European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing, Transportation and Logistics

Scalable timing-aware network design via lagrangian decomposition

Cristiana L. Lara [a,*], Jochen Koenemann [a,b], Yisu Nie [d], Cid C. de Souza [a,c]

[a] Modeling and Optimization, Amazon.com, 425 106th Ave NE, Bellevue, WA, 98004, USA
[b] Department of Combinatorics & Optimization, University of Waterloo, Waterloo, ON, N2L 3G1, Canada
[c] Institute of Computing, University of Campinas, Av. Albert Einstein no 1251, Campinas, SP, 13083-852, Brazil
[d] Prime Video, Amazon.com, 2021 7th Ave, Seattle, WA, 98121, USA

ARTICLE INFO

Article history:
Received 18 January 2022
Accepted 11 January 2023
Available online xxx

Keywords:
Transportation
Temporal fixed-charge multi-commodity flow
Resource task network
Task scheduling
Lagrangian decomposition

ABSTRACT

This paper addresses instances of the *temporal fixed-charge multi-commodity flow* (tfMCF) problem that arise in a very large scale dynamic transportation application. We model the tfMCF as a discrete-time Resource Task Network (RTN) with cyclic schedule, and formulate it as a mixed-integer program. These problems are notoriously hard to solve due to their time-expanded nature, and their size renders their direct solution difficult. We exploit synergies between flows of certain commodities in the formulation to devise *model condensation* techniques that reduce the number of variables and constraints by a factor of 25%–50%. We propose a solution algorithm that includes balanced graph partitioning, Lagrangian decomposition and a linear programming filtering heuristic. Computational results show that the proposed algorithm allows the solution of previously intractable instances, and the primal solution obtained by the heuristic step is within 2% duality gap of the linear relaxation of the original problem.

1. Introduction

The growth in online retailing has disrupted shopping patterns and directly impacted logistic networks. More than 85% of online shoppers consider shipping speed a top priority in the decision to shop online (Cui, Lu, Sun, & Golden, 2020). The demand for quick delivery requires the outbound transportation network to be operated efficiently around-the-clock in order to achieve better shipping speed in a cost-effective manner. This necessitates taking into account timing information when making decisions to accurately model intra-day capacity needs when designing the network. We formulate this problem as a *temporal fixed-charge multicommodity flow problem*.

In the classical *fixed-charge multicommodity flow problem* (fMCF), the input consists of a (directed or undirected) graph $G = (V, E)$, non-negative edge costs F_e, capacities u_e for all $e \in E$, and a set $\mathcal{C}$ of commodities (o_c, d_c), each having an integer demand θ_c. The goal is to install $y_e \in \mathbb{N}$ units of capacity on each edge $e \in E$ (where a unit of capacity for edge e is u_e) so that the resulting capacitated graph admits a feasible multicommodity flow transporting all demands. The unit-capacity installation cost for each edge $e \in E$ is F_e, and the goal is to minimize the total capacity installation cost. The fMCF problem is NP-hard as the well-known NP-hard *Steiner tree* problem is a special case (see Karp (1972)).

The fMCF problem stated above is *static* in nature in that its focus is on *one-shot* decisions about capacity installation in an underlying fixed network. The focus of this paper is on *dynamic* aspects of network design, where the timing of when to move trucks and sort packages needs to be decided. We are interested in extensions of fMCF where for each integral capacity installation, we now also require to know *when* this capacity is supposed to be installed. In the end, we aspire to find timed capacity installations that allow for the existence of a feasible multicommodity flow *over time*. Such dynamic models reflect the reality of many practical decision processes, where the detailed timing of actions is crucial, and where decisions at a certain point in time impact future ones. Dynamic models have been widely studied, and we provide a short review and pointers to the extensive literature in Section 1.2.

Models capturing temporal aspects of scheduling traditionally either use a *discrete* or *continuous* representation of time. Continuous time formulations tend to be more compact in comparison to their discrete counter-parts. Continuous formulations are, however, known to have weak linear programming (LP) relaxations which, using current IP solver technology, limits their use to the solution of small instances only. Discrete formulations with variables indexed by time tend to have stronger relaxations (e.g., see Pochet & Warichet, 2008). They are also more convenient in modeling synchronized events and shared resources. However, in order to obtain accurate models for the given timing application, a very large

* Corresponding author.
E-mail address: larcrist@amazon.com (C.L. Lara).

https://doi.org/10.1016/j.ejor.2023.01.018
0377-2217/

Acronyms

fMCF	Fixed multi-commodity flow problem
tfMCF	Temporal fixed-charge multicommodity flow problem
DDD	Dynamic Discretization Discovery
DS	Delivery station
FC	Fulfillment center
IB	Inbound
IP	Integer program
LB	Lower bound
LP	Linear program
MIP	Mixed-integer program
OB	Outbound
OD	Origin-destination pair
RTN	Resource task network
SC	Sort center
SND	Service network design
UB	Upper bound

Parameters

(o_c, d_c)	*(origin, destination)* pair of commodity c
$\text{arrival}^{UB}_{v,j,d,t}$	Cumulative number of packages ordered by ODs in bucket $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$ between time 0 and time $(t - \text{minArrival}_{v,j,d})$
$\text{cube}^{\max}_{v}$	Max total volume of the OD among the ODs in a bucket
$\text{cube}^{\min}_{v}$	Min total volume of the OD among the ODs in a bucket
$\delta(\pi)$	Set of edges in graph G whose ends lie in different parts of π
ϵ	Condensation factor
$\gamma_{i,m}$	Conversion factor between unit of extent size m and unit of task capacity i
$\hat{\Phi}^{cost}$	Upper bound to the total line-haul cost
λ	Vector of Lagrangian multipliers
$\lambda_{m,t}$	Lagrange multiplier of the equality $b^{out}_{m,t} = b_{m,t}$
$\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$	Bucket of ODs with origin v, target $\tilde{t}$ and horizon $\tilde{h}$
$\text{minArrival}_{v,j,d}$	Minimum transit time of the path of any OD in bucket $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$
μ	Step-size parameter in the subgradient method
pkgCount^{e}	Upper bound to total number of packages in edge e
pkgCount_{c}	Packages to be shipped along path P_c of commodity c
$\text{pkgCount}_{v,j,d}$	Total number of packages of ODs in in bucket $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$ that have destination d
pkgCube_{c}	Total volume to be shipped along path P_c of commodity c
$\pi = (V_1, \ldots, V_q)$	Balanced partition of the vertex set $V(G)$
$\Pi_{r,t}$	Amount of flow injected into resource r at time $t \in \mathcal{T}_r$
$\sigma_{v,j}$	Average cube parameter for each bucket $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$
τ_i	Duration of task i
τ_m	Duration of task i corresponding to extent m
$\text{iter}^{LP} \in \{0, 1, \ldots, \text{iter}^{\max}\}$	Set of iterations for the Lagrangian decomposition to solve the LP relaxation
θ_c	Demand of commodity c
$\tilde{\Pi}_{\tilde{r},t}$	Amount of flow injected into condensed resource $\tilde{r}$ at time $t \in \mathcal{T}_{\tilde{r}}$
$\tilde{\sigma}$	Maximum of $\tilde{\sigma}_{v,j}$ over all buckets bucket $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$ if at least 1, or 1 otherwise
$\tilde{\sigma}_{v,j}$	Maximum ratio of $\sigma_{v,j}$ and the average cube of any commodity in the bucket
ζ	Rounding threshold in the heuristic
$B^{\max}_{i,t}$	Capacity of task i at time t
C_i	Cost of adding a vehicle for transportation task i
$d \in \mathcal{D}^{\tilde{t},\tilde{h}}_{v,j}$	Set of destinations d of ODs in bucket $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$
F_e	Edge cost
$G(V, E)$	Graph with vertices V and edges E
$m \in \mathcal{M}^{\tilde{t},\tilde{h}}_{v,j}$	Extent set of bucket $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$
N_c	Relevant horizon of commodity c in days
$N_{\tilde{r}}$	Relevant horizon of condensed resource $\tilde{r}$ in days
P_c	Path for commodity c
r^+_m	Resource fed by extent m
r^-_m	Resource depleted by extent m
$R^{\max}_{r,t}$	Upper bound to level of resource r at time $t \in \mathcal{T}_r$
$target_c$	Target arrival time of commodity c in days
$target_{\tilde{r}}$	Target arrival time of condensed resource $\tilde{r}$ in days
u_e	Edge capacity
w_e	Weight of edge e based on the number of paths P_c for OD pair $(o_c, d_c) \in \mathcal{C}$ with $e \in E_c$
$y_e \in \mathbb{N}$	Units of capacity of edge e, where a unit of capacity is u_e

Sets

$\mathcal{T}^{day} = \{0, \ldots, T-1\}$	Time period in a day, where T is the total number of time periods in a day
$\mathcal{T}_c = \{0, \ldots, N_c T - 1\}$	Time horizon of commodity c
$\mathcal{T}_m = \{0, \ldots, n^{day}_r T - 1\}$	Time horizon of extent m associated with commodity $c \in \mathcal{C}$
$\mathcal{T}_r = \mathcal{T}_c$	Time horizon of resources $r \in \mathcal{R}_i$ of commodity c
$\tilde{h} \in \mathcal{H}$	Set of horizon lengths
$\tilde{m} \in \tilde{\mathcal{M}}$	Set of condensed extents
$\tilde{m} \in \tilde{\mathcal{M}}^{\tilde{t},\tilde{h}}_{v,j,d}$	Set of condensed extents of ODs in bucket $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$ that have destination d
$\tilde{m} \in \tilde{\mathcal{M}}_e$	Set of condensed extents with transportation tasks on edge e
$\tilde{r} \in \tilde{\mathcal{R}}$	Set of condensed resources

$\tilde{t} \in \mathcal{TA}$	Set of targets
$c \in \mathcal{C}$ or $(o_c, t_c) \in \mathcal{C}$	Commodities
$e \in E$ or $uv \in E$	Edges in graph $G(V, E)$ with vertices u and v
$e \in E_c$	Set of edges in OD path P_c
$i \in \mathcal{I}$	Tasks
$i \in \mathcal{I}^{LM}$	Last-mile tasks
$i \in \mathcal{I}^{S}$	Sortation tasks
$i \in \mathcal{I}^{T}$	Transportation tasks
$i \in \mathcal{I}_p^{in,dest} \subset \mathcal{I}_p^{in}$	Subset of tasks that start in a vertex that belongs to region p, but have as destination a vertex outside the region
$i \in \mathcal{I}_p^{in}$	Set of tasks that start in a vertex in region p
$i \in \mathcal{I}_p^{out}$	Set of tasks that start in a vertex that is not in p, but end in a vertex of p
$j \in \mathcal{J}_v$	Set of indices j used in the logarithmic definition of OD buckets
$m \in \mathcal{M}_{r,p}^{+,in}$	Extents of tasks in $\mathcal{I}_p^{in}$ that feed resource r
$m \in \mathcal{M}_{r,p}^{+,out}$	Set of extents of tasks in $\mathcal{I}_p^{out}$ that feed resource r
$m \in \mathcal{M}_r^{+}$	Extents that feed resource r
$m \in \mathcal{M}_{r,p}^{-,in}$	Extents of tasks in $\mathcal{I}_p^{in}$ that depletes resource r
$m \in \mathcal{M}_r^{-}$	Extents that deplete resource r
$m \in \mathcal{M}_c$	Extents of commodity c
$m \in \mathcal{M}_i$	Extents of task i
$m \in \mathcal{M}_r^{+}$	Extents that feed resource r
$m \in \mathcal{M}_r^{-}$	Extents that deplete resource r
$p \in [q]$	Set of balanced partitions (regions)
$r \in \mathcal{R}^{inter}$	Intermediate resources for each commodity c
$r \in \mathcal{R}^{prod}$	Product resources for each commodity c
$r \in \mathcal{R}^{supply}$	Supply resources for each commodity c
$r \in \mathcal{R}_c$	Resources for each commodity c
$t \in \mathcal{T}_e$	Union of horizons of all ODs whose path uses edge e
$t \in \mathcal{T}_{\tilde{r}}$	Time horizon of condensed resource $\tilde{r}$ in days
$uv \in \mathcal{A}$	Arcs with origin vertex u and destination vertex v and transit time τ

Decision variables

Φ^{LB}	Lower bound for the optimum of (8)
Φ_p^{LR}	Optimum of the Lagrangian subproblem (7)
$b_{m,t}^{out}$	Duplicate of extent size variable $b_{m,t}$ for extent $m \in \mathcal{M}_i$ where corresponding transportation task i has its origin in a region different from that of its destination
$b_{\tilde{m},t}$	Size of condensed extent $\tilde{m}$ at time $t \in \mathcal{T}_{\tilde{m}}$
$b_{m,t}$	Size of extent m at time $t \in \mathcal{T}_m$
$R_{\tilde{r},t}$	Level of condensed resource $\tilde{r}$ at time $t \in \mathcal{T}_{\tilde{r}}$
$R_{r,t}$	Level of resource r at time $t \in \mathcal{T}_r$
$y_{i,t}$	Discrete occurrence of task $i \in \mathcal{I}^T$ at time $t \in \mathcal{T}^{day}$. It represents the number of vehicles installed each day.

number of time points, and hence (discrete) variables are needed. Since solvability is the top priority, we model the temporal nature of tfMCF using a fine-grained discrete, time-indexed model.

1.1. Problem statement

We propose a version of tfMCF that arises in very large scale dynamic transportation applications. In this setting, the vertices of the underlying graph $G = (V, E)$ correspond to *fulfilment centers* (FCs; warehouses that store the goods to be shipped), *sort centers* (SCs; intermediate locations that facilitate sortation), and *delivery stations* (DSs; terminal network nodes for packages), respectively. For each edge $uv \in E$, one is given the set of vehicle types available to transport packages between u and v, and their capacities. These vehicles can be of different transportation modes (e.g. air and ground). Once again, a set $\mathcal{C}$ of *commodities* is given at the input; we will sometimes refer to (o_c, d_c) as a *origin, destination* pair associated with commodity c, and write $(o_c, d_c) \in \mathcal{C}$ and $c \in \mathcal{C}$ interchangeably. In the context of this work, o_c will always be an FC, and d_c could either be an SC, or a DS. In the application of interest, each pair $(o_c, d_c) \in \mathcal{C}$ comes with a *shipment path* P_c connecting o_c to d_c, the number $\mathtt{pkgCount}_c$ of packages to be shipped along this path, and the total volume $\mathtt{pkgCube}_c$ of these packages (also referred as cube, interchangeably). Precise information on how the $\mathtt{pkgCount}_c$ packages of commodity (o_c, d_c) are released at o_c are provided, and this determines how packages for this commodity are released into the network over time. Finally, the traffic associated with (o_c, d_c) is supposed to be classified as *grouped* (say in shipping containers) or *ungrouped* (where packages are loaded into transportation vehicles by themselves). In the following, the terms *containerized* and *fluid-loaded* will be used sometimes to refer to grouped and ungrouped ODs, respectively. Whether an OD is grouped or ungrouped affects the volume of each *elementary unit* (package or container) associated with the commodity, the processing time at intermediate vertices, and the utilization of transport vehicles, among other things.

The overall goal of the proposed model is to compute a feasible multicommodity flow over time that maximizes *speed*, where the latter is measured by the number of packages (over all commodities) that arrive *on time* (i.e., it arrives at the destination within $target_c$ days, where the latter is a given non-negative integer parameter). The main decisions to be optimized are the transportation schedule (timing and lot sizing), and the hourly throughput at each node. For operational consistency, the optimal transportation schedule is repeated day-over-day (cyclic schedule). We will provide more details on this in Section 2.4.

1.2. Literature review

The history of the Steiner tree problem (and therefore the history of work on fixed charge network design problems) goes back to a problem posed by Fermat, and was first defined by Gauss in a letter to one of his students (see page 37 in Vazirani, 2001). It first was coined the *Steiner* tree problem in much later work by Courant & Robbins (1941). The Steiner tree problem is one of Karp's 21 original NP-hard problems (Karp, 1972). The more general class of fixed charge network design problems was first defined in an articles of Hirsch & Dantzig (1968), a technical report version of which appeared in 1954. Single and multi-commodity versions, respectively, of the fixed charge network design appeared first in seminal papers of Balinski (1961), and Gomory & Hu (1961).

