
Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

# The Continuous-Time Service Network Design Problem

Natashia Boland, Mike Hewitt, Luke Marshall, Martin Savelsbergh

To cite this article:
Natashia Boland, Mike Hewitt, Luke Marshall, Martin Savelsbergh (2017) The Continuous-Time Service Network Design Problem. Operations Research 65(5):1303-1321. https://doi.org/10.1287/opre.2017.1624

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

OPERATIONS RESEARCH
Vol. 65, No. 5, September–October 2017, pp. 1303–1321
ISSN 0030-364X (print), ISSN 1526-5463 (online)

# The Continuous-Time Service Network Design Problem

**Natashia Boland,[a] Mike Hewitt,[b] Luke Marshall,[a] Martin Savelsbergh[a]**

[a] H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332; [b] Quinlan School of Business, Loyola University Chicago, Chicago, Illinois 60660

**Contact:** natashia.boland@gmail.com (NB); mhewitt3@luc.edu, http://orcid.org/0000-0002-9786-677X (MH); luke.jonathon.marshall@gmail.com (LM); martin.savelsbergh@isye.gatech.edu (MS)

**Received:** January 5, 2015
**Revised:** June 10, 2016; October 28, 2016
**Accepted:** January 25, 2017
**Published Online in Articles in Advance:** July 25, 2017

**Subject Classifications:** networks/graph: multicommodity; programming: integer: algorithms; industries: transportation/shipping
**Area of Review:** Transportation

**https://doi.org/10.1287/opre.2017.1624**



**Abstract.** Consolidation carriers transport shipments that are small relative to trailer capacity. To be cost effective, the carrier must consolidate shipments, which requires coordinating their paths in both space and time; i.e., the carrier must solve a *service network design* problem. Most service network design models rely on discretization of time—i.e., instead of determining the exact time at which a dispatch should occur, the model determines a time interval during which a dispatch should occur. While the use of time discretization is widespread in service network design models, a fundamental question related to its use has never been answered: *Is it possible to produce an optimal continuous-time solution without explicitly modeling each point in time*? We answer this question in the affirmative. We develop an iterative refinement algorithm using partially time-expanded networks that solves continuous-time service network design problems. An extensive computational study demonstrates that the algorithm not only is of theoretical interest but also performs well in practice.

**Funding:** This material is based upon work supported by the National Science Foundation [Grant CMMI-1435625].

**Keywords:** service network design • time-expanded network • iterative refinement

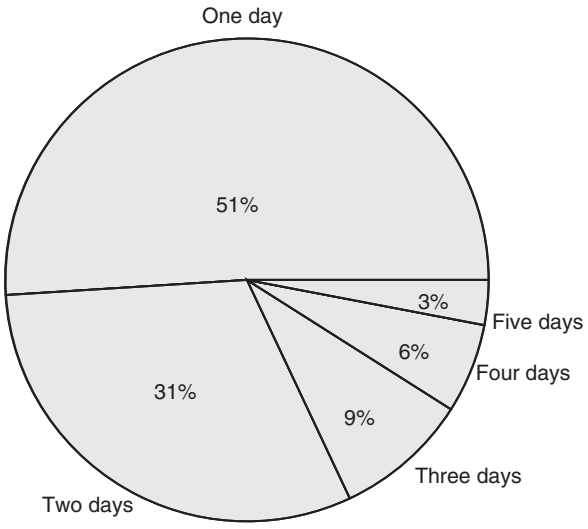
## 1. Introduction

Consolidation carriers transport shipments that are small relative to trailer capacity. Such shipments are vital to e-commerce. Consolidation carriers operate in (1) the less-than-truckload (LTL) freight transport sector, a sector with annual revenues in the United States alone of about $30 billion (Schulz 2014), and (2) the small package/parcel transport sector, a sector with much larger annual revenues, with one player alone (UPS) reporting $54 billion in revenue in 2012. Both LTL and small package carriers play a prominent role in the fulfillment of orders placed online (as well as other channels). Fast shipping times (and low cost) are critical to the success of the online sales channel, and e-tailers, such as Amazon.com, are continuously pushing the boundary, aiming for next-day and even same-day delivery. These trends result in increased pressure on LTL and small package transport companies to deliver in less time (without increasing their cost). This phenomenon is reflected in Figure 1, which shows the freight profile for a large LTL carrier by service level. It shows that over 80% of the carrier's shipments need to be delivered within two days.

To deliver goods in a cost-effective manner, a consolidation carrier must consolidate shipments, which requires coordinating the paths for different shipments in both space and time. The push toward rapid delivery reduces the margin for error in this coordination, which necessitates planning processes that accurately time dispatches. These planning processes have long been supported by solving the so-called *service network design* problem (Crainic 2000, Wieberneit 2008), which decides the paths for the shipments and the services (or resources) necessary to execute them. Service network design decisions for a consolidation carrier have both a geographic and a temporal component (e.g., dispatch a truck from Chicago, Illinois to Atlanta, Georgia at 9:05 P.M.). A common technique for modeling the temporal component is discretization; instead of deciding the exact time at which a dispatch should occur (e.g., 7:38 P.M.), the model decides on a time interval during which the dispatch should occur (e.g., between 6 P.M. and 8 P.M.).

When discretizing time, service network design problems can be formulated on a time-expanded network (Ford and Fulkerson 1958, 1962), in which a node encodes both a location and a time interval, and solutions prescribe dispatch time intervals for resources (trucks, drivers, etc.) and shipments. Service network design models calculate the costs for a set of dispatch decisions by estimating consolidation opportunities—i.e., by recognizing that prescribed dispatch time intervals for shipments allow travel together using the same resource. For example, shipments that should dispatch from the same origin node to the same destination node in the same dispatch time interval (say, from

**Figure 1.** Freight Profile for a Large LTL Carrier by Service




Louisville, Kentucky to Jackson, Michigan between 6 and 11 P.M.) are candidates for consolidation.

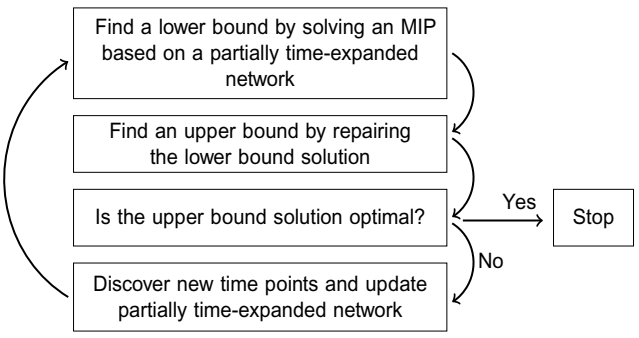
Clearly, the granularity of the time discretization has an impact on the candidate consolidation opportunities identified. At the same time, the granularity of the time discretization also impacts the computational tractability. With an hourly discretization of a week-long planning horizon, 168 timed copies of a node representing a location will be created. With a five-minute discretization of a week-long planning horizon, 2,016 timed copies of a node representing a location will be created. The latter discretization will likely yield a service network design problem that is much too large to fit into memory or solve in a reasonable amount of time. (In his introduction to network flows over time, Skutella 2009 also notes that the use of a discretization that includes each possibly relevant time point can be challenging computationally in many problem settings.)

While there is widespread use of discretizations of time and time-expanded networks in service network design models (Jarrah et al. 2009, Andersen et al. 2011, Erera et al. 2013, Crainic et al. 2016), we postulate that the fundamental question related to their use has not yet been answered: *Is it possible to produce an optimal "continuous"-time solution without explicitly modeling each point in time*? In this paper, we show that this question can be answered in the affirmative. We refer to a service network design problem in which time is modeled in such a way that it accurately captures the consolidation opportunities as a continuous-time service network design problem (CTSNDP). For all practical purposes, a time-expanded network based on a one-minute time discretization gives a CTSNDP. (Therefore, in the remainder, we will assume that the travel times and the times at which commodities become available and are due are specified as integers.) Furthermore, we call a time-expanded network that does not include all the time points a *partially time-expanded network*.

We develop a *dynamic discretization discovery algorithm* that manipulates partially time-expanded networks and allows the solution of a CTSNDP without ever creating a fully time-expanded network. The algorithm repeatedly solves a service network design problem defined on a partially time-expanded network and refines the partially time-expanded network based on an analysis of the solution obtained. Each partially time-expanded network is such that the resulting service network design problem is a relaxation of the CTSNDP. Furthermore, the solution to this relaxation can be converted to a feasible solution to the CTSNDP by solving an appropriately defined linear program. If the converted (or repaired) solution has the same cost, it will be optimal. If not, then the linear programming solution identifies time points that can be added to the partially time-expanded network to ensure that an improved solution is obtained in the next iteration. A flowchart of the high-level structure of the dynamic discretization discovery algorithm can be found in Figure 2. Thus, the dynamic discretization discovery algorithm solves a sequence of small mixed-integer programs (MIPs), rather than a single large MIP.

An extensive computational study shows the efficacy of the algorithm: instances with networks consisting of 30 nodes and 700 arcs, with 400 commodities, and a planning horizon of about 8 hours, which, when using a full-time discretization of 1 minute, leads to integer programs with more than 1,500,000 variables and close to 1,400,000 constraints, can often be solved to proven optimality in less than 30 minutes. Furthermore, the algorithm solves 97% of the several hundred instances in our test set and does so, on average, in less than 15 minutes. For those it does not solve, the algorithm produces, on average, a solution with a provable optimality gap of 2.5% or less in two hours. Computational results on a few instances derived from data from a

**Figure 2.** Flowchart of a Dynamic Discretization Discovery Algorithm



real-world less-than-truckload carrier are also promising with high-quality solutions produced in two hours or less.

To summarize, the main contributions of the paper are (1) the development of an algorithm for efficiently solving a continuous-time service network design problem and (2) demonstrating that an optimization problem defined on a time-expanded network can be solved to proven optimality without ever generating the complete time-expanded network. As time-expanded networks are frequently used to model transportation problems, we hope that the latter will stimulate other researchers to explore similar approaches in other contexts, and that that will ultimately result in an improved ability to solve practically relevant problems.

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 CTSNDP and discuss a property that (to some extent) motivates our approach. In Section 4, we introduce an iterative refinement algorithm for solving CTSNDP. In Section 5, we present and interpret the results of an extensive computational study of the algorithm's performance. Finally, in Section 6, we finish with conclusions and a discussion of future work.

## 2. Literature Review

The importance of incorporating temporal aspects into flow models has been recognized since their inception. Already in 1958, Ford and Fulkerson (1958) introduced the notion of *flows over time* (also called *dynamic flows*). They considered networks with transit times on the arcs, specifying the amount of time it takes for flow to travel from the tail of the arc to the head of the arc, and sought to send a maximum flow from a source to a sink within a given time horizon. They showed that a flow-over-time problem in a network with transit times can be converted to an equivalent standard (static) flow problem in a corresponding time-expanded network. The fundamental concept of an $s$-$t$-cut in a network was extended to an $s$-$t$-cut over time as well (Anderson et al. 1982, Anderson and Nash 1987). A comprehensive overview of this research area can be found in Skutella (2009).

Similarly, researchers have extended the minimum-cost $s$-$t$-flow problem to include a temporal component. Klinz and Woeginger (2004) show that, unlike the static problem, the minimum cost $s$-$t$-flow-over-time problem is weakly NP-hard. Fleischer and Skutella (2007) provide a polynomial-time approximation scheme for this (and other) problems (see also Fleischer and Skutella 2003).

A problem that is more closely related to the CTSNDP is the multicommodity-flow-over-time problem (Hall et al. 2007), in which demands must be routed from sources to sinks within a given time horizon. Hall et al. (2007) characterizes when this problem is weakly NP-hard. Topaloglu and Powell (2006) study a time-staged stochastic integer multicommodity flow problem.

The problems mentioned above assume a fixed time horizon is provided as part of the input. Researchers have also looked at flow models where the objective is to minimize the time it takes to send a given amount of flow. For example, Burkard et al. (1993) present an algorithm that solves the quickest $s$-$t$ flow problem in strongly polynomial time. Similarly, Hoppe and Tardos (2000) provide a polynomial-time algorithm to solve the quickest transshipment problem. Researchers have also studied the quickest multicommodity flow problem, for which Fleischer and Skutella (2007) provide an approximation algorithm with performance guarantee of 2. Researchers have also studied problems that seek flows with an earliest arrival property, in which the flows arriving at the destination at each time point are maximized (Gale 1958, Minieka 1973, Megiddo 1974, Jarvis and Ratliff 1982, Hoppe and Tardos 1994, Tjandra 2003, Baumann and Skutella 2006).

The CTSNDP adds an additional layer of complexity to the multicommodity flow over time problem by also incorporating network design decisions, which introduces a packing component to the problem. Kennington and Nicholson (2010) study a related problem—the uncapacitated fixed-charge network flow problem defined on a time-expanded network—but focus on choosing appropriate artificial capacities on the arcs to strengthen the linear programming relaxation of the natural integer programming formulation, and they only consider instances with relatively small time-expanded networks. Fischer and Helmberg (2014) develop methods for dynamically generating time-expanded networks but do so in the context of solving shortest path problems, and without having to make design decisions.

Powell et al. (1995) discuss the use of time-expanded networks in logistics planning models, noting that (at that time) most models create a time-expanded network by simply replicating the underlying network each time period.

Research on using partial time discretizations and dynamically adjusting a time discretization is scarce. Fleischer and Skutella (2007) use partial discretizations to generate (near-)optimal solutions to quickest-flow-over-time problems. (They refer to a partially time-expanded network as a *condensed* time-expanded network.) Wang and Regan (2009) analyze the convergence of a time window discretization method for the traveling salesman problem with time windows (TSPTW) introduced by Wang and Regan (2002) to obtain lower bounds on the optimal value. Their analysis shows that iteratively refining the discretization converges to the

optimal value. Dash et al. (2012) present an extended formulation for the TSPTW based on partitioning the time windows into subwindows called buckets (which can be thought of as discretizing the time window). They present cutting planes for this formulation that are computationally more effective than the ones known in the literature because they exploit the division of the time windows into buckets. They propose an iterative refinement scheme to determine appropriate partitions of the time windows. We provide more detail on the similarities and differences between our method and that of Dash et al. (2012) in Section 4.5.

Unlike the quickest-flow-over-time problems mentioned above, optimal solutions to CTSNDP need to strike a balance between the flow time from origin to destination and the capacity utilization of the arcs in the network, with flows waiting at the tail of an arc to be consolidated with other flows using the same arc. Furthermore, the continuous time flow models described above do not explicitly capture the delivery time constraints encountered in many transportation problems. To the best of our knowledge, we are the first to look at dynamically generating a (partially) time-expanded network for a problem that captures design decisions as well as flow time windows.

