A Time-Expanded Network Reduction Matheuristic for the Logistics Service Network Design Problem

Simon Belieres, Mike Hewitt, Nicolas Jozefowiez, Frédéric Semet

A Time-Expanded Network Reduction Matheuristic for the Logistics Service Network Design Problem

Simon Belieres[a,*], Mike Hewitt[b], Nicolas Jozefowiez[c], Frédéric Semet[d]

[a] *CNRS, LAAS, 7 Avenue du Colonel Roche, 31077 Toulouse Cedex 4, France*
[b] *Quinlan School of Business, Loyola University, 16 E. Pearson Ave., IL, Chicago 60611, USA*
[c] *LCOMS EA 7306, Université de Lorraine, Metz 57000, France*
[d] *Univ. Lille, CNRS, Centrale Lille, Inria, UMR 9189 - CRIStAL, F-59000 Lille, France*

Abstract

Planning cost-effective logistics operations involve the integration of multiple decision-making levels. In the domain of supply chain management, the last decades have seen the emergence of 3PL service providers that specialize in integrating warehousing and transportation services. In this paper, we study the operations performed by a 3PL in the supply chain management of a French restaurant chain. The transportation planning process is assisted by solving the Logistics Service Network Design Problem (LSNDP). As realistic instances are too large for on-the-shelf optimization solvers to solve in acceptable run-times, we develop a network reduction heuristic inspired by the recent Dynamic Discretization Discovery algorithm. Through an extensive series of experiments carried out on instances based on the operations of an industrial partner, we demonstrate the efficiency of the proposed approach. We also investigate the impact of the distribution strategy used in practice to determine the transportation plan and how this distribution strategy can be modified to reduce the overall logistics costs.

Keywords: Logistics, Service Network Design, Dynamic Discretization Discovery, Integer Programming, Heuristics

1. Introduction

We consider a transportation problem encountered by a third-party logistics (3PL) supporting the supply chain of a restaurant chain. The problem involves planning the transportation operations within a four-echelon network to fulfill customer product orders over the near future, typically the next month. To design the transportation plan, the 3PL follows a *distribution strategy* that enforce the use of centralized delivery paths, wherein all shipments must go through a primary warehouse and a secondary warehouse before reaching their destination.

The 3PL planning process can be assisted by solving the Logistics Service Network Design Problem (LSNDP). This problem combines features from the Service Network Design Problem (SNDP) [9, 24] and the Logistics Network Design Problem (LNDP) [20]. The LSNDP and the SNDP are both tactical planning problems that

*Corresponding author
Email address: simon.belieres@hec.ca (Simon Belieres)

seek to determine shipment itineraries within a terminal network and allocate transportation services to support the deliveries. However, while the SNDP makes no presumptions regarding the direction of shipment flows, the LSNDP considered in this paper seeks to design a "forward flow" network. In addition, most SNDP models studied in the literature presume that shipment origins and destinations are specified *a priori*, contrarily to the LSNDP, wherein customers request product deliveries that can be sourced by different suppliers. In that sense, the LSNDP is similar to supply chain optimization problems [3] as the Logistics Network Design Problem (LNDP) [21]. Nevertheless, the LNDP is a strategic problem that focuses on long-term decisions such as facility location [2, 8, 22], whereas the LSNDP assumes that the facilities and their capacities are already established. In Table 1, we compare the characteristics of the LSNDP, the SNDP, and the SNDP, and we report the decisions involved by each problem.

Table 1: Comparison of the LNDP, the SNDP and the LSNDP

	Logistic features		Decisions involved				
Problem	Multi-echelon network	Shipment origin not fixed	Location	Production	Distribution	Inventory	Vehicle utilization
LNDP	X	X	X	X	X	X	-
SNDP	-	-	-	-	X	X	X
LSNDP	X	X	-	-	X	X	X

Location = location of the platforms in the network; Production = quantities of raw materials to purchase and manufacture;
Distribution = flows of shipments along the network; Inventory = management of inventory levels;
Vehicle utilization = vehicle allocation to support distribution

The LSNDP is a recent problem with only two dedicated studies. Dufour et al. [12] study the role played by third-party logistics in the humanitarian sector. The paper describes a case study in East Africa and aims to optimize distribution costs in a humanitarian multi-product supply chain. To assess the value of adding a new regional distribution center to the supply chain, an optimization model is solved on two classes of instances that are based on the existing and the planned distribution network, respectively. The model is solved with a generic commercial solver and the results highlight the potential to significantly improve logistics costs by integrating the regional distribution center into the humanitarian supply chain. Belieres et al. [4] described the problem studied in this article. The problem is solved through a decomposition algorithm based on a partial Benders decomposition [10].

The multi-echelon distribution network considered in the LSNDP is modeled as a static network where each node corresponds to a supply chain stakeholder. A common way to capture the temporal dimension in network design problems consists in constructing a time-expanded network [13, 14] obtained from the static network, the planning horizon, and a chosen time discretization. On the one hand, as the duration of the time intervals used to define the time-expanded network impacts the quality of the solutions to the resulting model, a fine discretization is required to produce high-quality solutions [7]. On the other hand, how time is discretized has

a significant impact on the size of the model and its computational tractability. To face this challenge, Boland *et al.* [6] proposed the Dynamic Discretization Discovery (DDD) algorithm. The approach determines a sparse time-expanded network with certain properties, such that the solution of the associated program provides a lower bound for the original program. The sparse time-expanded network is refined while maintaining its properties and the reduced program is solved iteratively until the lower bound is feasible, and thus optimal, for the original program. Successful DDD-related solution approaches can be found in [15, 16, 17, 18, 19, 23].

For both theoretical and computational reasons, the DDD approach is not appropriate for solving the LSNDP. From a theoretical point of view, the proof, that DDD is converging to the optimal solution, is based in part on the assumption that product storage costs are zero, which is not the case for the LSNDP. From a computational perspective, DDD exploits the fact that shipments have known origins and destinations to maintain a network that is sparse and yields optimization problems that are computationally tractable. The LSNDP deals with product requests that can be sourced by multiple suppliers at different times. Therefore, a straightforward application of the DDD-approach leads to a dense initial network. To overcome these difficulties, we propose a time-expanded network reduction metaheuristic (TENMR) driven by the principles of DDD. We develop multiple innovations, such as the use of transportation arcs with underestimated costs, a new refinement mechanism to accomodate multi-echelon network structure, and a new procedure for constructing the initial network. Finally, we combine the metaheuristic with a decomposition algorithm to solve industrial size instances.

The paper makes the following contributions. First, it proposes an iterative matheuristic that produces an integer program that is significantly smaller than the integer program for the full problem and still provides close-to-the-optimum solutions. Second, it presents a real-life case study and proposes insights into how the distribution strategy influences the resulting transportation plan. Specifically, the computational study demonstrates that the 3PL can achieve substantial savings by modifying its distribution strategy.

The article is organized as follows. The problem and the current logistics management of our 3PL partner are described in Section 2. The mathematical model is proposed in Section 3. In Section 4, we present the Time-Expanded Network Reduction Matheuristic (TENRM). In Section 5, we report and discuss the results of an extensive computational study to assess the algorithm performance as well as the impact of the new distribution strategy considered by the 3PL. Conclusions and avenues for future work are presented in Section 6.

2. Problem description

In this section, we first characterize the supply chain setting considered in this study. We then describe the industrial application that falls within this setting. Finally, we define the decisions that are involved in the design of the transportation plan supporting this supply chain, as well as two distribution strategies to guide these decisions. The first distribution strategy is currently implemented by the 3PL and involves choosing delivery paths through the distribution network that utilize a centralized warehouse. The second distribution strategy is more flexible and

includes delivery paths that skip a central warehouse, including those that are direct from suppliers to customers.

2.1. A four-echelon supply chain

Supply chains are networks of partner companies that collaborate to create and distribute products to satisfy a consumer market. Stakeholders in a supply chain typically include suppliers, plants, warehouses, retailers, and customers. To distribute products to customers, the supply chain process involves a series of stages, with each stage associated with a type of stakeholder. Consequently, the underlying distribution network can be decomposed into multiple echelons, where each echelon groups together all stakeholders of the same type, and where links between the echelons characterize the order in which the stages are executed. In this paper, we study a commonly-used multi-product four-echelon supply chain setting that involves suppliers, primary warehouses, secondary warehouses, and customers. An example is illustrated in Figure 1, where $\mathcal{S}_x$ designates a supplier facility, $\mathcal{W}_x^{\mathcal{P}}$ denotes a primary warehouse, $\mathcal{W}_x^{\mathcal{S}}$ indicates a secondary warehouse and $\mathcal{C}_x$ characterizes a customer.

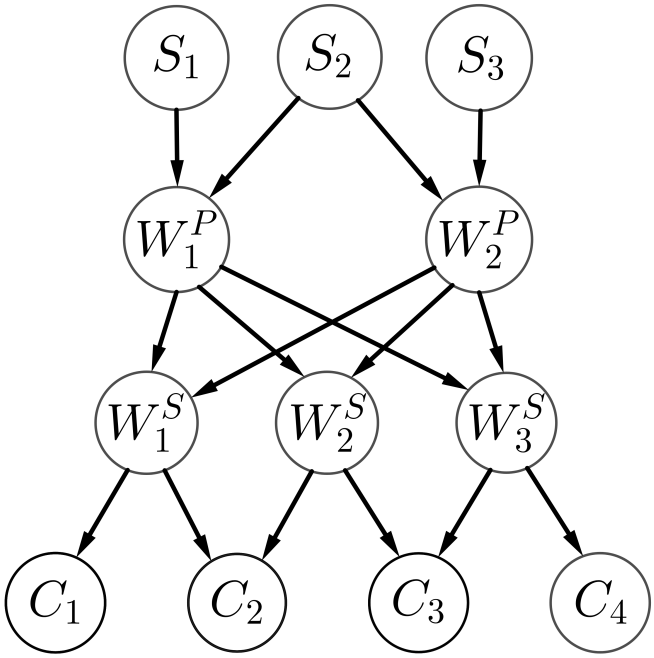

Figure 1: Four-echelon supply chain

The first tier is composed of suppliers that provide the products to be distributed to consumers. Each supplier is characterized by its product line, i.e. the products it can offer. While it is unlikely that a supplier supplies all types of products, it is common that each product can be obtained from multiple suppliers. The second and third tiers of the supply chain are composed of primary and secondary warehouses, respectively. Warehouses can store products, which incurs a per-unit, per-unit-of-time cost. Primary and secondary warehouses essentially differ by their storage cost and storage capacity. Specifically, primary warehouses have higher storage capacity limits

and lower warehousing costs. Besides, primary warehouses are fewer and tend to have more central geographic positions. The fourth tier is composed of customers that request products over a time horizon. Note that customers may request products several times over the horizon and that the types and quantities of products requested may vary from order to order.

To fulfill these requests, products follow *itineraries*, i.e. physical paths from supplier facilities to customer locations, through the distribution network. The possibility to transport a product between a pair of facilities is called a *service* and is defined by times and locations of departure and arrival. To execute a given service, one must allocate vehicle units to that service, with each vehicle providing a capacity of $\hat{u}$. The transport of products incurs a cost that is proportional to the service distance and the amount of freight moved. Note that products from the same customer order may be sourced by different suppliers. At the same time, transportation services may contain products that are part of multiple customer orders. In this supply chain setting, the product distribution follows a "forward flow" structure, since a service can only be defined from a given stakeholder to another stakeholder of a subsequent echelon.

2.2. Application

The supply chain considered consists of: 177 suppliers, 4 primary warehouses, 40 secondary warehouses, and 239 customers/restaurants. Figures 2, 3, 4 and 5 shows the distribution of these stakeholders.

The supply chain involves a thousand standardized products that are classified into five families: frozen products, fresh products, dry products, beverages, and non-food products. These product families are partitioned into two categories of products. Frozen and fresh products belong to the category of temperature-controlled products, while dry products, beverages, and non-food products belong to the category of ambient products. Each category is managed independently as temperature-controlled products require temperature control for food safety while ambient products do not. A classification of the products is provided in Table 2. Note that products are packed and shipped in pallets of homogeneous size that contain a single type of product, and always in the same quantity. Therefore, in the rest of the article, we define a unit of product as a pallet of this product.

