Politecnico di Torino

Department of Management and Production Engineering

Master's degree in Management Engineering

Service Network Design integrated with Bin Packing

Thesis in Operational Research

Candidate:

Maria Chiara Di Fazio 317327

Supervisors:

Guido Perboli

Sara Khodaparasti

Alla bellezza.

Contents

1 Introduction 9

2 What is Service Network Design? 11

2.1 SND & SSND 11

2.2 Why is SND important? 13

3 Literature Review 16

4 Service Network: basic definitions 19

4.1 Planning activities levels 25

5 Service Network Design Issues: integration with Bin Packing Problem 27

5.1 The Bin Packing Problem 29

5.2 SND with Bin Packing considerations 31

5.3 The mathematical model 35

6 The model on AIMMS 37

6.1 AIMMS Software 37

6.2 Sets 38

6.3 Parameters 40

6.4 Variables 49

6.5 Objective Function 51

6.6 Constraints 52

6.7 Mathematical Program 56

6.8 Data initialization 57

6.9 Output report 58

6.10 The procedures 59

7 Data Generation 60

7.1 Service creation 61

7.2 Demand Creation 62

7.2.1 Second level duplicates verification for demand 67

8 Execution & Results 70

8.1 Results with no fixed-contract 71

8.2 Results with fixed-contracts 75

8.3 Observations 79

8.3.1 Differences with classical Service Network Design 82

9 Conclusion 85

References 88

List of Figures

Figure 1-Physical and time-space service network 12
Figure 2- High-level classification of freight carriers (Crainic & Rei, 50 Years of Operations Research for Planning Consolidation-based Freight Transportation, 2024) 20
Figure 3 - M1M system structure, stakeholders, communications & decisions (Crainic & Rei, 50 Years of Operations Research for Planning Consolidation-based Freight Transportation, 2024) 22
Figure 4-Decision & planning levels and related O.R. methodology (Crainic & Rei, 50 Years of Operations Research for Planning Consolidation-based Freight Transportation, 2024) 25
Figure 5-Physical network example 33
Figure 6- Physical Network Instances (Taherkhani, Bilegan, Crainic, Gendreau, & Rei, 2022) 60
Figure 7- Macro for service creation (Cyclic case) 62
Figure 8- Macro for commodities creation (Non-cyclic case) 64
Figure 9- Macro for commodities creation (Cyclic case) 65
Figure 10- Macro for data duplication verification (Non-cyclic case) 68
Figure 11-Macro for data duplication verification (Cyclic case) 68
Figure 12 - Demand 30 allocation, from grid 1 80
Figure 13- Demand 19 allocation, from grid 1 81

List of Tables

Table 1- Bin types, capacities and costs 70
Table 2- Execution results for hyper-corridor, no fixed-contract (1) 72
Table 3- Execution results for hyper-corridor, no fixed-contract (2) 72
Table 4-Execution results for bipartite, no fixed-contract (1) 73
Table 5-Execution results for bipartite, no fixed-contract (2) 73
Table 6-Execution results for grid, no fixed-contract (1) 74

Table 7- Execution results for grid, no fixed-contract (2) 74
Table 8-Execution results for hyper-corridor, fixed-contract (1) 76
Table 9-Execution results for hyper-corridor, fixed-contract (2) 76
Table 10-Execution results for bipartite, fixed-contract (1) 77
Table 11-Execution results for bipartite, fixed-contract (2) 77
Table 12-Execution results for grid, fixed-contract (1) 78
Table 13-Execution results for grid, fixed-contract (2) 78
Table 14 - Percentage of used capacity with the integrated approach 82
Table 15 - Inefficiency of classical SND, KPI: Objective Function 83

List of Captures

Capture 1- Demand_set 39
Capture 2- Set 39
Capture 3- BinTypes_set 40
Capture 4-NumBinTypes 41
Capture 5- NumDemands 41
Capture 6- par_order 41
Capture 7- par_Link 41
Capture 8- par_yy 42
Capture 9- par_binTime 42
Capture 10- FirstOrigin 42
Capture 11- FirstDestination 42
Capture 12- early_demand 43
Capture 13- late_demand 43
Capture 14- NewDemand 43
Capture 15- GlobalU 44
Capture 16- Q 44
Capture 17- C 44
Capture 18- volume_item 44

Capture 19- T_cost 45
Capture 20- hold 45
Capture 21- f2 45
Capture 22- profit 45
Capture 23- par_profit 46
Capture 24- serviceCost 46
Capture 25- bincost 46
Capture 26- holdcost 47
Capture 27- TransportationCost 47
Capture 28- totalcost 47
Capture 29- totalProfit 48
Capture 30- DurationNew 48
Capture 31- Max_Travel_Item 49
Capture 32- lambda_accepted 50
Capture 33- xx 50
Capture 34- zz 50
Capture 35- yy 50
Capture 36- ww 51
Capture 37- OBJ_SND_Profit 52
Capture 38- Flow_con_WithProfit 53
Capture 39- Con_BinCap_Simple 54
Capture 40- Con_GlobalCap 54
Capture 41- Con_Link_zz_yy 55
Capture 42- Lambda_equal_one 55
Capture 43- Limit_travel_time 55
Capture 44- SND_withProfit 57
Capture 45- my_win 58
Capture 46- create_profit_for_demand 59
Capture 47- Run_this_procedure 59

Abstract

Freight transportation is crucial for the global economy, enabling the movement of goods between multiple locations. To address this issue, Operational Research (OR) methodologies, particularly Service Network Design (SND), are employed to optimize carrier services and meet shipper demands. The purpose of the work is to explore SND integrated with Bin Packing constraints. This necessity derives from the scarcity of research conducted on this topic. The first part of the thesis introduces the basic concepts and describes how freight transportation works. In particular, the need to integrate bin packing considerations into the traditional model is emphasized. This section ends with the presentation of the mathematical model for SND integrated with Packing constraints. In the second part, the process of data generation, for the following computation of the model, is described. For this aim, the procedures used to create consistent data for three types of networks are exposed. In the last section, the model is executed using the AIMMS software and applied to the instances generated. The work concludes by highlighting the main differences between Classical and Integrated Service Network Design, demonstrating the advantages of an integrated approach, not only from a mathematical standpoint, but also on a logistical and operational level.

1 Introduction

Service Network Design (SND) is the process of strategically planning and optimizing a network of service delivery systems to efficiently meet customer demands while minimizing operational costs. As can be imagined, it plays a crucial role in transportation and logistics systems which nowadays are of utmost importance, since the complexity and the scale of modern logistics networks have increased.

Factors such as globalization, e-commerce growth, and customer demand for faster delivery times place unprecedented demands on service networks. Consequently, designing an effective service network involves balancing multiple competing objectives, including cost minimization, capacity utilization, environmental impact, and service reliability. Traditionally, SND has focused on static design solutions, treating demand as a fixed input, and optimizing based on stable assumptions. However, dynamic factors such as fluctuating demand, variable travel times, and resource availability present additional challenges that call for more adaptable and robust SND models. For this reason, a new version of Service Network Design takes these aspects into account, and it is the so called Scheduled Service Network Design (SSND).

Service Network Design and all the other kinds of network design (such as Telecommunications Network Design, Supply Chain Network Design, Computer Network Design etc.) are intrinsically linked to Operational Research (OR) as it employs various OR methodologies, techniques and tools. Therefore, the scope of this thesis is under the umbrella of Operational Research.

This work starts delving into the field of Service Network Design, understood in its scheduled version, and explain why it is relevant to modern reality. Next, the classical version of SND is integrated with constraints related to Bin Packing. The Bin Packing problem is a classic and famous problem in operational research, and integrating it into SND means considering aspects that, although fundamental, have rarely been integrated until now.

Then, a mathematical model reflecting the integration of SND with Bin Packing problem is presented and translated into the AIMMS language. Based on this model, data representing realistic instances are randomly generated and fed into the model to evaluate its execution and outputs.

In the final section, the results are analyzed alongside some concluding remarks. Specifically, the differences between an integrated and a non-integrated approach are highlighted, and it is demonstrated why integrating Bin Packing proves to be more advantageous.

2 What is Service Network Design?

The goal of Service Network Design is to optimize the design of a service network by selecting routes, scheduling services, and allocating demands while minimizing costs and satisfying demand requirements and capacity constraints. The objective is typically to minimize the total cost of operating the network, which includes both fixed costs for setting up services and variable costs that depend on the amount of flow. The model may have different types of constraints. Typical constraints are:

- Flow Conservation: Ensures that demands flow properly from origins to destinations without being lost or misrouted.
- Capacity Constraints: Limits the amount of demand that can be carried on a given route to the capacity of the service being used.
- Demand Satisfaction: Ensures that every demand is completely satisfied.

Typical decision variables are:

- Continuous variable representing the flow of the commodity on the arc.
- Binary variable indicating whether the arc is used (1 if open, 0 otherwise).

2.1 SND & SSND

The classic formulations for Service Network Design are not integrated with time perspective. This is why it is necessary to distinguish between Service network Design and Scheduled Service Network Design (SSND).

The basic formulations of SND can be used in situations where there is no variation in demand for the duration of the schedule length. Either the schedule length is short and one assumes that everything "happens simultaneously," or the demand arrivals and service departures are assumed to be uniformly distributed over the schedule length for longer time spans.

Basically, SSND is a more complex and realistic version of SND because it considers that demands and services vary in time, while for SND it is like they are constant and uniform. Because the time attributes of the various problem elements are not explicitly included, those SND models are qualified as *static* (Crainic T. , Consolidation-based Transportation Planning: The Service Network Design Methodology, 2024).

Differently, time attributes are explicitly considered in the SSND. Since SSND models incorporate scheduling considerations into the design, they are called *dynamic* or *time-sensitive*.

Because not only terminals are involved, for SSND a time-space network is considered (Figure 1). This means that each node corresponds to a terminal on a specific time (e.g. node 1 at time 2 is a different entity from node 1 at time 3).

Before delving into the subject in depth, it is important to expound on the basic definitions of the Service Network, which will be explained in the next section.

From now on, for the sake of simplicity, by SND will be intended the scheduled version (SSND), as the time factor is considered.

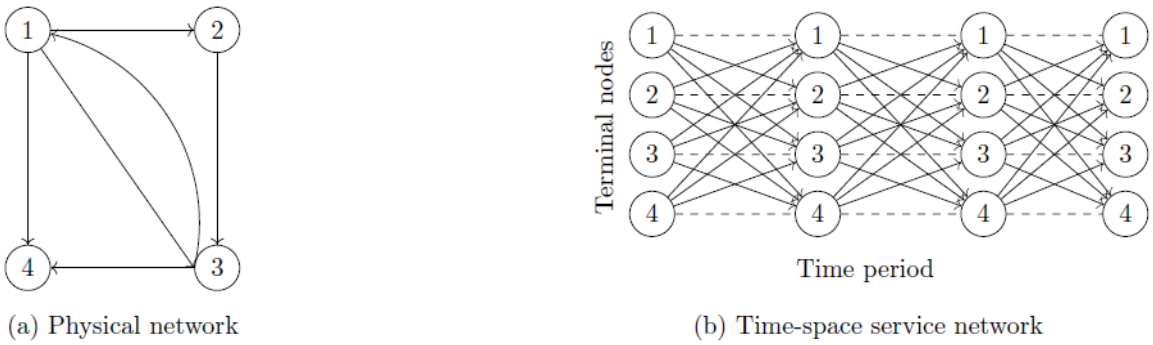


Figure 1-Physical and time-space service network

2.2 Why is SND important?

In today's highly interconnected and fast-paced world, Service Network Design plays a crucial role in optimizing the performance of large-scale operations across various industries, particularly in logistics and transportation. These industries rely heavily on the efficiency of their networks to deliver services and goods, making the optimization of these systems more critical than ever.

When integrated with the Bin Packing Problem (BPP), a well-known problem in combinatorial optimization, the relevance and importance of this problem become even more pronounced considering several modern challenges and trends.

One of the primary factors driving the significance of Service Network Design is the exponential growth of e-commerce, which has drastically increased the complexity of supply chains. With an unprecedented surge in demand for fast, efficient, and cost-effective logistics solutions, companies face the challenge of managing vast networks of transportation routes, hubs, and vehicles. In such scenarios, combining SND with BPP becomes essential, as it ensures that transportation resources such as trucks, containers, or delivery vans are used as efficiently as possible. The bin packing aspect allows for the optimal consolidation of goods, reducing the number of containers or vehicles required. This, in turn, lowers operational costs, improves delivery times, and reduces inefficiencies throughout the supply chain.

In addition to economic pressures, the growing emphasis on sustainability and green logistics further underlines the importance of this integrated problem. Companies today are increasingly focused on reducing their environmental footprint, and optimizing logistics networks is a key strategy in achieving these sustainability goals. Efficiently packing goods and designing streamlined service networks helps minimize the number of vehicles on the road, thereby reducing fuel consumption and emissions. This not only aligns with corporate sustainability initiatives but also helps companies comply with environmental regulations. The integration of SND

and BPP thus directly contributes to greener operations while maintaining high service efficiency, a vital consideration in modern logistics.

The rise in resource scarcity and the corresponding increase in the cost of resources like fuel, labor, and transportation assets also highlight the need for optimization. With rising costs, businesses are under pressure to make the most out of their available resources. Service Network Design, when integrated with the Bin Packing Problem, allows companies to optimize their use of space and assets, minimizing underutilized capacity. For instance, by ensuring that vehicles and containers are packed efficiently, businesses can reduce empty miles and eliminate wasted resources. This careful optimization translates into significant cost savings and improved overall efficiency, making it a valuable solution in today's cost-conscious environment.

Moreover, urbanization and the accompanying rise in last-mile delivery challenges bring another layer of complexity to service network design. As cities continue to expand and become more congested, logistics operations, especially those related to last-mile deliveries, face new challenges. The final leg of delivery is often the most expensive and inefficient part of the supply chain. By integrating the Bin Packing Problem into the design of service networks, companies can optimize the way they consolidate and route deliveries in dense urban areas, ensuring that vehicles are fully loaded and routes are optimized to minimize travel time and distance. This leads to quicker and more cost-effective deliveries, while also helping reduce the strain on urban infrastructure.

Another significant trend that underscores the importance of SND is the shift toward dynamic and real-time logistics. With the increasing demand for fast deliveries, including same-day and next-day services, logistics networks must be more flexible and adaptable than ever before. The ability to reconfigure service networks in real-time, adapt to fluctuating demand, and optimize the use of resources in a constantly changing environment is critical. The combination of Service Network Design with

the Bin Packing Problem allows businesses to respond more quickly to these real-time logistics challenges. It ensures that transportation resources are allocated efficiently while maintaining the flexibility required to handle variability in shipment sizes and delivery frequencies.

Globalization has also made supply chains more complex, requiring multi-modal transportation solutions that span international borders. This complexity increases the importance of optimizing across different modes of transportation—whether it be air, sea, rail, or road. The integration of SND with BPP helps companies manage this complexity by ensuring that shipments are packed and routed optimally, regardless of the transportation mode. This smooths global logistics operations and ensures that businesses can manage their networks efficiently on an international scale.

The concept of the Physical Internet, introduced by ALICE (Alliance for Logistics Innovation through collaboration in Europe) (ALICE) brings new layers of complexity. The Physical Internet 2030 is a concept aimed at revolutionizing global logistics, making it more efficient, sustainable, and interconnected, much like the architecture of the digital internet. It involves the use of standardized containers and interoperable transportation systems, with the goal of facilitating resource sharing and reducing costs. In essence, the Physical Internet pushes SND to broader networks and higher attention on the container space.

Fortunately, advancements in computational power and optimization algorithms have made it more feasible to solve such complex problems. In the past, solving the integrated SND-BPP problem at a large scale would have been computationally prohibitive.

3 Literature Review

The vast body of work on Service Network Design spans various problem settings, model formulations, solution techniques, and practical applications (Crainic, Service network design in freight transportation, 2000) (Crainic T. , Consolidation-based Transportation Planning: The Service Network Design Methodology, 2024).
Much of the work on SSND methodologies emerged after 2009 (Jarrah, Ahmad, Johnson, & Neubert, 2009) and has since been applied to areas such as freight rail transportation (Zhu, Endong, Crainic, & Gendreau, 2014), less-than-truckload transportation (Hewitt & Lehuédé, 2023) and urban logistics (Ricciardi, Storchi, & Crainic, 2009).

On the supply side, consolidated freight services are commonly modeled using multi-commodity Service Network Design (SND) frameworks (Crainic, Gendreau, & Gendron, Network design with applications to transportation and logistics, 2021). Many studies examine either the perspective of transportation providers (with an emphasis on minimizing costs under capacity constraints) or the perspective of shippers.

The study "Tactical capacity planning in an integrated multi-stakeholder freight transportation system" (Taherkhani, Bilegan, Crainic, Gendreau, & Rei, 2022) examined a multi-stakeholder system that factored in revenues, transportation, and warehousing costs. It explored the SSND problem on a hyper-corridor network with multiple terminals and zones, highlighting the challenge of scheduling services to maximize fulfillment of shipper demand. Research on demand management in consolidated freight services remains relatively scarce (Tawfik & Christine and Sabine Limbourg, 2018). Revenue management decisions, such as demand segmentation, pricing strategies, and capacity allocation across different segments or channels are examples of this (Talluri & Kalyan and Garrett van Ryzin, 2004). The integration of revenue management into tactical or operational planning introduces additional complexity to already difficult computational problems. For example, "Pricing intermodal freight transport services: A cost-plus-pricing strategy" (Li, Lin, Negenborn, & De Schutter, 2015) studied the pricing strategies used by intermodal freight operators. Other research (Crevier, Cordeau, & Savard, 2012) applied revenue

management to railway operations, while others (Wang & Yunfei et al., 2016) focused on accept/reject decisions in barge transport networks. “Revenue management for rail container transportation” (Bilegan & Ioana et al., 2015) proposed a dynamic approach for managing bookings in a railway corridor, based on demand forecasts, while others (Luo, Ting, Long Gao, & Yalçin Akçay, 2016) investigated capacity leasing and demand acceptance in intermodal transport under uncertain conditions. A further investigation (Riessen, Bart van, Negenborn, & Dekker, 2017) demonstrated how offering multiple fare classes can substantially increase revenues along a single corridor.

The integration of packing constraints into SND models is relatively uncommon. In “The transportation problem with packing constraints“ (Flamand, Iori, & Haouari, 2023) the authors addressed a multi-commodity transportation problem that combined transportation logistics with the variable-sized bin packing problem. In their study, different commodities were shipped from supply nodes to demand nodes using a heterogeneous fleet of vehicles, and packing constraints were used to model vehicle capacity. The model was a network design problem with integer flows, allowing for demand splitting across vehicles. Crainic, Fomeni and Rei (Crainic, Fomeni, & Rei, 2021) presented a multi-period bin packing model to address capacity planning in corridor-based logistics. They developed a system that optimizes the assignment of commodities to bins over a multi-period planning horizon, using a pre-existing service corridor between two regions. Different heuristic algorithms were proposed to solve the complex problem.

Hewitt and Lehuédé (Hewitt & Lehuédé, 2023) proposed a novel formulation for a basic variant of SSND, in which each origin-destination demand is characterized by volume and time windows at both the origin and destination. Their model minimizes vehicle costs while satisfying demand through service consolidation. In this context, the consolidation of multiple shipments on a single terminal-to-terminal service is encouraged to reduce the number of vehicles used. Their approach offered a more compact formulation compared to classic time-space network models. Furthermore, they introduced a hybrid formulation for larger-scale problems, addressing dispatch times as consolidations or time-space knapsacks depending on the planning period and problem size. Their work stands out by comparing the

effectiveness of the two approaches, and in particular, how packing is incorporated into the model.

While much of the existing literature focuses on vehicle routing problems, which generally feature simpler networks and operational structures, the integration of more advanced packing considerations remains limited in Service Network Design. Even in routing problems, packing constraints are typically addressed as subproblems, such as verifying the feasibility of assigned loads within a vehicle and applying valid inequalities when packing is not feasible. However, the selection of appropriate loading units, in terms of attributes and total cost, is often overlooked. This aspect is crucial in packing problems, especially in SSND contexts where mode selection, carrier choice, and the management of heterogeneous vehicle fleets are necessary. Despite these advancements, there is still no comprehensive framework that fully integrates packing considerations into network design problems.

The main point of this review is that, while there are many studies on Service Network Design (SND), few of them fully consider the packing problems and the challenges related to optimally managing vehicle capacity. The review suggests that there are still many opportunities to improve these models, especially by integrating considerations related to packing and managing heterogeneous vehicle fleets.

4 Service Network: basic definitions

Carriers operate on infrastructure networks. The nodes of these networks are the *terminals* where freight and vehicles are handled. Concretely, terminals are facilities where goods are consolidated/deconsolidated and, eventually, stored for varying periods. Terminals may belong to the carriers or not, in the last case the carrier would be paying for their utilization. Terminals, then are connected by physical *infrastructure* paths. Similarly to terminals, the infrastructure may be carrier-owned or the carrier may rent and pay for the passage and usage of it. Over the physical network, carriers offer capacity in terms of services from an origin terminal to a destination terminal, without intermediate stops.

A shipper demand for transportation involves requesting the movement of a specific quantity and type of freight between two terminals. The generic term *"freight type"* is used to recall that each demand is for a particular product, with specific physical and transportation characteristics and requirements, including weight, size, fragility, risk as well as handling and product-vehicle adequacy rules. Demand is also characterized by economic and service elements. The latter generally involves the time allocated to the delivery of freight to destination. The former takes the form of the fee, tariff, shippers pay for the transportation of their goods, which is conditioned by a combination of freight type, distance, service requirements, and commercial understanding (penalties, for late deliveries and damage, for example, may be part of the commercial deal). The carrier answers this demand by operating vehicles according to a set of services defined between the terminals of its network.

The literature has identified four dimensions (Figure 2) addressing how customer demand is loaded and moved, the geographical scope of operations, the transportation mode, and the structure of the system and associated decision making (Crainic & Rei, 50 Years of Operations Research for Planning Consolidation-based Freight Transportation, 2024).

