Computers & Operations Research 166 (2024) 106625

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

Real-time schedule adjustments for conflict-free vehicle routing

Tommaso Adamo, Gianpaolo Ghiani [*], Emanuela Guerriero

Dipartimento di Ingegneria dell'Innovazione, Università del Salento, Via per Monteroni, 73100 Lecce, Italy

ARTICLE INFO

Keywords:
Conflict-free vehicle routing and scheduling
Real-time optimization

ABSTRACT

Conflict-Free Vehicle Routing Problems (CFVRPs) arise in manufacturing, transportation and logistics applications where *Automated Guided Vehicles* (AGVs) are utilized to move pallets and containers. A peculiar feature of these problems is that collision avoidance among vehicles must be considered explicitly. To make things more complex, the uncertainty affecting both travel times and machine ready times often results in vehicle delays or anticipations that require real-time modifications to the fleet nominal plan. In this paper, the determination of such modifications (schedule adjustment problem in CFVRPs) is modeled as a sequential decision problem for which we develop a tailored fast exact algorithm suitable for any objective function that is non-decreasing in the arrival times. Computational results show that optimal solutions can be found within at most 3.3 milliseconds for instances with up to 300 vehicles with improvements of various performance measures up to 74% compared to state-of-the-art solution algorithms.

1. Introduction

Using Automated Guided Vehicles (AGVs) to move pallets and containers can considerably improve productivity and reduce costs in warehouses, production plants and port terminals. As stated in the latest report of Grand View Research (2022), the market for automated guided vehicles (AGVs) is growing rapidly and was valued USD 3.81 billion in 2021. Experts predict that this trend will continue with a projected annual growth rate of 10.2% from 2022 to 2030.

AGVs navigate through a network of configurable guide paths, where the limited arc capacity must be taken into account explicitly to prevent collisions among vehicles. This leads to a relevant vehicle routing problem known as the Conflict-Free Vehicle Routing Problem (CFVRP).

We now revise some of the most relevant contributions to the solution of the CFVRP. For a more detailed analysis of the literature, the reader is referred to Vis (2006) and Fragapane et al. (2021).

Kim and Tanchoco (1991) present a constructive heuristic that generates a solution in a sequential fashion. At each iteration, the procedure maintains, for each node of the flow path network, a list of time windows reserved by scheduled vehicles and a list of free time windows available for vehicles to be scheduled. Then the algorithm routes the vehicles through the free time windows by solving a *shortest path problem with time windows* (Desrosiers et al., 1983). (Krishnamurthy et al., 1993) propose a column generation approach to solve a problem arising in a Flexible Manufacturing System. The master problem implemented makespan and conflict-free constraints, while subproblems were constrained shortest path problems with time-dependent costs on the edges. Subsequently, Desaulniers et al. (2003) extend the problem by considering the assignment of requests to vehicles; they provide an exact method based on column generation embedded in a branch-and-cut scheme to dispatch and route AGVs while minimizing production delays. The problem of assigning, scheduling and routing vehicles simultaneously is solved by Corréa et al. (2007) through a decomposition method. They combine constraint programming and mixed integer programming (MIP) and use logic cuts to eliminate conflicting solutions. (Gawrilow et al., 2008) compare conflict-free routes dynamically computed using implicit time-expanded networks with an online static route computation procedure. The former proves to be more successful on graphs with high density, but very sensitive to delays or disruptions. In contrast, in the latter approach the computed routes do not change and are thus more robust against disturbances. Miyamoto and Inoue (2016) present an integer programming model, a local search and a random search for vehicle dispatching and conflict-free routing. Adamo et al. (2018) develop a branch-and-bound algorithm to determine energy efficient solutions in terms of routes and speeds. Their approach determines lower bounds by solving nonlinear problems in quadratic time and generates branching constraints to avoid conflicts. (Murakami, 2020) uses a time–space network and formulate the problem as a MIP problem.

In the context of planning the operations in an automated container terminal, one has to consider the integrated scheduling of handling equipment (quay cranes, yard cranes, etc.) and AGV conflict-free routing. For this problem (Zhong et al., 2020) develop a MIP model to

[*] Corresponding author.
E-mail addresses: tommaso.adamo@unisalento.it (T. Adamo), gianpaolo.ghiani@unisalento.it (G. Ghiani), emanuela.guerriero@unisalento.it (E. Guerriero).

https://doi.org/10.1016/j.cor.2024.106625
Received 8 August 2023; Received in revised form 8 March 2024; Accepted 14 March 2024
Available online 19 March 2024

minimize total delay for a given task allocation. For another relevant contribution see Cao et al. (2023) which also contains an in-depth analysis of the literature.

Another line of research studies the dynamic version of the CFVRP in which requests arrive in a random fashion. Recent remarkable contributions in this area are: (Hwang and Jang, 2020) that develop a reinforcement learning approach for the problem of conflict-free routing of overhead hoist transport vehicles in semiconductor fabrication facilities; Chen et al. (2022) that integrate ant colony optimization into a multi-agent system.