Categories	Families	Product number
Temperature-controlled	Frozen	315
	Fresh	290
Ambient	Dry	120
	Beverages	165
	Non-food	110

Table 2: Products distribution

On the supply side, each supplier is specialized in up to three product families and may provide both

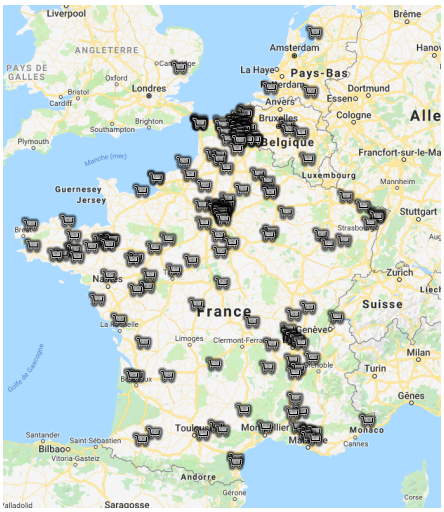

Figure 2: Suppliers

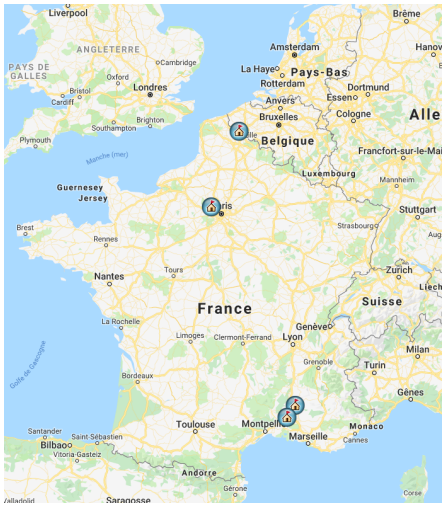

Figure 3: Primary warehouses

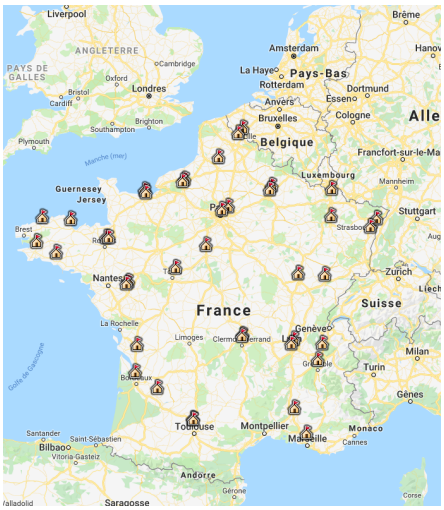

Figure 4: Secondary warehouses

Figure 5: Customers

temperature-controlled and ambient products. A supplier who is not specialized in a given product family cannot provide any product in that family. As an example, a supplier not specialized in non-food products cannot supply napkins. On the other hand, a supplier specialized in a given product family may not supply all products in this family. Thus, a beverage supplier may provide alcoholic beverages but no sodas.

On the demand side, each restaurant has a weekly schedule that indicates time windows for product deliveries. Table 3 shows examples of delivery schedules for a sample of 5 restaurants. Each restaurant places an order for each delivery time window that occurs within the planning horizon. An order is defined as a set of products that must be shipped to the restaurant. Note that the types and quantities of products requested may vary from one delivery to delivery along the time horizon. For example, the restaurant in Amiens may order one pallet of french fries and one pallet of soft drinks for the first Monday of the month, and order one pallet of napkins for the second Monday of the month.

Restaurant location	Mond.	Tues.	Wed.	Thurs.	Fri.	Sat.	Sun.
Amiens	6:30-7:30	-	-	-	-	-	-
Boulogne	-	8:00-9:00	-	10:15-11:15	-	8:30-9:30	-
Montpellier	5:30-6:30	-	-	6:30-7:30	-	-	-
Paris	6:15-7:15	5:30-6:30	-	-	5:30-6:30	7:30-8:30	-
Toulouse	-	7:00-8:00	-	8:15-9:15	-	10:30-11:30	-

Table 3: Example of restaurants delivery schedules

2.3. Design of the transportation plan

The management of the restaurant supply chain is performed by our 3PL partner that develops a transportation plan over a mid-term planning horizon to fulfill customer demands. The design of the transportation plan consists in (1) selecting the source of each product requested (2) determining product itineraries from suppliers to restaurants, (3) prescribing the vehicles required to support those deliveries, and (4) determining product storage times in intermediate terminals.

The 3PL does not manage production decisions. Since suppliers are manufacturers with sufficiently large production capacities and delivery requests are communicated long enough in advance, product stock-outs at suppliers do not occur. In addition, while the 3PL determines the transportation plan, it relies on a third-party carrier for its execution. More specifically, the 3PL communicates needs for point-to-point transportation moves to a carrier and does not make fleet management decisions such as vehicle maintenance timing or repositioning.

2.3.1. Existing distribution strategy

In practice, the design of the transportation plan is performed by a team of experienced logistics engineers that follows a *distribution strategy*. This distribution strategy aims to simplify the planning process by limiting feasible

shipment itineraries to *centralized delivery paths*. Specifically, given an origin supplier for a product requested, a valid centralized delivery path must transit through a primary warehouse, then through the nearest secondary warehouse from the destination, to finally reach its arrival restaurant. We illustrate in Figure 6 a distribution network based on the 3PL distribution strategy, where customers c_1 and c_2 have product requests that are sourced by suppliers s_1 and s_2, respectively. We illustrate in Figure 7 two centralized delivery paths that respect the existing distribution strategy.

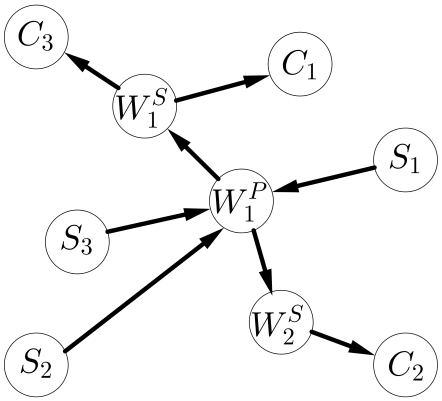

Figure 6: Current distribution strategy

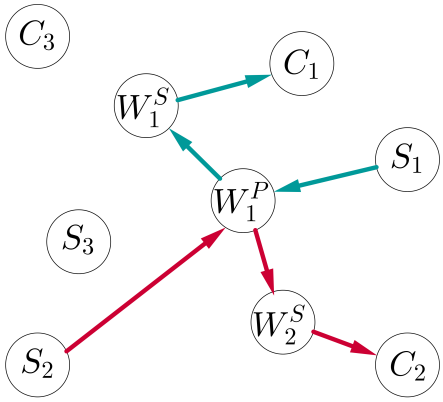

Figure 7: Centralized delivery paths

Consequently, the existing distribution strategy permits to significantly reduce the number of feasible itineraries, allowing supply chain managers to focus on other decisions, such as product shipment origins, operation timings, or warehouse inventory management. This distribution strategy is effective in practice, as centralized delivery paths enable consolidation and reduce transportation costs by increasing vehicle fill rates. However, the exclusive use of centralized delivery paths also has its drawbacks. Indeed, forcing transportation through a primary warehouse can lead to long travel distances for products, especially when shipment points of origin and destination are geographically close. Moreover, centralized delivery paths involve multiple handling at intermediate terminals, causing additional costs and impacting delivery times.

2.3.2. Extended distribution strategies

To reduce the cost of its transportation plans, the 3PL considers extending this distribution strategy and therefore enabling alternative distribution strategies to the centralized delivery paths. The two distribution strategies considered are *indirect delivery paths*, where products transit through a single secondary warehouse between their origin and destination without passing through a primary warehouse, and *direct delivery paths*, where products are shipped directly from a supplier to a restaurant. Extending the distribution strategy is performed by enhancing the distribution network with additional possible services as shown in Figure 8. In Figure 9, c_1's request is fulfilled via a direct delivery path (in blue), while c_2's request is fulfilled via an indirect delivery path (in red).

These alternative distribution strategies reduce the number of intermediate terminals visited, enabling to

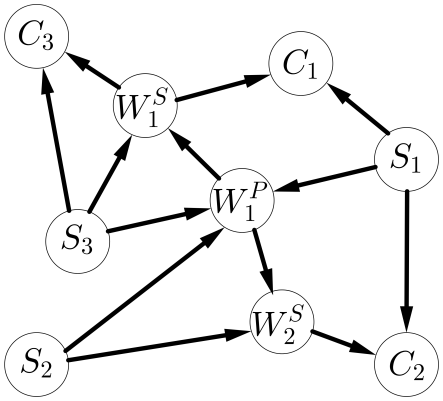

Figure 8: Extended distribution strategy

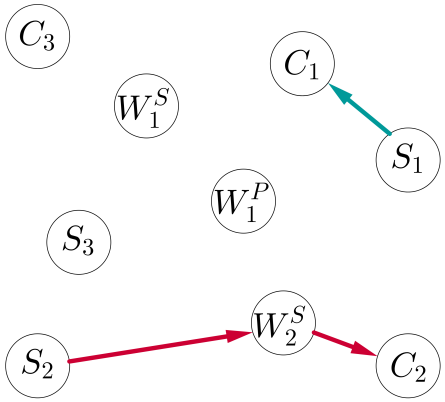

Figure 9: Alternative delivery paths

reduce not only the distances traveled by the products but also the delivery times, and product handling costs. Conversely, by short-cutting primary warehouses and potentially secondary warehouses, direct and indirect delivery paths limit consolidation opportunities and can lead to an increase in the number of services used.

3. Mathematical model

The supply chain is modelled as a directed network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where the set $\mathcal{N}$ represents the stakeholders. A node $n \in \mathcal{N}$ can be either a supplier (set $\mathcal{S}$), a primary warehouse (set $\mathcal{W}^{\mathcal{P}}$), a secondary warehouse (set $\mathcal{W}^{\mathcal{S}}$) or a customer (set $\mathcal{C}$). Each warehouse i is associated with a per-unit per-day cost for holding inventory c_i. The set $\mathcal{A}$ contains arcs that model freight transportation between two facilities. Due to the forward-flow structure of the supply chain, the only transportation arcs defined in $\mathcal{A}$ are: i) from a supplier to a primary warehouse; ii) from a supplier to a secondary warehouse; iii) from a supplier to a customer; iv) from a primary warehouse to a secondary warehouse; v) from a secondary warehouse to a customer. As a result, $\mathcal{A}$ is a subset of $(\mathcal{S} \times \mathcal{W}^{\mathcal{P}}) \cup (\mathcal{S} \times \mathcal{W}^{\mathcal{S}}) \cup (\mathcal{S} \times \mathcal{C}) \cup (\mathcal{W}^{\mathcal{P}} \times \mathcal{W}^{\mathcal{S}}) \cup (\mathcal{W}^{\mathcal{S}} \times \mathcal{C})$. An arc $a = (i, j) \in \mathcal{A}$ is characterized by its travel time $t_{ij} \in \mathbb{N}^*$, its linear cost per unit of flow, $c_{ij} \in \mathbb{R}^{+*}$, a unit of capacity, $\hat{u}$, and its fixed cost per vehicle, $f_{ij} \in \mathbb{R}^{+*}$. The set $\mathcal{P}$ represents all the products. The set of products offered by supplier $i \in \mathcal{S}$ is denoted as $\mathcal{P}^i$.

The 3PL aims to schedule transportation operations for a fixed planning horizon of $\mathcal{D}$ days. To model the temporal aspect of the problem, we use a time granularity, Δ, that indicates the number of time points per day in the complete time-expanded network and defines the time interval duration. For example, $\Delta = 4$ corresponds to 4 time points per day, with each pair of consecutive time points being separated by a time interval of 6 hours. As a result, the planning horizon contains $\mathcal{T}$ time periods, with $\mathcal{T} = \mathcal{D} \times \Delta$.

We extend the static network $\mathcal{G}$ to a time-expanded network $\mathcal{G}_{\mathcal{T}} = (\mathcal{N}_{\mathcal{T}}, \mathcal{H}_{\mathcal{T}} \cup \mathcal{A}_{\mathcal{T}})$, where the set $\mathcal{N}_{\mathcal{T}}$ is obtained by duplicating $|\mathcal{T}|$ times all the nodes from the static network. Thus, each time-expanded node $(i, t) \in \mathcal{N}_{\mathcal{T}}$ represent a stakeholder $i \in \mathcal{N}$ at a given period of time $t \in \mathcal{T}$ and can either model a time-expanded

supplier (set $\mathcal{S}_\mathcal{T}$), a time-expanded primary warehouse (set $\mathcal{W}^\mathcal{P}_\mathcal{T}$), a time-expanded secondary warehouse (set $\mathcal{W}^\mathcal{S}_\mathcal{T}$), or time-expanded customer (set $\mathcal{C}_\mathcal{T}$).

The set of time-expanded arcs in this network is partitioned into holding arcs $\mathcal{H}_\mathcal{T}$, that model the storage of products, and transportation arcs $\mathcal{A}_\mathcal{T}$, that model the transportation of goods between two distinct stakeholders. Holding arcs $((i,t),(i,t+1)) \in \mathcal{H}_\mathcal{T}$ are defined for each warehouse $i \in \mathcal{W}^\mathcal{P} \cup \mathcal{W}^\mathcal{S}$ and each period of time $t \in [1, |\mathcal{T}|-1]$. A per-unit-of-flow cost $\dot{c}_i = \frac{c_i}{\Delta}$ and a storage capacity $wlim_i$ are associated with each holding arc.

