Ride-Sharing in Medical Transportations: Dealing with Temporal Requirements

Giovanni Alberto Beltrame[1,†], Carlo Combi[2,†], Alessandro Farinelli[2,†], Roberto Posenato[2,†] and Giuseppe Pozzi[3,*,†]

[1]Former student, Università di Verona, Strada Le Grazie, I-37134, Verona, Italy
[2]Dipartimento di Informatica, Università di Verona, Strada Le Grazie, I-37134, Verona, Italy
[3]DEIB, Politecnico di Milano, P.za L. da Vinci 32, I-20133, Milano, Italy

Abstract

The ride-sharing problem aims at optimizing the path from one starting point to one destination point. The problem can be enriched by intermediate stops, spatio-temporal constraints, and external constraints (e.g. traffic congestion), adding uncertainty and increasing the overall complexity.

Spatio-temporal networks can properly describe the problem by graphs, helping to identify the optimal or sub-optimal solution. We face here the specific issue, where a driver picks up several patients from their respective pick-up locations and drops them off at one care center. Ride-sharing of patients has specific requirements due to the particular health state of every patient. Indeed, every patient has his/her own constraints, which could be related to the maximum sustainable duration of the trip, according to the patient's conditions, the maximum waiting time, and the time when the visit or treatment is scheduled.

In our approach, we first consider the spatial facets, and then we superimpose the temporal facets, to recommend the best paths and schedules, allowing some kind of temporal uncertainty in the specification of different possible constraints.

Keywords

Spatio-temporal networks, Uncertainty, Graphs, Ride-sharing, Patient transportation

1. Introduction

Ride-sharing [1] is a mode of transportation in which individual travelers share a vehicle and, eventually, its costs. Typically, passengers have the same unique destination. Passengers may leave from the same starting point or may be collected along the way of the first passenger to the shared destination. Ride-sharing combines the flexibility and speed of private cars with the reduced cost of fixed-line systems. In static ride-sharing, passenger arrangements are pre-computed and cannot be modified during the service. In dynamic ride-sharing, automatic ride-matching between participants can occur on very short notice or even en route.

The problem of ride-sharing aims at optimizing the path of the vehicle. Optimization is helpful under several terms: travel costs, travel time, and environmental pollution are just a few of them. The problem can be enriched by several intermediate stops, spatiotemporal constraints, and external constraints (e.g. traffic congestion), adding uncertainty and increasing the overall complexity.

Ride-sharing in healthcare has been considered as a way of increasing the number of people possibly accessing medical care, as it is less expensive than other services and available also in places where public transportation is missing [2, 3, 4]. Besides several policy- and healthcare-related issues, ride-sharing in healthcare has some specific features, which need to be considered when designing software systems supporting ride-sharing activities for patients. Indeed, not considering the specific requirements of ride-sharing in the context of healthcare domains may produce low-quality services, which possibly prevent delivering the right care to the weaker patients' categories because of transportation barriers. Among the specific requirements that need to be addressed when planning ride-sharing for patients, we consider here:

- the maximum allowed duration of the trip for specific patients, who cannot afford too long trips;
- the strict ranges of allowed waiting times, as patients are not able to face too long waiting times (or to rush for too short deadlines);
- the flexibility in reaching the final destination, avoiding both a rush and a too-long waiting time at the healthcare center, a not feasible situation especially for patients and in this pandemic context.

In the following, we shall consider the general issue of medical transportation, where a driver picks up some patients from their respective starting points (e.g., homes), and drops them off at the same care center. The approach we propose in this paper considers the integrated application of both temporal and spatial reasoning, focusing on the management of temporal uncertainty, taking into account the specific requirements of such kind of transportation. Temporal issues refer to the preferred arrival time every passenger may have. Spatial issues refer to the path of the shared vehicle. External constraints refer to traffic conditions, which may also occur dynamically, i.e. during the journey, and not just before the journey starts.

The paper is structured as follows: Section 2 describes related work from the literature on both temporal and spatial topics, and the background methodological concepts; Section 3 describes the application domain and how we model the problem; Section 4 describes a proof-of-concept prototype we implement; Section 5 highlights the achieved conclusions and sketches out some future research directions.

Published in the Workshop Proceedings of the EDBT/ICDT 2024 Joint Conference (March 25-March 28, 2024), Pæstum, Italy

*Corresponding author.
†These authors contributed equally.
beltramegiovannialberto@gmail.com (G. A. Beltrame); carlo.combi@univr.it (C. Combi); alessandro.farinelli@univr.it (A. Farinelli); roberto.posenato@univr.it (R. Posenato); giuseppe.pozzi@polimi.it (G. Pozzi)
https://www.deib.polimi.it/pozzi (G. Pozzi)
— (G. A. Beltrame); 0000-0002-4837-4701 (C. Combi); 0000-0002-2592-5814 (A. Farinelli); 0000-0003-0944-0419 (R. Posenato); 0000-0002-2828-862X (G. Pozzi)

2. Background and Related Work

This section describes the background and the related work on temporal networks and on the ride-sharing problem, and how we select the proper methodology to cope with the selected application domain. Ride-sharing problems are commonly formalized by graphs, which are then analyzed by temporal networks.

2.1. Background: STNUs

A Simple Temporal Problem (STP) is a problem involving quantitative time constraints [5] between pairs of time points. A Simple Temporal Network (STN) is a framework for planning and scheduling applications of a STP, which is represented through a set of nodes, i.e./ time points, and weighted edges between nodes, representing quantitative temporal constraints: the formalization is adopted to check consistency in many constraint-based planning systems [6].

STNU refers to STN with uncertainty, where the occurrences of some time points, named *contingent*, are within specified time ranges, but beyond the control of the planning agent [7]. An STNU is *controllable* if a solution satisfying every constraint in the network can be found. One of the strategies for STNU is the RTED (Real Time Execution Decision) strategy [7, 8], to manage contingent time points (events) which occur at run-time, and cannot be controlled (scheduled) by the agent, but are simply observed. An STNU is dynamically controllable iff (i.e., "if and only if") an execution strategy based on RTED exists. Intuitively, according to RTED, the agent, responsible for the network execution, can only observe the occurrence of contingent time points but is capable to react to such occurrences, by deciding when to execute the other time points, which are under its control.