Research efforts following the definition of the problem in the above mentioned early work are summarized in two survey papers by Magnanti & Wong (1984) and Minoux (1989). The paper of Magnanti and Wong formally defines an uncapacitated version of fMCF. We refer the reader to a recent and up to date survey on fixed charge network design in Chapter 3 of the recent book by Crainic, Gendreau, & Gendron (2021).

Temporal aspects of network design were first studied by Ford and Fulkerson in their seminal work in Ford & Fulkerson (1958, 1962). The authors focused on network flows over time, and more specifically on the single-commodity maximum flow special case. In their work, the authors introduced a so called *time-expanded* version of the underlying base graph. These static networks have multiple *layers*, one for each discrete time step, and each containing a copy of the node set of the underlying base graph. An arc $uv \in \mathcal{A}$ with transit time τ in the base graph gives rise to several *copies* in the time expanded graph, one for each pair of copies of u and v that are in layers of temporal distance τ. Static flows in the time expanded graph correspond to dynamic flows in the base graph and vice-versa; we refer the reader to a thorough discussion of this correspondence in the work of Fleischer & Tardos (1998).

In many applications, the size of the time-expanded network for a given base graph has pseudo-polynomial rather than polynomial size. This in turn often means that discrete formulations based on these time-expanded networks have super-polynomial size as well. Thus, using such a formulation to obtain a practical algorithm for a given problem is often challenging. In fact, temporal versions of combinatorial optimization problems are often provably more difficult than their static counterparts: Klinz & Woeginger (2004) showed that the dynamic version of the minimum-cost (o, d)-flow problem is (weakly) NP-hard. Hall, Hippler, & Skutella (2007) later also showed that dynamic variants of the most basic (unweighted) multicommodity flow problems are (weakly) NP-hard. Subsequently, Fleischer & Skutella (2007) introduced carefully constructed, polynomial-sized *partial* versions of complete time expanded networks to obtain a fully-polynomial-time approximation scheme for the (weakly NP-hard) *quickest* multicommodity flow problem over time. For a more thorough survey of the literature in the area of dynamic flows, we refer the reader to the recent work of Skutella (2009).

The fMCF problem broadly falls into the class of *service network design* (SND) problems. Roughly speaking, in these problems we are given origin and destination nodes corresponding to customers, in an underlying network. Customers are the recipients of a service which, for us, corresponds to the timely and efficient transportation of goods from the customers' origins to their respective destinations. Transportation is accomplished by a service provider which, in our setting, is a *consolidation-based* freight carrier; i.e., normally, the goods transported for one customer do not consume all of a standard transportation vehicle. Hence, goods from several customers can be consolidated to achieve transportation efficiency. The overall objective in typical SND instances is to minimize the total cost of providing service.

Mathematical programming, and in particular mixed integer linear programming, are among the most widely used tools in formulating and solving SND problems. Most prominent formulations extend classical fixed-charge, capacitated multicommodity flow models as described above (see Crainic, 2000; Crainic, 2003 and Wieberneit, 2008). SND problems instances arise in *static* and *dynamic* variants. Static SND problem instances are assumed to have no time-dependent problem aspects. In dynamic SND instances on the other hand, aspects of the model are time-dependent; e.g., like in our setting, the traversal of an arc in the network is assumed to take a certain non-negative transit time. The main classes of mathematical models used for dynamic SND problems are either based on continuous and compact formulations, or they inherently rely on the time-expanded version of the underlying graph as discussed above. As mentioned, both classes of formulations have drawbacks: continuous formulations are known to be compact but weak, while discrete, time-expanded formulations are known to have strong relaxations that are hard to solve because of their size (e.g., see Boland & Savelsbergh, 2019). We also refer the reader to Floudas & Lin (2004, 2005) for a comparison of discrete and continuous time models for scheduling applications.

Static and especially dynamic versions of fixed-charge multicommodity flow problems are known to be extremely challenging. As noted, one reason for this is the pseudo-polynomial size of the time-expanded formulation. Another reason lies in the fact that these models are often highly degenerate. Not surprisingly, classical decomposition techniques for integer programming formulations like Lagrangian relaxation, Benders and Dantzig-Wolfe reformulations are useful in this context as well. An in-depth discussion of these topics is beyond the scope of this paper. We refer the reader to standard text books on integer programming (Conforti, Cornuéjols, & Zambelli, 2014; Nemhauser & Wolsey, 1988) as well as the treatment in (Chapter 3 of) the excellent recent survey in Crainic et al. (2021).

In their work, Boland, Hewitt, Marshall, & Savelsbergh (2017) observed that optimal solutions for practical SND instances are often supported in rather small subgraphs of the time-expanded graph of the model. Motivated by this, the authors present an iterative algorithm – the so called *dynamic discretization discovery* (DDD) method. This algorithm starts with a small subgraph of the time-expanded graph, and solves a *lower-bound* model whose optimum value is guaranteed to be no larger than that of the given SND problem instance. Given the solution to this lower-bound model, the authors then either convert it into a feasible solution for the original problem, or show how to augment the partial time-expanded graph in order to obtain a stronger lower-bound model. Boland et al. (2017) provide empirical evidence of the effectiveness of the DDD method using a family of SND instances described in previous work by Crainic, Frangioni, & Gendron (2001). Since its invention, the DDD method has been applied to numerous temporal problems; e.g., see the work of Hewitt, Crainic, Nowak, & Rei (2019) on SND instances that arise in the less-than-truckload freight transportation setting, or more recent work of Lagos, Boland, & Savelsbergh (2020, 2022) on continuous time versions of the inventory routing problem.

Temporal modeling is also of essential importance in the chemical process industry (Wassick & Ferrio, 2011). Grossmann, Quesada, Raman, & Voudouris (1996) as well as Pinto & Grossmann (1998) provide an overview of the use of mathematical programming in the area of chemical batch processing for production planning and scheduling. Especially important in this context is the work of Kondili, Pantelides, & Sargent (1993) who develop a general framework to represent processes. The latter work was extended later by Pantelides (1994) who introduced the flexible *resource-task network* framework for the modeling of complex interconnected processes. In this paper, we present a discrete RTN-based mathematical model for tfMCF. The specific problem at hand in this paper requires our model to encode cyclic temporal capacity installation plans. Schedule cyclicity, especially in the context of RTN models is a well known model feature. We refer the reader to previous work of Yee & Shah (1998) who develop integer programming models for periodic production scheduling problems. Castro, Barbosa-Póvoa, & Matos (2003), Castro, Barbosa-Póvoa, & Novais (2005) present RTN-based, discrete and continuous models for periodic scheduling problems arising in the chemical industry. In more recent work Wu & Maravelias (2019), Wu and Maravelias propose a *state task network* integer programming model for a periodic production scheduling application. For an extensive survey on previous work on scheduling applications in the process industry, we refer the reader to the survey by Harjunkoski et al. (2014).

Another recent example of temporal network modeling can be found in the work of Hoch, Liers, Neumann, & Martınez (2020), where the authors discuss disjoint path problems in temporal graphs that arise in aircraft trajectory planning applications. The main focus of the latter paper lies in studying complexity issues.

Finally, Adjiashvili, Bosio, & Weismantel (2012), Adjiashvili, Bosio, Weismantel, & Zenklusen (2014) introduced dynamic counterparts of a class of packing problems. The authors study the complexity of these problems, and present approximation algorithms.

We conclude this brief review section, mentioning that the literature on static and dynamic network design is much too large to adequately cover in this paper. There are several excellent surveys in the area that can be consulted by the interested reader: Powell, Jaillet, & Odoni (1995) provide an in-depth discussion on the use of time-expanded networks in formulations arising in logistic planning models. The very recent book by Crainic et al. (2021) provides a thorough survey of the state of the art in network design when applied to transportation and logistics; Chapters 2, 3, and 12 of the book are particularly relevant for this paper as they focus of static and dynamic fixed-charge network flow problems and algorithms.

1.3. Gaps in the literature

While there exists a large body of research on time-expanded networks, temporal fixed-charge multi commodity flow and resource task network, to the best of our knowledge none of the existing literature attempts to solve problems of the size of our instances, taking advantage of specific properties of the problem such as its geometric nature and synergies between commodities. Our work is motivated by the need to capture operationally relevant constraints that are not present in the classic versions of these problems, and have a scalable solution methodology that can provide a good feasible solution and valid optimality guarantees to instances of the order of 100M+ variables and constraints. The modeling and solution techniques proposed in this paper were developed with this problem in mind, but can be extended to other real-world applications of tfMCF.

1.4. Contributions

This paper presents a mixed-integer-programming (MIP) model for tfMCF. MIP formulations encoding discrete timing models are well known to be very large and notoriously difficult to solve. Timing-aware optimization has become increasingly important in recent years, and much work has gone into the development of techniques for finding more efficient models (e.g., see Boland et al., 2017; Boland & Savelsbergh, 2019; Newman & Kuchta, 2007; Pessoa, Uchoa, de Aragão, & Rodrigues, 2010; Wang & Regan, 2002; Wang & Regan, 2009 for examples).

The proposed tfMCF model is extremely large, even for moderately-sized input graphs. For example, in the application considered in this study, while input graphs often have moderate size (up to a few thousand vertices and edges), the resulting mixed integer programs sometimes have up to a half billion variables and constraints. Hence, there is no hope to solve the resulting MIPs as is, even on the largest available hardware. The main contribution of this paper is to present several techniques that make use of inherent problem structure in order to reduce its size. We develop a solution algorithm based on our new model and demonstrate that its solution times are vastly superior to standard MIP based strategies. The main ingredients in the present work are:

1. the modeling of this *temporal fixed-charge multicommodity flow problem* (tfMCF) as a discrete-time Resource Task Network (RTN) (Pantelides, 1994) with cyclic schedule (Castro, Barbosa-Póvoa, & Matos, 2003).
2. the exploitation of synergies between flows of certain commodities in the formulation in order to devise *model condensation* techniques. This yields models whose number of variables and constraints are reduced by a factor of 25%–50%.
3. the utilization of the geometric nature of the instances in the application studied to break down instances into *regions* using balanced graph partitioning techniques (Karypis, Aggarwal, Kumar, & Shekhar, 1999). The resulting decomposition is then used in a Lagrangian decomposition-based framework to solve our IP. The objective function of the balanced decomposition step is designed to yield a small number of dualized constraints, when optimized. Partition-balance of the resulting cuts yields MIP subproblems that are similar in difficulty. Lagrangian subproblems are independent, and solved in parallel.
4. the development of a linear programming filtering heuristic that allows to *fix* certain decision variables based on solutions to linear relaxation of the Lagrangian subproblems. Computational experiments reveal that this heuristic yields a large number of fixed variables, and that this in turn results in significant improvements in the algorithm's running time.

The scalability of our implementation allows the solution of previously intractable instances.

A formal description of tfMCF's model, and its mixed integer program is provided in Section 2. Section 3 describes the proposed condensed formulation that bucketizes OD paths based on their origin vertex and average cube (as well as transportation mode and containerization decision), which considerably reduces the size of the model. The spatial partition algorithm based on *split Lagrangian* decomposition is discussed in Section 4 and, in Section 4.3, a heuristic leveraging the LP relaxation of the subproblems from early iterations of the *split Lagrangian* is proposed. Finally, Section 5 is devoted to the presentation and discussion of the results of the computational experiments.

2. A mixed-integer model for tfMCF

In this section, we develop a mixed integer programming formulation for tfMCF. As mentioned, to accomplish this, we cast tfMCF as an RTN. The versatility and generality of this framework makes it easy to add constraints to match evolving business operations. We introduce RTN in the context of tfMCF first.

2.1. Resource task networks

The main ingredients in RTNs are *resources* and *tasks*. Resources often represent *consumable materials* like goods, but also available labor, equipment, etc., while tasks *operate on resources* and thereby modify (i.e., deplete, feed, or alter otherwise) these.

In the case of tfMCF, we define sets $\mathcal{R}_c$ of resources for each commodity $c \in \mathcal{C}$, and we then let the set of all resources $\mathcal{R}$ be the union of the $\mathcal{R}_c$'s. We emphasize that $\mathcal{R}_c$ and $\mathcal{R}_{c'}$ are disjoint for any two distinct commodities $c, c' \in \mathcal{C}$. The resources for commodity (o_c, d_c) help us keeping track of packages that reside at critical points along the path P_c in the underlying network. More specifically, for each (o_c, d_c), we have resources for each vertex on path P_c. We will later call $r \in \mathcal{R}_c$ a *supply* resource if it corresponds to vertex o_c and if it is the *starting point* of any package order associated with commodity c. Similarly, we will say that $r \in \mathcal{R}_c$ is a *product* resource if it corresponds to vertex d_c, and if it is the final resource a package belonging to commodity c is associated with. We will also call all other, non-supply and non-product resources *intermediate*. We let $\mathcal{R}^{supply}$, $\mathcal{R}^{prod}$, and $\mathcal{R}^{inter}$ be the corresponding sets of all supply, product, and intermediate resources. We will also $\mathcal{R}^X_c$, for $X \in \{supply, prod, inter\}$ for sets of resources corresponding to a particular commodity c. Our model defines a set $\mathcal{I}$ of *tasks* that is further subdivided into

- *Transportation tasks* $\mathcal{I}^T$. Such a task is associated with a physical transportation process corresponding to an arc uv in the underlying transportation network. Nodes u and v correspond to ge-

ographic locations, and the task represents the process of moving goods from u to v. In terms of the underlying RTN, a transportation process depletes the resource at node u, and feeds that at node v. In this paper, transportation tasks are associated with specific vehicle types, and this determines, for example, the throughput capacity of the task.
- *Sortation tasks* $\mathcal{I}^S$. These tasks model processing that needs to happen at the various vertices in our network, like splitting the flow of incoming packages into various output streams, depending on their destination. Separate sortation tasks are defined for containerized and fluid-loaded packages.
- *Last-mile tasks* $\mathcal{I}^{LM}$. These tasks model processing that is applied to packages between the destination network node and final customer delivery. Last-mile tasks are introduced to model the end-to-end transit time of package flows.

Each task i has a fixed *duration* and a set of associated *task extents* $\mathcal{M}_i$. A task extent is an extension of the concept of batch size, allowing a single task to operate on multiple disjoint sets of resources (Wassick & Ferrio, 2011). In this case, the task involves multiple operations that are synchronized in time, but the amount of resources processed (throughput) by each operation can be optimized independently. In our context, it can be interpreted as the flow associated with a commodity going through its associated task at each time period. The presence of multiple extents for each task models the fact that the material flow associated with various commodities may share the capacity of a transport vehicle or a sortation process at a certain time. For ease of notation, we let τ_m be the duration of the task i corresponding to extent m, and let $t \in \mathcal{T}_m$ be the horizon of the extent m associated with commodity $(o_c, d_c) \in \mathcal{C}$.

Extents interact with certain resources. In the case of tfMCF this interaction is quite special, and we exploit this here to make notation easier (in comparison to general RTNs defined in Pantelides (1994) and Wassick & Ferrio (2011). We refer readers to Section Appendix A in the appendix for details). In our formulation setting, every extent interacts with exactly two resources of a single OD; let these be r_m^-, and r_m^+. Extent m *depletes* resource r_m^- (corresponding to the usual *interaction parameter* of -1), and it *feeds* resource r_m^+ (corresponding to an interaction parameter of $+1$). For a resource $r \in \mathcal{R}$ we let $\mathcal{M}_r^-$ be those extents that deplete r, and similarly, $\mathcal{M}_r^+$ is the set of extents that feed r. From the above it follows that each extent $m \in \mathcal{M}$ belongs to a unique commodity $c \in \mathcal{C}$. We can therefore define $\mathcal{M}_c$ to be the set of extents of commodity c, and once again observe that $\mathcal{M}_c$ and $\mathcal{M}_{c'}$ are disjoint for distinct commodities c and c'. The set of all extents $\mathcal{M}$ is the disjoint union of the $\mathcal{M}_c$ sets, over all commodities c.

