

Dependable Dynamic Routing for Urban Transport Systems Through Integer Linear Programming

Davide Basile^{2 1}, Felicita Di Giandomenico¹, and Stefania Gnesi¹

¹ I.S.T.I “A.Faedo” ² Dept. of Information Engineering
CNR Pisa, Italy University of Florence, Italy

Abstract. Highly automated transport systems play an important role in the transformation towards a digital society, and planning the optimal routes for a set of fleet vehicles has been proved useful for improving the delivered services. Traditionally, routes are planned beforehand. However, with the advent of autonomous urban transport systems (e.g. autonomous cars), possible obstructions of tracks due to traffic congestion or bad weather conditions need to be handled on the fly. In this paper we tackle the problem of dynamically computing routes of vehicles in urban lines in the presence of potential obstructions. The problem is formulated as an integer linear optimization problem. The proposed algorithm will assign routes to vehicles dynamically, considering the track segments that are no longer available and the positions of the vehicles in the urban area. The recomputed routes guarantee the minimal waiting time for passengers. Safety of the computed routes is also guaranteed.

1 Introduction

Nowadays, most of the research in the transport sector is devoted to build smart solutions for moving people within the cities, to reduce costs and improving sustainability while ensuring reliability and safety of the transport services. Highly automated transport systems play an important role in the transformation towards a digital society and technologies as driver-less transports are already adopted in metropolitan cities [17]. In particular, planning the optimal routes for a set of fleet vehicles has been proved useful for reducing costs and energy consumption of vehicles while improving user satisfiability in terms of waiting time.

This problem has been widely studied in the literature [16, 15]. Traditionally, two-step approaches based on planning fixed routes and execute them have been studied and are adopted in the railway industry. These approaches rely on the availability of tracks, which is in general guaranteed for railway tracks but it is no longer possible in urban area, where events such as obstructions of tracks must be handled. More recently, newly dynamic routing applications are emerging, thanks to a number of technological advances. For example, the increasing hardware performances for data processing, together with accurate positioning systems as Global Positioning Systems (GPS) and Geographic Information Systems (GIS) led to the development of Intelligent Transport Systems (ITS). These systems combine the above technologies and made possible to track fleet vehicles and to manage them in real time.

In particular, the possibility of dynamically computing new routes for vehicles opens new opportunities for reducing operational costs and environmental impact while improving customer services dependability. Indeed, especially in urban area it is often the case that itineraries may be temporarily unavailable due to obstructions. In this case, a mechanism can be adopted to recompute dynamically new routes for the affected vehicles, such that they are able to complete their missions. This aspect is of crucial importance for improving the overall dependability of these urban transport services and improving the user satisfiability.

In this paper we propose a routing algorithm for handling possible detected obstructions of tracks in urban area, by assigning new routes dynamically and by considering a set of tracks temporarily unavailable. The proposed algorithm takes in input a graph abstracting an urban map, where edges correspond to itineraries and nodes to points, the locations and destinations of vehicles in the urban area and the set of detected obstructed tracks. The output of the algorithm is the set of optimal routes for each vehicle, to be communicated to the vehicles until the obstructed tracks are restored to their normal operation. The optimal routes computed are safe by construction. In particular, it is guaranteed that no collisions among vehicles will ever occur both on itineraries and on points. Moreover, the computed routes guarantee progress of the overall network of vehicles: no deadlocks will ever occur, i.e. each vehicle eventually reaches its destination. Note that, although there are specific subsystems strongly tailored to assure safety (e.g., interlocking), also at the level of route planning safety can be considered by developing solutions that avoid potential train collisions, as we pursue in our study.

We modelled the dynamic vehicle routing as an optimization combinatorial problem, through a set of linear equations. In particular, the vehicle routes are modelled as flows in a graph such that the objective function minimises the arrival time of each vehicle to its destination. This in turns guarantees an improvement in user satisfiability by minimising the waiting time. Safety aspects are enforced by a set of constraints allowing only one vehicle in each itinerary and only one vehicle to traverse a point in a given time step.

The proposed model has been implemented in *A Mathematical Programming Language* [8](AMPL). Preliminary experiments were performed showing the feasibility of the proposed approach. The implementation of the dynamic vehicle routing algorithm is open source. It can be downloaded at <https://github.com/davidebasile/routingproblem>, together with data and set-ups of experiments.

Structure of the paper The paper starts with a description of the problem in Section 2. The proposed architecture of a dependable dynamic vehicle routing system is introduced in Section 3. Section 4 contains some background on Integer Linear Programming (ILP) and flow problems; and the proposed model for solving the routing problem is described in Section 5. The implementation of the algorithm and some experiments are, respectively, in Section 6 and Section 7. Finally, related work is in Section 8 while conclusion and future work are in Section 9.

2 Description of the Problem

Planning the time schedule and routing of vehicles (known as Dynamic Vehicle Routing Problem) is a problem that has been widely researched and nowadays several transport systems adopt automatic solutions for planning the routes of vehicles and for supervising their movements [9, 16, 15].

Recently, these systems have been extended from subway and train lines to comprehend other urban systems, as tramway lines. Tramway lines are generally less expensive than subway lines and automatic systems can be applied to optimize the time scheduling and energy consumption. Solutions as signals, priority management and traffic lights are adopted to regulate the circulation and ensure safety. While metropolitan lines widely adopt automated guidance systems, in tramway systems the driver is in charge of enforcing speed, braking and safety distances. Generally, signal entities are used to allow trams to occupy the specified route.

An important problem in urban scenarios is the presence of possible obstructions in the assigned routes. This can be due, for example, to other vehicles or to accidents. Generally technologies as, for example, radars and gps are used to detect these hazardous situations. Hence, implementing innovative dependable routing solutions while enforcing rail safety represents a challenge for the research community.

