applied sciences

MDPI

Article

Route Planning Algorithms for Fleets of Connected Vehicles: State of the Art, Implementation, and Deployment

Mattia D'Emidio *,†, Esmaeil Delfaraz †, Gabriele Di Stefano †, Giannantonio Frittella † and Edgardo Vittoria †

Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, 67100 L'Aquila, Italy; esmaeil.delfarazpahlevanloo@univaq.it (E.D.); gabriele.distefano@univaq.it (G.D.S.); giannantonio.frittella@univaq.it (G.F.); edgardo.vittoria@student.univaq.it (E.V.)

* Correspondence: mattia.demidio@univaq.it; Tel.: +39-0862-434466

† These authors contributed equally to this work.

Abstract: The introduction of 5G technologies has enabled the possibility of designing and building several new classes of networked information systems that were previously impossible to implement due to limitations on data throughput or the reliability of transmission channels. Among them, one of the most interesting and successful examples with a highly positive impact in terms of the quality of urban environments and societal and economical welfare is a system of semi-autonomous connected vehicles, where IoT devices, data centers, and fleets of smart vehicles equipped with communication and computational resources are combined into a heterogeneous and distributed infrastructure, unifying hardware, networks, and software. In order to efficiently provide various services (e.g., patrolling, pickup and delivery, monitoring), these systems typically rely on collecting and broadcasting large amounts of data (e.g., sensor data, GPS traces, or maps), which need to be properly collected and processed in a timely manner. As is well documented in the literature, one of the most effective ways to achieve this purpose, especially in a real-time context, is to adopt a graph model of the data (e.g., to model communication networks, roads, or interactions between vehicles) and to employ suitable graph algorithms to solve properly defined computational problems of interest (e.g., shortest paths or distributed consensus). While research in this context has been extensive from a theoretical perspective, works that have focused on the implementation, deployment, and evaluation of the practical performance of graph algorithms for real-world systems of autonomous vehicles have been much rarer. In this paper, we present a study of this kind. Specifically, we first describe the main features of a real-world information system employing semi-autonomous connected vehicles that is currently being tested in the city of L'Aquila (Italy). Then, we present an overview of the computational challenges arising in the considered application domain and provide a systematic survey of known algorithmic results for one of the most relevant classes of computational problems that have to be addressed in said domain, namely, pickup and delivery problems. Finally, we discuss implementation issues, adopted software tools, and the deployment and testing phases concerning one of the algorithmic components of the mentioned real-world system dedicated to handling a specific problem of the above class, namely, the pickup and delivery multi-vehicle problem with time windows.

Keywords: applied algorithmics; autonomous vehicles; combinatorial optimization; algorithm engineering

Citation: D'Emidio, M.; Delfaraz, E.; Di Stefano, G.; Frittella, G.; Vittoria, E. Route Planning Algorithms for Fleets of Connected Vehicles: State of the Art, Implementation, and Deployment. *Appl. Sci.* **2024**, *14*, 2884. https://doi.org/10.3390/app14072884

Academic Editor: Mohammed Chadli

Received: 29 February 2024
Revised: 15 March 2024
Accepted: 26 March 2024
Published: 29 March 2024

1. Introduction

The field of telecommunications has undergone a transformative evolution in the domain of cellular networks with the introduction of 5G technologies, which have represented a substantial advancement in terms of transfer rates and latency and therefore have introduced breakthrough innovation in the design and deployment of information

systems that rely on effective, secure, and fault-tolerant data transmission. More specifically, such innovation has enabled the possibility of building entirely new classes of information systems and networked applications that were previously impossible to implement due to limitations in terms of data throughput or the reliability of communication channels [1]. Among the new types of systems that have experienced major diffusion, one of the most interesting and successful examples, due to the highly positive impact in terms of the quality of urban environments and societal and economical welfare, is a system of semi-autonomous connected vehicles.

In a typical system of semi-autonomous connected vehicles (commonly associated with the broader class of *intelligent transport systems*), we have a set (also known as *fleet*) of *smart* vehicles (or generically, *mobile entities*) that are equipped with high-end communication, sensing, and computational technologies and are deployed and can move/perform operations (with a certain, pre-specified degree of autonomy) in a given environment (e.g,. the road network of a city). Such vehicles are integrated and interact with IoT devices, computational units, data centers, satellites, and various sensing technologies to form a heterogeneous and distributed infrastructure, unifying hardware, networks, and software. The synergy between the fleet and the infrastructure forms a cohesive system designed to let vehicles perform complex tasks, often requiring coordination (e.g., pickup or delivery operations, search and rescue activities, traffic engineering [2,3]), while achieving the superior efficiency, safety, and adaptability of transportation within diverse and dynamic environments.

In order to achieve such levels of performance, these systems typically rely on collecting and broadcasting large amounts of data (e.g., sensor data, GPS traces, or maps), which need to be properly collected and processed in a timely manner in order to support tasks and to optimize various related services, such as route planning, task scheduling, or monitoring. Examples of studies that have considered such systems, as well as their design and performance analysis, are numerous and almost always of the multi-disciplinary type, involving computer scientists/engineers and experts in the telecommunications, materials science, electronics, and automotive fields (see, e.g., [4,5]).

In this direction, as is well documented in the literature, one of the most effective and widely used strategies, especially in a real-time context, to efficiently collect, store, and process data for decision-support and optimization purposes is to consider a graph model of the data and to employ suitable graph algorithms to solve properly defined computational problems of interest. Successful applications of graph-based modeling and graph algorithms for supporting the provision of services in modern information systems are various and span diverse domains, from navigation systems, where graphs are used to model and represent road networks and shortest-path algorithms are employed to determine the best routes [6,7], to airline management systems, where graphs are used to model flights connecting airports and maximum-flow algorithms are used to schedule departures and arrivals [8], up to distributed systems of entities, where graphs are used to model interactions between entities and coloring algorithms are exploited to achieve various forms of consensus or coordination [9,10]. Thus, on the one hand, research in this context has been extensive from a theoretical perspective in both computer science and engineering, mostly due to the plethora of domains where graph-based modeling and processing find applications [11–13]. On the other hand, works that have focused on the implementation, deployment, and evaluation of the practical performance of graph algorithms for real-world systems and specifically for systems of autonomous vehicles have been much rarer.

In this paper, we contribute to filling this gap in the literature in the following ways: (i) We describe the main features of a real-world information system that employs semi-autonomous connected vehicles; this system was designed and deployed and is currently under test in the city of L'Aquila (Italy) within the broader EMERGE initiative, a smart mobility and communications project to develop technologies for autonomous and assisted driving systems to support both "everyday" and "emergency" operations. (ii) We present

an overview of computational challenges arising in the considered application domain and provide a systematic survey of known algorithmic results for one of the most relevant classes of computational problems that have to be addressed in said domain, namely, *pickup and delivery problems*. (iii) We showcase the main phases of the implementation, deployment, and testing of one of the algorithmic components of the mentioned real-world system.

This paper is organized as follows. Section 2 describes, at a high level, the architecture of the real-world system considered here from both physical and virtual/software perspectives and discusses the three primary services that are enabled and currently supported by the EMERGE system through the integration of 5G technologies and smart vehicles. Section 3 introduces general algorithmic issues that arise in systems like EMERGE to effectively support such services and provides an overview, from a computational perspective, of the relevant class of problems named *pickup and delivery problems*, which is exploited for decision-making and optimization purposes in essentially all systems that employ fleets of semi-autonomous connected vehicles. In Section 4, we discuss implementation, deployment, and testing efforts that have been addressed within the EMERGE initiative with respect to the algorithmic components of the system. Specifically, among all algorithms that have been considered and implemented in said system, we provide details for the component that takes care of handling a specific problem of the above class, namely, the *pickup and delivery multi-vehicle problem with time windows*, whose solution is essential to coordinating vehicles to perform time-constrained operations requested by customers/users of the system. Section 5 concludes the paper by highlighting open problems and future research directions.

2. Architecture: The EMERGE System

In this section, we introduce the EMERGE system and describe its architectural details from both physical/hardware and software perspectives.

The EMERGE system (http://www.radiolabs.it/en/emerge/ accessed on 25 March 2024), or simply EMERGE, for short, is a heterogeneous platform capable of offering innovative navigation services through the integration of advanced solutions for location, communication, and cyber-security for smart vehicle equipment and distributed infrastructure dedicated to information management and service provision. Figure 1 shows a high-level overview of technologies, infrastructural components, users, and interactions between elements that form the EMERGE system. Specifically, note that the system is designed to manage and optimize various kinds of operational modes for a fleet of *smart* vehicles, i.e., vehicles equipped with both computational resources and communication, with high-throughput 5G-based capabilities. Vehicles are assumed to start their "operational" daily activity from a pre-defined depot and are then allowed to move in a geographical area, according to requests or plans provided to the system by external users, to perform various types of operations. In the process, vehicles can collect data via sensors and can communicate both with each other and with a dedicated two-level infrastructure via cyber-secure protocols. Such infrastructure is organized in a hierarchical fashion, with local units of the multi-access computing type placed close to vehicles and with limited computational capabilities (to handle basic tasks), and a single, remote, ground center (named *EMERGE Ground Control*, or *EGC* from here onward) with larger computational resources (to handle more complex tasks).

Both the vehicles and local units of multi-access computing have access to cloud-based resources for data collection and processing and to satellites (multi-frequency and multi-constellation satellite positioning systems for effective positioning and geo-referencing (also known as GNSSs, such as GPS or GALILEO)). In more detail, each vehicle is equipped with both standard (e.g., tachometer, odometer) and additional (e.g., IMU, LIDAR, RADAR) sensors, advanced aggregation/processing data platforms, and multi-protocol communication. Each vehicle can be customized with different technological setups depending on two possible pre-defined *operational modes*, namely, ordinary and emergency operational modes. The vehicles are *semi-autonomous* in the sense that they can have a driver to perform tasks

that are difficult to automate while at the same time having the ability to communicate and perform distributed tasks autonomously, without the driver's intervention. The driver can access the EMERGE network only through the vehicle, and there is no interaction between the driver's mobile device and the vehicle for EMERGE services.

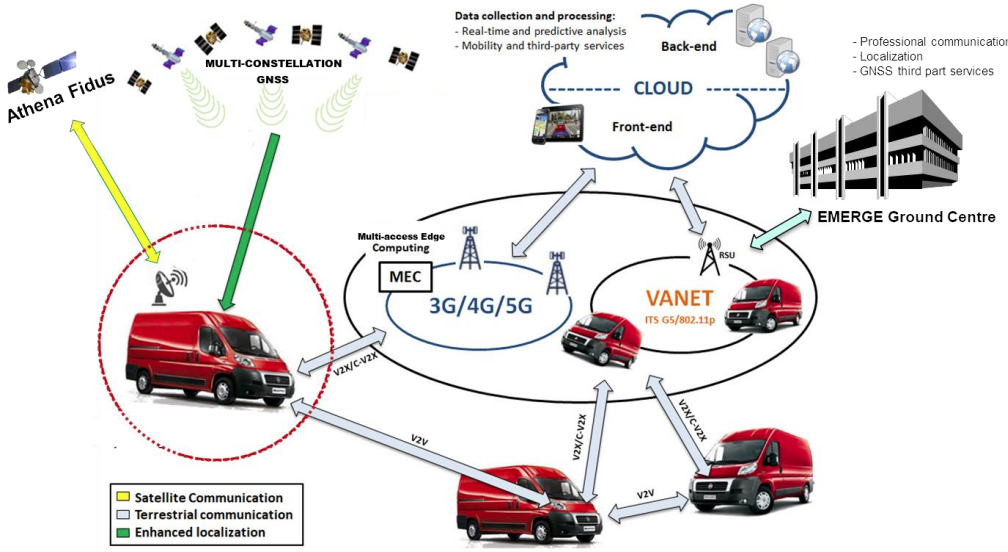

Figure 1. The EMERGE system: an overview of physical/hardware components and interactions.