The RTED strategy is based on a table known as all-pair-shortest-semi-reducible paths (APSSRP), which for every couple of nodes in a weighted graph returns the measure of the shortest path (or weighted edge) connecting those two nodes [7], that represent the strongest constraints that any reliable execution strategy must satisfy. If no semi-reducible negative loop is in the APSSRP, then the RTED strategy can dynamically assign values to the network time points, satisfying all the given temporal constraints.

Major related work on STNU refers to the algorithms to check the consistency of the network and its dynamic controllability (i.e. there exists a dynamic strategy for guaranteeing all the constraints, no matter when contingent time points occur), as well as supporting the dynamic execution of the network. Morris et al [9] present a polynomial time algorithm to check the dynamic controllability of a STNU: the complexity of the algorithm is $O(N^5)$, being N the number of nodes in the network. The algorithm is based on constraint propagation, where the edges, which represent time constraints between two nodes, are expanded to explicitly state all the constraints that affect each node of the network. However, the authors assumed that non-shortest labeled edges in an STNU could be disregarded – which turns out to be far from trivial to prove [7, 10]. Morris [11] presents a faster $O(N^4)$ algorithm, which relies on a new approach to analyze some graphical properties of the simple temporal network with no uncertainty. The same author [12] presents an even faster version of the algorithm: the algorithm has a time complexity of $O(N^3)$. All the algorithms for dynamic controllability checking, currently proposed by other authors, have the same complexity as in [12]. For sake of simplicity, as these last algorithms are quite complex and full of technicalities, in this paper, we consider as the fundamental starting point the $O(N^5)$ dynamic controllability checking algorithm of [9] for STNUs.

2.2. Related Work

The issue of patient transportation is of great relevance: it is estimated that 5.8 million people in the US during 2017 delayed non-emergency medical care due to lack of transportation [13]: the CoViD-19 pandemic hardened the problem. A taxonomy of innovative health care mobility services is reported in [14].

The problem of ride-sharing of patients falls within the wider topic of patient transportation. Many issues have been faced in this direction: intra-hospital patient transportation; optimizing the use of Advance Life Support (ALS) services (managing patients requiring high level of medical monitoring and emergency care) and Basic Life Support (BLS) (managing patients requiring non-emergency medical transportation); evaluating ride-sharing services. Without being complete, in the following we shall briefly discuss some technical contributions, providing an overall picture of the context, within which we propose our original contribution.

In [15], the authors propose a generalization of the dial-a-ride problem, modeling some real-life requirements for patient transportation. A multi-directional local search algorithm is developed to solve this problem, taking into account the fundamental tradeoff between operational efficiency and service quality, by considering specific constraints for patients and drivers. Moreover, the authors propose an original scheduling procedure, minimizing the total user ride time.

As already mentioned, ambulance providers support both ALS and BLS ambulances. In [16], the authors propose a model that determines the routes for BLS ambulances while maximizing the remaining coverage by ALS ambulances. Indeed, while BLS ambulances deal with non-urgent transportations, ALS have to deal with urgent ones. However, BLS ambulances often do not suffice for the required transportation, and the use of ALS for not urgent transportation is deployed, if any critical event occurs. Some specific features of the faced issue are that only one patient can be transported at a time, and the requests are known dynamically, especially for urgent transportation.

Fulgenzi et al. in [17] propose a simulation-based system to improve the quality and efficiency of (intra) hospital transportation system, according to the patient's condition, the human and technical resources, and the time requirements.

In [18], the authors consider patient transportation in the Republic of Korea. They propose a web-based software system able to optimize patient transportation, by considering patients' pathologies, distances from the specialized hospitals, required times, travel costs, and so on. Routes and hospitals are identified, also through the use of crawled data, suitably collected and analyzed in big-data, distributed context, to support decision-makers.

3. Problem Definition and Modelling

This section describes the application domain of ride-sharing and the modeling technique we deployed. We start with spatial modeling, define the ride-sharing graph, and then enrich the modeling by the temporalities of the graph.

3.1. Ride-Sharing

The problem of ride-sharing is a general problem where one (or more) driver, equipped with one (or more respective) vehicle, has to pick up one or more passengers, dropping them off at one or more arrival bases. Major features of the problem refer to:

i. **Independence**: every driver is independent from the others:
ii. **Automatic-matching**: a central logic unit is the matching agency (system), facilitating the ride-sharing arrangement;
iii. **Cost-sharing**: the grand total travel cost is considered, only;
iv. **Carpooling**: the ride-sharing participants are known in advance, and the matched commuters usually have similar schedules, starting locations, and arrival destinations, or the driver who provides the ride service does not need to detour from his/her preferred route;
v. **Dynamic**: ride-sharing arrangement system may re-adjust strategies at run-time, to facilitate the ride-sharing services according to run-time input.

To find the optimal, or sub-optimal, solution, some of the optimization goals can be:

i. **Number of drivers**: minimize the total number of required drivers;
ii. **Total distance/time**: minimize the total travel distance/time of drivers' trips;
iii. **Travelling time of passengers**: minimize the total travel time of passengers' trips;
iv. **Served requests**: maximize the number of matched (served) requests, thus collecting as many passengers as possible;
v. **Cost for drivers' trips**: minimize the cost for the drivers' trips;
vi. **Cost for passengers' trips**: minimize the cost for the passengers' trips.

We initially focus on static ride-sharing, i.e. all the constraints are known before starting the journey, and on temporal aspects. We assume to have one vehicle, one driver, many passengers (home patients), and one unique common arrival destination – the hospital or care center, where patients have their visits scheduled, and where patients must arrive on time. We specifically focus on temporal constraints, involving both the patients and the driver.

