“Enhanced data management techniques for real time logistics planning and scheduling”

Deliverable D5.1: State of the Art in Negotiation and Re-optimization Techniques in Logistics

Dissemination level: ☒ Public ☐ Confidential, only for members of the consortium (including the Commission Services)

Version number: 1.0

Submission deadline: 31/03/2019

www.logistar-project.eu

DOCUMENT INFORMATION

Authors

Name	Organisation
Filippo Bistaffa	CSIC
Dave de Jonge	CSIC
Jordi Levy	CSIC
Pedro Meseguer	CSIC

Reviewers

Name	Organisation
Antonio Masegosa	Deusto
Naia Merino	Deusto
Enrique Onieva	Deusto
Bastien Pietropaoli	UCC

Document control

Version	Date	Comment
0.1	08 / 04 / 2019	First version
0.2	16 / 04 / 2019	Second version
0.3	26 / 04 / 2019	Comments from Deusto and UCC included
1.0	20 / 05 / 2019	Added section on Transshipment

Document approval

Version	Date	Partners
1.0	24 / 05 / 2019	All partners

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

BASIC PROJECT INFORMATION

Horizon 2020 programme

H2020 - Mobility for Growth- 5-2-2017. Innovative ICT solutions for future logistics operations

Grant Agreement No. 769142

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

TABLE OF CONTENTS

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

Executive Summary

In this document we discuss the state-of-the-art in the topics of Automated Negotiations and Re-optimization applied to the logistics domain.

The topic of Automated Negotiation in general has mainly focused on purely abstract problems and much less attention in the literature has been given to the application of this technique to real-world domains. Although some papers have been published about industrial applications of Automated Negotiations, only very few of them are related to logistics. We have been able to find eight such papers, which we discuss in this deliverable.

The two papers that are arguably the closest related to LOGISTAR are [48] and [23]. In [48] the authors describe a case study in which they explore the negotiation of backhauling and co-loading opportunities between a single 4PL company and several 3PL companies, while [23] discusses the possibility of package delivery companies to exchange packages with one another, resulting in lower costs (and thus higher profits) for each of them. Somewhat less relevant, but still very much related to LOGISTAR is [11] which describes negotiations between logistics companies and a container terminal of a port, in which they negotiate the arrival times of the trucks at the terminal. In [41] and [35] the authors treat a very similar scenario as in [48], except that they now use auction mechanisms instead of true negotiations. This severely limits the possibility to agree on complex co-loading and backhauling solutions and are therefore less applicable to LOGISTAR. Finally, we discuss [2], [7] and [18], even though they involve a different 'kind' of negotiations, in the sense that the agents there are assumed to be working together to find a globally optimal solution, rather than representing self-interested companies such as in LOGISTAR. We still discuss these papers, because the techniques applied may still be relevant, even though they are used with a different goal than ours. [2] describes a system for logistics planning of military operations. They assume every vehicle of the army is represented by its own agent, and all agents negotiate with one another in order to find the global solution that can be executed in the shortest possible time and with the lowest costs. In [7] the authors present a model in which each truck, as well as each order owned by a single company is represented by an agent. These agents work together to find the optimal solution for that company. [18] involves a decision support system for air-freight planning. It consists of two agents, one trying to find the financially optimal solution, while the other is occupied with minimizing risks. The two agents then negotiate with each other to find the optimal balance between the two criteria.

Regarding the topic of re-optimization, we have followed the approach of classical optimization using a constraint programming perspective. A lot of work has been devoted to the academic version of the problem of interest, the vehicle routing problem (up to the point that reducing the quantity of references –in order to provide an affordable number– has been an important task when producing this document). In the incoming pages, the reader will find a lot of technical material regarding cost functions, constraints and propagation strategies. He or she will also find that the classical exhaustive search performed in the CP framework is unfeasible, given the NP-hardness of the problem at hand. Then, local search and meta-heuristics appear as the basic elements that allow to efficiently traverse the search space (at the extra cost of losing the optimality proof, or even the optimality itself).

We finalize this document with a review about ridesharing. This is a problem that has strong similarities with the problem of co-loading addressed in this project.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

1. Introduction

The main goal of the WP5 is to provide sound and quick reactions in front of unexpected events and sudden changes; a new valid plan is proposed that accommodates these changes in the current solution such that (i) it remains valid, and (ii) a minimal number of agents needs to alter their individual plans. Negotiation among involved agents seems to be a key technique at this point.

With this purpose, this deliverable summarizes the work done on automated negotiation and re-optimization techniques in the area of logistic networks. One of the main results of task 5.1 is the state-of-the-art review of global optimization approaches when some of the problem parts change. This document will be an input for task 5.2 (Negotiation techniques) and task 5.3 (Re-optimization techniques), in the context of logistic systems.

Behind this introduction, the rest of the document is structured as follows. We provide an overview of Automated Negotiation in Logistics, a relatively new field addressing an old problem with original approaches in Section 2. Negotiation, Auctions and Argumentation are some of the techniques discussed in this part. We consider the issue of Re-optimization in Logistics, with the classical approaches of Integer Linear Programming and Constraint Programming in Section 3. In addition, we describe the role of Local Search for this kind of problems, and the research done on dynamic versions of VRP (Vehicle Routing Problem). We also consider the role of transshipment and its impact in the intermodal transport strategy in Section 4. Finally, we provide a short description of ride-sharing systems, also facing dynamic scenarios in Section 5. We present our conclusions from this study in Section 6.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

2. Automated Negotiations in Logistics

Automated Negotiation is a process in which multiple pieces of software (so-called 'agents') jointly need to find a solution to some problem, while each of these agents has its own individual goals, which may be partially conflicting with the other agents' goals.

A typical example would be the case that two or more logistics companies have a number of orders to fulfill (i.e. truck loads to deliver). Although each company could simply deliver its own orders, this may not be the most efficient or profitable solution. Therefore, these companies could mutually profit if they were able to somehow exchange (parts of) their loads and benefit from the resulting co-loading and backhauling opportunities. However, since each company ultimately only aims to maximize its own profit, one cannot implement a central algorithm that will find the 'best' or 'fairest' solution for everyone. After all, each company may have its own views on what constitutes a 'fair' solution. If any company does not agree with a proposed solution, there is no way to force that company to obey it. Furthermore, even if there was a certain notion of fairness that everyone would agree on, then still it is unlikely that such a solution could be found by an impartial agent, because it would require extensive strategic knowledge about the individual companies. Such companies would, in general, not be willing to share such information.

Therefore, in order to be able to profit from co-loading and backhauling opportunities, each company must have its own agent that represents the company's interests and engages in a negotiation dialogue with the other companies' agents. Such a negotiation dialogue consists of agents proposing solutions to one another. Whenever an agent receives a solution proposed by another agent, the receiving agent should evaluate whether or not that solution is profitable enough for the company it represents. If this is indeed the case the agent will accept the solution, and once the proposal is accepted by all agents involved in the solution, it becomes a commitment (although this may depend on the details of the negotiation protocol in use). On the other hand, if the proposal is not profitable enough, the agent may reject the proposal, and optionally make a counter-proposal.

Although Automated Negotiation in general has drawn ample attention in the Artificial Intelligence community, it has mainly focused on purely abstract problems. The application of this technique to real-world domains is still a largely underdeveloped topic. Furthermore, among those studies that did focus on industrial applications, only a few were related to logistics. In this document we give an overview of those few studies on Automated Negotiations applied to logistics that we have been able to find.

It is important to point out that two of those papers actually focus on an auction-based mechanism, rather than true negotiations, which is less applicable to LOGISTAR, because it only allows the agents to make agreements on a price, rather than on the question of which company is going to deliver which loads. Furthermore, three other papers we discuss apply a model in which several agents negotiate on behalf of only one single company, which also makes them less relevant to LOGISTAR.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

2.1. Negotiation of Time Slots for Trucks Arriving at a Container Terminal of a Port

In [11] the authors discuss the application of Automated Negotiations at a port's container terminal. They introduce "a planning and scheduling system that carries out the negotiation between the port terminal and truck companies to negotiate time slots for arriving at the terminal".

They aim to solve the problem that, upon arrival at a port, trucks are often waiting in long queues, due to the fact that the port only has a limited number of serving cranes, and there is limited parking capacity, which is very costly.

Their system allows existing tools for modeling truck companies and container terminals to cooperate. They tested their proposed system in a distributed simulation environment in which the trucking companies and the port handling system were built as separate simulation models, combined with a planning system that was available as a separate application, interfacing with the simulation models.

In their system, every party (truck company or terminal operator) is represented by its own agent. The agent of the terminal operator and the agents of the truck companies will then negotiate an arrival time for the trucks that is acceptable for both sides. A truck company's agent starts the negotiations by making a request, specifying a desired arrival time at the terminal. However, since the container terminal has limited capacity, it may happen that many truck companies request the same time slot. The terminal operator agent, therefore, needs to check availability of resources in the requested time slot. If there are sufficient resources in the requested slot, the terminal agent confirms the booking to the truck agent. If the terminal is fully booked at the requested time slot, the terminal agent will propose another slot, taking into account the band-width for negotiation that was sent with the request, the estimated transportation time based on the truck's departure information, and information about the deep-sea vessel on which the container has to be loaded or from which the container has to be unloaded. On the other hand, the truck company's agent needs to take into account its own constraints, such as the currently available trucks, the currently available drivers, and the time limits for the trucks and drivers (e.g. not driving at night). When a request for a specific timeslot has been rejected and a new timeslot has been proposed by the terminal, the truck company can accept the proposed time slot or try to get another slot which suits him better.

2.2. Automated Supply Chain Negotiations in Transportation Logistics

In [48] the authors present a case study for the application of agent-mediated negotiation techniques in transportation logistics. This case study is very similar to LOGISTAR, in that they explore the negotiation of backhauling- and co-loading opportunities between logistics companies, although in their case they focus on a one-to-many negotiation in which one 4PL company negotiates with several 3PL companies about which of those 3PL companies will transport which loads.

In this paper a 3PL provider is defined to be a company that has its own truck fleet, and that plans its own movements, whereas a 4PL provider is defined as a company that does not have its own transport capacity. 4PL providers receive large transport orders from shippers and then distribute them among a set of 3PL companies. This process involves direct negotiation with 3PL companies and often entails breaking up large orders or bundling orders for partial loads. The 3PL companies

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

are assumed to form a closed group of around 5 to 10 partners that are trusted by the 4PL provider, based on a history of good business relationships. In other words, the negotiations were not open for any other random company to join.

The authors focus on daily outsourcing (i.e. *spot orders*), because they argue that this is the main area that could be automated through agent technologies, "since it is unlikely that automated agents can be entrusted to take higher-level or strategic decisions".

They identify three main parameters that can be negotiated, namely,

1. the price,
2. the time windows for delivery, and
3. the bundling of loads from different shippers.