In this paper, we present a fast exact algorithm for the *schedule adjustment problem* in conflict-free vehicle routing problems (SA-CFVRP). The motivation of the study is as follows: in AGV systems, unpredictable AGV delays and anticipations may be caused by a number of random events, such as processing times greater than expected at numerically controlled machines, AGV low battery levels, vehicles slowing down to avoid human operators cutting off their road, etc. As a consequence, the fleet nominal plan can be become suboptimal or even unfeasible (i.e., conflicts among vehicles may arise). We assume that AGVs are under the control of a centralized system that monitors vehicle positions at a high rate (e.g., once every 300 ms). A peculiar feature of the problem is that vehicles move very fast (at speeds in the order of one/two meters per second) which requires SA-CFVRP algorithms to recover feasibility in few milliseconds.

To the best of our knowledge, the only contribution to the SA-CFVRP is Adamo et al. (2023) in which corrective actions were prescribed *only at the current positions* of the AGVs in the nominal plan. In this paper we go one step further. First, we model the problem as a deterministic sequential decision process for which we devise a preprocessing phase to reduce problem complexity. Second, we develop an algorithm to adjust the nominal plan *at multiple time points*, which enlarges the solution space and allows to achieve better performances. In order to keep the computational burden low enough for real-time applications, the prescribed corrective actions do not include modifications to the paths followed by vehicles but only the insertion of waiting times along the routes of the nominal plan. This is done while preserving the precedence relationships among the nominal arrival times at each single vertex visited by more than one vehicle. More specifically, the main contributions of this work can be summarized as follows.

- We introduce and motivate the SA-CFVRP as a deterministic sequential decision problem. We propose a *policy* that maps the *state* (i.e. what positions the vehicles are at) to an *action* (i.e. vehicle waiting times). We show that the proposed policy is based on the solution of a linear program and prove that its solution is optimal with respect to any objective function that is a non-decreasing function of arrival times.
- We prove that the SA-CFVRP can be reformulated as a one-to-all shortest paths problem on an auxiliary graph. In addition, we propose two polynomial-time algorithms that aim to reduce the instance size through a pre-processing step to be executed just once before the first stage, with the goal of improving computational efficiency and scalability.
- We conduct a computational study to assess the performance of the proposed approach and determine the maximum problem instance size that can be solved in a real-time setting. The computational results showed that the proposed algorithm is able to find the optimal solutions of instances with up to 300 vehicles within at most 3.3 milliseconds.

The rest of this paper is organized as follows. The next section formally defines the problem and introduces the notation used throughout the paper. In Section 3 we propose a policy based on the solution of a linear program and discuss its properties. In Section 4 we show that each base problem of SA-CFVRP is equivalent to a one-to-all-pairs shortest paths problem defined on an auxiliary graph. In Section 5 we present two polynomial-time algorithms for reducing the size of the auxiliary graph. Finally, Section 6 presents some computational results, followed by the conclusions in Section 7.

2. Problem definition and notation

Let $G = (V, A, \tau)$ be a directed graph representing the internal transportation network of a material handling system served by a fleet K of identical vehicles. The AGVs move along a guide path consisting of unit-capacity segments that can intersect only at their endpoints. The vertex set V represents the endpoints of such segments, where other components of the material handling system (such as loading/unloading stations and storage positions) might be located. Given a segment and its endpoints $i \in V$ and $j \in V$, both arcs (i, j) and (j, i) belong to A, i.e. we assume that each segment is two-way. Function $\tau : A \rightarrow \mathbb{N}$ returns the *nominal* traversal time of any arc $(i, j) \in A$. Without any loss of generality, we made the following assumptions: traversal times are discrete, and vehicles are allowed to wait only at the end of the segments. For the sake of notational convenience the vehicle shape is ignored. In the following we shall discuss how notation can be extended in a quite natural way to a realistic setting where a single vehicle might occupy simultaneously more than one segment.

Notation for the initial nominal conflict free plan. We assume that *an initial nominal plan* has been previously determined as a feasible solution of a conflict free vehicle routing problem defined on G. The individual vehicle plan prescribes a route and a schedule, including possible waiting times at some vertices of V. The initial nominal plan is conflict-free and hopefully optimizes some performance measure. This implies that it satisfies a set of non-overlapping constraints, modeling the unit-capacity of both vertices and arcs. In particular, two types of conflicts may occur: a *vertex-conflict*, which happens when multiple vehicles on different routes are headed towards the same vertex and arrive at the same time, and an *arc-conflict*, which happens when a vehicle tries entering a track segment already occupied by another vehicle (in the same or in opposite direction).