We formalize the problem, considering the grand total travel time and the requests from every patient, in terms of pick-up and drop-off time constraints. Moreover, we want to model some temporal uncertainty, resulting in a more complex problem with respect to the simpler version with no temporal uncertainty. This enhances the ability of the system to deal with real-case scenarios, where passengers want to share rides, but they want also to reach their destination within a certain schedule.

3.1.1. Problem Formalization for Ride-Sharing by Graphs

The entire problem can be formalized as a graph $G = (V, E)$, with a non-empty set of vertexes (or nodes) V, and a non-empty set of edges E. Each edge is a connection between two nodes $v_i, v_j \in V$. The cardinality of V, denoted as $|V|$, is the number of nodes in G: analogously, $|E|$ is the number of edges. Given a pair of nodes $v, u \in V$, the edge e between u and v is represented as $e = \{u, v\}$. The degree of a vertex v, namely $deg(v)$, is the number of edges incident to v. An undirected graph features edges with no direction: given $e = \{u, v\}$, we can "traverse" the edge from u to v, as well as backward. A directed graph requires edges to have a direction, so they can be traversed in one direction, only. A graph can be traversed, namely, some paths can be constructed through it. We define a walk as an alternating sequence between nodes and edges, and if the edges are all different, then we define the walk as a path. A graph is connected if at least one path between every node exists. An edge can have a weight, i.e. a value associated with that edge. In a more complex scenario, a weight can involve more values, with a particular meaning, thus increasing the overall complexity of the graph.

We can express a road network by an undirected weighted multi-graph G, which consists of a set V of vertexes (crossroads in the network) and a set E of edges, where each edge $\{u, v\}$ represents a road between u and v. The multi-graph is a special kind of graph where more edges between a pair of nodes are permitted. Thus some edges can exist like:

$$e_1 = \{u, v\}, e_2 = \{u, v\}, ..., e_3 = \{u, v\} \tag{1}$$

In our case, the weight of an edge $e = \{u, v\}$ represents the *length* (in Km) of a specific road from u to v.

By the graph, we can then construct the following formalization. Given a set of persons (one driver d_i and many passengers p_i), everyone has a ride r_i, which is composed of two nodes in G: for instance $r_i = \{u_i, v_i\}$, where u_i is the starting point and v_i is the ending point for passenger p_i. Nodes in G are road intersections: thus every passenger p_i has as a starting/ending point one of such intersections. This is a simplification of the problem, assuming that p_i will reach the nearest intersection from his/her original position. In the following, we perform spatial reasoning at the intersection granularity.

Each passenger p_i has also two time constraints: a leaving time constraint $t(u_i) = [a_{start}, a_{end}]$; and an arrival time constraint $t(v_i) = [b_{start}, b_{end}]$. These constraints are depicted as temporal ranges: a passenger p_i needs to leave the starting place between time a_{start} and a_{end}, and must reach the arrival destination between time b_{start} and b_{end}. The driver, denoted as d_i, has his/her own leaving and arrival temporal constraints. Since we are focusing on static ride-sharing, the order according to which passengers are picked up by the driver is decided in advance: thus, the path must be *one valid* sequence of starting and ending points. A sequence is said to be valid if, for every p_i, its starting point u_i precedes its ending point v_i.

The described ride-sharing problem aims at finding a valid sequence of starting and ending points where, given some temporal constraint, every passenger p_i leaves the starting point in a time $t_{leave} \in t(u_i)$ and arrives at the ending point in a time $t_{arrive} \in t(v_i)$. Next, we formalize the temporal aspects and provide a workflow for the resolution of an instance of the ride-sharing problem.

3.2. Network Modelling

The road network of Subsection 3.1.1 is a graph. We have to detect a valid sequence of starting and ending points in the graph, minimizing the total travel cost, i.e. the overall length of the trip, for both the driver and the passengers. This introduces a complexity element, i.e. the minimization of the total distance, to increase the satisfaction both of the driver and of the passengers.

Once we have identified in the graph the starting and arrival points, we can construct a *distance network* to include distances between points, and a *temporal constraint network* to consider temporal constraints when moving from one point to another one. We thus obtain one network where the weight of every edge represents the distance between two points and one network where the weight represents time ranges (intervals or durations). These networks are composed of $2n$ nodes, where n is the number of persons sharing the ride (the driver is included). Each node represents a starting point or an ending point. Every network is a complete graph: every edge $e_k = \{u, v\}$ has a weight $w(e_k)$ which represents the DISTANCE or the TEMPORAL CONSTRAINT of the shortest path between node u and node v.

The symmetric $2n \times 2n$ matrix Q depicts the distance network, where $Q[i, j]$ defines the DISTANCE of the shortest path between node i and j. We assume in the following that the shortest path between every couple of nodes is already computed in the distance network, e.g. by OSMnx [19].

By Q, we compute the valid sequence which minimizes the total travel distance. By brute force, we compute all the possible permutations for the n persons, therefore $2n$ points (every person has a starting and ending point). The valid sequence that minimizes the total travel distance can be found in $O(2n!) \in O(n!)$. In real-case applications, cars can have up to five seats (i.e., n = 5, five persons including the driver): the application will have to deal with five persons, including the driver. Considering that the driver starting point will be the first one in the permutation, and the ending point will be the last one, we expect at most to compute $((5 - 1) \times 2)! = 8! = 40320$ permutations.

Example 1. We assume to have three persons, i.e. one driver and two passengers, sharing one ride. Every person has starting and ending points, and temporal constraints (pick-up and drop-off times).