Although time windows are usually considered fixed, the paper mentions that several existing shippers have indicated they would be willing to be more flexible if that would result in savings to them.

Furthermore, regarding to bundling and pricing, they identify two main characteristics that are particularly important. Namely,

1. the percentage a load takes from the total volume of a truck, and
2. the *fruitfulness* of the region the order originates from or is to be delivered, by which they mean the likelihood that there will be return freight from that region.

The authors consider that orders can exhibit utility dependencies of two types:

1. Complementary dependencies, if the orders can be transported by the same truck
2. Incompatibility dependencies, if orders cannot be bundled together, due to a mismatch between their delivery time windows and/or capacity.

In their model, dependencies are represented in the form of a utility graph, which encodes the complementarity and incompatibility relations between orders. This model is based on previous work in which a general framework was proposed to handle this type of dependencies, but now applied to the logistics setting.

The authors have built a software tool to enable planners to compute (close to) optimal bundling of different sets of orders and to explore different scenarios, by changing the constraints (e.g. time window constraints) and the information shared between parties.

The model was tested on historical data of Vos Logistics, which is one of the larger logistics service providers in Europe, with (in 2006) over 3,000 trucks, 10,000 trailers and containers, 5,000 employees and an annual turnover of around 800 million euro. They analyzed the gains that can be achieved by bundling transportation orders and conclude that using more efficient bundling the average truck

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

utilization (load) increases from 18% to 39%. Using actual carrier price data, this results in savings of 19% in terms of price per unit load.

Finally, the authors remark that they only focused on the allocation of orders to 3PL providers and not on the optimal planning of the truck movements under such allocations. They mention that for their approach to work well in practice, the planning of transportation within each company would need to be automated as well and connected to the negotiating agent.

2.3. Auction-Based Allocation of Loads in Transportation Logistics

In [41] and [35] the authors treat a very similar scenario as in [48], except that they now use auction mechanisms instead of negotiations. This means that the 3PL providers only have the option to offer a certain price according to some auction protocol, upon which the 4PL provider acts as an auctioneer and determines a 'winner'.

These papers describe the software component that allows both humans and software agents to make bids on an auction of truck loads. The winners are those participants that ask the lowest price for transporting the loads under auction. The papers, however, mainly describe the interface for the auction mechanism, and do not discuss in much detail the algorithms used by the agents to choose their bids. In fact, these bids are basically randomly generated.

Furthermore, the auction-based approach seems less relevant to LOGISTAR, because it assumes there is one 'owner' of the orders (the 4PL provider) which 'sells' them to the 3PL providers. In contrast, in LOGISTAR every partner is essentially the owner of a number of orders, which will be exchanged among themselves.

2.4. Multi-Agent Logistics Planning for Military Operations

In [2] the authors discuss Automated Negotiations for the logistics planning of military operations. They discuss a scenario in which a set of military vehicles jointly aim to transport a number of loads from their respective origins to their respective destinations in the shortest possible time. They assume that each of these vehicles is represented by a single agent. Two locations can be connected by a sequence of routes, where each route can potentially involve a different mode of transportation. Hence, a single job may involve traversing routes of multiple modes. Therefore, the agents need to collaborate to formulate a distribution plan, by means of automated negotiation. More specifically, the paper compares two types of Multi-Agent Systems, which they call a Flat Communication Structure (FCS) and a Hierarchical Communication Structure (HCS), respectively.

The authors begin by looking at the flat agent structure. Each agent is aware of its capabilities as a means of transport, its current schedule, the locations it can travel, and the other agents in the system with which it can communicate. In this flat structure each individual agent is capable of communicating with all other agents in the system. Each time a new job is announced to the agents, each agent can form a bid to transport the load directly from origin to destination, or if it is unable to reach the destination, it may transport the load to an intermediate location and subcontract the remainder of

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

the journey. Once the agent decides to subcontract, it will send a new sub-job announcement to all other agents outlining the requirements to transport the given load. The agents that receive the announcement will then undergo the same bidding process.

The problem with this FCS, is that as the number of agents increases, so too does the communication overhead, increasing the time taken to produce a solution to the distribution problem. In fact, the computational complexity increases exponentially.

Therefore, the authors propose to group the agents into a two-tiered hierarchical structure (the HCS), where all agents in a given group have similar capabilities. A new type of agent, which they call a transport agent, acts as the head of an agent group and directs communication down the hierarchy. Bidding, communication and hence subcontracting are performed at the transport agent level. The transport agents are responsible for broad planning through negotiation and select the most suitable agent in their group to participate in a bid, while the low-level agents perform detailed planning of their respective transportation requirements. This allows agent communication to be regulated, reducing redundancy in communication as encountered in the flat structure.

Testing was completed in two parts; one assessing computational complexity and the other testing the quality of the solution. Through multiple trials, it was found that the application of hierarchy vastly improved the computational performance of the system, without compromise to solution quality.

In general, the FCS performed poorly, growing at a near exponential rate. Due to this exponential growth, calculating a result became unfeasible for all except the smallest problem instances. In contrast, the HCS demonstrated improved scalability, suggesting that it is better suited for handling the complexity of real-world scenarios. When a solution was found, both communication structures produced the same solution however the time required to reach such a solution varied.

Finally, the quality of the solution was also tested against a centralized approach for distribution planning, known as the Greedy Distribution Algorithm (GDA). Instead of using a multi-agent system, the GDA contains a central entity that considers the constraints of the planning scenario and formulates a distribution plan by applying heuristics. The quality of the solutions was measured by the time difference between the start of the first movement and the end of the last movement. The HCS was able to produce better quality solutions when compared with the GDA, while remaining comparable in computational performance.

2.5. Argumentation-based Negotiations for Air-freight Planning

In [18] the authors present an argumentation-based multi-agent decision support system for air-freight planning. A key component of their system is an agent argumentation mechanism that allows decision support agents to discuss, argue, and come to a compromise in order to derive well-explained freight planning solutions.

Their system collects and monitors important information, such as news and weather, and political developments in risk areas, from various sources including airline agent platforms, third party websites, and clients. Furthermore, an argumentation mechanism is implemented for determining the best freight plan based on the principles of both cost minimization and risk reduction.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

Two agents are defined, namely the financial agent (FA) and the risk management agent (RMA). During the freight planning process, each of the two agents first proposes a freight plan according to their own goals and criteria: the FA plan is based on cost-minimization, whereas the RMA plan is based on risk reduction. If the two paths are not the same then an argumentation process is executed, where the two agents challenge the other's decision, exchange dialogues, and provide evidence justifying their proposed freight path. This continues until the difference is resolved. The output is a freight plan with well-explained rationales.

However, this work is different from LOGISTAR, because the authors assume a purely cooperative approach, in which all of the proposed agents belong to one single organization and jointly work toward a common objective.

2.6. Multi-agent Based Dynamic Load/Truck Planning

In [7] the authors present a multi-agent based platform for the negotiation of optimal assignments of orders to trucks. However, again, this work is different from the negotiations in LOGISTAR, because they assume all agents belong to a single organization. In their model, each truck is represented by a single agent, and each order is also represented by a single agent. In order to find a near-optimal solution for the assignment of orders to trucks, the order-agents engage in a kind of negotiation with the truck-agents.

2.7. Package Delivery

In [23] the authors describe an algorithm that can be used by package delivery companies to negotiate with one another who will deliver which package. Just as in LOGISTAR, each of these companies is considered a competitor to the others, so each has its own private profit to maximize. The problem of exchanging packages between postmen is modelled as a multi-agent variant of the Traveling Salesman Problem, and proposed algorithm is based on a Branch & Bound tree search to explore the space of possible agreements.

Although this is a highly abstracted version of a real-world problem, the end of the paper does suggest which adaptations should be made to make it suitable for real applications. Therefore, we think that these suggestions would be the ideal starting point for the implementation of negotiating agents in LOGISTAR.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

3. Re-optimization in Logistics

In the academic literature, the formal version of the main problem addressed in this project is called the *Vehicle Routing Problem* (VRP) [47, 27], and it is a kind of generalization of the famous TSP (formally introduced below). VRP is defined as follows. We consider a set of n customers that have to be served by a set of m vehicles or trucks. Each customer must be visited by exactly one vehicle. Customer i has demand r_i, and the sum of demands of customers assigned to vehicle k must be less than the vehicle capacity Q_k (if a customer has a demand greater than that capacity, we can allocate dedicated vehicles and only the *leftover amount* is considered here). All vehicles begin and finish their route at the same single depot. The objective is to minimize the sum of travel costs, where the generalized cost of travelling from i to j is c_{ij}. Here, we assume a symmetric ($c_{ij} = c_{ji}$) cost between any pair of points.

This formulation implicitly assumes that every customer receives the same type of good. It is not difficult to extend this formulation to different types of goods. This generalization does not make the problem theoretically more difficult. We can also consider the loading problem. In the simplest case, we assume that goods are loaded in the inverse order they will be unloaded. This constraint may seem too simple but it has a clear practical importance: a customer may refuse its package if it has to move packages of others (because liability issues, for instance), and this may cause the complete failure of the delivery route assigned to that vehicle.

Solving optimally VRP is NP-hard (it includes the TSP as special case, with a single route of infinite capacity). For problems of practical size, computing VRP optimal solutions can be too time consuming. This is the main reason to solve this kind of problems using local search. These approaches have been followed by several researchers, in order to avoid the theoretical limitations of current computing technology. In other words, in order to find a good –not necessarily optimal– solution within a reasonable computing time.

There are two variants of the VRP of interest for this project:

- Each vehicle k has a limited capacity and the total demand for that vehicle cannot exceed this capacity. This version is called the capacitated vehicle routing problem (CVRP).
- Each customer i requests to be served in a given time interval $[a_i, b_i]$ (= [earliest time, latest time]). The problem is named the vehicle routing problem with time windows (VRPTW). The addition of time constraints makes the problem much more difficult: even finding a feasible –not necessarily optimal– solution is NP-hard.

Additionally to the delivery time windows, we can also consider some extra constraints, in order to represent real-life conditions. Some of these constraints could be:

- A maximum time or length for every route, or between every pair of consecutively visited customers.
- A customer order r_i may be split $r_i = r'_i + r''_i$, and carried on separate vehicles.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

- Due to physical access restrictions, some particular vehicles cannot visit some particular customers.
- Some deliveries may be canceled (at some penalty cost).
- Customers may have preferences for particular drivers.
- Drivers may have assigned particular delivery areas.
- We can have some order restrictions on the order we visit some pairs of customers.
- There may be incompatibilities between the orders of some pairs of customers that impossibilities to carry them in the same vehicle.
- We can also have co-drivers not assigned to particular vehicles. Then, we can assume that they are picked up during (at the beginning of) the route.