For each vehicle $h \in K$, the nominal plan prescribes a sequence S_h consisting of p_h locations to be visited by the vehicle. In particular, we denote with s_{ih} the ith vertex visited by the vehicle h, with $i = 1, \ldots, p_h$ and $h \in K$. For notational convenience, we use the symbol τ_{hi} to denote the traversal time of the arc from vertex s_{hi} to vertex $s_{h,i+1}$, i.e. $\tau_{hi} := \tau(s_{hi}, s_{h,i+1})$ with $i = 1, \ldots, p_h - 1$ and $h \in K$. The *nominal* values of the arrival time and waiting time of vehicle h at vertex s_{hi} are respectively denoted with $\bar{\mathrm{t}}_{hi}$ and $\bar{w}_{hi}$, for $i = 1, \ldots, p_h$ and $h \in K$.

An illustrative example. In Fig. 1 we provide an example of a nominal plan for three vehicles (denoted as v_0, v_1 and v_2) moving on the physical graph showed in Fig. 2(a), where arcs represent paths followed by vehicles, with $S_{\mathrm{v}_0} = [0, 1, 2, 5, 6, 7, 6, 9, 8]$, $S_{\mathrm{v}_1} = [12, 11, 9, 6, 5, 2, 3]$ and $S_{\mathrm{v}_2} = [10, 11, 9, 6, 5, 4]$.

Conjunctive relationships. Given a vehicle $h \in K$, the corresponding nominal values of the arrival and waiting times satisfy the following condition:

$$\bar{\mathrm{t}}_{h,i+1} = \bar{\mathrm{t}}_{hi} + \bar{w}_{hi} + \tau_{hi}, \tag{1}$$

with $i = 1, \ldots, p_h - 1$. In the following we refer to (1) as a *conjunctive* precedence relationship between the values of nominal arrival times at vertices s_{hi} and $s_{h,i+1}$.

Disjunctive precedence relationships. Since the initial nominal plan is conflict-free, the nominal values of arrival times and waiting times also induce a set of *disjunctive* precedence relationships for each vertex visited by more than one vehicle. Each disjunctive precedence relationship refers to the avoidance of either a *node-conflict* or an *arc-conflict*. As far as *node-conflict* is concerned, the *disjunctive* precedence relationship (2) holds between the nominal values of arrival times $\bar{\mathrm{t}}_{hi}$ and $\bar{\mathrm{t}}_{kj}$ of two

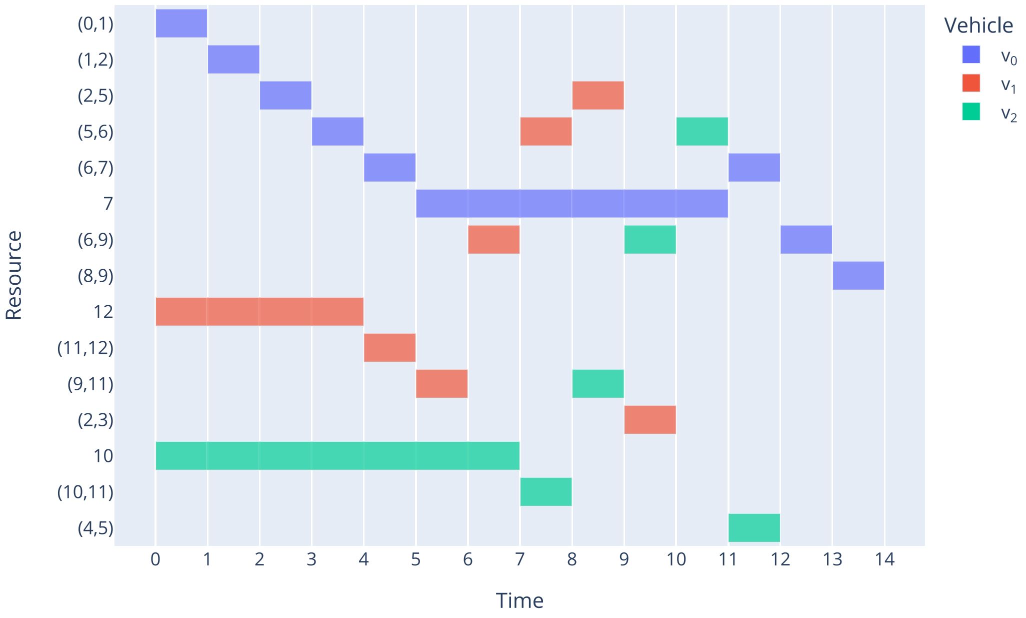


Fig. 1. The Gantt chart of the nominal plan.

distinct vehicles arriving at the same vertex (i.e., $s_{hi} = s_{kj}$) with vehicle k preceding vehicle h, that is:

$$\bar{t}_{hi} \geq \bar{t}_{kj} + \bar{w}_{kj} + 1, \tag{2}$$

with $h, k \in K$, $k \neq h$, $i = 1, \dots, p_h$ and $j = 1, \dots, p_k$. Regarding the *disjunctive* precedence relationships corresponding to arc-conflicts, for each segment with endpoints s_{hi} and $s_{h,i+1}$, the *disjunctive* precedence relationship (2) holds between the nominal values $\bar{t}_{hi}$ and $\bar{t}_{kj}$ of two distinct vehicles traversing the same segment with vehicle $k \in K$ entering the segment at time $\bar{t}_{k,j-1}$, earlier than vehicle h (i.e. $\bar{t}_{k,j-1} < \bar{t}_{hi}$), traversing it in the same direction (i.e. $s_{hi} = s_{k,j-1} \ \wedge \ s_{h,i+1} = s_{k,j}$) or in the opposite one (i.e. $s_{hi} = s_{k,j} \ \wedge s_{h,i+1} = s_{k,j-1}$).