To model transportation arcs, we first define $\dot{t}_{ij}$, i.e. the travel time of arc $(i,j) \in \mathcal{A}$ expressed in terms of time intervals. To ensure that static arcs $(i,j) \in \mathcal{A}$ can be mapped to time-expanded arcs, we set $\dot{t}_{ij} = \lceil\ t_{ij}\frac{\Delta}{24}\rceil$. Transportation arcs $((i,t),(j,t+\dot{t}_{ij}))$ are defined for each $(i,j) \in \mathcal{A}$ and each time $t \in \mathcal{T}$ such that $t+\dot{t}_{ij} < |\mathcal{T}|$. Specifically, an arc $((i,t),(j,t++\dot{t}_{ij}))$ models product flows shipped from i at time t and arriving at j at time $t+\dot{t}_{ij}$.

Arcs $((i,t),(j,t+\dot{t}_{ij}))$ may overestimate real transit times t_{ij}, which may have an impact on the overall transportation plan cost. For each static arc $(i,j) \in \mathcal{A}$, we express the difference between its original transit time and its estimate in the time-expanded network as: $approx^\Delta_{ij} = \Delta\dot{t}_{ij} - t_{ij}$. This approximation is taken into account when defining transportation arc linear costs to ensure that correct objective function values are reflected.

We illustrate this with an example. Let consider a time granularity $\Delta = 4$, and a static arc (i,j) with a travel time $t_{ij} = 5h$. Time-expanded copies of (i,j) have a transit time $\dot{t}_{ij} = \lceil\ t_{ij}\frac{\Delta}{24}\rceil = 1$ time interval, i.e. 6 hours, which yields an approximation of $approx^\Delta_{ij} = \Delta\dot{t}_{ij} - t_{ij} = 1$ hour. As a result, the chosen time granularity implies that a product shipped on arc $((i,t),(j,t+\dot{t}_{ij}))$ is kept during 1 hour in facility j before any operation can be proceeded. Thus, to accurately estimate solution costs in the time-expanded network, if j is a warehouse the linear cost of a transportation arc $((i,t),(j,t'))$ is defined as $\dot{c}_{ij} = c_{ij} + c_j\frac{approx^\Delta_{ij}}{24}$, where $\dot{c}_j\frac{approx^\Delta_{ij}}{24}$ reflects the storage cost induced by the time discretization. If j is not a warehouse the linear cost of a transportation arc $((i,t),(j,t'))$ is defined as $\dot{c}_{ij} = c_{ij}$.

Given a time-expanded network $\mathcal{G}_\mathcal{T}$, the Logistics Service Network Design Problem (LSNDP) can be modeled as follows. Let $y^{tt'}_{ij}$ be an integer variable that indicates the number of trucks dispatched on transportation arc $((i,t),(j,t')) \in \mathcal{A}_\mathcal{T}$. Let $x^{ptt'}_{ij}$ be a continuous variable that indicates the quantity of product p that flows along transportation arc $((i,t),(j,t')) \in \mathcal{A}_\mathcal{T}$. Note that if node i models a supplier that does not supply product p (i.e. $p \notin \mathcal{P}^i$) then the variable $x^{ptt'}_{ij}$ is not defined. Let x^{ptt+1}_{ii} be a continuous variable that models the quantity of product p stored at warehouse between t and $t+1$. Given the demands d^p_{ct} that indicate the quantity of product $p \in \mathcal{P}$ required by customer $c \in \mathcal{C}$ at time $t \in [1, |\mathcal{T}|]$, the LSNDP can be stated as:

$$\text{minimize } z(\mathcal{G}_\mathcal{T}) = \sum_{((i,t),(j,t'))\in\mathcal{A}_\mathcal{T}} f_{ij}y^{tt'}_{ij} \quad + \sum_{((i,t),(j,t'))\in\mathcal{A}_\mathcal{T}}\sum_{p\in\mathcal{P}} \dot{c}_{ij}x^{ptt'}_{ij} \quad + \sum_{((i,t),(i,t+1))\in\mathcal{H}_\mathcal{T}}\sum_{p\in\mathcal{P}} \dot{c}_i x^{ptt+1}_{ii} \qquad (1)$$

Under the following constraints :

$$\sum_{((i,t),(j,t'))\in\mathcal{A}_\mathcal{T}\cup\mathcal{H}_\mathcal{T}} x_{ij}^{ptt'} - \sum_{((j,t'),(l,t''))\in\mathcal{A}_\mathcal{T}\cup\mathcal{H}_\mathcal{T}} x_{jl}^{pt't''} = 0, \quad \forall (j,t') \in \mathcal{W}_\mathcal{T}^\mathcal{P} \cup \mathcal{W}_\mathcal{T}^\mathcal{S}, \forall p \in \mathcal{P} \tag{2}$$

$$\sum_{((i,t),(j,t'))\in\mathcal{A}_\mathcal{T}} x_{ij}^{ptt'} \geq d_{jt'}^p, \quad \forall (j,t') \in \mathcal{C}_\mathcal{T}, \forall p \in \mathcal{P} \tag{3}$$

$$\sum_{p\in\mathcal{P}} x_{ij}^{ptt'} \leq \hat{u} y_{ij}^{tt'}, \quad \forall ((i,t),(j,t')) \in \mathcal{A}_\mathcal{T} \tag{4}$$

$$\sum_{p\in\mathcal{P}} x_{ii}^{ptt+1} \leq wlim_i, \quad \forall ((i,t),(i,t+1)) \in \mathcal{H}_\mathcal{T} \tag{5}$$

$$x_{ij}^{ptt'} \in \mathbb{R}^+, \quad \forall ((i,t),(j,t')) \in \mathcal{A}_\mathcal{T} \cup \mathcal{H}_\mathcal{T}, \ \forall p \in \mathcal{P}^i \tag{6}$$

$$y_{ij}^{tt'} \in \mathbb{N}^+, \quad \forall ((i,t),(j,t')) \in \mathcal{A}_\mathcal{T} \tag{7}$$

The objective function (1) minimizes the overall total cost, which is the sum of the fixed costs on transportation arcs (first term), the variable costs on transportation arcs (second term), and the variable costs on holding arcs (third term), i.e. storage costs. Constraints (2) enforce flow conversation at warehouses, i.e. for each product and each warehouse, the stock and incoming flow at a given period must balance with the stock and outgoing flow of the following period. Constraints (3) enforce the fulfillment of all customer demands. Constraints (4) ensure that a sufficient number of vehicles are allocated on each transportation arc. Constraints (5) ensure that warehouse storage capacities are never exceeded. Constraints (6) and (7) define the variable domains.

4. Methodology

To solve the LSNDP, we propose a heuristic approach based on the recently proposed Dynamic Discretization Discovery (DDD) Algorithm [6] for solving Service Network Design Problems. We first describe this method and explain why a straightforward application to the LSNDP is not appropriate. We then present the Time-Expanded Network Reduction Matheuristic (TENMR), which determines a subset of the complete time-expanded network by iteratively solving linear programming relaxations of the LSNDP.

4.1. The Dynamic Discretization Discovery algorithm

Instances of the LSNDP of sizes relevant to the operations of the restaurant chain are challenging, mainly because they are based on extremely large time-expanded networks. As a result, such instances of the LSNDP

are computationally intractable for on-the-shelf optimization solvers. Boland et al. [6] recently addressed the Continuous Time Service Network Design Problem (CTSNDP), a version of the SNDP where time discretization is fine enough to capture every real-life consolidation opportunity, which generally induces extremely large time-expanded networks and intractable models. To overcome this difficulty, the authors propose the DDD algorithm.

The motivation for the algorithm is that while the time-expanded network can be very large, only very small subsets of nodes and arcs appear in the optimal solution. The aim of DDD is to solve the problem without considering the complete time-expanded network. To do so, the first step of the DDD method is to generate a *partially time-expanded network* $\mathcal{X}_\mathcal{T}$ that contains a subset of the nodes and arcs of the complete network $\mathcal{G}_\mathcal{T}$. The network $\mathcal{X}_\mathcal{T}$ may also contain arcs that are not part of $\mathcal{G}_\mathcal{T}$. Indeed, for some arcs $((i,t),(j,t'))$ in $\mathcal{X}_\mathcal{T}$, t' may be strictly lower than $t + t_{ij}$. Such arcs are called *short arcs* as they underestimate travel times. These short arcs cause a SNDP formulated on the partially time-expanded network to overestimate consolidation opportunities. Thus, solving such a SNDP yields a lower bound on the optimal value of the problem formulated on the complete time-expanded network.

Specifically, let SNDP($\mathcal{X}_\mathcal{T}$) be the integer program defined on the partial network and let SNDP($\mathcal{G}_\mathcal{T}$) be the integer program defined on the complete network. The algorithm iterates these three steps: i) solve SNDP($\mathcal{X}_\mathcal{T}$) optimally; ii) determine whether the obtained solution can be converted to a primal solution to SNDP($\mathcal{G}_\mathcal{T}$); iii) extend and repair $\mathcal{X}_\mathcal{T}$. The first step determines a lower bound for the original problem. If this solution is feasible for SNDP($\mathcal{G}_\mathcal{T}$), it is also optimal and the algorithm stops. Otherwise, if this solution is infeasible for SNDP($\mathcal{G}_\mathcal{T}$), it necessarily includes short arcs. In that case, the second step aims at removing these decisions by repairing the used short arcs while maintaining the conditions ensuring that the optimum of SNDP($\mathcal{X}_\mathcal{T}$) is a lower bound of SNDP($\mathcal{G}_\mathcal{T}$).

Boland *et al.* [6] prove that the DDD algorithm converges to an optimal solution in a finite number of iterations. An extensive computational study shows that this approach is more efficient than solving instances of SNDP($\mathcal{G}_\mathcal{T}$). Nevertheless, the DDD approach is not appropriate for solving the LSNDP for multiple reasons. First, it assumes that product storage does not incur costs. Second, the procedure that constructs the initial network is designed for commodities $k \in \mathcal{K}$ that have origins (o_k, t_k^o) and destination (d_k, t_k^d) specified a priori, with t_k^o and t_k^d designating the avail and due time of commodity k, respectively. Specifically, to ensure convergence, the initial network $\mathcal{G}_\mathcal{T}$ must contain all nodes (o_k, t_k^o) and (d_k, t_k^d). Third, the network repair procedure is designed for general distribution networks, as all short arcs are repaired according to the same procedure.

In the LSNDP, we consider storage costs that are positive and may vary by facility. However, as shown in Appendix A, DDD as initially designed may converge to a sub-optimal solution in this case. Even when holding costs are all equal to zero, and thus DDD would still be guaranteed to converge, there would be computational tractability issues regarding the initial network. Specifically, since shipment origins are not specified a priori in the LSNDP, $\mathcal{G}_\mathcal{T}$ should contain a node for each supplier at each time point, which would result in a too-large initial

network and would significantly affect the speed of convergence. Finally, to address the multi-echelon structure of the distribution network, the network repair procedure should vary depending on the arc involved. Specifically, a short arc directed to a customer should not be repaired like a short arc directed to a warehouse.

4.2. Time-expanded network reduction matheuristic

Although a direct application of the DDD algorithm is not appropriate for solving the LSNDP, we retain its main concepts to design the Time-Expanded Network Reduction Matheuristic (TENMR). Because commercial solvers are powerful and often able to solve instances based on reasonably sized networks, the rationale of TENMR is to construct iteratively a sparse time-expanded network $\mathcal{X}_\mathcal{T}$ that is likely to contain a high-quality solution and then solve $IP(\mathcal{X}_\mathcal{T})$.

In this section, we provide an overview of TENMR with an emphasis on the enhancements developed to address the specific features of the LSNDP. The exhaustive description of the algorithm components is deferred to Appendix B. In the remaining, $IP(R)$ denotes the LSNDP defined over the time-expanded network R, while $LP(R)$ denotes the linear relaxation of the same problem. Note that contrary to the DDD algorithm, we only solve linear programs during the iterative process to speed-up the network construction. The flow chart of TENMR is provided in Figure 10.