From a theoretical perspective, VRP is related with the famous *Travelling Salesman Problem* (TSP) [34, 4]. This is the problem of finding the shortest path that visits a set of customers exactly once, and returns to the first point. Mathematically, this is the problem of finding the shortest Hamiltonian path in a weighted graph. This problem has received a lot of attention in Computer Science. Once we have assigned a vehicle to each customer, the problem of finding the route (the sequence of customers) of a single vehicle is a TSP. If we consider Time Windows, we get the TSP with Time Windows (TSPTW), i.e. the VRPTW with a single vehicle. Some heuristics for the VRP and the VRPTW consists of allocating customers to vehicles, and then solving the resulting TSP or TSPTW problems. In particular, this is the approach followed in an application for car-sharing developed at CSIC (described at the end of this document), and the approach that we will consider in this project.

Another related problem is *Pickup and Delivery* (PDP) [47]. In this case, rather than goods being delivered from the same depot, the goods are picked up during the route and delivered later. Then, additionally to time windows, we consider precedence constraints that ensure that goods are picked up before being delivered. We can consider several variants. For instance, we can assume that the vehicle, after deliveries are completed, picks up some items on the way back to the depot.

In the following we review the main approaches to solve CVRP and VRPTW [14, 15, 17, 29, 33, 46]. We concentrated on Integer Linear Programming (ILP) formulations (from the well-established field of Operations Research), and on the approach based on Constraint Programming (based on [8, 42]). We also consider the role of Local Search for this kind of problems [1, 32] and we finalize focusing on their dynamic versions [39].

3.1. Operations Research

We can formulate CVRP and VRPTW in terms of ILP. Let the set of customers be $C = \{1, \ldots, n\}$, and the set of vehicles be $M = \{1, \ldots, m\}$. Use 0 as the index of the depot, at route start, and $n+1$ to also index the depot at route end, and define $N = C \cup \{0, n+1\}$. The decision variables are x_{ij}^k for $i, j \in N$ and $k \in M$, where $x_{ij}^k = 1$, if vehicle k travels directly from customer i to customer j, or $x_{ij}^k = 0$ otherwise. The problem is characterized by the following set of constants:

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

- c_{ij}, for $i, j \in N$, is the cost of travelling from i to j,
- r_i, for $i \in N$, is the demand of customer i,
- Q_k, for $k \in M$, is the capacity of vehicle k,

For the depot we assume $r_0 = 0$, $r_{n+1} = 0$, $[a_0, b_0] = [a_{n+1}, b_{n+1}] = [0, K]$.

Then, we can encode the VRP as the ILP:

$$\text{minimize} \quad \sum_{k \in M} \sum_{i \in N} \sum_{j \in N} c_{ij}\, x_{ij}^k \tag{1}$$

$$\text{subject to} \quad \sum_{k \in M} \sum_{j \in N} x_{ij}^k = 1 \qquad \forall i \in C \tag{2}$$

$$\sum_{i \in C} r_i \sum_{j \in N} x_{ij}^k \leq Q_k \qquad \forall k \in M \tag{3}$$

$$\sum_{j \in N} x_{0j}^k = 1 \qquad \forall k \in M \tag{4}$$

$$\sum_{i \in N} x_{ih}^k - \sum_{j \in N} x_{hj}^k = 0 \qquad \forall h \in C\ \forall k \in M \tag{5}$$

$$\sum_{i \in N} x_{i(n+1)}^k = 1 \qquad \forall k \in M \tag{6}$$

$$\sum_{i,j \in S} x_{ij}^k \leq |S| - 1 \qquad \forall k \in M\ \forall S \subseteq C \tag{7}$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall i, j \in N\ \forall k \in M \tag{8}$$

The objective function (1) is to minimize the sum of the costs c_{ij} of arcs used in all routes. Constraints (2) ensures that each customer is visited exactly once. Constraints (3) ensure that the capacity Q_k of vehicle k is not exceeded. Constraints (4) to (6) together ensure that the routes are circular and start and end at depot. We also require constraints (7) to ensure that there are no sub-cycles that do not include the depot. Unfortunately, there are an exponential number of constraints (7), which makes this formulation unpractical for real applications.

If we want to extend this formulation to time windows, we can use a decision variable t_i^k, that represents the time vehicle $k \in M$ begins service at customer $i \in N$, and the constants:

- τ_{ij}, for $i, j \in N$, is the time to travel from i to j and service customer i,
- $[a_i, b_i]$, for $i \in N$, is the delivery window for customer i,
- K is the maximum route time.

Then, we can encode the VRPTW as the ILP with the same objective function, and the same con-

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

straints, where constraints (7) are replaced by the following ones:

$$t_i^k + \tau_{ij} - K(1-x_{ij}^k) \leq t_j^k \qquad \forall i,j \in N, \forall k \in M \tag{9}$$

$$a_i \leq t_i^k \leq b_i \qquad \forall i \in N, \forall k \in M \tag{10}$$

Constraints (9) impose a restriction on the time sequence of arrival times, and constraints (10) ensure that the time windows are satisfied. Constraints (7) are no longer necessary, since the time constraints ensure that no sub-routes exists.

3.2. Constraint Programming Approaches

In general, Constraint Programming (CP) methods try to find a solution incrementally, assigning values to variables and backtracking when an inconsistency is detected. The process terminates when a solution is found or the problem is proven to be unsolvable. Therefore, these methods are basically an exhaustive search. The same scheme could be applied to the VRP, but it is usually too inefficient to deal with real-world world situations. Real-world problems always have a solution, the problem is finding a good quality or near optimal solution.

3.2.1. Modelling VRP with Constraints

We will start by presenting a formulation of the VRP in terms of Constraint Programming [43]. This formulation considers time and quantity of goods constraints maintained along each route. This model can be easily extended to the capacitated vehicle routing problem, the vehicle routing problem with time windows, the pickup and delivery problem, and many other variants. One of the advantages of CP is expressiveness. We can add different extensions without changing the basic model.

As for the ILP formulation, we consider n customers orders and a set of m vehicles. The term *visit* denotes each stop of a vehicle to attend a customer. As in the TSP, we assume that there is only one visit per customer and two special visits of the vehicle to the depot. These two additional visits model the starting and finishing places for the vehicle. Let $C = \{1, \ldots, n\}$ be the set of customers, $M = \{1, \ldots, m\}$ be the set of vehicles and $V = \{1, \ldots, n+2m\}$ be the set of visits. (Notice that there are one visit per customer, plus 2 visits per vehicle to the depot). We define $f_K = n + k$ as the first visit of vehicle k to the depot, and $l_k = n + m + k$ as the last visit of vehicle k to the depot. The sets $F = \{n+1, \ldots, n+m\}$ and $L = \{n+m+1, \ldots, n+2m\}$ indicate the set of first and last visits to the depot, respectively.

For every $i \in V$, we use the integer variable p_i, with domain V, to represent the predecessor of each visit i, and s_i for the successor of i. By convention, each *first* visit of a vehicle has as predecessor the vehicle's *last* visit (formally $\forall k \in M.\ p_{f_k} = l_k$). Notice that the predecessor variables form a permutation of V and satisfy:

$$p_i \neq p_j \quad \forall i,j \in V \wedge i \neq j \tag{11}$$

In the previous ILP model, this is equivalent to require that the in-degree and out-degree of each node must be equal to one, i.e. that the path is Hamiltonian. Let's represent the current domain of variable

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

x by D_x. Constraints (11) are difference constraint and can propagates according to the following rule:

$$x \neq y : \quad D_x = \{a\} \rightarrow y \neq a \tag{12}$$

where x and y are constrained integer variables, and a is an integer. We will represent propagation rules as LHS $\rightarrow$ RHS, where LHS is a logical combination of conditions on the variable domains whose truth can be easily checked and RHS is a unary (or conjunction of) constraint(s) that can be enforced by domain filtering. The previous rule only applies when the domain of x is reduced to the singleton $\{a\}$. The RHS is enforced by removing the value a from the domain of y.

The value of the successor variables are kept consistent with the predecessor variables using the following constraints:

$$\begin{aligned} s_{p_i} &= i \quad \forall i \in V \setminus F \\ p_{s_i} &= i \quad \forall i \in V \setminus L \end{aligned} \tag{13}$$

The *element constraint* takes two integer variables y and z and a vector of integer variables x. The constraint specifies that z must be equivalent to the yth element of x, formally $z = x[y]$. This type of constraint is very common in CP models. It propagates as shown below.

$$\begin{aligned} & z = x[y] \\ & \exists a, b.\ D_y = \{a\} \wedge b \notin D_z \quad \rightarrow x[a] \neq b \\ & \exists a.\ D_{x[a]} \cap D_z = \emptyset \qquad\quad \rightarrow y \neq a \\ & \exists b.\ (\forall a \in D_y.\ b \notin D_{x[a]}) \quad \rightarrow z \neq b \end{aligned} \tag{14}$$

The coherence constraints (13) can then be implemented as follows (we show only one of the two forms):

$$s_{p_i} = i : \quad z = s[p_i] \wedge z = i \tag{15}$$

The use of predecessor variables p_i and successor variables s_i is redundant. They are symmetric, which allow us to express the problem and solutions using only one of them. However, their joint use allows additional inferences that prune the search space.

To represent multiple vehicles, a vehicle variable v_i of domain $\{1, \ldots, m\}$ is introduced, which represents the vehicle which performs visit i. We introduce the following constraint for the first and last visits:

$$v_{f_k} = v_{l_k} = k \quad \forall k \in M$$

The following constraints state that all visits along a route are attended by the same vehicle. These constraints along routes form what is called a *path constraints*. The above are the simplest form of path constraint. More complex ones will be used to maintain the time and quantity of goods along every route.

$$\begin{aligned} v_i &= v_{p_i} \quad \forall i \in V \setminus F \\ v_i &= v_{s_i} \quad \forall i \in V \setminus L \end{aligned} \tag{16}$$

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

Alternatively, we can ensure a more relaxed form of consistency, called *bounds consistency*, that requires to preserve some (lower and upper) bounds. They are much more easy to compute in terms of time and space. The capacity of the vehicle is one these bounds. It is modelled by the introduction of a variable q_i at each visit, representing the quantity of goods on the vehicle after performing the visit i. The following path constraints maintain the load on the vehicles at each point of its route.

$$\begin{aligned} q_i &= q_{p_i} + r_i \quad \forall i \in V \setminus F \\ q_i &= q_{s_i} - r_{s_i} \quad \forall i \in V \setminus L \end{aligned} \tag{17}$$

If the q_i variables have large domains, for efficiency, we can decide to only maintain consistency. The limit on the capacity Q_k of vehicle k is ensured then by:

$$q_i \leq Q_{v_i} \quad \forall i \in V \tag{18}$$

This constraints allow us to deal with heterogeneous fleets where each vehicle k may have different capacity Q_k. We could also consider constraints of the form $\forall i \in F.\ q_i = 0$ to require vehicles to begin empty. Alternatively, we can consider pickup and drop off models, where vehicles may start and finish partially occupied.

