consider that k_1 and k_2 can only be consolidated in a solution

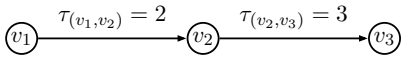

Figure 1: Flat-network $\mathcal{G}$ for the example instance

to the relaxed problem if k_1 is transported along arc (v_1, v_2) in interval $[2, 8)$, as this is the only feasible interval for k_2. In this case, the relaxed arrival interval of k_1 at node 2 is $[4, 5)$. Then, dispatching k_1 on arc (v_2, v_3) in interval $[1, 4)$ is not possible, since the relaxed dispatch interval for this arc interval is $[3, 4)$. Since $[1, 4)$ is the only feasible interval for k_3, it is not possible to consolidate k_1 and k_3. Lastly, we note again that the discretization of each arc is completely independent of that of any other arc or the time points for the commodities.

	o_k	d_k	r_k	ℓ_k
k_1	v_1	v_3	1	8
k_2	v_1	v_2	2	5
k_3	v_2	v_3	3	6

(a) Commodity parameters

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

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

a	$\mathcal{T}_a$
(v_1, v_2)	$\{[1,2),[2,8)\}$
(v_2, v_3)	$\{[1,4),[4,8)\}$

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

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

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 dispatch 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 $t_i^k \in [h^{a_i}_{n_i^k}, h^{a_i}_{n_i^k+1})$. These dispatch intervals are feasible because S is feasible and therefore $t_i^k \in [\underline{t}_{kv}, \bar{t}_{kv}]$. 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_{i+1}^k, this node interval either also contains t_{i+1}^k or is later than this time point. Now consider the relaxed arrival dispatch interval for arc a_i, which is the earliest node interval in $\mathcal{T}_{kv}$ that overlaps with the arc interval chosen before for a_i shifted by the travel time τ_{a_i}. Since that arc interval contains t_i^k, this node interval either contains $t_i^k + \tau_{a_i}$ or is earlier than this time point. Finally, since S is a feasible solution we know that $t_i^k + \tau_{a_i} \leq t_{i+1}^k$ 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. 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.

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 each time interval for each arc contains exactly one time point and each relevant time point exists at each node for each commodity, 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 dispatch 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 becomes tight. □

4.2 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 commodities' 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 keep 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 transport 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, for each arc $a \in \mathcal{A}$ we denote the corresponding set of transport arcs by $A^k_{\mathcal{T}a} = \{((u, q), (v, q')) \in A^k_{\mathcal{T}\mathrm{t}} \mid (u, v) = a\}$ (which is empty if $a \notin \mathcal{A}^k$).

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

$$\min \quad \sum_{k \in \mathcal{K}} \sum_{\hat{a} \in A^k_{\mathcal{T}\mathrm{t}}} c^k_{\hat{a}} x^k_{\hat{a}} + \sum_{a \in \mathcal{A}} \sum_{q=1}^{n_a} f_a y_{aq} \tag{6}$$

$$\text{s.t.} \quad \sum_{\hat{a} \in \delta^+(v)} x^k_{\hat{a}} - \sum_{\hat{a} \in \delta^-(v)} x^k_{\hat{a}} = \begin{cases} 1 & \text{if } v = (o_k, 1) \\ -1 & \text{if } v = (d_k, n_{kd_k}) \\ 0 & \text{else} \end{cases} \quad \forall k \in \mathcal{K}, \forall v \in V^k_{\mathcal{T}} \tag{7}$$

$$\sum_{k \in \mathcal{K}} \sum_{\hat{a} \in A^k_{\mathcal{T}a} \colon q_{\hat{a}} = q} q_k x^k_{\hat{a}} \leq u_a y_{aq} \qquad \forall a \in \mathcal{A}, \forall q \in [n_a] \tag{8}$$

$$\sum_{\bar{a} \in A^k_{\mathcal{T}}} \tau_{\hat{a}} x_{\hat{a}} \leq \ell_k - r_k \qquad \forall k \in \mathcal{K} \tag{9}$$

$$x^k_{\hat{a}} \in \{0, 1\} \qquad \forall k \in \mathcal{K}, \forall \hat{a} \in A^k_{\mathcal{T}} \tag{10}$$

$$y_{aq} \in \mathbb{N}_0 \qquad \forall a \in \mathcal{A}, \forall q \in [n_a] \tag{11}$$