2.2. Timing assumptions

We adopt a discrete time representation where the modeled time horizon is divided into slots of uniform length. Each day has a set $\mathcal{T}^{day}$ of discrete time units $0, \dots, T-1$, where *events* (i.e., arrivals, departures, etc.) may happen. We define the overall *time horizon* of commodity c as the set $\mathcal{T}_c$; i.e.

$$\mathcal{T}_c = \{0, \dots, N_c T - 1\},$$

where N_c is the relevant *horizon*, in number of days, during which we are expected to model each commodity c. N_c should be large enough to accommodate the maximum transit hours needed for commodity c. Notice that the length of the time horizon can differ between OD pairs to avoid generating unnecessary time indices, and, in turn, variables and constraints. Each commodity c specifies a parameter $target_c \le N_c$, and a package for commodity c is considered to be delivered *on time*, whenever it arrives at the destination within $target_c$ days. We refer to *speed* as the number of days between when the package is injected into supply resource $r \in \mathcal{R}^{supply}$ and when it reaches its product resource $r \in \mathcal{R}^{prod}$. The cyclicity assumptions are defined in Section 2.4

2.3. A mixed-integer programming formulation

Throughout this paper, we make use of the fact that each resource and extent corresponds to a distinct commodity, and associate certain commodity-specific parameters and sets with resources and extents, respectively. In particular, we will sometimes write X_r and X_m in place of X_c for $X \in \{target, N, \mathcal{T} \dots\}$, whenever $r \in \mathcal{R}_c$, and $m \in \mathcal{M}_c$.

tfMCF is a cost constrained *speed maximization* problem, formulated as a mixed-integer linear program (MIP). The MIP uses continuous variables $R_{r,t}$ encoding the level of resource r at time $t \in \mathcal{T}_r$. These variables are constrained to have value no larger than a parameter $R_{r,t}^{\max}$. We also have continuous variables $b_{m,t}$ representing the size of extent m at time $t \in \mathcal{T}_m$. Our model also has discrete variables $y_{i,t}$ for transportation tasks $i \in \mathcal{I}^T$, and times $t \in \mathcal{T}^{day}$ encoding the resource provisioning of transportation task i at time t. In a solution, the value of $y_{i,t}$ will represent the number of vehicles installed for $i \in \mathcal{I}^T$ at time t, *each day*.

The MIP's objective function is designed to maximize fulfillment with an emphasis on *speed*, and this is accomplished through its two terms:

(i) $\sum_{r \in \mathcal{R}^{prod}} R_{r,N_r T-1}$ – which describes the total flow arriving within the respective commodity's horizon, and hence encodes the goal to maximize fulfillment, and
(ii) $\sum_{r \in \mathcal{R}^{prod}} R_{r,target_r T-1}$ – which describes the total flow arriving on time, and hence encodes the speed objective.

As we will see shortly, feasibility implies that flow that contributes to the speed term (ii) above, will also contribute to term (i). The implied double counting emphasises speed; i.e., it emphasises our primary goal of maximizing on-time fulfillment.

We introduce parameters $\Pi_{r,t}$ to denote the amount of flow (possibly 0) that is *injected* into (supply) resource r, at time t. For each task $i \in \mathcal{I}$, and corresponding extent m, we define *conversion parameters* $\gamma_{i,m}$ to account for multiple units of measure used for different types of processing capacities. Finally, $\hat{\Phi}^{cost}$ is a given cost upper bound of transport vehicles. We also define the initial resource level $R_{r,-1}$ as 0 for all resources $r \in \mathcal{R}$.

$$\max \sum_{r \in \mathcal{R}^{prod}} \left(R_{r,target_r T-1} + R_{r,N_r T-1} \right) \tag{1a}$$

$$\text{s.t.} \quad R_{r,t} = R_{r,t-1} + \sum_{m \in \mathcal{M}_r^+} b_{m,t-\tau_m} - \sum_{m \in \mathcal{M}_r^-} b_{m,t} + \Pi_{r,t} \quad \forall r \in \mathcal{R}, t \in \mathcal{T}_r \tag{1b}$$

$$\sum_{m \in \mathcal{M}_i} \sum_{t' \in \mathcal{T}_m : t' \equiv t \bmod T} \gamma_{i,m} b_{m,t'} \le B_{i,t}^{\max} \quad \forall i \in \mathcal{I}^S \cup \mathcal{I}^{LM}, t \in \mathcal{T}^{day} \tag{1c}$$

$$\sum_{m \in \mathcal{M}_i} \sum_{t' \in \mathcal{T}_m : t' \equiv t \bmod T} \gamma_{i,m} b_{m,t'} \le B_{i,t}^{\max} y_{i,t} \quad \forall i \in \mathcal{I}^T, t \in \mathcal{T}^{day} \tag{1d}$$

$$0 \le R_{r,t} \le R_{r,t}^{\max} \quad \forall r \in \mathcal{R}, \ t \in \mathcal{T}_r \tag{1e}$$

$$\Phi^{cost}(y) := \sum_{i \in \mathcal{I}^T} \sum_{t \in \mathcal{T}^{day}} C_i y_{i,t} \le \hat{\Phi}^{cost} \tag{1f}$$

$$y_{i,t} \in \mathbb{Z}_+ \quad \forall i \in \mathcal{I}^T, t \in \mathcal{T}^{day} \tag{1g}$$

Constraint (1b) expresses *resource balance*: the level of resource r at time t is comprised of its level at time $t-1$, minus negative extents at time t, plus positive extents at earlier times, plus injections $\Pi_{r,t} \ge 0$. Constraint (1c) limits the total size of extents of

a task $i \in \mathcal{I}$ at some time $t \in \mathcal{T}^{day}$ to a given amount $B^{\max}_{i,t}$. Constraint (1d) is similar to the previous one and bounds the total size of the extents of transportation tasks. However, in this case, the total capacity depends on the number of transportation resource units allocated for task i at time t ($y_{i,t}$), with each of them adding $B^{\max}_{i,t}$ to the total capacity. While all tasks have capacity constraints limiting the extent size at some time $t \in \mathcal{T}^{day}$, only transportation tasks $i \in \mathcal{I}^T$ have a cost associated with its operations, C_i. Therefore, there is no need to model the occurrence of tasks $i \in \mathcal{I}^S \cup \mathcal{I}^{LM}$ as discrete variables. Their activity is controlled by the capacity bound $B^{\max}_{i,t}$ alone. Constraint (1e) enforces *resource limits*, and can be used to model storage capacity in the system. Constraint (1f) imposes a user-specified upper bound $\hat{\Phi}^{cost}$ on the total line-haul cost (C_i is the cost of adding a vehicle for transportation task i); we use this constraint to de-incentivize solutions that achieve high speed at the price of using many nearly empty vehicles. Throughout the remainder of this paper, we will refer to the MIP comprised of (1a)–(1g) simply as (1).

2.4. Cyclic schedule

In our specific application, the demand associated with OD pairs behaves in a cyclic way; i.e., we assume that demand exhibits daily repeating patterns and that the optimal schedules are repeated day-to-day. Like it was done in previous work by Castro et al. (2003, 2005), we adapt the general-purpose mathematical formulation of an RTN to account for the ensuing cyclical patterns. While task schedule and demand can be modeled in a single day horizon, it is important to keep track of commodity flows in a multiple-day horizon to be able to compute the time in days between order and delivery and, in turn, check if an package of commodity c reaches its destination within a given $target_c$ number of days.

Therefore, the proposed formulation has a multi-day time window that is consistent with the cyclic assumption. Specifically, for package orders of a commodity c, our model only registers the shipments at the origin facility o_c in the first day, i.e., $\Pi_{r,t} = 0\ \forall r \in \mathcal{R}^{supply}, t \in \mathcal{T}_r : t \geq T$. To account for processing capacity consumption downstream, our model combines package flows for the same time-in-a-day in different days. In physical terms, this means that a task carries packages ordered on multiple days of the extended time window, $n \in \{0, \ldots, N\}$. This is mathematically done by the use of the *modulo operation* in task capacity constraints (1c)-((1d)), which enforces that all extents m assigned to a task i on different days but at the same time of day consume capacity of such task i. Hence, $b_{m,t}$, $b_{m,t+n^{day}}$, $b_{m,t+2n^{day}}$, ... contribute to the same task load.

Raw Network, RTN and MIP Network Flow Model: an example. Consider the physical network in Fig. 1 where there are 2 FCs, 1 SC and 2 DSs. Assume that there are three ODs: OD1, OD2 and OD3 with (origin,destination) pairs given by (FC1,DS1), (FC1,DS2) and (FC2,DS1), respectively. The paths associated to these ODs all have the SC as the unique intermediate vertex. In the following figures, we associate a unique color with each of the ODs: OD1 is associated with color red, OD2 is blue, and OD3 is purple. We show elements of the figure pertaining to a specific OD in the respective color. For example, Fig. 1 displays the path of OD1 using dashed red arcs.

The RTN corresponding to the physical network Fig. 1 is shown in Fig. 2, where resources are represented by circles and tasks by rectangles. Resources are labeled by triples (x, y, z) where x is the OD identifier, y is the site name in the physical network that is associated to the resource, while the value of z is "IB" (inbound) or "OB" (outbound), depending on whether the resource is entering or leaving site y, respectively. Task labels are of two types: "X-Y MV" when the task moves resources from site X to site Y and "Z SORT" when it sorts the resource in facility Z. In this figure we assign identifiers r_i to each of the 15 resources. Similarly, identifiers I_j are assigned to the seven tasks. Each task has several associated extents that we label as $m_1, m_2, \ldots$. Extent labels are placed next to the label of the task they are associated with. We emphasize that extent labels are relative to their task, and that, for example, extent m_1 of task I_1 and extent m_1 of task I_2 represent distinct objects of the RTN model.

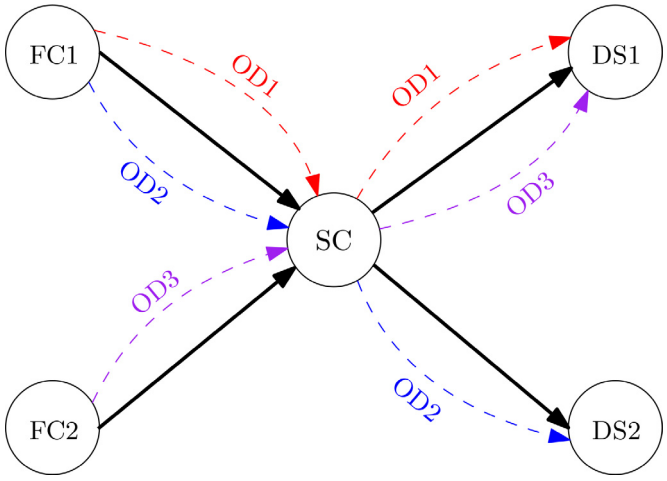


Fig. 1. Raw network with four arcs (filled lines) and three ODs (dashed lines).

Figure 3 sketches how the RTN in Fig. 2 is transformed into the corresponding flow network that forms the basis of the MIP model in Section 2. Notice that, the network in this figure is a simplification of the true network as it does not reflect the fact that we are modeling flow over time. As such, there are as many copies of a vertex as there are periods in the planning horizon. To be precise, a vertex x should be represented by x_t, where t is a valid time period. Similarly, an arc representing an extent of duration τ that transforms resource x into resource y corresponds to *timed* arcs between vertex copies x_t and $y_{t+\tau}$ for appropriate values of t. In the figure, we also write the name of the flow variables b over the corresponding arc. Note, however, that according to the previous observation, these names omit the subscript corresponding to the time period. Also, observe that the Π constants in the flow balance constraints (1b) are only indicated for resources associated to the origin of the OD pair as this is the only place where packages can be injected into the network. As a final remark, notice that a dashed (purple) rectangle is used in this figure to highlight the arcs (i.e., task extents) that belong to the task whose identification is written next to that rectangle.

3. Reducing the MIP model via condensation

The instances of tfMCF considered in this work have the property that the number of vertices on each commodity path is a small constant (that is independent of the size of the input). Furthermore, the RTN model in these instances defines a constant number of resources per vertex. Under these two assumptions, problem (1) has $O(\sum_{c \in \mathcal{C}} |\mathcal{T}_c|)$ variables and constraints. In practical instances of tfMCF the number $|\mathcal{C}|$ of commodities tends to be prohibitively large. Thus, solving MIP model (1) for such instances can be computationally infeasible. In this section we describe *condensation* techniques that allow us to write a more compact MIP.

Consider an origin vertex $v \in V^{origin}$ where $V^{origin} \subset V$, and let $\text{cube}^{\min}_v$ and $\text{cube}^{\max}_v$ be the minimum and maximum average cube of OD pairs with origin v, respectively. Let $\mathcal{TA}$ and $\mathcal{H}$ be the sets of distinct target values $target_c$ and horizon lengths $|\mathcal{T}_c|$, respectively. For a parameter $\epsilon > 0$ (called the *condensation factor*), target $\tilde{t} \in$

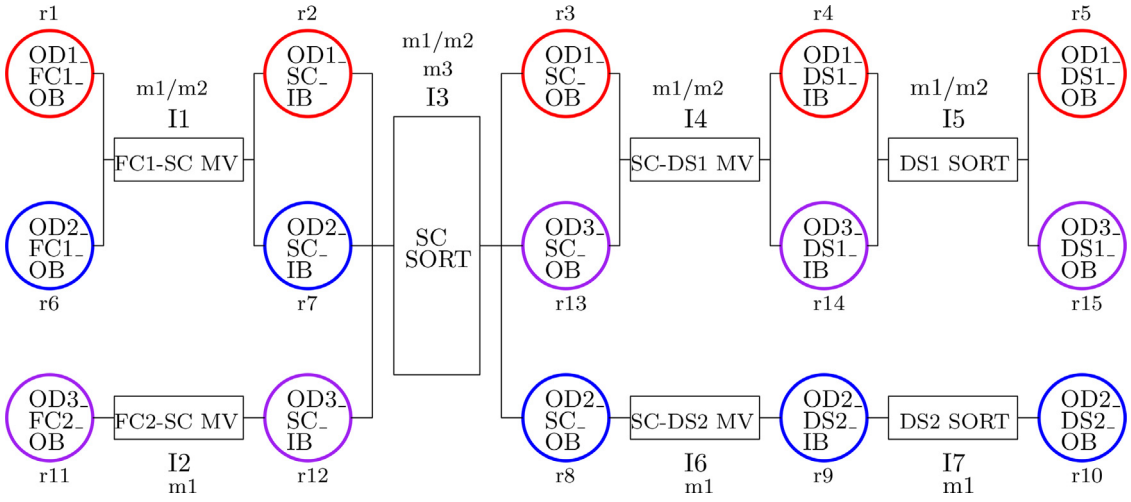


Fig. 2. RTN corresponding to physical network in Fig. 1.

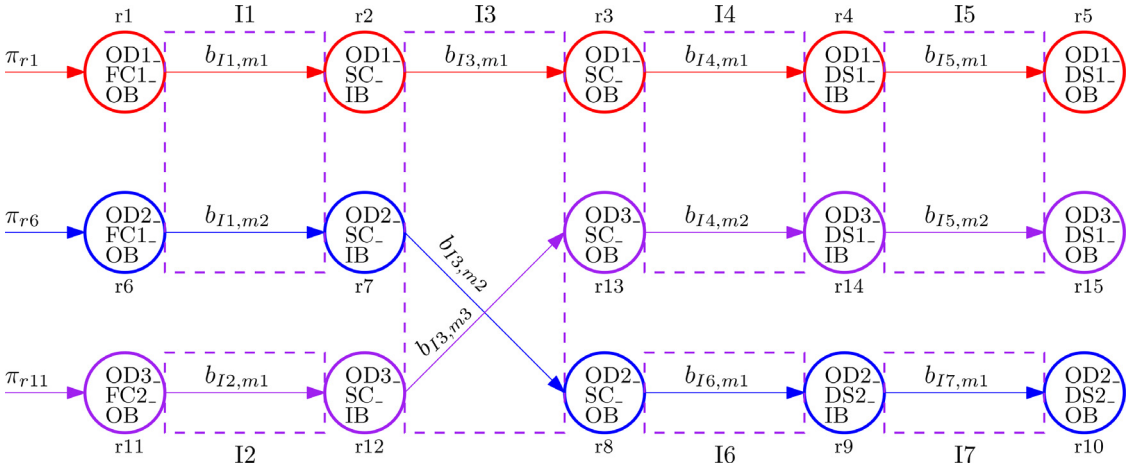


Fig. 3. MIP Network corresponding to the RTN in Fig. 2.