For the time, we will use constraints similar to load. However, in this case, we will allow vehicles to wait. Therefore, the constraints are inequalities rather than qualities. We introduce an integer variable $t_i \geq 0$ representing the time at which service for visit i begins.

The following constraints maintain these time variables along vehicle routes:

$$\begin{aligned} t_i &\geq t_{p_i} + \tau_{p_i,i} \quad \forall i \in V \setminus F \\ t_i &\leq t_{s_i} - \tau_{i,s_i} \quad \forall i \in V \setminus L \end{aligned} \tag{19}$$

For propagation methods, notice that this expression is a cumulative inequality, instead of a cumulative equality, in order to allow waiting time. Time windows on customers are specified by adding constraints on the t_i variables. For example, the constraint

$$a_i \leq t_i \leq b_i \tag{20}$$

ensures that customer i will be visited between times a_i and b_i. Multiple time windows can also be expressed. For instance, the constraints $a_i \leq t_i \leq d_i$ and $t_i \leq b_i \vee t_i \geq c_i$ express that customer i must be visited in the window $[a_i, b_i]$, or the window $[c_i, d_i]$.

Every vehicle k can have a different availability window $[O_k, H_k]$. To ensure that every vehicle is available when it is required, we can use the constraints:

$$O_{v_i} \leq t_i \leq H_{v_i} \quad \forall i \in V \tag{21}$$

There are several ways to avoid cycles of visits which do not involve a first and last visit to the depot. For instance, we can require that the time for each visit is positive and finite.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

If we want to compute the total distance d travelled for all vehicle (the cost function), we can define it as:

$$\begin{aligned} d &= \sum_{i \in V \setminus F} \delta_{p_i,i} \\ d &= \sum_{i \in V \setminus L} \delta_{i,s_i} \end{aligned} \tag{22}$$

where $\delta_{i,j}$ is the travel distance from visit i to j. Note again the use of both the predecessor and successor variables. This results in a more efficient propagation during the search. Each term in the sums is maintained by an element constraint, and the total sum by a summation constraint. We avoid here the description of the propagation details.

We can also assume a distinct cost-per-kilometer C_k for every vehicle k. The cost function is then given by:

$$\begin{aligned} d &= \sum_{i \in V \setminus F} C_{v_i}\, \delta_{p_i,i} \\ d &= \sum_{i \in V \setminus L} C_{v_i}\, \delta_{i,s_i} \end{aligned} \tag{23}$$

These costs are maintained as the cost in terms of total distance, except that now there is an additional multiplication constraint per term.

Considering additional constraints the model can also be used on pickup-and-delivery problems (PDP) and problems with back-hauls. We use negative values of r_i for a delivery, and positive values of r_i for a pickup.

Suppose that each pickup and deliver order has a pickup visit o_p and a delivery visit o_d. Then, the following constraints state that the the pickup and delivery visits must be attended by the same vehicle and that the pickup must be performed before the delivery.

$$\begin{aligned} v_{o_p} &= v_{o_d} \\ t_{o_p} &< t_{o_d} \end{aligned} \tag{24}$$

In the case of back-haul, interleaving of pickup and delivery visits is excluded via the following unary constraints on successor and predecessor variables:

$$s_i \neq j \quad \forall i \in P\ \forall j \in D \tag{25}$$

where $P = \{i | r_i > 0\}$ and $D = \{i | r_i < 0\}$. It express that no delivery is allowed just after pickup.

3.2.2. An Alternative Formulation

In the formulation presented here, the solution requires a single value for successor and predecessor of a visit to be identified. In the context of the TSP, an alternative formulation requires the set of all successors and predecessors of the visit to be identified.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

Two set variables, B_i and A_i, are maintained for each visit that determine all visits which must come before (respectively after) visit i. In a TSP, any visit j other than i must either come before or after i. In a VRP, this is not the case, and we identify three possibilities (a) i before j on the same vehicle route ($i \in B_j \wedge j \in A_i$) (b) j before i on the same vehicle route ($j \in B_i \wedge i \in A_j$) (c) i and j served by different vehicles ($i \notin B_j \cup A_j \wedge j \notin B_i \cup A_i$). The A and B sets are maintained by examination of the time constraints between pairs of visits. This leads to the following constraints for all visits $i \in V$:

$$\begin{array}{llll} (a) & B_i \cap A_i = \emptyset & (b) & j \in A_i \iff i \in B_j \\ (c) & s_i = j \iff A_i = A_j \cup j & (d) & j \in A_i \wedge l \in A_j \Rightarrow l \in A_i \\ (e) & v_i \neq v_j \vee j \in B_i \vee j \in A_i & (f) & A_i \cap B_j \neq \emptyset \Rightarrow s_i \neq j \\ (g) & j \in A_i \Rightarrow t_j \geq t_i + \tau^*_{i,j} & (h) & j \in B_i \Rightarrow t_j \leq t_i - \tau^*_{j,i} \end{array} \tag{26}$$

Here $\tau^*_{i,j}$ is the shortest path (in terms of time) between the start of service of i and the start of service of j. This can be found using a shortest path algorithm, but any lower bound is legal to use, including $\tau_{i,j}$ (making the reasonable assumption that the triangle inequality holds on travel times).

In the above expressions, constraint (a) says that no visit can be both before and after another visit; constraint (b) states that if i is before j, then j is after i; constraint (c) links the direct successors with the complete successors; constraint (d) enforces the transitive closure which states that if i is before j and j is before l, then i is before l; constraint (e) says that if i and j are served by the same vehicle, they must be ordered; constraint (f) says that when at least one visit occurs between two other visits, those two visits cannot be directly linked; finally constraint (h) requires that when visits i and j are ordered in a certain sense, there must be a minimal time gap between them.

These constraints result in more powerful propagations eliminating certain arcs from consideration, or enforcing that two visits must be performed by different vehicles, for example. The before and after sets can also be used to tighten time bounds on visits to strengthen the method proposed. The following constraints are valid for any customer visit $i \in C$.

$$\begin{aligned} d_B &= \sum_{j \in B_i} t_{j,s_j} \\ d_A &= \sum_{j \in A_i} t_{p_j,j} \\ t_i &\geq t_f v_i + d_B \\ t_i &\leq t_l v_i - d_A \end{aligned} \tag{27}$$

That is, the earliest time that visit i can be started is the start time of the vehicle on which i will be serviced, plus the travel for all visits up to i. A symmetrical constraint limits the latest time that visit i can be completed.

3.2.3. Lower Bounds and Cost-Based Propagation

The cost can be used to limit search through a global constraint. The basic CP model will perform propagation through constraints. Consider in particular, the first of the constraints $d = \sum_{i \in V \setminus F} \delta_{p_i,i}$.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

Suppose further that a goal cost G has been identified, so that we require $d \leq G$. (Such a cost is usually provided by finding a solution to the VRP with some cost $G + \epsilon$, resulting in the new tighter cost bound G.) Constraint CST will maintain lower and upper bounds on d computed from a sum of terms, each term of which is the distance from each node's predecessor to the node. During search, not all p_i variables will be bound to a single value, and bounds on the contribution of each term will be computed by propagation of the element constraints which constitute each term. Assume that T_i is the term in CST corresponding to visit i. Then:

$$\begin{aligned} \underline{T_i} &:= min_{j \in p_i} \delta_{j,i} \\ \overline{T_i} &:= max_{j \in p_i} \delta_{j,i} \end{aligned} \tag{28}$$

The summation will then maintain bounds on d, the total distance travelled, notably the lower bound:

$$d_{LB} = \sum_{i \in V \setminus F} T_i \tag{29}$$

The propagation rule $d := LB$, ensures that this lower bound is enforced. Of course, whenever the domain of d will become empty and force the search to backtrack. However, what happens more often is that propagation will occur, removing arcs which would, if they appeared in the final solution, exceed the cost bound G. Consider the lower bounds on the cost when one visit i has as predecessor a visit h other than its closest predecessor. The new lower bound d_i, h_{LB} would then be:

$$d_{LB}^{i,h} = \sum_{j \in V \setminus F \setminus i} \underline{T_i} + \delta_{h,i} \tag{30}$$

The following propagation rule then applies:

$$\exists h \in p_i d_{LB}^{i,h} > G \rightarrow p_i \neq h \qquad \forall i \in V \setminus F \tag{31}$$

This reasoning can also be symmetrically applied to the successor variables.

Although the above method performs cost-based propagation, the lower bound d_{LB} is poor as it does not consider any routing aspects, such as the fact that all predecessor (or successor) variables must take different values. This has the effect of significantly weakening the propagation. Accordingly, better bounds have been proposed in the context of CP, as well as a cost-based propagation methods based on reduced costs.

3.2.4. Constraint Programming in Search

CP can offer many advantages when solving routing problems, due to the increased pruning achieved through propagation. We have also seen that local search methods have been effective in solving the problems. However, a difficulty arises when one attempts to put these two methods together in a naive way. In local search, we may move a customer from one route to another, assigning a new successor $s_i = j$. Later, we may decide to move node j, so that s_i receives a new value. However, this contradicts a basic operating principle in classic CP: so called *chronological backtracking*. Under

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

chronological backtracking, decisions must be undone in the reverse of the order they were made. So in order to undo the decision $s_i = j$, we would have to undo all operations performed since that time. This would undo all the progress made by local search, and hence is unacceptable.

Several ways around this problem seem to have been identified so far. The first is to allow a heuristic or meta-heuristic to control search. In this case, the constraint system is used simply as a rule checker. The second way is to insulate the CP system from the changes being made at the lower level, by wrapping up local search changes within an operator, which is then used within the constraint system. The use of these sorts of operators also allows the user to identify parts of the search that can be handled using traditional backtrack search. These methods are discussed in more detail below. A third way of using CP is as a subproblem solver. Here, an independent search process generates a sequence of subproblems (usually closely related to the original routing problem) which are solved by CP. This sort of use has more of the flavour of the second class identified above, as the CP system is being used as more than just a rule checker.

Constraint Programming as a Rule-Checker. The easiest way to use a CP framework in solving the VRP is to simply use CP-based methods as a rule checker to be applied to individual routes, or collections of routes. The main advantage over other possible methods is that CP is very expressive, and a wide variety of side constraints can be specified and checked efficiently. In addition, a CP-based solver that can handle the core VRP constraints can often be used without algorithmic modification to handle a problem with additional side constraints. Using CP as a rule-checker means that the search procedure is handled outside the constraint system. The constraint system is simply called each time a new route (or partial route) is to be tested. However, the CP system is still able to perform the usual propagations, limiting the scope of decision variable and potentially identifying unfeasible partial solutions.