From the real-world map and having 3 passengers, we find six nodes ($3! = 6$), and we build the *distance network* of Figure 1. Considered nodes are:

- **Driver** d:
  - Starting point: F
  - Ending point: D
  - Departure time interval: $d_{start} = [t_{s1}, t_{s2}]$
  - Arrival time interval: $d_{end} = [t_{e1}, t_{e2}]$
- **Patient** p_1:
  - Starting point: A
  - Ending point: B
  - Departure time interval: $p_{1start} = [q_{s1}, q_{s2}]$
  - Arrival time interval: $p_{1end} = [q_{e1}, t_{e2}]$
- **Patient** p_2:
  - Starting point: C
  - Ending point: E
  - Departure time interval: $p_{2start} = [w_{s1}, w_{s2}]$
  - Arrival time interval: $p_{2end} = [w_{e1}, w_{e2}]$

The *distance network* results in the complete graph of Figure 1, with $|V| = 6$. Each edge $e_k = \{u, v\}$ depicts the shortest path between nodes u and v, and its weight $w(e_k) \in \mathcal{R}$ depicts the length of the such shortest path. The *distance network* can be represented by the 6×6 symmetric matrix Q:

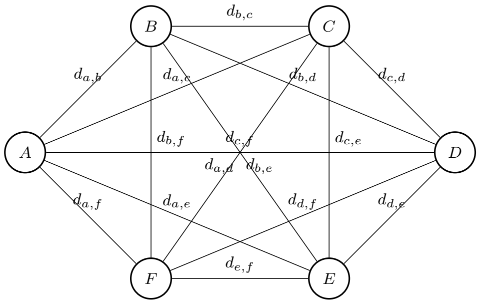

Figure 1: Complete graph with distances (*distance network*).

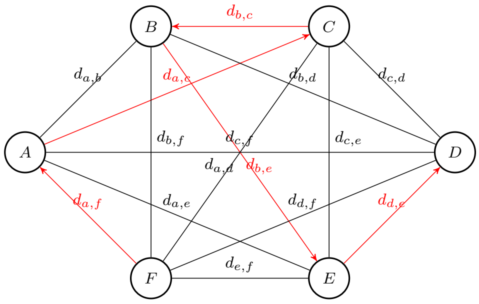

Figure 2: Shortest path α, according to distances.

$$Q = \begin{bmatrix} 0 & d_{ab} & d_{ac} & d_{ad} & d_{ae} & d_{af} \\ d_{ba} & 0 & d_{bc} & d_{bd} & d_{be} & d_{bf} \\ d_{ca} & d_{cb} & 0 & d_{cd} & d_{ce} & d_{cf} \\ d_{da} & d_{db} & d_{dc} & 0 & d_{de} & d_{df} \\ d_{ea} & d_{eb} & d_{ec} & d_{ed} & 0 & d_{ef} \\ d_{fa} & d_{fb} & d_{fc} & d_{fd} & d_{fe} & 0 \end{bmatrix}$$

We now have to find the valid sequence of nodes minimizing the overall travel distance. We recall that a sequence is valid if, for every person, its starting point precedes its ending point. The considered permutations concern passenger nodes, only, since the driver's nodes are fixed. Let's suppose that the resulting valid permutation featuring the shortest path of Example 1 is (Figure 3):

$$\alpha = [F, A, C, B, E, D] \tag{2}$$

We now compute the temporal range for every edge in α: this adds one more element of complexity, namely the temporal dimension. The path from a place to a destination requires some time, depending on traffic conditions, speed limits, and other aspects. In the *temporal constraint network*, we assign to each edge e_k, i.e. to the shortest path between two nodes in the road network, a range $e_k(t) = [t_1, t_2]$, where $t_1, t_2 \in \mathcal{N}$. The range depicts the possible duration required to traverse the path, according to two hypothetical average speeds. For example, a route inside a city has some speed limits. In this case, the lower bound of $e_k(t)$, namely t_1, refers to an average speed of 30 km/h, whereas the upper bound t_2 refers to an average speed of 50 km/h (see Figure 3). Specific lower and upper bounds can be set, according to the possible speed limits in the different route segments. By the distances from the road network, very simple computations assign the time range for every edge in the network. The

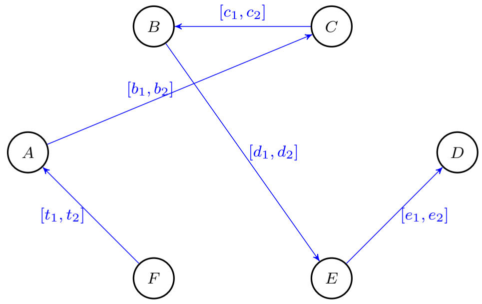

Figure 3: Shortest path α with temporal intervals.

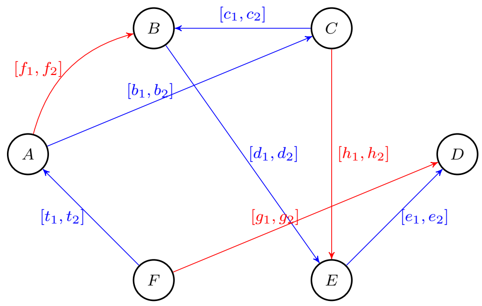

Figure 4: Shortest path α with temporal intervals and arrival constraints.

number of edges in a complete graph with n nodes is $\binom{n}{2}$: in our scenario – up to 5 passengers (i.e., 4 patients and 1 driver) – we obtain a *temporal constraint network* with at most $5 \times 2 = 10$ nodes, which results in $\binom{10}{2} = 45$ edges, assuming that each passenger has starting and ending points different from the ones of the other passengers. In our examples, we shall consider as starting points the different points of each passenger (the first point being that of the driver) and one single ending point (i.e., the location of the healthcare center). Thus, we shall have 6 nodes, with at most $\binom{6}{2} = 15$ edges.

The path α has now a time range, α_t. Every person (driver or passenger) has to reach the destination within a given temporal constraint between the departure time and the arrival time. The constraint is expressed as a temporal range, i.e. an upper bound and a lower bound. This constraint further increases the complexity of the requests and could depend on the patient's condition. For instance, a patient requires to have an overall journey not longer than 30 minutes. Thus, the allowed temporal range for the patient's journey could be [0, 30] minutes. To define this kind of constraint, we add an edge e_{dest} for every participant from the starting point to the destination point in α_t, where $e_{dest}(t)$ is derived from $p_{i_{start}}$ and $p_{i_{end}}$ (or d_{start} and d_{end} in case of the driver) or it is a further ad-hoc temporal range. Constraint edges are depicted by red lines, as in Figure 4.