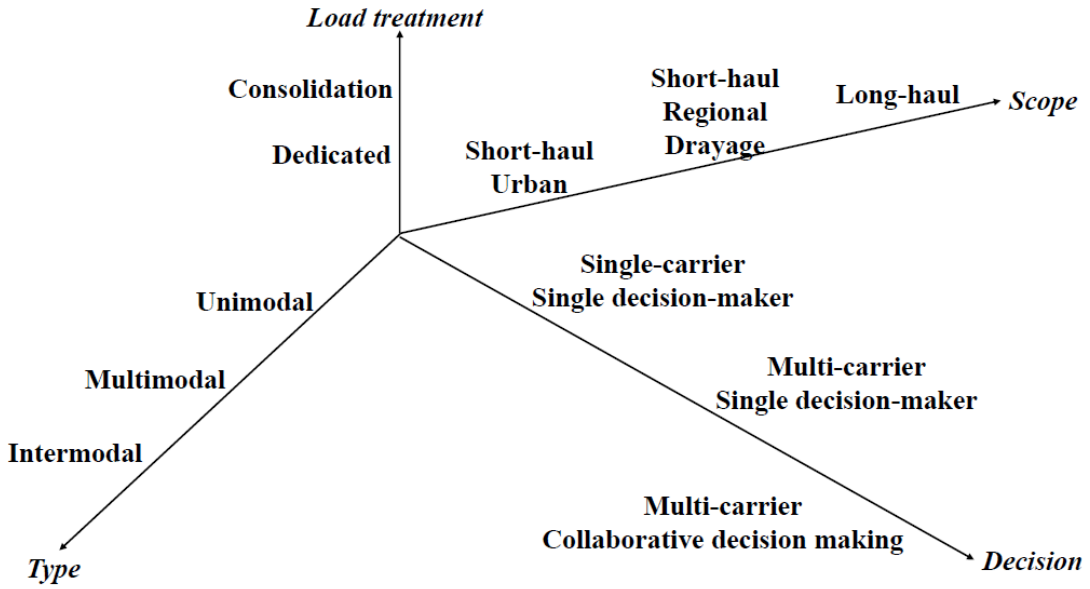


Figure 2- High-level classification of freight carriers (Crainic & Rei, 50 Years of Operations Research for Planning Consolidation-based Freight Transportation, 2024)

The first dimension, *Load treatment*, indicates both the commercial shipper-carrier relation regarding how the carrier capacity is allocated to the shipper request, and the subsequent loading and movement activities. *Dedicated* refers to the case when a loading unit is dedicated to a unique shipper demand, which pays for the complete journey. Once loaded, the corresponding shipment travels untouched until its final destination.

Consolidation is the process of combining multiple shipments from different shippers, potentially with varying origins and destinations, into the same vehicle or container for part or all of their journey. On the shipper's side, the volume or value of most shipments is too small to justify the costs of direct, dedicated transport. On the carrier's side, these same shipment characteristics make it unfeasible to provide a profitable direct service with acceptable service quality. Furthermore, with consolidation, the carriers can use their resources more efficiently (e.g., fill up their loading units with multiple shipper requests) to meet the overall demand. Consolidation serves here as a general logistic strategy that benefits both shippers and carriers.

Then, the second dimension is the *scope,* which differentiates between *long-haul* and *short-haul* transportation.

The *Type* dimension refers to the *mode* of transportation used by the carrier or multi-stakeholder system. It is worth mentioning that, "mode" is a general term, which may refer to different concepts/definitions according to the topic at hand. Thus, for example, a very general definition refers to fundamental "elements of nature", that is, land, water, air, and space transport. At the other end of the precision spectrum, a mode may refer to a particular combination of infrastructure, motorization, and even ownership. The widely accepted, "classical" definition of modes is based on a high-level combination of elements such as nature, infrastructure, and vehicle/traction and can be classified in *Full-Truckload (TL)* and *Less-than-truckload (LTL)*, *Rail*, *Maritime Navigation, River & canal navigation*, *Air* and *Pipeline* for liquids (not a consolidation mode, hence not discussed). *Unimodal*, or single-mode transportation is then defined when a single mode is used from the origin to the destination of the cargo journey. Many journeys are unimodal and they are performed mostly using trucks. *Intermodal* transportation is generally defined as a chain of transportation services, most often of different modes but not necessarily moving cargo packaged in such way that it is not touched when transferred.

Finally, the *decision* dimension reflects how the decision process is structured within the organization, or the group of organizations, supplying the services. Most of these cases may be described as (more or less) integrated *Many-to-One-to-Many (M1M)* systems. An M1M system involves shippers on one side, making shipper demand requests for cost and time-efficient transportation for their loads, and carriers on the other side, which make carrier-capacity offers for transportation and warehousing space, while requesting profitable loads. The "One" decision maker in the middle, named as *Intelligent Decision Support Platform (IDSP)* plans and optimizes operations and resource-utilization to profitably and simultaneously satisfy the needs of both shippers and carriers, as illustrated in Figure 3.

Shipper-demand requests and the carrier-capacity offers are made available to the system at different time periods. Hence, the IDSP receives time dependent requests from both stakeholders and optimizes in time and space the selection of shipper-demand requests, carrier-capacity offers, shipment-to-carrier assignments, and shipment itineraries through the consolidation of loads of different shippers into the same vehicles and synchronization of activities. The centralized decision maker can

be an intermediary, or a carrier. In the first case, the decision maker does not completely own the infrastructure nor the resources (including transport means), but uses a number of shared multi-carrier resources (trucks, Vans, cargo-bikes, vessels, planes, barge trains) from different companies, as common in the synchromodal networks, to manage the freight transportation system. If the centralized decision maker is the carrier, the decision problem is which services to activate and how to efficiently and profitably use the self-owned and managed resources. In this case, the decision to take concerns the number of loading units like trailers, railcars/wagons, containers to use.

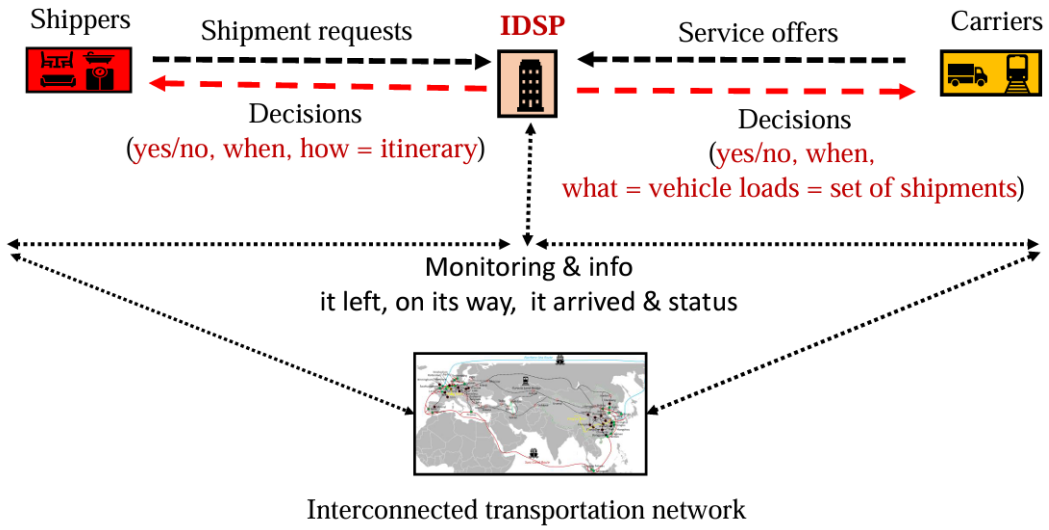


Figure 3 - M1M system structure, stakeholders, communications & decisions (Crainic & Rei, 50 Years of Operations Research for Planning Consolidation-based Freight Transportation, 2024)

A *service* in the latter sense is characterized by origin and destination terminals, without intermediate stops. It has operational and economic attributes, including type of vehicle and traction, speed, capacity, costs and frequency or schedule. Both material, e.g., vehicles and traction (power) units, and human, crews within terminals and operating vehicles and convoys, are required to support the operations of the planned services. In most cases, the customer service does not correspond to a unique transportation service. Indeed, the demand volume, value, or both, are not sufficiently high to justify an economically and service-quality efficient direct and rapid shipment, neither for the carrier, nor for the shipper. Hence, carriers will move demand through so-called *hub-and- spoke* physical and service networks, which

provide the means to take advantage of the economies of scale of consolidation-based transportation.

Thus, two major types of terminals are encountered in such networks. *Local/regional terminals* make up the largest set, they both are used to aggregate the demand in a restricted region (the beginning of the process) and to bring the goods in the local terminals to separate and deliver them to their consignees (the end of the process).

The *hubs* make up the second category and form the core of consolidation based transportation through the consolidation/de-consolidation of the flows in and out of their associated regional terminals for efficient long-haul transportation and economies of scale.

Carriers thus first move low-volume loads available at a regional terminal to a hub, through what is known as feeder services. At hubs, loads are sorted (classified is the term used in a number of settings, freight railroads in particular) and consolidated into larger flows, which are routed to other hubs by high-frequency, high-capacity services. Loads may go through more than one intermediary hub before reaching the regional-terminal destination, being transferred from one service to another or undergoing reclassification and re-consolidation. Notice that, more than one service of possibly different modes, may be operated between consolidation and regional terminals.

The *cost* associated to the use of a particular service has three main components. The first is a fixed service-activation cost, which is incurred to set up the service, the second is a fixed cost associated with the use of a loading unit and generally is determined by its type, and the third is the variable transportation cost which depends on the freight quantity (volume) transported and to the specific loading unit type. These costs may also be influenced by environment-related concerns. For instance, more environment-friendly transportation modes, such as the cargo bikes, electric vehicles may receive incentives, in the form of privileged access and charging tariffs in some parts of the city. Costs are also associated to terminal activities and generally depend on the layout and physical and operative organization of a terminal. Shippers pay a tariff for the transportation of their goods, which depends on the commercial agreements in force.

Regarding the last mentioned agreements, it is possible to distinguish between two types of customers (shippers).

Regular customers often have long-term contracts offering discounts on regularity and volume, and it is compulsory to accept such requests, while the so-called *non-contract (or non-regular) customers* correspond to shippers without formal contracts requesting a transportation service not on a regular basis. The decision to satisfy the demand in this case is in charge of the intermediary, which performs customer selection on the basis of the associated profitability. While generally not accounted for in tactical-planning models, such categories are particularly important in M1M systems, that can benefit of servicing non-contract customers when building the transportation plan. The aim is to design the demand (i.e., shippers) and supply (i.e., carriers) sides of this integrated system simultaneously and plan for them in the most efficient way. Decisions on acceptance or rejection of non-contract shipper requests and carrier service offers should be taken with the main aim of satisfying both categories of stakeholders, through consolidation in time and space of multi-shipper requests. As mentioned before, the intermediary determines the shipper-demand packages itineraries, defined as the scheduled sequence of services from their origin to their destination and the time spent in terminals to wait for the next service, considering the storage cost in each terminal.

4.1 Planning activities levels

Service Network Design is an OR methodology belonging to one of the three levels of the planning activities consolidation-based carriers (Crainic & Rei, 50 Years of Operations Research for Planning Consolidation-based Freight Transportation, 2024). Figure 4 provides a schematic representation of these planning levels, showing their respective planning horizons, scope, interconnections, and associated core Operational Research methodologies.

Before expanding on this topic, it must be distinguished between planning and execution. Planning is the preparation of operations in anticipation of a future situation, whereas execution is the acting, including updating/adjusting, of the pre-established plans.

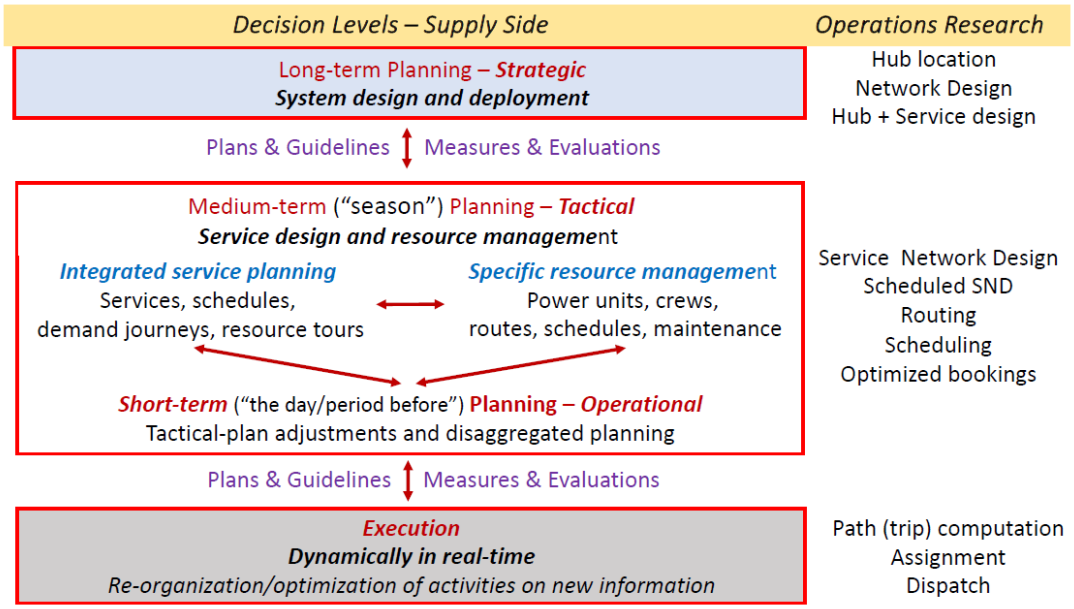


Figure 4-Decision & planning levels and related O.R. methodology (Crainic & Rei, 50 Years of Operations Research for Planning Consolidation-based Freight Transportation, 2024)

Strategic planning involves long-term decisions related to market deployment, system design, operational strategies, and the acquisition of key resources. It focuses on extended planning horizons and involves high-level management. In contrast, *tactical* and *operational* planning consist of medium to short-term planning and are the ones which garnered the most attention and generated the majority of O.R. contributions.

Tactical planning is typically conducted by mid- to high-level management over a medium-term horizon, often referred to as a "season" in the literature, which can span anywhere from a few weeks to six months or more. The objective of tactical planning is to develop a transportation plan and schedule for services and the resources needed to carry them out. It aims to counterbalance the potential drawbacks of consolidation, such as increased delays and costs at terminals, while ensuring the satisfaction of regular customer demand, service-quality requirements, and profitable, efficient operations.

The duration of the tactical plan does not coincide with the planning horizon, but is much shorter, determined by the repetition pattern of regular demand. The activities planned for this program's duration are then executed repeatedly for the entire duration of the tactical planning horizon.

Tactical models and methods can also be employed as tools to evaluate policies and performance in strategic scenarios, either by optimizing operational details, demand, and costs at a higher level of abstraction, or by simulating scenarios with simplified carrier and shipper characteristics.

Operational planning may be carried out by the same managers responsible for tactical plans, but it generally falls under the purview of decision-makers closer to day-to-day operations. The objective is to execute the plan based on observed conditions, such as actual demand (whether it matches expectations or not) and service operations (e.g., mechanical issues, delays). Adjustments can be made iteratively and frequently, such as weekly or even daily. While tactical and operational planning are often treated separately in surveys and articles, they are grouped together to highlight that the same methodologies can often be applied to both, though with varying levels of detail and time frames (Ricciardi, Storchi, & Crainic, 2009)

The final class of decision-making structure focuses on real-time management and the execution of operations in a dynamic environment, ideally adhering to the pre-established plan.

5 Service Network Design Issues: integration with Bin Packing Problem

Tactical planning for consolidation-based carriers is a complex problem, usually addressed by the Service Network Design (SND) methodology. However, the SND literature typically ignores some important issues, such as how cargo is loaded into storage facilities or transportation vehicles (referred to as capacity units in the following) and how to choose the right number and type of capacity units for each selected service within the network design. These factors could significantly affect the system's performance, so they should be carefully considered to prevent underestimating expenses or creating unfeasible itineraries that go above the capacity of the chosen service's capacity units (Bruni, Crainic, & Perboli, 2023).

Therefore, Bin Packing dimensions should be explicitly considered when solving SND problems. Such dimensions not only directly bring more operational considerations in the tactical planning settings performed by carriers, but they also lead towards the study of new discrete optimization model variants, which seek to design networks while imposing bin packing requirements on the network's flow. There are very few papers in the SND literature explicitly integrating packing constraints into SND models. Hewitt and Lehuédé (Hewitt & Lehuédé, 2023) present new formulations for the Scheduled SND problem, where shipments cannot be split, while consolidation is desirable to reduce the number of homogeneous vehicles used when multiple shipments dispatch simultaneously on the same direct service. In this case, SND and Bin Packing are not addressed simultaneously. The packing and the SND-related decisions may then be addressed separately. Flamand, Iori and Haouari (Flamand, Iori, & Haouari, 2023) combine two classical problems, transportation and variable size and cost bin packing, aiming to ship multiple types of commodities on different types of vehicles moving between the supply and demand nodes of a bipartite network, while minimizing the transportation and resource-acquisition cost.

Only "The value of integrating loading and routing" (Côté, Guastaroba, & Speranza, 2017) addressed the issue of quantifying the potential benefit deriving from tackling routing and packing problems in a fully integrated approach. The authors compared

the solutions of their integrated approach with non-integrated solution methods, i.e., solution methods where the routing and packing sub-problems are addressed separately, often one after the other. The results showed how the integrated approach is much more computationally expensive, but gives better solutions, both in terms of total cost, number of vehicles routed and load factors achieved.

Combining two NP-hard combinatorial problems, Service Network Design and Bin Packing, does not make the problem easier to address and requires careful investigation.

Before proposing a model integrating Service Network Design with Bin Packing, however, it is necessary to expose the bin packing definition. In the following section, a classical formulation for the bin packing problem will be exposed.

5.1 The Bin Packing Problem

When the Service Network Design is combined with Bin Packing considerations, the problem extends beyond just deciding routes and schedules. Considering Service Network Design (SND) with Bin Packing problem (BPP) considerations refers to a complex optimization challenge that combines two distinct but related problems: designing an efficient service network while also ensuring that resources (such as vehicle capacities) are utilized optimally, as in the bin packing problem.
Let's start, however, by first exposing the classic Bin Packing problem.

The Bin Packing problem is an optimization problem, in which items of different sizes must be packed into a finite number of bins or containers, each of a fixed given capacity, in a way that minimizes the number of bins used (Wikipedia Contributors, 2024).

Given a set *I* of n items with a size *s(i)* for each *i*, an integer non-negative bin capacity *B*, and a positive integer *K*, find a partition of *I* into disjoint sets I,..., I_K such the sum of the sizes of the items in each I_j is *B* or less. The aim is to minimize the number of bins used, thus research is interested in the smallest possible value of *K*.

The mathematical formulation of the problem is:

$$Minimize\ K = \sum_{j=1}^{n} y_j$$

Subject to

$\sum_{i \in I} s(i)x_{ij} \leq By_j, \forall\, j \in \{1, \ldots, n\}$

$\sum_{j=1}^{n} x_{ij} = 1, \forall\, i \in I$

$K \geq 1$

$y_j \in \{0,1\}, \forall\, j \in \{1, \ldots, n\}$

$x_{ij} \in \{0,1\}, \forall\, i \in I, \forall\, j \in \{1, \ldots, n\}$

Where y_j is equal to 1 if bin *j* is used and x_{ij} is equal to 1 if item *i* is put into bin *j*.

Both problems involve minimizing costs, whether it's the cost of services used (as in SND) or minimizing the number of containers (bins) used (as in BPP).

In SND+BPP, the aim is to design a service network that minimizes both the operational costs of running services and the cost of inefficient packing (underutilized capacity).

From a mathematical point of view, combining SND and BPP typically results in a mixed-integer programming (MIP) problem, where:

- The SND part focuses on determining which routes to establish and which services to operate (binary decision variables).
- The BPP part involves deciding how to pack demands (items) into available capacities (bins) such that capacity constraints are met.

Below, the mathematical formulation for the SND model integrated with Bin Packing will be presented. Note that the features of the bin packing correspond to the third term of the Objective Function (1) and constraint (3). In fact, the third term of the Objective Function (1) aims to minimize the total cost associated with the use of bins, which is somewhat synonymous with minimizing the number of bins used, as their use clearly incurs a cost. Meanwhile, constraint (3) ensures that the capacity of each bin is never exceeded.

5.2 SND with Bin Packing considerations

The model described below is a unified problem setting, where decisions concern the design of service and the construction of the demand itineraries within the selected service network, the selection of the type and number of transportation units to load and move the origin-destination demands, the selection and assignment of demands to the loading units. This means that just by some small simplifications, the present problem setting is also valid for the classical network design.

Decisions will now concern the design of service and the construction of the demand itineraries within the selected service network, the selection of the type and number of transportation units to load and move the origin-destination demands, the selection and assignment of demands to the loading units.

Let's consider the physical network $G^{PH} = (N^{PH}, A^{PH})$, where N^{PH} is the set of nodes (representing transfer and consolidation terminals, which include the origins and destinations of demand), and A^{PH} is the set of physical or conceptual links joining these nodes.

Each potential service $\sigma \in \Sigma$ is identified by its origin $o(\sigma)$, destination $d(\sigma)$, service departure time $\alpha(\sigma)$, and arrival time $\beta(\sigma)$, and fixed service cost (f_σ). Each service σ operates using a set of existing resources (or loading units) (J_σ) already assigned to it that could differ in terms of type, numbers, operating cost, capacity, etc. Let $J = \cup_{\sigma\in\Sigma} J_\sigma$, $J_\sigma \cap J_{\sigma'} = \emptyset$, $\forall \sigma \neq \sigma' \in \Sigma$ be the set of loading units available in the service network.