Regarding the use of CP within a non-chronological framework, two representations of the solution are considered: an active representation within a CP system that is only instantiated when a constraint check is required, and an passive representation used by the local search routines. Within this framework, a full test of all constraints through the whole (partial) solution can be very expensive. They therefore describe a number of methods for improving the efficiency of the CP system.

First, many local search operators operate using a particular criterion, for example, identifying the node that can be moved at least cost or maximum regret. A large gain in efficiency can be made by simply testing that criterion first, before performing any constraint checks. So rather than seeing whether it is legal to perform a particular move, the system should first see whether the cost of the move is less that the best found so far. If not, then the constraint checks can be skipped altogether. Another possibility for improving efficiency is to have specialized propagators for the core constraints. For example, each *move operator* within a local search method may have its own propagator which examines the values produced during the path variable propagation. Illegal moves can then be identified very quickly. Even though they concentrate on the core constraints, since any side constraints will also be potentially affecting the path constraints, these propagators make good use of information from all constraints. Finally it is important to observe that many move operators affect only a subset of the routes in a solution. Most constraint checking can therefore be limited to just those routes that have changed.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

Local Search within a Constraint Framework. As noted above, operators can be defined within a CP framework to insulate the Constraint System from non-chronological backtrack. This allows more of the search to take place within a Constraint framework, and hence allows more scope for propagations and other techniques to prune the search space.

Many of the operators used within this type of framework are based on serial insertion and block deletion. These types of modification are well suited to use within a Constraint Programming framework, as they allow the propagations previously described, to be used in full –both narrowing the potential sites for insertion, and quickly identifying partial solutions that cannot lead to a feasible solution. The formulation is also able to cope with arbitrary side constraints easily. Within-route constraints (like capacity) –which usually form the bulk of constraints– can be checked during insertion.

One such insertion-based technique, developed specifically for use in constraint programming environment, is Large Neighbourhood Search [5]. In this method, a group of *related* customers is removed, and then re-inserted into the existing runs with optimal cost. The customers are related geographically, temporally, or by use of a particular resource. For example, a simple relatedness function is $R(i,j) = 1/(c_{ij} + V_{ij})$, where c_{ij} is the cost of travel from i to j, and V_{ij} is $K > 0$ if i and j are on the same vehicle, 0 otherwise. K is chosen to be comparable to the c_{ij}s.

The number of visits removed starts at one, and increases if no improvements have been found, up to some maximum. Re-insertion uses exact branch-and-bound procedure with constraint propagation, which can find the minimum insertion cost rapidly.

The method has proved particularly successful on VRP, and has since been used in CP methods for other problems, including crew scheduling and maximal satisfaction problems.

The strength of propagations from insertion-style local search is further exploited as follows. Three meta-heuristics that involve repeated insertion —Large Neighbourhood Search, Limited Discrepancy Search, and Ejection Chains— are combined. These three methods are the *building-blocks* in a system designed to discover new heuristics based on automated learning. Depending on the data presented, the methods produces heuristics, made up of calls to these building blocks, that are able to produce good solutions. The CP system is acting as a rule checker here, but in addition, traditional CP techniques are used to solve small TSPs with side constraints that appear as subproblems.

Another general-purpose search procedure that has been developed within the constraint literature is Limited Discrepancy Search (LDS). Many problem-solving methods involve a sequence of steps, a set of actions that can be taken at each step, and a heuristic for ordering the preference for those actions. Following the first preference at each step gives a solution to the problem. LDS systematically looks at the solutions that differ from the heuristic solution by making a different choice at a number of points. So for instance a 1-discrepancy would follow the heuristic at all but one step. At that step it would take the second-best choice according to the heuristic. Going against the heuristic at stage one (then following it for the rest of the procedure) gives a 1-discrepancy solution. Going against the heuristic at stage two gives another such solution, etc. So if there were n steps in the solution, n different 1-discrepancy solutions can be generated. A 2-discrepancy solution either twice uses the heuristic's second-best choice during construction, or once uses the third-best choice. In the context of the VRP, LDS can be used during construction. For example, in choosing the next customer to insert, or the insert position.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

The Constraint framework is exploited further, to implicitly search large neighbourhoods. As mentioned previously, large neighbourhoods are more likely to contain good solutions, or allow good solutions to be discovered in fewer steps. However, such neighbourhoods can be expensive to search. A method, based on branch-and-bound, searches efficiently these larger neighbourhoods.

They characterize a neighbourhood N of a particular solution by a set of finite-domain variables $V = \{v_1, \ldots, v_n\}$, so that each feasible combination of values $\bar{v} = v_1 \times \cdots \times v_n$ maps to exactly one neighbour of the current solution, and vice-verse. A systematic, branch and bound exploration of feasible values of $\bar{v}$ then implicitly explores the neighbourhood efficiently. Using lower-bounding functions allows non-productive areas of the neighbourhood to be identified and eliminated.

For example, in the TSP with time windows, an effective neighbourhood is the orientation- preserving 3-opt neighbourhood described before. This can be characterized using 3 indices –I, J and K– defining the break points, with the constraint $I \prec J \prec K$ (where $I \prec J$ means "I precedes J in the current tour"). Each feasible I, J, K combination represents one possible 3-opt exchange (and all 3-opt exchanges are represented by an I, J, K combination). This I, J, K space can be explored using a branch and bound tree of depth three, using two bounding procedures to prune the search.

Bounding, propagation and pruning are being used to implicitly eliminate subsets of neighbours. This allows the more complex and larger neighbourhood structures to be explored effectively.

In a hybrid CP/OR approach, methods such as (Large Neighbourhood Search, GENI exchange, Ejection Chains) are embedded as operators within a CP framework. Again, the building blocks chosen are based on serial insertion and deletion –precisely the operators able to benefit most strongly from propagation.

Basic operators are defined which remove and insert customers in a route. New operators can then be defined –for instance the following code implements a simple ejection chain:

$$NEC(c) : -Insert(c, R) \vee (Remove(C, R) \wedge Insert(c, R) \vee NEC(C))$$

where R and C are variables representing any route or customer. The code attempts to insert c in a feasible route or, if no such route can be found, removes another customer from a route, inserts c in that route, and resources to find a new home for C. Obviously this method must be modified to prevent cycling etc, but the flavour of CP-based programming is evident.

In the method described, the construction phase, a local search phase and a post-optimization improvement phase are all defined in terms of the operators. They explore the neighbourhoods defined by each operator using branch-and-bound search within a CP framework. As with the previous method, propagation and pruning are working to reduce the number of solutions actually visited.

3.2.5. Using Constraint Programming as a Subproblem Solver

We focus here on methods that involve solving a sequence of constrained subproblems. These subproblems often share many of the constraints of the original problem, and hence CP may be a useful technique to employ.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

An example an insertion-based technique that interleaves insertion with local search. It uses CP to check the feasibility of insertions and of the neighbourhood moves used in local search. However, it also uses a constraint-based framework to solve the TSPTW sub-problems exactly. Because the whole system is set up within a constraint-based framework, it is easy to implement Limited Discrepancy Search as a solution improvement technique.

Another strategy uses a CP-based solver to find exact solutions to the TSPTW subproblems arising in the context of the Large Neighbourhood Search procedure. A branch-and-price approach has been used to solve the Travelling Tournament Problem —which is related to the routing problems we examine here. In this approach, integer programming is used to solve the master problem, while CP is used to solve the pricing problem. The branch-and-price approach is also used to solve the Vehicle Routing Problem with Time Windows.

3.2.6. Real-World Constraints

We have studied the theoretical version of VRP, as it has been established in the specialized literature. However, there are significant differences between VRP in academia and routing practice in the real world, due to the great variety of constraints and objective functions that are seen in day-to-day use, many of which have never been examined by academic research.

In the following, we describe a number of examples of operating practice that diverge in some way from the standard definition (from [42]):

- Minimizing vehicles is seldom an optimization criteria in day-to-day scheduling. The fleet size is determined periodically, but in between times, drivers are typically on contract, and so are paid for some minimum time whether they drive or not. As a result of this, there is often a constraint that route time exceeds the paid minimum, while being less than the contracted maximum time.
- Meal and rest breaks, within a certain time of starting, and importantly, within a certain time relative to one another, must be inserted into the route automatically
- Subcontracting is common, whereby a shipment is sent using another carrier. The constraint that all visits are completed is therefore often dropped in favour of a cost term in the objective that will automatically drop uneconomic visits.
- A situation has been seen whereby the company only pays for travel to and from the first and last visits of the route, not for the distance on the route itself. There is a constraint which forces the route to be linear, rather than petal-shaped, which in effect forces the last visit to be one of the most distant from the first. In fact the constraint is that the total length of the route is no more than 30% greater than the distance from the first to the last stop.
- In some workplaces, some drivers have secured a strong negotiation position which allows them to choose which stops they wish to perform. Others are able to choose some of their route.
- Cross-docking is another very common practice. For example, a major supermarket has a regional distribution centre. This centre receives goods from suppliers, and distributes goods

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

to stores. To avoid double handling and storage costs, where-ever possible the deliveries are cross-docked: the goods are taken straight across the dock from a suppliers vehicle to the distribution vehicle. If the supermarket is using its own vehicles to pick up from the supplier, then this constraint is a movable time window linking two or more routes.

- A similar inter-tour constraint occurs when one vehicle delivers to a scheduled service, such as a ferry or train, and another picks up from the other end. There may be a choice of many scheduled services to use, but two routes must coordinate on the same service. Note that the receiving vehicle is not necessarily identified, but one of a number of vehicles must be tasked to meet the scheduled service.
- In (telecommunications) technician dispatching there can be a requirement to have two technicians at different locations at the same time, so that they can perform end-to-end tests of communications hardware. A constraint is required to force these two visits into different vehicles and for the two visits to occur in the same time window.
- Some long-haul companies use trailer change-over. Here, one vehicle (perhaps while making other deliveries) carries a trailer. At some location and time (decision variables) it meets another vehicle and the trailer is moved to the second vehicle for onward travel. This may be repeated a number of times. Each such change-over represents an inter-tour constraint. Again the vehicles involved are variable.
- The resources at a site may be limited. For instance there may be a limited number of docks or forklifts. The number of vehicles visiting at any one time must therefore be limited. This will be a consideration for example in a supermarket where multiple vehicles are sent to replenish stocks.
- Overtime rates must be taken into account in the objective, but are often expressed in a complicated fashion, including discontinuities, and penalty payments. E.g. $x up to the first half hour, $y for each subsequent half hour, plus $z meal allowance (plus 30 minutes break) if working more than 2 hours.
- In technician dispatching, the truck inventory must be maintained. The inventory of up to 300 types of parts may have to be tracked, and only technicians with sufficient supplies assigned. The technician must return to base to restock when appropriate, or may be able to be resupplied en-route by either meeting another technician and swapping parts, or by a special delivery from base.
- Lots of little, but still important, preferences have been seen, for example shipments to a particular customer in a week must be done by the same driver; or a husband and wife must/may not work together as a team.