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 dispatch 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 (7) ensures that the selected arcs for each commodity form a path in the time-expanded network. Constraint (8) 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 (9) ensures that the resulting flat-paths are k-feasible.

As an example, Figure 2 depicts the relaxed time-expanded networks for our previously introduced example instance on the flat-network depicted in Figure 1 and using the parameters in Table 1. Again we observe that these networks, even though they are quite small compared to the full time-expanded networks, already capture the main decision to be made in this instance: either commodity k_1 is trans-

ported along the arc (v_1, v_2) in the later interval $[2, 8)$ and can be consolidated with commodity k_2, or it is transported in the earlier interval $[1, 2)$ and can be consolidated on arc (v_2, v_3) with commodity k_3.

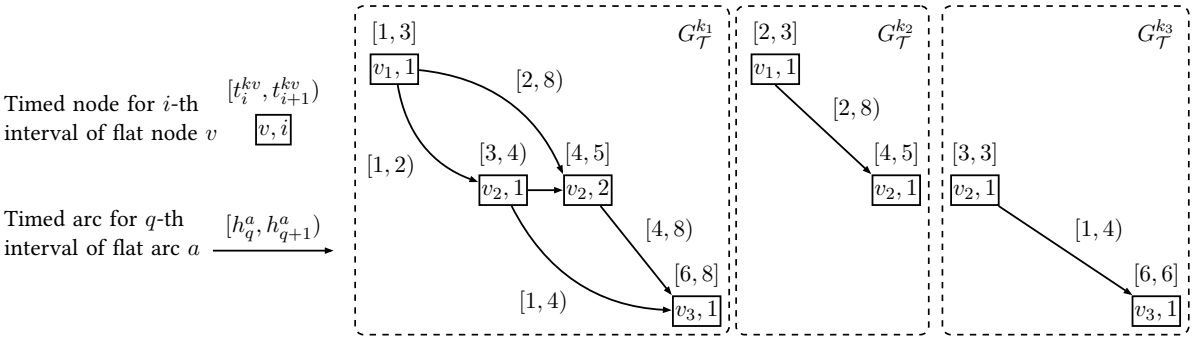

Figure 2: Relaxed time-expanded network for each commodity in the example instance

4.3 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 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 so that we do not obtain the same solution again. 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 5 (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 6 (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 node-dispatch 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 7 (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 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 commodities' 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 8 (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 $r_{k_1} + \sum_{a \in p} \rho_a + \tau_{v_n d_{k_n}}^{k_n} > \ell_{k_n}$.*

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 node-dispatch graphs of representable flat-solutions, the original too-long path can also no longer occur. This motivates the following definition:

Definition 9 (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 node dispatch 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. For some instances, our initial relaxation lead to node-dispatch graphs with many cycles and the label-propagating algorithm sometimes ran out of memory on these graphs, since they seem to contain many potential paths. We therefore use a heuristic modification of their algorithm that is not guaranteed to find all but only some minimal too-long paths. Specifically, at every node in the dispatch-node graph we store the largest time of any label seen so far at that node and whenever we would extend a label to a node, we do not do so if that label does not have a larger time. Note that this discarding step can lead to minimal too-long paths recurring in the node-dispatch graph over multiple iterations, but in preliminary experiments we found the overall algorithm to work just as well with this heuristic discarding while not running into memory problems. Importantly for the overall algorithm to terminate, if the node-dispatch graph contains any too-long paths, our modified algorithm will still find at least one of them.

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 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 dispatch 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 not representable. We formalize this idea in the following theorem.