In what follows, the generic term of *bin* will be used to refer to the loading unit associated to services. The loading units are service-specific in the sense that a limited number of loading units of each type is available for each service, and not all types of loading units can operate over the network. For instance, vessels can not be used in roads, cargo bikes can not travel on highways etc. Often physical characteristics of the network or define additional constraints on the service capacity e.g., number of vessels that may simultaneously navigate a river during a given period of time, number of trucks that may simultaneously traverse a bridge. The set of bin types is denoted by $\Pi = \{1, 2, \cdots, \pi, \cdots n_\Pi\}$ and each bin $j \in J$ has a type $\phi(j) \in \Pi$) that is characterized by capacity Q_π, fixed usage cost c_π^F and a variable cost c_π^a

representing transportation, environmental cost). Each service is thus characterized by the attributes, characteristics and capacity. Each service σ has a global capacity Uσ, that is defined on the basis of the characteristics of the system considered. For instance, it can be equal to the maximum capacity of the loading units allocated to that service. Depending on the application, the global capacity can be interpreted differently and not necessarily as the maximum capacity of loading units of that service; for example, in North-American railroad, two standard types of rail cars (loading bins) of 40 or 53 feet are available. Regardless the number of railcars available on the service, the maximum length of 400 feet for the train imposes a global capacity. One could also impose other capacity restrictions in terms of the maximum number of bin types allowed to operate simultaneously on the service. On the demand side, there are two set of contract (K^C) and non-contract (K^{NC}) demands where each shipper-demand request $k \in K = K^C \cup K^{NC}$ calls for the transportation of a set of packages (referred to items, for analogy with the bin packing problem and denoted by $I(k) = \{1, 2, \cdots, i, \cdots, n_k\}$) from its origin $o(k)$ to its destination $d(k)$. Each item is characterized by a volume v_i (expressed in the same unit as the bin capacity) and the size of the demand is the sum of the volume of its items, denoted by $d_i = \Sigma_{i \in I(k)} v_i$.

All items belonging to the same demand are available at the same time (availability time $\alpha(k)$) at the origin $o(k)$, and need to be delivered to the final destination $d(k)$ not later than a common due-date $\beta(k)$. Depending on the application and context, some demands can have specific requirements in terms of bin types that can be used for moving them. To reflect this feature, a set of compatible bin types $\Pi_k \subset \Pi$ for demand k is defined. In all cases, items arriving at a terminal different from their destination are unloaded from the bins of the preceding service, eventually held at the terminal for a while incurring a holding cost per unit h_k, and then loaded into different bins associated with different departing services, continuing their journey to the final destination.

In general, due to the capacity restrictions (the case where the total demand volume is greater than the maximum capacity of service) or to allow a better consolidation of demands, items belonging to the same demand can be loaded into different services following separate itineraries on the network. This resembles the classical

demand splitting policy in freight transportation. In case the demand splitting is prohibited, such as *Hazardous materials transportation*, all items are integrated into one single item, called *mega item*, a unit comprising all items of the same demand that is transported as a whole, ensuring that all items share the same itinerary. Clearly, the volume of the mega item is the summation of the volume of items of that demand (d_i).

Each potential request (belonging to the non-contract customers) could be accepted or rejected on the basis of its revenue p_k. Clearly, accepting a demand request requires the delivery of all its items $i \in I(k)$. On the contrary, the requests of contract customers are compulsory to get accepted. representing transportation, environmental cost).

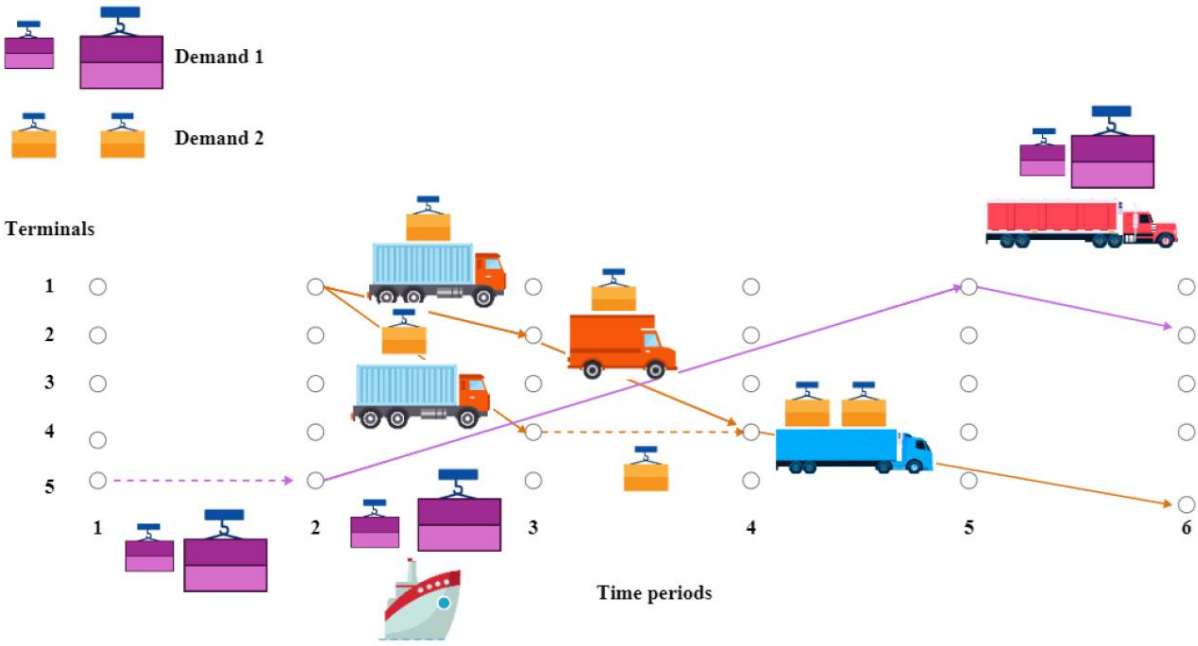


Figure 5-Physical network example

Figure 5 presents an example of a physical network comprising five nodes (terminals) over a schedule spanning six periods, with two distinct demands (illustrated in yellow and violet). Each demand consists of two items with varying volumes. Demand 1 is available at terminal five during the first period and must be delivered to terminal two by the end of period six. Demand 2 begins at terminal one in the second

period and needs to reach terminal five by the sixth period. The routes for each item are indicated using the same color as their respective demand. Dashed arrows illustrate holding actions at terminals, while solid arrows represent the transport of demands through active (open) services. As shown, the items for Demand 2 are divided and follow separate paths. In contrast, all items for Demand 1 follow a single route and share the same loading unit.

To model the temporal attributes of the service, a time-space network will be defined, G = (N , A), built by extending the physical network $G^{P\ H}$ along the dimension of time for the fixed duration of the schedule length, discretized into *time periods t* ∈ T of equal length. Operations at terminals in different periods are modeled with different nodes of the form (*n, t*) ∈ N. There are two types of arcs in A. A *service arc*, joining nodes (*n, t*) and ($n^{'}$, $t^{'}$), models the operation of a single-leg service $\sigma \in \Sigma$ between its origin $o(\sigma) = n$ and destination $d(\sigma) = n^{'}$, starting at time $\alpha(\sigma) = t$ and arriving at time $\beta(\sigma) = t^{'}$. A *holding arc*, joining nodes (*n, t*) and (*n, t* + 1), models the possibility of holding items at node *n* from period *t* to *t* + 1. A^{Σ} and A^{H} stand for the sets of service and holding arcs, respectively, with $A = A^{\Sigma} \cup A^{H}$.

The following sets of decisions variables are going to be used:

$y_\sigma \in \{0,1\}, \sigma \in \Sigma$ for the selection of service σ

$z_j \in \{0,1\}, j \in J$ for the selection or not of bin j

$x_{aj}^{i} \in \{0,1\}$ represents the possible assignment of item $i \in I(k)$ to bin j of service σ and it is defined for given demand $k \epsilon K, \forall a \in A^{\Sigma}, \forall i \in I(k), \forall j \in J_\alpha, \varphi(i,j) \in \Pi_{i,k}$

$w_a^i \in \{0,1\}, a \in A^H, i \in I$ indicates if item i is held on arc a

$\lambda_k \in \{0,1\}, k \in K$, whether or not demand k is accepted

5.3 The mathematical model

(1) Maximize $\sum_{k\in\mathcal{K}} p_k\lambda_k - \big(\sum_{\sigma\in\Sigma} f_\sigma y_\sigma + \sum_{j\in J} c^F_{\phi(j)} z_j + \sum_{a\in A^H}\sum_{k\in\mathcal{K}} h_k\big(\sum_{i\in I(k)} w^i_a\big) + \sum_{a\in A^\Sigma}\sum_{i\in I}\sum_{j\in J} c^a_{\phi(j)} v_i x^i_{aj}\big)$

Subject to

(2) $\sum_{a\in A^+_{(n,t)}}\sum_{j\in J_{\sigma_a}} x^i_{aj} + \sum_{a\in A^+_{(n,t)}} w^i_a - \Big(\sum_{a\in A^-_{(n,t)}}\sum_{j\in J_{\sigma_a}} x^i_{aj} + \sum_{a\in A^-_{(n,t)}} w^i_a\Big) =$

$$\begin{cases} \lambda_k, & \text{if } (n,t) = \big(o(k), \alpha(k)\big), \\ -\lambda_k, & \text{if } (n,t) = \big(d(k), \beta(k)\big), \\ 0, & \text{otherwise,} \end{cases} \quad \forall (n,t)\in\mathcal{N}, \forall k\in\mathcal{K}, \forall i\in I(k)$$

(3) $\sum_{i\in I} v^a_i x_{aj} \le Q\big(\phi(j)\big) z_j, \quad \forall a\in A^\Sigma, \quad \forall j\in J_{\sigma_a}$

(4) $\sum_{j\in J_\sigma} Q\big(\phi(j)\big) z_j \le U_\sigma y_\sigma, \quad \forall a\in A^\Sigma$

(5) $\sum_{j\in J_\sigma} z_j \le |J_\sigma| y_\sigma, \quad \forall \sigma\in\Sigma$

(6) $\lambda_k = 1, \quad \forall k\in K$

(7) $y_\sigma \in \{0,1\}, \quad \forall \sigma\in\Sigma$

(8) $z_j \in \{0,1\}, \quad \forall j\in J$

(9) $x_{aj}^{i} \in \{0,1\}, \quad \forall k \in K, \quad \forall a \in A^{\Sigma}, \quad \forall i \in I(k), \quad j \in J_{\sigma_a}\big(\phi(i)\big) \in \Pi_k$

(10) $w_a^i \in \{0,1\}, \quad \forall a \in A^H, \quad \forall i \in I$

(11) $\lambda_k \in \{0,1\}, \quad \forall k \in K$

Where $\mathcal{A}^{+}_{(n,t)} = \left\{ a = \Big((n'', t''), (n', t') \Big) \Big) \in \mathcal{A} \;\middle|\; n'' = n,\, t'' = t \right\}$ and $\mathcal{A}^{-}_{(n,t)} = \left\{ a = \Big((n', t'), (n'', t'') \Big) \Big) \in \mathcal{A} \;\middle|\; n' = n,\, t' = t \right\}$, for each $(n,t) \in \mathcal{N}$.

The Objective Function (1) maximizes the total profit expressed as a difference between revenues and costs. The total cost of selecting services, securing bins, holding items at terminals, transporting items through the designed system is considered. Constraints (2) ensure that each item is routed from its origin node to its destination node, respecting the temporal constraints. Constraints (3) enforce a feasible assignment of items to bins, respecting the bin capacity. Constraints (4) express the service maximum capacity in terms of the total capacity of bins operating on that service. Constraints (5) link the z and y variables ensuring that only if a service is opened, then the bins of that service can be used.

Constraints (6) require that the demand of all regular customers are totally accepted and delivered. Finally, constraints (7)-(11) express the nature of the variables.

In the classical SND formulation, there is a constraint that ensures that all demands are satisfied. Unlike the classical SND formulation, note how in this formulation does not exist a constraint that ensures that all questions are satisfied, but rather only some will be selected.

6 The model on AIMMS

The mission of this chapter is to transform the mathematical model presented above into a model that can be processed on the AIMMS software. Therefore, it is required to declare in the form of code what are the sets, parameters, variables, constraints and the objective function. To do this, it is necessary to keep an eye on the logic of the coding and the correlations between all the elements.

6.1 AIMMS Software

For the computation of the model described above, a software named AIMMS will be used. The AIMMS (acronym for Advanced Interactive Multidimensional Modeling System) was founded in 1989 by mathematician Johannes Bisschop and nowadays is considered to be one of the five most important algebraic modeling languages (Wikipedia Contributors, AIMMS, 2024).

Its Prescriptive Analytics Platform consists of an algebraic modeling language, an integrated development environment for both editing models and creating a graphical user interface around these models, and a graphical end-user environment. AIMMS is linked to multiple solvers through the AIMMS Open Solver Interface. Supported solvers include CPLEX, MOSEK, FICO Xpress, CBC, Conopt, MINOS, IPOPT, SNOPT, KNITRO and CP Optimizer.

AIMMS features a mixture of declarative and imperative programming styles. Formulation of optimization models takes place through declarative language elements such as sets and indices, as well as scalar and multidimensional parameters, variables and constraints, which are common to all algebraic modeling languages, and allow for a concise description of most problems in the domain of mathematical optimization. Units of measurement are natively supported in the language, and compile- and runtime unit analysis may be employed to detect modeling errors.

Procedures and control flow statements are available in AIMMS for the exchange of data with external data sources such as spreadsheets, databases, XML and text files data pre- and post-processing tasks around optimization models

user interface event handling the construction of hybrid algorithms for problem types for which no direct efficient solvers are available.

To support the re-use of common modeling components, AIMMS allows modelers to organize their model in user model libraries.

AIMMS supports a wide range of mathematical optimization problem types: Linear programming, Quadratic programming, Nonlinear programming, Mixed-integer programming, Mixed-integer nonlinear programming, Global optimization, Complementarity problems (MPECs), Stochastic programming, Robust optimization.

6.2 Sets

A simple set in AIMMS is a finite collection of elements. These elements are either strings or integers. Strings are typically used to identify real-world objects such as products, locations, persons, etc. Integers are typically used for algorithmic purposes. Every simple set can associate indices through which the user can refer (in succession) to all individual elements of that set in indexed statements and expressions (Set Declaration, 2024).

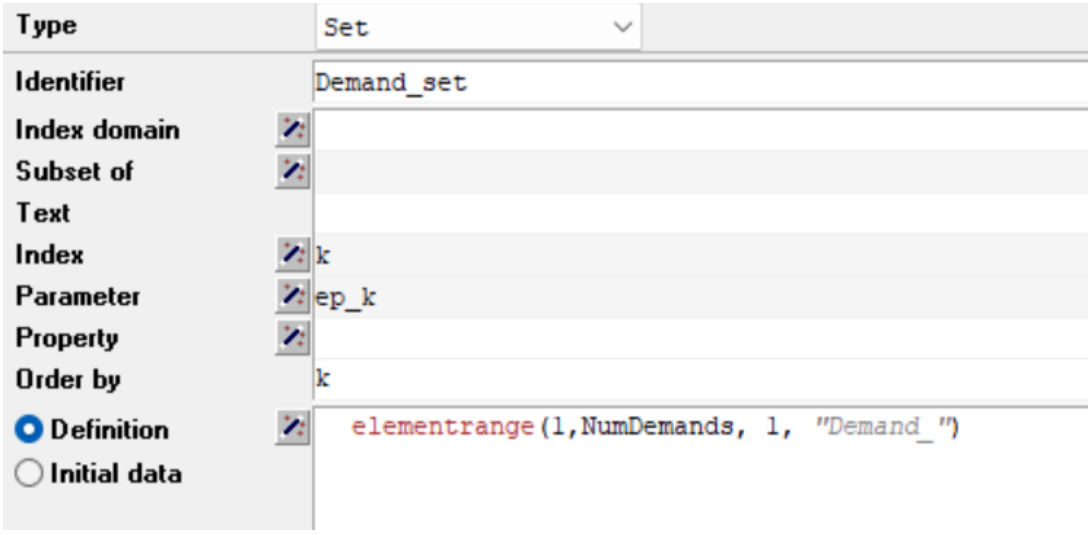
Each set has an optional list of attributes which further specify its intended behavior in the model. The main attributes of a Set are IndexDomain, SubsetOf, Index, Parameter, Text and Definition.

It's important to underline that sets can have an IndexDomain, but not for this they should be confused with parameters. AIMMS allows for multi-dimensional sets, which are essentially sets parameterized over other sets, AIMMS allows for multi-dimensional sets, which are essentially sets parameterized over other sets. For instance, by specifying an index domain (*n, np, t, tp*) means that each element of a set is related to a specific combination of values for n, np, t, and tp. The distinction between an indexed set (or set with an index domain) and a parameter in AIMMS can indeed be subtle, especially because both can involve multiple dimensions and similar syntax. While a set is a collection of elements, a parameter stores numerical values or data.

Perplexity could also arise about the attribute named "parameter" of a set. This attribute indicates that there is a parameter associated to each element of the set. It is typically a property or attribute of the set's elements.

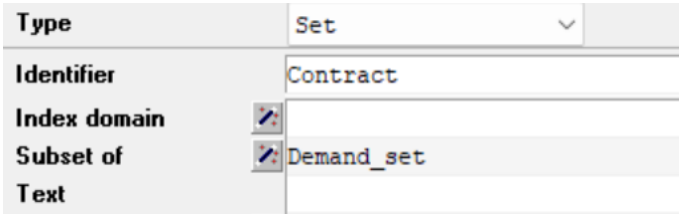
The AIMMS model sets are those in the following captures, with their respective explanations when it is required.

Capture 1- Demand_set



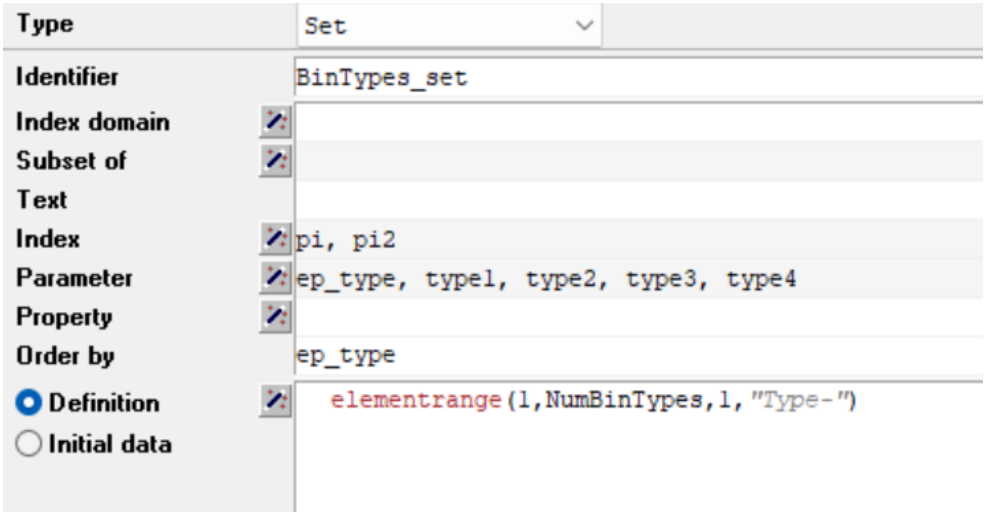
Comment on *Demand_set*: The index *k* is defined for this set, which will be used to refer to individual elements (instances of demand) when the set is referenced in constraints, expressions, or other calculations. There's a parameter *ep_k,* which is probably associated with elements of the set. The set is ordered by the index *k*. This is a function to generate a range of elements for the set based on the number of demands (*NumDemands*). *Elementrange(1, NumDemands, 1, "Demand_")* means that the elements of the set start from 1 and go up to Numdemands: this indicates the total number of elements (Demands). The third component inside the parenthesis means that each element in the range is incremented by 1, while the fourth establishes the prefix of each element, which in this case must be "Demand_". For instance, "Demand_1", "Demand_2" etc.

Capture 2- Set



Comment on *Contract*: this is a subset of Demand_set, indicating that among all the demands, there are regular customers which have a long-term contract and others who do not.

Capture 3- BinTypes_set



Comment on *BinTypes_set*: this is the set containing all the types of bins used. The indices *pi, pi2* are likely used to refer to elements of the set in constraints or other expressions. The set is ordered by the parameter *ep_type*, so the elements in BinTypes_set will be ordered based on this parameter. As for the set *Demand_set* seen before, the number of bin types is determined by the parameter *NumBinTypes*, and the elements are named automatically as "Type-1", "Type-2", and so on, according to the elementrange function. Associated to each element of the set are the parameters *ep_type*, *type1, type2, type3, type4*.

6.3 Parameters

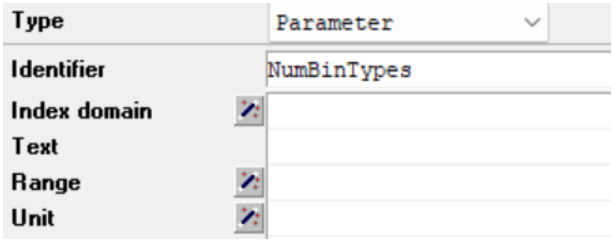
In AIMMS the word parameter denotes a known quantity that holds either numeric or string-valued data (Parameters Declaration, 2024). In programming languages the term variable is used for this purpose. However, this is not the convention adopted in AIMMS, where, in the context of a mathematical program, the word variable is reserved for an unknown quantity. Outside this context, a variable behaves as if it were a parameter. The terminology in AIMMS is consistent with the standard operations research terminology that distinguishes between parameters and variables. Rather than putting the explicit data values directly into the expressions, it is a much better practice to group these values together in parameters and to write all the expressions using these symbolic parameters. Maintaining a model that contains explicit data is a painstaking task and error prone, because the meaning of each separate number is not clear. Maintaining a model in symbolic form, however,

is much easier and frequently boils down to simply adjusting the data of a few clearly named parameters at a single point.

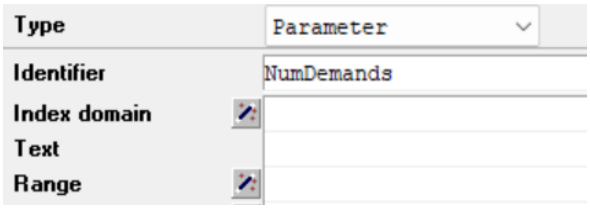
The main attributes of a parameter are IndexDomain, Text, Range, Unit, Default, Property and Definition.

The AIMMS model parameters are those in the following captures, with their respective explanations when it is required.