In terms of software architecture, the above system organization translates into the software platform in Figure 2. On one side of the figure, we observe the vehicle component: each vehicle is equipped with an *EMERGE on-board unit (OBU)*, a software/hardware unit that takes care of performing computational tasks, data collection, and analysis and implements communication with both other vehicles and the infrastructure. Such communication happens through appropriate software modules providing support to transmit, securely and reliably (by exploiting a Crypto-engine unit), over both terrestrial (LTE/5G) and satellite (e.g., Athena Fidus) protocols. On the other side of the figure, we report a description of the software units running on both edge-computing infrastructure and EMERGE Ground Control (EGC). These units' responsibility is to support more complex optimization, analysis, and coordination tasks, such as the optimization of routes traveled by vehicles, the selection of locations to be visited by vehicles, and massive data analysis for traffic predictions. The vehicle equipment can be of any of the following three types, named *OBU configurations*, with different sets of supported services:

1. Configuration **EMERGE FULL:**
    - Supports emergency management services through dynamic and collaborative navigation applications with the use of preferential and/or dedicated lanes;
    - Facilitates efficient traffic flow management for "everyday" operations, mobility support services, and third-party services;
    - Includes modules for satellite communication, multi-standard terrestrial communication, advanced high-accuracy and high-integrity navigation techniques and algorithms, and 360-degree video surveillance of the vehicle's surrounding environment.
2. Configuration **EMERGE MEDIUM:**
    - Supports traffic flow management in "everyday" operations, mobility support services, and third-party services;
    - Includes multi-standard terrestrial communication, advanced high-accuracy and high-integrity navigation techniques and algorithms, and 360-degree video surveillance of the vehicle's surrounding environment.

3. Configuration **EMERGE SMALL:**
    - Supports traffic flow management in "everyday" operations, mobility support services, and third-party services;
    - Includes multi-standard terrestrial communication and Commercial Off-The-Shelf (COTS) GPS navigation systems;
    - Integrates 360-degree video surveillance of the vehicle's surrounding environment.

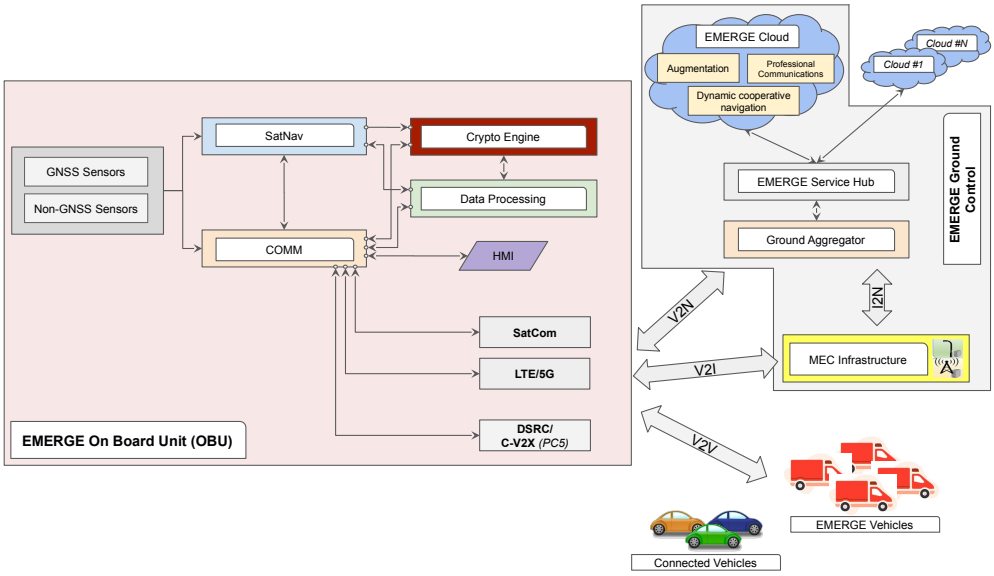

Figure 2. The software architecture of the EMERGE system.

Vehicles equipped with an EMERGE OBU are, if necessary, able to interface with external vehicles outside the EMERGE system equipped with V2V connectivity. In these cases, the reference V2X standards (DSRC or C-V2X) are used [14]. The terrestrial architecture consists of four main components:

1. Roadside infrastructure and MEC, which includes processing and communication elements placed along the road (RSU, eNB, gNB, etc.) for the local processing of information and support for URLLC services. The roadside infrastructure may potentially interface with external vehicles outside the EMERGE system equipped with V2I connectivity.
2. Ground aggregation of heterogeneous information flows.
3. EMERGE Service Hub, with integrated processing capabilities for information from the field, both locally and remotely through the EMERGE Cloud, for the implementation of EMERGE applications and the provision of EMERGE services to third parties.
4. EMERGE Cloud, dedicated to the remote processing of information for specific dynamic and cooperative navigation applications, augmentation, and professional communications in support of EMERGE services.

We refer the interested reader to [14,15] and references therein for more details on some employed technologies and operational modes.

Services and Computational Tasks

Given the above organization and architecture, the following are identified as the three primary services provided by the EMERGE system and its fleet and currently (either fully or partially) supported by the deployed prototype under test:

1. *Vehicle task planning:* Given a set of "goal" operations to be performed by vehicles in the fleet in a geographical zone and in specified time windows of the day (e.g., pickup or delivery operations), provided as input, the system determines a detailed scheduling plan for each vehicle, including routes to be followed, departure or waiting times, the

order of the locations to visit, and items to carry, to allow the vehicles to accomplish all goals while optimizing metrics of interest (e.g., fuel consumption);
2. *Network monitoring:* Continuous, real-time, data acquisition is performed by vehicles moving in the given geographical zone of interest through their sensor equipment for monitoring (and predicting) the status of the network over time (traffic, disruptions, congestions, detours);
3. *React and reoptimize:* To respond to changes in the status of the network (e.g., disruptions, roadworks, emergency vehicles) or that of the vehicles (e.g., malfunctions), vehicles coordinate and cooperate (by communicating and exchanging data both with the infrastructure and between each other) in order to update/reoptimize the scheduling plan to consider status changes that have occurred while continuing with the assigned operations.

In order to provide these services, appropriate data models, computational problems, and algorithms have been identified, studied, and implemented to be part of the software architecture of the EMERGE system, respectively.

3. Computational Problems and Algorithms

In this section, we describe and formalize the computational problems that have been identified and addressed throughout the phases of development of the EMERGE system in order to provide services of interest through the platform. Specifically, we first walk through relevant variants of the pickup and delivery problem that have been studied in the literature and for which algorithmic results with guarantees are known. The discussion is organized by following a classification of problem variants in terms of optimization objectives. All problems consider a graph discretization, based on different levels of detail, of the environment in which vehicles move. Most of the considered problems are known to be hard to solve exactly, i.e., to belong to the class of NP-hard optimization problems (due to reductions to variants of the well-known Traveling Salesman Problem (TSP) [8]; for a survey on the TSP and its variants, the reader is referred to [16]). Therefore, our survey focuses only on available options with guarantees on the approximation factor in the worst case. Then, we consider the pickup and delivery problem with time windows and multiple vehicles, for which we had to resort to heuristics without guarantees, since no approximation algorithm with guarantees is known to be practically effective.

3.1. Maximum Prize Problems

Orienteering Problem (OP). In this first basic problem formalization, we are given a graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges, weighted with a cost function $l : E \to \mathbb{R}^{\geq 0}$ on the edges and a prize function $\pi : V \to \mathbb{R}^{\geq 0}$ on the vertices, a budget $B \in \mathbb{R}^{>0}$ and two distinguished nodes s, t, and we are asked to find a walk P_{st} from s to t that: (i) satisfies the budget constraints, i.e., the sum of the costs of the edges in the walk is smaller than or equal to B; (ii) maximizes a collected prize for the walk, where the collected prize of a walk is given by the sum of the prizes associated to the vertices in the walk. More formally, the problem can be described as in Figure 3.

Problem: Orienteering Problem (OP)
Input: an undirected graph $G = (V, E)$, a prize function $\pi : V \to \mathbb{R}^{\geq 0}$, a cost function $l : E \to \mathbb{R}^{\geq 0}$, specific nodes $s, t \in V$ and a budget $B \in \mathbb{R}^{>0}$.
Feasible Solution: a walk P_{st} s.t. $l(P_{st}) = \sum_{e \in E(P_{st})} l(e) \leq B$.
Goal: $\pi(P_{st}) = \sum_{v \in V(P_{st})} \pi(v)$ maximized.

Figure 3. Formalization of OP.

Due to the plethora of applications that the problem finds, the literature on algorithms for finding approximate solutions with guarantees is ample. Specifically, Blum et al. [17] gave the first constant factor approximation algorithm for OP with approximation ratio of 4 when $s = t$ and showed that no polynomial-time approximation algorithm can achieve a factor better than $\frac{1481}{1480}$ to OP. In the same paper, the authors also showed that OP is APX-hard

while Bansal et al. [18] improved the bound of Blum et al. by designing an algorithm with approximation ratio of 3 when $s = t$. Moreover, Chekuri et al. [19] proposed a $2 + \varepsilon$ approximation algorithm that works for any positive constant ε while Friggstad and Swamy [20] designed, via LP-rounding, a 3-approximation algorithm when $s = t$. The solution given by Paul et al. [21], instead, is a 2-approximation algorithms that solves OP when s and t are not given in advance. Finally, Chen and Har-Peled [22] gave a polynomial-time approximation scheme (PTAS) when the points lie in a constant dimensional Euclidean metric.

Orienteering Problem with Time Windows (OPTW). In this problem variant (formalization is given in Figure 4), the goal is to find a walk from a starting node to an ending node that visits a set of vertices with the maximum profit within their time windows respecting a given budget B. Given a walk P_{uv} from u to v, for any $z \in V(P_{uv})$, let $t_P(u, z)$ denote the time taken (distance) to reach $z \in V(P_{uv})$ from u along P_{uv}. Note that, the OPTW problem is also known as the Repairman problem, (Speeding) Delivery Man problem [23], Time Window Prize Collecting problem [24], and Prize-Collecting TSP with Time Windows [25].

Problem: Orienteering Problem with Time Windows (OPTW)
Input: an undirected graph $G = (V, E)$, a prize function $\pi : V \to \mathbb{R}^{\geq 0}$, a cost function $l : E \to \mathbb{R}^{\geq 0}$, a deadline function on nodes $D : V \to \mathbb{R}^{\geq 0}$, a release function on nodes $R : V \to \mathbb{R}^{\geq 0}$, specific nodes $s, t \in V$ and a budget $B \in \mathbb{R}^{>0}$.
Feasible Solution: a walk P_{st} s.t. $l(P_{st}) = \sum_{e \in E(P_{st})} l(e) \leq B$.
Goal: $\pi(P_{st}) = \sum_{v \in V(P_{st}) : R(v) \leq t_P(v) \leq D(v)} \pi(v)$ maximized.

