Recursive Partitioning and Batching for Massive-Scale Network Design with Service Time Guarantees

Myungeun Eom Alan Erera Alejandro Toriello
Stewart School of Industrial and Systems Engineering, Georgia Tech, Atlanta, USA

Keywords: freight transportation, service network design, heuristic, service guarantee

1 Introduction

Small package carriers, who specialize in shipments that are small relative to vehicle capacity, have become core elements of many supply chains, particularly in e-commerce. To efficiently handle a high volume of small packages, these carriers design repeatable primary service routes, which serve as the backbone of their daily operations. More precisely, given a repetitive pattern of shipment demands, or *commodities*, carriers plan routes and schedules for each shipment, aiming to minimize total transportation and operational costs. Some carriers also offer a *service time guarantee*, typically defined as the number of days within which a shipment is delivered (Bakir et al., 2021). Motivated by the operations of such carriers in the United States, we study a service network design problem with service time guarantees at countrywide scale.

Service network design for small package carriers is notoriously challenging, especially when dealing with large networks and many commodities (Lara et al., 2023). While researchers have proposed various solution approaches (Jarrah et al., 2009, Teypaz et al., 2010, Crainic et al., 2016, Boland et al., 2017, Hewitt and Lehuédé, 2023), many do not scale up to the instance sizes required by our motivating application. For example, Crainic et al. (2016) developed a method combining slope scaling, column generation, and meta-heuristics to solve the problem with resource constraints, effective on instances with up to 350 nodes, 1,850 arcs, and 400 commodities. In contrast, our problem involves over 1,000 nodes, 1 million arcs, and 40,000 commodities. Lara et al. (2023) studied an instance with a similar number of nodes and more commodities, but focused on scheduling given fixed paths for the commodities. Erera et al. (2013) proposed an improvement heuristic to solve an instance of similar scale to ours; in contrast, our goal is constructing a solution, and we use recursive graph partitioning and batching. We demonstrate the scalability and efficiency of our approach through computational studies on real-world instances from an industry partner.

2 Problem Description

As in many service network design studies, we model the operations of a carrier with a time-space network, $G = (N, A)$, where terminal activities in different shifts are modeled with different nodes. Specifically, given the set of terminals L in the carrier's network and the set of labor shifts T in the terminals, we define the set of nodes as $N = L \times T$. Shipments can occur between any two terminals, defining arcs A over the network. Each arc $(i, j) \in A$ has a transit time t_{ij} and distance m_{ij}; the transit time and origin shift (arc tail) determine the destination shift (arc head) in the time-space network. Only a subset of nodes can act as transfer nodes, so the set of feasible arcs for each commodity is not necessarily identical. We consider a representative day of demand and assume the time-space network is a single, "wrapped" day.

The set of commodities K captures this daily demand; commodity k represents an aggregated demand from origin node o_k to destination node d_k with a service time guarantee of g_k days and volume q_k. We assume homogeneous trailers with capacity Q. The transportation cost is incurred by trailer movements, where the number of trailers per arc is determined by the total volume of commodities flowing on the arc and the trailer capacity. We also consider handling cost of f^H per volume, incurred each time a commodity is processed at a transfer node. The goal is to

determine paths for all commodities that satisfy service guarantees while minimizing the sum of transportation and handling costs.

We formulate the problem as MIP (1) with two types of decision variables: $x_{ij}^k \in \{0,1\}$ indicates whether the path for commodity k includes arc (i,j), defined only for a commodity's feasible arcs; $z_{ij} \in \mathbf{Z}_+$ represents the number of trailers on arc (i,j). Our approach, detailed in the next section, iteratively solves a restriction of this model on top of a partial solution. That is, predetermined arc flow p may be given as an input in addition to graph $G = (N, A)$ and commodity set K. To clarify the input, we use the notation MIP (G, K, p):

$$\min_{x_{ij}^k \in \{0,1\},\ z_{ij} \in \mathbf{Z}_+} \sum_{(i,j)\in A} m_{ij} z_{ij} + f^H \sum_{k\in K} q_k \sum_{(i,j)\in A:\ i,j \text{ in different terminals}} x_{ij}^k$$

$$\text{s.t.} \sum_{(i,j)\in A} x_{ij}^k - \sum_{(j,i)\in A} x_{ji}^k = \begin{cases} 1 & \text{if } i = o_k \\ -1 & \text{if } i = d_k \\ 0 & \text{otherwise} \end{cases} \quad \forall k \in K, i \in N \tag{1a}$$

$$p_{ij} + \sum_{k\in K} q_k x_{ij}^k \le z_{ij} \times Q \quad \forall (i,j) \in A \quad (1b), \qquad \sum_{(i,j)\in A} t_{ij} x_{ij}^k \le g_k \quad \forall k \in K \tag{1c}$$

The objective function minimizes the total cost. Constraints (1a) are flow balance constraints, (1b) determine the number of trailers on each arc, and (1c) ensure service guarantees.