$\mathcal{TA}$, and horizon length $\tilde{h} \in \mathcal{H}$, we then let:

$$\mathcal{O}_{v,j}^{\tilde{t},\tilde{h}} = \{(o_c, d_c) : c \in \mathcal{C}, o_c = v,\ target_c = \tilde{t},\ |\mathcal{T}_c| = \tilde{h},\ \text{and} \quad \text{pkgCube}_c/\text{pkgCount}_c \in [\text{cube}_v^{\min}(1+\epsilon)^j, \text{cube}_v^{\min}(1+\epsilon)^{j+1})\} \tag{2}$$

be the *bucket* of ODs with origin v whose average cube is at least $\text{cube}_v^{\min}(1+\epsilon)^j$, and less than $(1+\epsilon)$ times that, whose target is $\tilde{t}$, and whose horizon length is $\tilde{h}$. Note that, for a given origin v, indices j may take on values in the set

$$\mathcal{J}_v := \{0, \ldots, \lceil \log_{1+\epsilon} \text{cube}_v^{\max}/\text{cube}_v^{\min} \rceil\}.$$

For the sake of simplicity, we describe here the creation of buckets for each origin vertex, average cube range, target and horizon values. Our implementation also considers transportation mode (i.e. air and ground) and containerization decision (i.e. containerized and fluid loaded) in its definition of buckets.

In the following, we let $\sigma_{v,j} \in [\text{cube}_v^{\min}(1+\epsilon)^j, \text{cube}_v^{\min}(1+\epsilon)^{j+1})$ be a user-chosen *average cube parameter* for each bucket $\mathcal{O}_{v,j}^{\tilde{t},\tilde{h}}$. We further let $\tilde{\sigma}_{v,j} \leq 1+\epsilon$ be the maximum ratio of $\sigma_{v,j}$ and the average cube of OD (o_c, d_c), over all $(o_c, d_c) \in \mathcal{O}_{v,j}^{\tilde{t},\tilde{h}}$.

The main idea behind condensation is now as follows: since the ODs in $\mathcal{O}_{v,j}^{\tilde{t},\tilde{h}}$ are *similar* – they have the same origin, and average cube within a factor of $(1+\epsilon)$ – we may treat them as *one* commodity. Merging the commodities in $\mathcal{O}_{v,j}^{\tilde{t},\tilde{h}}$ now allows us to merge the corresponding resource and extent sets.

We point out that the basic idea of condensation – the merging of commodities that share certain characteristics (like a common origin) – is folklore, and has been applied before. The authors were inspired by prior work in the field approximate fractional multi-commodity flow solvers, where the running time of earlier algorithms (Fleischer, 2000; Garg & Könemann, 2007) was improved in subsequent work of Karakostas (2008) by combining commodities that share an origin. Related ideas were also used by Jarrah, Johnson, & Neubert (2009) in their work on problems in the context of the less-than-truckload freight shipment industry. The authors make use of their specific application, where each shipment has a single path, and where the paths destined for the same vertex form a tree. In this setting the authors reformulate their problem by considering flows on trees instead of on paths. While in our setting, paths with the same destination do not necessarily form a tree, the ideas are nevertheless related.

In the following, we start by defining the elements of the condensed resource task network, before we continue defining the condensed MIP (1) of (1).

Condensing resources. Recall from the discussion in Sections 2.1 and 2.3 that the RTN underlying MIP (1) defines a set $\mathcal{R}_c$ of resources for each commodity $c \in \mathcal{C}$. This set contains resources for all vertices v on the path P_c in the underlying physical network, and for each resource type. For example, in the RTN depicted in Figure 2 corresponding to the tfMCF instance with network given in Fig. 1, the unique SC has 6 associated resources

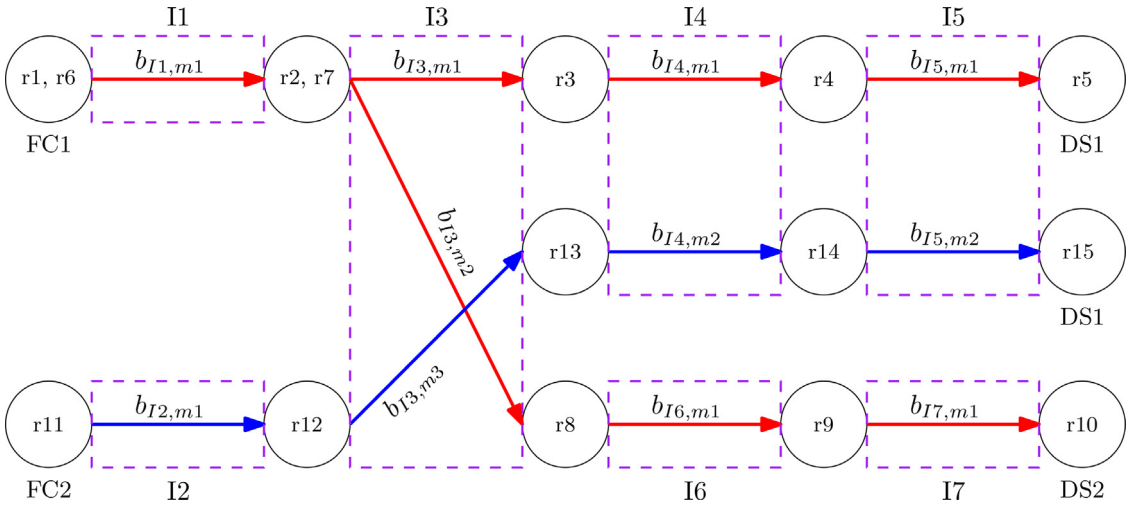


Fig. 4. Condensed MIP Network.

for the three ODs, and 2 resource types (IB and OB) given in the instance.

Let $P^{\tilde{t},\tilde{h}}_{v,j}$ be the set of vertices of paths P_c corresponding to ODs $(o_c, d_c) \in \mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$. In the condensed RTN, we now create a resource for *every bucket* $O^{\tilde{t},\tilde{h}}_{v,j}$, for every vertex in $P^{\tilde{t},\tilde{h}}_{v,j}$, and for every resource type. A particular resource r for OD $(o_c, d_c) \in \mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$, vertex $u \in P_c$, and resource type t is now mapped to the *condensed resource* $\tilde{r}$ corresponding to the bucket containing (o_c, d_c), vertex u, and type t. Notice that several original resources may be mapped to the same condensed resource, yielding the wanted *condensation*. For example, Fig. 4 shows the condensation of the RTN in Fig. 2 where resources $r2$ and $r7$ of the original RTN are mapped to the same condensed resource. Given the above, the set of resources of the condensed RTN is now given by $\tilde{\mathcal{R}} = \{\tilde{r} : r \in \mathcal{R}\}$.

We also point out that whenever resources r' and r are mapped to the same condensed resource $\tilde{r}$, then these resources must belong to commodities c' and c, respectively, that lie in the same bucket; we will refer to this bucket as *the bucket of* $\tilde{r}$. For this reason, we can now define $target_{\tilde{r}}$ and $N_{\tilde{r}}$ as the target value and horizon, respectively, of the bucket of $\tilde{r}$.

Because paths of ODs that lie in the same bucket overlap (at the very least in the origin), the above resource condensation yields an often significant reduction in the number of resources in the RTN. This is again exemplified in Figs. 2 and 4.

Condensing extents. Recall that each extent $m \in \mathcal{M}_c$ specifies a pair (r^-_m, r^+_m) of $\mathcal{R}_c$ resources that it depletes, and feeds, respectively. The fact that we combine multiple similar ODs into buckets now allows us to condense extents of commodities in $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$ whose resources map to the same condensed resources; i.e., define the extent set of bucket $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$ as

$$\tilde{\mathcal{M}}^{\tilde{t},\tilde{h}}_{v,j} = \{(\tilde{r}^-_m, \tilde{r}^+_m) : m \in \mathcal{M}_c, (o_c, d_c) \in \mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}\},$$

where we use $\tilde{r}^-_m$ and $\tilde{r}^+_m$ to denote the condensed resources associated with r^-_m and r^+_m, respectively. This is the case of extent $b_{I1,m1}$ in Fig. 4 which is the condensed extent obtained from extents $b_{I1,m1}$ and $b_{I1,m2}$ in Fig. 3. We let $\tilde{\mathcal{M}}$ be the extents of the condensed formulation; the set is comprised of the union of extent sets $\tilde{\mathcal{M}}^{\tilde{t},\tilde{h}}_{v,j}$ for all buckets $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$. In the following we say that extent $m \in \mathcal{M}_c$ for some OD (o_c, d_c) of bucket $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$ with associated resources (r^-_m, r^+_m) *maps* to condensed extent $(\tilde{r}^-_m, \tilde{r}^+_m)$.

Condensing the MIP – Balance constraints. Consider bucket $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$, and let $\mathcal{D}^{\tilde{t},\tilde{h}}_{v,j}$ be the set of destinations of OD pairs in $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$. For any $d \in \mathcal{D}^{\tilde{t},\tilde{h}}_{v,j}$ let $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j,d}$ be the set of ODs in $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$ with destination d. Notice that resource condensation discussed above will map the product resources of ODs in $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j,d}$ to the same condensed product resource. We will abuse notation in the following, and refer to this resource as d as well. The condensed formulation models the fact that packages of ODs in $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$ travel between v and $\mathcal{D}^{\tilde{t},\tilde{h}}_{v,j}$ (more precisely, it models flow between the single condensed supply resource at v, and the condensed product resources at $\mathcal{D}^{\tilde{t},\tilde{h}}_{v,j}$ in the time-expanded RTN graph). The main ingredient for describing the feasible region is a suitably modified system of balance and capacity constraints (1b) – (1d).

Recall that, for a condensed resource $\tilde{r}$, the set of resources r whose condensed resource is $\tilde{r}$ belongs to the same bucket. Hence, all such resources r share the target and horizon values of their bucket; we will use $target_{\tilde{r}}$ and $\mathcal{T}_{\tilde{r}}$ to refer to these. The condensed MIP has a balance constraint for each condensed resource $\tilde{r}$, and for each time $t \in \mathcal{T}_{\tilde{r}}$. The balance constraint for condensed resource $\tilde{r}$ and $t \in \mathcal{T}_{\tilde{r}}$ is obtained by summing the original balance constraints (1b) for pairs (r, t), where $r \in \tilde{\mathcal{R}}^{-1}(\tilde{r})$. Note that the summed balance constraints do not share any variables. In the resulting sum, we replace original extent variables by their respective condensed variables (dropping duplicates). Similarly, replace the sum of original induction variables by the one condensed variable $\Pi_{\tilde{r},t}$.

For any condensed extent $\tilde{m} \in \tilde{\mathcal{M}}$, we let $\tilde{\mathcal{M}}^{-1}(\tilde{m})$ be the set of extents in $\mathcal{M}$ whose condensed extent is $\tilde{m}$. For any such condensed extent $\tilde{m}$, and for any time $t \in \mathcal{T}_m$, we replace the set of variables $\{b_{m,t} : m \in \tilde{M}^{-1}(\tilde{m})\}$ by $b_{\tilde{m},t}$.

Condensing the MIP – Capacity constraints. As described, in the capacity constraints (1c) and (1d) we replace terms of extent variables $b_{m_1,t}, \ldots, b_{m_q,t}$ that have the same condensed extent $\tilde{m}$ by a single term for variable $b_{\tilde{m},t}$. The coefficient of this new variable is the average cube parameter $\sigma_{v,j}$ of the bucket associated with the ODs of $m_1, \ldots, m_q$. We also replace the $B^{\max}_{i,t}$ term on the right-hand side of the constraint by $\tilde{\sigma}\, B^{\max}_{i,t}$, where $\tilde{\sigma}$ is the maximum of $\sigma_{v,j}$ over all buckets $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$, if this is at least 1, and we let $\tilde{\sigma} = 1$, otherwise.

Condensing the MIP – Resource limits. In (1) we have a resource limit constraint for each condensed resource $\tilde{r} \in \tilde{\mathcal{R}}$, and for each

time $t \in \mathcal{T}_{\tilde{r}}$. This constraint bounds the value of the condensed resource variable by the sum of the bounds of those resource variables that map to $\tilde{r}$; similar to the extent case, we let this set be $\tilde{\mathcal{R}}^{-1}(\tilde{r})$. We then have

$$R_{\tilde{r},t} \leq \sum_{r \in \tilde{\mathcal{R}}^{-1}(\tilde{r})} R^{max}_{r,t}. \tag{3}$$

Condensing the MIP – Induction. The original model has an induction parameter $\Pi_{r,t}$ for each original resource r, and time $t \in \mathcal{T}_r$. We now define $\tilde{\Pi}_{\tilde{r},t}$ for condensed resource $\tilde{r}$ and time $t \in \mathcal{T}_{\tilde{r}}$ by summing $\Pi_{r,t}$ over all resources $r \in \tilde{\mathcal{R}}^{-1}(\tilde{r})$ whose condensed resource is $\tilde{r}$.

Condensation example. To illustrate the condensation procedure, we apply it to the example in Fig. 1. There, if a condensation is done that transforms *OD*1 and *OD*2 into a single OD with origin in *FC*1 and destinations in *DS*1 and *DS*2, the flow network corresponding to the condensed MIP model, following the conventions above, is the one seen in Fig. 4. The paths corresponding to the condensed OD are identified by the red arcs. In this simple example the number of resources (task extents) was reduced from 15 (12) to 13 (11).

3.1. Strengthening the condensed MIP

On the one hand, combining ODs into buckets reduces the number of extent variables as well as the number of constraints. On the other hand, it results in several subtle issues that all stem from the fact that the condensed LP can not distinguish between flow from different ODs in the same bucket. For example, packages associated with ODs in a bucket are now allowed to travel along paths belonging to different ODs in the same bucket. In the application discussed in this paper this is permissible.

There are problems arising from the inability to distinguish packages that are less harmless, and for which we need to add model strengthenings.

Bounding the flow into destinations. Let $\mathcal{D}^{\tilde{t},\tilde{h}}_{v,j}$ be the set of product resources of ODs in bucket $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$, and let $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j,d}$ be the subset of ODs in $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$ that have destination d. We need to ensure that the total number of packages from bucket $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$ reaching product resources at $d \in \mathcal{D}^{\tilde{t},\tilde{h}}_{v,j}$ does not exceed the total number of packages of ODs in $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$ that have destination d (henceforth referred to by $\mathsf{pkgCount}_{v,j,d}$). This is accomplished by adding the following constraint to (1):

$$\sum_{\tilde{m} \in \tilde{\mathcal{M}}^{\tilde{t},\tilde{h}}_{v,j,d}} \sum_{t \in \mathcal{T}_d} b_{\tilde{m},t} \leq \mathsf{pkgCount}_{v,j,d} \qquad \forall v \in V^{origin},\ j \in \mathcal{J}_v,\ \ d \in \mathcal{D}^{\tilde{t},\tilde{h}}_{v,j}, \tag{4}$$

where we let $\tilde{\mathcal{M}}^{\tilde{t},\tilde{h}}_{v,j,d}$ be the set of condensed extents of such ODs feeding the product resource corresponding to destination d.