Theorem 2. *Let p denote a too-long path in the dispatch 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 dispatch 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 $[h^{(v_1,v_2)}_{q_1}, h^{(v_1,v_2)}_{q_1+1}) \in \mathcal{T}'_{(v_1,v_2)}$ with $h^{(v_1,v_2)}_{q_1} = r_{k_1}$. 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 $[h^{(v_2,v_3)}_{q_2}, h^{(v_2,v_3)}_{q_2+1} \in \mathcal{T}'_{(v_2,v_3)})$ with $h^{(v_2,v_3)}_{q_2} = t_2$. 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 $h^{(v_1,v_2)}_{q'_1} + \tau_{(v_1,v_2)} \geq h^{(v_1,v_2)}_{q_1} + \tau_{(v_1,v_2)} = t_2$). So for the dispatch intervals for commodity k_2 to be relaxed time consistent (see Definition 4), 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 $[h^{(v_{n-1},v_n)}_{q'_{n-1}}, h^{(v_{n-1},v_n)}_{q'_{n-1}+1}) \in \mathcal{T}'_{(v_{n-1},v_n)}$ with $h^{(v_{n-1},v_n)}_{q'_{n-1}} \geq t_{n-1}$ 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 dispatch 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 dispatch intervals given this discretization. Therefore, the flat-solution is not representable with respect to this discretization. □

4.4 Strengthening the Initial Relaxation

The minimal initial discretization $\mathcal{T}$ necessary for our relaxation is to chose $\mathcal{T}_{kv} = \{[\underline{t}_{kv}, \bar{t}_{kv}]\}$ 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 latest 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 less dispatch 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 dispatch intervals for arc (u, v) as $\mathcal{T}_{(u,v)} = \{[0, t_1), [t_1, t_2), \ldots, [t_m, H)\}$.

5 Algorithm Implementation Details

We combine the previously introduced concepts into Algorithm 1 to solve SND-RR to optimality. This algorithm is conceptually identical to the one proposed by Boland et al. (2017). We first initialize our discretization as described in Section 4.4. Then we iteratively solve the relaxation R-SND-RR($\mathcal{T}$) and use the heuristic from Boland et al. (2017) (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 2 describes our refinement approach, which is based on adding time points according to Theorem 2. As noted before, since there are only finitely many flat-solutions representable in a discretization, and we only ever remove some by adding time points, the algorithm will eventually terminate. For practical purposes it is often enough to terminate once the best found primal solution is sufficiently close to optimal, which can be achieved by adjusting the termination criterion of the loop.

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

Require: An instance to SND-RR
Ensure: S is an optimal solution
Initialize discretization $\mathcal{T}$
$\overline{z} \leftarrow \infty, \underline{z} \leftarrow -\infty, S \leftarrow \emptyset$
while $\overline{z} > \underline{z}$ **do**
    Find an optimal solution W to R-SND-RR($\mathcal{T}$), $\underline{z} \leftarrow c(W)$
    Obtain solution S' to SND-RR from primal heuristic
    if $\overline{z} > c(S')$ **then**
        $\overline{z} \leftarrow c(S'), S \leftarrow S'$
    end if
    if $\overline{z} > \underline{z}$ **then**
        Refine discretization according to Algorithm 2
    end if
end while

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 node-dispatch graph $\mathscr{G}(P, C)$
Find a set $\mathscr{P}'$ of too-long paths
for all $p \in \mathscr{P}'$ **do**
    for all $((k_j, v_j), (k_{j+1}, v_{j+1})) \in p$ **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

6 Computational Study

Our computational study has two parts: In Section 6.1, we study the benefit of our arc-based compared to a node-based discretization on the instances proposed by Boland et al. (2017) for the problem without restricted routes, i.e., with $\mathcal{G}^k = \mathcal{G}$ for all $k \in \mathcal{K}$. In Section 6.2 we compare our algorithm to that of Van Dyk and Koenemann (2024) for instances with restricted routes that they introduced. These instances are specifically designed such that one would expect arc-based discretization to be beneficial.