The temporal network. Conjunctive/disjunctive precedence relationships of the initial nominal plan can be represented by a temporal network $\mathcal{N}_{\mathcal{T}} = (\mathcal{V}_{\mathcal{T}}, \mathcal{A}_{\mathcal{T}})$, where each node in $\mathcal{V}_{\mathcal{T}}$ corresponds to a time point $\bar{t}_{hi}$ with two types of outgoing arcs, with $h \in K$ and $i = 1, \dots, p_h - 1$. The former refers to conjunctive relationship (1) and connects time point $\bar{t}_{hi}$ with time point $\bar{t}_{hi+1}$. The latter refers to the disjunctive relationship (2) with one arc connecting time point $\bar{t}_{kj}$ to time point $\bar{t}_{hi}$, if $\bar{t}_{hi} > \bar{t}_{kj}$ and one of the following conditions are satisfied:

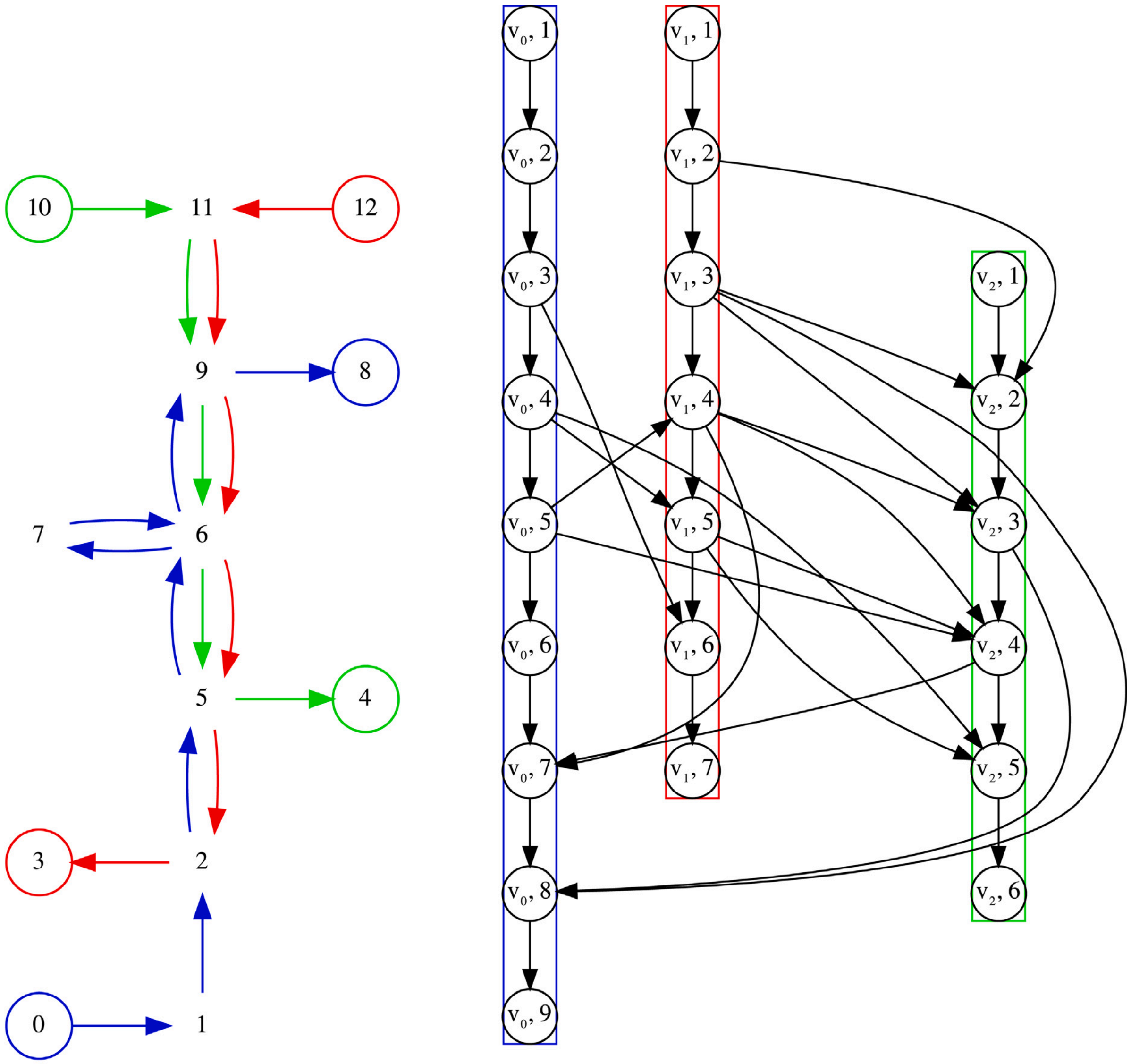
- $s_{hi} = s_{k,j-1} \ \wedge \ s_{h,i+1} = s_{k,j}$, i.e. vehicles h and k traverse the same segment in the same direction;
- $s_{hi} = s_{k,j} \ \wedge s_{h,i+1} = s_{k,j-1}$, i.e. vehicles h and k traverse the same segment in opposite directions;
- $s_{hi} = s_{kj} \wedge s_{hi+1} \neq s_{k,j+1}$, i.e. vehicles h and k traverse two distinct intersecting segments;