Figure 4. Formalization of OPTW.

Bansal et al. [18] gave an $O(\log^2(n))$ approximation algorithm for OPTW when $s = t$. They also proposed an $O(\log(1/\varepsilon))$ approximation algorithm for OPTW when $s = t$ if exceeding the deadlines by a factor of $1 + \varepsilon$ is permitted. Chekuri and Kumar [26] gave a constant factor approximation algorithm for the special case of this problem when the number of time windows is a constant. Given an α approximation for OP, Chekuri et al. [19] proposed an $O(\alpha \cdot \max\{\log OPT, \log \frac{L_{max}}{L_{min}}\})$ approximation for OPTW, where: (i) $OPT \leq n$ is the number of vertices visited by an optimal solution; (ii) L_{max} and L_{min} are the lengths of the longest and shortest time windows, respectively.

Bar-Yehuda et al. [25] provided two approximation algorithms for OPTW on a line. The first algorithm is an 8-approximation algorithm, where its complexity time is $O(n^2)$. The second one is a $(4 + \varepsilon)$-approximation algorithm and its complexity time is $O(n^8/\varepsilon)$. Perez et al. [27] provided a 4-approximation algorithm for OPTW with unitary length time windows on a line whose complexity time is $O(n^2)$. Miguel and Pilar [28] proposed an integer linear programming approach for OPTW on a line with unit time windows, which produces approximate solutions within a factor of 4 to the optimal solution. Frederickson and Wittman [23] studied OPTW with unit time windows. For this variant, they provided $6 + \varepsilon$ and 3 approximation algorithms on graphs and trees respectively. In the case when all time windows have length in $[1, 2]$, Frederickson and Wittman [23] showed that OPTW admits a 10-approximation algorithm.

Finally, Garg et al. [29] studied a different variant of OPTW where instead of having a time window for each node v, each node v specifies a subset T_v of available time slots when the tour can visit it and the goal is to find a tour starting from s that maximizes the number of visited nodes. They call this problem *OrientMTW* and show that, unless class NP is contained in class $DTIME(n^{O(\log n)})$, there is no polynomial-time $o(\frac{\log \log n}{\log \log \log n})$-approximation algorithm for OrientMTW even if we restrict the graph topology to trees.

Deadline Traveling Salesman Problem (D-TSP). A well-known variant of OPTW is the Deadline Traveling Salesman Problem (D-TSP). In this case, the release time of each node is assumed to be zero, i.e., $R(v) = 0$ for any vertex v (formally defined in Figure 5). Since D-TSP is a special case of OPTW, all results for OPTW hold for D-TSP. Moreover, Bansal et al. [18] gave an $O(\log n)$ approximation algorithm for D-TSP while Farbstein and Levin [30] proposed a $((1 + \varepsilon).\alpha)$-approximation algorithm for every $\varepsilon > 0$ for D-TSP while exceeding

the deadlines by a factor of $1+\varepsilon$, where α is the approximation ratio for D-TSP with a constant number of deadlines (currently $\alpha = 3$ by Chekuri and Kumar [26]). Observe that, on weighted trees, D-TSP admits $(1+\varepsilon)$-approximation algorithm while violating the deadlines by a factor of $1+\varepsilon$ [30]. Finally, Friggstad and Swamy [31] proposed a $(7.63+\varepsilon)$-approximation algorithm for D-TSP in $n^{O(\log n\Delta)}$ time, where n is the number of points (nodes), and Δ is the diameter of the (scaled) metric space.

Problem: Deadline Traveling Salesman Problem (D-TSP)
Input: an undirected graph $G = (V, E)$, a prize function $\pi : V \to \mathbb{R}^{\geq 0}$, a cost function $l : E \to \mathbb{R}^{\geq 0}$, a deadline function on nodes $D : V \to \mathbb{R}^{\geq 0}$, specific nodes $s \in V$.
Feasible Solution: a walk P starting at s.
Goal: $\pi(P) = \sum_{v \in V(P): t_P(v) \leq D(v)} \pi(v)$ maximized.

Figure 5. Formalization of D-TSP.

Capacitated Orienteering Problem (C-OP). The C-OP is a generalization of OP in which we also consider node demands $r : V \to \mathbb{N}$ and a capacity bound C. Formally, the problem is specified as in Figure 6 Perhaps the most remarkable achievement with respect to this problem is the result of Gupta et al. [32] who showed that, given an α-approximation algorithm for OP, it is possible to derive a 2α-approximation algorithm for C-OP. Bock and Sanità [33] improved this result by giving a $(1+\alpha+\varepsilon)$-approximation algorithm for C-OP and by presenting a PTAS on trees and a PTAS on Euclidean metrics.

Team Orienteering Problem (TOP). This problem is the first of a different category that considers optimizing operations of multiple vehicles. Specifically, the problem has been formalized (see Figure 7) and studied for the first time by Xu et al. [34] is as follows. We are given a graph $G = (V \cup \{s, t\}, E)$ and two specific nodes s and t. Notice that nodes s and t may or may not be co-located. There is an edge in E between any two nodes in $V \cup \{s, t\}$ and there are $K \geq 1$ vehicles to serve the nodes in V. All vehicles are located at source node s initially and, for any vehicle k, we have a cost function $c_k : E \to \mathbb{R}^{\geq 0}$ on the edges (representing, e.g., the traveling cost of vehicle k from one vertex to another) and a cost function $h_k : V \to \mathbb{R}^{\geq 0}$ on the nodes representing, e.g., the service cost of vehicle k at each vertex). We assume that $h(s) = h(t) = 0$. Then, the cost $w(P_{st}^k)$ of a simple path P_{st}^k from s to t is given by:

$$w(P_{st}^k) = \sum_{v \in V(P_{st}^k)} h_k(v) + \sum_{e \in E(P_{st}^k)} c_k(e).$$

For any vehicle k, let B_k denote the cost budget of vehicle k, meaning that $w(P_{st}^k) \leq B_k$. Moreover, for any vertex v_i, let n_i be the number of vehicles among K vehicles that serve v_i. For any vertex v_i, let $u_i : K \to \mathbb{R}^{\geq 0}$ be a non-decreasing sub-modular profit function, where $u_i(n_i)$ represents the profit that we accumulate after serving vertex v_i with n_i vehicles.

Problem: Capacitated Orienting Problem (C-OP)
Input: an undirected graph $G = (V, E)$, a prize function $\pi : V \to \mathbb{R}^{\geq 0}$, a cost function $l : E \to \mathbb{R}^{\geq 0}$, a node demand function $r : V \to \mathbb{N}$, specific nodes $s, t \in V$, a budget $B \in \mathbb{R}^{>0}$ and a capacity $C \in \mathbb{N}$.
Feasible Solution: a walk P_{st} s.t. $l(P_{st}) = \sum_{e \in E(P_{st})} l(e) \leq B$ and $r(P_{st}) = \sum_{v \in V(P_{st})} r(v) \leq C$.
Goal: $\pi(P_{st}) = \sum_{v \in V(P_{st})} \pi(v)$ maximized.

Figure 6. Formalization of C-OP.

The objective in TOP, in G, is to find K paths $P_{st}^1, P_{st}^2, \ldots, P_{st}^K$ for K vehicles each starting from node s and ending at t, such that the profit sum of the nodes served by the K vehicles, i.e., $\sum_{v_i \in \bigcup_k^k P_{st}^k} u_i(v_k)$, is maximized, subject to that the cost $w(P_k)$ of each path P_{st}^k for vehicle k is no greater than its cost budget B_k, i.e., $w(P_{st}^k) \leq B_k$ with $1 \leq k \leq K$. That is, we seek:

$$\max \sum_{v_i \in \bigcup_k^k P_{st}^k} u_i(v_k)$$

subject to

$$w(P_{st}^k) \leq B_k, 1 \leq k \leq K$$

Xu et al. [34] showed that There is a $1/(1-e^{-\alpha})$-approximation algorithm for TOP, where e is the base of the natural logarithm, and α is an approximation factor to OP.

Problem: Team Orienteering Problem (TOP)
Input: an undirected graph $G = (V, E)$, for any vertex $v_i \in V$ a prize function $u_i : K \to \mathbb{R}^{\geq 0}$, a set of K vehicles, for any vehicle $k \in [K]$, an edge cost function $l_k : E \to \mathbb{R}^{\geq 0}$, a node cost function $h_k : V \to \mathbb{R}^{\geq 0}$ and a budget $B_k \in \mathbb{R}^{>0}$, and specific nodes $s, t \in V$.
Feasible Solution: a walk P_{st}^k for any vehicle $k \in [K]$ s.t. $w(P_{st}^k) = \sum_{e \in E(P_{st}^k)} l_k(e) + \sum_{v \in V(P_{st}^k)} h_k(v) \leq B_k$ for any $k \in [K]$.
Goal: $\sum_{v_i \in \bigcup_k^k P_k} u_i(v_k)$ maximized.

Figure 7. Formalization of TOP.

Prior to these works, a simpler variant of TOP was studied by Blum et al. [17]. In such variant, for any two vehicles $k, j \in [K]$, any edge $e \in E$, and any node $v \in V$, we have that: (i) $l_k(e) = l_j(e)$ (i.e., the cost of any edge is the same for all vehicles); (ii) $\pi_k(v) = \pi_j(v) = \pi(v)$ (i.e., the prize of any node is the same for all vehicles) and the prize function π is additive; (iii) $B_k = B_j$ (i.e., all vehicles have identical budget); (iv) $h_k(v) = 0$ (i.e., for any vehicle $k \in [K]$, the cost of any vertex is 0). Furthermore, all the vehicles have the same starting point s, and the end point is arbitrary. Blum et al. [17] called this problem *Multi-Path Orienteering* and showed that any α approximation for OP when $s = t$, can be translated into a $1/(1-e^{-\alpha})$ approximation for Multi-Path Orienteering. Blum et al. [17] also showed that their algorithm has a factor of $\alpha + 1$ when the starting point of each vehicle is arbitrary in Multi-Path Orienteering. Friggstad et al. [35] studied a variant of Multi-Path Orienteering in the case where each vehicle needs to find a tour (i.e., $s = t$), each node has a cost, and for any two tours P_k, P_j, $V(P_k) \cap V(P_j) = \{s\}$. The goal is to find K tours so that the minimum total prize among all tours is maximized, i.e., $\max \min_P \pi(P)$. They called this problem *max-min* orienteering and showed that any α-approximation algorithm for OP results in an $(\alpha + 2)$-approximation for *max-min* orienteering.

To complete the overview, it is worth mentioning the work by Xu et al. [36] who focused on TOP when $s = t$, the prize function for each vehicle is additive and each vertex should be visited once. Such variant has been named *monitoring reward maximization* problem and in [36] a corresponding 3-approximation for the problem has been presented.