Early induction. Another subtle side effect has to do with how flow for ODs is introduced into the network. In our application, we are given an *induction curve* that specifies what fraction $\Pi_{r,t}$ of $\mathsf{pkgCount}_c$ for each OD $c \in \mathcal{C}$ is introduced at supply resource r of the OD at each time t of the day. Condensation, as described above, allows the model to ship more flow of OD c to d_c as would normally be possible (see also Example 3.1). To counteract this phenomenon, here called *early induction*, we add new constraints that limit the cumulative package count associated with the ODs of a common bucket arriving at a destination vertex by some time t. To make this precise, we first let $\mathsf{minArrival}_{v,j,d}$ be the minimum transit time of the path of any OD in $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j,d}$. We then let $\mathsf{arrival}^{UB}_{v,j,d,t}$ be the cumulative number of packages ordered by ODs in $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j,d}$ between time 0 and time $(t - \mathsf{minArrival}_{v,j,d})$. The new constraint now is as follows:

$$\sum_{\tilde{m} \in \tilde{\mathcal{M}}^{\tilde{t},\tilde{h}}_{v,j,d}, t' \leq t} b_{\tilde{m},t'} \leq \mathsf{arrival}^{UB}_{v,j,d,t} \quad \forall v \in V^{origin},\ j \in \mathcal{J}_v,\ \ d \in \mathcal{D}^{\tilde{t},\tilde{h}}_{v,j},\ \ t \in \mathcal{T}_d \tag{5}$$

Limit total flow on arcs. Consider the path P_c associated with OD $c \in \mathcal{C}$, and define $\mathsf{pkgCount}^e$ to be the total number of packages traveling on OD paths containing edge $e \in E$; i.e.,

$$\mathsf{pkgCount}^e = \sum_{c \in \mathcal{C} : e \in P_c} \mathsf{pkgCount}_c.$$

Let $\tilde{\mathcal{M}}_e$ be the set of all (condensed) extents associated with transportation tasks on edge e. We then require the total value of extents in $\mathcal{M}_e$ (the *total flow* on e) to be bounded by $\mathsf{pkgCount}^e$; i.e.,

$$\sum_{\tilde{m} \in \tilde{\mathcal{M}}_e, t \in \mathcal{T}_e} b_{\tilde{m},t} \leq \mathsf{pkgCount}^e \quad \forall e \in E, \tag{6}$$

where $\mathcal{T}_e$ is the union of horizons of all ODs whose path uses e. To summarize our discussion, we provide a full description of $(\tilde{1})$.

$$\max \sum_{\tilde{r} \in \tilde{\mathcal{R}}^{prod}} \left(R_{\tilde{r},target_{\tilde{r}}T-1} + R_{\tilde{r},N_rT-1} \right) \tag{$\tilde{1}a$}$$

$$\text{s.t.} R_{\tilde{r},t} = R_{\tilde{r},t-1} + \sum_{m \in \tilde{\mathcal{M}}^+_{\tilde{r}}} b_{m,t-\tau_m} - \sum_{m \in \tilde{\mathcal{M}}^-_{\tilde{r}}} b_{m,t} + \Pi_{\tilde{r},t} \quad \forall \tilde{r} \in \tilde{\mathcal{R}}, t \in \mathcal{T}_{\tilde{r}} \tag{$\tilde{1}b$}$$

$$\sum_{\tilde{m} \in \tilde{\mathcal{M}}_i} \sum_{t' \in \mathcal{T}_{\tilde{m}} : t' \equiv t \bmod T} \sigma_{\tilde{m}} b_{\tilde{m},t'} \leq \tilde{\sigma} B^{\max}_{i,t} \quad \forall i \in \mathcal{I}^S \cup \mathcal{I}^{LM}, t \in \mathcal{T}^{day} \tag{$\tilde{1}c$}$$

$$\sum_{\tilde{m} \in \tilde{\mathcal{M}}_i} \sum_{t' \in \mathcal{T}_{\tilde{m}} : t' \equiv t \bmod T} \sigma_{\tilde{m}} b_{\tilde{m},t'} \leq \tilde{\sigma} B^{\max}_{i,t} y_{i,t} \quad \forall i \in \mathcal{I}^T, t \in \mathcal{T}^{day} \tag{$\tilde{1}d$}$$

$$0 \leq R_{\tilde{r},t} \leq \sum_{r \in \tilde{\mathcal{R}}^{-1}(\tilde{r})} R^{\max}_{r,t} \tag{$\tilde{1}e$}$$

$(1f), (4) - (6)$

In the above MIP, we let $\tilde{\mathcal{M}}_i$ be the set of condensed extents for condensed task i, and we let $\sigma_{\tilde{m}}$ be the average cube parameter of the bucket corresponding to extent $\tilde{m} \in \tilde{\mathcal{M}}_i$. Recall from the discussion in the resource condensation section that resources $\tilde{r} \in \tilde{R}$ have well-defined target value and horizon length, corresponding to those of resources that lie in the same bucket as $\tilde{r}$.

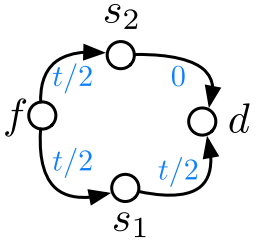
The following small example shows that, despite adding constraints (4) – (6) to (1), condensed and un-condensed flows may behave quite differently, and early induction is not entirely prevented through the added constraints.

Example 3.1. The example given in the figure below has two ODs, each having total demand volume equal to one truck load (henceforth denoted as 1 tl), the same average cube, and target time $t > 0$. Arcs in the figure are annotated by their transit times. For both ODs, tl/2 packages are induced at time 0, and the same amount at time $t/2$. For $i = 1, 2$, OD_i has path $\langle f, s_i, d \rangle$. For this simple example, we assume that delay is only incurred while traversing arcs (as opposed to also at nodes for sortation operations, etc.).

Using the above setup, notice that at most tl/2 units of OD_1 can make it to destination d by time t in any un-condensed solution. Note that the two ODs are in a common bucket in the condensed model. For the ODs of this bucket tl units of demand are available at time 0. Using early induction, we designate this demand as belonging to OD_1, and send it to d along path $\langle f, s_1, d \rangle$. Another tl

units of demand becomes available at f at time $t/2$, and is sent to d along path $\langle f, s_2, d\rangle$. This way, the entire 2tl units of demand for both of ODs arrives at d in time.

Note that constraint (4) is clearly satisfied in this solution. To see that (5) is satisfied, notice that both ODs in this example are in the same bucket; let this be $\mathcal{O}^{\tilde{t},\tilde{h}}_{f,j}$. Further, note that $\text{minArrival}_{v,j,d}$ equals $t/2$, and hence $\text{arrival}^{UB}_{f,j,d,t}$ equals 2 tl. Thus (5) is satisfied. Finally, note that each arc in the example given belongs to exactly one OD path, and that our solution sends exactly the demand of the corresponding OD along this path, and hence satisfies (6) as well.



The above example shows that we cannot always assume that a condensed solution may be converted *losslessly* into an uncondensed one (e.g., via standard *flow decomposition* techniques; see Ahuja, Magnanti, & Orlin, 1988). This and other theoretical aspects involving the two formulations are discussed in the next subsection. For now, we just highlight the fact that the condensed model proved to be of sufficient quality for the instances present in our application.

3.2. Properties of the condensed formulation

Our first theorem relates the optimum values of (1) and (1).

Theorem 3.2. *The optimum value of* $(\tilde{1})$ *is at least the optimum value of (1).*

Proof. Let (b, y, R) be a feasible solution for (1). We now define natural projections of b' and R' of b and R, respectively, such that (b', y, R') is feasible for $(\tilde{1})$, and has value equal to that of (b, y, R). Let $\mathcal{M}$ be the set of all extents of the original formulation, and recall the $\tilde{m}$ is the condensed extent corresponding to $m \in \mathcal{M}$, and that we defined $\tilde{\mathcal{M}}$ as the set of condensed extents. Further, recall that we defined $\tilde{\mathcal{M}}^{-1}(\tilde{m})$ as the set of extents in $\mathcal{M}$ whose corresponding condensed extent is $\tilde{m}$. For $\tilde{m} \in \tilde{M}$, and time $t \in \mathcal{T}_{\tilde{m}}$ we now define

$$b'_{\tilde{m},t} = \sum_{m \in \tilde{\mathcal{M}}^{-1}(\tilde{m})} b_{m,t}.$$

We also let

$$R'_{\tilde{r},t} = \sum_{r \in \tilde{\mathcal{R}}^{-1}(\tilde{r})} R_{r,t},$$

for all condensed resources $\tilde{r}$ of the condensed formulation, and for all times t. Recall that, by our choice of condensation, $\tilde{r} = \tilde{r'}$ for product resources r and r' only if the corresponding ODs have the same target values and horizon lengths. This implies that the defined map from (b, y, R) to (b', y, R') is objective value preserving. It remains to show that (b', y, R') is feasible for $(\tilde{1})$.

Let $\tilde{r}$ be a condensed resource, and t some time. Recall that constraint $(\tilde{1}b)$ for $\tilde{r}$ and t is the sum of the original balance constraints of resources r and t where $r \in \tilde{\mathcal{R}}^{-1}(\tilde{r})$ and, therefore, is satisfied by the vector (b', y, R').

Constraints $(\tilde{1}c)$ and $(\tilde{1}d)$ are the same as their uncondensed counterparts, *except* the possibly altered σ coefficients. However, by definition, whenever σ is replaced by σ' then, by definition, $\sigma'/\sigma \leq \tilde{\sigma}$. Thus, the scaling of $B^{\max}$ parameters on the right side of the capacity constraints ensures that the capacity constraints hold.

The adjusted resource bound constraints $(\tilde{1}e)$ are clearly satisfied by the definition of $\tilde{R}$ above. Finally, since the variables y are the same in both models, constraint (1f) is fulfilled. Constraints (4) - (6) are satisfied by b' as the latter vector is obtained from a feasible solution b for (1). □

Note that $\tilde{\sigma} = 1$ in (1) if we choose all average cube parameters equal to the lower bounds of their respective average cube intervals. Theorem 3.2 therefore shows that the introduced scaling of right-hand sides of capacity constraints (1c) and (1d) is not needed to obtain a relaxation in this case. We note however that, in our implementation, we choose σ parameters equal to the *upper* bounds of their respective average cube intervals *without* scaling the $B^{\max}$ parameters. In theory, this could of course result in a condensed formulation whose objective value is significantly smaller than that of (1). However, in our practical experience, this tends not to happen.

Choosing σ parameters equal to the average cube interval upper bounds, allows us to translate condensed solutions back to their uncondensed counterparts (under certain conditions). In fact, under the previous condition $\tilde{\sigma}$ is equal to one and the following result holds.

Theorem 3.3. *Let* (b', y, R') *be a solution for* $(\tilde{1})$*, and assume that all average cube parameters* σ *are chosen at their upper bounds. Then we can find* (b, y, R) *that satisfies the constraints of (1) under the following provisos:*

(i) *For any bucket* $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$*, and any OD* $(o_d, d_c) \in \mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$ *we allow packages of the OD to be routed along any* o_c, d_c*-path in the union of paths of ODs in this bucket.*
(ii) *We are allowed to modify induction parameters* Π *so that the sum of variables associated with resources of any given bucket remains unchanged.*
(iii) *We are allowed to relax resource capacity (storage capacity) constraints.*

The objective value of (b, y, R) *in (1) is at least that of* (b', y, R') *in (1).*

Proof. The linear systems describing feasible solutions of (1) as well as $(\tilde{1})$ are easily seen to have *network flow* structure. We will shortly make this explicit, and proceed to show that this structure enables us to use well known *flow-decomposition* techniques to *strip* the given bucket-granularity solution into one that has OD-granularity.

In our proof we convert (b', y, R') into (b, y, R) iteratively. The y variables are not changed in this conversion. Variables b', and R' are associated with specific buckets. We make use of this, and translate variables associated with buckets $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$, for all v and j, in isolation, one by one.

For now, fix a bucket $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$, and let us construct a digraph $H^{\tilde{t},\tilde{h}}_{v,j}$. The graph has one vertex for each condensed resource associated with the bucket; i.e.,

$$V(H^{\tilde{t},\tilde{h}}_{v,j}) = \{(\tilde{r}, t) : r \in \mathcal{R}_c, (o_c, d_c) \in \mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}, t \in \mathcal{T}_r\}.$$

Recall that every condensed extent $\tilde{m}$ has associated condensed resources $\tilde{r}^-_{\tilde{m}}$, and $\tilde{r}^+_{\tilde{m}}$ as well as duration $\tau_{\tilde{m}}$. We add arc $\big((\tilde{r}^-_{\tilde{m}}, t), (\tilde{r}^+_{\tilde{m}}, t + \tau_{\tilde{m}})\big)$ whenever the corresponding variable $b'_{\tilde{m},t}$ is positive. We also add *hold-over* arcs $((\tilde{r}, t-1), (\tilde{r}, t))$ for resource $\tilde{r}$, and $t \in \mathcal{T}_{\tilde{r}}$ whenever $R'_{\tilde{r},t-1}$ is positive; i.e., whenever packages reside at resource $\tilde{r}$ in the time interval $[t-1, t)$. Thus,

$$E(H^{\tilde{t},\tilde{h}}_{v,j}) = \{\big((\tilde{r}^-_{\tilde{m}}, t), (\tilde{r}^+_{\tilde{m}}, t + \tau_{\tilde{m}})\big) : \tilde{m} \in \tilde{\mathcal{M}}^{\tilde{t},\tilde{h}}_{v,j}, b'_{\tilde{m},t} > 0\} \cup$$

$$\{((\tilde{r}, t-1), (\tilde{r}, t)) : \tilde{r} \in \tilde{\mathcal{R}}_i \text{ for } i \in \mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}, t \in \mathcal{T}_{\tilde{r}} \text{ and } R'_{\tilde{r},t-1} > 0\}.$$

Extent variables b' yield natural associated arc flows by letting

$$b'_{((\tilde{r}^-_{\tilde{m}},t),(\tilde{r}^+_{\tilde{m}},t+\tau_{\tilde{m}}))} = b'_{\tilde{m},t},$$

for all $\tilde{m} \in \tilde{\mathcal{M}}^{\tilde{t},\tilde{h}}_{v,j}$ and for all $t \in \mathcal{T}_{\tilde{m}}$. Similarly, we define flows on hold-over arcs, by defining $b'_{((\tilde{r},t-1),(\tilde{r},t))} = R'_{\tilde{r},t-1}$, for all resources $\tilde{r}$, and for all $t \in \mathcal{T}_{\tilde{r}}$.

In the following, we say that a vertex $(\tilde{r}, t)$ is *terminal* if $(\tilde{r}, t+1) \notin V(H)$; i.e., if t is maximal in $\mathcal{T}_{\tilde{r}}$. Call a vertex *non-terminal* otherwise. In our setting, terminal vertices correspond to resources at the end of their respective horizons, and there are therefore no out-going arcs at these vertices. With the above, we can now define *net in-flow* parameters $\bar{\pi}_{\tilde{r},t}$ for each vertex $(\tilde{r}, t) \in V(H)$ as follows:

- If $(\tilde{r}, t)$ is non-terminal, we let $\bar{\pi}_{\tilde{r},t} = -\Pi_{\tilde{r},t}$; i.e., the total flow entering $(\tilde{r}, t)$ minus the total flow leaving (including hold-over) should equal the negative injection volume at the node.
- If $(\tilde{r}, t)$ is terminal, we let $\bar{\pi}_{\tilde{r},t}$ be equal to the total inflow on arcs incident to $(\tilde{r}, t)$.

Note that, by definition, $\bar{\pi}_{\tilde{r},t}$ is positive for all terminal nodes in $V(H)$, and that it is non-positive anywhere else. Let $\delta^{in}(\tilde{r}, t)$ and $\delta^{out}(\tilde{r}, t)$ be the set of arcs in $E(H)$ whose head and tail respectively, is $(\tilde{r}, t)$, for all $(\tilde{r}, t) \in V(H)$. With the definitions above, and from the fact that variables b' satisfy constraint $(\tilde{1}b)$, it now follows that

$$\sum_{e \in \delta^{in}(\tilde{r},t)} b'_e - \sum_{e \in \delta^{+}(\tilde{r},t)} b'_e = \bar{\pi}_{\tilde{r},t},$$

for all $(\tilde{r}, t) \in V(H)$. In other words, b' is a feasible *flow* for the flow instance given by H and $\bar{\pi}$. For a thorough introduction to network flow theory, we refer the reader to the excellent textbooks (Ahuja et al., 1988; Williamson, 2019).

Observe now that any arc $((\tilde{r}, t), (\tilde{r}', t')) \in E(H)$ satisfies $t' > t$, and hence H is acyclic. Standard flow-decomposition arguments together with the acyclicity of H now imply that there are directed paths $P_1, \ldots, P_q$ and positive values b'_{P_i} for all i such that

$$b'_e = \sum_{i : e \in P_i} b'_{P_i},$$

for all $e \in E(H)$. In other words, we can split the arc flow b' into paths. Each of these paths begins in a non-terminal, and ends in a terminal vertex.