In particular when a specific route is no longer available due to obstructions of the path or other possible failures, the preassigned routes are no longer valid. It is important to recompute efficiently a new route from the location of each vehicle to its destination, to avoid obstructed tracks and potential deadlocks. Signalling systems are in charge of communicating to the drivers the newly assigned routes, set up the traffic lights, commute points, and set up the other devices composing the signalling system.

3 Dynamic Vehicle Routing

In Figure 1 our proposed dynamic vehicle routing system is depicted. In particular, through the on-board equipment each vehicle can communicate its precise location thanks to GPS coordinates or similar systems. Moreover, communications with the control station are also handled. In case of possible obstructions in one of the assigned tracks (detected by sight or by automatic devices as, e.g. radar) the preassigned standard routes are no longer valid; and the blocked vehicle will communicate to the central control station its coordinates and will identify such obstructed track. In this scenario it is necessary to adopt alternative routes until the unavailable tracks are restored to their normal operation conditions. We assume that the unavailable tracks notified to the control station will remain so for an amount of time worthy of recomputing new routes. On the contrary, vehicles will wait until the obstructed tracks are restored to normal operation conditions.

Once the communication has been received by the control unit, the coordinates (also called locations) and the destinations of all vehicles in the urban area will be collected by the control unit. These data will be used by the control system to compute new routes for each vehicle dynamically, given its current location and its destination,

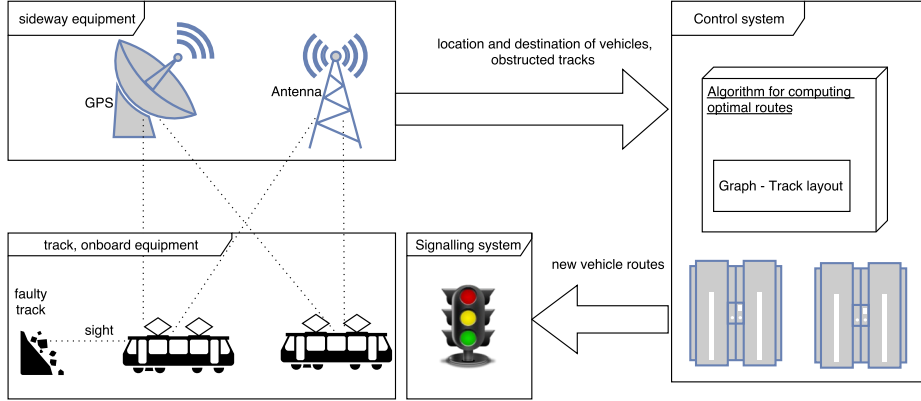


Fig. 1: The Dependable Dynamic Vehicle Routing System for Urban Lines

and communicate them to the signalling system. In our framework destinations are, for example, next stops, i.e. we divide a round trip of a vehicle into a sequence of stops that are computed dynamically. Our proposed model will compute the optimal solution by minimising the overall time needed by all vehicles to arrive at their destination, that is optimising the user satisfiability in terms of minimal waiting time. Moreover, the model will ensure route safety, i.e. no collisions on tracks or points will ever occur.

The newly computed routes are communicated to the drivers and to the signalling system. Indeed, the problem of setting traffic lights, commutating points and other operations on the tracks useful to implementing the selected routes are managed by other systems. It is assumed that other systems are in charge of communicating to the drivers the assigned route and to implement the signalling system to allow each vehicle to move according to its selected route (see Section 2).

In the following sections the algorithm for computing new routes (right block in Figure 1) is specified, implemented and tested. Note that the proposed algorithm is not tailored to a specific urban transport system, but can be reused in different scenarios such as, among the others, autonomous cars and tramway lines.

4 Network Flow Problem

In this section we introduce network flow problems and their formalisations. The dynamic vehicle routing problem will be formalised and solved as a network flow problem in the following section.

A flow network [6] (also known as a transportation network) is a directed graph where each edge has a capacity and each edge receives a flow. Let $G = (Q, T)$ be a graph with set of nodes Q and edges T , that are pair of nodes. Generally there are two types of special nodes: *source nodes*, that are generating flow, and *sink nodes*, that are consuming the flow. Given a node $q \in V$, the *forward star* $FS(q)$ is the set of outgoing edges of q , while the *backward star* $BS(q)$ is the set of incoming edges in the node q .

For each edge $t \in T$, the *flow variable* x_t represents the flow that is passing through the edge t . Generally, a maximum capacity a_t is assigned to each edge t , representing the maximum amount of flow allowed, and a cost c_t representing the cost of utilising the edge t . A network flow problem is a type of network optimization problem where the objective function requires to optimize a flow such that the solution respects the following constraints:

- the amount of flow on an edge cannot exceed the capacity of the edge (*capacity constraints*), written $\forall t \in T. x_t \leq a_t$;
- the amount of flow incoming into a node equals the amount of flow leaving it, unless it is a source, with only an outgoing flow d , or a sink, with only an incoming flow d (*flow conservation*), written:

$$\forall q \in Q. \sum_{t \in BS(q)} x_t - \sum_{t \in FS(q)} x_t = \begin{cases} -d & \text{if } q = q_s \\ 0 & \text{if } q \neq q_s, q_f \\ d & \text{if } q = q_f \end{cases}$$

- depending on the studied problem, it can be required that the computed flow must be an integer value (*integrality constraints*), written: $\forall t \in T. x_t \in \mathbb{N}$.

Examples of network flow problems are the *Maximum flow problem* [7] or the *Minimum-cost flow problem* [11]. The first problem consists in maximizing the amount of flow that can be sent from the source nodes to the sink nodes. The objective function is then $\max d$. In the second problem a cost is associated with each edge of the network, and the objective function is minimised in order to find the optimal cost for sending a given amount of flow from the source nodes to the sink nodes, that is $\min \sum_{t \in T} x_t c_t$.