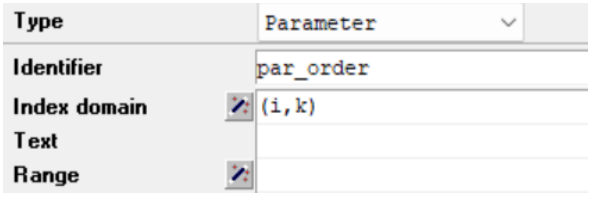
Capture 4-NumBinTypes



Capture 5- NumDemands

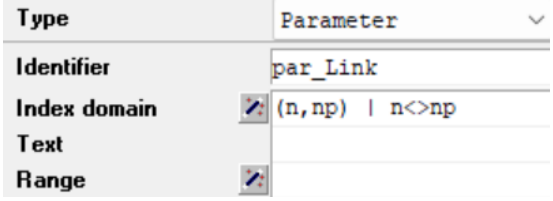


Capture 6- par_order



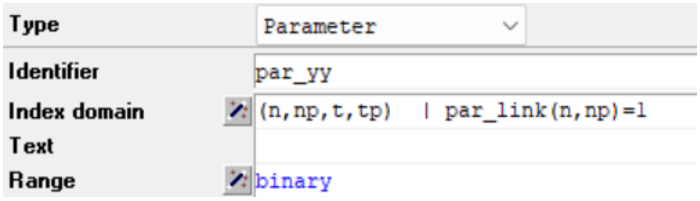
Comment on *par_order*: it suggests what items each demand is composed of. For instance, Demand-01 requires the transportation of item 1, item 3 and item 11.

Capture 7- par_Link



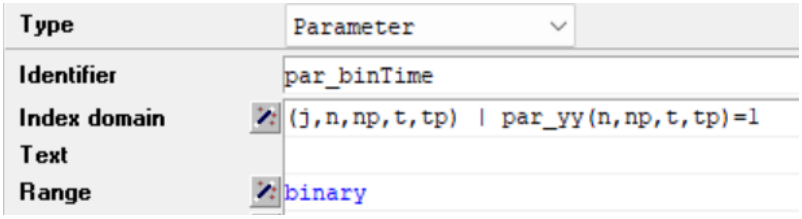
Comment on *par_Link:* it expresses if there is a link (connection) between the node *n* and the node *np*.

Capture 8- par_yy



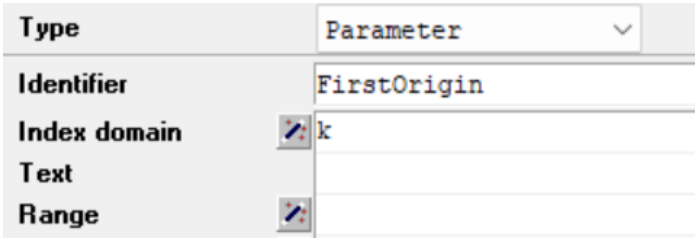
Comment on *par_yy*: for an existing connection between the node n and the node np, the parameter *par_yy=1* indicates that there is the possibility of choosing a service departing from terminal n at time t and arriving at terminal np at time tp. So the duration of service is tp-t. For instance, a service could be available from node 1 to node 4 starting at *t*=3 and ending at *t*=8.

Capture 9- par_binTime



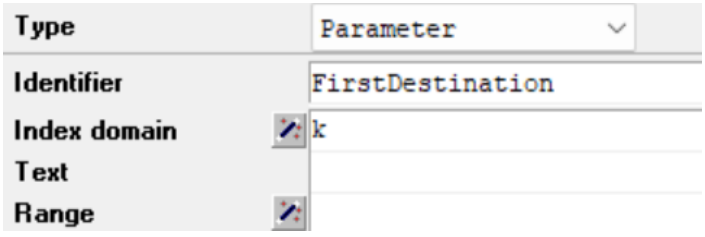
Comment on *par_binTime*: when this parameter is equal to 1, it means that bin *j* can operate the service defined between the two terminals *n* and *np* from *t* to *tp*.

Capture 10- FirstOrigin



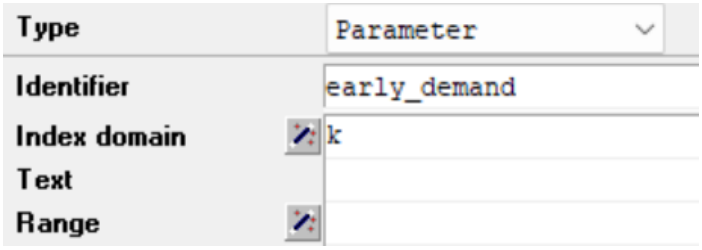
Comment on *FirstOrigin*: it suggests the departure terminal of each demand.

Capture 11- FirstDestination



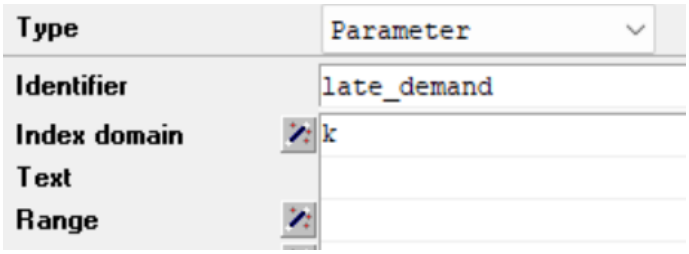
Comment on *FirstDestination*: it suggests the destination terminal of each demand.

Capture 12- early_demand



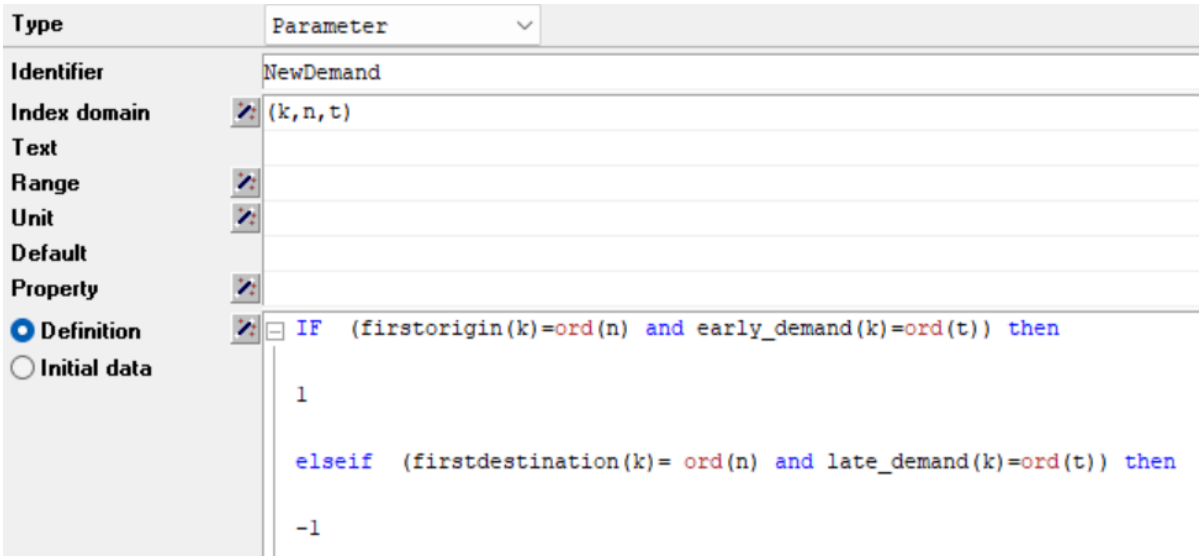
Comment on *early_demand*: it suggests the availability time of the demand.

Capture 13- late_demand



Comment on *late_demand*: it suggests the due-date of the demand (the items of the demand need to be delivered to the final destination not later than a common due-date).

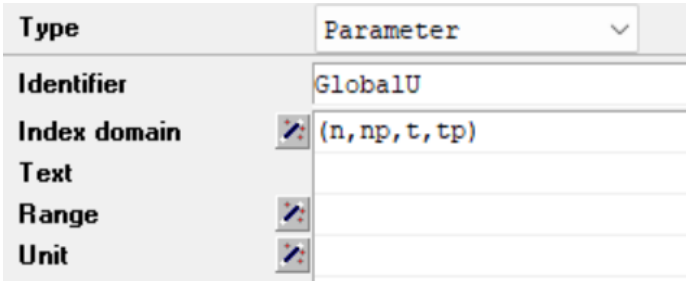
Capture 14- NewDemand



Comment on *NewDemand*: this parameter will be useful for the first constraint of the model. Considering the demand *k*, a node *n* and a time *t*, the parameter takes the value of 1 if the node *n* corresponds to the origin of *k* and the period *t* corresponds to the availability time of *k*, takes the value of -1 if the node *n* corresponds

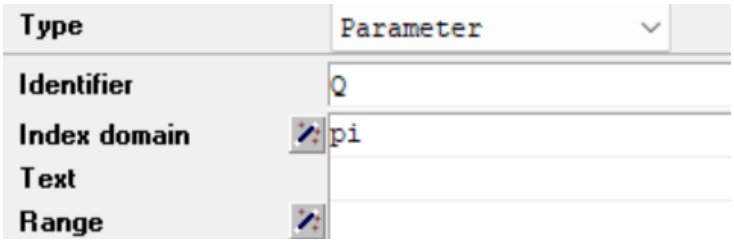
to the final destination and the period *t* corresponds to the due-date, otherwise it takes the value of 0.

Capture 15- GlobalU



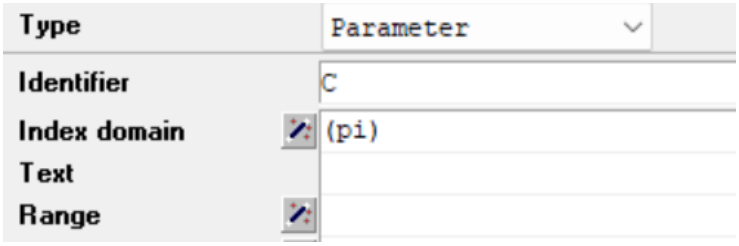
Comment on *GlobalU*: it is the maximum capacity of the service.

Capture 16- Q



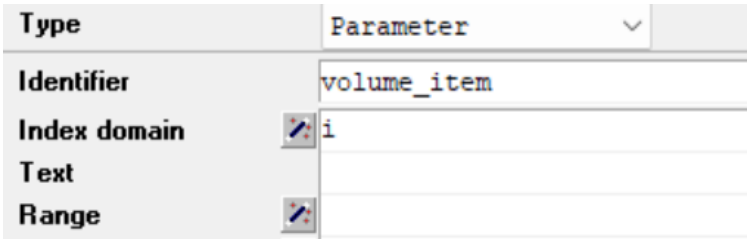
Comment on *Q*: it is the capacity for each bin type.

Capture 17- C

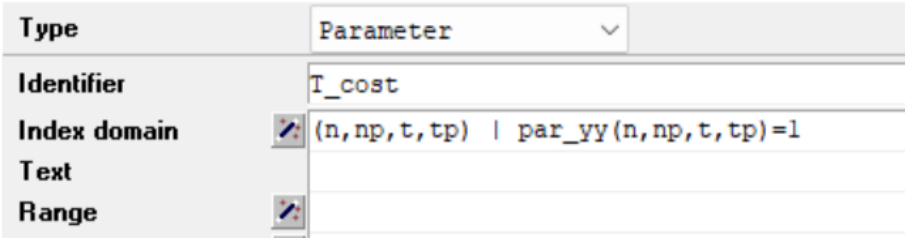


Comment on *C:* it is the fixed cost associated to each bin type when it is used.

Capture 18- volume_item

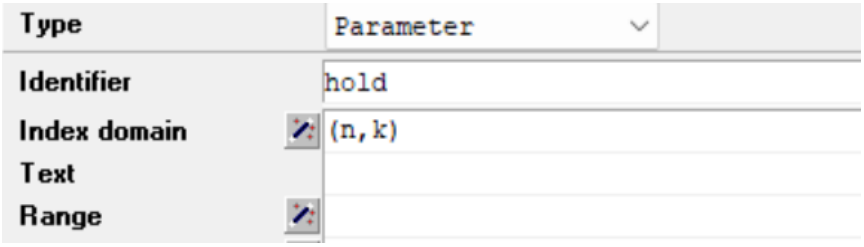


Capture 19- T_cost



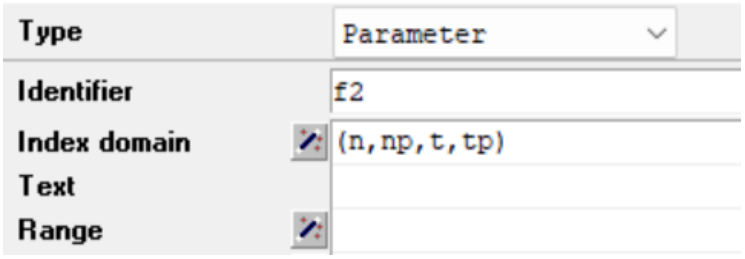
Comment on *T_cost:* it is the variable cost of using a service (*n,np,t,tp*) for transportation.

Capture 20- hold



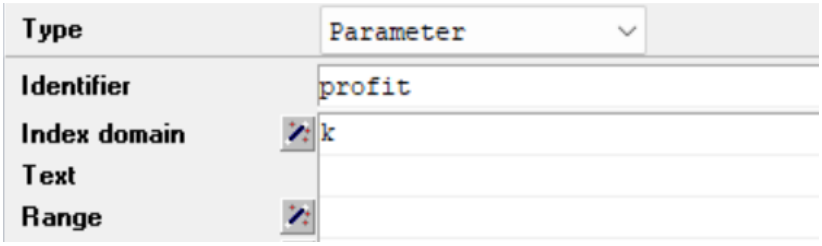
Comment on *hold:* it is the variable cost of holding whatever item from demand *k* on node *n*. Example: holding demand 2 on terminal 3 can cost 5.

Capture 21- f2

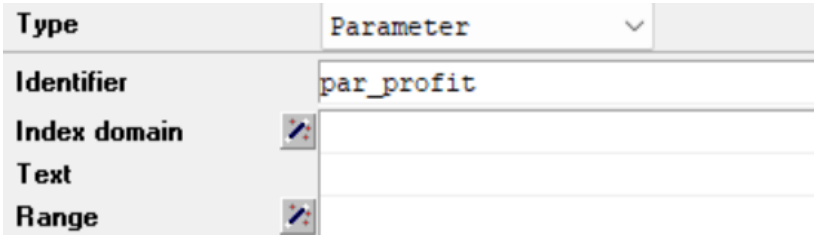


Comment on *f2*: it is the fixed cost of the service when it is activated.