4.2.1. The initial $\mathcal{X}_\mathcal{T}$: dealing with shipment origins not specified a priori

The DDD algorithm constructs an initial network $\mathcal{X}_\mathcal{T}$ that contains the origin and destination of all shipments. In the case of the LSNDP, this procedure results in a significant network that contains a node for each supplier at each time point, since shipment origins are not specified a priori. To determine an initial network of reasonable size while ensuring that the LSNDP defined on that network is feasible, we initialize $\mathcal{X}_\mathcal{T}$ with all nodes and arcs that appear in the optimal solution of $LP(\mathcal{G}_\mathcal{T})$. Indeed, because product demands are integer, only y-variables can have fractional values in the optimal solution of $LP(\mathcal{G}_\mathcal{T})$. By setting the y-variables to the ceiling of their value, the optimal solution of $LP(\mathcal{G}_\mathcal{T})$ can thus be transformed into a feasible solution of $IP(\mathcal{G}_\mathcal{T})$, such that initializing $\mathcal{X}_\mathcal{T}$ with the nodes and arcs that appear in the optimal solution of $LP(\mathcal{G}_\mathcal{T})$ ensures that at least one solution of $IP(\mathcal{X}_\mathcal{T})$ is feasible for $IP(\mathcal{G}_\mathcal{T})$. Because none of these initial nodes and arcs are modified during the iterative construction process, solving the last program $IP(\mathcal{G}_\mathcal{T})$ necessarily yields a feasible solution for the original problem.

Then, similarly to [6], we add short arcs in $\mathcal{X}_\mathcal{T}$, i.e. transportation arcs that underestimate real transit times. However, contrarily to [6] we generate short arcs $((i,t),(j,t'))$ with linear and fixed costs inferior to that of their original copy (i,j). This choice is motivated by the fact that we solve linear programs rather than integer programs during the construction process. Indeed, short arcs aim to allow consolidations that are infeasible in real life, which permits to compute cheaper solutions that use fewer vehicles than necessary. However, consolidation savings are not reflected in the objective function if the integrality of the vehicle variables is relaxed, such that when solving

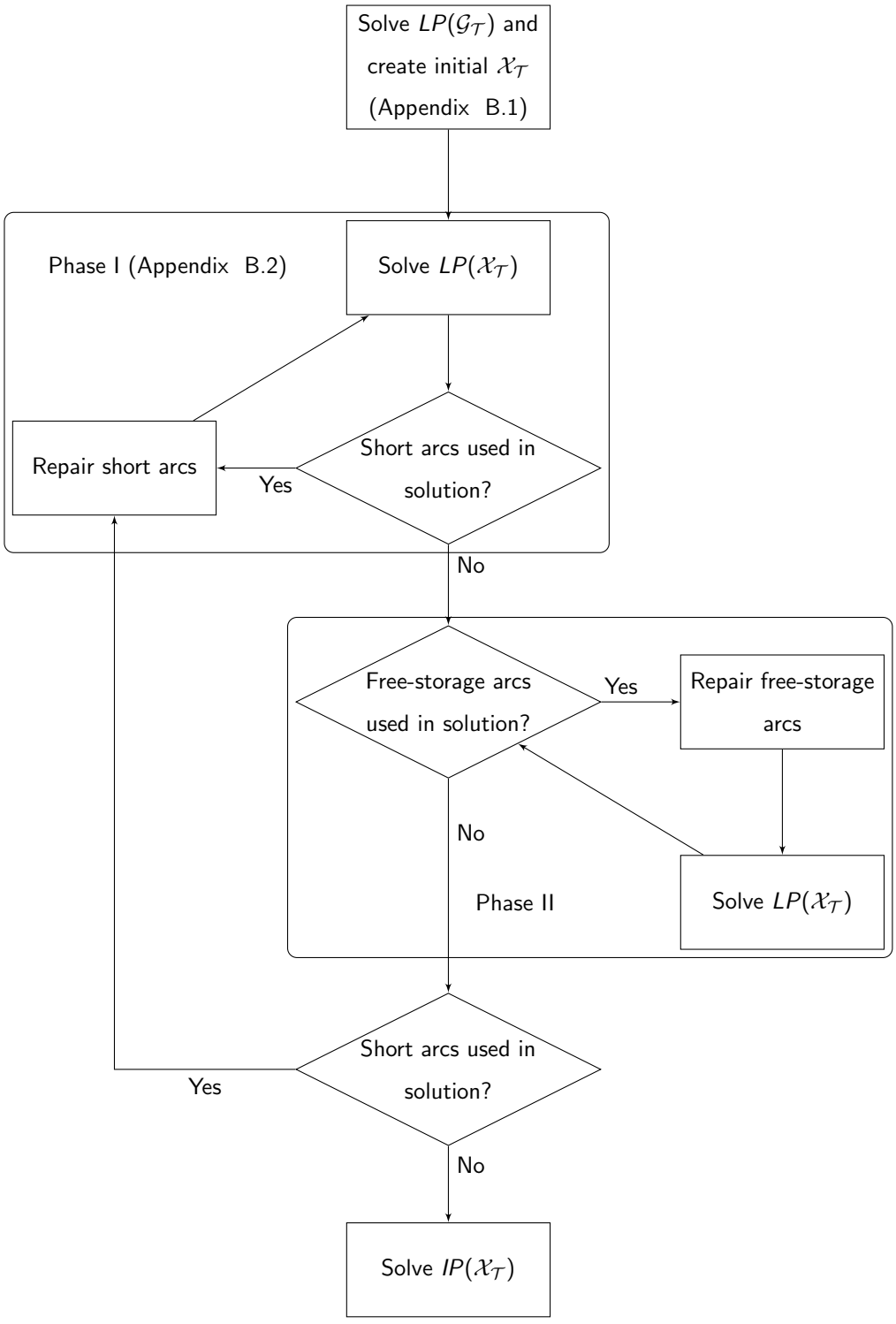

Figure 10: Flow Chart for TENMR

$LP(\mathcal{X}_\mathcal{T})$, utilizing short arcs with costs similar to that of their original copy (i, j) presents no particular advantage. To make short arcs more attractive than correct arcs, we underestimate their costs proportionally to their transit time. Finally, similarly to [6], the cost of each storage arc is set to 0. In the remaining, storage arcs with null costs are called *free-storage arcs* .

After constructing the initial network, we solve $LP(\mathcal{X}_\mathcal{T})$ and check whether or not the obtained solution leverages short arcs or free-storage arcs. If it does, two phases of repair mechanisms are performed, both of which operate on solutions to $LP(\mathcal{X}_\mathcal{T})$. The first focuses on the lengthening of the short arcs used in such a solution, as it is the case in DDD, but in a different way that takes account of the multi-echelon structure of the distribution network. The second focuses on correcting the costs of the free storage arcs used in such a solution.

4.2.2. Phase I: dealing with the multi-echelon network structure

The first phase aims at repairing short arcs. Specifically, as long as short arcs are used in the optimal solution of $LP(\mathcal{X}_\mathcal{T})$, $\mathcal{X}_\mathcal{T}$ is repaired and $LP(\mathcal{X}_\mathcal{T})$ is solved again. The procedure is similar to the DDD algorithm in the sense that it aims to replace short arcs with timed arcs that have correct transit times, but it differs from the DDD algorithm in the way short arcs are corrected. In the DDD algorithm, the correction of a short arc consists of lengthening it "forwardly". More specifically, the lengthening of a short arc $((i, t), (j, t'))$ consists in (1) creating the occurrence (j, t_{new}) such that $t_{new} = t + t_{ij}$, and (2) replacing $((i, t), (j, t'))$ with $((i, t), (j, t_{new}))$. In our procedure, short arcs $((i, t), (j, t'))$ such that j is not a customer are lengthened "forwardly". On the other hand, short arcs $((i, t), (j, t'))$ such that j is a customer are lengthened "backwardly", which consists in (1) creating the occurrence (i, t_{new}) such that $t_{new} = t' - t_{ij}$, and (2) replacing $((i, t), (j, t'))$ with $((i, t_{new}), (j, t'))$. This specific refinement is motivated by the multi-echelon network structure and the fact that $\mathcal{X}_\mathcal{T}$ contains all customers with positive demands. It prevents us to enhance $\mathcal{X}_\mathcal{T}$ with customers with null demand, which would not enable us to discover new potential solutions.

Figure 11 illustrates an example of these refinement strategies. On the left side is represented the static arc (i, j) with transit time $t_{ij} = 1$. On the right side is represented a short timed copy of (i, j) with a null transit time (in black) as well as corrected versions of $((i, t)(j, t))$ obtained by "forward" refinement (in blue) and "backward" refinement (in green).

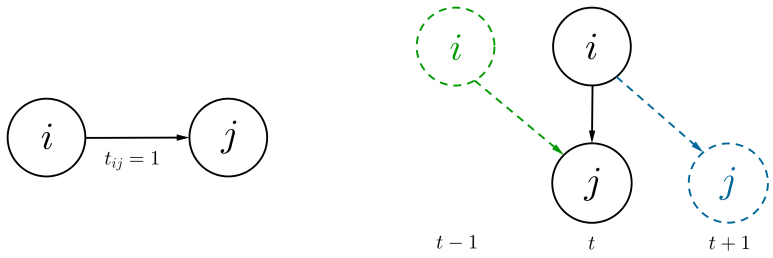

Figure 11: Backward and forward refinements of a short arc

4.2.3. Phase II: dealing with storage costs

The second phase, reported in Algorithm 1, aims at repairing free-storage arcs. Similarly to the first phase, it is iterative and takes the current optimal solution of $LP(\mathcal{X}_\mathcal{T})$ as an input parameter. At each iteration, each free-storage arc with a non-zero flow in the optimal solution of $LP(\mathcal{X}_\mathcal{T})$ is detected, its real cost is updated in $\mathcal{X}_\mathcal{T}$, and $LP(\mathcal{X}_\mathcal{T})$ is solved again. Note that, as the storage cost of an arc $((i,t),(i,t+1))$ is $\dot{c}_i$, and as storage arcs of $\mathcal{X}_\mathcal{T}$ may have a duration longer than one unit, we set the cost of $((i,t),(i,t'))$ to $c_i(t'-t)$. The procedure stops when no free-storage arc appears in the optimal solution of $LP(\mathcal{X}_\mathcal{T})$.

Algorithm 1 Repair free-storage arcs

Require: Optimal solution of $LP(\mathcal{X}_\mathcal{T})$
  while Optimal solution of $LP(\mathcal{X}_\mathcal{T})$ uses free-storage arcs **do**
    for free-storage arcs $((i,t),(i,t')) \in \mathcal{X}_\mathcal{T}$ with positive flow **do**
      Set the cost of $((i,t),(i,t'))$ to $\dot{c}_i(t'-t)$;
    end for
    Solve $LP(\mathcal{X}_\mathcal{T})$
  end while

4.2.4. Stopping criterion and final solution

The refining process of $\mathcal{X}_\mathcal{T}$ terminates when the solution obtained at the end of the second phase does not use short arcs or free-storage arcs. This occurs after a finite number of iterations since the number of arcs to be lengthened in $\mathcal{X}_\mathcal{T}$ is bounded by the number of transportation arcs in $\mathcal{G}_\mathcal{T}$, and the number of free storage arcs is bounded by the number of storage arcs in $\mathcal{G}_\mathcal{T}$.

As the optimal solution of $LP(\mathcal{X}_\mathcal{T})$ is feasible for $LP(\mathcal{G}_\mathcal{T})$, and as any solution of $LP(R)$ can be transformed into a feasible solution of $IP(R)$ by rounding-up the fractional values, there are solutions of $IP(\mathcal{X}_\mathcal{T})$ that are feasible for $IP(\mathcal{G}_\mathcal{T})$. However, $\mathcal{X}_\mathcal{T}$ may still contain short arcs at the end of the iterative process. Thus, the optimal solution of $IP(\mathcal{G}_\mathcal{T})$ may not be feasible for the original program. To avoid this, we remove all remaining short arcs (transportation arcs underestimating real transit times) of $\mathcal{X}_\mathcal{T}$. Besides, we update the cost of each remaining free-storage arc to reflect correct objective function values. Finally, we solve $IP(\mathcal{X}_\mathcal{T})$ and obtain a solution of $IP(\mathcal{G}_\mathcal{T})$.

5. Computational study

In section 5.1, we first describe the instances based on data provided by the 3PL. Then, the efficiency of the heuristic is evaluated in 5.2. In 5.3, we introduce a solution algorithm that combines TENMR with the Meta Partial Benders Decomposition solution method proposed by Belieres et al. [5]. Finally, we perform a qualitative

analysis of the solutions produced by this approach and we discuss the impact of the new distribution strategies in 5.4.

5.1. Instances

First, we describe the characteristics of the supply chain operated by our industrial partner, a third-party logistics provider. Then, we describe how instances are randomly generated based on those characteristics.

5.1.1. Data

The 3PL current logistics network involves 460 stakeholders. They are distributed in four areas of France according to the distribution reported in Table 4.

Area	Suppliers	Prim. Warehouses	Sec. Warehouses	Restaurants
North-West	36	1	16	51
North-East	91	1	9	87
South-West	18	1	7	44
South-East	32	1	8	57
Total	177	4	40	239

Table 4: Distribution of the stakeholders

The 3PL provides us with the product families that each supplier specializes in, but it does not indicate the detailed product line of each supplier. It also provides us with demand patterns for the products of the different families and the percentages of products concerned for each pattern. These patterns take into account the seasonality of the demand, which may vary from a product to another. For example, the demand for ice cream tends to be high in summer and low during the rest of the year, while the demand for potatoes is high in winter and medium during the rest of the year. These data are presented in Table 5.