with $k \in K$, $j = 1, \dots, p_k - 1$ and $h \neq k$. For the sake of notational convenience, we label each node of the temporal network with the corresponding pair (h, i) and denote with γ_h^i the set of starting nodes of disjunctive arcs ingoing to the node (h, i), with $h \in K$ and $i = 1, \dots, p_h$. The temporal network of the illustrative example is reported in Fig. 2(b). In particular, we cluster the nodes on the basis of the vehicle which they refer to. It is worth noting that each node (h, i) in the temporal network (for $h \in K, i = 1, \dots, p_h$) corresponds to a vertex s_{hi}, i.e. the physical vertex which the time point $\bar{t}_{hi}$ refers to. In the example of the temporal network of Fig. 2(b), vehicles v_1 and v_2 traverse segments $(6, 9)$ and $(5, 6)$ in the same direction, with vehicle v_1 preceding vehicle v_2. The disjunctive arc $((k, j), (h, i)) = ((\mathrm{v}_1, 4), (\mathrm{v}_2, 3))$ of Fig. 2(b) states that the vehicle v_2 can enter segment $(6, 9)$ only when vehicle v_1 leaves vertex 6, i.e. $s_{\mathrm{v}_2,3} = s_{\mathrm{v}_1,3} = 9 \ \wedge \ s_{\mathrm{v}_2,4} = s_{\mathrm{v}_1,4} = 6$. Similarly the disjunctive arc $((\mathrm{v}_1, 5), (\mathrm{v}_2, 4))$ states that vehicle v_2 can enter segment $(5, 6)$ only when vehicle v_1 leaves vertex 5, i.e. $s_{\mathrm{v}_2,4} = s_{\mathrm{v}_1,4} = 6 \ \wedge \ s_{\mathrm{v}_2,5} = s_{\mathrm{v}_1,5} = 5$.

Taking into account vehicles shape. More generally, each disjunctive precedence relationship arising in the initial nominal plan is representative of arrivals of two distinct vehicles either at the same vertex or at adjacent vertices, i.e., endpoints of a segment. Such precedence relationships represent a set of collision-avoidance rules that have guided the search for the *initial* conflict-free nominal plan. In particular, inequalities (2) state that the initial nominal values of both arrival times and waiting times have been determined so that a vehicle may only leave the vertex it currently occupies if it has claimed the next vertex on its path. After a segment is left by a vehicle, its endpoints are released and only then the segment can be claimed by another vehicle.

We can generalize the previous mechanism to model that the initial nominal plan has been generated by taking into account the shape of the vehicles. In this case the path followed by a single vehicle corresponds to a sequence of adjacent non-overlapping zones visited by a single vehicle. Moreover, S_h represents the sequence of adjacent vertices visited by a reference point of the vehicle h (typically its center of gravity), with the nominal plan prescribing for each zone a sequence of vehicles visiting it. The nominal plan of Fig. 3 reports the schedule associated to the *reference points* of two vehicles traversing two paths on the physical network of Fig. 4(a). Even though the paths followed by the reference points do not intersect at any vertex, the temporal network has to include a set of disjunctive arcs representing the avoidance of *zone-conflicts* depicted in Fig. 5. The aim is to model that each reference point is the center of a circle area occupied by the vehicle, with ray value equal to 1 unit (i.e. the length of a single segment in the considered example). Given the two reference point paths, a nominal plan is conflict-free if, at any time instant, the distance between the reference points is (strictly) greater than the diameter of vehicle shape. This (sufficient) *conflict-free* condition is satisfied as follows: when the vehicle v_0 arrives at node $s_{\mathrm{v}_0,2} = 5$, it claims a portion of the path of vehicle v_1 associated to vertices $s_{\mathrm{v}_1,3} = 9$, $s_{\mathrm{v}_1,4} = 6$, $s_{\mathrm{v}_1,5} = 7$. For the nominal plan of Fig. 3, such *zone-conflict avoidance* condition is encoded by the disjunctive relationships (2) associated to arcs $((\mathrm{v}_0, 2), (\mathrm{v}_1, 3))$, $((\mathrm{v}_0, 2), (\mathrm{v}_1, 4))$, $((\mathrm{v}_0, 2), (\mathrm{v}_1, 5))$ of Fig. 4(b), that is:

$$\bar{t}_{\mathrm{v}_1,3} \geq \bar{t}_{\mathrm{v}_0,2} + \bar{w}_{\mathrm{v}_0,2} + 1,$$