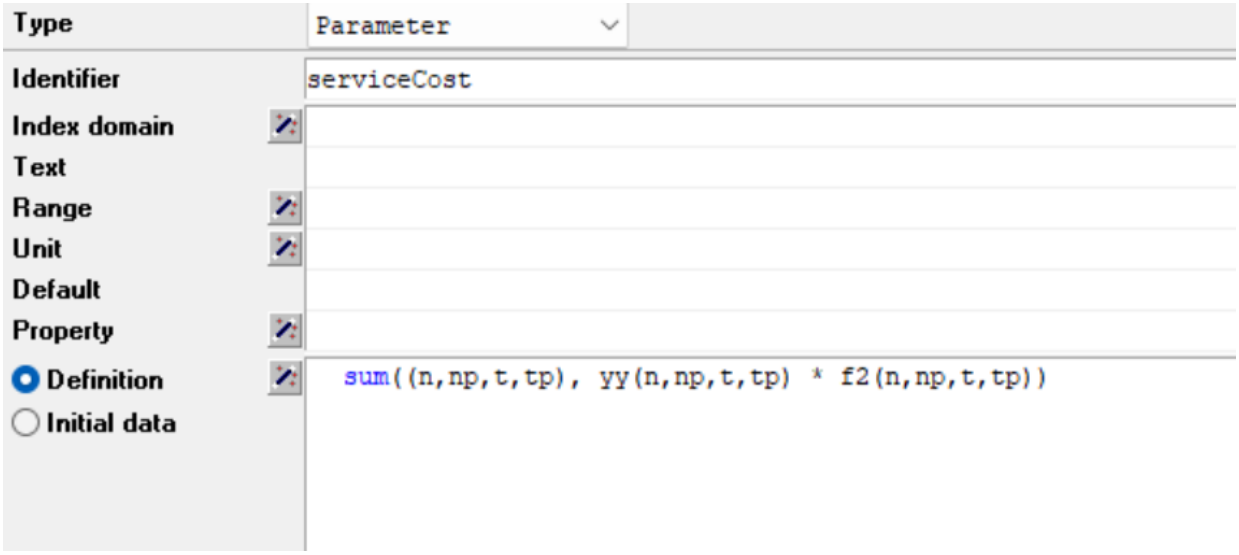
Capture 22- profit



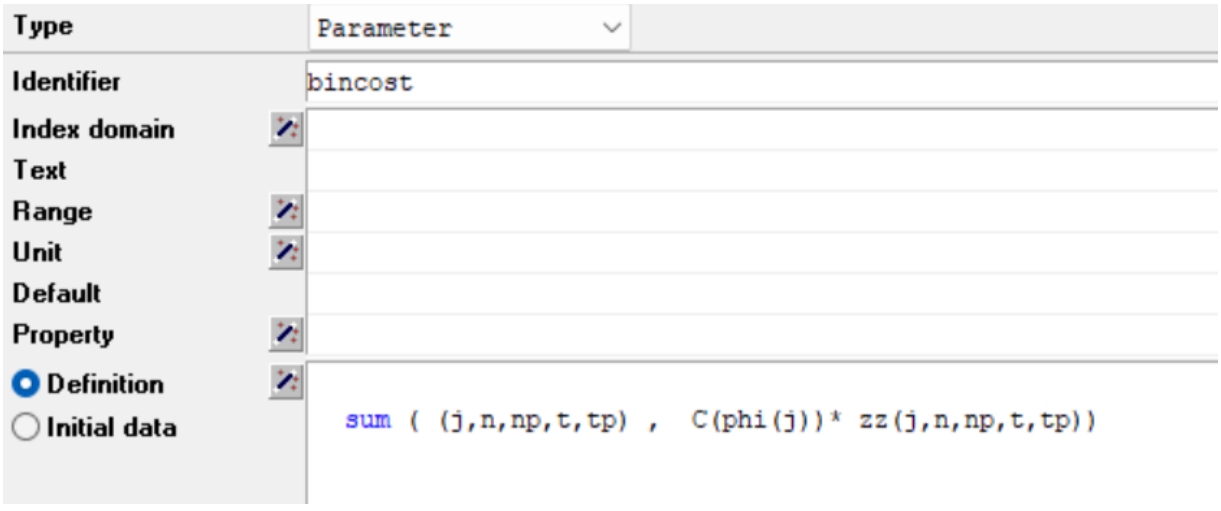
Capture 23- par_profit



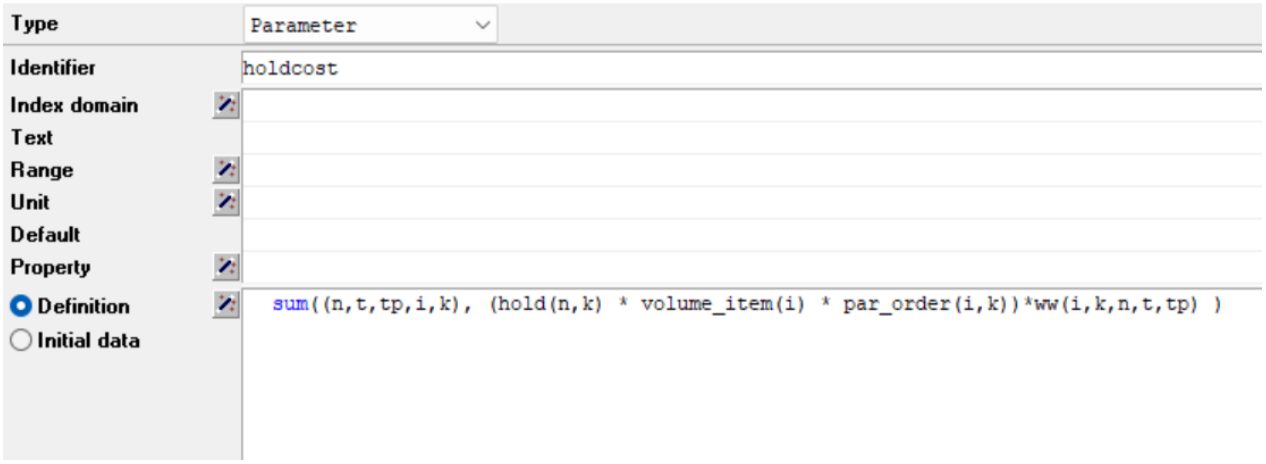
Capture 24- serviceCost



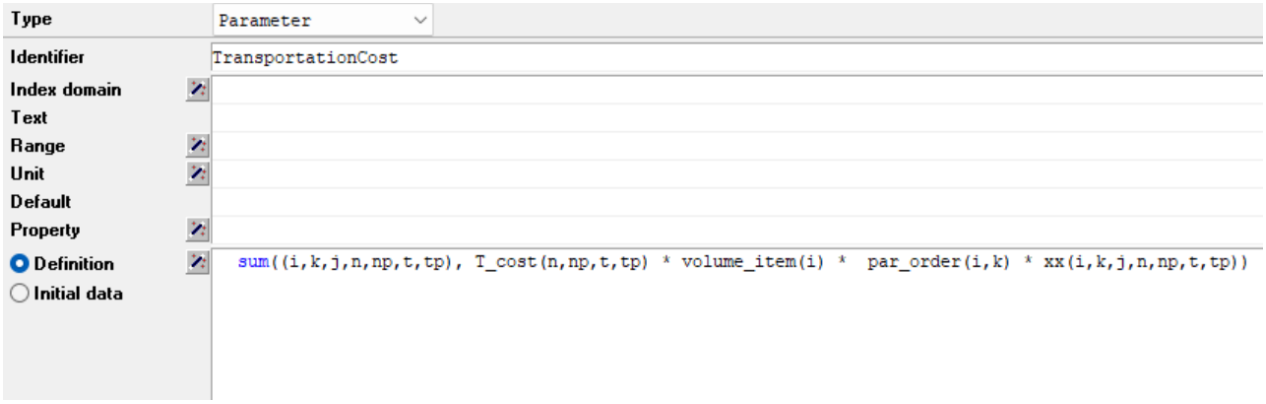
Capture 25- bincost



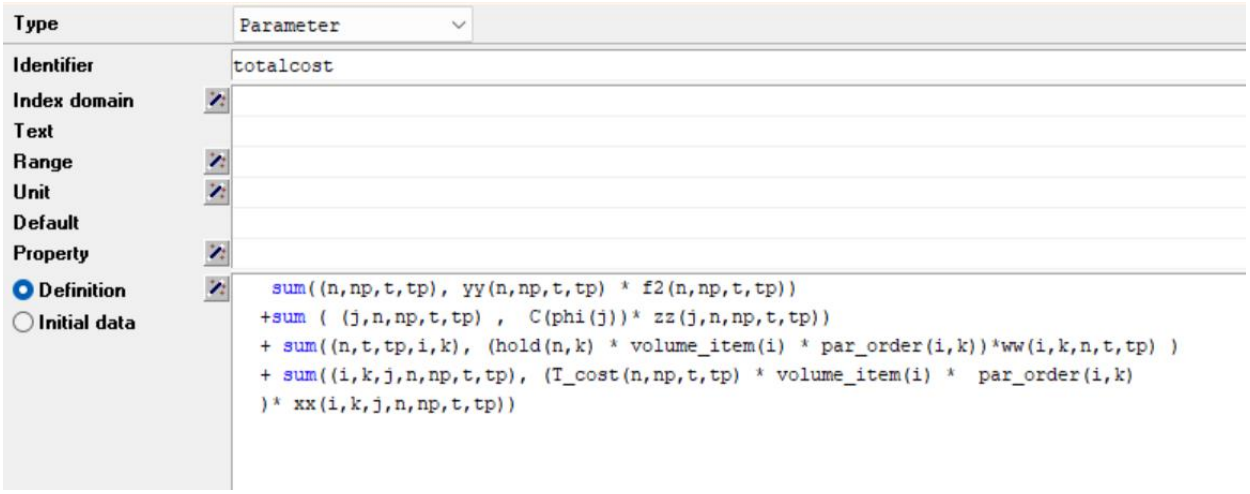
Capture 26- holdcost



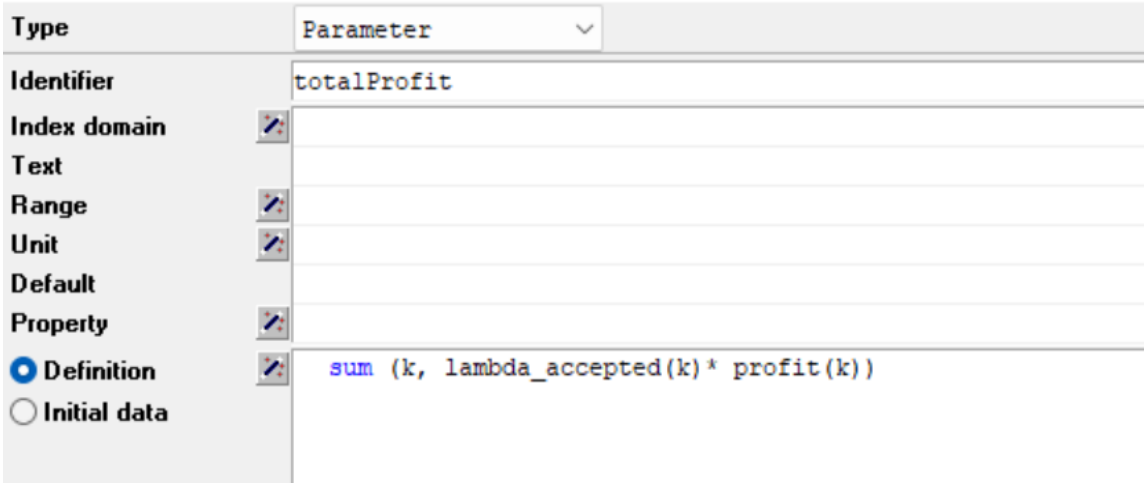
Capture 27- TransportationCost



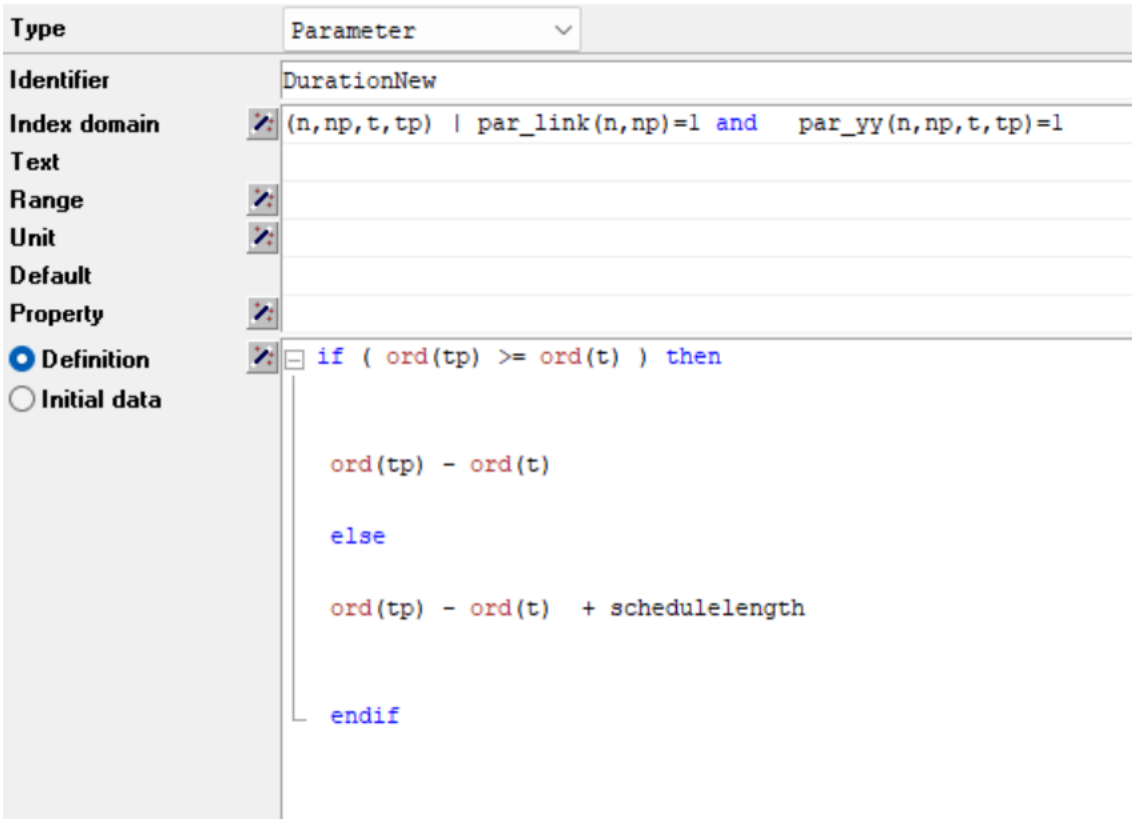
Capture 28- totalcost



Capture 29- totalProfit



Capture 30- DurationNew

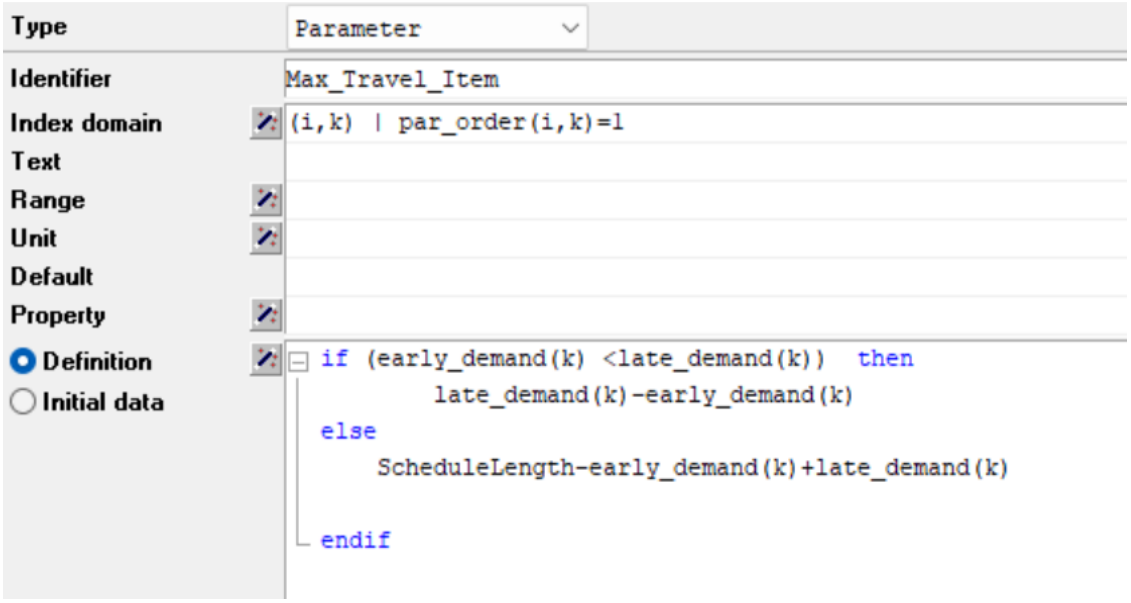


Comment on *DurationNew*: this parameter just represents the duration of each existing service, given that there is an existing link between the two nodes (*par_link(n,np)=1*). It states that, if *tp* is a quantity bigger or equal than *t*, this parameter will be equal to their difference. If instead *tp* is less than *t*, DurationNew will be equal their difference + *schedulelenght,* which is summed just to avoid a negative

duration.

This parameter is used in the procedure for solving the model, establishing that the cost of transportation is proportional to the time that it is spent by the service.

Capture 31- Max_Travel_Item



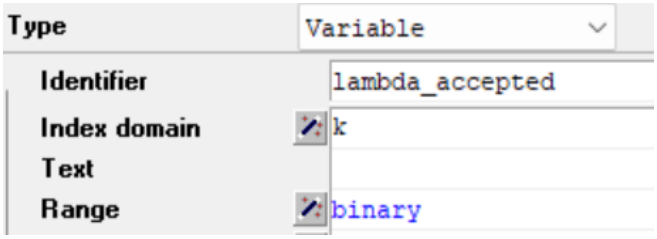
6.4 Variables

In the context of constraints in AIMMS, the word variable denotes an unknown quantity (Variable Declaration, 2024). Constraints can be grouped together to form a system of simultaneous equations and/or inequalities, which is referred to as a mathematical program. Variables in a mathematical program are assigned values when a solver (a solution algorithm) finds a solution for the unknowns in the system. When an instance is given to the software and the model is executed, the best obtainable values of the variables are given as output by the software, that is, the optimum values.

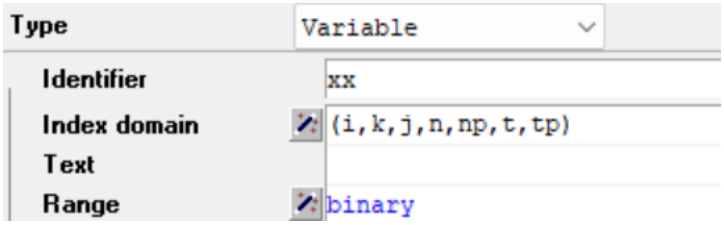
Variables have some additional attributes above those of parameters. These extra attributes are used to steer a solver, or to hold additional information about solution values provided by the solver. The possible attributes of variables are IndexDomain, Text, Range, Unit, Property, Default, Property and Definition.

The AIMMS model variables are those in the following captures, with their respective explanations when it is required.

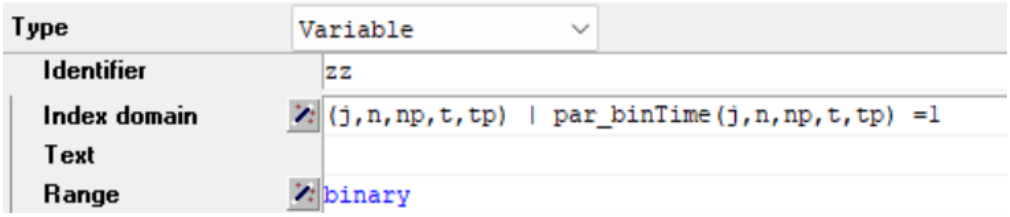
Capture 32- lambda_accepted



Capture 33- xx

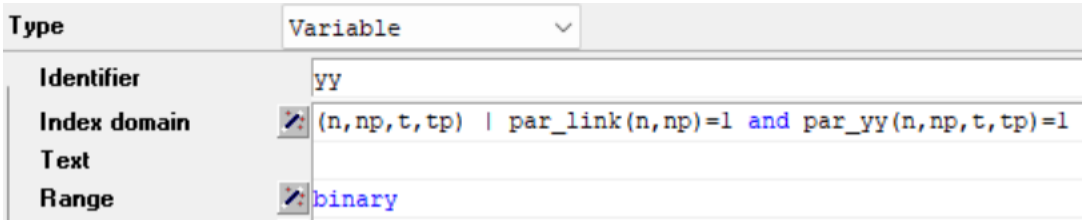


Capture 34- zz



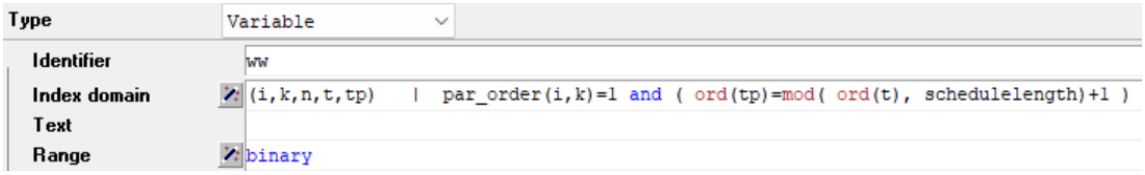
Comment on variable *zz*: the variable denotes if the bin *j* is selected or not and the Index domain indicates that the variable *zz* is defined only when the bin *j* is available between the two nodes *n* and *np* from time *t* to time *tp*.

Capture 35- yy



Comment on variable *yy*: the variable yy detones if the service is used or not. The respective Index Domain indicates that the variable *yy* is defined only for the cases in which exists a link between *n* and *np* and on that link a service between the period *t* and *tp* exists. The two conditions must coexist.

Capture 36- ww



Comment on variable *ww*: the variable *ww* indicates if item *i* from the demand *k* is held on node *n*. The Index Domain is expressing that the variable is defined only when the item *i* is inside the demand *k* and when *tp* is equal to the period which is sequentially next to *t*. The two conditions must coexist.

The mod function in AIMMS, as well as in many other programming platforms, represents the modulus operation (or remainder of division). The expression *mod(a, b)* returns the remainder of the integer division between *a* and *b*. In other words, it is the number that remains when *a* is divided by *b*.

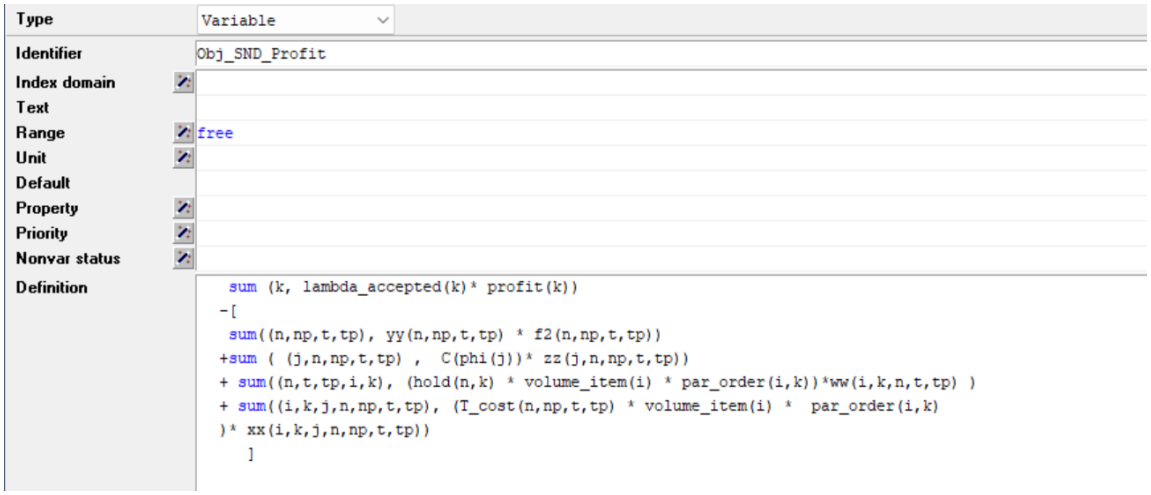
For example, the result of *mod(7, 3)* will be 1 because dividing 7 by 3 will get 2 as the quotient (integer division), and the remainder is 1 (7 = 3 * 2 + 1). The question that might arise is how come the period following *t*=7 can be *tp*=2. This is because the *mod* function refers to a cyclic context. *ScheduleLenght* in fact, suggests what the length of the cycle is, which in the example above is 3. So period number 7 actually corresponds to the first period of the cycle from 3. So even though *t* = 7, in a cycle of length 3, period *t* = 7 is treated as equivalent to period *t* = 1 . Adding 1 brings us to the next period in the cycle, i.e., tp = 2. This may seem strange because 7 sounds as a much larger number than 2, but in a cyclic context, the values repeat. It is important to note that cyclicality means that after reaching the maximum period (here, the third period), it goes back to the first.

Instead, when the first argument of *mod* is smaller than the second argument, the result of the modulus is simply the first number. So if *ord(t)*=3 and *ScheduleLenght*=15, *ord(tp)* will simply be equal to 4.

6.5 Objective Function

The objective function is enclosed in a variable named *Obj_SND_Profit*, which must then be maximized by the solver.

Capture 37- OBJ_SND_Profit



Comment on *Obs_SND_Profit*: it basically represents the difference between the return and the total costs, therefore the total profit. The first sum is the economic return given by the accepted demands, the second term is the sum of the fixed costs of a service when it is selected, the third term is the sum of the fixed costs when bin of type *j* is used. Regarding the fourth sum, it represents the total cost of holding items. Over a node *n*, period *t*, period *tp*, item *i* and demand *k,* the holding cost is given by the variable cost of holding the item *i* of demand *k* multiplied by the volume of item *i*, provided that the item *i* is within order *k* (*par_order(i,k)*=1) and that the item *i* of demand *k* is held on node *n* (*ww(i,k,n,t,tp)*=1).

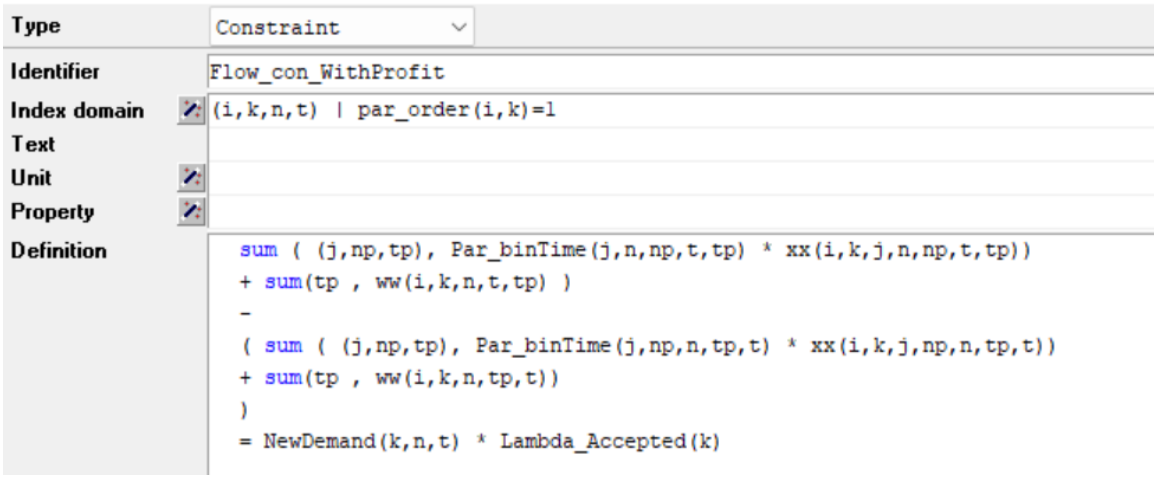
Finally, the last sum represents the total cost of transporting items. This total cost is given by the sum over an item *i*, demand *k*, bin *j,* service (*n,np,t,tp)* between the variable transportation cost over a service multiplied by the volume of item *i,* if the item *i* is inside demand *k* and it has been assigned to bin *j* of service *(n,np,t,tp)*.

6.6 Constraints

Constraints are numerical relations containing expressions in terms of variables, parameters and constants and form the major mechanism for specifying a mathematical program in AIMMS (Constraint Declaration, 2024). They are used to restrict the values of variables with interlocking relationships. Thus, it is only by respecting the constraints that the variables can take on their final values that the software will provide as output.

The AIMMS model constraints are those in the following captures, with their respective explanations when it is required.

Capture 38- Flow_con_WithProfit



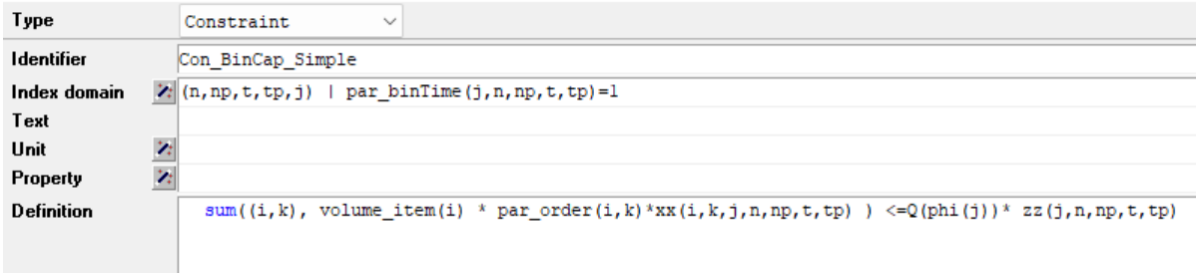
Flow_con_WithProfit corresponds to expression number (2) of the model.