Transportation fixed costs are calculated based on data from the National Road Committee (CNR) [1], which states that the average transportation cost for a carrier is 0.519 euros per kilometer. The transportation linear cost for loading/unloading a pallet of product into a vehicle is 0.4 euros. The 3PL provides us with the per-unit per-day inventory cost and the storage capacity of each warehouse. Specifically, the average holding cost of primary warehouses is 0.65 euros per pallet and per day, against 1.5 euros for secondary warehouses. The maximum storage capacities of primary and secondary warehouses are 10,000 and 5,000 pallets, respectively. Data relative to the warehouses are reported in Table 6.

5.1.2. Instance generation

Based on this data, a static representation of the logistic network is built as follows. First, nodes $n \in \mathcal{N}$ that models the different stakeholders, are generated accordingly to the real physical locations. Then, we generate a

			Demand level					
			Summer			Winter		
Families	Number of products	% of products	Low	Medium	High	Low	Medium	High
Frozen	315	80			x			x
		20			x	x		
Fresh	290	50			x			x
		30		x			x	
		20			x	x		
Dry	120	50			x			x
		50		x			x	
Beverages	165	50		x			x	
		50			x		x	
Non-Food	110	80	x			x		
		20		x			x	

Table 5: Demand level per product family

	Average inventory cost	Maximum storage capacity
Prim. warehouses	0.65 euros	10,000 pallets
Sec. warehouses	1.5 euros	5,000 pallets

Table 6: Warehouse storage costs and capacities

set of transportation arcs, $\mathcal{A}$, that enable the centralized delivery paths performed by the 3PL. More specifically, a transportation arc is added from each supplier to each primary warehouse, from each primary warehouse to each secondary warehouse, and towards each restaurant from its nearest secondary warehouse. Extra transportation arcs are added using a connectivity radius α. A transportation arc is built from each supplier to any secondary warehouse or any restaurant located in a radius of α kilometers, and from each secondary warehouse to any restaurant located in a radius of α kilometers. As a result, instances with $\alpha = 0$ kilometers only allow *centralized delivery paths* and model the current distribution strategy. Travel times in the static network, t_{ij}, are expressed in hours and obtained using Google Maps, considering light traffic.

Each supplier product line is randomly generated, based on the product families it provides. Specifically, a supplier that specializes in a product family has a 10% chance to provide a product in that family. Finally, we generate restaurant demands. For each restaurant, the product deliveries are requested at times that match the delivery schedule provided by the 3PL. The volume of each product demand is randomly chosen according to its seasonal demand and the season considered. Vehicle capacities are set to 60 pallets. In Table 7, we summarize which components of the instances are based on real data and which components are randomly generated.

Nodes	Arcs	Costs	Supplier product families	Supplier product lines	Demand schedules	Demand volumes
Real data	Real data + extra arcs for the alternative distribution strategies	Real data	Real data	Random data following suppliers product families	Real data	Random data following product seasonalities

Table 7: Instance components

5.1.3. Regional instances

We focus on planning the transportation operations within a single regional area, for two main reasons. The first reason is that, in practice, restaurant product requests are sourced by suppliers from the same regional area (if it is possible), which enables to limit the distances traveled. Thus, we build instances based on the south-western part of the logistics network. The category of ambient products is considered here. The second reason is the computational tractability. Indeed, an instance for the family of temperature-controlled products based on the whole network during the summer season considering the 3PL distribution strategy, a time horizon of 15 days, and time discretization of 6 hours leads to integer programs with approximately 17,000,000 variables and 2,000,000 constraints. To the best of our knowledge, this is substantially larger than any SNDP/LSNDP instance solved in the literature, either exactly or heuristically.

Two sets of instances are generated: a set of *easy* instances and a set of *difficult* instances that differ in the order of magnitude of the parameter values. An *easy* instance is obtained by randomly selecting 20 stakeholders (6 suppliers, one primary warehouse, 3 secondary warehouses, 10 customers) from the south-western part of the logistics network. Consequently, the products involved in such instances vary according to the set of selected

suppliers. We generate 90 *easy* instances based on: 5 subsets of the south-western network, a planning horizon $D = 14$ days, $\Delta \in \{1, 2, 3\}$, $\alpha \in \{0, 25, 50\}$ and 2 demand seasonalities (summer and winter). *Difficult* instances are based on the whole south-western logistics network, apart from 3 suppliers that are specialized in temperature-controlled products only (recalling that temperature-controlled products and ambient products are managed independently). We generate 24 *difficult* instances based on: a planning horizon $D = 14$ days, $\Delta \in \{2, 4, 6\}$, $\alpha \in \{0, 20, 40, 60\}$ and 2 demand seasonalities (summer and winter).

Details of the instances are reported in Appendix C. Tables C.22 to C.26 provide parameter values for each network configuration considered in the *easy* instances. Table C.27 provides parameter values for the *difficult* instances.

5.2. Efficiency of the heuristic

The efficiency of the heuristic is evaluated by solving the instances with two methods. The first method, **CPLEX**, consists in solving the LSNDP defined over the complete time-expanded network with the CPLEX branch-and-cut algorithm. The second method, **TENMR+CPLEX**, performs the network reduction heuristic and solves the resulting program with CPLEX branch-and-cut algorithm. All algorithms are executed on an Intel Xeon E5-2695 processor with 16 GB of memory under Linux 16.04, with stopping criteria of a proven optimality gap of 1% or less and a maximum run-time of 5 hours. Linear and integer programs were solved using Cplex 12.7. Note that we initiate both methods with the same heuristic solution (x_h, y_h), obtained by setting each vehicle variable $y_{ij}^{tt'}$ to the ceiling of its value in the optimal solution of the linear relaxation of the LSNDP.

We investigate the ability of our heuristic to generate sparse time-expanded networks that can lead to good quality solutions. To do so, for each instance we measure a primal gap between the solutions computed by **CPLEX** and **TENMR+CPLEX**:

$$\text{primal-gap} = \frac{z_{TENMR+CPLEX} - z_{CPLEX}}{z_{CPLEX}} \times 100$$

where $z_{TENMR+CPLEX}$ and z_{CPLEX} denote the objective function values of the solutions returned by **TENMR+CPLEX** and **CPLEX**, respectively. A positive ratio indicates that the solution computed by **CPLEX** is better than that computed by **TENMR+CPLEX**, which is always the case when the solver succeeds in solving the complete program. A negative ratio indicates that **TENMR+CPLEX** outperforms **CPLEX**, which can only occur when the solver does not manage to solve optimally the complete mixed integer program.

We first discuss the results obtained on the set of *easy* instances, that were all solved to optimality by **CPLEX**. Figure 12 presents the distribution of the primal gaps. The primal gap is below 0.5% for more than 75% of the instances, and it never exceeds 4%. For 26 out of 90 instances, **TENMR+CPLEX** identified a solution that is optimal for the original model. These results suggest that solving the reduced formulation instead of the complete mathematical program has little impact on the quality of the solutions computed.

In Tables 8 and 9, we report the optimality gaps at termination and the computational times obtained by **CPLEX** and **TENMR+CPLEX**, as well as the primal gaps. Note that since **TENMR+CPLEX** may compute

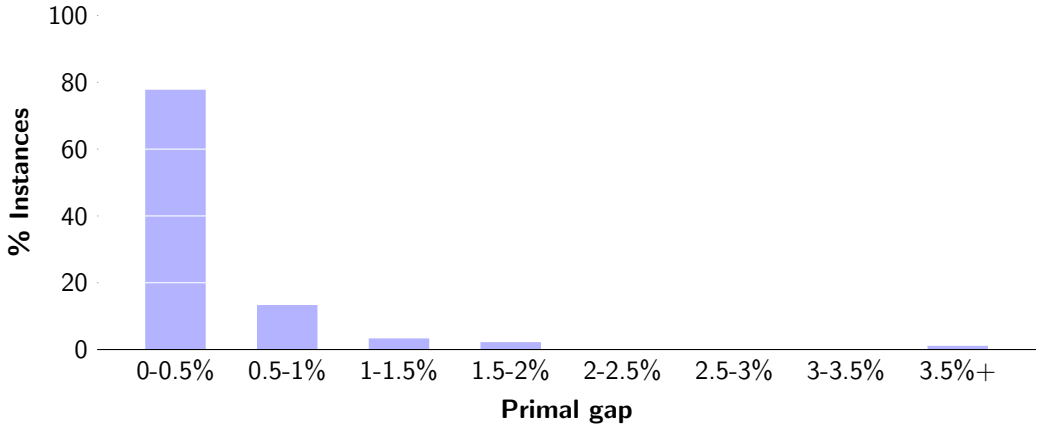

Figure 12: Primal gap distribution for the easy instances

	CPLEX		TENMR+CPLEX		
Δ	Gap	Time	Gap	Time	Primal gap
1	0.94%	6.7	1.00%	4.3	0.11%
2	0.96%	86.6	1.38%	22.2	0.49%
3	0.99%	801.0	1.42%	63.3	0.45%

Table 8: Performance for different Δ (easy instances)

	CPLEX		TENMR+CPLEX		
α	Gap	Time	Gap	Time	Primal gap
0	0.96%	280.5	1.16%	13.7	0.26%
25	0.90%	231.2	1.33%	34.8	0.42%
50	0.98%	382.5	1.32%	41.2	0.38%

Table 9: Performance for different α (easy instances)

invalid lower bounds, optimality gaps reported for **TENMR+CPLEX** take the lower bounds produced by **CPLEX** as a reference. In Table 8, values are averaged over instances with the same value for parameter Δ. In Table 9, values are averaged over instances with the same value for parameter α.

The choice of the time granularity has a substantial impact on the instance difficulty, as the computational time required by **CPLEX** raises significantly as Δ increases. Indeed, changing Δ from 1 to 3 increases the computational time by a factor of 120. The choice of the time granularity has a similar effect on **TENMR+CPLEX** but to a much smaller extent. When the heuristic is used, the computational time increases by a factor of less than 15. Overall, **TENMR+CPLEX** manages to produce solutions with comparable quality to those computed by **CPLEX**, as the primal gap does not exceed 0.5% regardless of the value of Δ. Moreover, these solutions are obtained with significant reductions in computational time. The same results can be inferred from Table 9. For the *easy* instances, we observe that α has less impact on the instance difficulty than Δ. This can be explained by the fact that, for these instances, increasing α does not yield significantly more transportation arcs in the static network, as observed in Tables C.22 to C.26.

We now compare the performance of **TENMR+CPLEX** and **CPLEX** on the *difficult* instances. To do so, we report the same performance indicators in Tables 10 and 11. Note that neither **CPLEX** nor **TENMR+CPLEX** converged within the time limit for any of the *difficult* instances. Except for $\Delta = 2$, **CPLEX** yields very weak optimality gaps at termination regardless of the parameter chosen and its value. **TENMR+CPLEX** provides

	CPLEX		**TENMR+CPLEX**		
Δ	Gap	Time	Gap	Time	Primal gap
2	5.50%	18,000	4.22%	18,000	-1.33%
4	44.15%	18,000	7.14%	18,000	-39.81%
6	44.05%	18,000	9.28%	18,000	-38.40%

Table 10: Performance for different Δ (*difficult* instances)

	CPLEX		**TENMR+CPLEX**		
α	Gap	Time	Gap	Time	Primal gap
0	26.01%	18,000	6.14%	18,000	-21.34%
20	30.90%	18,000	7.05%	18,000	-25.90%
40	35.30%	18,000	6.85%	18,000	-30.90%
60	32.72%	18,000	7.47%	18,000	-27.91%

Table 11: Performance for different α (*difficult* instances)

better results as it produces solutions of significantly higher quality that yield an overall primal gap of -26.5%. Thus, we conclude that solving the simplified model rather than the complete model is preferable when large-scale industrial instances are involved. This result holds as we reduce the time limit imposed on **TENMR+CPLEX** to 30 minutes. Overall, the solutions obtained by **TENMR+CPLEX** after 30 minutes of computation are 9.1% better on average than those obtained by **CPLEX** after 5 hours of computation. These results are reported in Tables 12 and 13.