These problems are solved by using *Integer Linear Programming* (ILP) [10, 19]. Indeed, all constraints are represented by linear inequalities, and the objective function is linear. Several solvers are available for solving linear optimization problems automatically and efficiently, by using for example the simplex algorithm [8].

In the next section we will formalise the automatic route scheduling as a flow problem. The flow variables will be split into time steps $1, \dots, K$, where K will be the upperbound to the maximum number of edges that a route can traverse. The maximum capacity for each edge will be of one unit, that is only one vehicle can be on a specific track in a specific moment. Similarly, the flow d will be of one unit, that is each flow will be in correspondence with a single route. We will not consider costs for edges, which are left as future work (e.g. energy, performance). Finally, the flow variables will be split into a set of binary variables $x_{u,k,t}$ where u identifies the vehicle, k identifies the discrete step considered in our analysis and t will identify the itinerary (i.e. edge). In particular, $x_{u,k,t} = 1$ if and only if vehicle u at moment k is in itinerary t . These flow variables will describe the optimal routes computed by our ILP model. The goal will be to minimise the overall routing time.

5 Description of the Model for the Vehicle Routing Problem

In this section we formalise the dynamic vehicle routing problem as a network flow problem. Similarly to [20, 14, 3], we abstract a generic urban tramway layout as a graph.

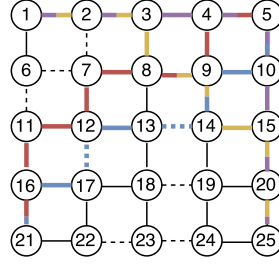


Fig. 2: The grid used for the experiments with the routes computed in Experiment 1 (thick edges) and obstructed itineraries (dotted edges)

At our level of abstraction, we are only interested in modelling the path that each vehicle must traverse in order to arrive at its destination. A destination could be the next stop that the vehicle needs to reach. Each edge of the graph will possibly represent a sequence of segments where a single vehicle is allowed (i.e. an itinerary), that must be traversed in order to move from one point to another. Nodes in the graph are in correspondence with points in the track. We assume that vehicles may only get stuck in points, and not while traversing itineraries. Indeed, unavailability of itineraries is ascertained in the nearest points.

We firstly introduce the notation used in this section. We assume a finite set of vehicles U , where each vehicle has one route, a finite set of itineraries (i.e. edges) T , a finite set of nodes Q . Trivially, no segment has the same point as source and destination. Indeed, we assume that no inner cycles are present in the graph, i.e. $\forall q \in Q. FS(q) \cap BS(q) = \emptyset$. This requirement can be imposed as a constraint in the model (see Equation 13).

Moreover, let $|S|$ be the cardinality of a set S . Then $K = |U| * |T|$ is the upper bound to the maximum amount of time needed by each vehicle to reach its destination provided that at each discrete step $k \in 1 \dots K$ at least one vehicle in U moves into an itinerary in T . In particular in the worst case $|U| * |T|$ only one vehicle moves at each step, all vehicles need to traverse all itineraries in the graph and each route traverses each itinerary at most one time (i.e. no loops). Moreover, given a vehicle $u \in U$, let $location(u), destination(u) \in Q$ be the location and destination of vehicle u . Finally, we assume the presence of a subset of itineraries $F \subset T$ that are temporarily unavailable and cannot be traversed. The output of the ILP model will be the new routes assigned to each vehicle.

Example 1. Before providing the details of the model we explain the formalisation with the help of an intuitive example. In Figure 2 a graph representing a sub-portion of an urban area is depicted. Adjacent nodes are connected in both directions, to improve readability for each pair of connected nodes only one edge is reported in Figure 2. In Section 7 this graph will be used for testing the proposed model. Assuming that the obstructed tracks are itineraries in $F = \{(1, 2), (1, 6), (2, 1), (2, 7), (6, 1), (7, 2), (10, 2)\}$

and that two vehicles u_1 and u_2 are present such that $location(u_1) = 15$, $location(u_2) = 9$, $destination(u_1) = 12$, $destination(u_2) = 11$. The optimal route for u_1 is represented by the variables $x_{u_1,1,(15,14)} = 1$, $x_{u_1,2,(14,13)} = 1$, $x_{u_1,i,(13,12)} = 1$ where $i = 3, \dots, K$ (all other variables $x_{u_1,k,t}$ having value zero). The optimal route for u_2 is represented by the variables $x_{u_2,1,(9,8)} = 1$, $x_{u_2,2,(8,7)} = 1$, $x_{u_2,3,(7,6)} = 1$, $x_{u_2,j,(6,11)} = 1$, where $j = 4, \dots, K$ (all other variables $x_{u_2,k,t}$ having value zero). In particular, at step $k = 1$ we have that vehicle u_1 is on itinerary (15, 14) and vehicle u_2 on (9, 8); at step $k = 2$ vehicle u_1 has moved to the adjacent itinerary (14, 3) and u_2 to (8, 7), at step $k = 3$ vehicle u_1 has moved to (13, 12) (so reaching its destination) and u_2 to (7, 6). Finally at step $k = 4$ vehicle u_1 remains idle while u_2 reaches its destination (6, 11).

These $x_{u,k,t}$ variables are computed automatically by the ILP model described below, and are such that each vehicle reaches its destination in the shortest number of steps possible.

5.1 Integer Linear Programming model

The ILP model is defined below.

Objective function We start by defining the objective function:

$$\max \gamma \quad (1)$$

$$\gamma \geq 0 \quad (2)$$

The objective function maximises a threshold γ , which is constrained to be a positive integer. The parameter γ will represent the overall amount of time spent by vehicles in their destinations in terms of number of discrete steps (see Equation 4), i.e. the earliest a vehicle reaches its destination the higher γ will be.