Observations on *Flow_con_WithProfit*: this constraint is a flow budget. For each node *n* and period *t*, the total outflow minus the total inflow must satisfy a certain condition. Summarily, as already seen in the initial model, if n and t correspond to the origin terminal and the availability time of demand *k*, then the net flow must equal λ_k, if n and t correspond to the destination node and the due-date of demand k, then the net flow must be -λ_k. In all other cases the net flow will be 0 since what goes in comes out.

Thus, given a node *n*, a time *t*, a demand *k* and an item *i* inside this demand: the first sum over indices *j*, *np*, and *tp* represents the flow entering in *n*, the second sum over *tp* represents the flow held in *n*, the third sum over represents the flow going out from *n* and the fourth sum represents the flow held. The overall sum among these must be equal to *NewDemand(k,n,t)* multiplied by λ_k. Given that k has been accepted and λ_k is equal to 1, considering the NewDemand parameter, the right-hand side of the equation is 1 if n is the origin of k and t the availability time, is -1 if they correspond to the destination and the due date and it is 0 otherwise.

As shown by the Index Domain, the constraint applies only when the general item *i* is required to be transported by the demand *k*.

Capture 39- Con_BinCap_Simple

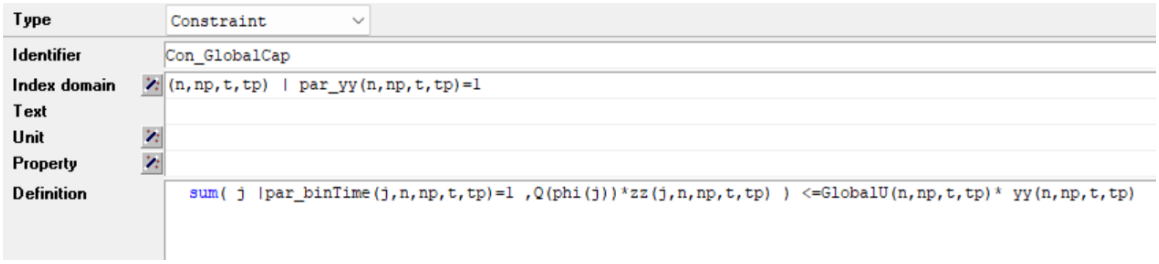


Con_BinCap_Simple corresponds to expression number (3) of the model.

Comment on *Con_BinCap_Simple:* this constraint ensures that the capacity of each bin is not exceeded. For each bin *j* that can operate a service *(n,np,t,tp),* the sum of all the items effectively contained in that bin *j* belonging to different demands *k* must be not bigger than the capacity of the bin *j* if it operates on the service.

The Index Domain suggests that the constraint is for each bin *j* able to operate on service *(n,np,t,tp)* while the sum on the left hand moves over the items *i* and demands *k* once the bin *j* has been fixed.

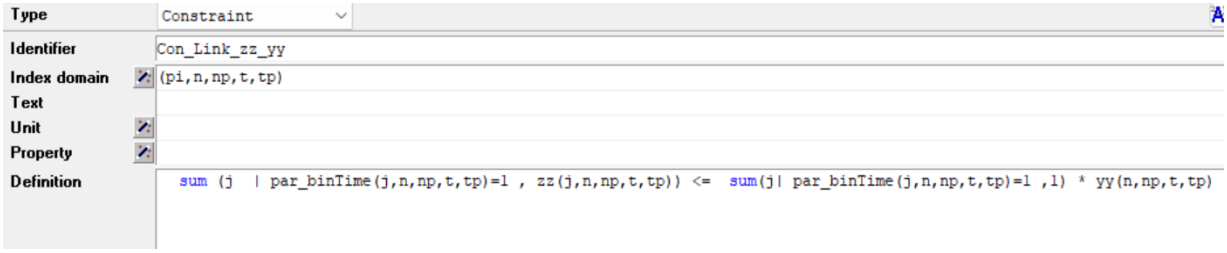
Capture 40- Con_GlobalCap



Con_GlobalCap corresponds to expression number (4) of the model.

Comment on *Con_GlobalCap:* this constraint ensures that the total capacity of the bins used in the service is not bigger than the service maximum capacity. For an available (look at the Index Domain) service, the sum of the capacity of the bins used on that service must be minor or equal than the capacity of the service if it is active (if the service is inactive, the sum must be 0).

Capture 41- Con_Link_zz_yy

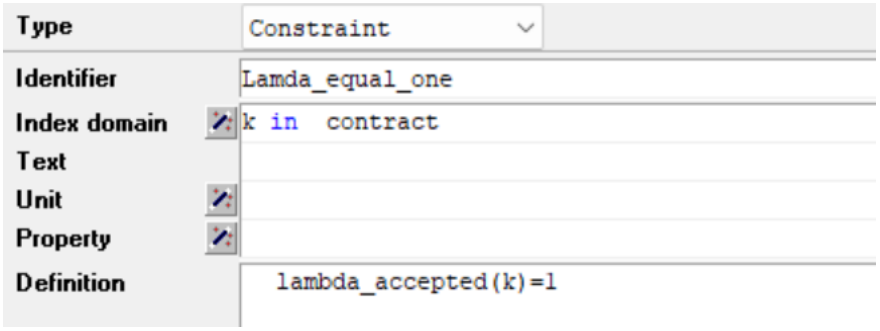


Con_Link_zz_yy corresponds to expression number (4) of the model.

Comment on *Con_Link_zz_yy:* this constraint states the link between the variable *zz* and the variable *yy*.

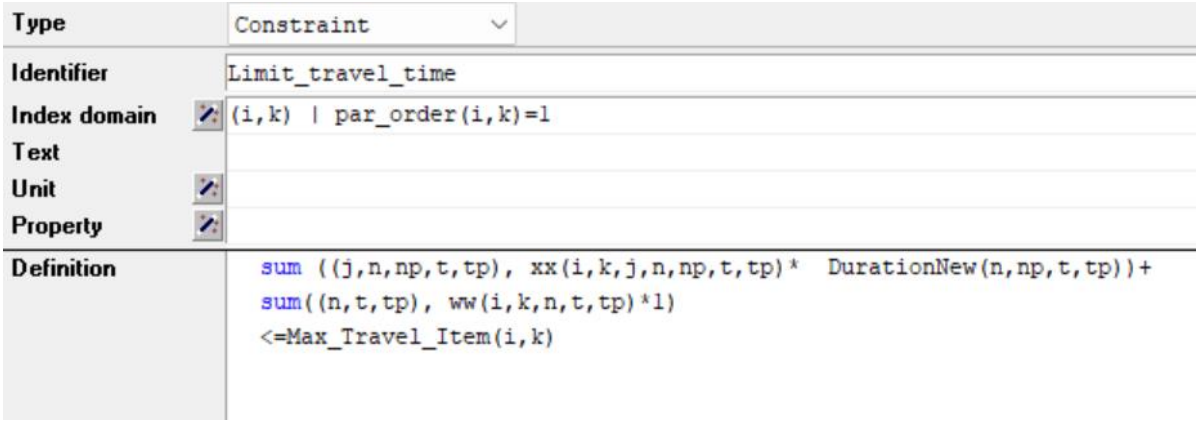
It suggests that, for each bin type operating on a service, the sum of the selected bins must be minor or equal than the number of bin available on that service if the service has been selected, if the service hasn't been activated this sum must be 0.

Capture 42- Lambda_equal_one



Lambda_equal_one corresponds to expression number (5) of the model.

Capture 43- Limit_travel_time



Limit_travel_time requires that each item in each order does not exceed the maximum travel time possible.

6.7 Mathematical Program

A mathematical program consists of a set of unknowns to be determined, a collection of constraints that must be satisfied, and an (optional) objective function to be optimized.

The aim of a mathematical program is to find a solution with the aid of a solver such that the objective function assumes an optimal (i.e. minimal or maximal) value. Depending on the characteristics of the variables and constraints, a mathematical program in AIMMS can be classified as one of the following (Solving Mathematical Programs, 2024).

- If the objective function and all constraints contain only linear expressions (in terms of the variables), and all variables can assume continuous values within their ranges, then the program is a linear program.
- If some of the variables in a linear program can assume only integer values, then the program is a linear mixed integer program.
- If the objective is a quadratic function in terms of the variables while the constraints are linear, then the program is a quadratic program.
- If the objective is neither linear nor quadratic, or some of the constraints contain nonlinear expressions, the program is a nonlinear program.

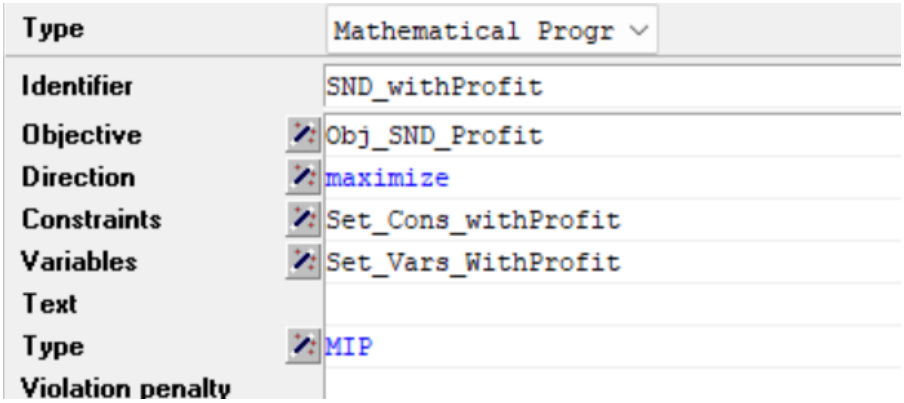
In this case, the program is MIP (Mixed Integer Programming). AIMMS will automatically call the appropriate solver to find an (optimal) solution. The possible attributes of a mathematical program are Objective, Direction, Constraints, Variables and Type.

The *Objective* attribute is used to specify the objective of the mathematical program. If no objective is specified, the mathematical program will be solved to find a feasible solution and then terminate. In conjunction with an objective, the *Direction* attribute must be used to indicate whether the solver should minimize or maximize the objective. The *Constraints* attribute allows specifying which constraints are part of the mathematical program. Its value must either be the predefined set *AllConstraints* or a subset thereof. If the set *AllConstraints* is specified, AIMMS will generate individual constraints for all declared constraints and variables with a definition.

If a subset of *AllConstraints* is specified, AIMMS will generate individual constraints only for the declared constraints and defined variables in that subset.

The *Variables* attribute is used to specify which set of variables are to be included in the mathematical program. Its value must either be the predefined set *AllVariables* or a subset thereof. The set *AllVariables* is predefined by AIMMS and contains the names of all variables declared in the model.

Capture 44- SND_withProfit



6.8 Data initialization

After presenting the model code on AIMMS, this chapter aims to show its application. The functionality will be observed, and the results related to different instances will be analyzed. First, it is important to emphasize how the model is something distinct from the data. In general, it is a good strategy to separate the initialization of data from the specification of the model structure. This is particularly true for large models like this. The separation improves the clarity of the model text, but more importantly, it allows to use the same model structure with various data sets. Thus, the model is something extensible and adaptable, as long as the characteristics of the data match those expected by the model.

There are several methods to input the initial data of the identifiers in the model (Data Initialization, 2024). It is possible to supply initial data for a particular identifier as part of its declaration or to read in data from various external data sources, such as text data files, AIMMS cases and databases, and to initialize data by means of algebraic statements.

In the case of this work, the data are read from loading external data files.

6.9 Output report

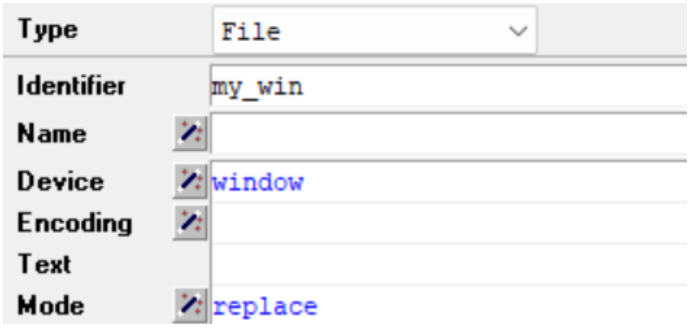
The AIMMS system has several reporting features to present model results (Text Reports and Output Listing, 2024). A text report lets allows to save the model results in files.

The result can be written to either a file or to a text window in the graphical user interface. The *Name* attribute specifies the actual name of the disk file or window being referenced. If the file identifier refers to a disk file, the Name corresponds to the file name on disk. When it refers to a window, the Name attribute serves as the window's title. If a name is not specified, AIMMS automatically generates a default name, using the internal identifier as the root and *".put"* as the extension. The *Device* attribute can take three values: disk (default), window, and void. It indicates whether the output should be directed to an external file on disk, a window in the graphical user interface, or if no output should be generated. The void device is particularly useful for hiding output statements during model development, which should not appear in the end-user version.

The *Mode* attribute determines whether the file or window should be overwritten (replace mode, default) or appended to (merge mode). Unlike files, graphical windows in the user interface can be manually closed, leading to the loss of their contents, and AIMMS will start writing to a new instance regardless of the mode.

The figure below shows the creation of the reporting feature for the results.

Capture 45- my_win



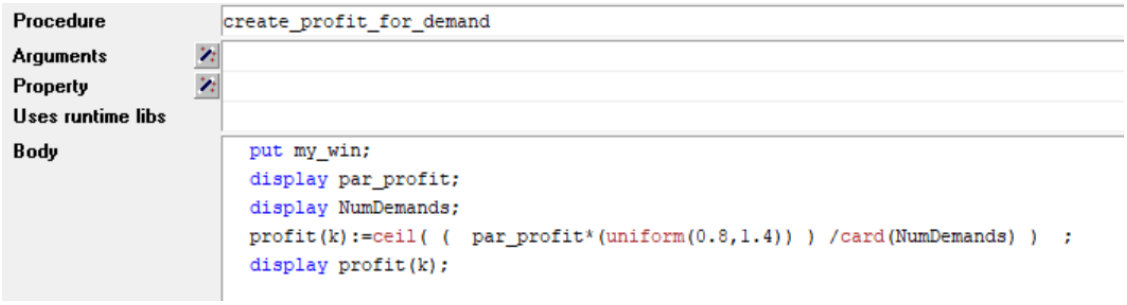
Furthermore, AIMMS provides two statements to create a customized text output report in either a file or in a text window in the user interface. They are the *put* and the *display* statements. It is possible to use the *display* statement to print the data associated with sets, parameters and variables to a file or window in AIMMS format.

It is possible to send output to a particular file by providing the associated File identifier as the first argument of a *put* statement. If the *display* statement or any of the *put* operators are used without a file identifier, AIMMS will direct the output to the current file, i.e., the file last opened through the *put* statement.

6.10 The procedures

This section shows codes for the procedures used to solve the model on instances.

Capture 46- create_profit_for_demand



Procedure: create_profit_for_demand

Arguments

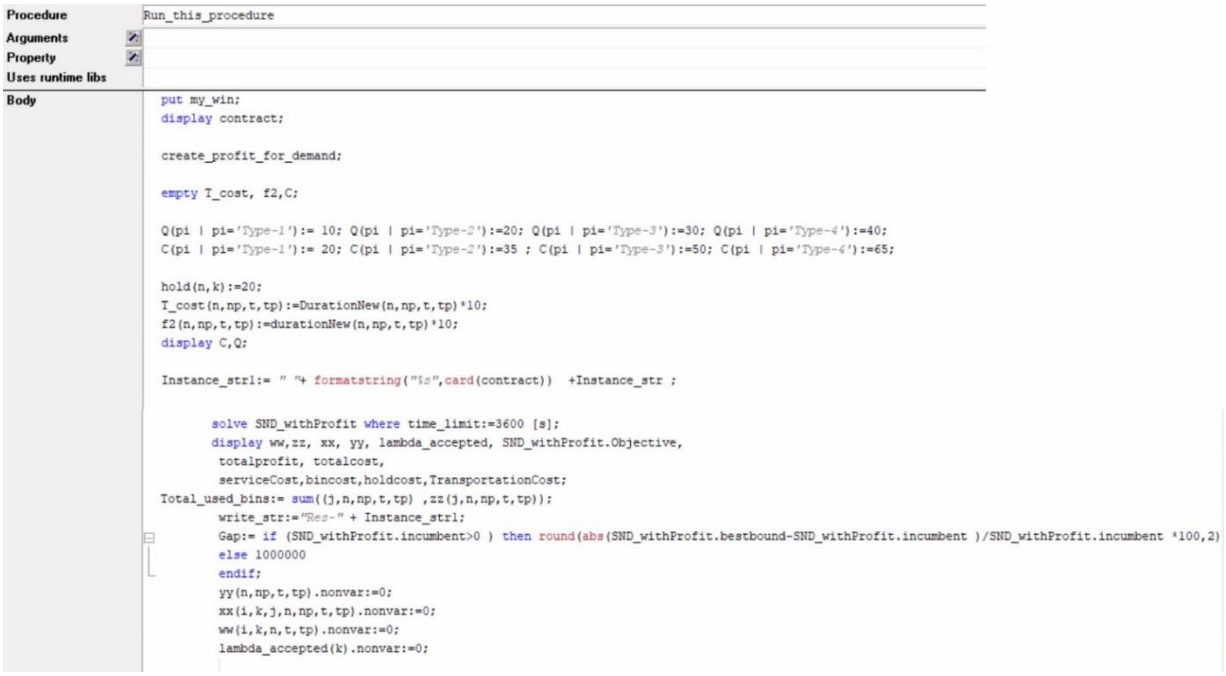
Property

Uses runtime libs

Body

```
put my_win;
display par_profit;
display NumDemands;
profit(k):=ceil( (  par_profit*(uniform(0.8,1.4)) ) /card(NumDemands) )  ;
display profit(k);
```

Capture 47- Run_this_procedure



Procedure: Run_this_procedure

Arguments

Property

Uses runtime libs

Body

```
put my_win;
display contract;

create_profit_for_demand;

empty T_cost, f2,C;

Q(pi | pi='Type-1'):= 10; Q(pi | pi='Type-2'):=20; Q(pi | pi='Type-3'):=30; Q(pi | pi='Type-4'):=40;
C(pi | pi='Type-1'):= 20; C(pi | pi='Type-2'):=35 ; C(pi | pi='Type-3'):=50; C(pi | pi='Type-4'):=65;

hold(n,k):=20;
T_cost(n,np,t,tp):=DurationNew(n,np,t,tp)*10;
f2(n,np,t,tp):=durationNew(n,np,t,tp)*10;
display C,Q;

Instance_str1:= " "+ formatstring("%s",card(contract))  +Instance_str ;

      solve SND_withProfit where time_limit:=3600 [s];
      display ww,zz, xx, yy, lambda_accepted, SND_withProfit.Objective,
       totalprofit, totalcost,
       serviceCost,bincost,holdcost,TransportationCost;
Total_used_bins:= sum((j,n,np,t,tp) ,zz(j,n,np,t,tp));
       write_str:="Res-" + Instance_str1;
       Gap:= if (SND_withProfit.incumbent>0 ) then round(abs(SND_withProfit.bestbound-SND_withProfit.incumbent )/SND_withProfit.incumbent *100,2)
       else 1000000
       endif;
       yy(n,np,t,tp).nonvar:=0;
       xx(i,k,j,n,np,t,tp).nonvar:=0;
       ww(i,k,n,t,tp).nonvar:=0;
       lambda_accepted(k).nonvar:=0;
```

7 Data Generation

The generation of instances for Network Design problems is a complex problem itself, as it requires balancing realism and computational manageability. The challenge lies in creating diverse instances that reflect real-world complexity while remaining solvable with existing optimization methods. It is not a coincidence that a literature on this exists: the paper "Pseudo-random Instance Generators in C++ for Deterministic and Stochastic Multi-commodity Network Design Problems" (Larsen, Frejinger, Bisaillon, & Cordeau, 2023) explores methods to create pseudo-random instances efficiently and aims at advancing research in network design by providing standardized, flexible instances for comparison and benchmarking.

This section explains the creation process of the dataset used for model execution, detailing the duplication challenges encountered and the careful adjustments required to address them effectively.

The standard use of excel was not sufficient to perform necessary actions for data creation. For this reason, the development section of excel, or more specifically Macros, has been used. Macros are programming codes that run on Excel environment to perform a procedure, that is, a set of commands and instructions carried out during the execution of a program. The programming language used to create Macros is VBA (Visual Basic for Applications).

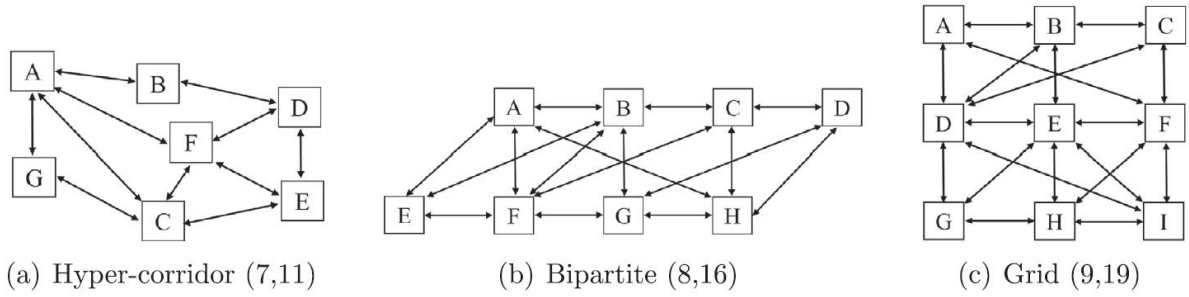


Figure 6- Physical Network Instances (Taherkhani, Bilegan, Crainic, Gendreau, & Rei, 2022)

The instances considered in the experiments are inspired by those used in "Tactical capacity planning in an integrated multi-stakeholder freight transportation system" (Taherkhani, Bilegan, Crainic, Gendreau, & Rei, 2022).

They have been generated based on three different physical network topologies, as

represented in the figure: (a) a hyper-corridor network with seven terminals, (b) a bipartite network with eight terminals, and (c) a grid network with nine terminals (see Figure 6). These instances have been used to evaluate the impact of different network topologies on the computational performance of the proposed model.

Both cyclic and non-cyclic cases have been considered in the instances generation, but only the cyclic case has been computed.