	CPLEX		**TENMR+CPLEX** (30 min)		
Δ	Gap	Time	Gap	Time	Primal gap
2	5.50%	18,000	9.20%	1,800	4.73%
4	44.15%	18,000	32.61%	1,800	-12.12%
6	44.05%	18,000	30.42%	1,800	-19.96%

Table 12: Performance for different Δ (*difficult* instances)

	CPLEX		**TENMR+CPLEX** (30 min)		
α	Gap	Time	Gap	Time	Primal gap
0	26.01%	18,000	23.9%	1,800	-1.27%
20	30.90%	18,000	26.7%	1,800	-5.08%
40	35.30%	18,000	21.8%	1,800	-17.83%
60	32.72%	18,000	24.0%	1,800	-12.30%

Table 13: Performance for different α (*difficult* instances)

To better understand the results provided by **TENMR+CPLEX**, we study several performance indicators relative to our heuristic: the number of linear programs solved in each phase, the computational time spent in each phase, the computational time spent solving the final integer program LSNDP($\mathcal{X}_\mathcal{T}$), the percentage of variables of LSNDP($\mathcal{X}_\mathcal{T}$) compared to the full model LSNDP($\mathcal{G}_\mathcal{T}$), the number of time-expanded arcs created by the iterative refinement scheme and the percentage of these arcs that are used in the final solution. Recall that phases 1 and 2 correspond to the elimination of short arcs and the update of storage costs, respectively. In Tables 14 and 15 we report the indicators stated above averaged over instances with the same value for parameters Δ and α, respectively.

Δ	P1 LP solved	P1 time	P2 LP solved	P2 time	LSNDP($\mathcal{X}_\mathcal{T}$) time	% of variables	Arcs created	% of arcs created in sol.
2	15.3	33.5	5.9	11.4	16,957	32.5%	559.6	9.6%
4	30.0	94.0	5.1	13.1	17,855	24.3%	1516.6	22.2%
6	40.9	219.4	5.3	20.7	16,766	21.5%	2829.1	31.1%

Table 14: Heuristic behaviour for different Δ (*difficult* instances)

α	P1 LP solved	P1 time	P2 LP solved	P2 time	LSNDP($\mathcal{X}_\mathcal{T}$) time	% of variables	Arcs created	% of arcs created in sol.
0	17.7	53.4	5.7	13.1	17,895	25.9%	1221.7	19.4%
20	31.0	152.5	5.7	15.8	16,535	26.2%	1583.7	19.8%
40	32.7	129.1	5.2	16.7	16,527	26.3%	1829.7	22.9%
60	33.5	127.5	5.2	14.6	17,812	26.1%	1905.5	21.7%

Table 15: Heuristic behaviour for different α (*difficult* instances)

Regarding Table 14, we observe that the number of linear programs solved and the time spent during the first phase increase with Δ. As a result, the number of arcs created in the iterative refinement scheme increases with Δ as well. Nevertheless, the percentage of variables from the original program that appears in LSNDP($\mathcal{X}_\mathcal{T}$) decreases as the time granularity raises. This result indicates that the time-expanded network produced by the heuristic tends to be sparser as the time discretization is finer. This result is not surprising as it is computationally demonstrated by Boland et al. in the context of solving general the SNDP, and justifies the potential of our heuristic to determine high-quality transportation plans on an industrial scale. Also, we observe that the larger is Δ, the more arcs created by the iterative refinement scheme appear in the final solution. The same comments can be made on the results presented in Table 15, except that the relative size of LSNDP($\mathcal{X}_\mathcal{T}$) compared with LSNDP($\mathcal{G}_\mathcal{T}$) does not vary clearly with α.

Overall, around 6 iterations are performed during the second phase, regardless of the parameter considered and its value. We note that the time spent constructing the sparse time-expanded network is of approximately 4 minutes in the worst case, leaving almost the entire computation time for the solution of the reduced program. This is the main advantage of dealing with linear programs in the heuristic.

5.3. A hybrid matheuristic

Although the reduced program is considerably more computationally tractable than the complete program, it cannot be solved to optimality by the generic industrial solver when we consider the *difficult* instances. To improve the quality of the solutions computed, we combine our heuristic with the Meta Partial Benders Decomposition (Meta-PBD) solution method proposed by Belieres et al., which is the current state-of-the-art exact method for the LSNDP. Meta-PBD relies on partial Benders decompositions [10, 11], wherein information derived from aggregating subproblem data is used to reinforce the Benders master problem. In the context of solving the LSNDP, the Benders master problem is strengthened with variables and constraints that model the need to route one or more "super-products" that are aggregations of a subset of products. To improve the convergence speed of the Benders algorithm, Meta-PBD intelligently switches from one master problem to another by changing both the number of super-products to include in the master as well as how these super-products are built. We refer the interested reader to [5] for more details on Meta-PBD.

The new approach, **TENMR+Meta-PBD**, performs the network reduction heuristic and solves the resulting

program with Meta-PBD. To motivate this choice, we compared the performance of **TENMR+CPLEX** and **TENMR+Meta-PBD**. Both methods were executed on the *difficult* instances, with a maximum run-time of 24 hours. Overall, **TENMR+Meta-PBD** provided an optimality gap at termination of 4.17%, against 4.77% for **TENMR+CPLEX**. In addition, **TENMR+Meta-PBD** computed a better upper bound than **TENMR+CPLEX** for 83.3% of the instances. As a result, the transportation plans studied in the qualitative analysis are those produced by **TENMR+Meta-PBD**.

5.4. Qualitative analysis

In this subsection, we study the impact that more flexible distribution strategies would have on the overall transportation plan cost. We also investigate how these alternative distribution strategies would affect the operations performed by the 3PL. The transportation plans studied in this subsection correspond to the *difficult* instances. They are produced by **TENMR+Meta-PBD** after a maximum run-time of 24 hours and given a 48 GB memory-limit. For each instance, we report in Table 16 the lower bound, the upper bound, the optimality gap, and the computational time returned at termination.

α	Δ	Season	LB	UB	Gap	Time	α	Δ	Season	LB	UB	Gap	Time
0	2	S	112,554	114,329	1.55%	86,400	40	2	S	102,407	105,878	3.28%	86,400
0	2	W	95,261	96,992	1.78%	86,400	40	2	W	87,116	90,117	3.33%	86,400
0	4	S	108,450	111,119	2.40%	86,400	40	4	S	99,050	103,455	4.26%	86,400
0	4	W	91,923	94,383	2.61%	86,400	40	4	W	84,090	88,669	5.17%	86,400
0	6	S	107,607	111,899	3.84%	86,400	40	6	S	98,806	104,161	5.14%	86,400
0	6	W	91,390	95,662	4.47%	86,400	40	6	W	83,959	91,087	7.83%	86,400
20	2	S	105,208	108,228	2.79%	86,400	60	2	S	101,895	105,243	3.18%	86,400
20	2	W	89,156	91,966	3.05%	86,400	60	2	W	86,281	89,801	3.92%	86,400
20	4	S	101,799	105,234	3.26%	86,400	60	4	S	98,420	102,550	4.03%	86,400
20	4	W	86,150	89,842	4.11%	86,400	60	4	W	83,595	88,027	5.03%	86,400
20	6	S	100,961	106,897	5.55%	86,400	60	6	S	98,029	103,845	5.60%	86,400
20	6	W	86,087	91,276	5.69%	86,400	60	6	W	83,403	90,980	8.33%	86,400

Table 16: **TENMR+Meta-PBD** results for the *difficult* instances

We note that more than 24 hours are required to solve the reduced program, as none of the instances are solved optimally within the time limit. However, the average optimality gap is of 4.17%, which indicates that overall, we obtained provably high-quality solutions for the reduced program. Since we compare suboptimal solutions, we also note that increasing the value of Δ or α does not necessarily yield a cheaper transportation plan. We now get insights into how extending the distribution strategy enables to reduce the overall transportation plan cost. We report in Table 17 the average savings achieved on the overall transportation plan cost by increasing the parameter value of α. The baseline is the transportation overall cost obtained with $\alpha = 0$.

α	0	20	40	60
Cost savings	0%	5.0%	6.6%	7.0%

Table 17: Relative savings on the overall transporation plan cost

Overall, increasing α yields cheaper transportation plans. This is not surprising since extending the distribution strategy permits to increase the number of delivery opportunities and thus enlarge the solution search space. When considering a connectivity radius of 60 kilometers, the transportation plan average cost has a value of 96,741 euros, against 104,064 euros when using the current distribution strategy, thus generating savings of 7%. Since these results are obtained on instances modelling a quarter of the logistics network and covering a 14-day time horizon, one can expect substantial annual savings for the entire supply chain. We further investigate the reasons for such savings on the transportation plan overall cost. We report in Figure 13 the percentages of products delivered via centralized, indirect, and direct delivery paths for all distribution strategies.

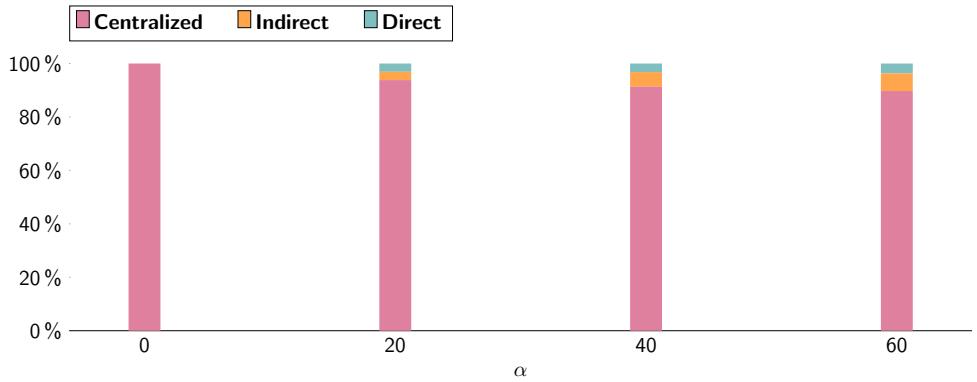

Figure 13: Percentages of centralized, indirect and direct delivery paths

As expected, the initial rate of 100% of centralized delivery paths decreases gradually as we extend the distribution strategy and allow extra shipment opportunities. Although extra shipment opportunities are successfully leveraged as α increases, nearly 90% of shipments still travel through the central warehouse when $\alpha = 60$. On the other hand, 6.6% of the products are transported through indirect delivery paths and the remaining 3.7% are shipped directly when $\alpha = 60$. This result suggests that the transportation plans obtained with the extended distribution strategies do not represent a significant change from the plans the 3PL currently executes. This means that the 3PL could experience savings whithout modifying its current its current network managment in depth. This means that the 3PL could experience savings without making major changes to its current network management, which is interesting from a practical point of view. We next investigate why allowing indirect and direct delivery paths leads to decrease overall costs. In Table 18, we report the average vehicle fill rate, the total number of trucks required, and the average distance traveled per truck, for all distribution strategies.

Integrating alternative shipment deliveries results in a reduction of the average vehicle fill rate, and thus in

α	0	20	40	60
Fill rate	77.3%	64.5%	66.3%	64.3%
Vehicles used	488	536	544	548
Distance travelled (km)	316	270	260	256

Table 18: Fill rate, number of vehicles used and distance travelled per truck

an increase in the number of vehicles required. This is a direct consequence of the decreasing use of centralized delivery paths that are particularly appropriate for product consolidations. On the other hand, we observe that the use of alternative shipment deliveries makes it possible to reduce the average distance traveled per truck, and thus to reducte of the distances traveled by the products. In Table 19, we report the average fill rate for vehicles dispatched from suppliers to primary warehouses, from suppliers to secondary warehouses, and from suppliers to customers, for all distribution strategies. In Table 20, we report the number of services used in the obtained transportation plan. Note that a service from *i* to *j* is counted once regardless if it is used once or multiple times.

α	0	20	40	60
$\mathcal{S} \to \mathcal{W}^{\mathcal{P}}$	94.3%	94.8%	93.8%	94.0%
$\mathcal{S} \to \mathcal{W}^{\mathcal{S}}$	-	18.2%	21.8%	21.2%
$\mathcal{S} \to \mathcal{C}$	-	8.0%	8.0%	8.8%

Table 19: Vehicle fill rates

α	0	20	40	60
$\mathcal{S} \to \mathcal{W}^{\mathcal{P}}$	15.0	15.0	15.0	15.0
$\mathcal{S} \to \mathcal{W}^{\mathcal{S}}$	0.0	2.0	3.0	5.0
$\mathcal{S} \to \mathcal{C}$	0.0	14.2	15.5	15.5

Table 20: Number of services used

The average fill rate for vehicles dispatched from suppliers to primary warehouses is higher than 90%, which indicates that shipments from suppliers to primary warehouses occur when truck capacities can be fully utilized. On the contrary, shipments from suppliers to secondary warehouses are of significantly smaller size as they do not occupy more than 25% of the truck capacity. Direct shipments are even smaller and occupy around 10% of the truck capacity. Unsurprisingly, allowing direct and indirect delivery paths changes the network design and leads to a diversification of the services used, as twice as many more services are used for $\alpha = 20$ than $\alpha = 0$. However, the number of services used does not increase linearly with α, as it is roughly the same for $\alpha = 20$, $\alpha = 40$, and $\alpha = 60$. In Table 21, we report a distribution of the savings achieved for each distribution strategy, according to the different elements of the objective function: **Vehicle**, **Handling**, **Storage**.