Analogously, one person can express a temporal constraint also for the pick-up time. The patient is not available until 10:00 a.m.: the temporal range will be $[120, \infty]$ minutes after 8:00 a.m.. To represent all these constraints in a homogeneous way, we define a special node (Z, anchor node), which depicts the initial time of the entire network. Z is set to a predefined time, and all the edges referring to a starting time constraint are depicted as *minutes after* the anchor node Z. In the above example, Z is set to 8:00 a.m.

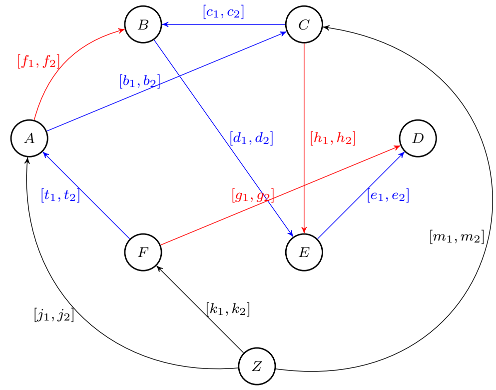

Figure 5: Shortest path α with temporal intervals with arrival and departure constraints.

Therefore, any further constraint is defined with reference to Z. The agent will exploit this to execute the network. Black edges, as in Figure 5, depict such temporal constraints.

A similar approach could be considered also for arrival time. For instance, a patient needs to have an exam in a hospital lab at 9:45 a.m., but the lab opens at 8:30 a.m. and, according to the CoViD-19-restrictions [20], patients cannot enter the hospital too early (more than 30 minutes before the appointment time). Therefore, for no reason, the patient has to reach the lab before. An allowable temporal interval could be [75, 105] minutes after 8:00 a.m.

The resulting network includes all the required temporal constraints. We now extend the network to consider uncertainty, too. Three starting point nodes, namely F, A, C, come with two types of temporal constraints:

i. the three nodes have one incoming edge from Z for the starting time constraint $p_{i_{start}}$;
ii. the three nodes have one outgoing edge for the ending time constraint $p_{i_{end}}$.

The setting of these time points is crucial, as they have implications all over the network. The agent has to consider also the time needed to reach the destination, which depends on the assignments of other nodes. Due to these hard constraints, it may sometime happen that the network is not controllable, i.e. some constraints cannot be fulfilled.

We can specify a temporal range in which we can reach a destination, starting from a location. However, we can encounter something that forces us to reach the destination with some delay, e.g. some unexpected traffic jam, an accident, or a detour. We can expect that in most cases we can move from node X to node Y in a given amount of time, but we cannot be sure of how effectively we can reach X. Therefore, we have to model this scenario in the network by means of *contingent time points*, which introduce *contingent edges* in the network. We define a contingent edge e as a path in a real-world scenario, where we assume the path to be traversed in a given amount of time $t \in [t_1, t_2]$: we can observe the time t only after the event occurred, without controlling it. That temporal range is defined: however, the agent during the execution phase can only observe the outcome of the time assignment, and act consequently to fulfill the temporal constraints.

Dynamic controllability plays a key role. In a not dynamically controllable network, if some contingent time points assume given values, the agent cannot schedule/re-schedule

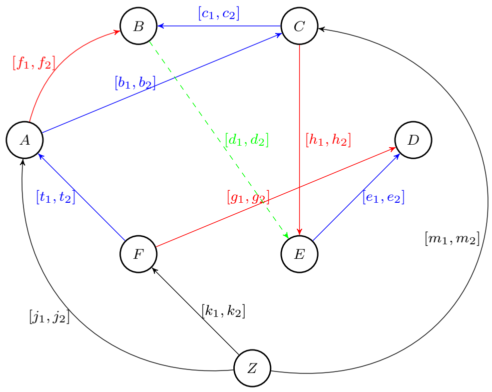

Figure 6: Shortest path α with temporal intervals, contingent edges, arrival, and departure constraints. Green edge: contingent travel time constraint; Blue edge: travel time constraint; Red edge: arrival time constraint; Black edge: departure time constraint.

other time points to fulfill all the temporal conditions, including both trip time and passenger time requirements. In a dynamically controllable network, every passenger's temporal constraints are satisfied, both starting and ending ones, no matter where the contingent time points will be.

One more parametric aspect of the problem refers to which edges are to be considered contingent: therefore their pointing nodes will be contingent time points. As an example, in our context, we may assume that if two locations are in the same district, reasonably the time needed to move between them is controllable: we can ride faster or slower, and we can foresee the arrival time with no particular problem. Otherwise, if we cross two districts, we can experience traffic congestion, or many intersections to cross can sometimes increase the travel time. Therefore, we define contingent edges as those about a path that involves more than one district. In our example, we suppose that points F, A, C, B belong to the same district, whereas points E, D belong to another different district: thus, the edge $B \rightarrow E$ will be considered contingent and depicted by a green line as in Figure 6.

4. Implementation Details

This section describes the proof-of-concept prototype. We first apply the algorithm to check the dynamic controllability of the network by [9]; then, we simulate a real scenario for the RTED strategy, where the agent has to react to a contingent time point and reschedule the ride.

4.1. System Description

We now describe the implementation of the system experimenting our approach. As development tools, we choose Python and the NetworkX package for managing networks. The overall architecture has three modules:

i. STNU management: the module reads the graph of the network, and computes the respective distance graph. Next, the module computes the APSSRP table and checks the dynamic controllability of the distance graph;
ii. network execution: the module analyzes the distance graph network from the previous step, processes all the possible contingency points, and by the RTED strategy computes the execution strategy;
iii. map and route planner: the module connects to an Open Street Map server, and retrieves the real-world map for the ride-sharing scenario. Next, the module identifies the starting and ending points on the map and computes the distances between all the points of the network: the resulting network comes with weighted edges with temporal intervals. Finally, the module computes all the possible permutations and extracts the shortest one.

Moreover, we used an open-source Java tool, allowing the graphical representation and the checking of STNU [21].