For both cases, a schedule length of seven days (one week) has been considered, divided into 14 time periods of half a day each. The instances have been generated following a uniform distribution, based on realistic cases, for each network topology.

In the steps below, whenever it is mentioned that values are generated randomly, it means that the excel RAND() function has been used.

7.1 Service creation

The grid has 9 terminals and 19 edges going forwards and backwards, thus 38 arcs. For each arc (which represents a link between two terminals) 7 services have been considered, for a total of 266 services.

The hyper-corridor has 7 terminals and 16 edges going forwards and backwards, thus 22 arcs. For each arc 7 services have been considered, for a total of 154 services.

In the same way, the Bipartite shows 8 terminals and 16 edges going forwards and backwards, thus 32 arcs, for a total of 224 services considering 7 services per arcs.

Duration of each service has been randomly generated between 1 and 4. Capacity of each service has been homogeneously set at 40.

Non-cyclic case

Since for each arc 7 different services must be generated, to avoid that same departure and arrival are replicated for the same arc (which would imply a wrong service duplication), it has been taken advantage of the fact that the periods in which services can depart are 7 just like the total number of services for each arc. Thus, to keep the solution as straightforward as possible and avoid further complicating the Excel file, one departure is selected for each period. Arrival of the service, then, is simply the sum of the departure period and the duration.

Cyclic case

For the cyclic case, instead, a Macro called "GenerateRandomValuesColumnJ_Services" (see Figure 7) has been created. It generates random values under the column "Departure of the service", under the condition that every 7 rows they must be all different values.

```
Sub GenerateRandomValuesColumnJ_Services()
    Dim ws As Worksheet
    Dim i As Long
    Dim initialRow As Long
    Dim finalRow As Long
    Dim rng As Range
    Dim casualNumbers() As Long
    Dim n As Long
    Dim temp As Long
    Dim j As Long

    Set ws = ThisWorkbook.ActiveSheet


    initialRow = 4
    finalRow = 157
    ' Cycle through every 7 lines
    For i = initialRow To finalRow Step 7
        ' Initialize the array of random numbers from 1 to 14
        ReDim casualNumbers(1 To 14)
        For n = 1 To 14
            casualNumbers(n) = n
        Next n

        ' Shuffles the array of random numbers (Fisher-Yates algorithm)
        For n = 14 To 2 Step -1
            j = Application.WorksheetFunction.RandBetween(1, n)
            temp = casualNumbers(n)
            casualNumbers(n) = casualNumbers(j)
            casualNumbers(j) = temp
        Next n

        ' Enter the first 7 shuffled numbers in the corresponding rows of column J
        Set rng = ws.Range("J" & i & ":J" & i + 6)
        For n = 1 To rng.Rows.count
            rng.Cells(n, 1).Value = casualNumbers(n)
        Next n
    Next i

    MsgBox "Random values under columnJ (departures of the services) have been successfully generated!"
End Sub
```

Figure 7- Macro for service creation (Cyclic case)

7.2 Demand Creation

For each of the three networks, four different sets of shipper demand with increasing size have been considered: 50, 75 and 100.

The process for generating the instances for the commodities is described as follows.

Initially, for each network, 50 commodities have been generated.

The origin of each commodity has been randomly generated. Similarly, the destination has been randomly generated. At this point, for the second time the problem related to duplication must be frothed, since it was not enough to randomly

generate the arrival terminal, but it was necessary to make sure that it was different from the departure terminal. In order to achieve this, a test column has been created. After a “hypothetical destination” is generated, under the “test destination=/origin” column it is tested that it is different from the origin. If the destination is equal to the origin, then the final destination will be that hypothetical +1, so that it is different from the origin. This is clearly a simple solution but one that has proved to work without having to create a Macro.

The volume of the shipper demand requests for each test instance has been generated randomly using an uniform distribution between 10% and 50% of the capacity of the service legs (i.e., between 4 and 20).

Non-cyclic case

The availability time of each commodity has been randomly generated between 1 and 8. The same has been performed for the duration, which can assume a value between 2 and 6. Consequently, the due date was the sum of the two. The maximum due date achievable was 14, which still is inside the schedule length, but an additional column has been added in case the schedule length of 14 is overcome by the due date (which could happen just by augmenting the duration range or the availability time range of 1). The column in question was column I called “final due date”. The latter returned the due date calculated in the previous column if this was less than or equal to 14 (the schedule length considered). If it was greater than 14, this couldn’t fit the considered schedule length, and so it would have returned just 14, which was then the last possible due date.

Cyclic case

As for the Non-cyclic case, the availability time and duration have been generated randomly respectively between 1 and 8 and 1 and 10. Due date was simply the sum of the two if it was less or equal than 14 (considered schedule length), otherwise it was the sum of the two minus 14. The latter calculation, as already considered previously, gives back the respective cyclic period starting from the period number 1 when the cycle is ended.

Again, in generating commodities, it was absolutely necessary to make sure that there were no duplicates. Initially, an attempt was undertaken to create a single Macro under two conditions that at the same time had to be met. The two conditions were that simultaneously rows from column B to to column I were all different from each other, and rows from column B to column F joined with rows from column L to column M were all different from each other. In this way it would have been possible to have both the non-cyclic case and the cyclic case in one launch, but because of the limitations of excel it was too complicated and, for this reason, two different macros have been created, one for the non-cyclic case (Figure 8) and one for the cyclic case (Figure 9).

```
Sub FindAndUpdateDuplicates_noncyclic()
    Dim ws As Worksheet
    Dim lastRow As Long
    Dim i As Long, j As Long
    Dim currentKey As String
    Dim comparisonKey As String
    Dim updateDone As Boolean

    Set ws = ThisWorkbook.ActiveSheet

    lastRow = ws.Cells(ws.Rows.count, "B").End(xlUp).Row

    ' The control cycle begins
    Do
        updateDone = False ' Reset the update flag

        ' Loop over all rows to check duplicates above and below
        For i = 2 To lastRow
            ' Create the string of the current line with B+F+H+I
            currentKey = ws.Cells(i, "B").Value & ws.Cells(i, "F").Value & ws.Cells(i, "H").Value & ws.Cells(i, "I").Value

            ' Compare with all other rows, both above and below
            For j = 2 To lastRow
                If i <> j Then ' Skip the same line
                    ' Create comparison line string with B+F+H+I
                    comparisonKey = ws.Cells(j, "B").Value & ws.Cells(j, "F").Value & ws.Cells(j, "H").Value & ws.Cells(j, "I").Value

                    ' If the current string is equal to the key comparison
                    If currentKey = comparisonKey Then
                        ' Update cells H and I of the current row with .Calculate
                        ws.Cells(i, "H").Calculate
                        ws.Cells(i, "I").Calculate

                        ' Signals that an update has been performed
                        updateDone = True

                        ' Exit the inner loop and start the check again from the first line
                        Exit For
                    End If
                End If
            Next j

            ' If an update has been performed, also exit the external loop
            If updateDone Then Exit For
        Next i
    Loop While updateDone ' Start the check again as long as there are updates

    MsgBox "Duplicates have successfully been updated!", vbInformation
End Sub
```

Figure 8- Macro for commodities creation (Non-cyclic case)

```
Sub FindAndUpdateDuplicates_cyclic()
    Dim ws As Worksheet
    Dim lastRow As Long
    Dim i As Long, j As Long
    Dim currentKey As String
    Dim comparisonKey As String
    Dim updateDone As Boolean

    Set ws = ThisWorkbook.ActiveSheet

    lastRow = ws.Cells(ws.Rows.count, "B").End(xlUp).Row

    ' The control cycle begins
    Do
        updateDone = False ' Reset the update flag

        ' Loop over all rows to check duplicates above and below
        For i = 2 To lastRow
            ' Create the string of the current line with B+F+L+M
            currentKey = ws.Cells(i, "B").Value & ws.Cells(i, "F").Value & ws.Cells(i, "L").Value & ws.Cells(i, "M").Value

            ' Compare with all other rows, both above and below
            For j = 2 To lastRow
                If i <> j Then ' Skip the same line
                    ' Create comparison line string with B+F+L+M
                    comparisonKey = ws.Cells(j, "B").Value & ws.Cells(j, "F").Value & ws.Cells(j, "L").Value & ws.Cells(j, "M").Value

                    ' If the current string is equal to the key comparison
                    If currentKey = comparisonKey Then
                        ' Update cells H and I of the current row with .Calculate
                        ws.Cells(i, "L").Calculate
                        ws.Cells(i, "M").Calculate

                        ' Signals that an update has been performed
                        updateDone = True

                        ' Exit the inner loop and start the check again from the first line
                        Exit For
                    End If
                End If
            Next j

            ' If an update has been performed, also exit the external loop
            If updateDone Then Exit For
        Next i
    Loop While updateDone ' Start the check again as long as there are updates

    MsgBox "Duplicates have successfully been updated!", vbInformation
End Sub
```

Figure 9- Macro for commodities creation (Cyclic case)

The code represents a loop that updates the sheet until the condition is satisfied for the entire sheet. With this code, if a duplicate is found and the row is updated (using .Calculate), the code immediately restarts a new cycle from the first row of the sheet. This means that the recently updated row will be checked again, so if it now matches another row, it will be detected as a duplicate. This ensures that all rows, including those just updated, are re-evaluated. The process stops only when, after a full scan, no further updates are performed.

Notice that, even though the .Calculate function is applied only to cells H and I for the non-cyclic case and L and M for the cyclic case, in reality the sheet is recalculated entirely by excel. This is because sheets with many dependencies and complex formulas tend to cause global recalculation, even when an attempt is made to limit the action to a specific cell or range.

It is important, now, to highlight three particularly challenging aspects of these Macros.

1. An error that could happen by trying to create a unique macro for both the non-cyclic case and the cyclic case might be to directly compare the line from cell B to cell M, creating a single condition. This is wrong and the conditions must be separate. Indeed, by doing this error, the rows will obviously always be different, since there is no correlation between the non-cyclic case and the cyclic case. In other words, there could be a duplicated non-cyclic commodity (so between the same terminals, same availability date and same duration) , but, comparing the entire row from cell B to cell M, despite having a duplicate for the non-cyclic case, it would not be duplicate for the code, since with a very high probability the cyclic case has different values, which then makes the two rows different from each other.
2. It is important that the macro checks the current row not only with the ones below but also with the ones above.
   One should not think that because the rows above have already been checked, then for each row this can be compared only with the ones below. Every time excel updates the values, it updates the entire sheet, so what might happen by comparing only the row with the ones below is that, in the meantime updating the sheet, new duplicates have been created in the rows above.
3. The caution in step two is not enough, the code must start again from the first row each time the sheet is updated, because even if the last rows checked have no duplicates, duplicates could be created between the rows already checked. Example: at row 45 the sheet is updated, and from row 45 to the last, these have no duplicates either above or below. But maybe by updating the sheet, row 27 and row 35 have become the same.

Depending on whether the non-cyclic case or the cyclic case is of interest, either "FindAndUpdateDuplicates_noncyclic" or "FindAndUpdateDuplicates_cyclic" will be launched, respectively, in the excel sheet for the generation of commodities.

The first output of the launch of the macro represents the "case 1" for 50 commodities. Then, 25 commodities are added, creating "case 1" for 75 commodities. From "case 1" of 75 commodities other 25 commodities are added, generating "case 1" for 100 commodities.

Each time, to generate a new case, it has been enough to run the macro described above in the generation sheet. Then, the cardinality of the commodities is amplified following the same procedure. In this way each case is completely different from the others since it concerns different randomly generated terminals. It is important that after doing this, what is generated is kept fixed, otherwise it will continue to be update because excel, at every operation, updates the randomly generated values.
In total, this resulted in 10 cases for 50 commodities, 10 cases for 75 commodities, and 10 cases for 100 commodities for the grid, the bipartite and the hyper-corridor network.

7.2.1 Second level duplicates verification for demand

To further verify that the data created for each case have no duplicates, two macro have been created “CheckCommoditesDuplicatedNonCyclic” (Figure 10) and “CheckCommoditiesDuplicatedCyclic” (Figure 11), which respectively gives as output the lines of duplicated commodities for the non-cyclic case and the cyclic case, in case they exist. For a second level verification, the two macro have been launched for all the cases, and of course the output has always been null. This means that for both the non-cyclic case and the cyclic case there was no duplication.

```
Sub CheckCommoditiesDuplicatedNonCyclic()
    Dim ws As Worksheet
    Dim lastRow As Long
    Dim count As Long
    Dim i As Long, j As Long
    Dim duplicates As String


    Set ws = ThisWorkbook.ActiveSheet

    ' Find the last row with data in column B
    lastRow = ws.Cells(ws.Rows.count, "B").End(xlUp).Row

    ' Initialize duplicate counter and string
    count = 0
    duplicates = "Duplicated rows: "

    ' Cycle through all rows
    For i = 2 To lastRow
        For j = i + 1 To lastRow
            ' Check that key cells are not empty
            If ws.Cells(i, 2).Value <> "" And ws.Cells(i, 6).Value <> "" And ws.Cells(i, 8).Value <> "" And ws.Cells(i, 9).Value <> "" Then
                ' Checks whether cells in columns B, F, H, and I are equal (insensitive to capitalization and spaces)
                If UCase(Trim(CStr(ws.Cells(i, 2).Value))) = UCase(Trim(CStr(ws.Cells(j, 2).Value))) And _
                   UCase(Trim(CStr(ws.Cells(i, 6).Value))) = UCase(Trim(CStr(ws.Cells(j, 6).Value))) And _
                   UCase(Trim(CStr(ws.Cells(i, 8).Value))) = UCase(Trim(CStr(ws.Cells(j, 8).Value))) And _
                   UCase(Trim(CStr(ws.Cells(i, 9).Value))) = UCase(Trim(CStr(ws.Cells(j, 9).Value))) Then

                    ' If all columns B, F, H and I are equal, increase the counter
                    count = count + 1
                    duplicates = duplicates & vbCrLf & "Row " & i & " and Row " & j
                End If
            End If
        Next j
    Next i

    ' Show the result with the number of duplicate rows
    If count > 0 Then
        MsgBox duplicates & vbCrLf & "Total number of duplicate rows: " & count
    Else
        MsgBox "There are no duplicate rows."
    End If
End Sub
```

Figure 10- Macro for data duplication verification (Non-cyclic case)

```
Sub CheckCommoditiesDuplicatedCyclic()
    Dim ws As Worksheet
    Dim lastRow As Long
    Dim count As Long
    Dim i As Long, j As Long
    Dim duplicates As String

    Set ws = ThisWorkbook.ActiveSheet

    ' Find the last row with data in column B
    lastRow = ws.Cells(ws.Rows.count, "B").End(xlUp).Row

    ' Initialize duplicate counter and string
    count = 0
    duplicates = "Duplicated rows: "

    ' Cycle through all rows
    For i = 2 To lastRow
        For j = i + 1 To lastRow
            ' Check that key cells are not empty
            If ws.Cells(i, 2).Value <> "" And ws.Cells(i, 6).Value <> "" And ws.Cells(i, 12).Value <> "" And ws.Cells(i, 13).Value <> "" Then
                ' Checks whether cells in columns B, F, H, and I are equal (insensitive to capitalization and spaces)
                If UCase(Trim(CStr(ws.Cells(i, 2).Value))) = UCase(Trim(CStr(ws.Cells(j, 2).Value))) And _
                   UCase(Trim(CStr(ws.Cells(i, 6).Value))) = UCase(Trim(CStr(ws.Cells(j, 6).Value))) And _
                   UCase(Trim(CStr(ws.Cells(i, 12).Value))) = UCase(Trim(CStr(ws.Cells(j, 12).Value))) And _
                   UCase(Trim(CStr(ws.Cells(i, 13).Value))) = UCase(Trim(CStr(ws.Cells(j, 13).Value))) Then

                    ' If all columns B, F, H and I are equal, increase the counter
                    count = count + 1
                    duplicates = duplicates & vbCrLf & "Row " & i & " e Row " & j
                End If
            End If
        Next j
    Next i

    ' Show the result with the number of duplicate rows
    If count > 0 Then
        MsgBox duplicates & vbCrLf & "Total number of duplicate rows: " & count
    Else
        MsgBox "There are no duplicate rows."
    End If
End Sub
```

Figure 11-Macro for data duplication verification (Cyclic case)

The central part of the macro has been introduced because, in the comparison between cells, there were cases where invisible characters might be present without

being noticed. This often happens if the data comes from copy/paste. Thus, even if the cells looked identical, VBA might have treated them differently because of small, unseen variations (such as case-sensitive differences or interpretation between numbers and strings). This macro version introduces functions (UCase, Trim, CStr) to eliminate any possible cause of discrepancy and make the comparison accurate.

Profits for each commodity have been generated randomly through the *create_profit_for_demand* procedure on AIMMS, consistently and proportionally to costs. The reference value from which profits were generated following a uniform distribution between 0.8 and 1.4 is 2815 for bipartite, 1760 for grid, and 3570 for hyper-corridor. These values derives from a cost estimation.

8 Execution & Results

The execution of the model has been performed both on the bipartite, grid and hyper-corridor, considering a cardinality of 50 for the demand of all the three networks. Not all cases have been computed due to computational time limitations, but clearly similar results for them are expected.

Moreover, although instances have been generated for both the cyclic and non-cyclic cases, only the cyclic one has been computed, hoping that the non-cyclic case will be useful for future experimental research.

The total number of services, as previously anticipated in the section about data creation, is 154 for the hyper-corridor, 224 for the bipartite and 266 for the grid. The respective total capacity, considering that each service has a capacity of 40, is 6160 for the hyper-corridor, 8960 for the bipartite and 21640 for the grid.

Then, four types of bins with the following characteristics have been assigned to the services.

Table 1- Bin types, capacities and costs

	Capacity	*Cost*
Type-1	10	20
Type-2	20	35
Type-3	30	50
Type-4	40	65

All experiments have been run on a laptop with CPU AMD Ryzen 7 7000 series with 3,8-5,1 GHz and 16 GB of RAM. The mathematical model has been solved by the software by using CPLEX 22.1 with a time limit of 3600 seconds.

In the next two paragraphs, the behavior and structural characteristics of the solutions obtained from the proposed model using the generated instances for each of the three topologies are presented.

8.1 Results with no fixed-contract

This part presents the results obtained when there is no fixed-contract, thus, the solver is free to choose the questions to be satisfied without any kind of constraint. The CPU indicates the computation time taken to solve the instance, expressed in seconds. This time includes the actual processing done by the processor and serves as a performance indicator for the model and the solver. A lower value indicates a faster solution time, while a higher value indicates a longer solution time. While, the term gap typically refers to the optimality gap, which measures the difference between the best known solution (or bound) and the current solution found by the solver during the optimization process.

The tables below highlight how costs are partitioned. As expected, transportation costs are the largest and have the greatest impact on the total cost. This is because transportation represents a substantial portion of operational expenses, especially if the network is large and shipping volumes are high. If this were not the case, it would mean either inconsistency in data generation or malfunctioning of the constructed transportation network. If, for example, hold costs had been the largest, it would have meant that goods spend more time blocked at a terminal rather than being transported, thus indicating a flow problem on the network.

Holding cost have a medium-high impact. Generally, they can have a considerable impact, particularly if inventory volumes are high and storage space is expensive to maintain. These costs include the value of the inventory, the space needed, and the management and maintenance of the terminal. The lowest costs are the ones for the usage of bins and service activation.

Usually, bin costs are associated with managing and utilizing containers for storing and transporting goods. If the company has an efficient container management system, this cost can be relatively low. Service activation costs usually do not have a significant impact. However, they can have a larger impact if the company is implementing a new network or system.

The results also show that all the optimal solutions found turn out to be profitable.

Table 2- Execution results for hyper-corridor, no fixed-contract (1)

Hyper-corridor, $|K^C|=0$

Instance	Total Profit	Service Cost	Bin Cost	Hold Cost	Transportation Cost	Total Cost
1	95229	820	1065	18480	12090	32455
2	65418	390	830	13440	6550	21210
3	84010	520	940	12880	9560	23900
4	74508	720	800	10980	9420	21920
5	77758	800	1160	10260	11470	23690

Table 3- Execution results for hyper-corridor, no fixed-contract (2)

Hyper-corridor, $|K^C|=0$

Instance	Non-Cont	OF	CPU	Gap	Bins used	Services used
1	50	62774	4.22	0.00	33	30
2	50	44208	5.06	0.00	28	21
3	50	60110	3.16	0.00	29	25
4	50	52588	2.58	0.00	25	25
5	50	54068	2.97	0.00	40	31

Table 4-Execution results for bipartite, no fixed-contract (1)

Bipartite, $|K^C|=0$

Instance	Total Profit	Service Cost	Bin Cost	Hold Cost	Transportation Cost	Total Cost
1	105731	1190	2200	11880	21710	36980
2	95796	1070	1820	14040	18170	35100
3	110633	1190	2110	12420	18830	34550
4	104572	1360	2360	10460	21910	36090
5	127246	1400	2650	12200	25120	41370

Table 5-Execution results for bipartite, no fixed-contract (2)

Bipartite, $|K^C|=0$

Instance	Non-Cont	OF	CPU	Gap	Bins used	Services used
1	50	68751	3143.23	0.00	68	53
2	50	60696	74.78	0.00	55	46
3	50	76083	52.09	0.00	65	54
4	50	68482	302.16	0.00	73	58
5	50	85876	126.77	0.00	77	62

Table 6-Execution results for grid, no fixed-contract (1)

Grid, $|K^C|=0$

Instance	Total Profit	Service Cost	Bin Cost	Hold Cost	Transportation Cost	Total Cost
1	57238	1080	2065	7840	19280	30265
2	63930	1280	2325	5960	18670	28235
3	65671	1430	2515	7180	20880	32005
4	68053	1410	2610	8120	22760	34900
5	71974	1270	2525	8540	21810	34145

Table 7- Execution results for grid, no fixed-contract (2)

Grid, $|K^C|=0$

Instance	Non-Cont	OF	CPU	Gap	Bins used	Services used
1	50	26973	93.19	0.00	68	48
2	50	35695	82.13	0.00	69	58
3	50	33666	91.91	0.00	74	63
4	50	33153	80.81	0.00	75	63
5	50	37829	52.09	0.00	70	57

8.2 Results with fixed-contracts

Instead, this part presents the results obtained by setting a number of 3 contracts that must necessarily be served. Note that the Objective Function value of the instances with fixed-contract shippers always provide a lower bound for their counterpart instances with non-contract shippers.