Overall, integrating alternative shipment deliveries results in a significantly cheaper utilization of the vehicle fleet. We also note that extending the distribution strategy enables to reduce handling costs, which is a direct consequence of the elimination of intermediaries between suppliers and customers as direct and indirect delivery paths are used. On the other hand, there is an overall increase in storage costs. It is therefore assumed that the combination of centralized, indirect, and direct delivery paths requires longer storage times at warehouses to

α	0	20	40	60
Vehicle	0.0%	6.3%	8.4%	9.0%
Handling	0.0%	3.1%	4.0%	4.7%
Storage	0.0%	-6.8%	-9.7%	-11.7%

Table 21: Relative changes in cost elements of objective function

allow consolidations and ensure high vehicle fill rates.

6. Conclusions

In this article, we study a tactical transportation problem encountered by a 3PL partner in the management of a restaurant supply chain. As the corresponding model is formulated on large time-expanded networks, we propose a network reduction heuristic based on the recent Dynamic Discretization Discovery Algorithm [6]. We present multiple enhancements of DDD to our problem as it has features that DDD was not designed to handle. Our approach constructs a sparse initial time-expanded network that includes transportation and storage arcs with underestimated costs and/or underestimated transit times. Ultimately, a reduced integer reduced program is solved, providing a feasible solution to the original program.

The algorithm is tested on realistic instances based on data provided by the 3PL and compared with CPLEX. The computational results show that the heuristic generates time-expanded networks that yield significantly smaller integer programs than those based upon a complete time-expanded network. Experiments on a set of easy instances show that this problem reduction is consistent, as the heuristic computes close-to-the-optimum solutions in computation times that are significantly less than those needed to solve the original formulation. Results on a set of difficult instances show that the heuristic produces solutions within 30 minutes that outperforms the solutions computed by CPLEX after 5 hours.

A solution approach combining the network reduction heuristic with a Benders decomposition algorithm is used to solve real-life instances and to assess the impact of the distribution strategies contemplated by the 3PL. The method enables us to evaluate and validate these new distribution strategies as they can substantially reduce overall logistics costs without disrupting the 3PL current organization. Results show that more flexible distribution strategies lead to significant reductions in transportation and handling costs and slightly higher storage costs.

A clear avenue for work with a strong practical interest would be to develop a method to extend the distribution strategy efficiently. Indeed, the difficulty of an instance is strongly related to the number of transportation arcs considered in the distribution network. In this study, we use a straightforward approach to extend the distribution strategy, with no guarantee that the extra transportation arcs bring an added value to the design of the transportation plan. Therefore, an interesting extension would be to develop an optimization model that identifies the most promising transportation arcs to integrate into the distribution strategy to improve the transportation

plan overall cost while minimally affecting the model computational tractability. Another avenue of future work is to develop a stochastic programming variant of the LSNDP and investigate how including uncertainty in product demands would impact the transportation plans.

References

[1] Comité National Routier, `http://www.cnr.fr/indices-statistiques/regional-ea/referentiel-prix-de-revient`, Accessed: 2020-04-16.

[2] Amiri, A. (2006). Designing a distribution network in a supply chain system: Formulation and efficient solution procedure. *European journal of operational research*, 171(2):567–576.

[3] Beamon, B. M. (1998). Supply chain design and analysis:: Models and methods. *International journal of production economics*, 55(3):281–294.

[4] Belieres, S., Hewitt, M., Jozefowiez, N., Semet, F., and van Woensel, T. A benders decomposition-based approach for logistics service network design. *European Journal of Operational Research (Accepted for publication)*.

[5] Belieres, S., Hewitt, M., Jozefowiez, N., Semet, F., and van Woensel, T. Meta partial benders decomposition for the logistics service network design problem. *Submitted to Transportation Research Part B: Methodological*.

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

[7] Boland, N., Hewitt, M., Marshall, L., and Savelsbergh, M. (2018). The price of discretizing time: a study in service network design. *EURO Journal on Transportation and Logistics*, pages 1–22.

[8] Cordeau, J.-F., Pasin, F., and Solomon, M. M. (2006). An integrated model for logistics network design. *Annals of operations research*, 144(1):59–82.

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

[10] Crainic, T. G., Hewitt, M., and Rei, W. (2014). Partial decomposition strategies for two-stage stochastic integer programs. *CIRRELT, Centre interuniversitaire de recherche sur les réseaux d'entreprise.*

[11] Crainic, T. G., Rei, W., Hewitt, M., and Maggioni, F. (2016). Partial benders decomposition strategies for two-stage stochastic integer programs. *CIRRELT, Centre interuniversitaire de recherche sur les réseaux d'entreprise . . . .*

[12] Dufour, É., Laporte, G., Paquette, J., and Rancourt, M.-È. (2018). Logistics service network design for humanitarian response in east africa. *Omega*, 74:1–14.

[13] Ford, L. R. and Fulkerson, D. R. (1958). Constructing maximal dynamic flows from static flows. *Operations Research*, 6:419–433.

[14] Ford, L. R. and Fulkerson, D. R. (1962). Flows in networks. *Princeton University Press (1962)*.

[15] Gnegel, F. and Fügenschuh, A. (2020). An iterative graph expansion approach for the scheduling and routing of airplanes. *Computers & Operations Research*, 114:104832.

[16] He, E., Boland, N., Nemhauser, G., and Savelsbergh, M. (2019). Dynamic discretization discovery algorithms for time-dependent shortest path problems. *Optimization online*, 7082.

[17] Hewitt, M. (2019). Enhanced dynamic discretization discovery for the continuous time load plan design problem. *Transportation Science*, 53(6):1731–1750.

[18] Lagos, F., Boland, N., and Savelsbergh, M. (2020). The continuous-time inventory-routing problem. *Transportation Science*.

[19] Medina, J., Hewitt, M., Lehuédé, F., and Péton, O. (2019). Integrating long-haul and local transportation planning: The service network design and routing problem. *EURO Journal on Transportation and Logistics*, 8(2):119–145.

[20] Melo, M. T., Nickel, S., and Saldanha-Da-Gama, F. (2009). Facility location and supply chain management–a review. *European journal of operational research*, 196(2):401–412.

[21] Srivastava, S. K. (2008). Network design for reverse logistics. *Omega*, 36(4):535–548.

[22] Thanh, P. N., Péton, O., and Bostel, N. (2010). A linear relaxation-based heuristic approach for logistics network design. *Computers & Industrial Engineering*, 59(4):964–975.

[23] Vu, D. M., Hewitt, M., Boland, N., and Savelsbergh, M. (2019). Dynamic discretization discovery for solving the time-dependent traveling salesman problem with time windows. *Transportation Science*.

[24] Wieberneit, N. (2008). Service network design for freight transportation: a review. *OR Spectrum*, 30:77–112.

Appendix A. Counter-example

We demonstrate via a counter-example that the DDD algorithm is no longer an exact approach when it is applied to the SNDP with strictly positive storage costs.

Let us assume that a single commodity, available at $(A, 1)$, must be routed to $(B, 3)$. The costs of arcs $((A, t), (A, t+1))$, $((A, t), (B, t+1))$ and $((B, t), (B, t+1))$ are 1, 1 and 2 respectively. Figure A.14 represents the complete time-expanded network $\mathcal{G}_\mathcal{T}$, and the optimal solution obtained when solving the SNDP($\mathcal{G}_\mathcal{T}$). The objective function of this optimal flow has a value of 2. Figure A.15 represents the initial partially time-expanded network $\mathcal{X}_\mathcal{T}$ obtained with Boland *et al.* procedure, as well as the optimal solution of SNDP($\mathcal{X}_\mathcal{T}$). This solution is not feasible for the original problem, as the flow transits along the short arc $((A, 1), (B, 1))$. After refining $\mathcal{X}_\mathcal{T}$, we obtain the partially time-expanded network depicted in Figure A.16. The optimal solution obtained when solving the SNDP($\mathcal{X}_\mathcal{T}$) is feasible for the original problem. However, it has an objective value of 3. Therefore, in this example, the DDD method converges to a sub-optimal solution.

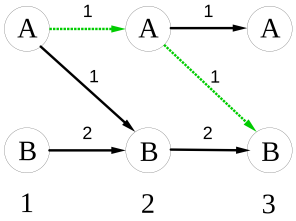

Figure A.14: SNDP($\mathcal{G}_\mathcal{T}$) optimal solution

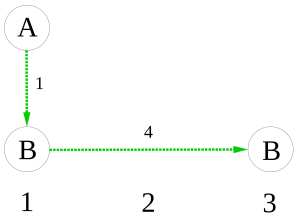

Figure A.15: Optimal solution of the SNPD associated with initial $\mathcal{X}_\mathcal{T}$

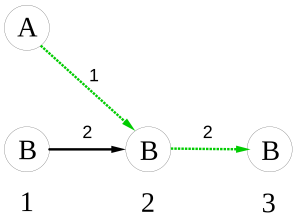

Figure A.16: Optimal solution of the SNDP associated with refined $\mathcal{X}_\mathcal{T}$

Appendix B. Algorithm description

Appendix B.1. Building the initial $\mathcal{X}_\mathcal{T}$

Algorithm 2 Algorithm to build the initial network $\mathcal{X}_\mathcal{T}$

Require: Static network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, time horizon $\mathcal{T}$, product demands

Create complete time-expanded network $\mathcal{G}_\mathcal{T}$ and solve $LP(\mathcal{G}_\mathcal{T})$

Create $\mathcal{X}_\mathcal{T}$, add all arcs $((i,t),(j,t+t_{ij}))$ that appear in the solution, $i \neq j$

Add all $(i,0)$ to $\mathcal{X}_\mathcal{T}$ for all $i \in \mathcal{N}$

for $(i,t) \in \mathcal{X}_\mathcal{T}, i \in \mathcal{W}^\mathcal{P} \cup \mathcal{W}^\mathcal{S}$ **do**

  Find node $(i,t') \in \mathcal{X}_\mathcal{T}$ with smallest t' such that $t' \geq t$

  Add $((i,t),(i,t'))$ to $\mathcal{X}_\mathcal{T}$ with null storage cost

end for

for $(i,t) \in \mathcal{X}_\mathcal{T}$ **do**

  for $(i,j) \in \mathcal{A}$ **do**

    Find node $(j,t') \in \mathcal{X}_\mathcal{T}$ with largest t' such that $t' \leq t + t_{ij}$

    if $((i,t),(j,t')) \notin \mathcal{X}_\mathcal{T}$ **then**

      Add $((i,t),(j,t'))$ to $\mathcal{X}_\mathcal{T}$ with a fixed cost $f_{ij} \times \frac{\mathcal{T}-(t+t_{ij}+-t')}{\mathcal{T}}$ and a linear cost $\dot{c}_{ij} \times \frac{\mathcal{T}-(t+t_{ij}+-t')}{\mathcal{T}}$

    end if

  end for

end for

The construction procedure of the initial $\mathcal{X}_\mathcal{T}$ is reported in Algorithm 2. First, the linear relaxation of the original program, $LP(\mathcal{G}_\mathcal{T})$, is solved. All the transportation arcs present in the optimal solution found are added to $\mathcal{X}_\mathcal{T}$. For each warehouse (i,t), we find the next occurrence (if it exists) (i,t'), i.e. t' the smallest value such that $t' > t$, and we construct an holding arc $((i,t),(i,t'))$ with null storage cost. As a result, the optimal solution of $LP(\mathcal{G}_\mathcal{T})$ is feasible and optimal as well for $LP(\mathcal{X}_\mathcal{T})$. Because any solution of $LP(\mathcal{G}_\mathcal{T})$ can be transformed into a feasible solution of $IP(\mathcal{G}_\mathcal{T})$ by rounding-up the fractional values, adding these nodes and arcs to $\mathcal{X}_\mathcal{T}$ ensures that at least one solution of $IP(\mathcal{X}_\mathcal{T})$ is feasible for $IP(\mathcal{G}_\mathcal{T})$. This is done to ensure that the algorithm returns at least one feasible solution for $IP(\mathcal{G}_\mathcal{T})$.

Next, short arcs are added to $\mathcal{X}_\mathcal{T}$ as in [6]. For each time-expanded node $(i,t) \in \mathcal{X}_\mathcal{T}$ and each static arc $(i,j) \in \mathcal{G}$, the node (j,t') with the largest t' such that $t' \leq t + t_{ij}$ is searched. If it does not already exist, the arc $((i,t),(j,t'))$ is added to $\mathcal{X}_\mathcal{T}$. The linear and fixed costs of $((i,t),(j,t'))$ are obtained by multiplying $\dot{c}_{ij}$ and f_{ij} by $\frac{\mathcal{T}-(t+t_{ij}+-t')}{\mathcal{T}}$. This value is between 0 and 1 and is proportional to the difference between the duration of $((i,t),(j,t'))$ and the original transit time t_{ij}. The reason is that short arcs are intended to allow for consolidations

that are infeasible in real life, which permits to compute cheaper solutions that use fewer vehicles than necessary. However, consolidation savings are not reflected in the objective function if the integrality of the vehicle variables is relaxed. Therefore, when solving $LP(\mathcal{X}_\mathcal{T})$, utilizing short arcs presents no particular advantage if all time copies $((i,t),(j,t'))$ of (i,j) have the same cost.