We implemented our algorithm 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 to a gap between lower and upper bound of $1\,\%$ with a time limit of one hour on machines with a Xeon L5630 Quad Core 2.13 GHz processor and 16 GB or 128 GB memory. For the comparison in Section 6.2 against the algorithm proposed by Van Dyk and Koenemann (2024), we use their open-source implementation, available at `https://github.com/madisonvandyk/SND-RR`. We want to mention that we are very grateful that they provide their code and instances openly accessible and will follow their example upon acceptance of this manuscript. Their algorithm is also implemented in Python and uses Gurobi to solve the relaxations, so we use the same versions of Python and Gurobi as for our algorithm on the same machines to obtain comparable results.

Note that various algorithmic features from the literature could be added to our algorithm to further improve the performance, but we want to focus our computational study on the benefits of the new relaxation we introduce. We therefore implemented only the features described before, for which an adaption was necessary to work with our relaxation. Features that would be compatible with our approach include the various refinement strategies and the valid inequality of Marshall et al. (2021) as well as the improved primal heuristic and the enhanced refinement scheme for too-long path elimination by Shu et al. (2025). It would also be possible to update the relaxation models in place instead of constructing them from scratch after updating the discretization, but in our experiments the model construction time was negligible.

6.1 Sparse versus Dense Relaxation

To evaluate the benefit of our relaxation leading to sparser time-expanded networks compared to previous approaches, we compare our algorithm as described in this paper to the same algorithm but with the modification that whenever an any node (arc) time interval would be split, the interval at the (tail) node is split for all commodities that have this node in their subnetwork and for all arcs leaving this node. This also applies to the initial time points added to strengthen the first relaxation as described in Section 4.4. We call our algorithm ADDD (for arc-based dynamic discretization discovery) and the second algorithm DDD since it simulates the dynamic discretization discovery approaches from the literature. Furthermore, for this experiment we use the adaptive tolerances for solving the relaxation problem as introduced by Marshall et al. (2021) and also utilized by Shu et al. (2025). We note that without such an adaptive tolerance even solving the first relaxation to optimality takes up the entire time limit for some instances.

We compare ADDD and DDD on the set of 558 instances proposed by Boland et al. (2017). Note that in these instances all commodities share the same network and cost parameters, so $\mathcal{G}^k = \mathcal{G}$ and $c_a^k = q_k c_a$

Class	Algorithm	Gap (%)	Time (s)	# Iterations	Solved (%)
HC/HF	ADDD	1.34%	1198.53	9.9	75.1%
	DDD	1.58%	1312.92	3.9	74.0%
HC/LF	ADDD	0.90%	258.04	8.2	96.7%
	DDD	0.96%	456.27	3.8	94.5%
LC/HF	ADDD	0.22%	0.13	1.2	100.0%
	DDD	0.38%	0.25	1.3	100.0%
LC/LF	ADDD	0.49%	0.64	2.0	100.0%
	DDD	0.62%	1.67	1.8	100.0%