Flow Constraints We now discuss the flow constraints used to model the routes of vehicles. As mentioned before, we will split the time window under analysis into discrete steps $1 \dots K$ such that K is the upper bound to the number of steps needed by each vehicle to reach its destination. In particular, at each discrete step $k \in 1 \dots K$, for each itinerary $t \in T$ and vehicle $u \in U$ a binary variable $x_{u,k,t}$ identifies if vehicle u at step k is in itinerary t . The set of variables $x_{u,1,t_1}, \dots, x_{u,K,t_n}$ set to one will identify the sequence of itineraries (i.e. route) t_1, \dots, t_n that must be traversed by vehicle u to reach its destination t_n starting from its location t_1 , and the discrete steps k that the vehicle must spent in these itineraries.

$$\forall k \in 1 \dots K, \forall u \in U, \forall t \in T. x_{u,k,t} \in \{0, 1\} \quad (3)$$

The following equation ensures that each vehicle u reaches its destination in the minimum possible amount of time. In particular, for all vehicles $u \in U$, steps $k \in K$, and for all itineraries $t \in T$ incoming into each vehicle destination ($t \in BS(destination(u))$), the sum of all variables $x_{u,k,t}$ must be greater or equal to γ .

Indeed, the objective function (1) maximises the threshold γ , and as a result the sum (left hand side term of Equation 4) will be maximised: the earliest each vehicle u

reaches its destination, the higher this sum will be (i.e. vehicles $u \in U$ will spend more time in itineraries $t \in BS(destination(u))$).

$$\sum_{u \in U, k \in K, t \in BS(destination(u))} x_{u,k,t} \geq \gamma \quad (4)$$

Note that constraint 4 also guarantees the problem to be bounded: in particular by constraint 4 it holds that $\gamma \leq |K| * |U|$.

The constraints ensuring that a set of variables $x_{u,1,t}, \dots, x_{u,K,t}$ correctly identify one route are now discussed. The following equation constraints a vehicle u to be in only one itinerary t at each step k .

$$\forall u \in U, \forall k \in 1 \dots K. \sum_{t \in T} x_{u,k,t} = 1 \quad (5)$$

The following equations ensure that each vehicle starts its trip from its current location and arrives at its destination (in the worst case it arrives at step K).

$$\forall u \in U. \sum_{t \in FS(location(u))} x_{u,1,t} = 1 \quad (6)$$

$$\forall u \in U. \sum_{t \in BS(destination(u))} x_{u,K,t} = 1 \quad (7)$$

The constraints below are necessary for ensuring that each vehicle only moves into a connected path or stays idle at each step k . In particular, fixing a vehicle u , for each node q , and step k such that vehicle u is incoming in q at step $k-1$ we require that the difference between the incoming itineraries in q at step $k-1$ and the sum of the incoming and outgoing itineraries at step k (for the same point q and vehicle u) must be equal to zero.

$$\begin{aligned} \forall q \in Q, \forall u \in U, \forall k \in 2 \dots K, \sum_{t \in BS(q)} x_{u,k-1,t} &> 0. \\ \sum_{t \in BS(q)} x_{u,k-1,t} - \left(\sum_{t \in FS(q)} x_{u,k,t} + \sum_{t \in BS(q)} x_{u,k,t} \right) &= 0 \end{aligned} \quad (8)$$

We further detail Equation 8; recall that by Equation 5 and the conditions on constraints 8 ($\sum_{t \in BS(q)} x_{u,k-1,t} > 0$, i.e. vehicle u at step $k-1$ is incoming into node q), it must be that at step k either u is still incoming (i.e. $\sum_{t \in BS(q)} x_{u,k,t} = 1$ and $\sum_{t \in FS(q)} x_{u,k,t} = 0$); or vice-versa (i.e. $\sum_{t \in BS(q)} x_{u,k,t} = 0$ and $\sum_{t \in FS(q)} x_{u,k,t} = 1$), that is u is outgoing from q . However, Equation 8 does not prevent scenarios in which a vehicle moves from one incoming itinerary t in q at step $k-1$ to another incoming itinerary $t' \neq t$ in q at step k . The following constraints are used to avoid this scenario:

$$\forall q \in Q, \forall u \in U, \forall k \in 2 \dots K, \forall t_1, t_2 \in BS(q), t_1 \neq t_2. x_{u,k-1,t_1} + x_{u,k,t_2} \leq 1 \quad (9)$$

Safety The following constraints are those entailing safety of the computed routes. In particular, the proposed model will compute optimal routes such that no collisions will

ever occur. Moreover, it is ensured that an obstructed itinerary will never be traversed by any vehicle. Note that the absence of deadlocks is entailed by constraints 7.

The constraints below are used to avoid possible collisions among vehicles. Firstly, only one vehicle is allowed in each itinerary t and step k :

$$\forall k \in 1 \dots K, \forall t \in T. \sum_{u \in U} x_{u,k,t} \leq 1 \quad (10)$$

Moreover, in the presence of more vehicles approaching a point $q \in Q$, they cannot be served at the same step k . The constraints below guarantee that at most one vehicle can be served by a point q for each step k .

$$\forall q \in Q, \forall k \in 2 \dots K. \sum_{u \in U} \sum_{t \in BS(q)} x_{u,k-1,t} - 1 \leq \sum_{u \in U} \sum_{t \in BS(q)} x_{u,k-1,t} x_{u,k,t} \leq \sum_{u \in U} \sum_{t \in BS(q)} x_{u,k-1,t} \quad (11)$$

Note that Equation 11 contains a product of two binary variables. Recall that given two binary variables v_1 and v_2 , their product $z = v_1 * v_2$ can be linearised through constraints: $z \leq v_1$; $z \leq v_2$; $z \geq v_1 + v_2 - 1$. For brevity, here we prefer to use this compact version than the linearised one.