Capacitated Team Orienteering Problem (C-TOP). C-TOP is a generalization of TOP in which we also consider node demands $r : V \to \mathbb{N}$ and a capacity bound C for each walk (vehicle). For such generalization, formalized in Figure 8, Bock and Sanità [33] designed a $(1 - e^{\frac{1}{\alpha}})$-approximation algorithm, where α is an approximation factor for C-OP, under the following assumptions:

- each vehicle has the same budget B and the same capacity C;
- for each vehicle, the cost of each vertex is 0, the cost of each edge is the same and the prize of each vertex is the same.

It is worth pointing out that Archetti et al. [37] proved that a β-approximation algorithm for C-TOP can be turned into a 2β-approximation algorithm for the *Capacitated Team*

Orienteering with Split Deliveries, a variant of C-TOP in which a node demand can be served by several walks (vehicles).

Problem: Capacitated Team Orienteering Problem (C-TOP)
Input: an undirected graph $G = (V, E)$, for any vertex $v_i \in V$ a prize function $u_i : K \to \mathbb{R}^{\geq 0}$ and a demand $r(v_i) \in \mathbb{N}$, a set of K vehicles, for any vehicle $k \in [K]$, an edge cost function $l_k : E \to \mathbb{R}^{\geq 0}$, a node cost function $h_k : V \to \mathbb{R}^{\geq 0}$, a budget $B_k \in \mathbb{R}^{>0}$ and a capacity $C_k \in \mathbb{N}$, and specific nodes $s, t \in V$.
Feasible Solution: a walk P_{st}^k for any vehicle $k \in [K]$ s.t. for any $k \in [K]$, $w(P_{st}^k) = \sum_{e \in E(P_{st}^k)} l_k(e) + \sum_{v \in V(P_{st}^k)} h_k(v) \leq B_k$ and $r(P_{st}^k) = \sum_{v \in V(P_{st}^k)} r(v) \leq C_k$.
Goal: $\sum_{v_i \in \bigcup_k^k P_k} u_i(v_k)$ maximized.

Figure 8. Formalization of C-TOP.

A summary of the results on OP and its variants is given in Table 1. In such summary we have $\varepsilon > 0$ while α and α' are approximation factors to OP and its directed version, respectively. Parameter β is an approximation factor to C-OP while OPT is the cost of the optimal solution to the problem. Moreover, L_{max} and L_{min} are the longest and shortest time windows, respectively. Finally, γ is an approximation ratio for D-TSP with a constant number of deadlines (currently the best known factor is $\gamma = 3$ by Chekuri and Kumar [26]). Note that a (a, b)-bicriteria approximation algorithm to OPTW is the one which has a b-approximation factor and violates the time windows by a factor of a.

Reoptimization of the Metric Deadline Traveling Salesman Problem (MD-TSP). In the Metric Deadline Traveling Salesman Problem (MD-TSP), we are given a *metric* graph $G = (V, E)$ (i.e., a graph where costs on the edges satisfy the triangle inequality), a specific node $s \in V$, and the goal is to find a Hamiltonian cycle, starting at s, having minimum cost and that visits each vertex v before its deadline $D(v)$, assuming that there exists at least one Hamiltonian cycle P in which each vertex is visited before its deadline (see Figure 9 for details on the formalization). The time of visit $t_P(v)$ of a vertex v is given by the sum of the costs of the edges of the sub-path of the Hamiltonian cycle starting at s and terminating at v. The problem has interest itself with respect to applications, but perhaps the MD-TSP has been investigated more from a re-optimization perspective, where one is given a good (even optimal) solution to the problem and some modifications that affect the original input, and the objective is to rearrange the given solution to obtain a provably good new solution for the new input, with a minimal number of operations.

Table 1. A summary of known results for OP and variants.

Problem	Best Known Approximation Factor	
	Undirected	**Directed**
OP	$2 + \varepsilon$ [19], 2 [21] (Unrooted)	$\frac{\log^2 n}{\log \log n}$ [38]
OPTW	$O(\alpha . \max\{\log OPT, \log \frac{L_{max}}{L_{min}}\})$ [19], $6 + \varepsilon$ [23] (Unit time windows)	$O(\alpha' . \max\{\log OPT, \log \frac{L_{max}}{L_{min}}\})$ [19]
D-TSP	$(1 + \varepsilon, \gamma(1 + \varepsilon))$ [30], $7.63 + \varepsilon$ [31] (Quasi polynomial-time)	$O(\alpha' . \max\{\log OPT, \log \frac{L_{max}}{L_{min}}\})$ [19]
C-OP	$(1 + \alpha + \varepsilon)$ [33]	-
TOP	$(1 - \frac{1}{e^{\alpha}})$ [34]	-
C-TOP	$(1 - \frac{1}{e^{\beta}})$ [33]	-

Problem: Reoptimization of the Metric Deadline Traveling Salesman Problem (MD-TSP)
Input: an undirected metric graph $G = (V, E)$, a cost function $l : E \to \mathbb{R}^{\geq 0}$, a deadline function on nodes $D : V \to \mathbb{R}^{\geq 0}$, specific node $s \in V$.
Feasible Solution: a Hamiltonian cycle P starting at s s.t. for any $v \in V(P)$, $t_P(v) \leq D(v)$
Goal: $c(P) = \sum_{e \in E(P)} c(e)$ minimized.

Figure 9. Formalization of MD-TSP.

In details, Böckenhauer and Komm [39], Böckenhauer et al. [40] and Böckenhauer et al. [41] defined and studied some reoptimization variants of MD-TSP under the following assumptions. We assume $G_O = (V_O, E_O)$ and $G_N = (V_N, E_N)$ are two complete undirected graphs with metric edge cost functions $c_O : E_O \to \mathbb{R}^{\geq 0}$ and $c_N : E_N \to \mathbb{R}^{\geq 0}$. Furthermore, we let $D_O : V_O \to \mathbb{R}^{\geq 0}$ be a deadline function for G_O and $D_N : V_N \to \mathbb{R}^{\geq 0}$ be a deadline function for G_N such that (G_N, c_N, D_N) can be constructed from (G_O, c_O, D_O) by a local modification. Böckenhauer and Komm [39] considered the following six local modifications:

- LM(D^-): Deletion of a deadline: In this case, we have $(G_O, c_O) = (G_N, c_N)$, $D_N(v) \neq D_O(v)$ for some $v \in V_N = V_O$, where $D_N(v) = \infty$ and $D_O(v) = d$ for some $d \in \mathbb{R}^{\geq 0}$, and $D_N(u) = D_O(u)$ for any $u \in V_N \setminus \{v\}$.
- LM(D^+): Addition of a deadline to an already existing vertex: In this case, we have $(G_O, c_O) = (G_N, c_N)$, $D_N(v) \neq D_O(v)$ for some $v \in V_N = V_O$, where $D_O(v) = \infty$ and $D_N(v) = d$ for some $d \in \mathbb{R}^{\geq 0}$, and $D_N(u) = D_O(u)$ for any $u \in V_N \setminus \{v\}$.
- LM(V^-): Deletion of a vertex without deadline: In this case, we have $V_N = V_O \setminus \{v\}$ for some $v \in V_O$, where $D_O(v) = \infty$, and E_N and c_N are the canonical restriction of E_O and c_O to the vertices of V_N, and $D_N(u) = D_O(u)$ for any $u \in V_N$.
- LM(V^+): Addition of a vertex without deadline: In this case, we have $V_O = V_N \setminus \{v\}$ for some $v \in V_N$ where $D_N(v) = \infty$, E_O and c_O are the canonical restriction of E_N and c_N to the vertices of V_O, and $D_N(u) = D_O(u)$ for any $u \in V_O$.
- LM($(D \wedge V)^-$): Deletion of a vertex with deadline: In this case, we have $V_N = V_O \setminus \{v\}$ where $D_O(v) = d$ and $d \in \mathbb{R}^{\geq 0}$, E_N and c_N are the canonical restriction of E_O and c_O to the vertices of V_N, and $D_N(u) = D_O(u)$ for any $u \in V_N$.
- LM($(D \wedge V)^+$): Addition of a vertex with deadline: In this case, we have $V_N = V_O \setminus \{v\}$ where $D_N(v) = d$ and $d \in \mathbb{R}^{\geq 0}$, E_O and c_O are the canonical restriction of E_N and c_N to the vertices of V_O, and $D_N(u) = D_O(u)$ for any $u \in V_O$.

Moreover, Böckenhauer et al. [40] and Böckenhauer et al. [41] considered the following two local modifications:

- LM(D): change deadline: In this case, we have $(G_O, c_O) = (G_N, c_N)$, $D_N(v) \neq D_O(v)$ for some $v \in V_N = V_O$, where for $1 \leq \xi < n$ $D_N(v) = D_O(v) + \xi$ (case of increase) or $D_N(v) = D_O(v) - \xi$ (case of decrease) and $D_N(u) = D_O(u)$ for any $u \in V_N \setminus \{v\}$.
- LM(E): change edge cost: In this case, $c_O(e) \neq c_N(e)$ for some $e \in E_N = E_O$, where, for some $1 \leq \xi < n$, $E_N(v) = D_O(v) + \xi$ (case of increase) or $E_N(v) = D_O(v) - \xi$ (case of decrease). Also, for any $u \in V_N = V_O$, $D_N(u) = D_O(u)$.

For $X \in \{D, E, D^-, D^+, V^-, V^+, (D \wedge V)^-, (D \wedge V)^+\}$, Böckenhauer and Komm [39], Böckenhauer et al. [40] and Böckenhauer et al. [41] defined the problem LM(X)-MD-TSP as to find an optimum solution for the MD-TSP instance (G_N, c_N, D_N), given the MD-TSP instance (G_O, c_O, D_O) together with an optimal solution $\bar{C}$ for it and an arbitrary feasible solution $\tilde{C}$ for (G_N, c_N, D_N). Moreover, for any constant k, let LM(X)-k-MD-TSP denote the subproblem of LM(X)-MD-TSP, where k is the cardinality of a set $S \subseteq V$, where for any $v \in S$, $D_N(v) \in \mathbb{R}^{\geq 0}$ and for any $u \in V \setminus S$, $D_N(u) = \infty$. The results on this subject from Böckenhauer and Komm [39], Böckenhauer et al. [40] and Böckenhauer et al. [41] are summarized in Table 2.

Table 2. A summary of the results on some re-optimization variants of MD-TSP.