Note that, by definition, $\Pi_{\tilde{r},t} = \sum_{r \in \tilde{\mathcal{R}}^{-1}(\tilde{r})} \Pi_{r,t}$, for any $(\tilde{r}, t) \in V(H)$, and that all resources $r \in \tilde{\mathcal{R}}^{-1}(\tilde{r})$ belong to to OD pairs in bucket $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$. Once again by standard flow decomposition arguments, it can be seen that we may assume that each of the directed paths P_j above belongs to a unique OD $(o_c, d_c) \in \mathcal{O}^{\tilde{t},\tilde{h}}_{v,r}$, and the total number of packages on these paths is no larger than $\mathsf{pkgCount}_c$. In the following, we will let $\mathcal{P}^c$ be the set of directed paths in $\{P_1, \ldots, P_q\}$ that belong to OD $(s_c, t_c) \in \mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$. Flow decomposition implies that we can pick these sets such that each path $P \in \mathcal{P}^c$ ends in a node $(\tilde{p}_c, t') \in V(H)$, where $\tilde{p}_c$ is a condensed product resource corresponding to OD (s_c, t_c), and some time t'. Flow decomposition also implies that the set of paths with start node $(\tilde{r}, t)$ for any $\tilde{r} \in \tilde{\mathcal{R}}^{\tilde{t},\tilde{h}}_{v,j}$, and $\tilde{t} \in \mathcal{T}_{\tilde{r}}$ has total flow equal to $\sum_{r \in \tilde{\mathcal{R}}^{-1}(\tilde{r})} \Pi_{r,t}$.

Note that we cannot argue that the paths in $\mathcal{P}^c$ starting at $(\tilde{r}, t)$ have total flow value at most $\Pi^c_{\tilde{r},t}$ (hence requiring proviso (ii)). Note also that paths in set $\mathcal{P}^c$ may in general be different from the original path shipment path P_c of an OD (o_c, d_c) (requiring proviso (i)).

It remains to define values of variables in the un-condensed solution corresponding to ODs in $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$. Consider a variable $b_{m,t}$ and let $e = (\tilde{r}^-_m, t), (\tilde{r}^+_m, t+\tau_m)$ be the corresponding arc in $E(H)$. Suppose that the OD corresponding to extent m is $(o_c, d_c) \in \mathcal{O}^{\tilde{n},\tilde{h}}_{v,j}$. We then let

$$b_{m,t} = \sum_{P \in \mathcal{P}^c, e \in P} b'_P$$

i.e., we sum the total flow value of paths of OD (o_c, d_c) that contain e (note, that, by definition, all ODs in a bucket have the same horizon, and hence variables exist).

Similarly, consider variable $R_{r,t}$ for some resource $r \in \mathcal{R}_c$, $(o_c, d_c) \in \mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$, and $t \in \mathcal{T}_r$. Let $e = ((r, t), (r, t+1))$ be the hold-over arc corresponding to this resource at time t, and we then let

$$R_{r,t} = \sum_{P \in \mathcal{P}^c, e \in P} b'_P$$

the total value of flow assigned to paths in $\mathcal{P}^i$ that contain e. Note that the above definition, and flow decomposition may yield values of R that violate (1e) (requiring relaxation (iii)). Note that constraint (1e) is relaxed in all our experiments, i.e., we do not impose storage capacity constraint.

Suppose that we apply the above procedure to all buckets $\mathcal{O}^{\tilde{t},\tilde{h}}_{v,j}$, and let the resulting solution be (b, y, R). Directly from the definitions of H above, and from the way we defined the flow decomposition, it follows that constraints (1b) are satisfied. Constraints (1c) and (1d) are satisfied as well; this follows (a) as the corresponding task constraints in $(\tilde{1})$ were satisfied by b', (b) b is essentially obtained by splitting b', and (c) our choice of average cube parameters σ equal to the upper bounds of the average cube interval of the respective bucket, and the definition of $\tilde{\sigma}$ imply that condensed flow over-estimates contribution to task capacity.

By our definition of buckets target and horizon values of product resources coincide for ODs in each bucket. Hence, flow decomposition can be seen to be objective function value preserving. □

4. Heuristic solution via Lagrangian decomposition

It is computationally challenging to solve (1) and $(\tilde{1})$ for large instances (e.g.; those arising from large real-world transportation problems) as a single, full-space optimization problem due to the complexity of timing decisions at fine (e.g., hourly) granularity. To improve its solution performance, and taking advantage of special structures in the formulation of the optimization problem, we propose a solution algorithm based on geographical decomposition combined with the so called *split Lagrangian* method (Guignard & Kim, 1987).

4.1. Balanced partitioning

Recall from Section 1 that we defined $G = (V, E)$ to be the graph associated with the input network. Furthermore, recall that each OD pair $(o_c, d_c) \in \mathcal{C}$ in the input has an associated o_c, d_c-path P_c. Finally, let $E_c \subseteq E(G)$ be the set of edges in o_c, d_c-path P_c.

The goal is to find a *balanced partition* $\pi = (V_1, \ldots, V_q)$ of the vertex set $V(G)$: (i) $V = V_1 \cup \ldots \cup V_q$; (ii) $V_v \cap V_u = \emptyset$, for all $v \neq u$; and (iii) $|V_v| \leq (1+\alpha)|V|/q$, for all v, where α is a given *imbalance* parameter. In the context of the tfMCF, the subgraph induced by a subset V_i of the partition is called a *region*.

Now, let $\delta(\pi)$ be the set of edges in G whose ends lie in different parts of π. We want to find a balanced partition that allows us to break the overall optimization problem into smaller problems that are as *independent* as possible. In light of the split-Lagrangian approach taken here, achieving approximate independence translates into minimizing the number of path edges in $\delta(\pi)$. More formally, we define the *weight* w_e of an edge $e \in E(G)$ as the number

of paths P_c for OD pair $(o_c, d_c) \in \mathcal{C}$ with $e \in E_c$. The goal is now to find a balanced partition π that minimizes $\sum_{e\in\delta(\pi)} w_e$.

The balanced separator problem is well-known to be NP-hard as well as hard to approximate within a constant factor (Bui & Jones, 1992). Because of its central place in divide-and-conquer type algorithmic strategies, graph partitioning algorithms are widely studied, and the body of previous work is vast. Here, we use the well-known graph partitioning package Metis (Karypis et al., 1999) which has been known to efficiently produce partitions of reliable performance. Metis employs the famed *multilevel graph partitioning* approach in which the input graph is recursively contracted to achieve smaller graphs. These smaller graphs are later uncontracted and refined.

4.2. Split Lagrangian

Lagrangian decomposition is a well-known technique to solve optimization problems. Given a mathematical formulation of the problem to be solved, in this technique the complicating constraints are *dualized* such that the remaining formulation is separable into multiple subproblems that are easier to optimize. The dualization of a constraint consists in adding a term to the objective function that penalizes solutions that violate the constraint. The penalty factor associated to this term is the *Lagrange multiplier* corresponding to the constraint. The optimization problem obtained by dualizing the complicating constraints is known to be a relaxation for the original problem. In the case of tfMCF, this means the optimum of the latter problem yields an *upper bound* for (1). To tighten this bound, the Lagrange multipliers need to be optimized. This task can be accomplished by solving the so called *Lagrangian dual* to produce the tightest bound which, for convex problems (not the case for MIPs), coincides with the optimum of the original problem. For a thorough presentation of Lagrangian Theory, the reader may refer to Nemhauser & Wolsey (1988).

We follow the *split Lagrangian* methodology proposed by Guignard & Kim (1987). In this approach copies of a carefully chosen subset of variables are created to generate separable subproblems that are easy to solve. The constraints that impose equalities between such variables and their copies are dualized in the usual Lagrangian way. Guignard and Kim showed that the bound obtained by split Lagrangian is never bigger (i.e., worse) and is often smaller (i.e., better) than that of the classical Lagrangian relaxation approach because every constraint in the original problem appears in one of the subproblems. Therefore, the optimization of multipliers through the Lagrangian dual problem can be interpreted as optimizing the primal objective function on the intersection of the convex hulls of the constraint sets of the subproblems (Guignard & Kim, 1987 Corollary 3.4). For a graphical interpretation of split Lagrangian as well as some practical examples, we refer the reader to this tutorial by Grossmann. For details on convergence, please refer to the original paper by Guignard & Kim (1987).

Here, we choose to demonstrate the application of Guignard and Kim's split Lagrangian approach to formulation (1) rather than its condensed version in (1). We do this for notational ease but observe that the work of Guignard & Kim (1987) can also be applied to the condensed MIP. Modifying (1), we create duplicates $b^{out}_{m,t}$ of extent size variables $b_{m,t}$ for $m \in \mathcal{M}_i$ where the corresponding transportation tasks i has its origin in a region different from that of its destination. We add constraints that force equality of $b^{out}_{m,t}$ and the original variable $b_{m,t}$ to our model. We obtain the Lagrangian subproblem of (1) by dualizing the latter equality constraints. The resulting Lagrangian subproblem has *block* structure, and decomposes neatly into independent regional problems.

Lagrangian subproblem. We state the complete formulation of the Lagrangian subproblem corresponding to (1) for the region induced by V_p, and $p \in \{1, \dots, q\}$. In the following, let λ be a given vector of Lagrangian multipliers.

$$\max \sum_{r\in\mathcal{R}^{prod}_p} \left(R_{r,n_r^{target}T-1} + R_{r,n_r^{day}T-1}\right) - \sum_{i\in\mathcal{I}^{out}} \sum_{m\in\mathcal{M}_i} \sum_{t\in\mathcal{T}_m} \lambda_{m,t} b^{out}_{m,t} + \sum_{i\in\mathcal{I}^{in,dest}} \sum_{m\in\mathcal{M}_i} \sum_{t\in\mathcal{T}_m} \lambda_{m,t} b_{m,t} \tag{7a}$$

$$\text{s.t. } R_{r,t} = R_{r,t-1} + \left(\sum_{m\in\mathcal{M}^{+,in}_{r,p}} b_{m,t-\tau_m} + \sum_{m\in\mathcal{M}^{+,out}_{r,p}} b^{out}_{m,t-\tau_m} \right) - \sum_{m\in\mathcal{M}^{-,in}_{r,p}} b_{m,t} + \Pi_{r,t} \quad \forall r \in \mathcal{R}_p,\ t \in \mathcal{T}_r \tag{7b}$$

$$\sum_{m\in\mathcal{M}_i} \sum_{t'\in\mathcal{T}_m : t'\equiv t \bmod t^{day}} \sigma_{i,m} b_{m,t'} \le B^{\max}_{i,t} \quad \forall i \in \mathcal{I}^S_p \cup \mathcal{I}^{LM}_p, t \in \mathcal{T}^{day} \tag{7c}$$

$$\sum_{m\in\mathcal{M}_i} \sum_{t'\in\mathcal{T}_m : t'\equiv t \bmod t^{day}} \sigma_{i,m} b_{m,t'} \le B^{\max}_{i,t} y_{i,t} \quad \forall i \in \mathcal{I}^T_p, t \in \mathcal{T}^{day} \tag{7d}$$

$$R_{r,t} \le R^{\max}_{r,t} \quad \forall r \in \mathcal{R}_p,\ t \in \mathcal{T}_r \tag{7e}$$

$$\sum_{i\in\mathcal{I}^T} \sum_{t\in\mathcal{T}^{day}} C_i y_{i,t} \le \hat{\Phi}^{cost}_p \tag{7f}$$

$$y_{i,t} \in \mathbb{Z}_+ \quad \forall i \in \mathcal{I}^T, t \in \mathcal{T}^{day} \tag{7g}$$

We will refer to the optimum of the above Lagrangian subproblem as Φ^{LR}_p. Above, $\lambda_{m,t}$ is the Lagrange multiplier of the equality constraint $b^{out}_{m,t} = b_{m,t}$. The sets $\mathcal{R}_p$ (and $\mathcal{R}^{prod}_p$) contain (product) resources corresponding to vertices in V_p. The set of tasks that start in a vertex in region p is represented by $\mathcal{I}^{in}_p$. The sets $\mathcal{M}^{+,in}_{r,p}$ and $\mathcal{M}^{-,in}_{r,p}$ are formed by the extents of tasks in $\mathcal{I}^{in}_p$ that feed and deplete resource r, respectively. Similarly, $\mathcal{I}^{out}_p$ is the set of tasks that start in a vertex that is not in p, but end in a vertex of p, and $\mathcal{M}^{+,out}_{r,p}$ is the set of extents of tasks in $\mathcal{I}^{out}_p$ that feed resource r. Also, $\mathcal{I}^{in,dest}_p \subset \mathcal{I}^{in}_p$ is the subset of tasks that start in a vertex that belongs to region p, but have as destination a vertex outside the region. Thus, one such task is part of the $\mathcal{I}^{out}_{p'}$ for some $p' \neq p$. Finally, we have $\mathcal{I}^{in,dest} = \bigcup_{p\in[q]} \mathcal{I}^{in,dest}_p$ and $\mathcal{I}^{out} = \bigcup_{p\in[q]} \mathcal{I}^{out}_p$.

The regional Lagrangian subproblems introduced above can be solved in parallel, and the sum of their objective values gives a relaxed upper bound (dual) to the original problem (1). For the split Lagrangian to converge, we need to optimize the Lagrange multipliers (i.e. solve the Lagrangian dual). For the maximization problem (1), the Lagrangian dual is given by the minimization problem in (8).

$$\Phi^{LD} = \left\{ \min_{\lambda} \left(\sum_{p\in[q]} \Phi^{LR}_p(\lambda) \right) : \lambda \in \mathbb{R}^n \right\} \tag{8}$$

In order to solve the Lagrangian dual (8), we update the Lagrange multipliers using the classical *subgradient method* where we let $\lambda^{k+1}_{i,m,t} = \lambda^k_{i,m,t} + \mu(b^{out\,k}_{i,m,t} - b^k_{i,m,t})$, in every iteration k. In the latter formula, μ is the *step-size parameter*, which we choose according to Poljak's rule (Poljak, 1969): $\mu = \eta_k \frac{\Phi^{LB} - \Phi^{LD}_k}{(b^{out\,k} - b^k)^2}$, with $\eta_k \in [0, 1]$ and Φ^{LB} being a lower bound for the optimum of (8). We refer the reader to the excellent survey of Beasley (1993) for more details on the choice of parameters in implementing the subgradient method.

4.3. Proposed heuristic algorithm

To be able to solve instances with hundreds of millions of variables and constraints (see Section 5), we combine the multiple strategies presented above (condensed formulation, balanced partition of the network, Split Lagrangian algorithm) into the heuristic algorithm described in Algorithm 1. Please refer to Appendix Appendix B for a diagram representation of the main steps in Algorithm 1.

Algorithm 1 Solution algorithm for tfMCF with rounding heuristic

1: Partition the full network into a set $\mathcal{P}$ of q balanced regions using Metis.
2: Condense the formulation for this reduced set of paths, bucketizing the origins for a given ϵ-approximation, and generate the RTN subproblem for each region using the condensed formulation.
3: Initialize the Lagrange multipliers $\lambda = 0$.
4: Calculate the $\hat{\Phi}_p^{cost} \forall p \in [q]$ splitting $\hat{\Phi}^{cost}$ proportionally to the total package cube assigned to each region p.
5: **for all** $\text{iter}^{LP} \in \{0, 1, \ldots, \text{iter}^{\max}\}$ **do**
6: **for all** $p \in [q]$ **do**
7: Solve the LP-relaxation of the regional (condensed) subproblems *in parallel*, given the regional cost bound $\hat{\Phi}_p^{cost}$.
8: Use subgradient method to update the Lagrange multipliers.
9: **end for**
10: **end for**
11: Compute the (relaxed) upper bound $UB = \sum_{p \in [q]} \Phi_p$.
12: Fix the task occurrence variables $y_{i,t}$ to zero if their solution in the LP-relaxation of the Lagrangian subproblems is less than or equal to a given rounding threshold ζ.
13: If $y_{i,t} \leq \zeta \forall i \in \mathcal{I}_e^T, t \in \mathcal{T}^{day}$ for an edge $e \in E$, fix to 1.0 the *transportation* task occurrence with the highest value in the LP.
14: Solve the resulting *reduced MIP* (*rounding heuristic*).
15: Update the lower bound *LB* with the cost of the solution of the *reduced MIP*.
16: **if** $\frac{UB-LB}{UB} > \delta$ **then** ▷ duality gap threshold not attained
17: Solve warm-started version original MIP (1), providing the LP solution and the MIP feasible solution of the *rounding heuristic* to the solver. When solving the LP relaxation of the root node with Dual Simplex, update the *UB* at each iteration. If at any point $\frac{UB-LB}{UB} \leq \delta$, **stop**.
18: **end if**

For some very-large-scale instances, even solving the linear relaxation can prove to be a challenge. Therefore, we use Lagrangian Decomposition to solve the linear relaxation of original MIP (1), and limit the number of iterations to $\text{iter}^{LP} \in \{0, 1, \ldots, \text{iter}^{\max}\}$. In our experiments (Section 5), we observe that there are significant similarities between the linear relaxation and the MIP solution, and 70–80% of the discrete variables have the same value in the linear relaxation and original MIP. We take advantage of this feature by proposing a *rounding heuristic* that leverages the solution of the linear relaxation of the Lagrangian subproblems to reduce the search space of the MIP, and improve its tractability.