Appendix B.2. Phase I: Short-arcs removal

Phase I takes the current optimal solution of $LP(\mathcal{X}_\mathcal{T})$ as an input parameter and repairs iteratively the time-expanded network. At each iteration, short arcs with a non-zero flow are corrected and $LP(\mathcal{X}_\mathcal{T})$ is solved again. Phase I terminates when no short arcs appear in the optimal solution of $LP(\mathcal{X}_\mathcal{T})$. An overview is given in Algorithm 3.

Algorithm 3 Modified DDD

Require: Optimal solution of $LP(\mathcal{X}_\mathcal{T})$
  while Optimal solution of $LP(\mathcal{X}_\mathcal{T})$ uses short arcs **do**
    for short arcs $((i,t),(j,t')) \in \mathcal{X}_\mathcal{T}$ with positive flow **do**
      Remove $((i,t),(j,t'))$ from $\mathcal{X}_\mathcal{T}$
      if j is not a customer **then**
        Add node (j, t_{new}) to $\mathcal{X}_\mathcal{T}$, $t_{new} = t + t_{ij}$
        Add arc $((i,t),(j,t_{new}))$ to $\mathcal{X}_\mathcal{T}$ with original costs
      else
        Add node (i, t_{new}) to $\mathcal{X}_\mathcal{T}$, $t_{new} = t' - t_{ij}$
        Add arc $((i,t_{new}),(j,t'))$ to $\mathcal{X}_\mathcal{T}$ with original costs
      end if
      Restore network based on the new node
    end for
    Solve $LP(\mathcal{X}_\mathcal{T})$
  end while

The correction of a short timed-copy of arc (i,j) consists of (1) removing the short-arc, and (2) creating a new timed-copy of (i,j) that reflects real transit time t_{ij}. In the original DDD algorithm, such timed-copy is obtained by lengthening "forwardly" the short-arc $((i,t),(j,t'))$, i.e. creating (j,t_{new}), the occurrence of the destination node with $t_{new} = t + t_{ij}$, and adding $((i,t),(j,t_{new}))$ to the time-expanded network. In our heuristic, short arcs that do not arrive to a customer are lengthened in a similar fashion. On the contrary, short arcs $((i,t),(j,t'))$ that arrive to a customer are lengthened "backwardly", i.e. by creating (i,t_{new}), the occurrence of the origin node with $t_{new} = t' - t_{ij}$, and by replacing $((i,t),(j,t'))$ by $((i,t_{new}),(j,t'))$.

The backward refinement is motivated by the multi-echelon network structure and the fact, by construction, the initial $\mathcal{X}_{\mathcal{T}}$ already contains all customers (i, t), $i \in \mathcal{C}$, with positive demands. Because customers only have incoming arcs, a customer with zero demand can be seen as a "dead-end node" that does not enable us to discover new potential solutions. Consequently, we avoid adding such nodes to the network by lengthening "backwardly" the short arcs $((i, t), (j, t'))$ with j a customer.

Algorithm 4 Restore network

Require: New node (j, t_{new})

  Restore transportation arcs (i, t_{new})

  if $i \in \mathcal{W}^{\mathcal{P}} \cup \mathcal{W}^{\mathcal{S}}$ **then**

    Restore storage arcs (i, t_{new})

  end if

Similarily to [6], the replacement of the short-arc is followed by an update of the network that aims to connect the newly created node (j, t_{new}) with already existing nodes. This restoration phase is described in Algorithm 4. It operates in two steps (1) updating the transportation arcs (Algorithm 5) in relation to (j, t_{new}), and (2) updating the storage arcs (Algorithm 6) in relation to (j, t_{new}) if j is a warehouse.

Algorithm 5 Restore transportation arcs

Require: New node (j, t_{new})

  for $((i, t), (j, t')) \in \mathcal{X}_{\mathcal{T}}$ **do**

    if $t' < t_{new}$ AND $t_{new} - t \leq t_{ij}$ **then**

      Remove $((i, t), (j, t'))$ from $\mathcal{X}_{\mathcal{T}}$

      Add arc $((i, t), (j, t_{new}))$ to $\mathcal{X}_{\mathcal{T}}$, with original costs multiplied by $\frac{\mathcal{T}-(t+t_{ij}+-t_{new})}{\mathcal{T}}$

    end if

  end for

  for $(j, l) \in \mathcal{G}$ **do**

    Find node $(l, t') \in \mathcal{X}_{\mathcal{T}}$ with largest t' such that $t' - t \leq t_{jl}$

    Add $((j, t_{new}), (l, t'))$ to $\mathcal{X}_{\mathcal{T}}$, with original costs multiplied by $\frac{\mathcal{T}-(t_{new}+t_{jl}+-t')}{\mathcal{T}}$

  end for

First, we restore transportation arcs in relation to (j, t_{new}). If a transporation arc $((i, t), (j, t'))$ is short and can be lengthened towards new node (j, t_{new}) while not exceeding (i, j) real transit time, we remove that arc from $\mathcal{X}_{\mathcal{T}}$ and add $((i, t), (j, t_{new}))$. For each arc (j, l) in the flat network, we determine the node (l, t') in $\mathcal{X}_{\mathcal{T}}$ with the largest value of t' such that $((j, t_{new}), (l, t'))$ does not exceeds (j, l) real transit time and we add the resulting arc to $\mathcal{X}_{\mathcal{T}}$. Note that the new transportation arcs are potentially short, in which case their costs are strictly lower than the original values.

Algorithm 6 Restore storage arcs

Require: New warehouse (j, t_{new})

Find node $(j, t_1) \in \mathcal{X}_{\mathcal{T}}$ with largest t_1 such that $t_1 \leq t_{new}$

Add $((j, t_1), (j, t_{new}))$ to $\mathcal{X}_{\mathcal{T}}$

Find node $(j, t_2) \in \mathcal{X}_{\mathcal{T}}$ with smallest t_1 such that $t_{new} \leq t_2$

if (j, t_2) exists **then**

Add $((j, t_{new}), (j, t_2))$ to $\mathcal{X}_{\mathcal{T}}$

Remove $((j, t_1), (j, t_2))$ from $\mathcal{X}_{\mathcal{T}}$

end if

Second, we restore storage arcs in relation to (j, t_{new}) if it models a warehouse. To enable the storage of products through the new node, we add storage arcs connecting (j, t_{new}) to adjacent occurrences of j. As $\mathcal{X}_{\mathcal{T}}$ contains $(j, 0)$ for all $j \in \mathcal{N}$, (j, t_{new}) necessarily has a previous occurrence. We create a storage arc from the immediately preceding occurrence (j, t_1) to (j, t_{new}). If (j, t_{new}) also has following occurrence, we find the immediately following occurrence (j, t_2), we create storage arc $((j, t_{new}), (j, t_2))$ and we delete obsolete arc $((j, t_1), (j, t_2))$.

Appendix C. Instance details

$\|\mathcal{N}\|$	α	$\|\mathcal{A}\|$	$\|\mathcal{P}\|$	D	Δ	Season
20	0	19	69	14	1	Summer
20	0	19	69	14	1	Winter
20	0	19	69	14	2	Summer
20	0	19	69	14	2	Winter
20	0	19	69	14	3	Summer
20	0	19	69	14	3	Winter
20	25	23	69	14	1	Summer
20	25	23	69	14	1	Winter
20	25	23	69	14	2	Summer
20	25	23	69	14	2	Winter
20	25	23	69	14	3	Summer
20	25	23	69	14	3	Winter
20	50	28	69	14	1	Summer
20	50	28	69	14	1	Winter
20	50	28	69	14	2	Summer
20	50	28	69	14	2	Winter
20	50	28	69	14	3	Summer
20	50	28	69	14	3	Winter

Table C.22: *Easy instances:* 1^{st} *network configuration*

$\|\mathcal{N}\|$	α	$\|\mathcal{A}\|$	$\|\mathcal{P}\|$	D	Δ	Season
20	0	19	77	14	1	Summer
20	0	19	77	14	1	Winter
20	0	19	77	14	2	Summer
20	0	19	77	14	2	Winter
20	0	19	77	14	3	Summer
20	0	19	77	14	3	Winter
20	25	23	77	14	1	Summer
20	25	23	77	14	1	Winter
20	25	23	77	14	2	Summer
20	25	23	77	14	2	Winter
20	25	23	77	14	3	Summer
20	25	23	77	14	3	Winter
20	50	30	77	14	1	Summer
20	50	30	77	14	1	Winter
20	50	30	77	14	2	Summer
20	50	30	77	14	2	Winter
20	50	30	77	14	3	Summer
20	50	30	77	14	3	Winter

Table C.23: *Easy instances:* 2^{nd} *network configuration*

$\|\mathcal{N}\|$	α	$\|\mathcal{A}\|$	$\|\mathcal{P}\|$	D	Δ	Season
20	0	19	60	14	1	Summer
20	0	19	60	14	1	Winter
20	0	19	60	14	2	Summer
20	0	19	60	14	2	Winter
20	0	19	60	14	3	Summer
20	0	19	60	14	3	Winter
20	25	24	60	14	1	Summer
20	25	24	60	14	1	Winter
20	25	24	60	14	2	Summer
20	25	24	60	14	2	Winter
20	25	24	60	14	3	Summer
20	25	24	60	14	3	Winter
20	50	25	60	14	1	Summer
20	50	25	60	14	1	Winter
20	50	25	60	14	2	Summer
20	50	25	60	14	2	Winter
20	50	25	60	14	3	Summer
20	50	25	60	14	3	Winter

Table C.24: *Easy instances:* 3^{rd} *network configuration*

$\|\mathcal{N}\|$	α	$\|\mathcal{A}\|$	$\|\mathcal{P}\|$	D	Δ	Season
20	0	19	65	14	1	Summer
20	0	19	65	14	1	Winter
20	0	19	65	14	2	Summer
20	0	19	65	14	2	Winter
20	0	19	65	14	3	Summer
20	0	19	65	14	3	Winter
20	25	22	65	14	1	Summer
20	25	22	65	14	1	Winter
20	25	22	65	14	2	Summer
20	25	22	65	14	2	Winter
20	25	22	65	14	3	Summer
20	25	22	65	14	3	Winter
20	50	23	65	14	1	Summer
20	50	23	65	14	1	Winter
20	50	23	65	14	2	Summer
20	50	23	65	14	2	Winter
20	50	23	65	14	3	Summer
20	50	23	65	14	3	Winter

Table C.25: *Easy instances:* 4^{th} *network configuration*

$\|\mathcal{N}\|$	α	$\|\mathcal{A}\|$	$\|\mathcal{P}\|$	D	Δ	Season
20	0	19	62	14	1	Summer
20	0	19	62	14	1	Winter
20	0	19	62	14	2	Summer
20	0	19	62	14	2	Winter
20	0	19	62	14	3	Summer
20	0	19	62	14	3	Winter
20	25	22	62	14	1	Summer
20	25	22	62	14	1	Winter
20	25	22	62	14	2	Summer
20	25	22	62	14	2	Winter
20	25	22	62	14	3	Summer
20	25	22	62	14	3	Winter
20	50	23	62	14	1	Summer
20	50	23	62	14	1	Winter
20	50	23	62	14	2	Summer
20	50	23	62	14	2	Winter
20	50	23	62	14	3	Summer
20	50	23	62	14	3	Winter

Table C.26: *Easy instances:* 5^{th} *network configuration*

$\lvert\mathcal{N}\rvert$	α	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{P}\rvert$	D	Δ	Season
67	0	66	131	14	2	Summer
67	0	66	131	14	2	Winter
67	0	66	131	14	4	Summer
67	0	66	131	14	4	Winter
67	0	66	131	14	6	Summer
67	0	66	131	14	6	Winter
67	20	101	131	14	2	Summer
67	20	101	131	14	2	Winter
67	20	101	131	14	4	Summer
67	20	101	131	14	4	Winter
67	20	101	131	14	6	Summer
67	20	101	131	14	6	Winter

$\lvert\mathcal{N}\rvert$	α	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{P}\rvert$	D	Δ	Season
67	40	116	131	14	2	Summer
67	40	116	131	14	2	Winter
67	40	116	131	14	4	Summer
67	40	116	131	14	4	Winter
67	40	116	131	14	6	Summer
67	40	116	131	14	6	Winter
67	60	135	131	14	2	Summer
67	60	135	131	14	2	Winter
67	60	135	131	14	4	Summer
67	60	135	131	14	4	Winter
67	60	135	131	14	6	Summer
67	60	135	131	14	6	Winter

Table C.27: *Difficult instances*