Local Modification	Bounds on Approximation Factor			
	LM(X)-k-MD-TSP		LM(X)-MD-TSP	
	Lower	Upper	Lower	Upper
Add vertex without deadline (LM(V^+))	$2-\varepsilon$ [39]	2 [39]	$2-\varepsilon$ [39]	2 [39]
Delete vertex without deadline (LM(V^-))	$2-\varepsilon$ [39]	2 [39]	$2-\varepsilon$ [39]	2 [39]
Add deadline to existing vertex (LM(D^+))	$2-\varepsilon$ [39]	2.5 [39]	$(0.5-\varepsilon)n$ [39]	$0.5n$ [39]
Delete deadline from vertex (LM(D^-))	$2-\varepsilon$ [39]	2.5 [39]	$(0.5-\varepsilon)n$ [39]	$0.5n$ [39]
Add vertex with deadline (LM($(V \wedge D)^+$))	$2-\varepsilon$ [39]	2.5 [39]	$(0.5-\varepsilon)n$ [39]	$0.5n$ [39]
Delete vertex with deadline (LM($V \wedge D^-$))	$2-\varepsilon$ [39]	2.5 [39]	$(0.5-\varepsilon)n$ [39]	$0.5n$ [39]
Increase deadline LM(D)	$2-\varepsilon$ [40,41]	2.5 [40,41]	$(0.5-\varepsilon)n$ [40,41]	$0.5n$ [40,41]
Decrease deadline LM(D)	$2-\varepsilon$ [40,41]	2.5 [40,41]	$(0.5-\varepsilon)n$ [40,41]	$0.5n$ [40,41]
Increase edges cost (LM(E))	$2-\varepsilon$ [40,41]	2.5 [40,41]	$(0.5-\varepsilon)n$ [40,41]	$0.5n$ [40,41]
Decrease edge cost (LM(E))	$2-\varepsilon$ [40,41]	2.5 [40,41]	$(0.5-\varepsilon)n$ [40,41]	$0.5n$ [40,41]

3.2. Minimum Cost Problems

Capacitated Vehicle Routing Problem with Time Windows (C-VRPTW). Khachay and Ogorodnikov [42] considered the following problem. We are given a set $X = \{x_1, \ldots, x_n\}$ of points (costumers) and a depot y on the Euclidean plane. Any customer $x_i \in X$ has a demand d_i. We are given an unbounded number of vehicles each with capacity q. Each $x \in X$ is assigned with time window T_x. A feasible route is an ordered pair $R_j = (P_j, D_j)$, where $P_j = (y, x_{i_1}, x_{i_2}, \ldots, x_{i_s}, y)$ represents the simple cycle taken by vehicle j in which $T_{i_z} \leq T_{i_{z+1}}$ for any $z \in [s]$, and $D_j = (d_{1j}, \ldots, d_{nj})$ in which d_{ij} is a part of the i-th customer demand covered by the route R_j and $1 \leq d_{ij} \leq d_i$.

The goal is to find a set of simple cycles starting at y such that $\sum_{i=1}^{n} d_{ij} \leq q$ for any rout R_j, $\sum_{j=1}^{K} d_{ij} = d_i$ for any customer x_i, and minimizes $\sum_{j=1}^{K} c(R_j) = \sum_{j=1}^{K} \sum_{e \in E(P_j)} c(e)$. For any $\varepsilon \in (0, 1)$ and the total customer demand d, Khachay and Ogorodnikov [42] provided a $(1+\varepsilon)$-approximation algorithm in time $O(n \log d)$ any time provided the capacity q and the number p of time windows does not exceed $2^{\log^{\delta} n}$ for some $\delta = O(\varepsilon)$.

Khachay and Ogorodnikov [43] provided a $(1+\varepsilon)$-approximation algorithm for C-VRPTW with a better running time than that of [42], which is $O(n^3 + \exp(\exp(\frac{1}{\varepsilon})))$. Khachay and Ogorodnikov [44] further investigated C-VRPTW when the number of depots is more than one and gave $(1+\varepsilon)$-approximation algorithm in time $O(n^3 + K^{O(\frac{1}{\varepsilon^2})} \exp(\frac{1}{\varepsilon^3}))$, where K is the number of depots.

Ordered Clustered Traveling Salesman Problem (OC-TSP). In this problem (also formalized in Figure 10 for the sake of clarify), a vehicle starting and ending at a given depot must visit a set of n points partitioned into K not necessarily disjoint clusters so that points of cluster k are visited prior to points of cluster $k+1$, for $k = 1, 2, \ldots, K-1$, and the total distance traveled is minimum.

Problem: Ordered Clustered Traveling Salesman Problem (OC-TSP)
Input: a complete undirected graph $G = (V, E)$, not necessarily disjoint $K+1$ clusters $C_i \subseteq V$ $(1 \leq k \leq K)$, where $\bigcup_{i=0}^{K+1} C_i = V$, $C_0 = \{s\}$ consists of a single node s called depot and a cost function $l : E \to \mathbb{R}^{\geq 0}$.
Feasible Solution: Hamiltonian path P s.t. points of cluster k are visited prior to points of cluster $k+1$, for $k = 1, 2, \ldots, K-1$.
Goal: $l(P) = \sum_{e \in E(P)} l(e)$ minimized.

Figure 10. Formalization of OC-TSP.

Observe that Anily et al. [45] provided $\frac{5}{3}$-approximation algorithm for OC-TSP when the clusters are disjoint which runs in $O(n^3)$ time. Furthermore, Guttmann-Beck et al. [46] showed the following results for OC-TSP:

- The starting and ending vertices in each cluster are given. A 1.9091-approximation algorithm is given [46] which is improved to 1.875 by Kawasaki and Takazawa [47].
- The two ending vertices in each cluster are given. We are free to choose any one as the starting vertex and the other one as the ending vertex. A 1.8-approximation algorithm is given [46] which is improved to 1.714 by Kawasaki and Takazawa [47].
- Only the starting vertex in each cluster is given. A 2.643-approximation algorithm is given [46] which is improved to 2.5 by Bao et al. [48].
- End vertices are not specified. A 2.75-approximation algorithm is given [46], which is improved to 2.67 by Kawasaki and Takazawa [47], improved to 2.167 by Bao and Liu [49], and improved to 1.9 by Bao et al. [48].

Minimum Cycle Cover Problem (MCCP). In this problem, whose formalization is shown in Figure 11, we are given a set of vertices in a metric space, a specified vertex s, and a distance bound B. The goal is to find a minimum cardinality set of tours starting at s that covers all vertices, such that each tour has length at most B. Xu et al. [50] showed that there is a 4-approximation algorithm for MCCP. In the same work, the authors also considered a different variant of MCCP called MCCP *without neighborhoods* where each IoT device can send its data to a UAV wirelessly, and the UAV can collect the data from the device as long as their Euclidean distance is no greater than a given wireless transmission range. In this case, they proved that there is a constant approximation algorithm. Nagarajan and Ravi [51], instead, studied MCCP when the cost of each vertex is zero and provided a $(1+\varepsilon, O(\log 1/\varepsilon))$-bicriteria approximation algorithm, i.e., for any ε, it obtains a solution violating the length bound B by a $1+\varepsilon$ factor while using at most $O(\log 1/\varepsilon)$ times the optimal number of vehicles. Nagarajan and Ravi [51] provided a 2-approximation algorithm on tree metrics. Finally, Mao et al. [52] considered a different variant of MCCP in which we are given a required edges $R \subseteq E$, and the goal is to find the minimum number of closed walks of length at most B which collectively traverse all the edges in R such that the number of closed walks used is a minimum. They provided a 5-approximation algorithm for this problem. For the case when $R = E$, they gave a 4-approximation algorithm.

Problem: Minimum Cycle Cover Problem (MCCP)
Input: an undirected graph $G = (V, E)$, an edge cost function $l : E \to \mathbb{R}^{\geq 0}$, a node cost function $h : V \to \mathbb{R}^{\geq 0}$ and a maximum data collection delay $B \in \mathbb{R}^{>0}$.
Feasible Solution: a set of K tours P_k s.t. $w(P_k) = \sum_{e \in E(P_k)} l(e) + \sum_{v \in V(P_k)} h(v) \leq B$ for any $k \in [K]$ and $\bigcup_{k=1}^{K} V(P_k) = V$.
Goal: K minimized.

Figure 11. Formalization of MCCP.

4. Implementation and Deployment

In this section, we discuss implementation, deployment, and testing efforts that have been addressed within the EMERGE initiative with respect to the algorithmic components of the system. Specifically, among all algorithms that have been considered and implemented in said system, we provide the details of the component that takes care of handling a specific problem of the class of problems discussed in Section 3, namely, the pickup and delivery multi-vehicle problem with time windows, whose solution is essential to coordinating vehicles to perform time-constrained operations requested by customers/users of the system.

In more detail, daily, on the basis of requests provided to the system by users, vehicles have to be provided with routes to follow in the geographical area (road network) where they are operating in order to visit locations, as indicated by users, within certain intervals of time to perform operations of pickup or delivery. These routes have to be computed by considering optimization criteria, such as minimizing the makespan for completing all operations. Each request of a user, more precisely, specifies (i) a given location to visit; (ii) a corresponding time window within the day when at least one vehicle of the fleet must reside in said location; (iii) the type of operation to perform (either pickup or delivery).

Pickup and delivery operations concern fixed, small-sized payloads, so the capacity of vehicles is not taken into account.

The above can be effectively modeled by the optimization problem on graphs, as shown in Figure 12. An example, in graphical form, of a generic input instance to the pickup and delivery problem with time windows taken from https://developers.google.com/optimization/routing/pickup_delivery (accessed on 25 March 2024) is shown in Figure 13. Observe that this problem, like the problems discussed in Section 3, is NP-hard since it includes instances of the TSP as a special case (when the fleet consists of a single vehicle and each time window is large enough to not represent a constraint). This implies that, for computing optimal solutions, no exact algorithm can be designed unless P = NP. Hence, an approximation algorithm must be considered.

Input: A set N of nodes, where node 0 represents the depot while nodes $1, 2, \ldots, n$ represent users' locations; a set A of arcs representing feasible routes between nodes; a set K of vehicles, where $k \in K$ represents a vehicle.
For each node $i \in N$, we have the following parameters: t_i: service time at a user's location i; (e_i, l_i): time window for user i, where e_i is the earliest arrival time, and l_i is the latest arrival time; and c_{ij}: cost (distance) of traversing arc (i, j).
Feasible Solution: Assignment to decision variables: (i) $\cup_{\forall (i,j,k) \in A \times K} \{x_{ij}^k\}$, where $x_{ij}^k = 1$ if vehicle k traverses arc (i, j) and 0 otherwise; (ii) $\cup_{\forall (i,k) \in N \times K} \{s_{ik}\}$, where s_{ik} is the arrival time at a user's location i for vehicle k, with the following constraints:

1. Each user's location must be visited exactly once, i.e., $\sum_{i \in N, i \neq j} x_{ij}^k = 1 \quad \forall j \in N, \forall k \in K$ and $\sum_{j \in N, j \neq i} x_{ij}^k = 1 \quad \forall i \in N, \forall k \in K$;
2. Flow conservation, i.e., $\sum_{i \in N, i \neq j} x_{ij}^k - \sum_{k \in N, k \neq j} x_{jik} = 0 \quad \forall j \in N, \forall k \in K$;
3. Time-window constraints, i.e., $e_i \leq s_{ik} \leq l_i \quad \forall i \in N, \forall k \in K$;
4. Arrival time computation, i.e., $s_{jk} \geq s_{ik} + t_i + M(1 - x_{ij}^k) \quad \forall i \in N, \forall j \in N, j \neq i, \forall k \in K$, where M is a sufficiently large constant;
5. Binary variable constraints, i.e., $x_{ij}^k \in \{0, 1\} \quad \forall i \in N, \forall j \in N, \forall k \in K$.

Goal: Minimization of the function $\sum_{k \in K} \sum_{i \in N, j \in N, j \neq i} c_{ij} x_{ij}^k$.

Figure 12. Formalization of pickup and delivery multi-vehicle problem with time windows.