Based on a given a *rounding threshold* $\zeta \in [0, 0.5]$, we fix a subset of the task occurrences (discrete variables) to zero if their solution in the linear relaxation of the Lagrangian subproblem is below the threshold, $y_{i,t} \leq \zeta$ (line 12 in Algorithm 1). However, this heuristic can make low volume edges infeasible by fixing all *transportation* task occurrences in an edge to zero (i.e., deactivate) because their values in the LP relaxation are all below the *rounding threshold*. In order to mitigate this behavior, for edges in which all *transportation* task occurrences are below the *rounding threshold*, $y_{i,t} \leq \zeta \forall i \in \mathcal{I}_e^T, t \in \mathcal{T}^{day}$ (where $\mathcal{I}_e^T$ is the set of *transportation* task in edge $e \in E$), we fix to 1.0 (i.e., activate) the *transportation* task occurrence with the highest value in the LP (line 13 in Algorithm 1)

The solution of this reduced MIP is a primal solution (lower bound) to (1), while the sum of the linear relaxation of the Lagrangian subproblems gives a valid dual bound (upper bound) to (1). If the optimality gap between the upper (*UB*) and lower bound (*LB*) is within a pre-specified tolerance, the algorithm stops, else, it goes to a final step where we try to solve the original MIP (1) but provide a warm-start solution to the root node LP (using the solution of the LP root node from the reduced MIP), and a warm-start feasible solution (using the optimal solution of the reduced MIP) (line 17 in Algorithm 1.). In order to stop the solution as soon as the duality gap tolerance, δ, is satisfied, we add a callback function to the Dual Simplex solver used in the root node LP of this warm-started original MIP. At each iteration of the Dual Simplex, we check if the current best solution of the LP is lower than the current upper bound *UB*. If so, we update the *UB* and check again the optimality criterion, $\frac{UB-LB}{UB} \leq \delta$. If satisfied, we interrupt the solution of the warm-started original MIP.

Table 1
Statistics of the tested instances.

Instance	Nodes	Arcs	Paths
Small	892	4,213	8,239
Mid-size	910	5,312	26,593
Large	960	7,337	63,844
Huge	1,226	10,003	102,476

5. Computational results

In this section we report on computational experiments with Algorithm 1 which was implemented in Python and uses Xpress, version 8.11, as the MIP solver. All runs were made on an AWS instance of type r5b.24xlarge equipped with 768.0 GiB of memory and 96 virtual CPUs.[1] The algorithm's performance was evaluated on four real instances of the tfMCF whose main characteristics are summarized in Table 1. Those instances were selected to be tested, as they illustrate the diversity of complexity we face in practice.

In each run reported in this section, the algorithm was allowed a maximum of 24 hours of computing time and the target duality gap threshold was fixed to 1%, meaning that the optimization was halted whenever the upper (*UB*) and lower (*LB*) satisfy $\frac{(UB-LB)}{UB} \times 100 \leq 1$. In all runs we assume that time is discretized into hourly intervals, and round up the task duration to the nearest hour.

We focus our analysis mainly on two aspects. The first aims to assess the benefits of the condensation technique discussed in Section 3. The second is devoted to measure the quality of the solutions produced by Algorithm 1 for different parameter settings.

The impact of condensation To evaluate the benefits of condensation, for each instance in our benchmark, we solve the MIP (1) obtained for different values of the *condensation factor* ϵ in Eq. (2). We compare the runs with no condensation and those with ϵ set to 0.5, 1 and 3. The results are summarized in Table 2. The symbols "-" in the cells in columns Dual bound and Primal bound

[1] More details on the hardware can be obtained at https://instances.vantage.sh/?min_memory=769filter=r5b.24xlargeregion=us-west-2r5b.24xlarge.

Table 2
Condensation results.

Instance	Condensation setup	OD buckets	Number of resources	Number of tasks	Number of extents	Number of constraints	Number of variables	Dual bound	Primal bound	Duality gap	Solution time (hours)
Small	No condensation	-	64,514	6,353	56,275	5,596,525	10,247,712	35,115,300	35,115,300	0.00%	0.66
	Eps = 0.5	766	40,079	6,353	39,323	4,293,626	6,877,608	35,174,041	35,020,778	0.44%	0.26
	Eps = 1	659	39,705	6,353	39,056	4,247,334	6,797,760	35,174,110	35,046,079	0.36%	0.28
	Eps = 3	581	39,462	6,353	38,891	4,219,442	6,750,720	35,174,248	35,040,885	0.38%	0.27
Mid-size	No condensation	-	250,151	7,588	224,050	32,722,462	61,689,408	38,985,868	24,451,854	37.28%	24.00
	Eps = 0.5	1,807	136,970	7,588	135,696	21,004,802	34,856,712	39,790,074	39,513,154	0.70%	7.34
	Eps = 1	1,409	132,705	7,588	131,858	20,326,220	33,580,944	39,806,583	39,426,535	0.95%	6.99
	Eps = 3	1,064	128,049	7,588	127,594	19,577,081	32,176,752	39,827,647	39,432,577	0.99%	5.72
Large	No condensation	-	681,255	9,699	618,265	98,681,068	187,452,840	46,808,804	-	-	24.00
	Eps = 0.5	2,768	304,976	9,699	305,123	51,023,455	84,314,904	47,034,695	43,282,693	7.98%	24.00
	Eps = 1	2,131	301,550	9,699	302,354	50,511,632	83,390,592	47,041,388	-	-	24.00
	Eps = 3	1,573	297,560	9,699	298,964	49,908,240	82,283,136	47,045,778	-	-	24.00
Huge	No condensation	-	1,083,133	12,908	982,555	154,952,398	294,331,056	-	-	-	24.00
	Eps = 0.5	3,326	478,861	12,908	479,485	79,604,076	130,994,592	70,203,569	-	-	24.00
	Eps = 1	2,576	474,200	12,908	475,620	78,893,752	129,700,944	70,208,038	-	-	24.00
	Eps = 3	1,948	469,238	12,908	471,362	78,130,717	128,290,704	70,212,201	-	-	24.00

mean that Xpress was unable to produce the respective bound. Similarly, in column Duality gap, this symbol is used to denote that the gap could not be computed due to the absence of at least one of the bounds.

Inspecting the data in the last two columns of the table already gives an overall idea of the gains produced by condensation. This is best illustrated by the results of the Mid-size instance. Without condensation, after 24 hours of computation, the duality gap was 37.28%, far above the admissible threshold of 1%. The smallest condensation factor was enough to reach this threshold in 7.34 hours, and increasing ϵ only made it faster to attain the required solution quality. For the Small instance, no condensation was necessary to find an optimal solution within the time limit. Nevertheless, for all three condensation factors tested the target duality gap threshold was reached in about half of the time. Notice that, in all but one case, Xpress was unable to produce a feasible (primal) solution for the MIPs associated to instances Large and Huge. This highlights the need of a heuristic approach as provided by Algorithm 1.

Note that the bigger the instance, the bigger the impact of condensation, potentially because there are more paths to be grouped. Recall that condensation affects the number of resources and extents, but not the number of tasks. Consequently, the number of integer variables in the MIP remains the same, while the overall number of variables and constraints reduces by a factor of 25%-50%. Also, one can see that the dual bounds get looser as the value of ϵ increases. But, remarkably, the deterioration is small, showing that the condensed formulation is an useful approximation for the original MIP. Finally, it is worth noting that condensation was essential to allow the solution of the LP relaxation of the Huge instance, for which Xpress failed to compute the optimal of the original LP after 24 hours.

The impact of the heuristic and its settings. We now report on the results we obtained using the heuristic in Algorithm 1. The heuristic was tested with two different partition strategies. In the first one, named the "Full-space heuristic", the network was left unpartitioned, which is equivalent to set $q = 1$ in line 1 of the algorithm. In the second one, called the "Decomposition heuristic", the network was partitioned into $q = 20$ regions by Metis using an imbalance factor of 0.2, and Lagrangian Decomposition step is limited to 3 iterations, $\text{iter}^{\max} = 2$. The goal of testing these two strategies was to identify when the decomposition becomes necessary to ensure scalability. Table 3 displays the results obtained. In all runs the condensation factor ϵ was set to 3. The rows identified by the "No heuristic" in the "Condensation setup" column reproduce results shown in Table 2 for $\epsilon = 3$ and are kept here for reference.

For each instance and each strategy, three values were tested for the rounding threshold ζ in line 12 in Algorithm 1: 0.1, 0.2, and 0.3. As expected, in all cases, the number of integer variables that are fixed increases as ζ augments with the percentage ranging from about 80% to 90% of their total. The trade-off here is clear: the higher the number of integer variables fixed, the easier/faster it should be to find the solution of the reduced MIP, but the more suboptimal it may be relative to the original MIP. However, for the values of ζ used in our tests, the loss in solution quality was small as can be observed in columns "Final primal bound" and "Final duality gap". For all instances, the "Decomposition heuristic" strategy fixes slightly more variables than "Full-space heuristic" when $\zeta \in \{0.1, 0.2\}$ while, for some as yet unknown reason, the reverse happens when the rounding threshold equals 0.3.

The importance of the network partitioning becomes evident for the Huge instance, which can not be solved with the "Full-space heuristic" for the two smallest rounding thresholds. Note that for this instance and also for Large, the "Decomposition heuristic" with $\zeta = 0.1$ is the only version that reached the threshold duality gap of 1% in less than 24 hours of run-time.

Another remarkable result can be noticed by inspecting columns "Primal bound from heuristic" and "Duality gap after solving LP relaxation of warm-started MIP". From there one sees that all primal bounds found in the reduced MIP heuristic are within 2% duality gap. Therefore, if the optimality tolerance was 2%, we could have stopped there and saved considerable time (compare the times reported in columns "Solution time LP relaxation of warm-started MIP" and "Total solution time"). We have observed that, if we stick to the 1% duality gap threshold, it pays off to choose a $\zeta = 0.1$ because, even though it takes longer to solve the reduced MIP, it was not necessary to enter the step to solve the warm-started original MIP (line 17 of Algorithm 1). In fact, the experiments revealed that, unless the problem is small enough (Small and Mid-size cases), this step runs out of time without improving the primal and dual bounds. This can be seen by inspecting the columns "Duality gap after solving LP relaxation of warm-started MIP" and "Final duality gap" for instances Large and Huge.

Another analysis we made refers to the reason that led the algorithm to stop. In line 17, the original MIP is solved after being fed with the warm-start solution obtained by the *rounding heuris-*

Table 3
Heuristic results.

Instance	Heurisic setup	Rounding threshold	Number of discrete variables (% fixed in the heuristic)	LP/Lagrangian relaxation solution time (hours)	Dual bound from LP/ Lagrangian relaxation	Primal bound from heuristic (reduced MIP)	Optimality gap after solving heuristic (reduced MIP)	Solution time until heuristic (reduced MIP)	Final dual bound (LP relaxation of warm-started MIP)	Optimality gap after solving LP relaxation of warm-started MIP	Solution time LP relaxation of warm-started MIP	Final primal bound (warm-started MIP)	Final optimality gap	Total solution time (hours)
Small	No heuristic	-	101,112 (0.00%)	0.03	35,174,248	-	-	-	35,174,248	-	-	35,040,885	0.38%	0.27
	Full-space heuristic	0.1	101,112 (84.13%)	0.03	35,174,248	35,073,706	0.29%	0.17	35,174,248	0.29%	0.17	35,073,706	0.29%	0.17
		0.2	101,112 (87.53%)	0.03	35,174,248	34,952,904	0.63%	0.17	35,174,248	0.63%	0.17	34,952,904	0.63%	0.17
		0.3	101,112 (89.79%)	0.03	35,174,248	34,764,609	1.16%	0.16	35,174,248	1.16%	0.16	35,174,248	0.00%	0.34
	Decomposition heuristic	0.1	101,112 (85.91%)	0.02	38,750,928	34,909,415	9.91%	0.15	35,179,329	0.77%	0.16	34,909,415	0.77%	0.16
		0.2	101,112 (87.89%)	0.02	38,750,928	34,931,036	9.86%	0.14	35,179,331	0.71%	0.15	34,931,036	0.71%	0.15
		0.3	101,112 (89.65%)	0.02	38,750,928	34,722,382	10.40%	0.14	35,174,248	1.28%	0.16	34,974,044	0.57%	0.31
Mid-size	No heuristic	-	127,704 (0.00%)	0.41	39,827,647	-	-	-	39,827,647	-	-	39,432,577	0.99%	5.72
	Full-space heuristic	0.1	127,704 (81.23%)	0.41	39,827,647	39,572,443	0.64%	1.68	39,827,647	0.64%	1.68	39,572,443	0.64%	1.68
		0.2	127,704 (86.70%)	0.41	39,827,647	39,533,959	0.74%	2.07	39,827,647	0.74%	2.07	39,533,959	0.74%	2.07
		0.3	127,704 (89.28%)	0.41	39,827,647	39,304,788	1.31%	1.17	39,827,647	1.31%	1.17	39,819,143	0.02%	3.31
	Decomposition heuristic	0.1	127,704 (84.41%)	0.14	40,254,551	39,498,095	1.88%	1.18	39,877,118	0.95%	1.25	39,498,095	0.95%	1.25
		0.2	127,704 (87.40%)	0.14	40,254,551	39,379,481	2.17%	0.95	39,827,647	1.13%	1.15	39,827,647	0.00%	5.95
		0.3	127,704 (89.03%)	0.14	40,254,551	39,224,201	2.56%	0.85	39,827,647	1.52%	1.08	39,436,453	0.98%	6.87
Large	No heuristic	-	176,256 (0.00%)	0.52	47,045,778	-	-	-	47,045,778	-	-	-	-	24.00
	Full-space heuristic	0.1	176,256 (81.09%)	0.52	47,045,778	46,478,133	1.21%	11.94	47,045,778	1.21%	11.94	46,478,133	1.21%	24.00
		0.2	176,256 (87.89%)	0.52	47,045,778	46,392,570	1.39%	6.59	47,045,778	1.39%	6.59	46,392,570	1.39%	24.00
		0.3	176,256 (90.73%)	0.52	47,045,778	46,132,479	1.94%	5.75	47,045,778	1.94%	5.75	46,132,479	1.94%	24.00
	Decomposition heuristic	0.1	176,256 (85.31%)	0.49	48,082,770	46,616,988	3.05%	6.52	47,078,799	0.98%	6.70	46,616,988	0.98%	6.70
		0.2	176,256 (89.01%)	0.49	48,082,770	46,519,167	3.25%	4.13	47,045,778	1.12%	5.07	46,519,167	1.12%	24.00
		0.3	176,256 (90.83%)	0.49	48,082,770	46,296,893	3.71%	3.69	47,045,778	1.59%	4.50	46,296,893	1.59%	24.00
Huge	No heuristic	-	240,336 (0.00%)	1.45	70,212,201	-	-	-	70,212,201	-	-	-	-	24.00
	Full-space heuristic	0.1	240,336 (79.45%)	1.45	70,212,201	-	-	24.00	70,212,201	-	24.00	-	-	24.00
		0.2	240,336 (86.59%)	1.45	70,212,201	-	-	24.00	70,212,201	-	24.00	-	-	24.00
		0.3	240,336 (89.63%)	1.45	70,212,201	69,361,047	1.21%	14.28	70,212,201	1.21%	14.28	69,361,047	1.21%	24.00
	Decomposition heuristic	0.1	240,336 (84.07%)	1.34	71,761,754	69,878,441	2.62%	21.47	70,508,661	0.89%	21.80	69,878,441	0.89%	21.80
		0.2	240,336 (88.10%)	1.34	71,761,754	69,682,942	2.90%	12.72	70,313,858	0.90%	13.09	69,682,942	0.90%	13.09
		0.3	240,336 (89.93%)	1.34	71,761,754	69,281,624	3.46%	11.39	70,212,201	1.33%	12.39	69,281,624	1.33%	24.00

tic. However, there is no need to wait for the solver to compute the optimal of the linear relaxation when a suboptimal LP solution is already enough to ensure that the 1% desired duality gap has been reached. This check can be done at any time by comparing the current LP objective function against the primal bound generated in line 17. Such a situation happened six times as indicated by the highlighted cells in column "Final dual bound".