3 Solution Approach

We begin by outlining the motivation behind our solution approach. The challenge of the large-scale service network design arises from two factors: the network size and the number of commodities. To reduce the network size, we partition the network into smaller *regions* and then solve the problem for each region, considering only commodities whose origin and destination are in the same region. This strategy naturally avoids paths with excessive detours. For commodities that cross between regions (referred to as *crossing commodities*), we first determine appropriate subnetworks and then solve the MIP (1) on the subnetworks. To address the complexity from a large number of commodities, we divide the commodities into batches and solve the problem iteratively; after solving for each batch, we fix the solution and proceed with the next batch, repeating this procedure until we consider all commodities.

We now formally describe our main algorithm. The set of terminals L and the set of commodities K are given as input. If graph G induced by L is small enough, then we directly apply the batching algorithm (Algorithm 1) on G, K, and zero initial arc flow. Otherwise, we partition L into $\{L_1, \ldots, L_r\}$ by using a partitioning logic. It results in partitioning K into $\{K_1, \ldots, K_r, K_c\}$ where K_r represents the set of commodities with both origin and destination terminals in L_r and K_c represents the set of crossing commodities. We then run the main algorithm for each region (i.e. using input L_r and K_r for region r) independently and in parallel. For any commodities without feasible paths within their regions, we merge them into K_c. After obtaining solutions for all regions, we run the crossing commodity algorithm (Algorithm 2) for K_c with initial arc flow $\sum_r x_r$ where x_r is the solution for region r.

Algorithm 1 describes how we determine batch size b and select commodity batches. When selecting a batch, we prioritize commodities with less flexibility to enable consolidation. Specifically, we compute a *slack* for each commodity and consider commodities in increasing order of slack. We note that the set of possible slack values is discrete and typically small, usually up to a size of 4. Among commodities with the same slack, we randomly choose up to b commodities with selection probability proportional to the commodity's volume.

For partitioning, we utilize existing geographic partitions, namely the census divisions of the US (United States Census Bureau, 2013). We divide the US into four regions (West, Midwest, South, Northeast) and, if necessary, partition each region into multiple sub-regions.

Algorithm 1 Batching (Least Flexible/High Volume First)

1: Input: Graph G, Set of commodity K, Predetermined arc flow x
2: Compute slack $s_k = g_k -$ shortest transit time in days from o_k to d_k in G for each $k \in K$
3: Set of infeasible commodities $K^0 \leftarrow \{k \in K : s_k < 0\}$, $K \leftarrow K \setminus K^0$
4: **for** each (slack s, service time guarantee g) **do**
5: $K_{s,g} = \{k \in K : s_k = s, g_k = g\}$
6: Calculate batch size $b \leftarrow \max\{1, \lfloor \frac{\text{estimated number of decision variables in MIP}(G, K_{s,g}, x)}{\text{threshold on the number of decision variables}} \rfloor\}$
7: **while** $K_{s,g} \neq \emptyset$ **do**
8: $\tilde{K} \leftarrow$ Randomly choose $\min\{b, |K_{s,g}|\}$ commodities (selection probability for $k \propto q_k$)
9: $x \leftarrow x +$ solution from MIP$(G, \tilde{K}, x)$, Remove $\tilde{K}$ from $K_{s,g}$
10: **return** x, K^0

Finally, Algorithm 2 presents how we construct subnetworks for crossing commodities and the order in which we process crossing commodities. It first selects a set of arcs A^* to use (line 2) and constructs a subcomponent-based adjacency graph (lines 3-10). Using A^* and the adjacency graph, it solves the problem for crossing commodities in increasing order of subcomponent-based distance (lines 5-10).

Algorithm 2 Crossing Commodity

1: Input: Partition of terminals $\{L_1, \ldots, L_r\}$, Set of commodity K, Predefined flow x
2: $A^* \leftarrow$ *ArcSelection*$(\{L_1, \ldots, L_r\}, K, x)$
3: Define set of subcomponents S by partitioning regions, Construct adjacency graph H of S
4: Sort $\{(s_o, s_d) : s_o, s_d \in S$ in different regions$\}$ in increasing order of distance in H
5: **for** each (s_o, s_d) **do**
6: $K^* \leftarrow \{k \in K : o_k$ is in s_o, d_k is in $s_d\}$
7: $L^* \leftarrow \{l \in L :$ subcomponent of l is included in any shortest path from s_o to s_d in $H\}$
8: Construct a subgraph G^* induced by L^* and A^*
9: For commodity $k \in K^*$ with no feasible paths in G^*, add arc (o_k, d_k) to G^*
10: Add solution from *Batching*(G^*, K^*, x) to x
11: **return** $x, \emptyset$

The arc selection process begins with arcs that have already been used within regions and adds additional arcs through four steps: (1) Add arcs from origin to destination nodes, say *OD arcs*, for the top α% of high-volume commodities in K. (2) Add the top β% nearest missing arcs between transfer nodes in different regions. (3) For each commodity with no feasible path using only A^*, add arcs from its origin/destination node to the γ nearest transfer nodes. (4) For each commodity still without any feasible path using arcs in A^*, add its OD arc (if it is the only feasible path in the original graph), or add the arcs in the shortest path with at least one transfer in the original graph. We choose the best combination of (α, β, γ) through preliminary experiments.