Note that while many of the situations listed can be addressed using the models discussed here, some require more elaborate models, and some can only be approximately modelled. In addition, some would require bespoke techniques to find good solutions.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

3.2.7. Dynamic Operation

Vehicle routing problems in the real world are really dynamic: the problem may change dynamically and the last computed solution may not be fully applicable because the problem has evolved into a new one.

Changes may be small –where some visits may be added or deleted without changing the general structure of the route–, or the entire routing process may be driven by the stops being added, such as in taxi or parcel delivery operations. Other dynamic aspects of operations include:

- A company may have a list of clients to visit, but not all clients will require visiting each day. The stops to be performed may only be finalized as close to dispatch time (or even after).
- Traffic incidents and road works can alter the time taken to drive between two stops.
- It may be difficult to calculate a-priory how long a visit will require. If visits take longer than expected, parts of a planned route may have to be re-scheduled.
- Vehicles may break down, requiring visits to be re-scheduled.

Approaches that consider these dynamic elements in the VRP context are contained in Section 3.4..

3.3. Local Search

Classically, solving approaches based on ILP or CP follow an exhaustive search: they explore every node of the search tree which could potentially be an ancestor of a solution or a solution itself. As mentioned before, this exact strategy is only feasible on instances of small size; it becomes impractical for problems of realistic size because it could require a too-long computing time. Therefore, it has been replaced with local search; this is an umbrella term for a class of incomplete methods, with two main theoretical limitations: (1) they may fail to find a solution even if it exists, and (2) they may report a solution which is not the optimum. In practice, however, these limitations are not so important because these methods usually find good-quality solutions (circumventing 2), on instances which are known to be solvable (or its solvability is treated by other means, circumventing 1). Their main advantages is that they require limited computing time, usually offering better quality-time ratio than exhaustive approaches.

3.3.1. Hill Climbing

Regarding local search [1, 32], its basic operation is denominated *hill climbing* and it is very simple. It works with complete solution candidates; at any time the procedure keeps the current solution. Hill climbing searches in a small space around that solution (its neighbourhood) and if it finds a better candidate, that becomes the new current solution. The process iterates until exhausting resources (typically time or a maximum number of iterations).

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

Solving VRP by hill climbing [22, 24, 45] requires to define precisely the used neighbourhood in terms of the move operator: the space that can be generated by applying the move operator around the current solution. The size of the neighbourhood is a key factor: the larger the neighbourhood, the more likely it is to contain new good solutions, but larger neighbourhoods also require more search effort (that is, computing time). Several of the move operators used in solving routing problems are described below.

- 2-opt. Choose nodes i and j from the same tour, with $i+1 < j$. Delete arcs $i \to i+1$ and $j \to j+1$. Replace with $i \to j$ and $i+1 \to j+1$. This reverses the order of nodes from$i+1$ to j. Neighbourhood size is $\mathcal{O}(n^2)$.
- 3-opt. Analogous to 2-opt, 3 arcs are deleted and re-attached in such as way that intermediate sections are not re-ordered. Neighbourhood size is $O(n^3)$. k-opts for $k > 3$ are also possible, but the cost is prohibitive.
- Or-opt. Choose a parameter K, the maximum length considered. Remove each sequence of customers of length $k = K, \ldots$ down to 1 customers. Test re-inserting the min forwards and backwards orientation between each remaining pair of customers. Re-insert in the cheapest position. Neighbourhood size is $\mathcal{O}(Kn^2)$.
- Relocate. Subset of Or-opt moves (move a single customer from one route to the best position in another). Neighbourhood size is $O(n^2)$.
- Exchange. Choose two customers in different routes, and swap their positions.
- Cross. Extension of Exchange: Choose a sequence of customers $i_1 \ldots j_1$ of length at most K in one route, and $i_2 \ldots j_2$ of length at most K in another, and exchange the two sections. Neighbourhood size is $O(K^2n^2)$.
- λ-exchange. As for cross except that the routes are re-optimized after the customers are exchanged. Neighbourhood size depends on re-optimization method, but is at least $O(K^3n^3)$.
- GENI Exchange. An extension of the relocate that allows the receiving route to be minimally reordered to accommodate the new customer. It allows i to be inserted into the new route between non-consecutive nodes j and k by reinserting the segment $j+1 \ldots k-1$ to a different position.
- Ejection chains. A sequence of customers is selected and inserted into another route. This may cause some sequence of customers to be ejected in order to accommodate the new ones. The procedure forms a chain of such ejections until no customers are left unassigned. This has proved to be a very powerful operator. Neighbourhood size depends on the restrictions imposed on the chain length, but is potentially very large.

Hill climbing typically looks at a subset of these operators. The neighbourhood of the current solution will be examined to find cost-reducing moves. Either the first found, or the best found will then be executed. This leads the procedure to a local minimum within the specified neighbourhood.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

3.3.2. Meta-Heuristics

Hill climbing has the issue of local minima, which appears when every state in the neighbourhood is not better that the current solution. What to do in that case? A number of strategies have been proposed to escape from local minima. They are called with the generic name of meta-heuristics [1, 32]. They escape from local minima in one of two ways: (1) the controlled acceptance of cost-increasing moves, or (2) by neighbourhood expansion. Many meta-heuristics have been applied to the VRP and related problems [17, 13, 20, 5]. Next, we give a telegraphic description of the most popular ones.

- Simulated Annealing. Acceptance of cost-increasing moves is controlled by a system parameter called temperature by analogy with a crystal annealing process. An increase in cost δ is accepted with probability proportional to $e^{-\delta/T}$. An updating procedure decreases T as the process continues so that solutions converges to a (hopefully better) local minimum. Any combination of the local search operators just described can be used to form a search neighbourhood.
- Tabu Search. Tabu search allows a cost-increasing move, and then places the reversal on a *tabu list* so that the move cannot be *undone*. This allows the neighbourhood of the new solution can be properly explored. Inspired by this method, a large number of specific techniques have been proposed for treating VRP versions.
- Genetics. Genetic (also Memetic and Evolutionary) algorithms follow a Darwinian evolution metaphor. In the basic form, a population of solutions is created. Individuals are crossed to form a new solution with characteristics from both parents. Mutation operators (similar to the local search operators above) can also be applied. Implementations differ in initialization, mutation and crossing methods. Again, many genetic methods have been suggested.
- Variable Neighbourhood Search. This method considers a sequence of neighbourhoods of increasing size around the current solution. It exhausts one neighbourhood before moving on to the next largest, and will return to the smallest if a significant change has been made. This allows the solution to be improved as much as possible from low-cost, small neighbourhoods before investing the time to examine the larger ones.

3.3.3. Initial Solution Construction Methods

They are used to create an initial solution. A popular strategy is the insertion procedure, where a customer is selected and inserted in a route in such a way as to minimize an incremental cost function. A number of constraints can be enforced during construction. It is particularly easy to check for constraints that only deal with a single route. There are a number of variants, indicated next.

- Parallel versus sequential. Sequential procedures focus on a single route, adding customers until no more will fit. Parallel methods build a set of routes simultaneously.
- Seed customers. The most successful parallel method uses seed customers as the sole customer on a chosen number of routes. These customers act to guide the emerging routes.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

- Insertion order. It focuses on the order in which customers are inserted. Some methods use a scoring system based proximity and other measures. Some use regret (difference in cost between best and next-best insertion points).
- Insert position. The objective function used to determine insert position may simply try to minimize the additional travel required to visit a customer. Others attempt to take time and other constraints into account, guiding the selection process to maximize the spare time or resources in resulting routes. (The same function may be used for both choosing the customer to insert and the insert position.)

Perhaps the most famous VRP construction method is the *Savings* method of Clarke and Wright [19]. It starts with each customer on their own route. The savings obtained by combining two routes is $S_{ij} = c_{i0} + c_{0j} - c_{ij}$. The method selects the maximum S_{ij} for which the combined route is feasible, and iterates. Other approaches to route construction include the *sweep heuristic* of Gillet and Miller [26]. Here, a ray centred at the depot is swept in a clockwise direction. Each customer it touches is added to the route until no more will fit. A new route is then begun.

3.4. Dynamic VRP

The basic idea behind the addition of the word *dynamic* to the acronym VRP is that a particular instance may not be completely known at solving time, and new information should be integrated in the solution as it arrives. Typically, some request are missing and they are known when the initial solution (obtained with the initial data) is being executed. These requests are named *inmediate requests*.

The study of dynamic VRP has been done mostly in the last 25 years. In the following, we summarize the main contributions on this topic, focusing on solving aspects [27, 4]. In the academic literature, the dynamic aspects that have received most attention are those *inmediate requests* that appear unexpectedly. In the project, we are also interested in other forms of dynamism, such as blocking roads, breakdown of vehicles or other contingencies. We believe that the proposed solving methods can also cope with these events.

Before entering in solution strategies, it is worth noting that the practical feasibility of dynamic VRP depends on a number of technological advances available today, but unknown in the past. The introduction of the Global Positioning System (GPS) the widespread use of smart-phones and the existence of Geographical Information Systems (GIS), have been essential to allow companies for an effective management of their fleet in real-time.

Next, we present approaches successfully applied to dynamic VRP, in absence of stochastic information. As new data are revealed over time, the complete instance is only known at the end of the planning horizon. When applied on the current state, exact methods can provide the optimal solution for what is known at that time, but not for the final problem configuration (who will be revealed in the future). Due to this fact, most dynamic solving approaches rely on heuristics that quickly compute a solution from the available information, under the convention that the final instance would differ from it.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

Solving approaches to include new information in VRP instances are divided in two main groups: periodic re-optimization and continuous re-optimization. Both are discussed below.

3.4.1. Solving by Periodic Re-optimization

A general strategy for periodic re-optimization is as follows. An initial optimization produces the first solution, that starts to execute. Then, an optimization procedure periodically solves the instance corresponding to the current state, either when new data are available or at fixed time intervals (called *decision epochs* or *time slices*), producing new solutions that replace previous ones. This strategy reuses algorithms developed for static routing, although the whole optimization process has to be carried out before updating the routing plan, which may cause delays prior execution of the new solution.

Under this generic approach, three real cases deserve some attention. They are:

- MYOPT [50]: in a PDP context, a fleet of trucks has to service point-to-point requests arriving dynamically. The proposed solution is based on a linear program that is solved whenever a new request arrives.
- DYCOL [16]: in a VRPTW context, it is based on column generation on the set partitioning model, using columns from the previous solving (decision epoch). Computational results show good performance in terms of the quality of the objective function, with limited CPU time.
- ACS [38]: in a VRP context, it is based on ants algorithms. It repeats the static optimization process at fixed intervals (time slices). This strategy is supported by the nature of requests, which never demand an urgent treatment.