Table 2: Results by instance class and algorithm

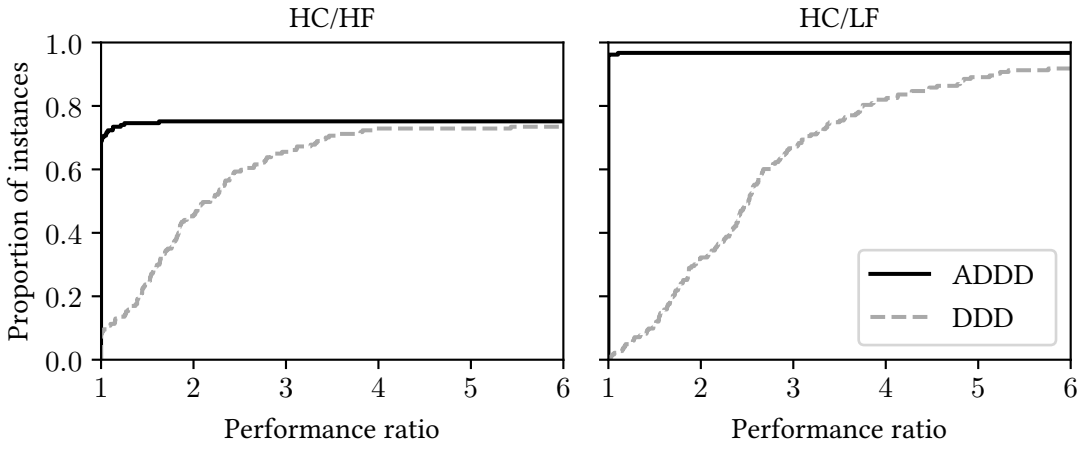

Figure 3: Performance profiles for both algorithms for the two challenging instance classes

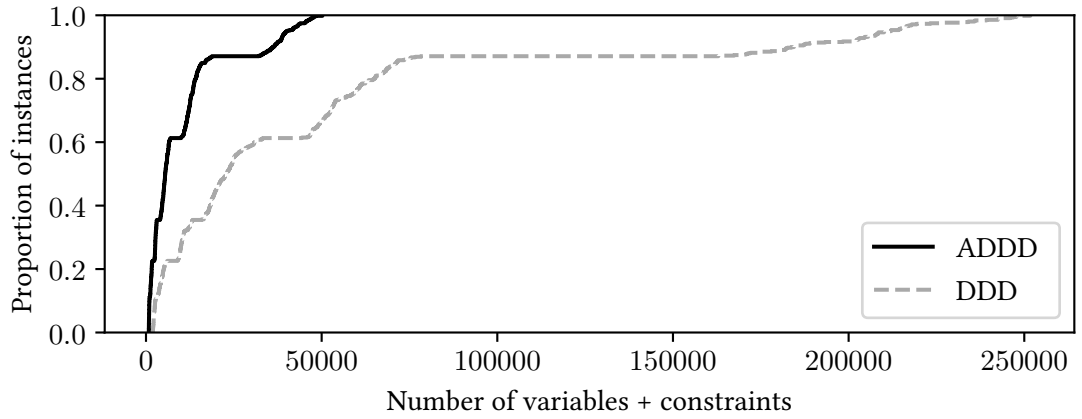

Figure 4: Distribution of model sizes of the last solved relaxation for each algorithm

for all $k \in \mathcal{K}$ and $a \in \mathcal{A}$. These instances were classified by Boland et al. (2019) and 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 $\min_{k\in\mathcal{K}}\{\ell_k - r_k + \tau^k_{o_k,d_k}\} < 227$ 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.

Table 2 displays the average gap between upper and lower bound at termination, solving time, number of iterations and the percentage of solved instances by algorithm and instance class. The arc-based discretization approach ADDD achieves significantly faster solving times and lower gaps for all four instance classes compared to DDD, even though the number of iterations is larger for all classes except LC/HF. The larger number of iterations is to be expected, since ADDD adds time points to less arcs, resulting in smaller but weaker relaxations. But these weaker relaxations are so much easier to solve that ADDD is overall faster than DDD. Despite this superior performance, the number of instances solved to a $1\,\%$ gap is almost identical for both approaches. To get a more detailed picture of the speed-ups achieved by ADDD, we plot the performance profiles for the two algorithms in Figure 3 for the two challenging instance classes HC/LF and HC/HF. This diagram shows the proportion of instances that each algorithm solves within a time that is at most the stated ratio larger than the time in which the fastest algorithm solves the instance. The lines level out at the proportion of instances that each algorithm solved within the time limit. We observe that for the most challenging class HC/HF ADDD solved about $70\,\%$ of instances faster than DDD, which only solved about $10\,\%$ of instances faster than ADDD (the remaining instances were not solved by either solver.) Additionally, ADDD was at least two times faster than DDD on about half the instances. The picture is similar for the instance class HC/LF but here ADD was faster on almost all instances. In Figure 4 we compare the model sizes between the two algorithms. It shows the proportion of instances for which each algorithm had less than a given number of variables and constraint in the last relaxation. We observe that the relaxations in ADDD are indeed significantly smaller. While the largest ADDD relaxation solved for any instance contained about 50,000 variables and constraints, the largest DDD relaxation had about five times as many variables and constraints. Overall, we conclude that our arc-based discretization results in smaller models that can be solved so much faster that most instances in this instance set can be solved faster even though more iterations are necessary.