## 3. Problem Description

Let $\mathscr{D} = (\mathscr{N}, \mathscr{A})$ be a network with node set $\mathscr{N}$ and directed arc set $\mathscr{A}$. We will often refer to $\mathscr{D}$ as a "flat" network, as opposed to a time-expanded network, because the nodes in $\mathscr{N}$ model physical locations. Associated with each arc $a = (i, j) \in \mathscr{A}$ is a travel time $\tau_{ij} \in \mathbb{N}_{>0}$, a per-unit-of-flow cost $c_{ij} \in \mathbb{R}_{>0}$, a fixed cost $f_{ij} \in \mathbb{R}_{>0}$, and a capacity $u_{ij} \in \mathbb{N}_{>0}$. Let $\mathscr{K}$ denote a set of commodities, each of which has a single source $o_k \in \mathscr{N}$ (also referred to as the commodity's origin), a single sink $d_k \in \mathscr{N}$ (also referred to as the commodity's destination), and a quantity $q_k$ that must be routed along a single path from source to sink. Finally, let $e_k \in \mathbb{N}_{\geqslant 0}$ denote the time commodity $k$ becomes available at its origin and $l_k \in \mathbb{N}_{\geqslant 0}$ denote the time it is due at its destination. Without loss of generality, we assume that $\min_{k \in \mathscr{K}} e_k = 0$. The service network design problem (SNDP) seeks to determine paths for the commodities and the resources required to transport the commodities along these paths so as to minimize the total cost (i.e., fixed and flow costs) and ensure that time constraints on the commodities are respected. The SNDP is typically modeled using a time-expanded network. A time-expanded network $\mathscr{D}_{\mathscr{T}} = (\mathscr{N}_{\mathscr{T}}, \mathscr{H}_{\mathscr{T}} \cup \mathscr{A}_{\mathscr{T}})$ is derived from $\mathscr{D}$ and a set of time points $\mathscr{T} = \bigcup_{i \in \mathscr{N}} \mathscr{T}_i$ with $\mathscr{T}_i = \{t_1^i, \ldots, t_{n_i}^i\}$. The node set $\mathscr{N}_{\mathscr{T}}$ has a node $(i, t)$ for each $i \in \mathscr{N}$ and $t \in \mathscr{T}_i$. The arc set $\mathscr{H}_{\mathscr{T}}$ contains the arcs $((i, t_k^i), (i, t_{k+1}^i))$ for all $i \in \mathscr{N}$ and $k = 1, \ldots, n_i - 1$, known as holdover arcs, and the arc set $\mathscr{A}_{\mathscr{T}}$ contains arcs of the form $((i, t), (j, \bar{t}))$ where $(i, j) \in \mathscr{A}$, $t \in \mathscr{T}_i$, and $\bar{t} \in \mathscr{T}_j$. Note that $\mathscr{N}_{\mathscr{T}}$ uniquely determines $\mathscr{H}_{\mathscr{T}}$, and that, henceforth, we will, for any given $\mathscr{N}_{\mathscr{T}}$, make use of $\mathscr{H}_{\mathscr{T}}$ without explicit definition.

Arcs of the form $((i, t_k^i), (i, t_{k+1}^i))$ model the possibility of holding freight in location $i$, which may be advantageous if the freight can be consolidated with freight that arrives in location $i$ at a later point in time. We assume that freight can be held at a location at no cost. The algorithm to be presented in the next section relies critically on this assumption. To achieve freight consolidation and be profitable, many consolidation carriers own and operate their own network of terminals. For those carriers, holding freight at a terminal for a short amount of time, if necessary, incurs little or no additional costs, and this assumption is appropriate and not limiting. Carriers that operate out of third-party-owned terminals may incur additional costs when holding freight. However, those costs are typically significantly less than the transportation savings achieved by holding freight to achieve consolidation, and thus not modeling them is unlikely to lead to the wrong decision.

Arcs of the form $((i, t), (j, \bar{t}))$ model the possibility to dispatch freight from location $i$ at time $t$ to arrive at location $j$ at time $\bar{t}$. Note that an arc $((i, t), (j, \bar{t}))$ does *not* have to satisfy $\bar{t} - t = \tau_{ij}$. In fact, the flexibility to introduce arcs $((i, t), (j, \bar{t}))$ with a travel time that deviates from the actual travel time $\tau_{ij}$ of arc $(i, j)$ is an essential feature of time-expanded networks and provides a mechanism to control the size of the time-expanded network. Unfortunately, deviating from the actual travel times also introduces approximations that may have undesirable effects. Consider, for example, using a discretization of time into hours and modeling travel from Chicago, Illinois to Milwaukee, Wisconsin, which takes about 95 minutes if departure is at 6 P.M. When creating an arc ((Chicago, 6 P.M.), (Milwaukee, $\bar{t}$)), one must choose whether to set $\bar{t} = 7$ P.M. or $\bar{t} = 8$ P.M. Both choices have downsides. Setting $\bar{t} = 7$ P.M. implies that a service network design model using this time-expanded network perceives freight traveling on this arc as arriving in Milwaukee in time to consolidate with freight departing from Milwaukee at 7 P.M., which is not actually possible. However, setting $\bar{t} = 8$ P.M. implies that a service network design model using this time-expanded network perceives freight destined for Milwaukee and due there at 7:45 P.M. traveling on this arc as arriving in Milwaukee too late, which is not the case. The latter shows that not only travel times have to be mapped onto the time-expanded network but also the times that commodities are available at their origin and due at their destination. The typical mapping rounds up travel times, rounds up times that commodities are available, and rounds down times that commodities are due, because this ensures that any feasible solution

to the SNDP model on the time-expanded network can be converted to a true feasible solution (i.e., a feasible solution in real or continuous time).

A regular and fully time-expanded network $\mathscr{D}_{\mathscr{T}}^{\Delta}$ associated with $\mathscr{D}$ and discretization parameter $\Delta \in \mathbb{N}_{>0}$ has $\mathscr{T}_i = \{0, \Delta, 2\Delta, \ldots, K\Delta\}$ for all $i \in \mathscr{N}$ and for $K \in \mathbb{N}_{>0}$ with $\max_{k\in\mathscr{K}} l_k/\Delta \leqslant K < \max_{k\in\mathscr{K}} l_k/\Delta + 1$. Furthermore, for every arc $(i,j) \in \mathscr{A}$ and every node $(i,t) \in \mathscr{N}_{\mathscr{T}}$, there is an arc $((i,t),(j,t+\Delta\lceil \tau_{ij}/\Delta\rceil))$ in $\mathscr{A}_{\mathscr{T}}$ (unless $t+\Delta\lceil\tau_{ij}/\Delta\rceil) > K\Delta$). The networks $\mathscr{D}_{\mathscr{T}}^{\Delta}$ have become a popular tool in the design of approximation algorithms for flow-over-time problems, where they are known as condensed time-expanded networks (Fleischer and Skutella 2007, Groß et al. 2012, Groß and Skutella 2012).

We define SND($\mathscr{D}_{\mathscr{T}}$) to be the service network design problem defined over a time-expanded network $\mathscr{D}_{\mathscr{T}}$. Let $y_{ij}^{t\bar{t}}$ denote the number of times arc $(i,j)$ must be installed to accommodate dispatches from $i$ at time $t$ arriving at time $\bar{t}$ in $j$. (Because these variables capture resource movements such as, e.g., truck or trailer movements, we allow $y_{ij}^{t\bar{t}}$ to take on values greater than 1.) Let $x_{ij}^{kt\bar{t}}$ represent whether commodity $k \in \mathscr{K}$ travels from $i$ to $j$ departing at time $t$ to arrive at $\bar{t}$. Since we have assumed that a commodity must follow a single path from its origin to its destination, the variables $x_{ij}^{kt\bar{t}}$ are binary. For presentational convenience, we assume that the nodes $(o_k, e_k)$ and $(d_k, l_k)$ are in $\mathscr{N}_{\mathscr{T}}$ for all $k \in \mathscr{K}$. (Otherwise, the nodes $(o_k, t)$ with $t = \arg\min\{s \in \mathscr{T}_i \mid s > e_k\}$ and $(d_k, t')$ with $t' = \arg\max\{s \in \mathscr{T}_i \mid s < l_k\}$ can be used instead.)

Thus, SND($\mathscr{D}_{\mathscr{T}}$) seeks

$$z(\mathscr{D}_{\mathscr{T}}) = \min\left\{ \sum_{((i,t),(j,\bar{t}))\in\mathscr{A}_{\mathscr{T}}} f_{ij} y_{ij}^{t\bar{t}} + \sum_{k\in\mathscr{K}} \sum_{((i,t),(j,\bar{t}))\in\mathscr{A}_{\mathscr{T}}} c_{ij} q_k x_{ij}^{kt\bar{t}} \right\}$$

subject to

$$\sum_{((i,t),(j,\bar{t}))\in\mathscr{A}_{\mathscr{T}}\cup\mathscr{H}_{\mathscr{T}}} x_{ij}^{kt\bar{t}} - \sum_{((j,\bar{t}),(i,t))\in\mathscr{A}_{\mathscr{T}}\cup\mathscr{H}_{\mathscr{T}}} x_{ji}^{k\bar{t}t} = \begin{cases} 1 & (i,t) = (o_k, e_k), \\ -1 & (i,t) = (d_k, l_k), \\ 0 & \text{otherwise;} \end{cases} \quad \forall\, k \in \mathscr{K},\ (i,t) \in \mathscr{N}_{\mathscr{T}}, \tag{1}$$

$$\sum_{k\in\mathscr{K}} q_k x_{ij}^{kt\bar{t}} \leqslant u_{ij} y_{ij}^{t\bar{t}} \quad \forall\, ((i,t),(j,\bar{t})) \in \mathscr{A}_{\mathscr{T}}; \tag{2}$$

$$x_{ij}^{kt\bar{t}} \in \{0,1\} \quad \forall\, ((i,t),(j,\bar{t})) \in \mathscr{A}_{\mathscr{T}} \cup \mathscr{H}_{\mathscr{T}},\ k \in \mathscr{K}; \tag{3}$$

$$y_{ij}^{t\bar{t}} \in \mathbb{N}_{\geqslant 0} \quad \forall\, ((i,t),(j,\bar{t})) \in \mathscr{A}_{\mathscr{T}}. \tag{4}$$

That is, SND($\mathscr{D}_{\mathscr{T}}$) seeks to minimize the sum of fixed costs (the first term, which models transportation-related costs) and variable costs (the second term, which models handling-related costs). Note that we implicitly assume that holding freight at a location does not result in additional costs. Constraints (1) ensure that each commodity departs from its origin when it becomes available and arrives at its destination when it is due. Note the presence of holdover arcs allows a commodity to arrive early at its destination or depart late from its origin. Constraints (2) ensure that sufficient trailer capacity is available for the commodities that are sent from location $i$ at time $t$ to location $j$ at time $\bar{t}$. Constraints (3) and (4) define the variables and their domains. We denote an optimal solution to this problem by $(x(\mathscr{D}_{\mathscr{T}}), y(\mathscr{D}_{\mathscr{T}}))$ and its value with $z(\mathscr{D}_{\mathscr{T}})$.

Observe that when using a regular and fully time-expanded network $\mathscr{D}_{\mathscr{T}}^{\Delta}$, no approximations are introduced when $\tau_{ij}/\Delta$, $e_k/\Delta$, and $l_k/\Delta$ are naturally integer. In that case, a feasible solution to SND($\mathscr{D}_{\mathscr{T}}^{\Delta}$) is also feasible in continuous time and an optimal solution to SND($\mathscr{D}_{\mathscr{T}}^{\Delta}$) is also optimal in continuous time. Let $\hat{\Delta} = \mathrm{GCD}(\mathrm{GCD}_{(i,j)\in\mathscr{A}} \tau_{ij}, \mathrm{GCD}_{k\in\mathscr{K}} e_k, \mathrm{GCD}_{k\in\mathscr{K}} l_k)$, where GCD is the greatest common divisor. We define CTSNDP to be SND($\mathscr{D}_{\mathscr{T}}^{\hat{\Delta}}$). We use $\hat{\mathscr{T}} = \bigcup_{i\in\mathscr{N}} \hat{\mathscr{T}}_i$ to denote the time points included in $\mathscr{D}_{\mathscr{T}}^{\hat{\Delta}}$, $\mathscr{N}_{\mathscr{T}}^{\hat{\Delta}}$ to denote its nodes, and $\mathscr{A}_{\mathscr{T}}^{\hat{\Delta}} \cup \mathscr{H}_{\mathscr{T}}^{\hat{\Delta}}$ to denote its arcs.

The fully time-expanded network $\mathscr{D}_{\mathscr{T}}^{\hat{\Delta}}$ tends to be prohibitively large for practical instances. Furthermore, it typically contains nodes that are superfluous. For example, a node $(i,t) \in \mathscr{N}_{\mathscr{T}}^{\hat{\Delta}}$ that no commodity $k \in \mathscr{K}$ can visit (possibly because doing so would prevent the commodity reaching its destination on time) is superfluous. Therefore, a fundamental question is whether a smaller set of nodes $\mathscr{N}_{\mathscr{T}} \subset \mathscr{N}_{\mathscr{T}}^{\hat{\Delta}}$ and set of arcs $\mathscr{A}_{\mathscr{T}} \subset \mathscr{A}_{\mathscr{T}}^{\hat{\Delta}}$ can be determined a priori, such that solving SND($\mathscr{D}_{\mathscr{T}}$) yields an optimal solution to SND($\mathscr{D}_{\mathscr{T}}^{\hat{\Delta}}$). Proposition 1 shows how to construct one such set, $\mathscr{T}$.

**Proposition 1.** *To ensure that any optimal solution to* SND($\mathscr{D}_{\mathscr{T}}$) *is an optimal solution to CTSNDP, it is sufficient to include only time points in* $\mathscr{T}$ *that are determined by direct travel time paths starting at the origin of a commodity at the time that commodity becomes available; i.e., it is sufficient for* $\mathscr{T}$ *to consist only of time points of the form* $e_k$ *for some commodity* $k \in \mathscr{K}$ *or of the form* $e_k + \sum_{a\in P} \tau_a$ *for some commodity* $k \in \mathscr{K}$ *and some path* $P \subseteq \mathscr{A}$ *originating at* $o_k$.

**Proof.** Consider an optimal (continuous-time) solution. Shift all dispatch times to be as early as possible without changing any consolidations. This implies that each dispatch time at a node is now determined by the time a commodity originating at that node becomes available or by the arrival time of another commodity at the node. Suppose there is a dispatch time that is not at a time point of the form defined in the statement of the theorem. Choose the earliest such dispatch time $t$. Because this dispatch time $t$ cannot occur at the time a commodity becomes available, it must be determined by the arrival time of a commodity; i.e., there must be a commodity dispatched on some arc $a \in \mathscr{A}$ at time $t' = t - \tau_a$. However, because of the choice of $t$ and the assumption that all travel times are positive, it must be

that $t'$ is one of the time points defined in the statement of the theorem. But, since $t = t' + \tau_a$, $t$ itself must be a time point of the form defined in the statement of the theorem, which contradicts its definition. Q.E.D.

The set of time points defined in Proposition 1 may still be prohibitively large for practical instances and is thus not enough, by itself, to enable solution of CTSNDP. However, it motivates, in part, one of the main ideas underlying our approach to solving CTSNDP. We iteratively refine (expand) a set of time points $\mathcal{T}$, containing time points 0, $e_k$, and $l_k$ for $k \in \mathcal{K}$, and some time points of the form $t = e_k + \sum_{a \in P} \tau_a$ for some commodity $k \in K$ and some path $P \subseteq \mathcal{A}$ originating at $o_k$, until we can prove that the solution to SND($\mathcal{D}_{\mathcal{T}}$) for a carefully chosen arc set $\mathcal{A}_{\mathcal{T}}$ can be converted to an optimal solution to CTSNDP. The details of the approach are provided in the next section.