(a) Vehicles routes on the physical network

(b) Temporal network

Fig. 2. Routes and temporal network of the nominal plan reported in Fig. 1.

$$\bar{\mathrm{t}}_{\mathrm{v}_1,4} \geq \bar{\mathrm{t}}_{\mathrm{v}_0,2} + \bar{w}_{\mathrm{v}_0,2} + 1,$$

$$\bar{\mathrm{t}}_{\mathrm{v}_1,5} \geq \bar{\mathrm{t}}_{\mathrm{v}_0,2} + \bar{w}_{\mathrm{v}_0,2} + 1.$$

According to the Gantt chart of Fig. 3, the nominal arrival times of vehicle v_0 and v_1 satisfy (2), i.e. $\bar{\mathrm{t}}_{\mathrm{v}_0,2} = 1$, $\bar{w}_{\mathrm{v}_0,2} = 0$, $\bar{\mathrm{t}}_{\mathrm{v}_1,3} = 2$, $\bar{\mathrm{t}}_{\mathrm{v}_1,4} = 3$ and $\bar{\mathrm{t}}_{\mathrm{v}_1,5} = 4$. We assume that relationships among non-adjacent vertices are taken into account as zone-conflict avoidance rules when the initial nominal plan is determined with the corresponding *zone-disjunctive* arcs included in the temporal network.

Modeling SA-CFVRP as a deterministic sequential decision problem. We assume that at a given time instant $\mathrm{t}^{(0)}$ an *initial* nominal plan is determined as a feasible solution of a conflict-free vehicle routing problem defined on G. The SA-CFVRP is then modeled and solved as a sequential decision problem by a centralized system, which is responsible for monitoring the positions of the vehicles. We assume that travel time τ_{hi} is just an average of prior observations and, therefore, is captured in the initial stage $t^{(0)}$, with $(h, i) \in \mathcal{V}_{\mathcal{T}}$. At each decision stage, $\mathrm{t}^{(q)}$ a new nominal plan is generated, with $q \geq 1$ and $\mathrm{t}^{(q)} > \mathrm{t}^{(q-1)}$. The corresponding arrival and waiting times $\bar{\mathrm{t}}^{(q)}_{hi}$ and $\bar{w}^{(q)}_{hi}$ (i.e. the *decision variables*) are provided as input to the next decision stage and referred to as the *current nominal plan* at time instant $\mathrm{t}^{(q+1)}$, with $q > 1$, $h \in K$ and $i = 1, \dots, p_h$. At decision stage $\mathrm{t}^{(q)}$ the centralized system validates the *current* nominal plan determined at the previous stage and (possibly) prescribes *corrective actions*. It is worth noting that our approach is *data-driven*, meaning that we do not explicitly model travel time uncertainty but rather rely on observed data (Powell, 2021). To this aim, we denote with $\xi_h(\mathrm{t}^{(q)})$ and $\chi_h(\mathrm{t}^{(q)})$ the planned and the observed position of vehicle $h \in K$ in G at time $\mathrm{t}^{(q)}$, respectively. For each vehicle $h \in K$, we denote with $d^{(q)}_h$ the deviation at time $\mathrm{t}^{(q)}$ from the nominal plan prescribed at previous stage, that is

$$d^{(q)}_h = \mathrm{t} - \xi^{-1}_h(\chi_h(\mathrm{t}^{(q)})),$$

with $q \geq 1$ and $\mathrm{t}^{(q)} > \mathrm{t}^{(q-1)}$. A positive deviation means that the vehicle is delayed with respect to the current nominal plan; in contrast, an early vehicle leads to a negative deviation. Finally, an on-time vehicle has null deviation.

Positive (or negative) deviations might *disrupt* the feasibility of the current nominal plan. In order to avoid *collisions* among vehicles, a set of corrective actions has to be prescribed in real-time. With the