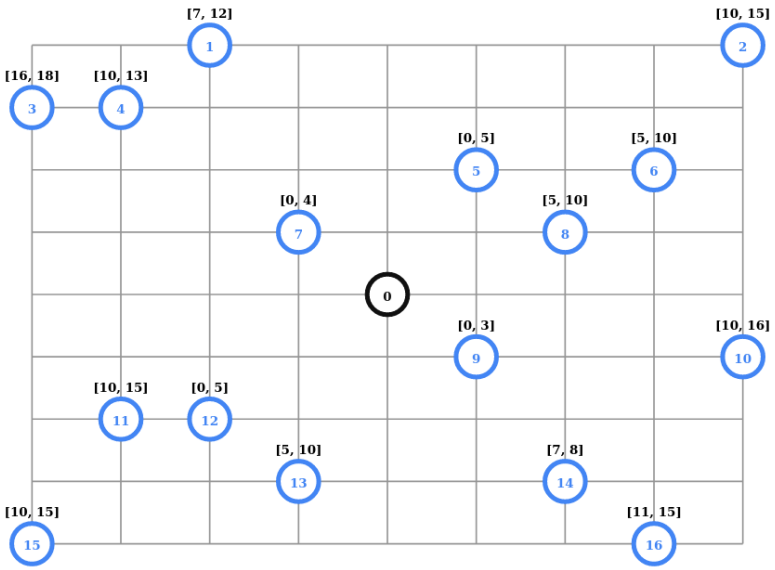

Figure 13. An example of a graphical representation of an input to the pickup and delivery problem with time windows, with 17 locations, identified by integer ids from 0 to 16, and their associated time windows, shown within square brackets.

Now, given the formalization, once an approximation algorithm is identified, the decision/optimization of the routes to assign to vehicles to satisfy requests is performed at the beginning of any working day by the EGC component of the system. After this initial phase, vehicles are loaded and the system is in a regular operational scenario, i.e., vehicles proceed to follow assigned routes and to visit specified locations. During such a scenario, the EMERGE system (the EGC component) starts collecting (i) data on the status of vehicles and the network via V2X (terrestrial and/or satellite) from the vehicles themselves; (ii) traffic data provided by third-party services, if available. The collected data are processed at EGC via suitable estimation/AI algorithms in order to detect/predict the possibility of congestion events occurring on road segments of routes assigned to vehicles (we refer the interested reader to [53–55] and references therein for some results in this area concerning the estimation of dynamical models with control purposes).

If EGC detects or predicts with sufficiently high probability that a change in the status of the network will affect portions of the network itself that do not overlap with routes being followed by vehicles or that overlap but with observed or predicted delays that are not large enough to induce violations in the time windows of locations to be visited, then vehicles are not "warned", and the system proceeds in a regular operational scenario: i.e., each vehicle keeps operating on the originally assigned route. Conversely, if observed or predicted delay/disrupting events happening in the network (or affecting vehicles) make it impossible to continue planned operations without violations of time windows (i.e., at least one scheduled visit cannot be performed correctly), then EGC implements the following attempts to reoptimize the plan for visits and corresponding routes:

1. EGC computes a set of alternative routes, one per vehicle, trying to preserve the satisfiability of the time-window constraints;
2. If the computation is successful, EGC sends updated routes to affected vehicles, bringing the system back to a condition of regular operation;
3. If the computation fails, i.e., there exists at least one location that cannot be visited by any of the vehicles in the prescribed time window, then EGC marks such location(s) as discarded, removes it/them from the set of users' requests, storing a trace of the events that caused such removal(s), and tries to compute a new, alternative plan of routes for remaining users' requests.

4.1. Input Data

To perform the above operations, EGC needs to store and manipulate information on the considered geographical zone where vehicles operate, i.e., on roads, traffic, and vehicle positions. In the current prototype, EGC exploits map data provided by the OpenStreetMap Foundation https://www.openstreetmap.org/ (accessed on 25 March 2024). Roads are represented by means of the so-called *ways*, which are ordered lists of *nodes*. Each node consists of a single point in space defined by its latitude, longitude, and an identifier. When two ways intersect at the same altitude (for example, at a road junction), the two ways must share a node. Hence, the result is a directed graph consisting of points connected by directed edges, representing the road network in an area. Additional sets of data can be added to the network built as above. In particular, such data include (i) vehicle positions (of both vehicles that belong to the fleet and those that do not), which can be added on the fly and, when added, are associated with the closest node in the network; (ii) the lengths of road segments between two nodes (representing, e.g., road junctions); (iii) the estimated travel times of road segments; (iv) the positions of users' locations; (v) distances between users' locations.

4.2. Software Modules and Prototyping

In order to address the above-mentioned decision/optimization problem, we considered both algorithms with guarantees and heuristics known in the literature. Eventually, we opted to exploit and customize the implementation of a solving algorithm provided as part of the Google OR-Tools suite (https://developers.google.com/optimization (accessed

on on 25 March 2024)), since this framework is known to perform very well on real-world inputs of the model that we have considered above.

In more detail, Google OR-Tools, developed by Google's Operations Research team, is a comprehensive suite of open-source software tools designed to tackle a broad spectrum of optimization challenges. Renowned for its versatility, the platform offers a rich set of libraries and algorithms that empower users to model and solve intricate combinatorial optimization and constraint satisfaction problems. One of the standout features of Google OR-Tools is its extensive range of applications, covering diverse domains such as linear programming, integer programming, vehicle routing, job scheduling, and constraint programming. This toolkit is known to be used by researchers, developers, and practitioners seeking effective solutions in fields ranging from logistics and transportation planning to resource allocation and scheduling. The framework is designed for accessibility and compatibility, supporting many popular programming languages (e.g., Python or C++). This support ensures that a wide community of users, both in academia and in industry, can leverage its capabilities. Additionally, the toolkit is known for its scalability, enabling the efficient handling of large-scale problem instances, a crucial aspect in addressing real-world complexities.

In what follows, we describe how the solver has been customized and integrated into the EMERGE system and provide details on how services provided by the EMERGE system are currently implemented in the advanced prototype under test. The implementation adheres to the Edge–Cloud Computing paradigm, and computational efforts are divided between modules: EGC (with a cloud component, a set of elastic servers), MEC (multi-access edge computing), and OBUs (on-board units of vehicles). Specifically, in EGC, we developed a software module named CARF (Critical Avoidance Route Finder), which is responsible for the computation of routes for the vehicles on the basis of an input road map of an area, users' requests (with associated time windows), and the structure of the fleet of vehicles. This computation is performed during the *vehicle task planning* phase. Since the level of detail of available maps of geographical areas can be unnecessarily high for the purpose of our computation, the road map is subject to preprocessing to be converted into a complete directed graph Gin which the nodes are only depot and users' locations. Directed edges connecting each pair of nodes are associated with the distance (or average travel time) between two users' locations, which are computed by an implementation of Dijkstra's algorithm by another component, namely, the SUMO toolkit, which runs on MEC.

SUMO is a software tool for urban mobility that can be used both for simulation purposes and as a toolbox to compute the parameters of maps treated in intelligent transport systems like EMERGE. In more detail, SUMO (Simulation of Urban Mobility) is open-source traffic simulation software developed for the modeling and analysis of urban mobility and transportation systems. It provides a comprehensive platform for simulating various aspects of traffic flow within urban environments and, thus, is considered a valuable tool for researchers and practitioners in the field of transportation engineering to test algorithms in a simulated realistic environment. In particular, the main purpose for which researchers/practitioners use SUMO is to investigate and analyze traffic flow dynamics, test new algorithms, and develop intelligent transportation systems. Specifically, the tool provides a virtual environment for experimenting with different strategies before implementing them in the real world. SUMO enables users to simulate the movement of vehicles, pedestrians, and other entities within a virtual urban environment. The simulation takes into account factors such as vehicle speed, acceleration, deceleration, and interactions with other road users. Users can model and define the road network, including intersections, traffic lights, and other infrastructure elements. This allows for the accurate representation of complex urban traffic scenarios.

The tool supports the modeling of various traffic control systems, including traffic signals, priority rules, and right-of-way policies. This makes it suitable for studying the impact of different traffic management strategies on the overall system performance. Moreover, it allows users to create and define specific traffic scenarios, considering factors like traffic density, road types, and user behaviors. This flexibility is essential for conducting

experiments and evaluating the performance of transportation systems under different conditions. The software supports various file formats, making it compatible and interoperable with other simulation tools and data sources. This makes SUMO suited for integration into larger transportation modeling frameworks.

After preprocessing, the CARF instantiates the model of the optimization problem on graphs, as shown in Figure 12, according to the input data (see Figure 13), and then exploits a routine offered by OR-Tools to find a solution to the obtained instance of the problem. The output is a sequence of users' locations for each vehicle. To transform each sequence into a route, a subroutine is called on a dataset representing the road map to find the shortest path for each pair of consecutive users' locations in the sequence. The concatenation of all the shortest paths for a given vehicle forms the route that the vehicle has to follow. The subroutine to find the shortest paths implements a customization of the well-known Dijkstra's algorithm. Finally, the routes are transmitted, properly encoded, to the OBUs of the vehicles, where they are displayed by means of an HCI (Human–Computer Interface), along with the positions of the users and of other vehicles in the fleet.

Clearly, the CARF module (in the cloud), the SUMO tool (on MEC), and the HCIs (in the vehicles) have to exchange data. At the service level, this coordination is implemented by a *broker system* consisting of modules running on all the computers in the EMERGE system. The broker system provides channels to which other running software modules can subscribe to push or read messages. Then, there are channels to share the positions of the vehicles in the fleet, to manage customer sequences, to send routes, and so on.

During the regular operational scenario, as highlighted at the beginning of this section, the EMERGE system collects data via V2X (terrestrial and/or satellite links) from the vehicles and sends it via the broker system to all the affected parts. In particular, EGC stores data about the users' locations visited by vehicles in the fleet. If available, the system aggregates data about the traffic provided by third parties. Collected data are processed by a module that executes estimation/AI algorithms to detect congestion events on road segments of the assigned routes. The details of these algorithms are omitted since they are out of the scope of this paper; we refer the reader to [53–55] for relevant examples. When such an event is detected, an alert message is sent to EGC via the broker system. EGC, as described above, evaluates whether a reoptimization of the routes is necessary.

Whenever the routes have to be updated due to modifications of the status of the network, EGC reads the position of each vehicle in the fleet and re-arranges a new instance of the pickup and delivery with time windows model by incorporating the occurred changes into the initial model. In particular, in this case, a new complete graph is built with as many vertices as current vehicle positions, and without considering a depot vertex. Other vertices in the graph are the locations of users not yet visited. Distances between vehicles and users' locations are computed from scratch with the Dijkstra's algorithm implementation of the SUMO module. New routes are then distributed to the vehicles via the broker system.

4.3. System Validation

All mentioned software components of the system (CARF, MEC, EGC, OBUs) have been validated separately. At the time of writing this article, the validation of the whole prototype of the EMERGE system is underway. As the number of smart vehicles available to the system is still limited to a few units, a simulation module—SIM—has been introduced to simulate the operations performed by some vehicles in the fleet, to report in the simulation the position of real vehicles in the fleet, and to simulate other vehicles that are not part of the EMERGE system but influence the traffic on the roads of the areas. The SIM module exchanges data with all other modules via the broker in order to simulate the behavior of all vehicles in the fleet in following their routes to visit customers. The simulation was executed by running an instance of the SUMO tool, directly controlled by the SIM. When a reoptimization of the routes is necessary, the SIM receives all the new routes for the simulated vehicles and, consequently, runs the SUMO simulation.