3.4.2. Solving by Continuous Re-optimization

In contrast with periodic approaches, continuous re-optimization repeats the optimization process as soon as new information is available. In order to integrate quickly there new data, it maintains information of good solutions in an adaptive memory (in a context of meta heuristic solving approach). Whenever new data are available, a decision procedure updates the current routing. It is worth noting that because the current routing is subject to change at any time, vehicles do not know their next destination until they finish their current request.

Three interesting cases are described below:

- Parallel Tabu Search [25]: in the context of VRPTW, this approach maintains a pool of good routes in the adaptive memory. Parallelization is done by partitioning the routes of the current solution and optimizing them in independent threads.
- Multiple Plan Approach [9]: also working with Tabu search, Multiple Plan Approach maintains a solution pool (the routing plans), used to compute a distinguished solution. When a new requiest

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

arrives, if it is accepted then it is inserted in the pool and incompatible solutions are discarded. The pool is always maintained coherent with the current state of vehicles and customers.

- Genetic algorithms [28, 30]: they have been used to treat DPD and VRP. Genetics in the dynamic context are very similar to those designed for static problems, although they run until the planning horizon and solutions are constantly adapting the changes made to the input.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

4. Transhipment and Intermodal Freight Transport

Transshipment is the act of shipping goods to an intermediate destination prior to reaching their ultimate location. It was introduced by [36] as an extensión to the “transportation problem”, where locations were not only origins or final destinations but also points acting as intermediate destinations. The initial formulation of the solving model for this problem was in terms of Linear Programming, an approach that remains clearly valid (but with limitations regarding the size of problems that can be solved in a reasonable time). Often, transshipment operations depend on the type of the modality of the used transport. Thus, it is frequent to find papers that qualify the type of transshipment (“sea-port containers”, or “rail-to-rail”). This is discussed in detail in the section devoted to Transshipment Operations. Also, intermodality is a true and very used option (for instance, sending goods by ships to cover long distances to a port acting as a hub, then switching to trains or trucks for a closer distribution). Because of that, we discuss these specific issues in the section of Transshipment and Intermodality. The amount of papers devoted to this topic is very high, encompasing both algorithmic approaches and solving proposals with user views and particular cases. In the following, we have tried to summarize the papers that look more fruitful for the project purposes.

4.1. Transshipment Operations

In this Section we concentrate on activities developed at seaports (that is, ship A carries a number of goods, which are unloaded in port X, and after some days, they are transferred to another transport medium —ship, rail or truck—, that delivers them into their final destination; typically this is done using containers). In [31] you can find a description of these operations in maritime transport, in the following we provide a summary. Aside, we have to mention the “hub-and-spoke” topology for a communication network (airline route-structure, or vessel route). There are one (or a few) “hubs”, and a series of points connected to these hubs by “spokes”. The classical point-to-point transportation model is replaced by transit through spokes, so every transit starts or ends in some hub. This has beneficial effects regarding the number of routes ($O(n)$ versus $O(n^2)$ in the point-to-point model), but adversarial effects regarding unexpected changes and the total distance travelled. Assuming a “hub-and-spoke” network of ports, the current trend is to use larger ships to connect hubs, and the distribution from hubs to other ports is done by smaller ships. In this way, larger ships —which usually are more expensive— can spend more time at sea. There is a growing number of container terminals around the world in which transhipping containers is the dominant activity [6]. Regarding the transshipment operations in seaports, [49] [31] group them in the following classes:

1. Ship Arrival. When a ship arrives to port, it has to moor at the quay using some berth points. The decision about the number of berths in quays should be done at strategical level. The disposition of ships regarding berths generates interesting optimization problems, in purity multi-objective (typically interested in minimizing overall staying time and dissatisfaction on order of berthing), but often treated as single objective by adding the two goals previously mentioned.

2. Loading and Unloading. The unloading plan indicates which containers should be unloaded and where they are located in the ship. A crane is used for the unloading process. The crane driver

LOGISTAR project has received funding from European Union’s Horizon 2020 research and innovation programme under grant agreement no. 769142.

has a lot of freedom to determine the order in which containers are unloaded. On the contrary, the loading process offers little flexibility: a good distribution of containers over the ship is really needed. Because of that, a stowage plan is made (see [Shields 1984] and [Wilson and Roach 2000] for different approaches of container stowage). Crane utilization generates new issues to optimize the global usage of avaliable resources at sea ports. Specifically, in this context is defined the crane scheduling problem.

3. Horizontal Transport (transport of containers from ship to stack and vice versa). Often, terminals use trucks, reach stackers, straddle carriers and AGVs (robotic vehicles that travel along predefined paths) to move containers from the maritime side to the terminal yard. In this task, the load sequence, the route selection and the sequence of pickup are some of the problems that have to be determined prior an effective operation. Coordination with cranes is an essential part of the process, since both are actors in the same job.

4. Yard Stack (stacking of containers). Policies regarding where to position and how to stack are needed. The type (export or import), size (20 foot or 40 foot), destination and ship line that owns the container are common ingredients of these policies (as well as if containers are transhipped into another ship or the expected waiting time).

5. Inter-terminal transport. Once containers are unloaded in the stack yard, they may be transported into other terminal or to other destinations. Between terminals, the vehicles mentioned in the horizontal transport entry can carry out the movement. Regarding other destinations, rail and trucks are the most popular modes to distrubute freight in an internal transport network.

4.2. Transshipment and Intermodality

As stated in [12], intermodal freigh transport is the movements of goods in one and the same loading unit or vehicle by sucessive modes of transport without handling of the goods themselves when changing modes. Intermodal transport can be used as an alternative to unimodal transport; in the last decades there is an increasing attention towards intermodality (specially because of environmental concerns, overall efficiency and growing transport flows). [12] contains a study of intermodal rail-truck transport; it considers the following ítems:

1. Drayage. This is the transport of godos over a short distance in logistics. Typically, it is carried out by trucks. Despite the relatively short distance of the truck movement compared with the rail line haul, drayage accounts for a large fraction of expenses, which limits the profitability of intermodal service. The mentioned study contains a number of specific research parts on drayage operations.

2. Rail haul. This is the terminal-to-terminal segment of the door-to-door intermodal trip. For a review on rail modelling see [21]. The main objective of intermodal rail haul is to organize the rail haul in an efficient, profitable and competitive way. With this global purpose, we can differentiate three levels of planning with respect to rail haul: strategic, tactical and operational planning. At the strategic level decisions consider which rail links to use, which origins and destination regions to serve, which terminals to use and where to locate them. At the tactical

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

level decisions consider train scheduling and routing, flows to consolidate, frequency and train length. The operational level deals with day-to-day management decisions on the load order of trains, redistribution of railcars and fleet management.

3. Transshipment: rail-road terminals and rail-rail terminals. Regarding trashipment, the classical case considers rail-road terminals, where the drainage is done by trucks. But rail-rain terminals are also possible, which require a number of changes with respect to conventional terminals (for example, size of shunting yards). This is an emerging área where a lack of methodological aspects have been detected [51].

4. Standarization. There is a basic consensus on the size of containers (20 feet or 40 feet). However, there still is a great deal of variation among load units, rail cars and truck-trailer skeletons. More research into the effects of standarization on intermodal efficiency is needed. It is reasonable to expect that further standarization would save costs, but this saving would appear when all actors in the supply-chain would use this standarization.

5. Multi-actor chain management and control. A major issue in transport chain is the existence of many actors, that operate at different points of the chain, under diverse national laws, which should provide a variety of information to the other actors in order to achieve a proper synchronization. Information and communication technology (ICT) provides new possibilities to support the complex chain co-ordination and control. The mentioned paper contains different studies on this topic.

6. Mode choice and pricing strategies. The general problema of intermodal transport is its competitiveness in relation to other models. We want to know for which markets intermodal transport is attractive, how intermodal performd compared to other modes, how market share can be increased, and which pricing strategy to follow. A large number of research papers considers these questions. After a number of studies, it is clear that actors do not decide on rational arguments only.

7. Intermodal transportation policy and planning. A general problema in intermodal policy is the formulation of effective measures that support policy objectives such as reducing congestion and pollution, and improving safety, but also with respect to spatial and economic objectives and infraestructure planning. Little is known with certainty on these dimensions.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

5. Ride-Sharing

The concept of ridesharing has recently been receiving significant attention, both in the transportation industry (thanks to companies such as *UberPool*, *Lyft*, and *Maramoja*) and in academia [3, 44, 37, 40, 10]. Despite the abundant literature on fleet management for *mobility-on-demand* (MoD) systems, the problem of servicing multiple rides with a single trip in large-scale dynamic scenarios has only very recently been considered. The approaches by Alonso *et al*. [3] and Pelzer *et al*. [37] introduce a carpooling matching algorithm, whose objective is to carpool passengers with the aim of maximizing *quality of service* (QoS), i.e, minimizing the delay experienced by the users. Bicocchi *et al*. [10] pursue the same objective, but they propose a recommendation-based approach that exploits historical data to compute potential matchings and minimize the inconvenience for the users. On the other hand, Rigby *et al*. [40] propose to shift from the usual discrete representation to a continuous space-time representation of vehicle accessibility to provide a client with a more realistic choice set. Despite being a significant contribution for *business-to-consumer ridesharing*, these works do not take into account the societal and environmental benefits of ridesharing, and, hence, they do not help policy makers in making the aforementioned assessments in the context of incentives design.

Finally, Santi *et al*. [44] evaluate the impact of ridesharing in terms of trade-off between the benefit and the passenger discomfort. However, such an approach is computationally limited to two riders (three with heuristics) and lacks the scalability needed to tackle large-scale ridesharing scenarios considered here, as also noted by Alonso *et al*. [3].

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

6. Conclusions

Although Automated Negotiations have been studied extensively in the literature, the topic has mainly focused on abstract toy-world problems. Very few papers have been published that tackle full scale real-world problems. Even less papers have focused on negotiation applied to logistics problems. All the papers discussed make simplifying assumptions that cannot be made if we aim to apply our work to real-world logistics problems.

The main problem is that currently existing negotiation algorithms assume that the value of any potential deal can be determined instantaneously, without any substantial computational effort. This of course, is unrealistic in the real world, where calculating the minimum cost of delivering a truck load can be a very complex problem.

Therefore, the challenge for us is to combine negotiation techniques with optimization algorithms. We have to use heuristic methods to estimate which co-loading and backhauling solutions may be mutually profitable for all parties involved, and then use optimization and re-optimization techniques to calculate the actual profit obtained from those solutions. After all, there may be millions of possible solutions, so it would be unfeasible to calculate the profit of every possible solutions. It is this combination of negotiation and optimization that will form a significant contribution to the state of the art.