In Equation 11, the term $\sum_{u \in U} \sum_{t \in BS(q)} x_{u,k-1,t} x_{u,k,t}$ represents the number of vehicles approaching point q that have not moved between consecutive steps $k-1$ and k . Indeed, vehicles that have approached q at step k but were not present at step $k-1$ are ruled out (their product is zero), as well as those that were approaching q at step $k-1$ and left at step k . This product is used for avoiding vehicles approaching u at step k but not present at step $k-1$. Since at most one vehicle must be served by q between steps $k-1$ and k , $\sum_{u \in U} \sum_{t \in BS(q)} x_{u,k-1,t} x_{u,k,t}$ must be equal to either:

- $\sum_{u \in U} \sum_{t \in BS(q)} x_{u,k-1,t}$, that is no vehicle has moved between steps $k-1$ and k from q , or
- $\sum_{u \in U} \sum_{t \in BS(q)} x_{u,k-1,t} - 1$, in this case only one vehicle has been served by point q between steps $k-1$ and k .

Finally, the last constraint ensures that no failed itinerary is ever traversed by any route computed by the ILP model.

$$\forall t \in F. \sum_{u \in U} \sum_{k \in 1 \dots K} x_{u,k,t} = 0 \quad (12)$$

Graph Structure The constraints below are used to verify that the graph does not contain inner cycles. Note that these constraints are not necessary for solving the routing problem. They are used for preprocessing the user input and can be avoided provided that the input is verified. The equation below could sum up to 2 only if there exists an itinerary $t \in T$ such that $t \in FS(q) \cap BS(q)$, i.e. an inner cycle.

$$\forall q \in Q, \forall u \in U, \forall k \in 1 \dots K, \forall t \in T. \sum_{t \in FS(q)} x_{u,k,t} + \sum_{t \in BS(q)} x_{u,k,t} \leq 1 \quad (13)$$

Cyclic routes are also ruled out in our model. Indeed, a round trip of a vehicle will be split into two separate routes, the first into one direction and the other in the opposite one (note that this assumption is crucial for ensuring $K = |U| * |T|$).

Output Recall that the output of the ILP model will be the set of routes U computed by our procedure. These routes will be communicated to the signalling system. Moreover, the routes of each vehicle u are described in terms of steps k and locations t , such that for each vehicle in correspondence with a variable $u \in U$ its route will be $\forall u \in U. \text{Route}(u) = \{x_{u,k,t} | x_{u,k,t} = 1, k \in 1 \dots K, t \in T\}$, that is, we identify for each step the position of vehicle u .

6 Implementation

In Figure 3 the implementation of the ILP model described in the previous section is displayed. This implementation is open source and can be downloaded at <https://github.com/davidebasile/routingproblem>. The ILP model has been implemented in *A Mathematical Programming Language* (AMPL) [8], a widely used language for describing and solving optimization problems. The model can be loaded and executed in AMPL through command line. In particular, script `routeplanning.run`, to be launched with the command `ampl`, is described below:

```

routeplanning.run
option solver cplex; // use the simplex algorithm in C
model routeplanning.mod; // select the route planning model
data routeplanning.dat; // load the input data
solve; //apply the simplex algorithm
display {i in U, j in K, s in Q, d in Q: x[i,j,s,d]>0} x[i,j,s,d];
//display the computed routes

```

Firstly the solver `cplex` is selected, that is the simplex method implemented in C. However it is possible to select other available solvers. The script loads the automaton from the file `routeplanning.dat`, displayed in Figure 3. The input file provides the number of vehicles u and nodes n , and two binary matrix $Q \times Q$ called t and F . In this implementation edges are represented as pairs of nodes, i.e. source and target nodes of the corresponding edge. The first matrix is used for identifying the graph structure, in particular $t[n_1, n_2] = 1$ if there is an edge connecting node n_1 with node n_2 , $t[n_1, n_2] = 0$ otherwise. Similarly, the second matrix F identifies the unavailable itineraries. Finally, two arrays *location* and *destination* are such that, for example, *location*[u] = n if the location of vehicle u is n .

The implementation file `routeplanning.mod` in Figure 3 follows the model described in Section 5, with few differences detailed in the following. The additional graph constraints $\forall i \in U, j \in K, s \in Q, d \in Q: x[i, j, s, d] \leq t[s, d]$ (lines 12-13) are used to ensure that the flow variables x only use edges of the graph. Indeed, if $t[s, d] = 0$ then the flow $x[i, j, s, d]$ on edge (s, d) is forced to be zero.

Moreover constraints `linearise 1 ... 6` (lines 27-33) are used to linearise the products of Equation 8 and Equation 11. In particular, for Equation 8 it is not possible to specify the condition $\sum_{t \in BS(q)} x_{u,k-1,t} > 0$ directly in AMPL, hence the following constraints (lines 43-47) have been used in the implementation:

$$\forall q \in Q, \forall u \in U, \forall k \in 2 \dots K.$$

$$\sum_{t \in BS(q)} x_{u,k-1,t} - \left(\sum_{t' \in FS(q), t \in BS(q)} x_{u,k,t'} x_{u,k-1,t} + \sum_{t \in BS(q)} x_{u,k,t} x_{u,k-1,t} \right) = 0$$

If $\sum_{t \in BS(q)} x_{u,k-1,t} = 0$ then the above term will sum up to zero. The binary variable $uxu[u, k, s, d]$ (line 6) identifies product $x[u, k, s, d]x[u, k-1, s, d]$ (also used in Equation 11, lines 56-60), while variable $uxu2[u, k, s, q, d]$ (line 6) identifies product $x[u, k, q, d]x[u, k-1, s, q]$.