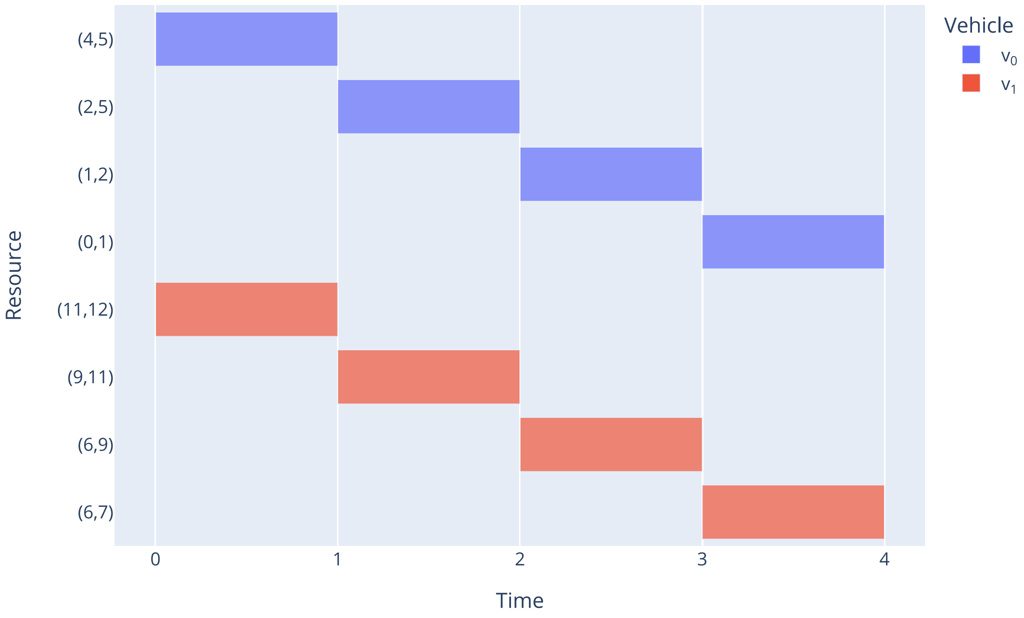
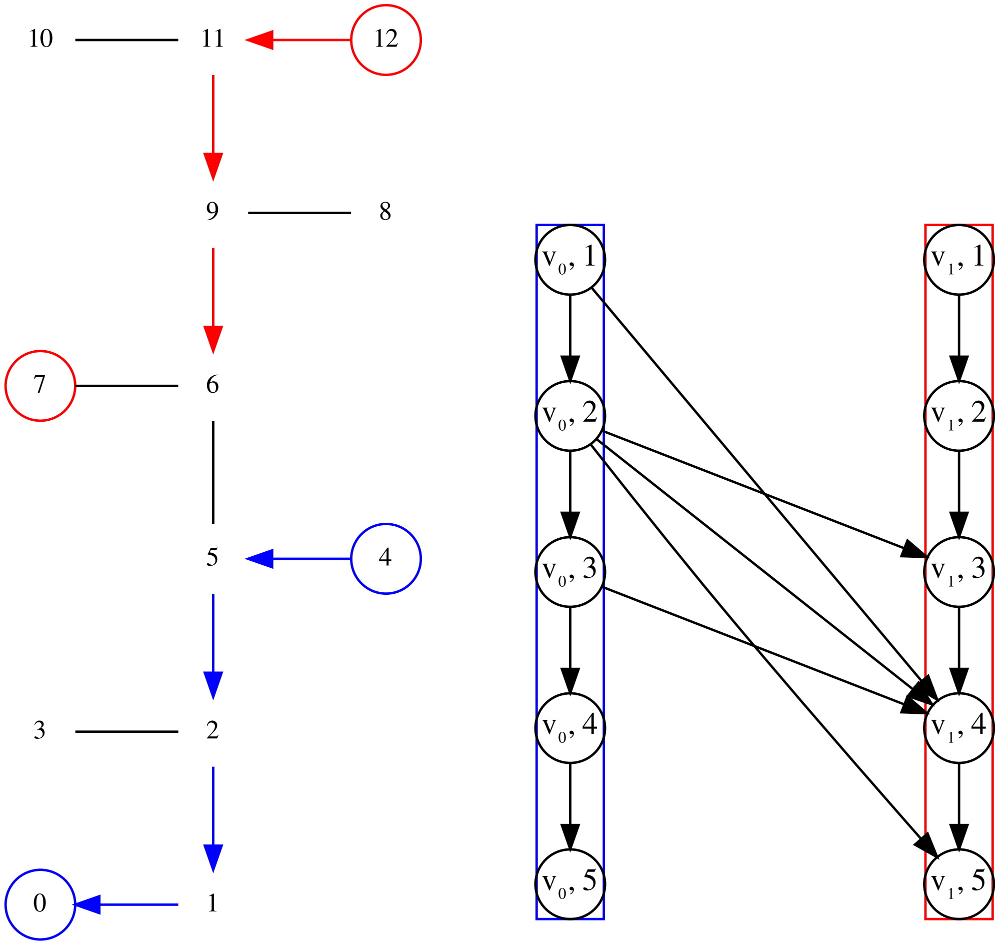


Fig. 3. The Gantt chart of a nominal plan with zone disjunctive precedence constraints.



(a) Vehicles routes on the physical network

(b) Temporal network

Fig. 4. Routes and temporal network of the nominal plan reported in Fig. 3.

aim to keep low the computational burden, no changes are allowed to both conjunctive and disjunctive precedence relationships of the *initial* nominal plan prescribed at time $t^{(0)}$. Therefore corrective actions are limited to anticipating/delaying some vehicles, i.e., by prescribing their