By fixing some contracts, in most cases the problem turns out to be infeasible. This is not surprising since, as stated in the section on data generation (Paragraph 14), it is difficult to generate data that survive different types of stresses by always resulting in feasibility.

The grid network, however, when contracts are set, has far fewer cases in infeasibility than other networks. The explanation for this may be in the fact that the grid turns out to be a more articulated network. The grid is the network that offers the highest number of links per number of nodes (links/nodes) in the network. Indeed, it has 72,73% more services than the hyper-corridor and 18.75% more services than the bipartite.

Table 8-Execution results for hyper-corridor, fixed-contract (1)

Hyper-corridor

Instance	Total Profit	Service Cost	Bin Cost	Hold Cost	Transportation Cost	Total Cost
1	95229	820	1065	18480	12090	32455
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

Table 9-Execution results for hyper-corridor, fixed-contract (2)

Hyper-corridor

Instance	Non-Cont	OF	CPU	Gap	Bins used	Services used
1	47	62774	2.97	0.00	33	30
2	47	na	1.39	na	0	0
3	47	na	0.22	na	0	0
4	47	na	0.42	na	0	0
5	47	na	0.20	na	0	0

Table 10-Execution results for bipartite, fixed-contract (1)

Bipartite

Instance	Total Profit	Service Cost	Bin Cost	Hold Cost	Transportation Cost	Total Cost
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	104572	1360	2390	10400	21940	36090
5	0	0	0	0	0	0

Table 11-Execution results for bipartite, fixed-contract (2)

Bipartite

Instance	Non-Cont	OF	CPU	Gap	Bins used	Services used
1	47	na	0.28	na	0	0
2	47	na	0.25	na	0	0
3	47	na	0.27	na	0	0
4	47	68482	1171.03	0.00	73	58
5	47	na	0.86	na	0	0

Table 12-Execution results for grid, fixed-contract (1)

Grid

Instance	Total Profit	Service Cost	Bin Cost	Hold Cost	Transportation Cost	Total Cost
1	57238	1080	2065	7840	19280	30265
2	63930	1280	2325	5960	18670	28235
3	65671	1430	2515	7180	20880	32005
4	68053	1410	2610	8120	22760	34900
5	0	0	0	0	0	0

Table 13-Execution results for grid, fixed-contract (2)

Grid

Instance	Non-Cont	OF	CPU	Gap	Bins used	Services used
1	47	26973	112.56	0.00	68	48
2	47	35695	61.20	0.00	69	58
3	47	33666	118.73	0.00	74	63
4	47	33153	89.69	0.00	75	63
5	47	na	0.36	na	0	0

8.3 Observations

The line of research proposed in this thesis aims to be extended by other research to better motivate this scientific study. The results shown are to serve as a reference point and starting point for future research.

What was expected at the beginning of this research, from the comparison with the results of the non-integrated problem, was:

- The integrated problem provides better solutions.
- The integrated problem provides a solution with lower cost.
- The integrated problem manages better the loading units and reduces the number of vehicles routed (or bins used).

Expectations have been met and to better highlight the advantage of Bin Packing, it is appropriate to analyze the steps of a possible solution.

Representing all the end services associated with each demand would be complicated, but fortunately it is sufficient to represent what happens locally, to an accepted demand, to understand which is the logic of the solution proposed by the software.

Let's consider Instance number 1 of the grid network. Among all the demands which have been accepted, there are Demand 19 and Demand 30. Attention is now directed toward a graphical analysis of the behavior of these two commodities.

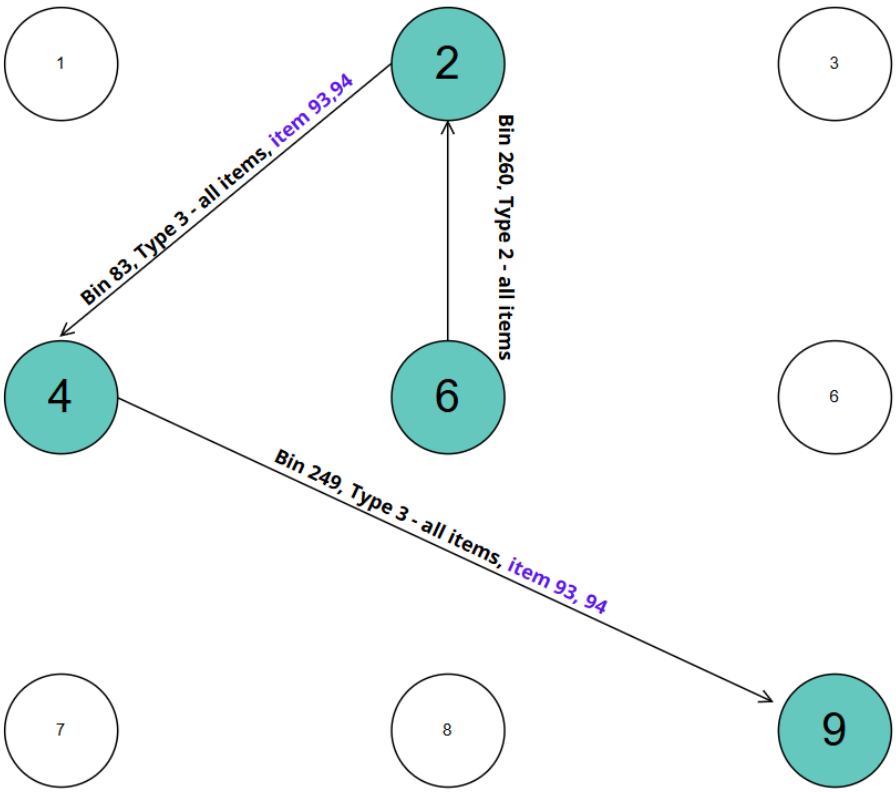


Figure 12 - Demand 30 allocation, from grid 1

Demand 30 (Figure 12) starts at terminal 5 and arrives at terminal 9 after exactly 14 periods (it departs at t=3 and arrives at t=3). The items to be transported are item 95 (volume 3), item 96 (volume 2), item 97 (volume 2), item 98 (volume 2), and item 99 (volume 6). The transport is carried out using bin 259 (capacity 20) from terminal 5 to terminal 2, bin 83 (capacity 30) from terminal 2 to terminal 4, and bin 249 (capacity 30) from terminal 4 to terminal 9. All the items are loaded together because their total volume fits within the capacity of a single bin. The items in purple belong to Demand 29 and are carried in the same bins used to carry Demand 30.

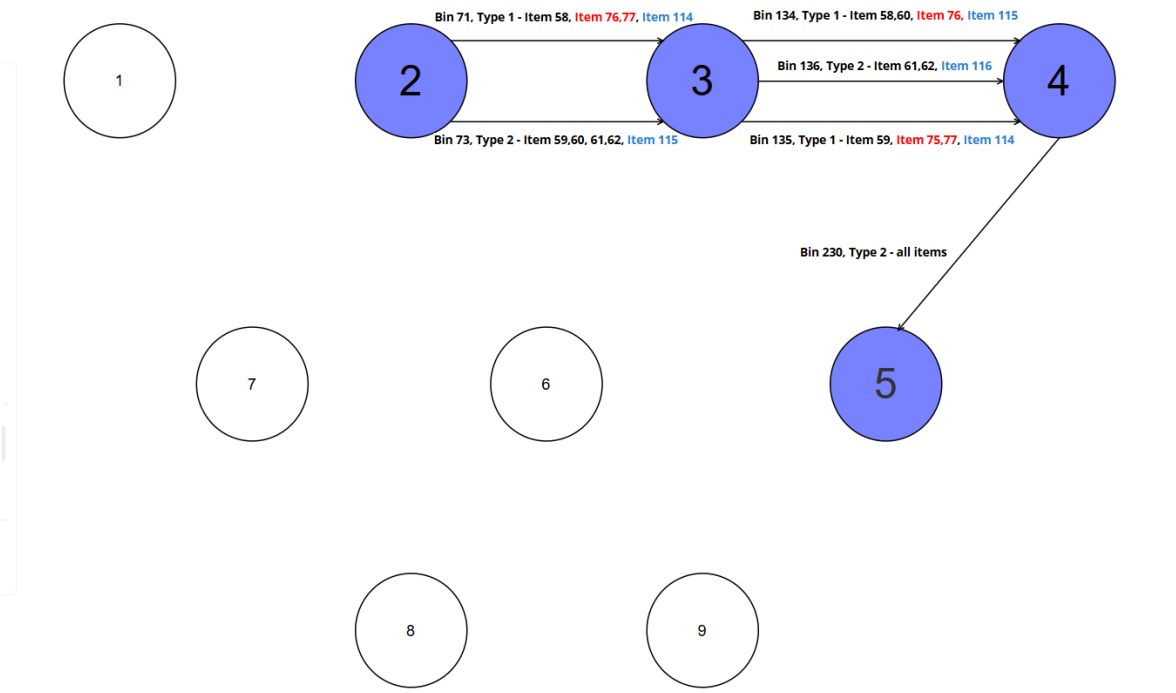


Figure 13- Demand 19 allocation, from grid 1

Demand 19 (Figure 13) behaves differently. It starts at terminal 2 at t=7 and arrives at terminal 5 at t=14, requiring the transport of items 60 (volume 3), 61 (volume 2), 62 (volume 7), 58 (volume 3), and 59 (volume 2). It is interesting to note that:

- The items are not placed in the same bin due to capacity constraints, and thus items related to the same demand are separated in different bins.
- Some of the items are transported in bins that also contain items from other demands, all in an effort to optimize the used capacity of each bin and minimize the cost of the bins used. In red there are the items belonging to Demand 23 and in blue are the items belonging to Demand 36.

Table 14 shows how, merging multiple demands together in the same bin, can get the bins to a very high usage rate, even as high as 100%.

This open the chance to use fewer bins to satisfy given number of demands, and consequently the possibility of satisfying more demands since there are still bins available.

In addition, using bins more efficiently reduces the fixed costs of using the bin and

increases income from satisfying more demands, compared to the non-integrated approach.

Table 14 - Percentage of used capacity with the integrated approach

	Bin 71	***Bin 73***	***Bin 134***	***Bin 135***	***Bin 136***	***Bin 230***
Max Capacity	10	20	10	10	20	20
Used volume by single demand	30%	70%	60%	20%	45%	85%
Overall used volume	100%	85%	100%	90%	90%	85%

8.3.1 Differences with classical Service Network Design

When using a classical formulation for SND, the software simply blindly chooses the services to associate with each demand, without considering how each service is composed. For this reason, the problem has always been divided into two sub-problems and subjected to two different implementations. As a first step, the classical SND model is executed, and then the provided solution is reworked again considering the constraints on the bins.

Unfortunately, this approach is inefficient. The result of this double operation could be unfeasible. In fact, even if a demand is associated with a service, there could be a mismatch between these two, leading the second execution to an unfeasible result. An example might be that there is a question consisting of two items, with a total volume of 20. This is associated with a service consisting of two bins and with a total volume of 20. The discrepancy originates when the volume of the first item is 8 and that of the second is 12, while the capacity of the two bins is 10. Clearly, while the 8 item fits into both bins, the 12 item does not fit with either. So while certainly the proposed solution made sense, in terms of total volume and capacity, when considering the fitting between items and bins, this solution might be infeasible.

Second, even if the solution proposed by the first execution turns out to be feasible for the second, it will surely be a sub-optimal solution, and thus with much worse Objective Function than that obtained by an integrated approach (optimal solution).

Beyond the total value of the Objective Function, the solution is not optimal because it does not make efficient use of the bins. In fact, it could happen that bins remain half empty, and this in addition to being an operational cost for carriers represents a missed potential revenue.

The fact that the classical model can spill over into these situations shows that there is a need to refine existing models so that the best outcome is achieved. The result provided by the classical SND is misleading and turns into a user problem, taking wrong decisions and facing operational and logistic challenges.

To make a comparison between the obtained solutions, it is necessary to execute both the classical SND and feed it to a check on the bins. Directly comparing the solution proposed by the classical SND (without check on bins) and that proposed by the integrated SND with Bin packing is misleading for two main reasons.
First, the solution proposed by classical SND does not consider bins, and therefore neither does the costs associated with using bins, thus leading to a higher profit than SND+BP. The two results cannot be directly compared because there are costs which are included in SND+BP and that, although real, are neglected in classical SND.
Second, the solution proposed only by an initial execution, without any kind of bins ascertainment, might be unfeasible. And thus, there is no point in comparing a result that later turns out to be unfeasible with one feasible for certain.

Table 15 - Inefficiency of classical SND, KPI: Objective Function

Instance 1, Grid network	**Integrated SND+BP**	**Classical SND**	**BP fed by classical SND**
OF	26973	60926	-74555
CPU	93.19	5.55	2523.22

An extreme example of this case is given by the grid 1 case (Table 15). From the

primp launch of the instance on a model of classical SND, the OF will be much better than that of SND+BP. But then, when this solution is reworked by considering the bins and their costs, not only the value of the objective function will be obviously worse than that of the SND+BP (which represents the absolute optimum), but it will even become extremely negative. The blindly chosen services from classical SND thus entail a real loss for the user.

The point being made is that, when the capacity of the individual bins that constitute a service is not considered, the proposed solution could lead to both subsequent breaks of bulk and inefficient use of the bins that make up the service. The break of bulk process involves typical operations such as unloading, transferring, and reloading goods into different vehicles or containers. This process can increase costs and delivery times, as well as the risk of damage or loss, since it requires additional handling of the goods. The reason why, on the other hand, there are fewer breaks of bulk in integrated approach is that the Bin Packing model focuses on packing bins to their maximum capacity, thus reducing the need to split loads into smaller parts or transfer them between multiple vehicles.
As a result, the issue goes beyond the optimization of theoretical solutions and affects physical processes. Both breaks of bulk and underutilized capacity represent a logistical challenge and a loss of potential profit, thereby increasing costs.

9 Conclusion

An increasing amount of research is dedicated to studying integrated Service Network Design problems, which involve combining traditional Network Design challenges with other optimization issues. This emerging focus is driven by the aim to close the gap between academic research and practical applications, as well as by improvements in optimization techniques and advancements in computing power. Despite this, integrated problems are computationally challenging. To support the use of integrated approaches, it is essential to evaluate the potential cost savings and efficiencies that can be gained by solving the integrated problem directly, rather than breaking it down and handling each component separately. Historically, operational researchers have approached these problems separately, sacrificing overall optimization for ease of solving each component individually. Even when exact methods are employed for solving individual problems and their partial solutions are then combined, the result is a sub-optimal solution for the integrated (global) problem. A suboptimal solution is a result that does not represent the best possible outcome for a given problem. In optimization contexts, this means the solution may be feasible and meet the basic requirements but falls short of being the most efficient, cost-effective, or highest-quality option available. Conversely, merging two or more complex NP-hard problems increases the computational load significantly, but it generally yields superior results compared to addressing each problem separately. It is crucial to advocate for an integrated approach and to quantify the potential improvements gained by addressing the combined problem as a whole rather than independently.

This research project, after introducing general concepts about Service Network Design, proposes an integrated model of it and describes the generation of instances to execute the model on AIMMS software.

The work done has shown what is a typical outcome of integrated Service Network Design with Bin Packing constraints. As output of this process, the issues with the traditional approach are highlighted, demonstrating why integrating Bin Packing proves to be more beneficial. The fact that the classical model can lead to disadvantageous situations underscores the need to refine existing models to achieve the best possible outcome. The outcome provided by the classical SND can be

misleading, resulting in user problems, wrong decisions, and operational and logistical challenges.

Acknowledgements

English:

I would like to express my gratitude to the Politecnico di Torino; coming to this city has been a unique opportunity for me, one that I will remember as one of the most wonderful periods of my life. I extend my acknowledgment to my supervisors Professor Guido Perboli and Doctor Sara Khoradapasti.

Thanks to the people around me; I feel fortunate to have ***each*** of you in my life. Every one of you makes me a richer person, and I hope you will continue to flatter me with your simple existence, just as I wish to do the same for you. I love you.

Among these people, a special acknowledgment goes to my parents,
Enzo and Gabriella.

Today, I am immensely happy, and I thank myself for making this happiness feel like something natural rather than rare. I could have become anything, taking countless paths, but I chose the right ones, the ones that led me here. I am proud of myself.

Italiano:

Vorrei esprimere la mia gratitudine al Politecnico di Torino; venire in questa città è stata per me un'opportunità unica, che ricorderò per sempre come uno dei periodi più belli della mia vita. Rivolgo un ringraziamento anche ai miei relatori, il Professore Guido Perboli e la Dottoressa Sara Khoradapasti.

Grazie alle persone che mi circondano; mi considero estremamente fortunata ad avere ***ognuno*** di voi nella mia vita. Ciascuno di voi mi rende una persona più ricca, e spero che continuerete a lusingarmi con la vostra semplice presenza, come io desidero fare lo stesso per voi. Vi voglio immensamente bene.

Tra queste persone, un ringraziamento speciale va ai miei genitori, Enzo e Gabriella.

Oggi tocco il cielo con un dito e ringrazio me stessa, perché la felicità che provo non è un'eccezione, ma una costante che ho sempre saputo mantenere viva. Sarei potuta diventare qualsiasi cosa, prendere infinite direzioni, ma ho saputo scegliere quelle giuste, quelle che mi hanno portata fino a qui. Sono fiera di me.

References

ALICE. (s.d.). *ALICE Roadmap to Physical Internet.* Tratto da etp-logistics.eu: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.etp-logistics.eu/wp-content/uploads/2022/11/Roadmap-to-Physical-Intenet-Executive-Version_Final-web.pdf

Bilegan, & Ioana et al. (2015). Revenue management for rail container transportation. *EURO Journal on Transportation and Logistics*.

Bruni, Crainic, & Perboli. (2023). Bin Packing Methodologies for Capacity Planning. *Contributions to Combinatorial Optimization*.

Constraint Declaration. (2024). Tratto da AIMMS: https://documentation.aimms.com/language-reference/optimization-modeling-components/variable-and-constraint-declaration/constraint-declaration-and-attributes.html

Côté, Guastaroba, & Speranza. (2017). The value of integrating loading and routing. *European Journal of Operational Research*.

Crainic. (2000). Service network design in freight transportation. *European Journal of Operational Research*.

Crainic, Fomeni, & Rei. (2021). Multi-period bin packing model and effective constructive heuristics for corridor-based logistics capacity planning. *Computers & Operations Research*.

Crainic, Gendreau, & Gendron. (2021). *Network design with applications to transportation and logistics*.

Crainic, T. (2024). Consolidation-based Transportation Planning: The Service Network Design Methodology.

Crainic, T., & Rei, W. (2024). *50 Years of Operations Research for Planning Consolidation-based Freight Transportation*.

Crevier, Cordeau, & Savard. (2012). Integrated operations planning and revenue management for rail freight transportation. *Transportation Research Part B: Methodological*.

Data Initialization. (2024). Tratto da AIMMS: https://documentation.aimms.com/language-reference/preliminaries/language-preliminaries/data-initialization.html

Flamand, Iori, & Haouari. (2023). The transportation problem with packing constraints. *Computers & Operations Research*.

Hewitt, & Lehuédé. (2023). New Formulations for the Scheduled Service Network Design. *Transportation Research Part B: Methodological*.

Jarrah, Ahmad, Johnson, & Neubert. (2009). Large-scale, less-thantruckload service network design. *Operations research*.

Larsen, E., Frejinger, E., Bisaillon, S., & Cordeau, J.-F. (2023). Pseudo-random Instance Generators in C++ for Deterministic and Stochastic Multi-commodity Netwrok Design Problems.

Li, Lin, Negenborn, & De Schutter. (2015). Pricing Intermodal Freight Transport Services: A Cost-Plus-Pricing Strategy. *Computational Logistics*.

Luo, Ting, Long Gao, & Yalçin Akçay. (2016). Revenue Management for Intermodal Transportation: The Role of Dynamic Forecasting. *Production and Operations Management*.

Parameters Declaration. (2024). Tratto da AIMMS: https://documentation.aimms.com/language-reference/non-procedural-language-components/parameter-declaration/index.html

Ricciardi, Storchi, & Crainic. (2009). Models for Evaluating and Planning City Logistics Transportation Systems. *Transportation Science*.

Riessen, Bart van, Negenborn, & Dekker. (2017). The Cargo Fare Class Mix problem for an intermodal corridor: revenue management in synchromodal container transportation. *Flexible Services and Manufacturing Journal*.

Set Declaration. (2024). Tratto da AIMMS: https://documentation.aimms.com/language-reference/non-procedural-language-components/set-declaration/index.html

Solving Mathematical Programs. (2024). Tratto da AIMMS: https://documentation.aimms.com/language-reference/optimization-modeling-components/solving-mathematical-programs/index.html

Taherkhani, Bilegan, Crainic, Gendreau, & Rei. (2022). Tactical capacity planning in an integrated multi-stakeholder freight transportation system. *Omega*.

Talluri, & Kalyan and Garrett van Ryzin. (2004). Revenue Management Under a General Discrete Choice Model of Consumer Behavior. *Management Science*.

Tawfik, & Christine and Sabine Limbourg. (2018). Pricing Problems in Intermodal Freight Transport: Research Overview and Prospects. *Sustainability*.

Text Reports and Output Listing. (2024). Tratto da AIMMS: https://documentation.aimms.com/language-reference/data-communication-components/text-reports-and-output-listing/index.html

Variable Declaration. (2024). Tratto da AIMMS: https://documentation.aimms.com/language-reference/optimization-modeling-components/variable-and-constraint-declaration/variable-declaration-and-attributes.html

Wang, & Yunfei et al. (2016). A Revenue Management Approach for Network Capacity Allocation of an Intermodal Barge Transportation System. *Computational Logistics*.

Wikipedia Contributors. (2024). *AIMMS*. Tratto da Wikipedia: https://en.wikipedia.org/wiki/AIMMS

Wikipedia Contributors. (2024). *Bin packing problem*. Tratto da Wikipedia: https://en.wikipedia.org/wiki/Bin_packing_problem

Zhu, Endong, Crainic, & Gendreau. (2014). Scheduled service network design for freight rail transportation. *Operations research*.