7 Experiments

In this section we report on preliminary experiments that have been performed for evaluating and validating the proposed model. Similarly to [20] we will use a grid 5x5 as graph to test the ILP model, displayed in Figure 2, which may represent a sub-portion of an urban area. We will assume the presence of four vehicles in the grid. The script `routeplanning.run` has been enriched with the automatic generation of obstructed tracks, locations and destinations of vehicles. These data are randomly generated according to a uniform distribution. Three experiments have been carried on, where in each of them a round of the ILP model has been executed. In Section 9 we discuss future extensions to simulate a whole day, with several obstructions and computations.

The ILP model successfully computed the routes of each experiment. The results are displayed in Table 1. For each experiment the failed tracks are reported, together with location, destination and computed routes of each vehicle. When a vehicle reaches its destination, it is assumed that in the remaining steps the vehicle stays idle. In particular, concerning Experiment 1, locations and destinations of vehicles are the furthest possible and have been inserted manually; the routes of the four vehicles are displayed in Figure 2 with different colours.

In Table 2 for each experiment we report the time, memory consumption and iterations of the simplex algorithm. In particular, the memory consumption is displayed as the cumulative sum of the memory allocated in the different phases of the execution (i.e. compile, genmod, collect, presolve, solve). The performances are similar in all experiments, and are mainly due to the size of the input graph (in terms of number of nodes) and the number of vehicles. It has been used a machine with CPU Intel Core i5-4570 at 3.20 GHZ with 8 GB of RAM, running 64-bit Windows 10 and the CPLEX solver version 12.7.1.0.

8 Related work

The dynamic vehicle routing problem (i.e. finding optimal routes for vehicles with minimum travel time) was firstly introduced by Dantzig and Ramser [4] as a generalization of the Traveling Salesman Problem introduced by Flood [5], and it has been surveyed in [16, 15]. Different solutions have been proposed in the literature, for example by using neural networks [13], dynamic programming [1], mixed integer non-linear programming [2] and random search strategy [12].

Standard vehicle routing problems solutions are not suitable when the conditions of the traffic layout can change dynamically, due for example to traffic congestions, accidents or bad weather conditions. More recently, with the advent of automatic driving systems, as for example autonomous vehicles, the dynamic routing problem has been

```

routeplanning.mod
param n; #points      param k; #discrete steps      param u; #vehicles
set Q := {1..n}; set K := {1..k}; set U := {1..u};
param t{Q,Q}; #itineraries      param location{U}; param destination{U};
param F{Q,Q} binary; #constraint 2      var gamma >= 0 integer;
#constraints 3      var x{U,K,Q,Q} binary;
var uxu{U,K,Q,Q} binary; var uxu2{U,K,Q,Q,Q} binary;

#objective function, equation 1
maximize time: gamma;

# FLOW CONSTRAINTS
#graph constraints: only itineraries can be traversed by vehicles
subject to graph{i in U, j in K, s in Q, d in Q}: x[i,j,s,d] <= t[s,d];

#minimise waiting time
subject to c4: (sum{i in U, s in Q, j in K, d in Q: d == destination[i]} x[i,j,s,d]) >= gamma;

#only one itinerary per time
subject to c5{i in U, j in K}: sum{s in Q, d in Q} x[i,j,s,d] = 1;

#each vehicle starts from its location
subject to c6{i in U}: sum{s in Q, d in Q: s==location[i]} x[i,1,s,d] = 1;

#all vehicles reach their destination eventually
subject to c7{i in U}: sum{s in Q,d in Q: d==destination[i]} x[i,k,s,d] = 1;

#constraints linearise 1,2,3 used to linearise uxu[i,j,s,d] = x[i,j-1,s,d]*x[i,j,s,d]
subject to linearise1{i in U, j in {2..k}, s in Q, d in Q}:
    uxu[i,j,s,d]<= x[i,j-1,s,d];
subject to linearise2{i in U, j in {2..k}, s in Q, d in Q}:
    uxu[i,j,s,d]<= x[i,j,s,d];
subject to linearise3{i in U, j in {2..k}, s in Q, d in Q}:
    uxu[i,j,s,d]>= x[i,j-1,s,d] + x[i,j,s,d] - 1;

#constraints linearise 4,5,6 used to linearise uxu2[i,j,s,q,d] = x[i,j-1,s,q]*x[i,j,q,d]
subject to linearise4{i in U, j in {2..k}, s in Q, q in Q, d in Q}:
    uxu2[i,j,s,q,d]<= x[i,j-1,s,q];
subject to linearise5{i in U, j in {2..k}, s in Q, q in Q, d in Q}:
    uxu2[i,j,s,q,d]<= x[i,j,q,d];
subject to linearise6{i in U, j in {2..k}, s in Q,q in Q, d in Q}:
    uxu2[i,j,s,q,d]>= x[i,j-1,s,q] + x[i,j,q,d] - 1;

#flow constraints, for each k each vehicle i stays idle or move into an adjacent itinerary
subject to c8{i in U, q in Q, j in {2..k}}:
    (sum{s in Q:s!=q} x[i,j-1,s,q]) -
    (sum{s in Q, d in Q:d!=q && s!=q} uxu2[i,j,s,q,d] + sum{s in Q:s!=q} uxu[i,j,s,q]) = 0;

#complement previous constraints: vehicles do not "jump" itineraries
subject to c9{i in U, q in Q, j in {2..k}, s1 in Q, s2 in Q: s1!=s2}:
    x[i,j-1,s1,q]+x[i,j,s2,q] <= 1;

# SAFETY
#no collisions on itineraries
subject to c10{j in K, s in Q, d in Q}: sum{i in U} x[i,j,s,d] <= 1;

#no collisions on points
subject to c11_1{q in Q, j in {2..k}}:
    sum{i in U, s in Q} x[i,j-1,s,q] - 1 <= sum{i in U, s in Q} uxu[i,j,s,q];
subject to c11_2{q in Q, j in {2..k}}:
    sum{i in U, s in Q} uxu[i,j,s,q] <= sum{i in U, s in Q} x[i,j-1,s,q];

#routes must not pass through damaged itineraries
subject to c12{s in Q, d in Q: F[s,d]==1}: sum{i in U,j in K} x[i,j,s,d]=0;

```

Fig. 3: The implementation in AMPL of the dynamic routing optimization problem.