Under this setting, several experiments were conducted in the metropolitan area of the city of L'Aquila (Italy). The performance of the system was measured for a fleet of vehicles of varying size (between 3 and 20 units) with respect to several parameters (e.g., time for executing necessary routines and data collection, quality of computed routes with respect to optimization criteria, number of users' locations that were correctly visited in various scenarios). Preliminary results show that the execution time necessary for the CARF during the vehicle task planning phase varies from a few milliseconds to less than 3.3 s when the number of users' locations to schedule ranges in the interval from 15 to 200 (see Figure 14 for data for a fleet of 15 vehicles). On top of the above, computed routes were observed to be satisfactory in terms of quality, with all prescribed user locations being successfully visited within the required time windows and with a reasonably low total makespan in all tested input instances. This is considered acceptable for the requirements of the EMERGE initiative. Similar times and quality of routes were observed in cases of reoptimization. Moreover, the execution times and the quality of routes were only marginally influenced by the number of vehicles forming the fleet until the fleet remained limited to a few tens of units. However, should the number of vehicles or users' locations increase, we expect that the quality of computed solutions might worsen, particularly when the routine offered by OR-Tools is forced to return the best solution that can be found within a limited interval of time. This is not surprising since the problem is NP-hard. Nonetheless, a more rigorous evaluation of the performance of the system is necessary and is currently being performed to consolidate preliminary empirical observations.

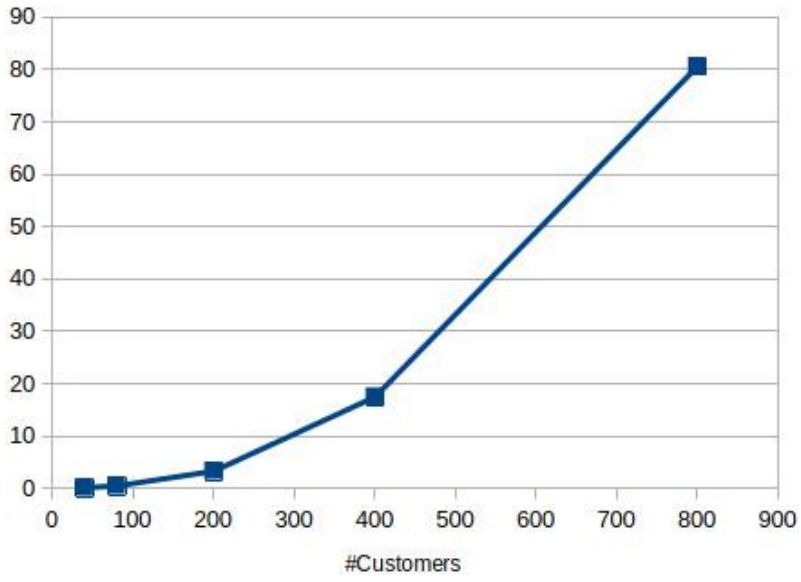

Figure 14. The execution time in seconds to compute a solution to the pickup and delivery multi-vehicle problem with time windows for a fleet of 15 vehicles. This figure and the one following it refer to sets of 40, 80, 200, 400, and 800 customers.

Observe that, to compute the complete directed graph G with distances between customers' locations (as described in Section 4.2), an off-line implementation of Dijkstra's algorithm is used. In Figure 15, we show the execution times taken to compute the shortest paths for all pairs of customers' locations. Execution times range from a few seconds to less than 1.5 min when the number of users' locations to schedule ranges in the interval from 15 to 200. For a larger number of customers, execution times increase significantly, reaching values of around half an hour for 800 customers. However, this does not represent a bottleneck since this kind of computation is executed only once; hence, it does not affect the normal operating scenario of the system. Moreover, more refined techniques for solving all pairs of shortest-path problems are available in the literature and will be considered in future releases of the system.

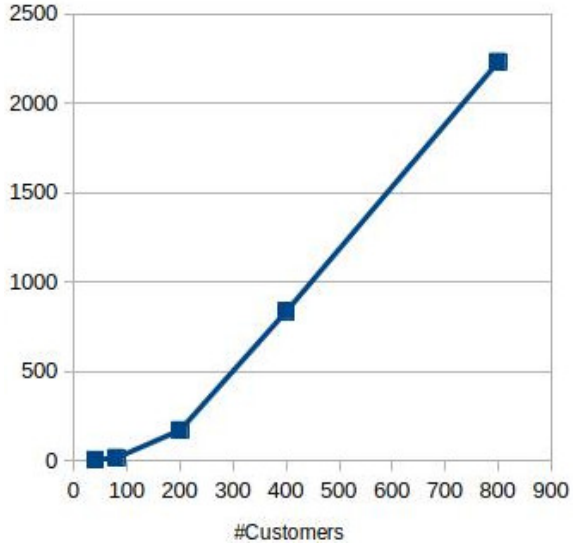

Figure 15. The execution time in seconds (on the *y* axis) to compute the shortest paths between all pairs of customers with Dijkstra's algorithm as a function of the number #Customers of customers (on the *x* axis).

5. Conclusions and Future Work

In this paper, we have presented a study on computational challenges and implementation issues arising in real-world systems of semi-autonomous vehicles. We have described the main features of a real-world information system employing semi-autonomous connected vehicles that was designed and deployed and is currently under test in the city of L'Aquila (Italy) within the broader EMERGE initiative, a smart mobility and communications project to develop technologies for autonomous and assisted driving systems. We have given a thorough overview of computational problems that often emerge in the considered application domain and a systematic survey of the most relevant known algorithmic results for pickup and delivery problems. We have showcased implementation, deployment, and testing efforts related the algorithmic components of the mentioned real-world system that takes care of computing solutions to instances of the pickup and delivery multi-vehicle problem with time windows that arise in such system. Concerning possible future investigation following this work, perhaps the most interesting one is that dedicated to adapting the system to handle entirely dynamic scenarios and to exploit cooperation among vehicles to perform more refined strategies of reoptimization (see, e.g., policies for task re-allocation that have been studied for systems of agents [2]).

Author Contributions: Conceptualization, G.F. and E.V.; Methodology, E.D. and G.D.S.; Software, G.F. and E.V.; Validation, G.F. and E.V.; Formal analysis, E.D.; Investigation, M.D., E.D. and G.D.S.; Data curation, G.F. and E.V.; Writing—original draft, E.D.; Supervision, M.D. and G.D.S.; Project administration, G.D.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partially supported by: (i) project EMERGE: innovation agreement between the Ministry of Economical Development, Abruzzo Region, Radiolabs, Elital, Leonardo, Telespazio, the University of L'Aquila, and the Centre of EXcellence EX-Emerge, funded by the Italian Government under CIPE n. 70/2017 (7 August 2017); (ii) Italian National Group for Scientific Computation-Istituto Nazionale di Alta Matematica (GNCS-INdAM).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Liu, Y.; Peng, C. A Close Look at 5G in the Wild: Unrealized Potentials and Implications. In Proceedings of the IEEE INFOCOM 2023—IEEE Conference on Computer Communications, New York, NY, USA, 17–20 May 2023; pp. 1–10. [CrossRef]
2. D'Emidio, M.; Khan, I. Collision-free allocation of temporally constrained tasks in multi-robot systems. *Robotics Auton. Syst.* **2019**, *119*, 151–172. [CrossRef] [CrossRef]
3. Cicerone, S.; Di Stefano, G.; Navarra, A. Asynchronous Arbitrary Pattern Formation: The effects of a rigorous approach. *Distributed Comput.* **2019**, *32*, 91–132. [CrossRef] [CrossRef]
4. Park, P.; Di Marco, P.; Jung, M.; Santucci, F.; Sung, T.K. Multidirectional Differential RSS Technique for Indoor Vehicle Navigation. *IEEE Internet Things J.* **2023**, *10*, 241–253. [CrossRef] [CrossRef]
5. D'Angelo, G.; D'Emidio, M.; Das, S.; Navarra, A.; Prencipe, G. Leader Election and Compaction for Asynchronous Silent Programmable Matter. In Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '20), Auckland, New Zealand, 9–13 May 2020; Seghrouchni, A.E.F., Sukthankar, G., An, B., Yorke-Smith, N., Eds.; International Foundation for Autonomous Agents and Multiagent Systems: Liverpool, UK, 2020; pp. 276–284. [CrossRef]
6. D'Emidio, M. Faster Algorithms for Mining Shortest-Path Distances from Massive Time-Evolving Graphs. *Algorithms* **2020**, *13*, 191. [CrossRef] [CrossRef]
7. Abraham, I.; Delling, D.; Fiat, A.; Goldberg, A.V.; Werneck, R.F. Highway Dimension and Provably Efficient Shortest Path Algorithms. *J. ACM* **2016**, *63*, 41:1–41:26. [CrossRef] [CrossRef]
8. Kleinberg, J.M.; Tardos, É. *Algorithm Design*; Addison-Wesley: Boston, MA, USA, 2006.
9. D'Emidio, M.; Frigioni, D.; Navarra, A. Explore and repair graphs with black holes using mobile entities. *Theor. Comput. Sci.* **2015**, *605*, 129–145. [CrossRef] [CrossRef]
10. D'Ascenzo, A.; D'Emidio, M.; Flammini, M.; Monaco, G. Digraph k-Coloring Games: From Theory to Practice. In Proceedings of the 20th International Symposium on Experimental Algorithms (SEA 2022), Heidelberg, Germany, 25–27 July 2022; Schloss Dagstuhl—Leibniz-Zentrum für Informatik: Wadern, Germany, 2022; Volume 233, pp. 20:1–20:18.
11. Even, S. *Graph Algorithms*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2011.
12. D'Angelo, G.; Delfaraz, E.; Gilbert, H. Budgeted Out-Tree Maximization with Submodular Prizes. In Proceedings of the 33rd International Symposium on Algorithms and Computation (ISAAC 2022), Seoul, Republic of Korea, 19–21 December 2022; Bae, S.W., Park, H., Eds.; Schloss Dagstuhl—Leibniz-Zentrum für Informatik: Wadern, Germany, 2022; Volume 248, pp. 9:1–9:19.
13. D'Angelo, G.; D'Emidio, M.; Frigioni, D. Distance Queries in Large-Scale Fully Dynamic Complex Networks. In Proceedings of the 27th International Workshop on Combinatorial Algorithms (IWOCA 2016), Helsinki, Finland, 17–19 August 2016; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9843, pp. 109–121.
14. Di Sciullo, G.; Zitella, L.; Cinque, E.; Santucci, F.; Pratesi, M.; Valentini, F. Experimental Validation of C-V2X Mode 4 Sidelink PC5 Interface for Vehicular Communications. In Proceedings of the 2022 61st FITCE International Congress Future Telecommunications: Infrastructure and Sustainability (FITCE), Rome, Italy, 29–30 September 2022; pp. 1–6. [CrossRef]
15. Cinque, E.; Valentini, F.; Persia, A.; Chiocchio, S.; Santucci, F.; Pratesi, M. V2X Communication Technologies and Service Requirements for Connected and Autonomous Driving. In Proceedings of the 2020 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE), Turin, Italy, 18–20 November 2020; pp. 1–6. [CrossRef]
16. Saller, S.; Koehler, J.; Karrenbauer, A. A Systematic Review of Approximability Results for Traveling Salesman Problems leveraging the TSP-T3CO Definition Scheme. *arXiv* **2023**, arXiv:2311.00604.
17. Blum, A.; Chawla, S.; Karger, D.R.; Lane, T.; Meyerson, A.; Minkoff, M. Approximation Algorithms for Orienteering and Discounted-Reward TSP. *SIAM J. Comput.* **2007**, *37*, 653–670. [CrossRef]
18. Bansal, N.; Blum, A.; Chawla, S.; Meyerson, A. Approximation algorithms for deadline-TSP and vehicle routing with time-windows. In Proceedings of the Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, 13–16 June 2004; Babai, L., Ed.; ACM: New York, NY, USA, 2004; pp. 166–174.
19. Chekuri, C.; Korula, N.; Pál, M. Improved algorithms for orienteering and related problems. *ACM Trans. Algorithms* **2012**, *8*, 23:1–23:27. [CrossRef]
20. Friggstad, Z.; Swamy, C. Compact, Provably-Good LPs for Orienteering and Regret-Bounded Vehicle Routing. In Proceedings of the Integer Programming and Combinatorial Optimization—19th International Conference, IPCO 2017, Waterloo, ON, Canada, 26–28 June 2017; Lecture Notes in Computer Science; Eisenbrand, F., Könemann, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2017; Volume 10328, pp. 199–211.
21. Paul, A.; Freund, D.; Ferber, A.M.; Shmoys, D.B.; Williamson, D.P. Budgeted Prize-Collecting Traveling Salesman and Minimum Spanning Tree Problems. *Math. Oper. Res.* **2020**, *45*, 576–590. [CrossRef]
22. Chen, K.; Har-Peled, S. The Euclidean Orienteering Problem Revisited. *SIAM J. Comput.* **2008**, *38*, 385–397. [CrossRef]
23. Frederickson, G.N.; Wittman, B. Approximation Algorithms for the Traveling Repairman and Speeding Deliveryman Problems. *Algorithmica* **2012**, *62*, 1198–1221. [CrossRef]
24. Gao, J.; Jia, S.; Mitchell, J.S.B.; Zhao, L. Approximation Algorithms for Time-Window TSP and Prize Collecting TSP Problems. In Proceedings of the Algorithmic Foundations of Robotics XII, Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics, WAFR 2016, San Francisco, CA, USA, 18–20 December 2016; Springer Proceedings in Advanced Robotics; Goldberg, K., Abbeel, P., Bekris, K.E., Miller, L., Eds.; Springer: Berlin/Heidelberg, Germany, 2016; Volume 13, pp. 560–575.