## 4. An Algorithm for Solving CTSNDP

Our approach for solving CTSNDP can be thought of as a dual ascent procedure, because it repeatedly solves and refines a relaxation of CTSNDP until the solution to the relaxation can be converted to a feasible solution to CTSNDP of the same cost (and hence it is an optimal solution). Specifically, the approach repeatedly solves an instance of SND($\mathcal{D}_{\mathcal{T}}$), where $\mathcal{D}_{\mathcal{T}}$ has carefully chosen time points, carefully chosen arcs, and carefully chosen arc travel times. Because $\mathcal{T}_i$ may only contain a small subset of the time points in $\hat{\mathcal{T}}_i$ for $i \in \mathcal{N}$ and the set of time points at different locations may differ (i.e., $\mathcal{T}_i$ may be different from $\mathcal{T}_j$ for $i \neq j$), we refer to the time-expanded network $\mathcal{D}_{\mathcal{T}}$ as a *partially time-expanded* network. In the description of our algorithm, we will often refer to a "timed copy" of arc $(i, j) \in \mathcal{A}$ at node $(i, t) \in \mathcal{N}_{\mathcal{T}}$, which will mean an arc of the form $((i, t), (j, \bar{t})) \in \mathcal{A}_{\mathcal{T}}$. These partially time-expanded networks will have four important properties, which we discuss next. In all that follows, we will, for notational convenience, but without loss of generality, assume that $\hat{\Delta} = 1$.

**Property 1.** *For all commodities $k \in \mathcal{K}$, the nodes $(o_k, e_k)$ and $(d_k, l_k)$ are in $\mathcal{N}_{\mathcal{T}}$.*

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

**Property 3.** *For every arc $a = (i, j) \in \mathcal{A}$ in the flat network, $\mathcal{D}$, and for every node $(i, t)$ in the partially time-expanded network, $\mathcal{D}_{\mathcal{T}} = (\mathcal{N}_{\mathcal{T}}, \mathcal{A}_{\mathcal{T}} \cup \mathcal{H}_{\mathcal{T}})$, there is a timed copy of $a$ in $\mathcal{A}_{\mathcal{T}}$ starting at $(i, t)$.*

Property 2 implies that we work with timed copies of an arc that are either of the correct length or are too short. This is illustrated in Figure 3, where we depict different timed copies of an arc $(j, k) \in \mathcal{A}$ that may be created by the algorithm. Observe that the lengths (or travel times) of the timed copies are different for the different dispatch times $t, t', t''$, and $t'''$, and that the travel time of a timed copy may even be negative (as is the case for dispatch time $t'''$).

**Definition 1.** If $\mathcal{D}_{\mathcal{T}}$ satisfies Properties 2 and 3, we say that $\mathcal{D}_{\mathcal{T}}$ has the *early arrival* property.

In what follows, it is useful to observe that any sequence of (non-holdover) arcs,

$$((i_1, t_1), (j_1, t_1')), ((i_2, t_2), (j_2, t_2')), \ldots, ((i_\eta, t_\eta), (j_\eta, t_\eta')),$$

in a time-expanded network, $\mathcal{D}_{\mathcal{T}} = (\mathcal{N}_{\mathcal{T}}, \mathcal{A}_{\mathcal{T}} \cup \mathcal{H}_{\mathcal{T}})$, where $((i_h, t_h), (j_h, t_h')) \in \mathcal{A}_{\mathcal{T}}$ for each $h = 1, \ldots, \eta$, induces a (unique) valid path in $\mathcal{D}_{\mathcal{T}}$, formed by the addition of appropriate holdover arcs, provided that $j_h = i_{h+1}$ and $t_h' \leqslant t_{h+1}$, for all $h = 1, \ldots, \eta - 1$.

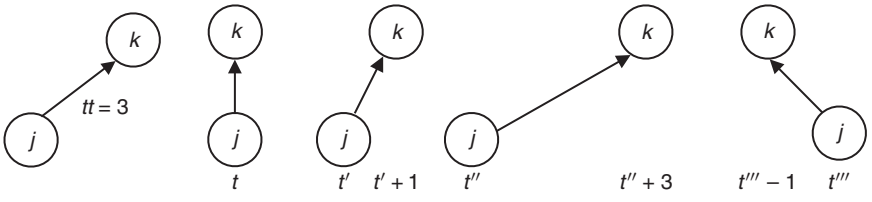
**Definition 2.** For any pair of nodes in the flat network, $j, j' \in \mathcal{N}$, we define the *distance* from $j$ to $j'$, denoted by $\underline{\tau}_{j,j'}$, to be the length of any shortest path, in the flat network, from $j \in \mathcal{N}$ to $j' \in \mathcal{N}$, with respect to the travel times, $\tau$.

**Lemma 1.** *Let $\mathcal{D}_{\mathcal{T}} = (\mathcal{N}_{\mathcal{T}}, \mathcal{A}_{\mathcal{T}})$ be a partially time-expanded network that satisfies Property 1 and has the early arrival property. Then for each commodity $k \in \mathcal{K}$ and each node in the flat network, $i \in \mathcal{N}$, there exists a timed node $(i, t) \in \mathcal{N}_{\mathcal{T}}$, with $t \leqslant e_k + \underline{\tau}_{o_k,i}$.*

**Proof.** Let $k \in \mathcal{K}$. The result holds trivially, by Property 1, if $i = o_k$, so consider $i \in \mathcal{N}$ with $i \neq o_k$. We proceed by induction on the number of arcs in the shortest path in the flat network from $o_k$ to $i$ with respect to $\tau$.

Suppose $i \in \mathcal{N}$ has a shortest path from $o_k$ given by the single arc $(o_k, i) \in \mathcal{A}$. Then it must be that $\underline{\tau}_{o_k,i} = \tau_{o_k,i}$. By Properties 1–3, there exists a $t \leqslant e_k + \tau_{o_k,i} = e_k + \underline{\tau}_{o_k,i}$ with $((o_k, e_k), (i, t)) \in \mathcal{A}_{\mathcal{T}}$, so it must be that $(i, t) \in \mathcal{N}_{\mathcal{T}}$, as required.

**Figure 3.** Travel Times of Timed Copies of $(j, k)$; Travel Times Do Not Exceed the Travel Time of Arc $(j, k)$




(a) Flat arc $(j, k)$ (b) Different timed copies of $(j, k)$ can model travel time differently

Now suppose that for all $i \in \mathcal{N}$ with a shortest path from $o_k$ having $\eta - 1$ arcs, with $\eta \geqslant 2$, there exists a timed node $(i,t) \in \mathcal{N}_{\mathcal{T}}$, with $t \leqslant e_k + \underline{\tau}_{o_k,i}$, and consider a node $j \in \mathcal{N}$ with a shortest path from $o_k$ having $\eta$ arcs—say, $P = ((j_0,j_1),(j_1,j_2),\ldots,(j_{\eta-1},j_\eta))$—where $j_0 = o_k$ and $j_\eta = j$ is such a path. By well-known properties of shortest paths, $((j_0,j_1),(j_1,j_2),\ldots,(j_{\eta-2},j_{\eta-1}))$ is a shortest path from $j_0 = o_k$ to $j_{\eta-1}$, so, by the inductive assumption, there exists $(j_{\eta-1},t) \in \mathcal{N}_{\mathcal{T}}$, with $t \leqslant e_k + \underline{\tau}_{o_k,j_{\eta-1}}$. By Property 3, there must exist $t'$ with $((j_{\eta-1},t),(j_\eta,t')) \in \mathcal{A}_{\mathcal{T}}$, and by Property 2, it must be that $t' \leqslant t + \tau_{j_{\eta-1},j_\eta}$, and hence

$$\begin{aligned} t' &\leqslant e_k + \underline{\tau}_{o_k,j_{\eta-1}} + \tau_{j_{\eta-1},j_\eta} \\ &= e_k + \sum_{h=1}^{\eta-1} \tau_{j_{h-1},j_h} + \tau_{j_{\eta-1},j_\eta} = e_k + \sum_{h=1}^{\eta} \tau_{j_{h-1},j_h} = e_k + \underline{\tau}_{o_k,j_\eta}. \end{aligned}$$

The result follows by induction. Q.E.D.

**Theorem 1.** *Let $\mathcal{D}_{\mathcal{T}}$ be a partially time-expanded network that satisfies Property 1 and has the early arrival property. Then* SND($\mathcal{D}_{\mathcal{T}}$) *is a relaxation of the CTSNDP.*

**Proof.** Consider an optimal solution $(\bar{x}(\mathcal{D}_{\mathcal{T}}^{\hat{\Delta}}), \bar{y}(\mathcal{D}_{\mathcal{T}}^{\hat{\Delta}}))$ to CTSNDP and let

$$\mathcal{A}^* = \{((i,t),(j,t+\tau_{ij})) \in \mathcal{A}_{\mathcal{T}}^{\hat{\Delta}} \mid \bar{y}_{ij}^{t,t+\tau_{ij}} > 0\}$$

(recalling the assumption that $\hat{\Delta} = 1$). Furthermore, let $\mathcal{K}_{((i,t),(j,t+\tau_{ij}))}$ represent the set of commodities dispatched on arc $((i,t),(j,t+\tau_{ij})) \in A^*$, in this optimal solution; i.e., let

$$\mathcal{K}_{((i,t),(j,t+\tau_{ij}))} = \{k \in \mathcal{K} \mid \bar{x}_{ij}^{k,t,t+\tau_{ij}} > 0\}.$$

In what follows, we will identify each arc $a \in \mathcal{A}^*$ with a unique arc in $\mu(a) \in \mathcal{A}_{\mathcal{T}}$ and construct $(x(\mathcal{D}_{\mathcal{T}}), y(\mathcal{D}_{\mathcal{T}}))$ so that the commodity flow represented by $x$ and trailer capacity represented by $y$, on each timed arc of the form $\mu(a)$, is exactly that of $\bar{x}$ and $\bar{y}$, respectively, on $a$, and so that $(x,y)$ is feasible for SND($\mathcal{D}_{\mathcal{T}}$) and has cost identical to the optimal value of CTSNDP.

Specifically, we define the mapping $\mu\colon A^* \to \mathcal{A}_{\mathcal{T}}$ as follows. For a given $a = ((i,t),(j,t+\tau_{ij})) \in A^*$, $\mathcal{K}_a \neq \emptyset$, and for all $k \in \mathcal{K}_a$, it must be that $e_k + \underline{\tau}_{o_k,i} \leqslant t$. Thus, by Lemma 1, there exists $t' \leqslant t$ with $(i,t') \in \mathcal{N}_{\mathcal{T}}$. Therefore $\rho_i(t)$, defined to be the latest time point at or before $t$ so that $(i,\rho_i(t)) \in \mathcal{N}_{\mathcal{T}}$ (i.e., $\rho_i(t) = \arg\max\{s \in \mathcal{T}_i \mid s \leqslant t\}$), is well defined. By Property 3, there must exist a $t'$ with $((i,\rho_i(t)),(j,t')) \in \mathcal{A}_{\mathcal{T}}$. Choose $\sigma(a)$ to be any such $t'$, and define $\mu(a) = ((i,\rho_i(t)),(j,\sigma(a)))$.

We may now define the trailer capacity on each timed arc $\tilde{a} = ((i,\tilde{t}),(j,\tilde{t}')) \in \mathcal{A}_{\mathcal{T}}$ by summing the trailer capacities in the optimal solution on all arcs that map to $\tilde{a}$ under $\mu$:

$$y_{ij}^{\tilde{t}\tilde{t}'} = \sum_{\substack{a=((i,t),(j,t+\tau_{ij}))\in A^*: \\ \mu(a)=\tilde{a}}} y_{ij}^{t,t+\tau_{ij}},$$

where the right-hand side is taken to be zero if no arcs in $A^*$ map to $\tilde{a}$, under $\mu$.

To construct the commodity flows in $\mathcal{D}_{\mathcal{T}}$, we will show that the sequence of (non-holdover) arcs, in each commodity's path in the optimal solution, maps, under $\mu$, to a sequence of arcs in $\mathcal{A}_{\mathcal{T}}$ that induce a valid path in $\mathcal{D}_{\mathcal{T}}$ from the commodity's origin node in $\mathcal{N}_{\mathcal{T}}$ to its destination in $\mathcal{N}_{\mathcal{T}}$.

For each commodity $k$, let $\bar{P}_{\mathcal{T}}^k$ denote the path from $(o_k,e_k)$ to $(d_k,l_k)$ in $\mathcal{D}_{\mathcal{T}}^{\hat{\Delta}}$ induced by the optimal commodity flow, $\bar{x}$; i.e.,

$$\begin{aligned} \bar{P}_{\mathcal{T}}^k &= \{((i,t),(j,t')) \in \mathcal{A}_{\mathcal{T}}^{\hat{\Delta}} \cup \mathcal{H}_{\mathcal{T}}^{\hat{\Delta}} \mid \bar{x}_{ij}^{k,t,t'} > 0\} \\ &= \{((i,t),(j,t')) \in A^* \cup \mathcal{H}_{\mathcal{T}}^{\hat{\Delta}} \mid \bar{x}_{ij}^{k,t,t'} = 1\}. \end{aligned}$$

Say $\bar{P}_{\mathcal{T}}^k$ is the path uniquely induced by the sequence of arcs $a^1,\ldots,a^\eta \in A^*$, together with holdover arcs linking from $(o_k,e_k)$ to the head of $a^1$ and from the tail of $a^\eta$ to $(d_k,l_k)$. Then we may write $a^h = ((i_h,t_h),(i_{h+1},t_{h+1}))$, with $t_{h+1} = t_h + \tau_{ij}$, for $h = 1,\ldots,\eta-1$.

We claim that the sequence of arcs $\mu(a^1),\ldots,\mu(a^\eta) \in \mathcal{A}_{\mathcal{T}}$ induces, with the addition of appropriate holdover arcs, a valid path in $\mathcal{D}_{\mathcal{T}}$ from $(o_k,e_k)$ to $(d_k,l_k)$. To prove this claim, observe that for any $h \in \{1,\ldots,\eta-1\}$, we have $\mu(a^h) = ((i_h,\rho_{i_h}(t_h)),(i_{h+1},\sigma(a^h)))$, where $\rho_{i_h}(t_h) = \arg\max\{s \in \mathcal{T}_{i_h} \mid s \leqslant t_h\}$, and so $\rho_{i_h}(t_h) \leqslant t_h$, and, by Property 2, $\sigma(a^h) \leqslant \rho_{i_h}(t_h) + \tau_{ij}$. Thus, $\sigma(a^h) \leqslant t_h + \tau_{ij} = t_{h+1}$. Recall that, by definition, $\sigma(a^h) \in \mathcal{T}_{i_{h+1}}$, and so it must be that $\rho_{i_{h+1}}(t_{h+1}) \geqslant \sigma(a^h)$. This shows that $\mu(a^1),\ldots,\mu(a^\eta)$ induces a valid path in $\mathcal{D}_{\mathcal{T}}$ from $(o_k,\rho_{o_k}(t_1))$ to $(d_k,\sigma(a^\eta))$. Now it must be that $\rho_{o_k}(t_1) \geqslant e_k$, since $t_1 \geqslant e_k$, and it must be that $\sigma(a^\eta) \leqslant l_k$ since $\sigma(a^\eta) \leqslant t_{\eta+1} \leqslant l_k$. Thus by including appropriate holdover arcs at $o_k$ and $d_k$, our claim follows, and we set $x_{ij}^{k,t,t'} = 1$ for all arcs $((i,t),(j,t')) \in \mathcal{A}_{\mathcal{T}} \cup \mathcal{H}_{\mathcal{T}}$ that are in the resulting path in $\mathcal{D}_{\mathcal{T}}$ from $(o_k,e_k)$ to $(d_k,l_k)$.