	Experiment 1
Obstructed tracks	(2,7) (6,7) (9,8) (11,6) (13,14) (17,12) (19,18) (22,23) (23,24)
Vehicle 1	location=1, destination=24 route=(1,2)(2,3)(3,8)(8,9)(9,14)(14,15)(15,20)(20,25)
Vehicle 2	location=25, destination=1 route=(25,20)(20,15)(15,10)(10,5)(5,4)(4,3)(3,2)(2,1)
Vehicle 3	location=5, destination= 21 route=(5,10)(10,9)(9,14)(14,13)(13,12)(12,17)(17,16)(16,21)
Vehicle 4	location=21, destination=5 route=(21,16)(16,11)(11,12)(12,7)(7,8)(8,9)(9,4)(4,5)
	Experiment 2
Obstructed tracks	(1,6) (2,1) (2,7) (6,1) (6,7) (7,2) (9,8) (10,2) (10,15) (23,22)
Vehicle 1	location=14, destination=11, route=(14,13)(13,12)(12,11)
Vehicle 2	location=8, destination=10, route=(8,9)(9,10)
Vehicle 3	location=13, destination=18, route=(13,18)
Vehicle 4	location=21, destination=12, route=(21,16)(16,11)(11,12)
	Experiment 3
Obstructed tracks	(1,2) (1,6) (2,1) (2,7) (6,1) (7,2) (10,2)
Vehicle 1	location=15, destination=12, route=(15,14)(14,13)(13,12)
Vehicle 2	location=9, destination=11, route=(9,8)(8,7)(7,6)(6,11)
Vehicle 3	location=14, destination=19, route=(14,19)
Vehicle 4	location=22, destination=13, route=(22,17)(17,12)(12,13)

Table 1: For each experiment the computed routes are displayed, together with obstructed tracks, location and destination of each vehicle.

revived. Modern technology, for example global positioning system and geographic information systems, can be used to collect dynamically the traffic situation to allow the dynamic assignment of routes to vehicle, as proposed by our methodology.

The problem of vehicle routing in urban traffic network is discussed in [20]. A notion of critical node is used to identify the current position of vehicles in the network, and each vehicle has associated a set of customers that must be visited in minimal time. The route of each vehicle is computed locally. An approximate initial starting solution is computed through a genetic algorithm. By dividing the flow variables into discrete steps we are able to identify the current location of each vehicle in the urban network, instead of using special nodes that would augment the state space of the problem. Moreover, our approach does not need to generate an initial solution. Indeed, once a vehicle reaches a destination, its new destination will be updated and the new routes will be recomputed. This is mainly due to the presence of possible faulty events in the tracks (e.g. obstructions), a condition not addressed in [20]. The routing solution is generated globally by considering routes of each vehicle in the network.

The problem of computing the train scheduling and routing in combination is addressed in [18]. A multi-objective function constituted by the minimum average travel time of all trains, the minimum energy consumption and the minimum delayed times is

Experiment	Time (seconds)	Cumulative Allocated Memory (byte)	MIP Simplex Iterations
1	49.5	13315294784	5532
2	48.9531	13315330472	5438
3	44.9531	13315215272	6127

Table 2: Performances of experiments.

used. The train scheduling problem is solved through a simulation algorithm according to the train control strategies, and a genetic algorithm is used in case of large scale networks for the train routing problem. Compared to our work, in [18] faulty events are not considered, whilst we only focus on the vehicle routing problem and we abstract away from details of time schedule. Indeed, our ILP model should be executed to restore the system to a working state when obstructions in the tracks are detected, until normal operation conditions are established.

The routing problem for freight trains is studied in [3]. Similarly to our approach, the minimum time in terms of vehicles reaching their destination is computed globally, by taking into account all routes of vehicles. Moreover, the layout structure of the rail road track is abstracted as a graph, and the day partitioned into time steps. A fixed number of vehicles is allowed to enter a particular track segment throughout the whole day. An important difference with respect to our approach is that routes are statically assigned to vehicles and are not adapted dynamically, i.e. faults in tracks are not considered. We also enforce safety properties for avoiding possible collisions among vehicles.

In [14] an Automatic Train Supervision for preventing the occurrence of deadlocks in train routes is studied. Similarly to our solution, the track layout is abstracted as a graph. However, the proposed solution does not account for possible failures in tracks (i.e. dynamic vehicle routes). Indeed, each route of a train is fixed and it is an input parameter. Trains decide whether to move at a given discrete steps autonomously and according to their routes, whilst we dictate when vehicles move (i.e. steps). The algorithm takes in input also the graph layout and a set of areas (i.e. nodes in the graph) where only a given number of trains are allowed to enter. The absence of possible deadlocks is verified through model checking given the aforementioned data.

We conjecture that our model can be extended with minor changes to solve the deadlock problem. Indeed, it suffices to add to our model additional constraints to only allow a fixed number of vehicles to enter a predetermined area (modelled as set of transitions), and to fix a specific route to each vehicle as an input parameter. Moreover by using a bi-level objective function $\min \max$ it is possible to determine if a configuration of routes into discrete steps exists such that vehicles are deadlocked.