4.2. Ride-sharing Instance

We consider a ride-sharing problem in Verona with one driver and three patients. All of them have one starting and one ending point, and temporal constraints for both departure and arrival times. The multiple objectives are:

- minimize the total travel distance, or minimize the costs for all the passengers;
- verify the consistency of the temporal constraints, by means of dynamic controllability;
- schedule the time arrival for every point, simulating a temporal dimension to react to contingent events by means of a RTED strategy.

We start by considering the spatial features of the problem. The driver collects the patients, drops them off at care centers, and then brings them back home. The driver moves from one of the University hospitals, point *Start* in Verona (Figure 7). The driver needs to reach, in the end, a care center in the East area of the city (point *End*, Figure 7). The three patients, located in three different areas of the city, need to reach three different destinations. Precisely, the participants of the ride-sharing process are:

- Patient p_1
  - Starting point 0: Southern District
  - Ending point 1: Southern District
  - Departure time: from 8:00 a.m. to 8:05 a.m.
  - Arrival time: from 8:05 a.m. to 8:15 a.m.
- Patient p_2
  - Starting point 2: Southern District
  - Ending point 3: Western District
  - Departure time: from 8:00 a.m. to 8:10 a.m.
  - Arrival time: from 8:05 a.m. to 8:25 a.m.
- Patient p_3
  - Starting point 4: Western District
  - Ending point *End*: Eastern District
  - Departure time: from 8:05 a.m. to 8:15 a.m.
  - Arrival time: from 8:10 a.m. to 8:30 a.m.

The driver's ending point, departure time, and arrival times are:

- starting point *Start*: Southern District
- ending point *End*: Eastern District
- departure time: from 8:00 a.m. to 8:05 a.m.
- arrival time: from 8:10 a.m. to 8:30 a.m.

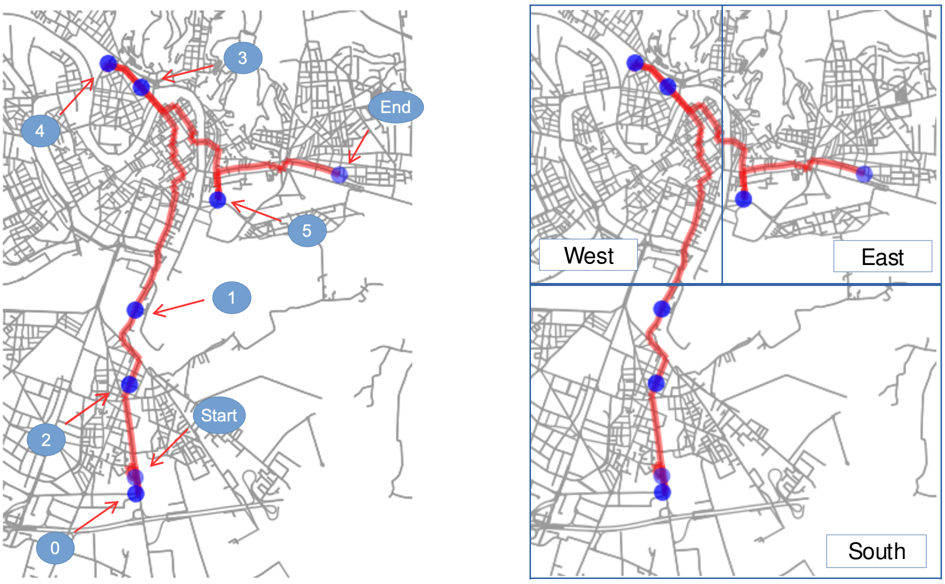

Figure 7: Planned trip (left) and district division (right).

We also add another constraint on the path, namely node 5: it depicts a request from patient p_3 to stop at node 5 to pick up one relative of his/hers for assistance. Thus, both patient p_3 and the driver need to reach a medical center in *E*, but the path must go through node 5, Eastern District. Figure 7 depicts the planned trip, along with the adopted division of districts. District division is relevant: in our formalization, a path that crosses one (or more) district borders is considered to be of uncertain duration. Traffic conditions and other factors could affect major connections in a city.

The resulting path, depicted in red in Figure 7, is the shortest path among all the possible valid permutations, where a permutation is defined as valid if every starting point of every patient precedes its respective ending point. The first and last points are fixed, describing the driver's starting point and ending point, respectively. The path chosen as the shortest one, called α, is composed as:

$$\alpha = [Start, 0, 2, 1, 3, 4, 5, End] \quad (3)$$

and it is 12.44 km long. In particular, the path is a combination of the shortest paths among points, whose distances and temporal constraints are: from *D* to 0: 0.7 km with time interval [1,1]; from 0 to 2: 1.504 km with time interval [2,3]; from 2 to 1: 1.109 km with time interval [1,2]; from 1 to 3: 4.004 km with time interval [5,8]; from 3 to 4: 0.685 km with time interval [1,1]; from 4 to 5: 2.303 km with time interval [3,5]; from 5 to *E*: 2.135 km with time interval [3,4].

We can now continue by considering the temporal features of the problem. Time ranges are estimated at a constant speed of 50 km/h as the lower bound, and at a speed of 30 km/h as the upper bound. The selected path minimizes the overall distance since the path will reduce the total travel cost: with a certain probability, the path can also be feasible with regard to the temporal constraints imposed by the participant (patient or driver).

Figure 8 depicts the formalized network. Green dotted edges depict contingent edges, which lead to contingent time points. Since those edges depict paths that cross district borders, we cannot predict exactly how much it will take to run these paths with respect to the computed time range.

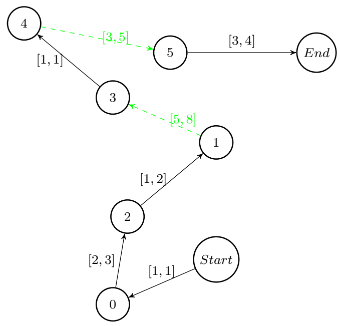

Figure 8: Real world network formalization.