Now, it is not hard to see that solution $(x,y)$ constructed in this way is feasible for SND($\mathcal{D}_{\mathcal{T}}$), and it replicates solution $(\bar{x},\bar{y})$ to CTSNDP in the sense that the commodities flow along the same paths (in the flat network) and the same consolidations occur. Hence the two solutions have identical objective function value. Thus SND($\mathcal{D}_{\mathcal{T}}$) is a relaxation of the CTSNDP. Q.E.D.

Note that the proof, and thus the theorem, relies on the fact that holding freight at a location does not result in additional cost.

The following lemma regarding partially time-expanded networks with the early arrival property will be useful when we refine a partially time-expanded network during the course of our algorithm. We omit its proof, since it follows immediately from the definitions of Properties 2 and 3.

**Lemma 2.** *If a partially time-expanded network $\mathcal{D}_{\mathcal{T}}$ has the early arrival property, $((i,t),(j,t')) \in \mathcal{A}_{\mathcal{T}}$, and $(j,t'') \in \mathcal{N}_{\mathcal{T}}$*

*with $t'' \leqslant t + \tau_{ij}$, then the partially time-expanded network in which arc $((i,t),(j,t'))$ is replaced with arc $((i,t),(j,t''))$ will also have the early arrival property.*

There are many partially time-expanded networks $\mathscr{D}_{\mathscr{T}}$ that satisfy Properties 1–3. We restrict ourselves to partially time-expanded networks with arc sets $\mathscr{A}_{\mathscr{T}}$ that satisfy one additional property.

**Property 4.** *If arc $((i,t),(j,t')) \in \mathscr{A}_{\mathscr{T}}$, then there does not exist a node $(j,t'') \in \mathscr{N}_{\mathscr{T}}$ with $t' < t'' \leqslant t + \tau_{ij}$.*

**Definition 3.** If $\mathscr{D}_{\mathscr{T}}$ satisfies Property 4, we say that $\mathscr{D}_{\mathscr{T}}$ has the *longest-feasible-arc* property.

Observe that, for a given $\mathscr{T}$, (and $\mathscr{N}_{\mathscr{T}}$), there is a unique set of timed arcs that satisfy both the early arrival and the longest-feasible-arc properties. To see this, first note that if $((i,t),(j,t'))$ and $((i,t),(j,t''))$ are both in $\mathscr{A}_{\mathscr{T}}$ for some $t' \neq t''$, where, without loss of generality, $t' < t''$, then $t' < t'' \leqslant t + \tau_{ij}$ by Property 2, and the longest-feasible-arc property fails. Thus for each $(i,t) \in \mathscr{N}_{\mathscr{T}}$ and each $(i,j) \in \mathscr{A}$, there can be at most one arc of the form $((i,t),(j,t'))$ in $\mathscr{A}_{\mathscr{T}}$ satisfying both properties. For $\mathscr{A}_{\mathscr{T}}$ satisfying Property 3 there must be at least one such arc. Hence, if $\mathscr{A}_{\mathscr{T}}$ satisfies both properties, there is exactly one arc of the form $((i,t),(j,t'))$ in $\mathscr{A}_{\mathscr{T}}$ for each $(i,t) \in \mathscr{N}_{\mathscr{T}}$ and $(i,j) \in \mathscr{A}$. By Properties 2 and 4, it must be that $t' = \arg\max\{s \mid s \leqslant t + \tau_{ij}, (j,s) \in \mathscr{N}_{\mathscr{T}}\}$.

We restrict ourselves to arc sets with the longest-feasible-arc property as a result of the following theorem.

**Theorem 2.** *For a fixed $\mathscr{T}$, (and $\mathscr{N}_{\mathscr{T}}$), among the partially time-expanded networks $\mathscr{D}_{\mathscr{T}}$ with the early arrival property, the one with the longest-feasible-arc property induces an instance of* SND($\mathscr{D}_{\mathscr{T}}$) *with the largest optimal objective function value.*

**Proof.** Consider a partially time-expanded network $\mathscr{D}^{LF}_{\mathscr{T}} = (\mathscr{N}_{\mathscr{T}}, \mathscr{A}^{LF}_{\mathscr{T}} \cup \mathscr{H}_{\mathscr{T}})$ with arc set $\mathscr{A}^{LF}_{\mathscr{T}}$ that has the longest-feasible-arc property and a partially time-expanded network $\mathscr{D}'_{\mathscr{T}} = (\mathscr{N}_{\mathscr{T}}, \mathscr{A}'_{\mathscr{T}} \cup \mathscr{H}_{\mathscr{T}})$ with arc set $\mathscr{A}'_{\mathscr{T}}$ that does not. Assume that both networks have the early arrival property. We will show that any solution to SND($\mathscr{D}^{LF}_{\mathscr{T}}$) can be converted to a solution to SND($\mathscr{D}'_{\mathscr{T}}$) of equal value. Thus, the optimal objective function value of SND($\mathscr{D}'_{\mathscr{T}}$) can be no greater than that of SND($\mathscr{D}^{LF}_{\mathscr{T}}$).

Consider a solution $(x(\mathscr{D}^{LF}_{\mathscr{T}}), y(\mathscr{D}^{LF}_{\mathscr{T}}))$ to the problem SND($\mathscr{D}^{LF}_{\mathscr{T}}$) and an arc $((i,t),(j,t')) \in \mathscr{A}^{LF}_{\mathscr{T}}$ such that $y^{tt'}_{ij}(\mathscr{D}^{LF}_{\mathscr{T}}) > 0$ and all arcs of the form $((i,t),(j,t'')) \in \mathscr{A}'_{\mathscr{T}}$ have $t'' < t'$. If no such arc exists, then the solution is clearly feasible for SND($\mathscr{D}'_{\mathscr{T}}$). Thus, suppose such an arc exists.

Because both networks are defined on the same node set $\mathscr{N}_{\mathscr{T}}$, the path from $(j,t'')$ to $(j,t')$ exists in $(\mathscr{N}_{\mathscr{T}}, \mathscr{H}_{\mathscr{T}})$. Consequently, we can adapt the solution $(x(\mathscr{D}^{LF}_{\mathscr{T}}), y(\mathscr{D}^{LF}_{\mathscr{T}}))$ for this arc to one for the SND($\mathscr{D}'_{\mathscr{T}}$) by assigning $y^{tt''}_{ij}(\mathscr{D}'_{\mathscr{T}}) = y^{tt'}_{ij}(\mathscr{D}^{LF}_{\mathscr{T}})$ and routing the corresponding commodity flows on the path formed by concatenating arc $((i,t),(j,t''))$ with the path from $(j,t'')$ to $(j,t')$. Note that the cost of this change is 0, because we have assumed that there is no cost associated with using the holdover arcs. Because this change leaves any commodities that traveled on the arc $((i,t),(j,t'))$ in the solution to SND($\mathscr{D}^{LF}_{\mathscr{T}}$) at the same node $(j,t')$, we can repeat this process one arc at a time and are left with a solution to the SND($\mathscr{D}'_{\mathscr{T}}$) of equal value. Q.E.D.

Theorem 1, and to a lesser extent Theorem 2, provide the basis for our iterative-refinement algorithm for solving CTSNDP; Algorithm 1 presents a high-level overview.

**Algorithm 1** (Solve-CTSNDP)
**Require:** Flat network $\mathscr{D} = (\mathscr{N}, \mathscr{A})$, commodity set $\mathscr{K}$
Create a partially time-expanded network $\mathscr{D}_{\mathscr{T}}$ satisfying Properties 1–4
**while** not solved **do**
  Solve SND($\mathscr{D}_{\mathscr{T}}$)
  Determine whether the solution to SND($\mathscr{D}_{\mathscr{T}}$) can be converted to a feasible solution to CTSNDP with the same cost
  **if** it can be converted **then**
    Stop. The converted solution is optimal for CTSNDP.
  **else**
    The solution to SND($\mathscr{D}_{\mathscr{T}}$) must use at least one arc that is "too short." Refine the partially time-expanded network $\mathscr{D}_{\mathscr{T}}$ by correcting the length of at least one such arc, in the process adding at least one new time point to $\mathscr{T}_i$ for some $i \in \mathscr{N}$.
  **end if**
**end while**.

Before discussing the various components of the algorithm in more detail, we prove the following result.

**Theorem 3.** Solve-CTSNDP *terminates with an optimal solution.*

**Proof.** The algorithm terminates when the optimal solution to SND($\mathscr{D}_{\mathscr{T}}$) can be converted to a solution of CTSNDP with the same cost. Because SND($\mathscr{D}_{\mathscr{T}}$) is a relaxation of CTSNDP (Theorem 1), the converted solution must be an optimal solution to CTSNDP.

Furthermore, at every iteration in which Solve-CTSNDP does not terminate, the length of at least one arc $a \in \mathscr{A}_{\mathscr{T}}$ is increased to its correct length. Because there are a finite number of time points and arcs, at some iteration all arcs in $\mathscr{A}_{\mathscr{T}}$ must have travel times that correspond to the actual travel time of the corresponding arc in the flat network, in which case the solution to SND($\mathscr{D}_{\mathscr{T}}$) is a solution to CTSNDP and the algorithm terminates. Q.E.D.

Because arcs in $\mathcal{A}_{\mathcal{T}}$ can be too short, it is possible that a solution to SND($\mathcal{D}_{\mathcal{T}}$) contains a path for a commodity $k \in \mathcal{K}$ that is too long; i.e., its actual length or duration exceeds the available time $l_k - e_k$. We avoid such solutions by adding valid inequalities to SND($\mathcal{D}_{\mathcal{T}}$). More specifically, we prevent paths that are too long by adding the following inequality to SND($\mathcal{D}_{\mathcal{T}}$):

$$\sum_{((i,t),(j,t'))\in\mathcal{A}_{\mathcal{T}}} \tau_{ij} x_{ij}^{k\tau'} \leqslant l_k - e_k. \tag{5}$$

### 4.1. Creating an Initial Partially Time-Expanded Network

The initial partially time-expanded network consists of nodes $(o_k, e_k)$ and $(d_k, l_k)$ for all $k \in \mathcal{K}$ and $(u, 0)$ for all $u \in \mathcal{N}$. For each node $(i,t) \in \mathcal{N}_{\mathcal{T}}$ and arc $(i,j) \in \mathcal{A}$, we find the node $(j,t')$ with the largest $t'$ such that $t' \leqslant t + \tau_{ij}$ and add arc $((i,t),(j,t'))$ to $\mathcal{A}_{\mathcal{T}}$. Note that because $\mathcal{N}_{\mathcal{T}}$ includes nodes $(u,0)$ for $u \in \mathcal{N}$, it is always possible to find such a node $(j,t')$. (Note, too, that we may have $t' < t$, in which case the arc travels backward in time.) Finally, for all nodes $(i,t)$ and $(i,t')$ such that $t'$ is the smallest time point with $t' > t$, we add arc $((i,t),(i,t'))$ to $\mathcal{H}_{\mathcal{T}}$. It is not hard to see that this partially time-expanded network satisfies Properties 1–4. For a detailed description of Create-initial, see Algorithm 2.

**Algorithm 2** (Create-initial)

**Require:** Directed network $\mathcal{D} = (\mathcal{N}, \mathcal{A})$, commodity set $\mathcal{K}$

1: **for all** $k \in \mathcal{K}$ **do**
2: Add node $(o_k, e_k)$ to $\mathcal{N}_{\mathcal{T}}$
3: Add node $(d_k, l_k)$ to $\mathcal{N}_{\mathcal{T}}$
4: **end for**
5: **for all** $u \in \mathcal{N}$ **do**
6: Add node $(u,0)$ to $\mathcal{N}_{\mathcal{T}}$
7: **end for**
8: **for all** $(i,t) \in \mathcal{N}_{\mathcal{T}}$ **do**
9: **for all** $(i,j) \in \mathcal{A}$ **do**
10: Find largest $t'$ such that $(j,t') \in \mathcal{N}_{\mathcal{T}}$ and $t' \leqslant t + \tau_{ij}$ and add arc $((i,t),(j,t'))$ to $\mathcal{A}_{\mathcal{T}}$
11: **end for**
12: Find smallest $t'$ such that $(i,t') \in \mathcal{N}_{\mathcal{T}}$ and $t' > t$ and add arc $((i,t),(i,t'))$ to $\mathcal{H}_{\mathcal{T}}$
13: **end for**.

### 4.2. Refining a Partially Time-Expanded Network

It is necessary to refine $\mathcal{D}_{\mathcal{T}}$ when the solution to SND($\mathcal{D}_{\mathcal{T}}$) cannot be converted to a feasible solution to CTSNDP with the same cost, which can happen when an arc in $\mathcal{A}_{\mathcal{T}}$ is "too short." When refining $\mathcal{D}_{\mathcal{T}}$, we ensure that (1) the length of at least one arc that is too short is corrected and (2) the resulting partially time-expanded network again satisfies Properties 1–4.

More specifically, when we lengthen an arc $((i,t),(j,t'))$ that is too short (i.e., $t' < t + \tau_{ij}$), we replace it with the arc $((i,t),(j,t+\tau_{ij}))$. Because $\mathcal{D}_{\mathcal{T}}$ has the longest-feasible-arc property, node $(j,t+\tau_{ij})$ was not in $\mathcal{N}_{\mathcal{T}}$ and will have to be added to $\mathcal{N}_{\mathcal{T}}$. Lengthening arc $((i,t),(j,t'))$ to $((i,t),(j,t+t_{ij}))$ is a two-step process based on the following two lemmas. Details of the two steps are provided in Algorithms 4 and 5, respectively, and applied in sequence in Algorithm 3.

**Lemma 3.** *If a time-expanded network $\mathcal{D}_{\mathcal{T}}$ has the early arrival property, and* (1) *a new time point $t^i_{\text{new}}$ with $t^i_k < t^i_{\text{new}} < t^i_{k+1}$ is added to $\mathcal{T}_i = \{t^i_1, \ldots, t^i_{n_i}\}$,* (2) *a new node $(i, t^i_{\text{new}})$ is added to $\mathcal{N}_{\mathcal{T}}$, and* (3) *for every arc $((i,t^i_k),(j,\bar{t}))$ in $\mathcal{D}_{\mathcal{T}}$, a new arc $((i,t^i_{\text{new}}),(j,\bar{t}))$ is added to $\mathcal{A}_{\mathcal{T}}$, then the resulting time-expanded network again has the early arrival property.*

**Proof.** The only new arcs added to $\mathcal{A}_{\mathcal{T}}$ are those of the form $((i,t^i_{\text{new}}),(j,\bar{t}))$ where $((i,t^i_k),(j,\bar{t}))$ was already in $\mathcal{A}_{\mathcal{T}}$ and $t^i_{\text{new}} > t^i_k$. If $\mathcal{D}_{\mathcal{T}}$ already satisfied Property 2, then $\bar{t} \leqslant t^i_k + \tau_{ij}$. Hence $\bar{t} < t^i_{\text{new}} + \tau_{ij}$, and Property 2 is preserved. The only new node added is $(i, t^i_{\text{new}})$. Now if $\mathcal{D}_{\mathcal{T}}$ already satisfied Property 3, it must be that for all $(i,j) \in \mathcal{A}$, there exists a timed arc $((i,t^i_k),(j,\bar{t})) \in \mathcal{A}_{\mathcal{T}}$ for some $\bar{t}$. But for each such arc, the new arc $((i,t^i_{\text{new}}),(j,\bar{t}))$ is added to $\mathcal{A}_{\mathcal{T}}$. Thus Property 3 is preserved, too. Q.E.D.