9 Conclusion and Future Work

We presented a dependable dynamic vehicle routing system, focussing on the ILP model for computing new routes of vehicles given their actual location, destination and detected obstructed tracks. Similarly to [20, 14, 3], we abstracted the urban map as a graph such that edges and nodes are in correspondence, respectively, with itineraries

and points of the urban area. The algorithm has been modelled as a flow problem, where each flow corresponds to a vehicle route. The newly computed routes are equipped with safety guarantees on the absence of deadlocks and possible collisions among vehicles, both in points and itineraries. The proposed solution has been implemented in *A Mathematical Programming Language* [8](AMPL) and preliminary experiments have been carried, on showing the effectiveness of the proposed solution; the implementation and all data are available at <https://github.com/davidebasile/routingproblem>.

Some possible future extensions of the proposed approach are discussed below. Whilst preliminary experiments showed the feasibility of our approach, we would like to apply the proposed solution to a real world urban scenario. Moreover, it would be valuable to extend the proposed model to include also aspects related to performances of vehicles (i.e. acceleration, speed, braking) and energy consumption (fuel, other energy dissipation). Indeed, it is possible to associate to each edge of the graph (i.e. itinerary) also a pair of cost and time for traversing the itinerary, which are inversely proportional. Different strategies could be adopted for synthesising the routes of vehicles, for example by minimising either the cost or time, or a linear combination of both. Concerning the experiments, we would like to include the proposed ILP model into a framework for simulating possible failures of tracks, to evaluate the ILP model in the presence of different conditions randomly generated and throughout a whole day. It would be then possible to measure the energy consumption and user satisfiability adopting different strategies for computing the routes, to select the best one.

Acknowledgements This work has been partially supported by the Tuscany Region project POR FESR 2014-2020 SISTER and H2020 2017-2019 S2R-OC-IP2-01-2017 ASTRail.

References

1. Assad, A.: Analysis of rail classification policies. *INFOR: Information Systems and Operational Research* 21(4), 293–314 (1983), <http://dx.doi.org/10.1080/03155986.1983.11731905>
2. Bodin, L.D., Golden, B.L., Schuster, A.D., Romig, W.: A model for the blocking of trains. *Transportation Research Part B: Methodological* 14(1), 115 – 120 (1980), <http://www.sciencedirect.com/science/article/pii/0191261580900375>
3. Borndörfer, R., Klug, T., Schlechte, T., Fügenschuh, A., Schang, T., Schüllndorf, H.: The freight train routing problem for congested railway networks with mixed traffic. *Transportation Science* 50(2), 408–423 (2016)
4. Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. *Management Science* 6, 80–91 (10 1959)
5. Flood, M.M.: The traveling-salesman problem. *Operations Research* 4(1), 61–75 (1956)
6. Ford, D.R., Fulkerson, D.R.: *Flows in Networks*. Princeton University Press, Princeton, NJ, USA (2010)
7. Ford, L.R., Fulkerson, D.R.: A simple algorithm for finding maximal network flows and an application to the hitchcock problem. *Canadian Journal Of Mathematics* pp. 210–218 (1957)
8. Fourer, R., Gay, D.M., Kernighan, B.W.: *AMPL: A mathematical programming language*. AT&T Bell Laboratories Murray Hill, NJ 07974 (1987)

9. Ghiani, G., Guerriero, F., Laporte, G., Musmanno, R.: Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *European Journal of Operational Research* 151(1), 1–11 (2003)
10. Hemmecke, R., Koppe, M., Lee, J., Weismantel, R.: Nonlinear integer programming. In: Junger, M., Liebling, T.M., Naddef, D., Nemhauser, G.L., Pulleyblank, W.R., Reinelt, G., Rinaldi, G., Wolsey, L.A. (eds.) *50 Years of Integer Programming 1958–2008*, pp. 561–618. Springer Berlin Heidelberg (2010)
11. Klein, M.: A primal method for minimal cost flows, with applications to the assignment and transportation problems (1967)
12. Li, F., Gao, Z., Li, K., Yang, L.: Efficient scheduling of railway traffic based on global information of train. *Transportation Research Part B: Methodological* 42(10), 1008 – 1030 (2008), <http://www.sciencedirect.com/science/article/pii/S0191261508000337>
13. Martinelli, D.R., Teng, H.: Optimization of railway operations using neural networks. *Transportation Research Part C: Emerging Technologies* 4(1), 33 – 49 (1996), <http://www.sciencedirect.com/science/article/pii/0968090X9500019F>
14. Mazzanti, F., Ferrari, A., Spagnolo, G.O.: Experiments in formal modelling of a deadlock avoidance algorithm for a CBTC system. In: *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications - 7th International Symposium, ISoLA 2016, Proceedings, Part II*. pp. 297–314 (2016)
15. Pillac, V., Gendreau, M., Gu  ret, C., Medaglia, A.L.: A review of dynamic vehicle routing problems. *European Journal of Operational Research* 225(1), 1–11 (2013)
16. Psaraftis, H.N., Wen, M., Kontovas, C.A.: Dynamic vehicle routing problems: Three decades and counting. *Netw.* 67(1), 3–31 (Jan 2016)
17. Schoitsch, E.: Introduction to the special theme - autonomous vehicles. *ERCIM News* 2017(109) (2017)
18. Sun, Y., Cao, C., Wu, C.: Multi-objective optimization of train routing problem combined with train scheduling on a high-speed railway network. *Transportation Research Part C: Emerging Technologies* 44, 1 – 20 (2014), <http://www.sciencedirect.com/science/article/pii/S0968090X14000655>
19. Wallace, S.W. (ed.): *Algorithms and Model Formulations in Mathematical Programming*. Springer-Verlag New York, Inc., New York, NY, USA (1989)
20. Yanfeng, L., Ziyu, G., Jun, L.: Vehicle routing problem in dynamic urban traffic network. In: *ICSSSM11*. pp. 1–6 (June 2011)