6.2 Comparison with Van Dyk and Koenemann (2024)

We compare the performance of our algorithms ADDD and DDD to that by Van Dyk and Koenemann (2024), which we denote as VDK. Note that in their open-source implementation the relaxations are solved to a $1\,\%$ tolerance. We modified this tolerance such that for both their and our algorithm we solved the relaxations to the default Gurobi tolerance of $0.01\,\%$.

They created 576 instances specifically to be challenging for the node-based dynamic discretization discovery paradigms. Specifically, in (near-)optimal solutions to these instances, many nodes will have dispatches on some 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 becomes difficult to solve

quickly. They divide their instances into three classes of 192 instances each, which we call *designated path*, *critical times* and *hub-and-spoke*. 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.

We again report aggregate statistics by algorithm and instance class in Table 3. Note that for four instances in the designated path class one or both of our algorithms were not able to solve the first relaxation in the time limit. For these instances, we removed the results for all three algorithms to calculate the statistics in Table 3. From the results it is evident that both of our implementations are significantly faster and able to solve more instances compared to the implementation by Van Dyk and Koenemann (2024). This is most likely due to the algorithmic features of Shu et al. (2025) that we adapted to our relaxation, i.e., the stronger initial relaxation and the better refinement due to the removal of too-long paths. Due to our interval-based relaxation our algorithm can also start with a much sparser initial model similar to that of Marshall et al. (2021). Van Dyk and Koenemann (2024) base their algorithm on the paradigm of Boland et al. (2017) which already contains significantly more time points in the initial relaxation, specifically for each commodity the release time and deadline at the corresponding nodes. Therefore, the fairer comparison between ADDD and DDD is probably more telling on these instances. Interestingly, even though these instances were specifically constructed to be amenable approaches like the one of Van Dyk and Koenemann (2024) and ours, the performance of ADDD is actually worse than that of DDD. For all three classes, DDD solves more instances, in less iterations, faster and to a smaller remaining gap. We also plot the performance profiles by algorithm and instance class in Figure 5 (note the logarithmic scale.) While both ADDD and DDD are fastest on about half the instances each, DDD can consistently solve more instances within at most a small factor more time than the fastest algorithm. The exception are the designated path instances, for which ADDD can solve more instances within at most two times the time of the fastest solver. One could argue that this is also the clearest-cut case for arc-based discretization, since in these instances each commodity can be transported only along a fixed path and the discretization for each arc only affects the set of commodities that traverse along this arc and none of the others that could traverse along the tail node of the arc. Still, in the aggregate statistics DDD performs better overall even for the designated path instances. In Figure 6 we plot again the sizes of the last solved relaxation model by algorithm. We observe that the ADDD relaxations are significantly smaller than those of DDD, but seemingly for these instances the resulting faster solving times are not sufficient to make up for the weaker relaxation. Interestingly enough, the difference in size between the ADDD and DDD relaxations is not as large as between either and the size of the relaxations of VDK. We therefore posit that using an interval-based relaxation as done by Marshall et al. (2021) would already have lead to a strong reduction in model size and possibly solving time compared to the approach implemented by Van Dyk and Koenemann (2024). Overall, for this set of instances we can not report a benefit to using arc-based discretization.