Unfortunately, after adding the new time point, the new node, and the new arcs, the partially time-expanded network no longer satisfies the longest-feasible-arc property. However, a few simple changes to the network restore the longest-feasible-arc property while maintaining the early arrival property, as is shown in the following lemma.

**Lemma 4.** *After refining a partially time-expanded network $\mathcal{D}_{\mathcal{T}}$ having the longest-feasible-arc property by adding new time point $t^i_{\text{new}}$ with $t^i_k < t^i_{\text{new}} < t^i_{k+1}$ to $\mathcal{T}_i$, adding new node $(i, t^i_{\text{new}})$ to $\mathcal{N}_{\mathcal{T}}$, and adding for every arc $((i,t^i_k)(j,\bar{t}))$ in $\mathcal{D}_{\mathcal{T}}$, a new arc $((i,t^i_{\text{new}}),(j,\bar{t}))$ to $\mathcal{A}_{\mathcal{T}}$, the longest-feasible-arc property will be restored by*

(1) *replacing every arc $((j,t'),(i,t^i_k))$ with $t' + \tau_{ji} \geqslant t^i_{\text{new}}$ with arc $((j,t'),(i,t^i_{\text{new}}))$, and*

(2) *finding, for every new arc, $((i,t^i_{\text{new}}),(j,\bar{t}))$, the node, $(i,t')$, with largest $t'$, such that $\bar{t} < t' \leqslant t^i_{\text{new}} + \tau_{ij}$ and, if such a node exists, replacing arc $((i,t^i_{\text{new}}),(j,\bar{t}))$ with arc $((i,t^i_{\text{new}}),(j,t'))$.*

**Proof.** The only arcs in $\mathcal{D}_{\mathcal{T}}$ that may violate the longest-feasible-arc property after the introduction of the new node $(i,t^i_{\text{new}})$ are those with head $(i,t^i_k)$. These arcs are replaced if needed. The newly added arcs may also violate the longest-feasible-arc property but are replaced if needed. Q.E.D.

When Algorithm 4, Refine, is applied to a partially time-expanded network with the early arrival property, Lemma 3 ensures that the resulting partially time-expanded network will also have the early arrival property. When Algorithm 5, Restore, is applied to a partially time-expanded network with the early

arrival property, Lemma 2 guarantees the property is maintained. Lemma 4 ensures both steps preserve the longest-feasible-arc property. Thus Algorithm 3, Lengthen-arc, preserves both the early arrival and longest-feasible-arc properties.

**Algorithm 3** (Lengthen-arc$((i,t),(j,t'))$)
**Require:** Arc $((i,t),(j,t')) \in \mathcal{A}_{\mathcal{T}}$
1: Refine$(j, t+\tau_{ij})$
2: Restore$(j, t+\tau_{ij})$.

**Algorithm 4** (Refine$(i, t^i_{\text{new}})$)
**Require:** Node $i \in \mathcal{N}$; time point $t^i_{\text{new}} \in \mathcal{T}_i$ with $t^i_k < t^i_{\text{new}} < t^i_{k+1}$
1: Add node $(i, t^i_{\text{new}})$ to $\mathcal{N}_{\mathcal{T}}$;
2: Delete arc $((i,t^i_k),(i,t^i_{k+1}))$ from $\mathcal{A}_{\mathcal{T}}$; add arcs $((i,t^i_k),(i,t^i_{\text{new}}))$ and $((i,t^i_{\text{new}}),(i,t^i_{k+1}))$ to $\mathcal{A}_{\mathcal{T}}$
3: **for** $((i,t^i_k),(j,t)) \in \mathcal{A}_{\mathcal{T}}$ **do**
4: Add arc $((i,t^i_{\text{new}}),(j,t))$ to $\mathcal{A}_{\mathcal{T}}$
5: **end for**.

**Algorithm 5** (Restore$(i, t^i_{\text{new}})$)
**Require:** Node $i \in \mathcal{N}$; time point $t^i_{\text{new}} \in \mathcal{T}_i$ with $t^i_k < t^i_{\text{new}} < t^i_{k+1}$
1: **for all** $((i,t^i_k),(j,t)) \in \mathcal{A}_{\mathcal{T}}$ **do**
2: Set $t' = \arg\max\{s \in \mathcal{T}_j \mid s \leqslant t^i_{\text{new}} + \tau_{ij}\}$.
3: **if** $t' \neq t$ **then**
4: Delete arc $((i,t^i_{\text{new}}),(j,t))$ from $\mathcal{A}_{\mathcal{T}}$; add arc $((i,t^i_{\text{new}}),(j,t'))$ to $\mathcal{A}_{\mathcal{T}}$
5: **end if**
6: **end for**
7: **for all** $((j,t),(i,t^i_k)) \in \mathcal{A}_{\mathcal{T}}$ such that $t+\tau_{ji} \geqslant t^i_{\text{new}}$ **do**
8: Delete arc $((j,t),(i,t^i_k))$ from $\mathcal{A}_{\mathcal{T}}$; add arc $((j,t),(i,t^i_{\text{new}}))$ to $\mathcal{A}_{\mathcal{T}}$
9: **end for**.

**Observation 1.** Because Lengthen-arc takes a timed arc that is too short and replaces it with a timed arc that has the correct length (i.e., the actual travel time), the length of an arc is corrected at most once. This implies that Solve-CTSNDP, with Lengthen-arc used in step 8, will terminate in a finite number of iterations. In particular, the number of iterations is bounded above by $|\hat{\mathcal{T}}||\mathcal{A}|$, since this is an upper bound on the number of timed arcs.

The reason for refining the partially time-expanded network is that the solution $(x(\mathcal{D}_{\mathcal{T}}), y(\mathcal{D}_{\mathcal{T}}))$ to SND$(\mathcal{D}_{\mathcal{T}})$ cannot be converted to a solution to CTSNDP with equal value. A solution $(x(\mathcal{D}_{\mathcal{T}}), y(\mathcal{D}_{\mathcal{T}}))$ specifies the path each commodity $k$ takes from its origin to its destination as well as the consolidations of commodities on arcs in the network, where a consolidation of commodities on an arc $(i,j) \in \mathcal{A}$ occurs when two or more commodities travel on that arc at the same time (i.e., $|\mathcal{K}_{((i,t),(j,t'))}| \geqslant 2$), where

$$\mathcal{K}_{((i,t),(j,t'))} = \{k \in \mathcal{K} \mid x^{k,t,t'}_{ij} = 1\}.$$

(Assuming $\sum_{k \in \mathcal{K}_{((i,t),(j,t'))}} q_k \leqslant u_{ij}$, these commodities will share the same resource (i.e., will be loaded into the same trailer), and the fixed cost $f_{ij}$ is incurred only once.)

Because constraints (5) ensure that the paths specified in the solution for the commodities are time feasible with actual travel times, the solution cannot be converted to a solution to CTSNDP with equal value because the consolidations specified in the solution cannot be realized when the actual travel times are observed. This implies that there has to be a commodity $k \in \mathcal{K}$ that flows on an arc that is too short; i.e., there has to be a commodity $k$ and an arc $((i,t),(j,t'))$ with $t' < t + \tau_{ij}$ for which $x^{k\tau'}_{ij} = 1$. Next, we discuss how to identify arcs that are too short.

### 4.3. Identifying Arcs to Lengthen

We formulate the problem of identifying arcs to lengthen as an MIP. For each $a \in \mathcal{A}_{\mathcal{T}}$, let $J_a$ be the set of all pairs of commodities dispatched on $a$ in the optimal solution to SND$(\mathcal{D}_{\mathcal{T}})$; i.e., $J_a = \{(k_1,k_2) \in \mathcal{K}_a \times \mathcal{K}_a \mid k_1 < k_2\}$. Furthermore, let $\bar{\mathcal{F}}$ denote the set of arcs $a \in \mathcal{A}_{\mathcal{T}}$ on which more than one commodity is dispatched; i.e., $\bar{\mathcal{F}} = \{a \in \mathcal{A}_{\mathcal{T}} : |\mathcal{K}_a| \geqslant 2\}$. To formulate the MIP, for each commodity $k \in \mathcal{K}$, let $P_k = \{i^k_1 = o_k, i^k_2, i^k_3, \ldots, i^k_p = d_k\}$ represent the path that commodity $k$ follows from its source to its sink, in terms of the nodes it visits along the way, in the optimal solution to SND$(\mathcal{D}_{\mathcal{T}})$, and for each arc $(i_j, i_{j+1}) \in P_k$, let $\bar{\tau}^k_{i_j i_{j+1}}$ represent the travel time modeled in $\mathcal{D}_{\mathcal{T}}$ for that arc. Then, for each $k \in \mathcal{K}$ and each node $i^k_j$ in path $P_k$, define variables $\gamma^k_{i_j} \geqslant 0$ to represent the dispatch time of commodity $k$ at node $i^k_j$, and for each $a \in \mathcal{A}_{\mathcal{T}}$ and each $k \in \mathcal{K}$, define two sets of variables: $\theta^k_{i_j i_{j+1}}$ to represent the travel time of arc $(i_j, i_{j+1})$ when taken by commodity $k$ and $\sigma^k_{i_1 i_{j+1}}$ to represent whether the arc is allowed to be too short when taken by commodity $k$. With these variables, we define the following MIP to determine the fewest number of arcs that must be too short for the consolidations that occur in a solution to SND$(\mathcal{D}_{\mathcal{T}})$ to be realized:

$$Z = \min \sum_{k \in \mathcal{K}} \sum_{j=1}^{|P_k|-1} \sigma^k_{i_j i_{j+1}}$$

$$\theta^k_{i_j i_{j+1}} \geqslant \tau_{i_j i_{j+1}}(1 - \sigma^k_{i_j i_{j+1}}), \tag{6}$$

$$\gamma^k_{i_j} + \theta^k_{i_j i_{j+1}} \leqslant \gamma^k_{i_{j+1}}, \quad \forall\, k \in \mathcal{K},\ j = 1, \ldots, |P_k| - 1, \tag{7}$$

$$e_k \leqslant \gamma^k_{o_k}, \quad \forall\, k \in \mathcal{K}, \tag{8}$$

$$\gamma^k_{|P_k|-1} + \theta_{i_{|P_k|-1} d_k} \leqslant l_k, \quad \forall\, k \in \mathcal{K}, \tag{9}$$

$$\gamma^{k_1}_i = \gamma^{k_2}_i, \quad \forall\, (k_1,k_2) \in J_{((i,t),(j,t'))}, \quad \forall\, ((i,t),(j,t')) \in \bar{\mathcal{F}}, \tag{10}$$

$$\gamma^k_{i_j} \geqslant 0, \quad \forall\, k \in \mathcal{K},\ j = 1, \ldots, |P_k|, \tag{11}$$

$$\theta^k_{i_j i_{j+1}} \geqslant \bar{\tau}^k_{i_j i_{j+1}},\ \sigma^k_{i_j i_{j+1}} \in \{0,1\}, \quad \forall\, k \in \mathcal{K},\ j = 1, \ldots, |P_k| - 1. \tag{12}$$

The objective is to minimize the number of arcs that are assigned a travel time shorter than the actual travel

time. Constraints (6) count the number of arcs that are assigned a travel time shorter than the actual. Constraints (7) ensure the dispatch times are in accordance with the assigned travel times. Constraints (8) and (9) ensure that the dispatch times prescribed for a commodity enable it to depart from its origin after it becomes available and arrive at its destination before it is due. Constraints (10) ensure that all consolidations seen in the solution to SND($\mathscr{D}_{\mathscr{T}}$) are maintained.

We note that when the optimal value to this MIP is zero, the dispatch times $\gamma_i^k$ show how to convert the solution to SND($\mathscr{D}_{\mathscr{T}}$) to a solution to CTSNDP of equal cost and Solve-CTSNDP can terminate. Conversely, when the optimal value is greater than zero, we choose to lengthen arcs $(i_j, i_{j+1})$ such that $\sigma^k_{i_j i_{j+1}} = 1$ for some $k$. We also note that we can speed up the MIP (without invalidating it) by fixing to 0 variables $\sigma^k_{i_j i_{j+1}}$ associated with arcs in the solution to SND($\mathscr{D}_{\mathscr{T}}$) that already have the correct length (i.e., are not "short").

### 4.4. Deriving a Solution to CTSNDP from a Solution to SND($\mathscr{D}_{\mathscr{T}}$)

When a solution to SND($\mathscr{D}_{\mathscr{T}}$) cannot be converted to a solution to CTSNDP of equal cost, we can still use it to construct a feasible solution of greater cost. Similar to the integer program above, we seek to find dispatch times for every commodity $k$ at every node in $P_k$ such that (1) the commodity's availability time and due time are respected, and (2) commodities that are dispatched together in the optimal solution to SND($\mathscr{D}_{\mathscr{T}}$) are still dispatched together as much as possible (so that the same consolidations are realized). However, unlike the integer program above, we now ensure that the dispatch times respect the actual travel times of the arcs.

To determine these dispatch times, we solve a linear program that is similar to the integer program above. We again use the variables $\gamma_{i_j}^k$, but now, for each pair of commodities $(k_1, k_2) \in J_{((i,t),(j,t'))}$, we define a variable $\delta_{ijt}^{k_1 k_2} \geqslant 0$ to capture any difference in dispatch time of the two commodities on arc $(i, j)$. With these variables we then solve the following linear program (LP):

$$Z = \min \sum_{((i,t),(j,t'))\in\bar{\mathscr{J}}} \sum_{(k_1,k_2)\in J_{((i,t),(j,t'))}} \delta_{ijt}^{k_1 k_2}$$

$$\gamma_{i_j}^k + \tau_{i_j i_{j+1}} \leqslant \gamma_{i_{j+1}}^k, \quad \forall\, k \in \mathscr{K},\ j = 1, \ldots, |P_k| - 1, \tag{13}$$

$$e_k \leqslant \gamma_{o_k}^k, \quad \forall\, k \in \mathscr{K}, \tag{14}$$

$$\gamma_{i_{|P_k|-1}} + \tau_{i_{|P_k|-1} d_k} \leqslant l_k, \quad \forall\, k \in \mathscr{K}, \tag{15}$$

$$\delta_{ijt}^{k_1 k_2} \geqslant \gamma_i^{k_1} - \gamma_i^{k_2}, \quad \forall\, (k_1, k_2) \in J_{((i,t),(j,t'))},\ \forall\, ((i,t),(j,t')) \in \bar{\mathscr{J}}, \tag{16}$$

$$\delta_{ijt}^{k_1 k_2} \geqslant \gamma_i^{k_2} - \gamma_i^{k_1}, \quad \forall\, (k_1, k_2) \in J_{((i,t),(j,t'))},\ \forall\, ((i,t),(j,t')) \in \bar{\mathscr{J}}, \tag{17}$$

$$\gamma_{i_j}^k \geqslant 0, \quad \forall\, k \in \mathscr{K},\ j = 1, \ldots, |P_k|. \tag{18}$$

Because the optimal solution to SND($\mathscr{D}_{\mathscr{T}}$) satisfies constraints (5), there will always be a feasible solution to the LP. The only reason the consolidations seen in the solution to SND($\mathscr{D}_{\mathscr{T}}$) cannot be seen in a feasible solution to the CTSNDP is if a commodity participating in a consolidation travels on a path that contains an arc that is too short.