Constraint Programming (CP) can be applied to a wide variety of relevant industrial problems. CP techniques have made it possible to make substantial progress in the problems of the Operations Research community (OR). Before the emergence of PC techniques, the usual approach to solving these problems was to model the problem so that the main objectives and constraints could be solved efficiently. If there were additional restrictions, they were often treated ad-hoc. Unfortunately, this leads to a wide variety of modeling, where each of them is specific to a particular problem. CP techniques allow us to handle the different types of constraints that appear in routing problems in a uniform and general manner. The formulations presented above allow us to model a very broad set of constraints in the field of logistics in the real world (capacity constraints, time, incompatibility, type of collection and delivery, ...) with the same set of variables and restrictions. In addition, they allow coding a great variety of secondary restrictions without affecting the central model. This allows incorporating them to the central model at a very low cost. We have shown how the expressiveness of the model allows us to define and also resolve constraints efficiently.

The routing problems that we have analyzed are all NP-hard. Therefore, it is important to traverse the search space efficiently. For this purpose, we can use propagators for generic constraints and other more specific restrictions, reducing the number of nodes actually visited, without affecting the quality of the solution. Local search can also help pruning the search space.

The greatest advantage of CP-based methods, compared to other OR methods, is their flexibility in dealing with new restrictions. These can be easily incorporated into the model, and automatic resolution methods can treat them. If later we see that they are fundamental to solve the problem, we can design new propagators for them.

In this project it seems that much remains to be done in terms of hybrid OR and CP methods. This is a

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

very active research area, where methods that apply techniques from both disciplines are developed. We also plan to focus our research efforts on the application of exact methods based on MaxSAT. Finally, we will take advantage of our experience in the development of systems for car-sharing to the re-optimization of co-loading plans.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

List of Abbreviations and Acronyms

3PL Third Party Logistics.

4PL Fourth Party Logistics.

CP Constraint Programming.

CVRP Capacitated Vehicle Routing Problem.

FCS Flat Communication Structure.

FCS Hierarchical Communication Structure.

GDA Greedy Distribution Algorithm.

GIS Geographical Information Systems.

GPS Global Positioning System.

ILP Integer Linear Programming.

LDS Limited Discrepancy Search.

LHS Left Hand Side.

MoD Mobility on Demand.

NP-hard Non-Polynomially Hard.

OR Operational Research.

PDP Pickup and Delivery Problem.

QoS Quality of Service.

RHS Right Hand Side.

RMA Risk Management Agent.

TSP Travelling Salesman Problem.

TSPTW Travelling Salesman Problem with Time Window.

VRP Vehicle Routing Problem.

VRPTW Vehicle Routing Problem with Time Windows.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

References

[1] Emile Aarts and Jan Karel Lenstra. *Local Search in Combinatorial Optimization*. Princeton University Press, 2003.

[2] Tony Allard and Slava Shekh. Hierarchical multi-agent distribution planning. In Michael Thielscher and Dongmo Zhang, editors, *AI 2012: Advances in Artificial Intelligence*, pages 755–766, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[3] Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proc. of the National Academy of Sciences*, 114(3):462–467, 2017.

[4] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2011.

[5] Bruno De Backer, Vincent Furnon, Paul Shaw, Philip Kilby, and Patrick Prosser. Solving vehicle routing problems using constraint programming and metaheuristics. *Journal of Heuristics*, 6(4):501–523, Sep 2000.

[6] A. J. Baird. Optimising the container transhipment hub location in northern europe. *Journal of Transport Geography*, 14(3):195 – 214, 2006.

[7] Adil Baykasoğlu and Vahit Kaplanoğlu. An application oriented multi-agent based approach to dynamic load/truck planning. *Expert Systems with Applications*, 42(15):6008–6025, 2015.

[8] Tolga Bektas. *Freight Transport and Distribution*. CRC Press, 2017.

[9] R. Bent and P. Van Hentenrick. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52:977–987, 2004.

[10] Nicola Bicocchi, Marco Mamei, Andrea Sassi, and Franco Zambonelli. On recommending opportunistic rides. *IEEE Trans. on Intelligent Transportation Systems*, 18(12):3328–3338, 2017.

[11] Csaba Boer, Alexander Verbraeck, and A. de Waal. Distributed e-services for road container transport simulation. In Alexander Verbraeck and Vlatka Hlupic, editors, *Simulation in Industry - 15th European Simulation Symposium*, pages 541–550, Delft, Erlagen en San Diego, 2003. SCS-European Publishing House.

[12] Y. M. Bontekoning, C. Macharis, and J. J. Trip. Is a new applied transportation research field emerging?, a review of intermodal rail-truck freight transport literature. *Transportation Research Part A: Policy and Practice*, 38(1):1–34, 2004.

[13] Olli Bräysy and Michel Gendreau. Tabu search heuristics for the vehicle routing problem with time windows. *Top*, 10(2):211–237, Dec 2002.

[14] Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, 39(1):104–118, February 2005.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

[15] Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*, 39(1):119–139, February 2005.

[16] Z. Chen and H. Xu. Dynamic column generation for dynamic vehicle routing with time windows. *Transportation Science*, 40:74–88, 2006.

[17] Wen-Chyuan Chiang and Robert A. Russell. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63(1):3–27, Feb 1996.

[18] Harry K. H. Chow, Winson Siu, Chi-Kong Chan, and Henry C. B. Chan. An argumentation-oriented multi-agent system for automating the freight planning process. *Expert Systems with Applications*, 40(10):3858–3871, 2013.

[19] G. Clarke and J. Wright. Scheduling of vehicles from as central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

[20] J.-F. Cordeau, G. Laporte, and A. Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52(8):928–936, Aug 2001.

[21] T. G. Crainic. *Long-Haul Freight Transportation*, pages 451–516. Springer US, Boston, MA.

[22] E. Danna and C. Le Pape. *Accelerating branch-and-price with local search: A case study on the vehicle routing problem with time windows*, page 99–130. Kluwer Academic Publishers, 2005.

[23] Dave de Jonge and Carles Sierra. Automated negotiation for package delivery. In *Sixth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASOW 2012, Lyon, France, September 10-14, 2012*, pages 83–88. IEEE Computer Society, 2012.

[24] Filippo Focacci, Francois Laburthe, and Andrea Lodi. *Local Search and Constraint Programming*, pages 293–329. Springer US, Boston, MA, 2004.

[25] M. Gendreau, F. Guertin, J. Y. Potvin, and R. Seguin. Parallel tabu search for real-time vehicle routing and dispaching. *Transportation Science*, 33:381–390, 1999.

[26] B. E. Gillett and L. R. Miller. A heuristic algorithm for the vehicle dispatching problem. *Operations Research*, 22, 04 1974.

[27] B. Golden, S. Raghavan, and E. Wasil. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, 2008.

[28] A. Haghani and S. Jung. A dynamic vehicle routing problem with time-dependent travel times. *Computers and Operations Research*, 32:2959–2986, 2005.

[29] R. Hall. Vehicle routing software survey. *OR/MS Today*, 31(3), 2004.

[30] J. I. Van Hemert and J. L. Poutre. Dynamic routing problems with fruitful regions: Models and evolutionary computation. In *Parallel Problem Solving from Nature*, volume 3242 of *Lecture Notes in Computer Science*, page 692–701. Springer, 2004.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

[31] L. Henesey. Overview of transhipment operations and simulation. In *Proceedings of the Med-Trade conference*, Malta, 2006.

[32] Holger Hoos and Thomas Stützle. *Stochastic Local Search*. Morgan Kaufmann, 2004.

[33] P. Kilby, P. Prosser, and P. Shaw. A comparison of traditional and constraint-based heuristic methods on vehicle routing problems with side constraints. *Constraints*, 5(4):389–414, 2000.

[34] E. L. Lawler, K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Traveling Salesman Problem: A Cuided Tour of Combinatorial Optimization*. Wiley, 1985.

[35] Han Noot, Valentin Robu, Han La Poutré, and Willem-Jan van Schijndel. A multi-agent platform for auction-based allocation of loads in transportation logistics. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Demo Papers*, AAMAS '08, pages 1699–1700, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[36] A. Orden. Transshipment problem. *Management Science*, 3:276–285, 1956.

[37] Dominik Pelzer, Jiajian Xiao, Daniel Zehe, Michael H Lees, Alois C Knoll, and Heiko Aydt. A partition-based match making algorithm for dynamic ridesharing. *IEEE Trans. on Intelligent Transportation Systems*, 16(5):2587–2598, 2015.

[38] Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés Medaglia. Any colony system for a dynamic vehicle routing problems. *Journal of Combinatorial Optimization*, 10:327–343, 2005.

[39] Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225:1–11, 2013.

[40] Michael Rigby, Stephan Winter, and Antonio Krüger. A continuous representation of ad hoc ridesharing potential. *IEEE Trans. on Intelligent Transportation Systems*, 17(10):2832–2842, 2016.

[41] Valentin Robu, Han Noot, Han La Poutré, and Willem-Jan van Schijndel. An interactive platform for auction-based allocation of loads in transportation logistics. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track*, AAMAS '08, pages 3–10, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[42] F. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming*. Elsevier, 2006.

[43] L.-M. Rousseau, M. Gendrea, and G. Pesant. Using constraint-based operators to solve the vehicle routing problem with time windows. *Journal of Heuristics*, 8(1):43–58, 2002.

[44] Paolo Santi, Giovanni Resta, Michael Szell, Stanislav Sobolevsky, Steven H Strogatz, and Carlo Ratti. Quantifying the benefits of vehicle pooling with shareability networks. *Proc. of the National Academy of Sciences*, 111(37):13290–13294, 2014.

[45] M. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*, 4:285–305, 1985.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.

[46] Paolo Toth and Daniele Vigo, editors. *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[47] Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem*. SIAM, 2002.

[48] Sander van der Putten, Valentin Robu, Han La Poutré, Annemiek Jorritsma, and Margo Gal. Automating supply chain negotiations using autonomous agents: A case study in transportation logistics. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, pages 1506–1513, New York, NY, USA, 2006. ACM.

[49] I. F. A. Vis and R. Koster. Transshipment of containers at a container terminal: An overview. *European Journal of Operational Research*, 147(1):1–16, 2003.

[50] J. Yang, P. Jaillet, and H. Mahmassani. Real-time multi vehicle truckload pickup and delivery problems. *Transportation Science*, 38:135–148, 2004.

[51] K. G. Zografos and I. M. Giannouli. A methodological framework for introducing its applications for improving the performance of intermodal freight terminals. pages 12–16, Seul, Korea, 1998.

LOGISTAR project has received funding from European Union's Horizon 2020 research and innovation programme under grant agreement no. 769142.