25. Bar-Yehuda, R.; Even, G.; Shahar, S. On approximating a geometric prize-collecting traveling salesman problem with time windows. *J. Algorithms* **2005**, *55*, 76–92. [CrossRef]
26. Chekuri, C.; Kumar, A. Maximum coverage problem with group budget constraints and applications. In Proceedings of the International Workshop on Randomization and Approximation Techniques in Computer Science, Cambridge, MA, USA, 22–24 August 2004; Springer, Berlin/Heidelberg, Germany, 2004; pp. 72–83.
27. Pérez-Pérez, S.L.; Rivero, L.E.U.; López-Bracho, R.; Martínez, F.J.Z. A fast 4-approximation algorithm for the traveling repairman problem on a line. In Proceedings of the 11th International Conference on Electrical Engineering, Computing Science and Automatic Control, CCE 2014, Campeche, Mexico, 29 September–3 October 2014; pp. 1–4.
28. Miguel-Pilar, Y.; Morales-Luna, G.; Troncoso, F.D.S.; Martínez, F.J.Z. An ILP approach for the traveling repairman problem with unit time windows. In Proceedings of the 12th International Conference on Electrical Engineering, Computing Science and Automatic Control, CCE 2015, Mexico City, Mexico, 28–30 October 2015; pp. 1–5.
29. Garg, N.; Khanna, S.; Kumar, A. Hardness of Approximation for Orienteering with Multiple Time Windows. In Proceedings of the Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, 10–13 January 2021; Marx, D., Ed.; SIAM: Philadelphia, PA, USA, 2021; pp. 2977–2990.
30. Farbstein, B.; Levin, A. Deadline TSP. *Theor. Comput. Sci.* **2019**, *771*, 83–92. [CrossRef]
31. Friggstad, Z.; Swamy, C. Constant-factor Approximation to Deadline TSP and Related Problems in (almost) Quasi-polytime. In Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP 2021), Wadern, Germany, 12–16 July 2021.
32. Gupta, A.; Krishnaswamy, R.; Nagarajan, V.; Ravi, R. Running Errands in Time: Approximation Algorithms for Stochastic Orienteering. *Math. Oper. Res.* **2015**, *40*, 56–79. [CrossRef]
33. Bock, A.; Sanità, L. The Capacitated Orienteering Problem. *Discret. Appl. Math.* **2015**, *195*, 31–42. [CrossRef]
34. Xu, W.; Liang, W.; Xu, Z.; Peng, J.; Peng, D.; Liu, T.; Jia, X.; Das, S.K. Approximation Algorithms for the Generalized Team Orienteering Problem and its Applications. *IEEE/ACM Trans. Netw.* **2021**, *29*, 176–189. [CrossRef]
35. Friggstad, Z.; Gollapudi, S.; Kollias, K.; Sarlós, T.; Swamy, C.; Tomkins, A. Orienteering Algorithms for Generating Travel Itineraries. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, 5–9 February 2018; Chang, Y., Zhai, C., Liu, Y., Maarek, Y., Eds.; ACM: New York, NY, USA, 2018; pp. 180–188.
36. Xu, W.; Wang, C.; Xie, H.; Liang, W.; Dai, H.; Xu, Z.; Wang, Z.; Guo, B.; Das, S.K. Reward Maximization for Disaster Zone Monitoring With Heterogeneous UAVs. *IEEE/ACM Trans. Netw.* **2023**, *32*, 890–903. [CrossRef]
37. Archetti, C.; Bianchessi, N.; Speranza, M.G.; Hertz, A. The split delivery capacitated team orienteering problem. *Networks* **2014**, *63*, 16–33. [CrossRef]
38. Nagarajan, V.; Ravi, R. The Directed Orienteering Problem. *Algorithmica* **2011**, *60*, 1017–1030. [CrossRef]
39. Böckenhauer, H.; Komm, D. Reoptimization of the metric deadline TSP. *J. Discrete Algorithms* **2010**, *8*, 87–100. [CrossRef]
40. Böckenhauer, H.J.; Kneis, J.; Kupke, J. Approximation hardness of deadline-TSP reoptimization. *Theor. Comput. Sci.* **2009**, *410*, 2241–2249. [CrossRef]
41. Böckenhauer, H.; Forlizzi, L.; Hromkovic, J.; Kneis, J.; Kupke, J.; Proietti, G.; Widmayer, P. Reusing Optimal TSP Solutions for Locally Modified Input Instances. In Proceedings of the Fourth IFIP International Conference on Theoretical Computer Science (TCS 2006), IFIP 19th World Computer Congress, TC-1 Foundations of Computer Science, Santiago, Chile, 23–24 August 2006; Navarro, G., Bertossi, L.E., Kohayakawa, Y., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; Volume 209, pp. 251–270.
42. Khachay, M.Y.; Ogorodnikov, Y. Approximation Scheme for the Capacitated Vehicle Routing Problem with Time Windows and Non-uniform Demand. In Proceedings of the Mathematical Optimization Theory and Operations Research—18th International Conference, MOTOR 2019, Ekaterinburg, Russia, 8–12 July 2019; Lecture Notes in Computer Science; Khachay, M.Y., Kochetov, Y., Pardalos, P.M., Eds.; Springer: Berlin/Heidelberg, Germany, 2019; Volume 11548, pp. 309–327.
43. Khachay, M.; Ogorodnikov, Y. Improved polynomial time approximation scheme for capacitated vehicle routing problem with time windows. In Proceedings of the International Conference on Optimization and Applications, Mohammedia, Morocco, 26–27 April 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 155–169.
44. Khachay, M.; Ogorodnikov, Y. Towards an efficient approximability for the Euclidean Capacitated Vehicle Routing Problem with Time Windows and multiple depots. *IFAC-PapersOnLine* **2019**, *52*, 2644–2649. [CrossRef]
45. Anily, S.; Bramel, J.; Hertz, A. A 5/3-approximation algorithm for the clustered traveling salesman tour and path problems. *Oper. Res. Lett.* **1999**, *24*, 29–35. [CrossRef]
46. Guttmann-Beck, N.; Knaan, E.; Stern, M. Approximation Algorithms for Not Necessarily Disjoint Clustered TSP. *J. Graph Algorithms Appl.* **2018**, *22*, 555–575. [CrossRef]
47. Kawasaki, M.; Takazawa, K. Improving Approximation Ratios for the Clustered Traveling Salesman Problem. *J. Oper. Res. Soc. Jpn.* **2020**, *63*, 60–70. [CrossRef]
48. Bao, X.; Liu, Z.; Yu, W.; Li, G. A note on approximation algorithms of the clustered traveling salesman problem. *Inf. Process. Lett.* **2017**, *127*, 54–57. [CrossRef]
49. Bao, X.; Liu, Z. An improved approximation algorithm for the clustered traveling salesman problem. *Inf. Process. Lett.* **2012**, *112*, 908–910. [CrossRef]
50. Xu, W.; Xiao, T.; Zhang, J.; Liang, W.; Xu, Z.; Liu, X.; Jia, X.; Das, S.K. Minimizing the Deployment Cost of UAVs for Delay-Sensitive Data Collection in IoT Networks. *IEEE/ACM Trans. Netw.* **2022**, *30*, 812–825. [CrossRef]

51. Nagarajan, V.; Ravi, R. Approximation algorithms for distance constrained vehicle routing problems. *Networks* **2012**, *59*, 209–214. [CrossRef]
52. Mao, Y.; Yu, W.; Liu, Z.; Xiong, J. Approximation algorithms for some Minimum Postmen Cover Problems. *Discret. Appl. Math.* **2022**, *319*, 382–393. [CrossRef]
53. De Iuliis, V.; Domenico Di Girolamo, G.; Smarra, F.; D'Innocenzo, A. A Comparison of Classical Identification and Learning-Based Techniques for Cyber-Physical Systems. In Proceedings of the 29th Mediterranean Conference on Control and Automation (MED), Puglia, Italy, 22–25 June 2021; pp. 179–185. [CrossRef]
54. De Iuliis, V.; Smarra, F.; Manes, C.; D'Innocenzo, A. Stability analysis of switched ARX models and application to learning with guarantees. *Nonlinear Anal. Hybrid Syst.* **2022**, *46*, 101250. [CrossRef] [CrossRef]
55. Smarra, F.; Di Girolamo, G.D.; De Iuliis, V.; Jain, A.; Mangharam, R.; D'Innocenzo, A. Data-driven switching modeling for MPC using Regression Trees and Random Forests. *Nonlinear Anal. Hybrid Syst.* **2020**, *36*, 100882. [CrossRef] [CrossRef]