Note that neither the MIP or LP presented above require that the consolidations take place at the times "suggested" by the solution to SND($\mathscr{D}_{\mathscr{T}}$). It only stipulates that the commodities have to follow the same path and that the consolidations that occurred are reproduced (as much as possible). However, the dispatch times $\gamma_i^k$ prescribed by the solution to the LP represent a feasible solution to CTSNDP. Let the value of this solution be $z$(P-CTSNDP). Thus, at each iteration of the algorithm, we can calculate an optimality gap with the following formula:

$$(z(\text{P-CTSNDP}) - z(\mathscr{D}_{\mathscr{T}}))/z(\text{P-CTSNDP}). \tag{19}$$

This also allows us to specify an optimality tolerance as a stopping condition when executing Solve-CTSNDP.

### 4.5. Comparison with the Time Bucket Formulation for TSPTW (Dash et al. 2012)

We finish this section by more closely considering the branch-and-cut algorithm of Dash et al. (2012) for the TSPTW, because it also employs dynamic discretization discovery ideas. Dash et al. (2012) present a formulation of the TSPTW that is based on partitioning the time windows into subwindows or *buckets*. The strength of the LP relaxation of the time bucket formulation depends on the partition of the time windows. In general, having more buckets results not only in stronger LP relaxations but also in larger LP relaxations. To develop an efficient branch-and-cut algorithm, it is therefore important to strike the right balance between the strength of the LP relaxation and the time it takes to solve the LP relaxation. To obtain a "good" partition of the time windows, Dash et al. (2012) employ an iterative linear programming-based partition refinement scheme; the scheme dynamically discovers an appropriate partition (or discretization) of the time windows. To enhance the iterative refinement scheme, a *bucket graph* is constructed in which arcs $((i, b_i), (j, b_j))$ indicate that it is possible to reach bucket $b_j$ at node $j$ from bucket $b_i$ at node $i$, and bucket preprocessing techniques are employed. This bucket graph can equivalently be viewed as a partially time-expanded network satisfying Properties 1–4. (A fully time-expanded network would correspond to partition of the time windows in subwindows of length 1.) Thus, Dash et al. (2012) also manipulate a partially time-expanded network.

However, there are critical differences between their work and ours, in terms of the problems studied

and the solution approaches developed. Regarding problems studied, we note that (1) in the TSPTW it is possible to restrict the searching for solutions to those without unforced waiting time, whereas in the CTSNDP waiting is often critical to achieve consolidations; and (2) in the CTSNDP it is trivial to find feasible solutions, whereas finding feasible solutions for the TSPTW is NP-hard (Savelsbergh 1986). Regarding solution approaches, the major differences are that (1) Dash et al. (2012) employ dynamic discretization discovery as a preprocessing scheme (i.e., once a partition has been established, it is never changed during the branch-and-cut search), whereas we continue to refine the discretization until the solution to SND($\mathscr{D}_{\mathscr{T}}$) can be converted to an optimal solution to CTSNDP; (2) Dash et al. (2012) use information from the solution to an LP to heuristically refine the set of time points, whereas we use information from the solution to an integer program to carefully refine the set of time points to guarantee convergence to an optimal solution to CTSNDP; and (3) we focus much more strongly on keeping the number of time points in the partially time-expanded network to a minimum.

## 5. A Computational Study

The goal of the computational study presented in this section is to demonstrate the effectiveness and efficiency of the (straightforward implementation of the) proposed iterative refinement algorithm for solving CTSNDP and to gain a better understanding of the factors that contribute to its performance. We first describe the instances used in the computational study (Section 5.1), then we illustrate some of the challenges associated with discretizing time (Section 5.2), and then we present the results of a series of experiments that demonstrate the efficacy of the proposed algorithm (Section 5.3).

To be able to assess the performance of the proposed algorithm on an instance, we also solve the instance using the formulation with full-time discretization. We will refer to this as using *full discretization*, or FD. (We note that the same preprocessing techniques are used when using full discretization as when solving SND($\mathscr{D}_{\mathscr{T}}$)). Abusing terminology, we will sometimes use FD to refer to the integer program it solves.

### 5.1. Instances

We derive the instances used in our computational study from the C and C+ instances described in detail in Crainic et al. (2001). These instances have been used to benchmark the performance of many algorithms for the capacitated fixed charge network design problem (Ghamlouch et al. 2003, 2004; Crainic et al. 2004; Katayama et al. 2009; Hewitt et al. 2010; Yaghini et al. 2012; Hewitt et al. 2013). The instances vary with respect to the number of nodes (20, 30), arcs (230, 300, 520, 700), commodities (40, 100, 200, and 400), whether the variable costs, $c_{ij}$, outweigh the fixed costs, $f_{ij}$, and whether the arcs are loosely or tightly capacitated. The results we present next are based on the 24 instances with 100, 200, or 400 commodities. The other, smaller instances are solved nearly instantaneously and, thus, do not yield insights into the performance of our algorithm. We provide a detailed list of these instances in Table 1. We refer to these instances as "untimed" as they do not have any time attributes; e.g., there are no travel times associated with arcs and there are no available and due times associated with commodities.

We "timed" these instances using the following scheme. First, for a given parameter $\nu$, we set the travel time in minutes, $\tau_{ij}$, of arc $(i, j)$ to be proportional to its fixed charge. Specifically, we set $\tau_{ij} = \nu f_{ij}\ \forall (i, j) \in \mathscr{A}$. We calculated the value $\nu$ based on the premise that $f_{ij}$ represents the transportation cost for a carrier that spends \$0.55 cents per mile and their trucks travel at 60 miles per hour.

When commodities become available and when they are due partially dictates whether they can consolidate. To be able to measure the degree to which these parameters impact consolidation, we generate for each untimed instance with associated arc travel times multiple instances with varying values for commodity available and due times. More specifically, we first calculate for each commodity $k \in \mathscr{K}$ the length of the shortest path from $o_k$ to $d_k$ with respect to the travel

**Table 1.** "Flat" Instances from Crainic et al. (2001) Used in Study

| Instance | $\lvert\mathscr{N}\rvert$ | $\lvert\mathscr{A}\rvert$ | $\lvert\mathscr{K}\rvert$ | Fixed (F) or variable (V) cost | Tight (T) or loosely (L) capacitated |
|---|---|---|---|---|---|
| c37 | 20 | 230 | 200 | V | L |
| c38 | 20 | 230 | 200 | F | L |
| c39 | 20 | 230 | 200 | V | T |
| c40 | 20 | 230 | 200 | F | T |
| c45 | 20 | 300 | 200 | V | L |
| c46 | 20 | 300 | 200 | F | L |
| c47 | 20 | 300 | 200 | V | T |
| c48 | 20 | 300 | 200 | F | T |
| c49 | 30 | 520 | 100 | V | L |
| c50 | 30 | 520 | 100 | F | L |
| c51 | 30 | 520 | 100 | V | T |
| c52 | 30 | 520 | 100 | F | T |
| c53 | 30 | 520 | 400 | V | L |
| c54 | 30 | 520 | 400 | F | L |
| c55 | 30 | 520 | 400 | V | T |
| c56 | 30 | 520 | 400 | F | T |
| c57 | 30 | 700 | 100 | V | L |
| c58 | 30 | 700 | 100 | F | L |
| c59 | 30 | 700 | 100 | V | T |
| c60 | 30 | 700 | 100 | F | T |
| c61 | 30 | 700 | 400 | V | L |
| c62 | 30 | 700 | 400 | F | L |
| c63 | 30 | 700 | 400 | V | T |
| c64 | 30 | 700 | 400 | F | T |

times $\tau_{ij}$. We call this length $\mathcal{L}_k$ and the average of these $|\mathcal{K}|$ lengths $\mathcal{L}$. We then create three normal distributions from which we draw the available time for each commodity, all of which are defined by a mean, $\mu_e$, of $\mathcal{L}$ minutes but vary with respect to their standard deviation. Specifically, we consider three values for the standard deviation, $\sigma_e$: $\frac{1}{3}\mathcal{L}$, $\frac{1}{6}\mathcal{L}$, and $\frac{1}{9}\mathcal{L}$. Given commodity $k$'s available time, $e_k$, at its origin, it can arrive at its destination no sooner than $e_k + \mathcal{L}_k$. Next, we introduce for each commodity $k \in \mathcal{K}$ a time flexibility, $f_k \geqslant 0$, and set its due time, $l_k$, to $e_k + \mathcal{L}_k + f_k$. Similar to determining the available times, we create two normal distributions from which we draw these time flexibilities. The first has a mean, $\mu_f$, of $\frac{1}{2}\mathcal{L}$, and the second has $\mu_f = \frac{1}{4}\mathcal{L}$. Both distributions have a standard deviation, $\sigma_f$ of $\frac{1}{6}\mu_f$.

In summary, there are three normal distributions from which we draw commodity available times and two normal distributions from which we draw commodity time flexibility. As such, we have six sets of instances, one for each combination of distributions, and randomly generate three instances for each set. When we randomly sample from one of the distributions, we repeatedly draw from the distribution until we generate a value that falls within three standard deviations of the mean of the distribution.

Therefore, we have a total of $24 \times 6 \times 3 = 432$ instances. Finally, we consider five discretization parameters, $\Delta$: 60 minutes, 30 minutes, 15 minutes, 5 minutes, and 1 minute. We summarize the parameter values used to generate the instances used in our computational study in Table 2.

### 5.2. The Impact of Discretizing Time

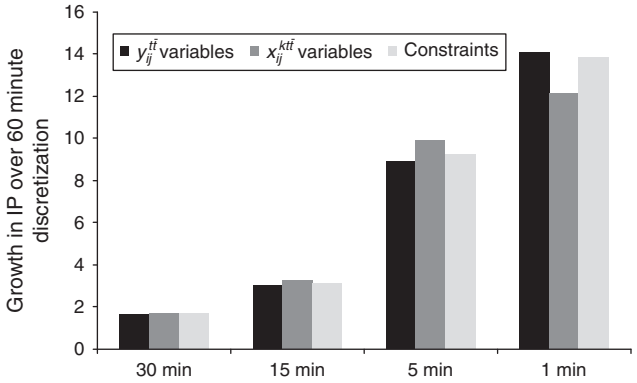
As mentioned in the introduction, the granularity of the time discretization has an impact on the accuracy of a formulation as well as its size.

With regard to the size of the formulation, we report in Figure 4 the growth in the full discretization integer program, in terms of the number of variables and the number of constraints, as the granularity is refined (i.e., when $\Delta$ is changed from 60 to 30, from 60 to 15, from 60 to 5, and from 60 to 1). (We report averages over all instances.) We see that the growth is substantial. Refining the granularity from a 60-minute discretization to a 1-minute discretization results in a factor 15 increase in the size of the integer program.

**Table 2.** Time-Oriented Characteristics

| Normal distribution | $\mu$ | $\sigma$ |
|---|---|---|
| For generating $e_k$ | $\mathcal{L}$ | $\frac{1}{3}\mathcal{L}$, $\frac{1}{6}\mathcal{L}$, and $\frac{1}{9}\mathcal{L}$ |
| For generating $f_k$ | $\frac{1}{2}\mathcal{L}$, $\frac{1}{4}\mathcal{L}$ | $\frac{1}{6}\mu$ |
| $\Delta$ | 60 min, 30 min, 15 min, 5 min, 1 min | |

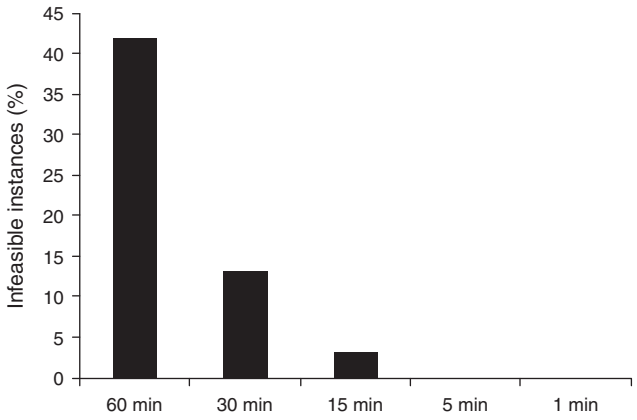
**Figure 4.** Growth in FD When the Discretization $\Delta$ Is Changed from 60 to a Smaller Value




With regard to the accuracy, consider the time feasibility of a path $p$ in the flat network. A path $p$ in the flat network is time feasible if $\sum_{(i,j)\in p} \tau_{ij} \leqslant l_k - e_k$. However, for discretization $\Delta$, the travel time of an arc $(i, j)$ will be modeled as $\Delta\lceil \tau_{ij}/\Delta \rceil$, which can be strictly greater than $\tau_{ij}$; the available time of a commodity $k$ will be modeled as $\Delta\lceil e_k/\Delta \rceil$, which can be strictly greater than $e_k$; and the due time of a commodity $k$ will be modeled as $\Delta\lfloor l_k/\Delta \rfloor$, which can be strictly smaller than $l_k$. As a result, paths that are time feasible when considering the true travel times and the true available and due time may be rendered infeasible for a discretization $\Delta$. Furthermore, if all time-feasible paths for some commodity $k$ in an instance are rendered infeasible for a discretization $\Delta$, then the instance itself will become infeasible. That this is a relevant and important issue is shown in Figure 5, where we display the percentage of the 432 instances that become infeasible as a result of discretization for different choices of $\Delta$. We see that for $\Delta = 60$, over 40% of the instances become infeasible (when they are in fact feasible for a one-minute discretization).

Figures 4 and 5 highlight the fundamental issue with modeling time-indexed decisions; while finer

**Figure 5.** The Percentage of Instances That Become Infeasible Because of Discretization for Different Choices of $\Delta$



discretizations of time lead to more accurate models, an enumerative approach to choosing time points to model can lead to significantly larger optimization problems.

### 5.3. Performance of Solve-CTSNDP

We conducted a set of a computational experiments to assess the performance of our implementation of the proposed dynamic discretization discovery algorithm for solving CTSNDP using the instances that were not rendered infeasible by discretization (recall Figure 5 in the previous subsection). In all experiments, we gave FD and Solve-CTSNDP the same stopping criteria: a proven optimality gap of less than or equal to 1%, where the optimality gap is calculated using (19), or a maximum run time of two hours. Note that when Solve-CTSNDP stops because the feasible solution to the relaxation, $SND(\mathscr{D}_{\mathscr{T}})$, can be converted to a feasible solution to CTSNDP, it terminates with a provably optimal solution. All experiments were run on a cluster of computers, and each job was allowed to use a maximum of 16 GB of memory. Experiments were run on a cluster of nodes containing 32 cores each, with speeds ranging from 2.3 to 2.8 GHz. Each node has 256 GB of RAM.