The network is then enriched by two other kinds of temporal constraints, namely arrival and destination constraints. As explained in Section 3.2 and by Figure 5, we add a special node Z which represents "time zero". In our example, Z is set as 8:00 a.m., which is the anchor timestamp according to which temporal constraints are defined.

Figure 9 depicts the network previously obtained–with temporal ranges related to the time required to move from a point to the next one according to the derived route– completed with the temporal constraints related to patient $p1$, who has to move from point 0 to point 1, and to the driver, who moves from *Start* to *End*.

After running the procedure, we extrapolate the complete network and are able to verify that, in this case, the network is controllable.

4.3. Feasible Solutions

Not every valid path, namely a sequence where each starting point is reached before its respective ending point, is feasible. In the above example, the driver does not participate in counting all the possible permutations, having a fixed starting point and a fixed ending point: the starting point is the first one, and the ending point is the last one, with respect to all the other participants. Thus, the remainder 3

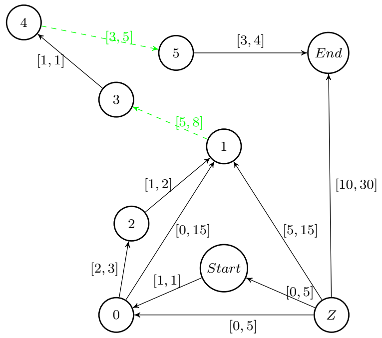

Figure 9: Real world network formalization with some passengers' constraints.

$patients \times (pick\text{-}up\ point + drop\text{-}off\ point) = 6$ points need to permute, resulting in $6! = 720$ possible paths. This set is the "all paths" set: among those paths, we obviously consider only the valid ones, i.e. we cannot drop a passenger off at the destination before picking the passenger up. This reduces the space to 90 valid paths, which is $12,5\%$ of all paths. We refer to them as the "valid paths". Moreover, we find the "feasible paths", that both are valid and fulfill all the temporal constraints. The "feasible paths" set is fully contained in the "valid paths" set: we shall have at most 90 feasible paths. Reasonably, the number of feasible paths is smaller that the number of valid paths, for a not-trivial network with reasonable time constraints.

A valid path could be not feasible due to two main reasons (or both of them):

- the *requested* time constraint is too strict;
- the distance between points is too large, so it results in a long travel time for that specific path, which will not satisfy the time constraint(s).

As an example, in Figure 10 we insert a time constraint that is too strict: the resulting network will be not dynamically controllable. In fact (Figure 10), the red line depicts a too-strict time constraint from node 0 to node 1. We remind that patient p_1's starting point is 0, and the ending point is 1, so the request edge can be translated as "patient p_1 needs to reach the destination between 1 and 2 minutes after departure". It can be easily observed that, since we have to pass through point 2, which is patient p_2's starting point, we can reach point 1 at least 3 minutes after departure: the added constraint is clearly not satisfiable.

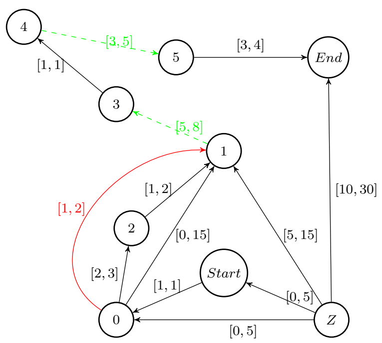

Figure 10: Non dynamically controllable example

5. Discussion and Conclusions

We faced the problem of ride-sharing, where two or more passengers want to share a ride: the goal is that of minimizing the overall length of the trip. To simplify the scenario, we assume to have one driver and one car, two or more passengers with their respective starting points, and one common final destination. In a real-case scenario, passengers may also have some temporal constraints, referring to the pick-up time or drop-off time. During the trip, some events may occur, such as traffic congestion, detour, and - more generally - delays: this adds uncertainty to the problem. Moreover, crossing city districts increases the probability of encountering such events, and may force them to switch from a statically planned trip to a dynamically planned one, where decisions must be taken at run time.

As an application domain, we considered medical transportation: passengers are patients who need to reach the common care center, where some visits/therapies/treatments are scheduled for them. This feature adds even more temporal constraints. In this paper, we formalized the problem by graphs, deploy spatial and temporal networks to analyze the graphs, and demonstrate the approach by a running prototype.

5.1. Future Research Directions

We consider here some future research directions. We plan to enrich the analysis to consider more complex situations, e.g. having more drivers, more cars, more than 5 passengers per car such as in vans, as well as considering the return trip, picking up the patients from the care center, and dropping them back home. More constraints need to be considered, e.g. a patient who went through a radio-therapy can't be transported in the same vehicle with a pregnant patient or with a kid. To this end, we have to face the scalability issues of this inherently intractable (NP) problem.

The analysis can be further extended to consider the patient's priority, which could help in increasing the revenues of the care center by avoiding dead times of highly expensive instrumentation, or in avoiding insurance claims.

The analysis can also consider emergency situations, thus prioritizing patients according to several facets, including the patient's status, type of disease or injury, and resource availability in the care centers.

Acknowledgments

C.C., A.F., and R.P. are partially funded by Dipartimento di Informatica, University of Verona, Italy. G.P. is partially funded by the EU H2020 program: "PERISCOPE: Pan European Response to the ImpactS of CoViD-19 and future Pandemics and Epidemics" (grant n. 101016233) and by Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy.

References

[1] N. Chan, S. Shaheen, Ridesharing in North America: Past, present, and future, Transport Reviews 32 (2012) 93–112. doi:10.1080/01441647.2011.621557.

[2] K. H. Chaiyachati, R. A. Hubbard, A. Yeager, B. Mugo, J. A. Shea, R. Rosin, D. Grande, Rideshare-based medical transportation for medicaid patients and primary

care show rates: A difference-in-difference analysis of a pilot program, Journal of General Internal Medicine 33 (2018) 863–868.

[3] B. W. Powers, S. Rinefort, S. H. Jain, Nonemergency medical transportation: Delivering care in the era of lyft and uber, JAMA 316 (2016) 921–922. URL: https://doi.org/10.1001/jama.2016.9970. doi:10.1001/jama.2016.9970.