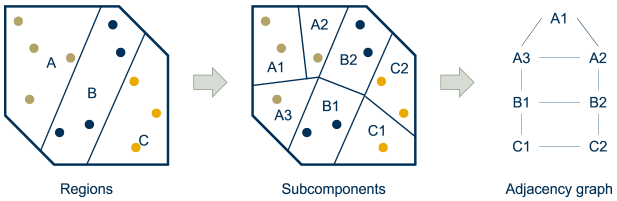

Figure 1 – *Construction of a subcomponent-based adjacency graph*

The number of selected arcs may still be large, resulting in small batch size. To handle this, we partition each region into subcomponents and identify subcomponents connecting each subcomponent pair (lines 3-10). For example, for commodities with origin in A3 and destination in C1, since the shortest path from A3 to C1 is A3-B1-C1 in Figure 1, we generate a subgraph using arcs in A^* whose endpoints are in {A3,B1,C1}.

4 Computational Study

We conduct experiments to demonstrate the effectiveness of our approach on an industrial-scale instance with over 1,000 nodes, 1 million arcs, and 40,000 commodities, using real-world data provided by an industrial partner. We compare the performance of our algorithm (Proposed) against the current plan used by the partner (Baseline), using two handling cost coefficients (0.2 and 0.4). We performed experiments on a high-performance computing cluster with AMD EPYC-Rome processors and 80 GB of RAM, using Python for implementation and Gurobi 10.01 for solving MIPs. We solved the problem for four regions (West, Midwest, South, and Northeast) in parallel, each with four random seeds and four subregion configurations. After that, on top of the best solutions from each region, we ran the crossing commodity algorithm with three configurations of (α, β, γ). We report the best results for each handling cost coefficient in Table 1.

f^H	Method	Obj	Tcost	Hcost	Tnum	APL	Util	Time (s)
0.2	Proposed	14,350,176	7,667,637	6,682,540	20,103	1.37	0.84	17794 + 29343
	Baseline	15,505,778	8,359,403	7,146,375	22,558	1.37	0.78	-
0.4	Proposed	20,765,120	8,067,411	12,697,709	20,317	1.31	0.79	16056 + 31342
	Baseline	22,652,153	8,359,403	14,292,750	22,558	1.37	0.78	-

Table 1 – *Results from our proposed method and the current plan used by our industrial partner*

Obj, Tcost, and Hcost refer to the objective, transportation cost, and holding cost, respectively. Tnum represents the total number of trailers, APL is volume-weighted average path length in solution paths, and Util measures distance-weighted trailer utilization. Time indicates the total solution runtime, where the first value represents the time to solve the region sub-problems and the second represents the time for the crossing commodity algorithm. As shown in Table 1, our method achieves solutions outperforming the baseline in 13.2 hours, which demonstrates both scalability and efficiency. Notably, we could not even build a Gurobi model for the instance defined by the South region. In addition, the average path length and trailer utilization of our solutions decrease with larger handling cost; this indicates we can control the structure of our solution by adjusting the handling cost coefficient.

References

I. Bakir, A. Erera, and M. Savelsbergh. Motor carrier service network design. In T. G. Crainic, M. Gendreau, and B. Gendron, editors, *Network Design with Applications to Transportation and Logistics*, pages 427–467. Springer, 2021.

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

T. G. Crainic, M. Hewitt, M. Toulouse, and D. M. Vu. Service network design with resource constraints. *Transportation Science*, 50(4):1380–1393, 2016.

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

M. Hewitt and F. Lehuédé. New formulations for the scheduled service network design problem. *Transportation Research Part B: Methodological*, 172:117–133, 2023.

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

C. L. Lara, J. Koenemann, Y. Nie, and C. C. de Souza. Scalable timing-aware network design via Lagrangian decomposition. *European Journal of Operational Research*, 309(1):152–169, 2023.

N. Teypaz, S. Schrenk, and V.-D. Cung. A decomposition scheme for large-scale service network design with asset management. *Transportation Research Part E: Logistics and Transportation Review*, 46(1): 156–170, 2010.

United States Census Bureau. Census regions and divisions of the united states. https://www.census.gov/geo/pdfs/mapsdata/maps/reference/us_regdiv.pdf, 2013. Accessed: 2024-08-12.