To compare the performance of FD and Solve-CTSNDP, we graph averages over instances with the same discretization parameter $\Delta$ of the time to termination (see Figure 6), the optimality gap at termination (see Figure 7), and the percentage of instances solved (see Figure 8). We note that Solve-CTSNDP never exceeds the 16 GB memory limit, whereas FD does so for 167 of the 432 (nearly 39%) instances with $\Delta = 1$ (and never for the other discretizations). Therefore, we do not display results for $\Delta = 1$ in Figures 6 and 7. Instead, for the instances with $\Delta = 1$, we report in Table 3 the performance of both methods separately for the instances where FD does not exceed the 16 GB memory limit (FD $\leqslant$ 16 GB) and the instances where FD does exceed the 16 GB memory limit (FD > 16 GB).

We see that the performance of Solve-CTSNDP is comparable to that of FD on instances with coarse discretizations (60 and 30 minutes) but clearly outperforms FD on fine discretizations (15 minutes, 5 minutes, and 1 minute), where it requires less time to solve more instances and when it cannot solve an instance yields a smaller optimality gap. Thus, as expected, the performance of FD significantly degrades as the discretization becomes finer, but, as anticipated, Solve-CTSNDP remains effective. We complement the averages reported in the previous figures with distributions (in deciles) for each discretization of the time to termination (see Figure 9) and the optimality gap at termination (see Figure 10) for Solve-CTSNDP. In these figures, we see that the distribution for the time to termination is positively skewed, with outliers pulling the average up. For example, for all discretizations the 60th percentile for the time to termination is less than seven minutes (and significantly less than the average). We note that for all discretizations the 80th percentile of optimality gaps is within our optimality tolerance of 1%.

**Figure 6.** Time to Termination for Different $\Delta$

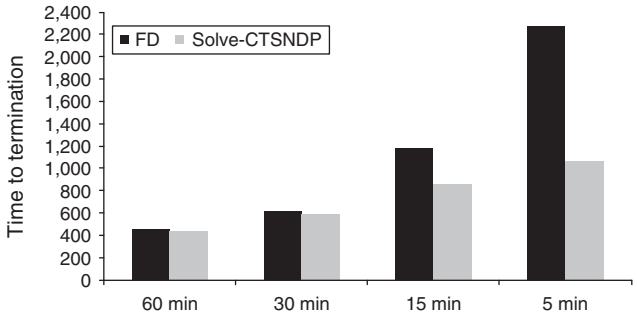



**Figure 7.** Optimality Gap at Termination for Different $\Delta$

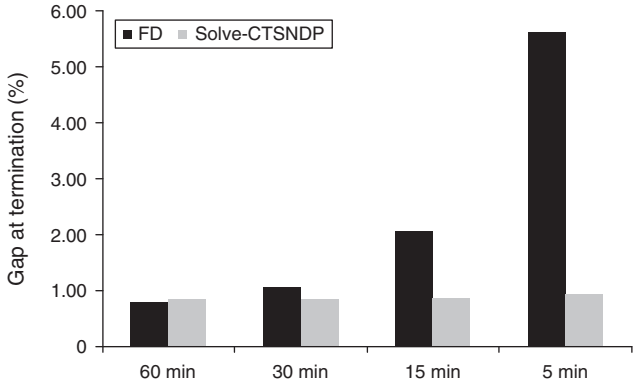
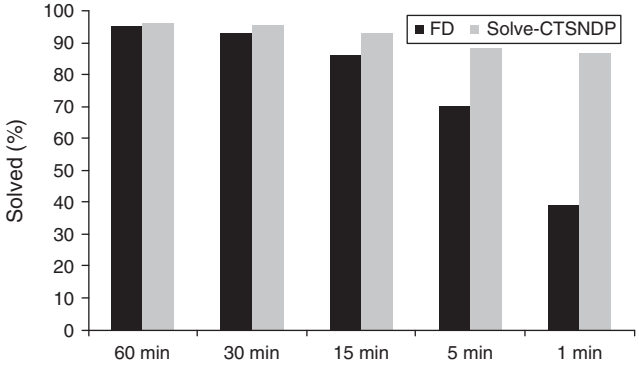



**Table 3.** Performance When $\Delta = 1$ Minute

| Instances | Method | Time (%) | Opt. gap (%) | Solved (%) |
|---|---|---|---|---|
| FD $\leqslant$ 16 GB | FD | 3,097.832 | 3.85 | 62.26 |
| | Solve-CTSNDP | 417.04 | 0.78 | 98.87 |
| FD > 16 GB | Solve-CTSNDP | 3,106.49 | 1.33 | 67.66 |

**Figure 8.** Fraction of Instances Solved Within the Memory and Time Limits for Different $\Delta$



To better understand why Solve-CTSNDP outperforms FD, we first compare $|\mathcal{N}_{\text{FD}}|$, the cardinality of the node set of the fully time-expanded network that forms the basis of the integer program solved by FD, and $|\mathcal{N}_{\mathcal{T}}|$, the node set of the partially time-expanded network that forms the basis of the *last* integer program solved by Solve-CTSNDP. In Figure 11, we show for the instances with a given discretization parameter $\Delta$ and for a given ratio $r$ $(0 < r < 1)$, the fraction of instances with $|\mathcal{N}_{\mathcal{T}}|/|\mathcal{N}_{\text{FD}}| \leqslant r$. We see that Solve-CTSNDP works with significantly smaller time-expanded networks while searching for a provably optimal solution to CTSNDP than FD, especially for instances with $\Delta = 1$, where $|\mathcal{N}_{\mathcal{T}}|/|\mathcal{N}_{\text{FD}}| \leqslant 0.04$ for *all* instances.

Next, in Figure 12, we compare the size of the integer programs solved by FD and the size of the last integer programs solved by Solve-CTSNDP (i.e., of the integer program associated with the final SND($\mathcal{D}_{\mathcal{T}}$)). Specifically, we show for the instances with a given discretization parameter $\Delta$ the averages of the following ratios: the number of $y_{ij}^{t\bar{t}}$ variables in the last integer program solved by Solve-CTSNDP and the number of $y_{ij}^{t\bar{t}}$ variables in the integer program solved by FD, the number of $x_{ij}^{kt\bar{t}}$ variables in the last integer program solved by Solve-CTSNDP and the number of $x_{ij}^{kt\bar{t}}$ variables in the integer program solved by FD, and the number of constraints in the last integer program solved by Solve-CTSNDP and the number of constraints in the integer program solved by FD. We see that the last integer programs solved by Solve-CTSNDP are significantly smaller than those solved by FD. Furthermore, as expected, we see that the finer the discretization, the smaller the relative size of the integer program associated with the final SND($\mathcal{D}_{\mathcal{T}}$) solved by Solve-CTSNDP.

**Figure 9.** Time to Termination

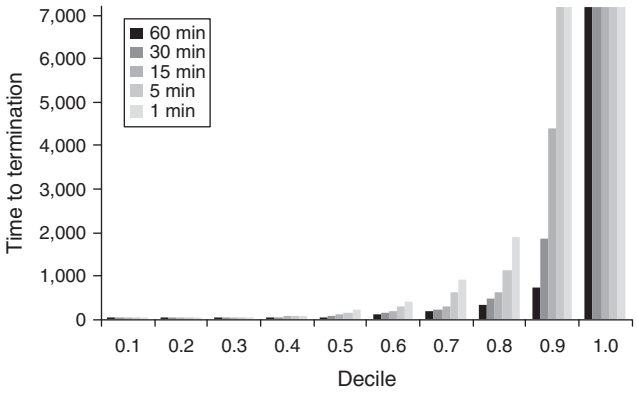
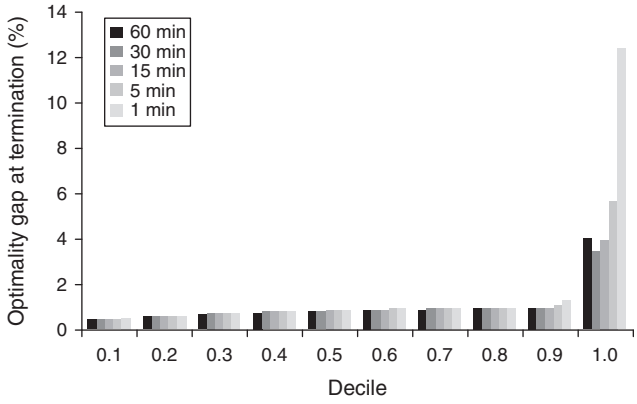



**Figure 10.** Optimality Gap at Termination




In Figure 11, we saw that the cardinality of the node set of the partially time-expanded network of the last SND($\mathcal{D}_{\mathcal{T}}$)) solved by Solve-CTSNDP is much smaller than the cardinality of the node set of the fully time-expanded network for the same instance. Next, we explore the growth of the partially time-expanded networks during the execution of Solve-CTSNDP. More

**Figure 11.** Relative Time-Expanded Network Size of the Final SND($\mathcal{D}_{\mathcal{T}}$)

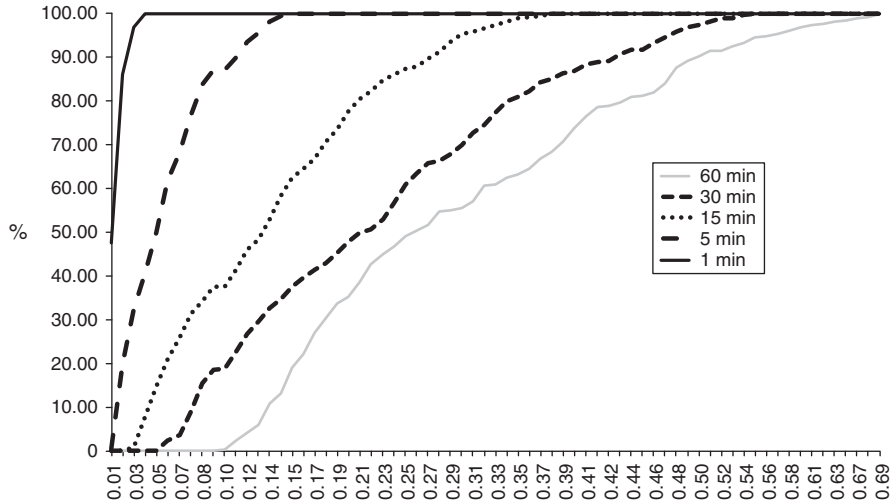


**Figure 12.** Relative Integer Programming Size Associated with the Final SND($\mathscr{D}_{\mathscr{T}}$)

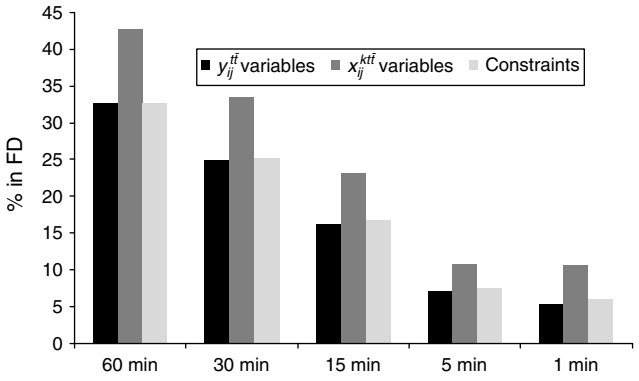
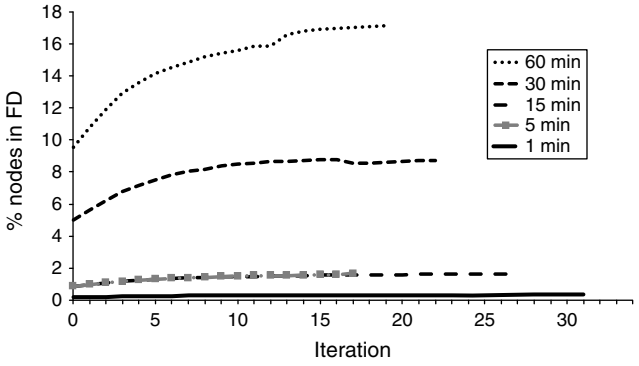



**Figure 13.** Relative Time-Expanded Network Size by Iteration for Instances with $|\mathscr{N}| = 20$, $|\mathscr{A}| = 200$, $|\mathscr{K}| = 200$, $\sigma_e = \frac{1}{9}\mathscr{L}$, and $\mu_f = \frac{1}{2}\mathscr{L}$




specifically, in Figure 13, we report for the instances with a given discretization parameter $\Delta$ the averages of $|\mathscr{N}_{\mathscr{T}}|/|\mathscr{N}_{\text{FD}}|$ by iteration of Solve-CTSNDP for a specific flat network and set of timing parameter values (this set of instances was chosen because Solve-CTSNDP struggled the most with them, solving the smallest fraction of instances).

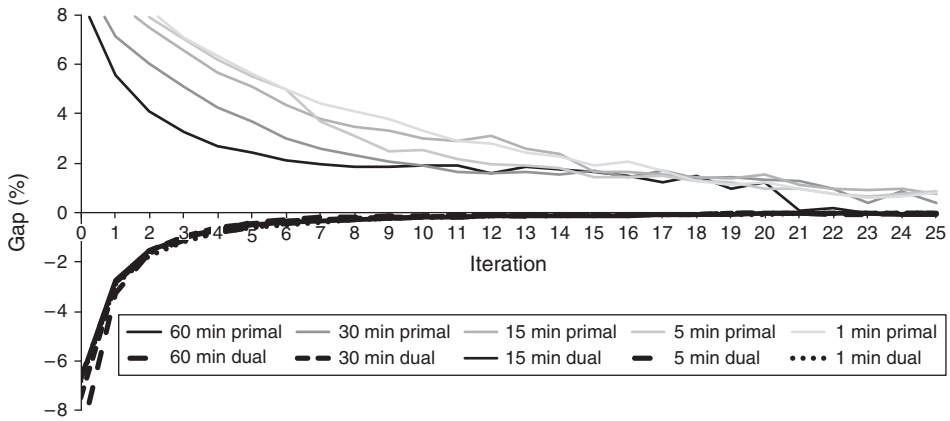
We see that the size of the partially time-expanded networks created and refined by Solve-CTSNDP remains fairly stable during the execution of algorithm (even for instances with discretization $\Delta = 60$, the relative size only increases from about 10% to about 18%).

The results of the computational experiments discussed above clearly demonstrate Solve-CTSNDP's superiority over FD. We conclude that Solve-CTSNDP outperforms FD because it starts with a significantly smaller time-expanded network than FD and refines it in such a way that the time-expanded network grows only modestly. As a result, the integer programs solved by Solve-CTSNDP are significantly smaller than those solved by FD.

Next, we analyze the changes in the lower and upper bounds during the execution of Solve-CTSNDP. We note that because the number of iterations required to solve an instance varies across the instances, the number of instances for which we have a lower and upper bound at the $k$th iteration is typically greater than the number of instances for which we have a lower and upper bound at the $k+1$th iteration, and this has to be kept in mind when interpreting the average values reported for an iteration.