6. Conclusion

In this paper we have proposed a *temporal fixed-charge multicommodity flow problem* (tfMCF) to capture the optimization of intra-day decisions in a transportation network in order to increase delivery speed, which in the application of interest means maximizing the number of packages delivered on time. We formulate the problem as a Resource Task Network (RTN), a framework commonly used for chemical process industry applications, and provide an example of how to map between the raw transportation network and the corresponding RTN.

The proposed mixed-integer program can become prohibitively large as the number of *origin, destination* (OD) pairs increases. Therefore, we have proposed a model condensation technique that groups similar OD pairs into buckets, and yields models whose number of variables and constraints are reduced by a factor of 25%–50%. We discuss theoretical properties of the condensed formulation and show that for some edge cases it is not guaranteed that the condensed formulation results in a feasible solution to the uncondensed problem. We prove that the condensed formulation is a relaxation to the uncondensed problem, and that, under mild assumptions, its solution is also feasible to the uncondensed problem. The computational results show that using the the condensed formulation was crucial to fully solve the Mid-size instance (62M variables and 33M constraints) and to solve the linear relaxation of the Huge instance (294M variables and 155M constraints) within the allowed computing time. Across all instances, the condensed formulation solves faster than the uncondensed formulation. The results also show that the dual bounds get looser as the value of the condensation factor ϵ increases, but the deterioration is small.

The proposed condensation alone is not sufficient to solve the two largest test instances (with 100M+ variables and constraints) efficiently. We have proposed a heuristic algorithm that combines balanced partitioning of the network, Split Lagrangian, and an LP filtering heuristic. The algorithm was tested for different rounding threshold ($\zeta = \{0.1, 0.2, 0.3\}$), as well as for a single region (full-space) and 20 regions. As expected, the number of fixed integer variables increases with the rounding threshold, ranging from 80 to 90% of the total discrete variables. The higher the number of variables fixed, the easier/faster it is to solve the reduced MIP, but the more suboptimal it may be compared to the original MIP. However, for all heuristic variations of all instances, if the algorithm was able to find a solution to the reduced MIP, this solution was already within 2% duality gap. For a stricter optimality tolerance of 1% it pays off to use $\zeta = 0.1$ and decompose the problem into 20 regions. This is the only heuristic configuration that found a solution within 1% optimality tolerance for the Large and Huge instances in less than 24 hours of run-time.

Overall, the modeling and solution methodology developed in this work made it possible to address a real-world problem with application to e-commerce logistics, and to solve practical instances of the order of 100M+ variables and constraints within a day of solution time. As next steps, we would like to explore ways to co-optimize cost and speed to be able to evaluate their trade-off, and to explore more heuristic techniques that can potentially reduce the solution time even further.

Appendix A. RTN Formulation for tfMCF

For our tfMCFproblem, a few simplifications are introduced to the conventional RTN MIP formulation notation, taking advantages of the specific task-resource interactions modeled in the transportation network. For illustration, starting from the multi-extent resource balance equation in Wassick & Ferrio (2011):

$$R_{r,t} = R_{r,t-1} + \sum_{i\in\mathcal{I}}\sum_{\theta=0}^{\tau_i} \alpha_{i,r,\theta} y_{i,t-\theta} + \sum_{i\in\mathcal{I}}\sum_{m\in\mathcal{M}_i}\sum_{\theta=0}^{\tau_i} \beta_{i,m,r,\theta} b_{i,m,t-\theta} + \Pi_{r,t},$$
$$\forall r \in \mathcal{R}, t \in \mathcal{T},$$

where, $r \in \mathcal{R}$ is the set of resources, $t \in \mathcal{T}$ is the set of time intervals, $i \in \mathcal{I}$ is the set of tasks, $m \in \mathcal{M}_i$ is the set of extents of task i, $R_{r,t}$ is the excess resource level of resource r at time t, $y_{i,t}$ represents task i starting at time t, $b_{i,m,t}$ represents the size of extent m of task i starting at time t, $\Pi_{r,t}$ represents the external supply or consumption of resource r at time t, τ_i is the length of task i in terms of integer multiple of the unit grid length, $\alpha_{i,r,\theta}, \beta_{i,r,m,\theta}$ are the discrete and continuous resource task interaction parameters.

1. The discrete resource task interactions are dropped, i.e. $\alpha_{i,r,\theta} = 0$. Packages are treated as continuous resources in this study. For transportation tasks, vehicle resources are not explicitly considered, and the cost is calculated using the integer task occurrence variables instead. This leads to fewer resources defined and therefore reduced number of variables in the resultant MIP problem.
2. For continuous interactions, there is no intra-task resource production or consumption. More specifically, resource depletion only occurs at the beginning of a task, and generation at the end. There are no tasks that split a resource into multiple ones (no fractional interaction parameters). Therefore, it holds that for all tasks, $\beta_{i,m,r,\theta=0} = -1$ for resource consumption, and $\beta_{i,m,r,\theta=\tau_i} = 1$ for resource generation (otherwise, $\beta_{i,m,r,\theta} = 0$). We further drop the task index i and use extent only for the ease of notation, and also use τ_m for duration in place of τ_i as they are always equal. Finally, the continuous task resource interaction term $\sum_{i\in\mathcal{I}}\sum_{m\in\mathcal{M}_i}\sum_{\theta=0}^{\tau_i} \beta_{i,m,r,\theta} b_{i,m,t-\theta}$ is rewritten as $\sum_{m\in\mathcal{M}_r^+} b_{m,t-\tau_m} - \sum_{m\in\mathcal{M}_r^-} b_{m,t}$, where $\mathcal{M}_r^+$ and $\mathcal{M}_r^-$ denote extents that feed and deplete resource r, respectively. Here we make the use of the fact that task extents of different tasks do not overlap, and dropping the task index does not change the resultant constraints.

Appendix B. Solution framework

Figure B.1 shows the flowchart of the proposed Algorithm 1.

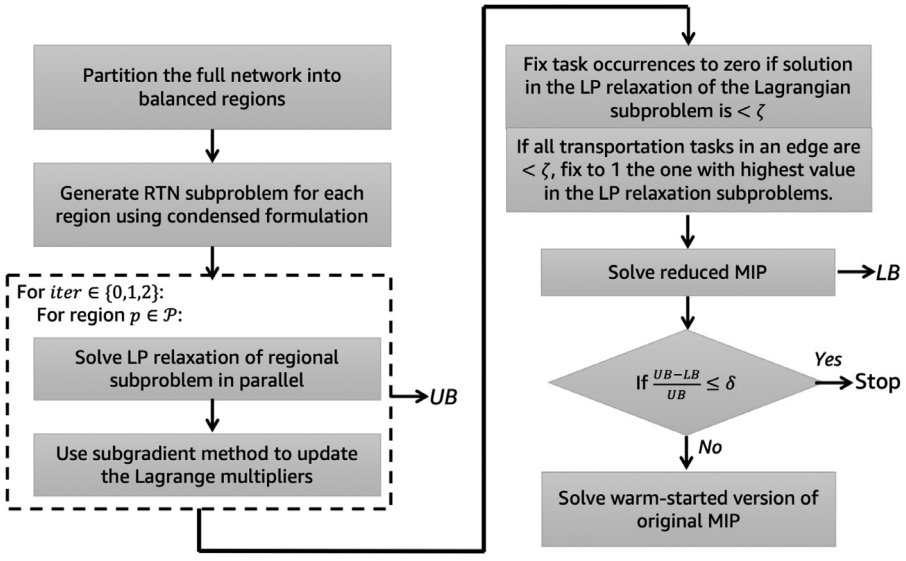


Fig. B.1. Overview of the proposed solution methodology.

References

Adjiashvili, D., Bosio, S., & Weismantel, R. (2012). Dynamic combinatorial optimization: A complexity and approximability study. http://www.iasi.cnr.it/aussois/web/uploads/2012/papers/bosios.pdf.

Adjiashvili, D., Bosio, S., Weismantel, R., & Zenklusen, R. (2014). Time-expanded packings. In J. Esparza, P. Fraigniaud, T. Husfeldt, & E. Koutsoupias (Eds.), *International colloquium on automata, languages, and programming* (pp. 64–76). Berlin, Heidelberg: Springer Berlin Heidelberg.

Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1988). *Network flows*. Cambridge.

Balinski, M. L. (1961). Fixed-cost transportation problems. *Naval Research Logistics Quarterly, 8*(1), 41–54.

Beasley, J. E. (1993). Lagrangian relaxation. In C. R. Reeves (Ed.), *Modern heuristic techniques for combinatorial problems* (pp. 243–303). USA: John Wiley & Sons, Inc..

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

Boland, N. L., & Savelsbergh, M. W. P. (2019). Perspectives on integer programming for time-dependent models. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research, 27*(2), 147–173.

Bui, T. N., & Jones, C. (1992). Finding good approximate vertex and edge partitions is NP-hard. *Information Processing Letters, 42*(3), 153–159.

Castro, P. M., Barbosa-Póvoa, A. P., & Matos, H. A. (2003). Optimal periodic scheduling of batch plants using RTN-based discrete and continuous-time formulations: A case study approach. *Industrial & Engineering Chemistry Research, 42*(14), 3346–3360.

Castro, P. M., Barbosa-Póvoa, A. P., & Novais, A. Q. (2005). A design and scheduling RTN continuous-time formulation. *Computer Aided Chemical Engineering, 20*, 1213–1218.

Conforti, M., Cornuéjols, G., & Zambelli, G. (2014). *Integer programming*: vol. 271. Springer.

Courant, R., & Robbins, H. (1941). *What is mathematics?* (pp. 354–360). New York: Oxford university press.

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

Crainic, T. G. (2003). Long-haul freight transportation. In *Handbook of transportation science* (pp. 451–516). Springer.

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

Crainic, T. G., Gendreau, M., & Gendron, B. (2021). *Network design with applications to transportation and logistics*. Springer.

Cui, R., Lu, Z., Sun, T., & Golden, J. (2020). Sooner or later? Promising delivery speed in online retail. https://doi.org/10.2139/ssrn.3563404.

Fleischer, L., & Skutella, M. (2007). Quickest flows over time. *SIAM Journal on Computing, 36*(6), 1600–1630.

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

Fleischer, L. K. (2000). Approximating fractional multicommodity flow independent of the number of commodities. *SIAM Journal on Discrete Mathematics, 13*(4), 505–520.

Floudas, C. A., & Lin, X. (2004). Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review. *Computers & Chemical Engineering, 28*(11), 2109–2129.

Floudas, C. A., & Lin, X. (2005). Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications. *Annals of Operations Research, 139*(1), 131–162.

Ford, L. R., & Fulkerson, D. R. (1958). Constructing maximal dynamic flows from static flows. *Operations Research, 6*(3), 419–433.

Ford, L. R., & Fulkerson, D. R. (1962). *Flows in networks*. Princeton University Press.

Garg, N., & Könemann, J. (2007). Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing, 37*(2), 630–652.

Gomory, R. E., & Hu, T. C. (1961). Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics, 9*(4), 551–570.

Grossmann, I. E.. Tutorial on Lagrangean decomposition: Theory and applications.

Grossmann, I. E., Quesada, I., Raman, R., & Voudouris, V. T. (1996). Mixed-integer optimization techniques for the design and scheduling of batch processes. In G. V. Reklaitis, A. K. Sunol, D. W. T. Rippin, & Ö. Hortaçsu (Eds.), *Batch processing systems engineering* (pp. 451–494). Springer.

Guignard, M., & Kim, S. (1987). Lagrangean decomposition: A model yielding stronger Lagrangean bounds. *Mathematical Programming, 39*(2), 215–228. https://doi.org/10.1007/BF02592954.

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

Harjunkoski, I., Maravelias, C. T., Bongers, P., Castro, P. M., Engell, S., Grossmann, I. E., & Wassick, J. (2014). Scope for industrial applications of production scheduling models and solution methods. *Computers & Chemical Engineering, 62*, 161–193.

Hewitt, M., Crainic, T. G., Nowak, M., & Rei, W. (2019). Scheduled service network design with resource acquisition and management under uncertainty. *Transportation Research Part B: Methodological, 128*, 324–343.

Hirsch, W. M., & Dantzig, G. B. (1968). The fixed charge problem. *Naval Research Logistics Quarterly, 15*(3), 413–424.

Hoch, B., Liers, F., Neumann, S., & Martınez, F. J. Z. (2020). The non-stop disjoint trajectories problem.

Jarrah, A. I., Johnson, E., & Neubert, L. C. (2009). Large-scale, less-than-truckload service network design. *Operations Research, 57*(3), 609–625.

Karakostas, G. (2008). Faster approximation schemes for fractional multicommodity flow problems. *ACM Transactions on Algorithms (TALG), 4*(1), 1–17.

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of computer computations* (pp. 85–103). Springer.

Karypis, G., Aggarwal, R., Kumar, V., & Shekhar, S. (1999). Multilevel hypergraph partitioning: Applications in VLSI domain. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 7*(1), 69–79.

Klinz, B., & Woeginger, G. J. (2004). Minimum-cost dynamic flows: The series-parallel case. *Networks: An International Journal, 43*(3), 153–162.

Kondili, E., Pantelides, C. C., & Sargent, R. W. (1993). A general algorithm for short-term scheduling of batch operations—I. MILP formulation. *Computers & Chemical Engineering, 17*(2), 211–227.

Lagos, F., Boland, N., & Savelsbergh, M. (2020). The continuous-time inventory-routing problem. *Transportation Science, 54*(2), 375–399.

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

Magnanti, T. L., & Wong, R. T. (1984). Network design and transportation planning: Models and algorithms. *Transportation Science, 18*(1), 1–55.

Minoux, M. (1989). Networks synthesis and optimum network design problems: Models, solution methods and applications. *Networks, 19*(3), 313–360.

Nemhauser, G. L., & Wolsey, L. A. (1988). Integer and combinatorial optimization. *Wiley Interscience Series in Discrete Mathematics and Optimization*. Wiley. https://doi.org/10.1002/9781118627372.

Newman, A. M., & Kuchta, M. (2007). Using aggregation to optimize long-term production planning at an underground mine. *European Journal of Operational Research, 176*(2), 1205–1218.

Pantelides, C. (1994). Unified frameworks for optimal process planning and scheduling. In *Proceedings on the 2nd conference on foundations of computer aided operations* (pp. 253–274). New York: Cache Publications.

Pessoa, A., Uchoa, E., de Aragão, M. P., & Rodrigues, R. (2010). Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation, 2*(3–4), 259–290.

Pinto, J. M., & Grossmann, I. E. (1998). Assignment and sequencing models for the scheduling of process systems. *Annals of Operations Research, 81*, 433–466.

Pochet, Y., & Warichet, F. (2008). A tighter continuous time formulation for the cyclic scheduling of a mixed plant. *Computers & Chemical Engineering, 32*(11), 2723–2744.

Poljak, B. (1969). Minimization of unsmooth functionals. *U.S.S.R. Computational Mathematics and Mathematical Physics, 9*, 14–29.

Powell, W. B., Jaillet, P., & Odoni, A. (1995). Stochastic and dynamic networks and routing. *Handbooks in Operations Research and Management Science, 8*, 141–295.

Skutella, M. (2009). An introduction to network flows over time. In W. Cook, L. Lovász, & J. Vygen (Eds.), *Research trends in combinatorial optimization* (pp. 451–482). Springer.

Vazirani, V. V. (2001). *Approximation algorithms*: vol. 1. Springer.

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

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

Wassick, J. M., & Ferrio, J. (2011). Extending the resource task network for industrial applications. *Computers & Chemical Engineering, 35*(10), 2124–2140. https://doi.org/10.1016/j.compchemeng.2011.01.010. https://www.sciencedirect.com/science/article/pii/S0098135411000123

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

Williamson, D. P. (2019). *Network flow algorithms*. Cambridge University Press.

Wu, Y., & Maravelias, C. T. (2019). A general model for periodic chemical production scheduling. *Industrial & Engineering Chemistry Research, 59*(6), 2505–2515.

Yee, K., & Shah, N. (1998). Improving the efficiency of discrete time scheduling formulation. *Computers & Chemical Engineering, 22*, S403–S410.