Class	Algorithm	Gap (%)	Time (s)	# Iterations	Solved (%)
Critical times	ADDD	0.57%	139.57	4.6	98.4%
	DDD	0.41%	99.93	1.9	99.0%
	VDK	1.92%	963.40	15.0	89.6%
Designated path	ADDD	0.77%	303.64	5.9	96.3%
	DDD	0.60%	204.76	3.3	98.4%
	VDK	2.55%	1325.80	22.2	78.7%
Hub-and-spoke	ADDD	0.30%	100.93	5.1	98.4%
	DDD	0.18%	23.49	2.5	100.0%
	VDK	6.32%	1159.75	20.8	79.7%

Table 3: Results by instance class and algorithm

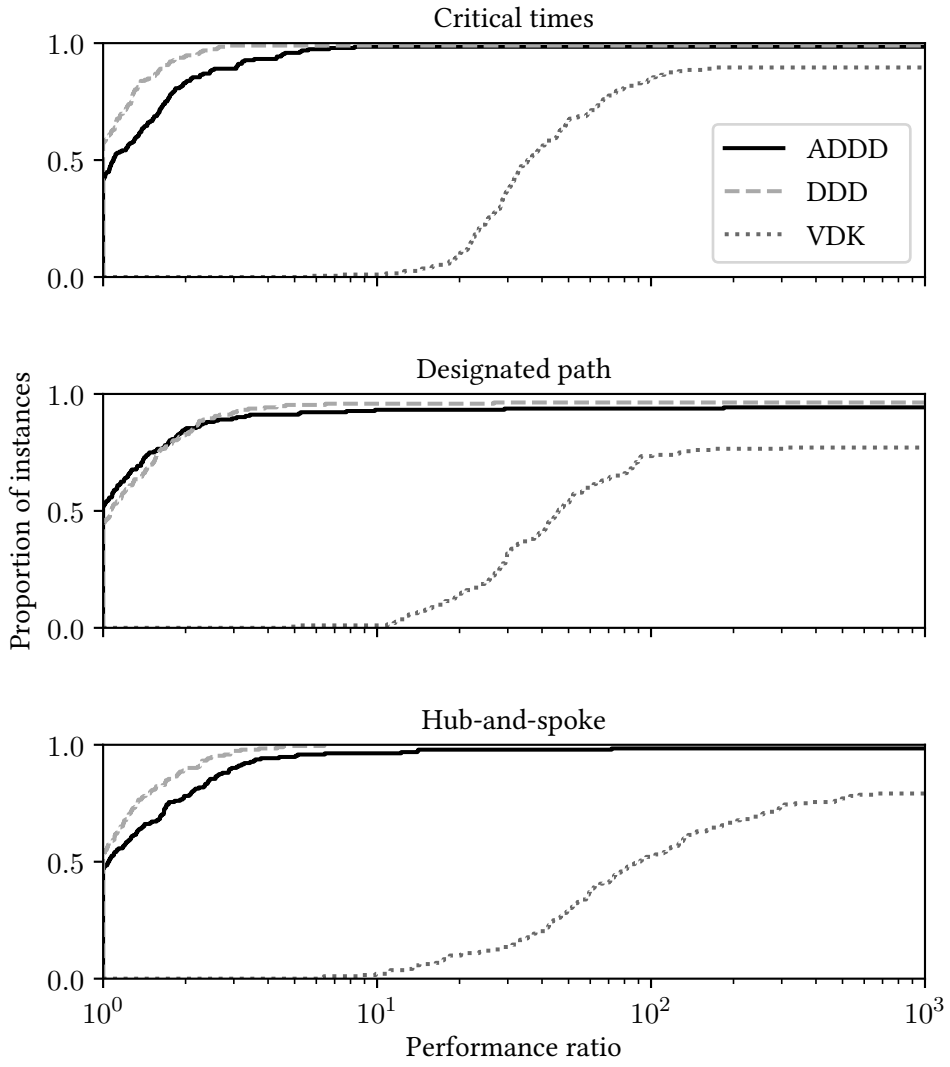

Figure 5: Performance profiles by instance class

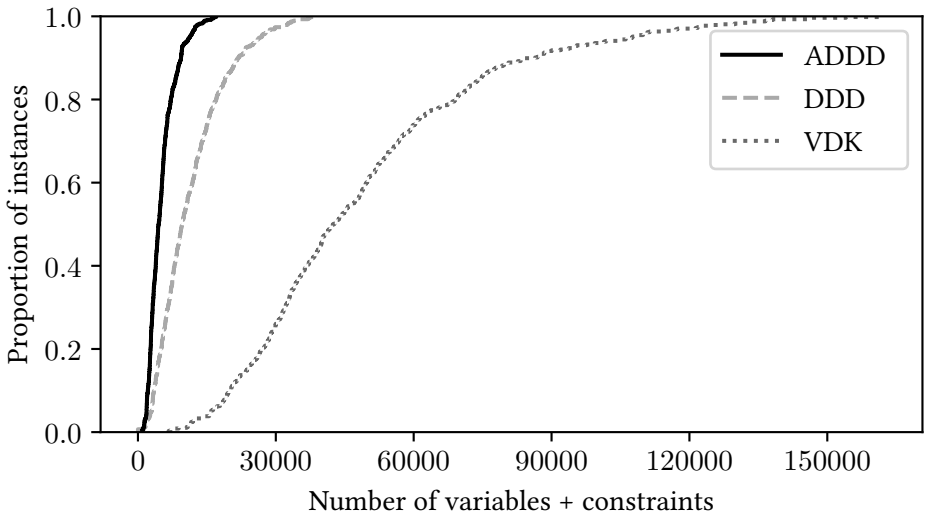

Figure 6: Distribution of model sizes of the last solved relaxation for each algorithm

7 Conclusions and Future Work

We have presented a novel arc-based relaxation for the continuous time service network design problem and have shown that various recent algorithmic improvements for node-based relaxation approaches can be adapted to this relaxation. 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.

A 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. Another interesting question is 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 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 relaxations.

Code and Data for our implementation and the used instance sets are submitted with this manuscript as a ZIP file and 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 proposed by Boland et al. (2017). 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:

$$\min \quad \sum_{a \in \mathcal{A}} \sum_{q=1}^{n_a} \sum_{\{k_1,k_2\} \subseteq \bar{\kappa}(a,q)} \delta_a^{\{k_1,k_2\}} \tag{12}$$

$$\text{s.t.} \quad \gamma_{a_i}^k + \tau_{a_i} \leq \gamma_{a_{i+1}}^k \qquad \forall k \in \mathcal{K}, \forall i \in [n_k - 1] \tag{13}$$

$$\gamma_{a_1}^k \geq r_k \qquad \forall k \in \mathcal{K} \tag{14}$$

$$\gamma_{a_{n_k}}^k + \tau_{a_{n_k}} \leq \ell_k \qquad \forall k \in \mathcal{K} \tag{15}$$

$$\delta_a^{\{k_1,k_2\}} \geq \gamma_a^{k_1} - \gamma_a^{k_2} \quad \forall a \in \mathcal{A}, \forall q \in [n_a], \forall \{k_1,k_2\} \subseteq \bar{\kappa}(a,q) \tag{16}$$

$$\delta_a^{\{k_1,k_2\}} \geq \gamma_a^{k_2} - \gamma_a^{k_1} \quad \forall a \in \mathcal{A}, \forall q \in [n_a], \forall \{k_1,k_2\} \subseteq \bar{\kappa}(a,q) \tag{17}$$

$$\gamma_a^k \geq 0 \qquad \forall k \in \mathcal{K}, \forall a \in p^k \tag{18}$$

$$\delta_a^{\{k_1,k_2\}} \geq 0 \qquad \forall a \in \mathcal{A}, \forall q \in [n_a], \forall \{k_1,k_2\} \subseteq \bar{\kappa}(a,q) \tag{19}$$

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 (13)–(15) 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.

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.