In Figure 14, we report averages over all instances, but by the discretization parameter and iteration, of gaps for both these measures. Specifically, we report the gap in value between the primal solution produced by Solve-CTSNDP at an iteration and the primal solution produced at termination (labeled $\Delta$ min primal). Similarly, we report the gap in value between the dual bound produced by Solve-CTSNDP at an iteration and the dual bound produced at termination (labeled $\Delta$ min dual). (We note again that because not all executions of Solve-CTSNDP require the same number of iterations to terminate, for each iteration, we are reporting averages over the instances that needed at least that many iterations to terminate.) First, we observe that both gaps converge fairly quickly, with the dual bound converging more quickly than the objective function value of the primal solution. Importantly, we note that the performance of Solve-CTSNDP with respect to

**Figure 14.** Primal and Dual Gaps by Iteration



both the quality of the primal solution and the strength of the dual bound produced at an iteration is consistent across discretizations. Coupled with the results discussed previously, we conclude that Solve-CTSNDP is robust with respect to the discretization parameter, $\Delta$.

### 5.4. Potential of Solve-CTSNDP in Practice

As a final test of the effectiveness of the algorithm, we created five instances using data from a large less-than-truckload freight transportation carrier in the United States. While this carrier operates in nearly all states, we limited our instances to activities in the northwest region of the United States. We construct the instances using five days worth of freight. When optimizing how freight moves within this region, we include freight that originates outside the region but is destined for a terminal inside the region and freight that originates inside the region but is destined for a terminal outside the region. This is done by exploiting knowledge of the carrier's planned path for routing the freight. As an example, consider freight that originates on day 1 in Miami, Florida, and is destined for Seattle, Washington on day 5. The carrier's planned path for that freight enters the Northwest in Boise, Idaho on day 3. We model that freight as freight that originates in Boise on day 3 and is due in Seattle on day 5.

Because of the operational practices of this particular carrier, these instances differ in several ways from those used in our earlier computational experiments. Specifically, it is assumed that on the day that freight becomes available, it becomes available at 7 p.m. at its origin terminal (as that is when the carrier expects trucks to return from pickup and delivery routes). Thus, by considering a five-day planning horizon, freight only originates at five different time points. Similarly, it is assumed that on the day freight is due at the customer destination, it must be at the associated terminal at 8 a.m. (as that is when the carrier needs it to be ready for the pickup and delivery route). Finally, we note that the carrier quotes service (how long freight will take from origin to destination) to customers in terms of days. Thus, the time the carrier has to deliver freight also has a discrete nature. It was determined that, based on discussions with the carrier, 30 minutes is the finest discretization that would yield plans that could be implemented in practice. We ran both Solve-CTSNDP and FD for two hours on each instance and report the results of their execution, as well as data regarding each instance, in Table 4. We calculate "Delivery window (avg.)" as $(\sum_{k\in\mathcal{K}} l_k - e_k)/|K|$, while "Planning horizon" is calculated as $\max_k(l_k - e_k)$. Finally, "Freight capacity fraction (avg.)" is calculated as

$$\frac{\sum_{k\in\mathcal{K}} q_k}{|K|} \Bigg/ \frac{\sum_{(i,j)\in\mathcal{A}} u_{ij}}{|A|}.$$

We report gaps in terms of objective function value of the primal solutions produced by each method. We calculate "Primal gap" as $(z_{\text{CTSNDP}} - z_{\text{FD}})/z_{\text{CTSNDP}}$, where $z_X$ represents the objective function value of the solution produced by method $X$.

We see that Solve-CTSNDP easily solves the instance with only four states but is unable to solve the larger instances (to optimality). However, it is able to produce a primal solution and a dual bound with a relatively small (provable) optimality gap. We also note that for the larger instances, Solve-CTSNDP reports an optimality gap of between 3% and 4% after a little over an hour, suggesting the algorithm may be used as a heuristic. Similarly, FD is only able to solve the smallest instance. However, FD produces a much larger provable optimality gap than Solve-CTSNDP after two hours on the larger instances, and, looking at the "primal gap," we see that Solve-CTSNDP is able to produce higher-quality solutions than FD.

Regarding our previous experiments, we note that for the instances generated with $\Delta = 30$, the average "delivery window" is 54.65, which is slightly larger than the average for the real-world-inspired instances. However, the planning horizon of the real-world-inspired instances (i.e., 240) is much longer than the planning horizon of the instances used in the previous experiments (i.e., 122 periods). Finally, we note that the commodities in the instances used in the previous experiments are, relatively speaking, much smaller, as the average "freight capacity fraction" for those instances is 0.16. We attribute the difficulty that

**Table 4.** Performance of Solve-CTSNDP on Instances Derived from Data from a U.S. LTL Carrier

| | | | | | | | Solve-CTSNDP | | FD | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| States | $\|N\|$ | $\|A\|$ | $\|K\|$ | Delivery window (avg.) | Planning horizon | Freight capacity fraction (avg.) | Time to termination (sec.) | Optimality gap (%) | Time to termination | Optimality gap (%) | Primal gap (%) |
| ID, MT, OR, WA | 10 | 54 | 224 | 34.22 | 240 | 1.25 | 30 | 0.92 | 213 | 0.94 | 0.26 |
| CO, ID, MT, OR, WA | 14 | 81 | 341 | 49.07 | 240 | 1.05 | 7,200 | 1.68 | 7,200 | 3.63 | −1.29 |
| CO, ID, MT, OR, WA, NV | 16 | 109 | 469 | 52.11 | 240 | 0.94 | 7,200 | 2.74 | 7,200 | 25.28 | −28.26 |
| CO, ID, MT, OR, WA, UT | 15 | 104 | 458 | 50.28 | 240 | 1.00 | 7,200 | 1.45 | 7,200 | 19.71 | −20.64 |
| ID, MT, OR, WA, NV, UT | 13 | 97 | 429 | 43.88 | 240 | 1.05 | 7,200 | 3.00 | 7,200 | 23.78 | −25.18 |

Solve-CTSNDP and FD have in solving the two larger real-world-inspired instances to the large number of commodities and the length of the planning horizon, as both have a direct impact on the size of the integer program solved at each iteration.

## 6. Conclusions and Future Work

We have presented an algorithm for solving service network design problems that uses time-expanded networks but that avoids having to introduce approximations (and thus uncertainty about the quality of the solution) to keep the size of the time-expanded network manageable.

Because time-expanded networks are commonly used in the solution of many transportation problems, we hope and expect that many of the fundamental ideas underlying our approach can be applied in other contexts as well.

Our computational study has demonstrated that the current implementation of Solve-CTSNDP is quite effective. However, there are many enhancements that will increase its efficiency. We mention only a few. First, it is unnecessary to solve SND($\mathscr{D}_{\mathscr{T}}$), which is the most time-consuming step in the algorithm, to optimality at each iteration. In the initial iterations, it may suffice to only solve the linear programming relaxation or to solve the problem to within a proven percentage of optimality. Second, if SND($\mathscr{D}_{\mathscr{T}}$) does not change much from one iteration to the next, it may be possible to construct a high-quality initial solution to speed up the solution time.

The ability to solve very large instances of a service network design problem also opens up the possibility to study new and innovative time-focused service network design models. For example, a carrier may be interested in understanding the degree to which altering the available and/or due times of a commodity impacts the costs. Such decisions can easily be incorporated in a service network design model but will increase the size of instance substantially. However, by using an iterative refinement algorithm based on partially time-expanded networks, the increase in size may be overcome.

Finally, we note that many real-world service network design problems are computationally intractable because of their scale on two dimensions: (1) time and (2) number of shipments. For the carrier that inspired this work, modeling a week's worth of actions in its network on a time-space network based on a 30-minute discretization of time would yield a network with nearly 40,000 nodes and over 1.2 million arcs. We believe that the algorithm presented in this paper presents a significant step forward in terms of tackling this explosion in size. However, the number of shipments (which would map to commodities in a service network design model) for the carrier over a five-day span would exceed 60,000. Such a large number of commodities would likely render the integer programs solved by the algorithm presented in this paper computationally intractable. As a result, in future work, we intend to develop a similar algorithm for an explosion in size along this dimension.

### Acknowledgments

The authors thank two anonymous reviewers for their helpful comments and insights.

### References

Andersen J, Christiansen M, Crainic TG, Grønhaug R (2011) Branch and price for service network design with asset management constraints. *Transportation Sci.* 45(1):33–49.

Anderson EJ, Nash P (1987) *Linear Programming in Infinite-Dimensional Spaces: Theory and Applications* (John Wiley & Sons, New York).

Anderson E, Nash P, Philpott A (1982) A class of continuous network flow subproblems. *Math. Oper. Res.* 7(4):501–514.

Baumann N, Skutella M (2006) Solving evacuation problems efficiently—Earliest arrival flows with multiple sources. *Proc. 47th Annual IEEE Sympos. Foundation Comp. Sci., FOCS '06* (IEEE Computer Society, Washington, DC), 399–410.

Burkard RE, Dlaska K, Klinz B (1993) The quickest flow problem. *Zeitschrift für Oper. Res.* 37(1):31–58.

Crainic T (2000) Service network design in freight transportation. *Eur. J. Oper. Res.* 122(2):272–288.

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

Crainic T, Gendron B, Hernu G (2004) A slope scaling/Lagrangean perturbation heuristic with long-term memory for multicommodity capacitated fixed-charge network design. *J. Heuristics* 10(5):525–545.

Crainic TG, Hewitt M, Toulouse M, Vu DM (2016) Service network design with resource constraints. *Transportation Sci.* 50(4): 1380–1393.

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

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

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

Fleischer L, Skutella M (2003) Minimum cost flows over time without intermediate storage. *Proc. 14th Annual ACM-SIAM Sympos. Discrete Algorithms, SODA '03* (SIAM, Philadelphia), 66–75.

Fleischer L, Skutella M (2007) Quickest flows over time. *SIAM J. Comput.* 36(6):1600–1630.

Ford LR, Fulkerson DR (1958) Constructing maximal dynamic flows from static flows. *Oper. Res.* 6(3):419–433.

Ford LR, Fulkerson DR (1962) *Flows in Networks* (Princeton University Press, Princeton, NJ).

Gale D (1958) Transient flows in networks. Technical report, Defense Technical Information Center, Fort Belvoir, VA.

Ghamlouch I, Crainic T, Gendreau M (2003) Cycle-based neighborhoods for fixed charge capacitated multicommodity network design. *Oper. Res.* 51(4):655–667.

Ghamlouch I, Crainic T, Gendreau M (2004) Path relinking, cycle-based neighborhoods and capacitated multicommodity network design. *Ann. Oper. Res.* 131(1):109–133.

Groß M, Skutella M (2012) Maximum multicommodity flows over time without intermediate storage. Epstein L, Ferragina P, eds.

*Algorithms—ESA* 2012*: Proc. 20th Annual Eur. Sympos.* (Springer, Berlin), 539–550.

Groß M, Kappmeier JPW, Schmidt DR, Schmidt M (2012) Approximating earliest arrival flows in arbitrary networks. Epstein L, Ferragina P, eds. *Algorithms—ESA* 2012*: Proc. 20th Annual Eur. Sympos.* (Springer, Berlin), 551–562.

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

Hewitt M, Nemhauser G, Savelsbergh M (2010) Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem. *INFORMS J. Comput.* 22(2):314–325.

Hewitt M, Nemhauser G, Savelsbergh MW (2013) Branch-and-price guided search for integer programs with an application to the multicommodity fixed-charge network flow problem. *INFORMS J. Comput.* 25(2):302–316.

Hoppe B, Tardos É (1994) Polynomial time algorithms for some evacuation problems. Sleator DD, ed. *Proc. Fifth Annual ACM-SIAM Sympos. Discrete Algorithms, SODA '94* (SIAM, Philadelphia), 433–441.

Hoppe B, Tardos É (2000) The quickest transshipment problem. *Math. Oper. Res.* 25(1):36–62.

Jarrah A, Johnson E, Neubert L (2009) Large-scale, less-than-truckload service network design. *Oper. Res.* 57(3):609–625.

Jarvis JJ, Ratliff HD (1982) Some equivalent objectives for dynamic network flow problems. *Management Sci.* 28(1):106–109.

Katayama N, Chen M, Kubo M (2009) A capacity scaling heuristic for the multicommodity capacitated network design problem. *J. Comput. Appl. Math.* 232(1):90–101.

Kennington JL, Nicholson CD (2010) The uncapacitated time-space fixed-charge network flow problem: an empirical investigation of procedures for arc capacity assignment. *INFORMS J. Comput.* 22(2):326–337.

Klinz B, Woeginger GJ (2004) Minimum-cost dynamic flows: The series-parallel case. *Networks* 43(3):153–162.

Megiddo N (1974) Optimal flows in networks with multiple sources and sinks. *Math. Programming* 7(1):97–107.

Minieka E (1973) Maximal, lexicographic, and dynamic network flows. *Oper. Res.* 21(2):517–527.

Powell WB, Jaillet P, Odoni A (1995) Stochastic and dynamic networks and routing. Ball MO, Magnanti TL, Monma CL, Nemhauser GL, eds. *Handbooks in Operations Research and Management Science*, Vol. 8 (Elsevier, Amsterdam), 141–295.

Savelsbergh M (1986) Local search for routing problems with time windows. *Ann. Oper. Res.* 4(1):285–305.

Schulz JD (2014) 2014 state of logistics: Less-than-truckload's welcomed rebound. *Logistics Management* (July 1), http://www.logisticsmgmt.com/article/state_of_logistics_less_than_truck-loads_welcomed_rebound.

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

Tjandra SA (2003) Dynamic network optimization with application to the evacuation problem. PhD thesis, Universitat Kaiserslautern, Kaiserslautern, Germany.

Topaloglu H, Powell WB (2006) Dynamic-programming approximations for stochastic time-staged integer multicommodity-flow problems. *INFORMS J. Comput.* 18(1):31–42.

Wang X, Regan AC (2002) Local truckload pickup and delivery with hard time window constraints. *Transportation Res. Part B: Methodological* 36(2):97–112.

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

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

Yaghini M, Rahbar M, Karimi M (2012) A hybrid simulated annealing and column generation approach for capacitated multicommodity network design. *J. Oper. Res. Soc.* 64(7):1010–1020.

**Natashia Boland** is a professor at the H. Milton Stewart School of Industrial and Systems Engineering at Georgia Institute of Technology. Her research focuses on integer programming. She has over 20 years of experience in modeling, theory, algorithms and applications. Her current research interests include multiobjective integer programming, time discretization in integer programming, and decomposition methods in integer and stochastic integer programming.

**Mike Hewitt** is an associate professor at the Quinlan School of Business, Loyola University Chicago. His research interests include developing optimization methods for solving large-scale planning problems that arise in practice in the freight and small package transportation industries; this paper focuses on the temporal dimension of these problems.

**Luke Marshall** is a PhD student at the H. Milton Stewart School of Industrial and Systems Engineering at Georgia Institute of Technology. His PhD research focuses on dynamic and continuous-time service network design.

**Martin Savelsbergh** is the James C. Edenfield Chair and Professor at the H. Milton Stewart School of Industrial and Systems Engineering at Georgia Institute of Technology. He is an optimization and logistics specialist with over 25 years of experience in mathematical modeling, optimization methods, algorithm design, logistics, supply chain management, and transportation systems. His current research interests include multiobjective integer programming, time discretization in integer programming, and last-mile logistics.