[4] R. Collier, Uber enters medicine but disrupting health care may prove difficult, CMAJ 190 (2018) E756–E757. URL: https://www.cmaj.ca/content/190/24/E756. doi:10.1503/cmaj.109-5615.

[5] R. Dechter, I. Meiri, J. Pearl, Temporal constraint networks, Artif. Intell. 49 (1991) 61–95. URL: https://doi.org/10.1016/0004-3702(91)90006-6. doi:10.1016/0004-3702(91)90006-6.

[6] N. Muscettola, P. P. Nayak, B. Pell, B. C. Williams, Remote agent: To boldly go where no AI system has gone before, Artif. Intell. 103 (1998) 5–47. URL: https://doi.org/10.1016/S0004-3702(98)00068-X. doi:10.1016/S0004-3702(98)00068-X.

[7] L. Hunsberger, Efficient execution of dynamically controllable simple temporal networks with uncertainty, Acta Informatica 53 (2016) 89–147. URL: https://doi.org/10.1007/s00236-015-0227-0. doi:10.1007/s00236-015-0227-0.

[8] M. Cairo, L. Hunsberger, R. Rizzi, Faster dynamic controllability checking for simple temporal networks with uncertainty, in: N. Alechina, K. Nørvåg, W. Penczek (Eds.), 25th International Symposium on Temporal Representation and Reasoning, TIME 2018, Warsaw, Poland, October 15-17, 2018, volume 120 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagsthul, Germany, 2018, pp. 8:1–8:16. URL: https://doi.org/10.4230/LIPIcs.TIME.2018.8. doi:10.4230/LIPIcs.TIME.2018.8.

[9] P. H. Morris, N. Muscettola, Temporal dynamic controllability revisited, in: M. M. Veloso, S. Kambhampati (Eds.), Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA, AAAI Press / The MIT Press, 2005, pp. 1193–1198. URL: http://www.aaai.org/Library/AAAI/2005/aaai05-189.php.

[10] L. Hunsberger, R. Posenato, Simpler and faster algorithm for checking the dynamic consistency of conditional simple temporal networks, in: J. Lang (Ed.), Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden, ijcai.org, 2018, pp. 1324–1330. URL: https://doi.org/10.24963/ijcai.2018/184. doi:10.24963/ijcai.2018/184.

[11] P. H. Morris, A structural characterization of temporal dynamic controllability, in: F. Benhamou (Ed.), Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings, volume 4204 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 375–389. URL: https://doi.org/10.1007/11889205_28. doi:10.1007/11889205_28.

[12] P. H. Morris, Dynamic controllability and dispatchability relationships, in: H. Simonis (Ed.), Integration of AI and OR Techniques in Constraint Programming - 11th International Conference, CPAIOR 2014, Cork, Ireland, May 19-23, 2014. Proceedings, volume 8451 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 464–479. URL: https://doi.org/10.1007/978-3-319-07046-9_33. doi:10.1007/978-3-319-07046-9_33.

[13] M. K. Wolfe, N. C. McDonald, G. M. Holmes, Transportation barriers to health care in the United States: Findings from the National Health Interview Survey, 1997–2017, American Journal of Public Health 110 (2020) 815–822. URL: https://doi.org/10.2105/AJPH.2020.305579. doi:10.2105/AJPH.2020.305579, pMID: 32298170.

[14] M. K. Wolfe, N. C. McDonald, Innovative health care mobility services in the US, BMC Public Health 20 (2020). doi:10.1186/s12889-020-08803-5.

[15] Y. Molenbruch, K. Braekers, A. Caris, G. V. Berghe, Multi-directional local search for a bi-objective dial-a-ride problem in patient transportation, Comput. Oper. Res. 77 (2017) 58–71. URL: https://doi.org/10.1016/j.cor.2016.07.020. doi:10.1016/j.cor.2016.07.020.

[16] P. L. van den Berg, J. T. van Essen, Scheduling non-urgent patient transportation while maximizing emergency coverage, Transp. Sci. 53 (2019) 492–509. URL: https://doi.org/10.1287/trsc.2018.0823. doi:10.1287/trsc.2018.0823.

[17] R. Fulgenzi, S. Gitto, G. Murgia, E. Pessot, Simulation of patient-centred scenarios for the improvement of transportation service in hospitals, in: L. M. Camarinha-Matos, Á. Ortiz, X. Boucher, A. L. Osório (Eds.), Collaborative Networks in Digitalization and Society 5.0 - 23rd IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2022, Lisbon, Portugal, September 19-21, 2022, Proceedings, volume 662 of *IFIP Advances in Information and Communication Technology*, Springer, 2022, pp. 356–365. URL: https://doi.org/10.1007/978-3-031-14844-6_29. doi:10.1007/978-3-031-14844-6_29.

[18] H. Thai, J. Huh, Optimizing patient transportation by applying cloud computing and big data analysis, J. Supercomput. 78 (2022) 18061–18090. URL: https://doi.org/10.1007/s11227-022-04576-3. doi:10.1007/s11227-022-04576-3.

[19] G. Boeing, OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks, Comput. Environ. Urban Syst. 65 (2017) 126–139. URL: https://doi.org/10.1016/j.compenvurbsys.2017.05.004. doi:10.1016/j.compenvurbsys.2017.05.004.

[20] C. Combi, G. Pozzi, Health informatics: Clinical information systems and artificial intelligence to support medicine in the CoViD-19 pandemic, in: 9th IEEE International Conference on Healthcare Informatics, ICHI 2021, Victoria, BC, Canada, August 9-12, 2021, IEEE, Los Alamitos, CA, USA, 2021, pp. 480–488. URL: https://doi.org/10.1109/ICHI52183.2021.00083. doi:10.1109/ICHI52183.2021.00083.

[21] R. Posenato, CSTNU tool: A Java library for checking temporal networks, SoftwareX 17 (2022) 100905. URL: https://doi.org/10.1016/j.softx.2021.100905. doi:10.1016/j.softx.2021.100905.