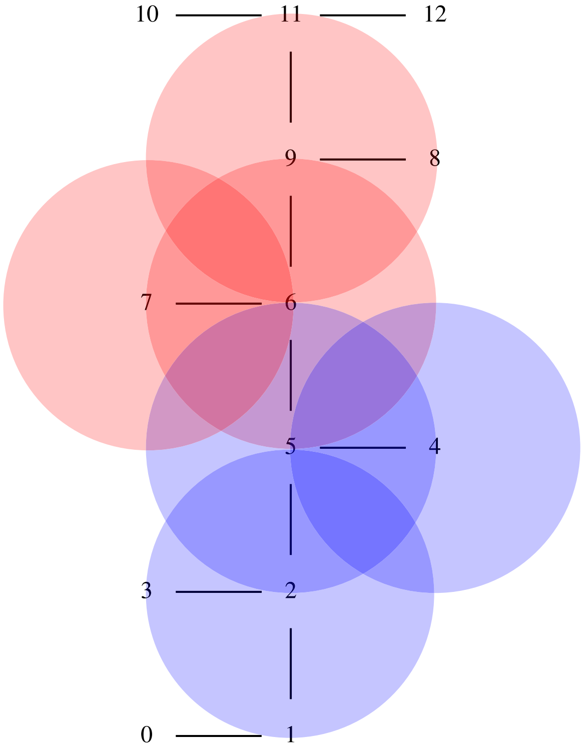


Fig. 5. Illustration of the zone disjunctive precedence constraints in the example of Fig. 3.

waiting times at instants $\mathrm{t}^{(q)}$, with $q > 1$. This leads to formulate a policy based on the solution of a deterministic optimization problem $\mathcal{P}^{(q)}$ defined as follows. Given the *current* nominal plan and the deviations detected at time $\mathrm{t}^{(q)}$, $\mathcal{P}^{(q)}$ aims to determine possible adjustments of the waiting times at several future time points, so that arrival times still satisfy conjunctive/disjunctive precedence relationships of the *initial* nominal plan and minimize the total deviation from the *current nominal arrival times*, with $q > 1$.

3. An optimal policy for schedule adjustments

In this section we model the optimization problem $\mathcal{P}^{(q)}$ as a Linear Program (LP) (3)–(7). Parameters of the proposed LP model are provided by the temporal network associated to the initial conflict-free nominal plan along with the current nominal arrival times and the corresponding (possible) deviations detected at a given time $\mathrm{t}^{(q)}$. Let $\bar{\mathrm{t}}^{(q)}_{hi}$ denote the current nominal arrival time for vehicle h, where $q > 1$, $h \in K$ and $i = 1, \dots, p_h$. As aforementioned, when not null deviations are detected at a given time $\mathrm{t}^{(q)}$, the current nominal plan must be revised in order to avoid collisions among vehicles. The corrective action consists in updating (increasing or decreasing) the waiting time at some visited vertex, while optimizing a performance measure. To this aim we define two continuous non-negative decision variables $\mathrm{t}^{(q)}_{hi}$ and $w^{(q)}_{hi}$, for each $h \in K$ and $i = 1, \dots, p_h$. The LP to be solved is as follows:

$$\min \quad \sum_{h\in K} \sum_{i=1}^{p_h} \left(\mathrm{t}^{(q)}_{hi} - \bar{\mathrm{t}}^{(q-1)}_{hi} \right) \tag{3}$$

s.t.

$$\mathrm{t}^{(q)}_{h,i+1} = \mathrm{t}^{(q)}_{hi} + w^{(q)}_{hi} + \tau_{hi} \qquad \forall\ h \in K;\ i = 1, \dots, p_h - 1 \tag{4}$$

$$\mathrm{t}^{(q)}_{hi} \geq \mathrm{t}^{(q)}_{kj} + w^{(q)}_{kj} + 1 \qquad \forall\ h \in K;\ i = 1, \dots, p_h;\ (k,j) \in \gamma^i_h \tag{5}$$

$$\mathrm{t}^{(q)}_{h1} = \bar{\mathrm{t}}^{(q-1)}_{h1} + d^{(q)}_h \qquad \forall\ h \in K \tag{6}$$

$$\mathrm{t}^{(q)}_{hi}, w^{(q)}_{hi} \geq 0 \qquad \forall\ h \in K;\ i = 1, \dots, p_h \tag{7}$$

The objective function (3) minimizes the total completion time. Constraints (4) and (5) require that time points t_{hi} satisfy both conjunctive and disjunctive precedence relationships of the nominal plan. Constraints (6) impose the detected deviations. Finally, constraints (7) describe non-negative conditions on arrival and waiting times.

We now demonstrate some properties of the LP problem (3)–(7).

Proposition 3.1. *Given a nominal plan, problem* (3)–(7) *is feasible for any* $d^{(q)} \in \mathbb{R}^{|K|}$.

Proof. We observe that if all vehicles share a common deviation, then no corrective actions are required. Therefore a feasible solution of LP problem (3)–(7) can be always determined by assigning the following values to waiting times:

$$w^{(q)}_{h1} = \bar{w}^{(q)}_{h1} + \max_{k\in K} d^{(q)}_k - d^{(q)}_h, \qquad \forall\ h \in K$$

$$w^{(q)}_{hi} = \bar{w}^{(q)}_{hi} \qquad \forall\ h \in K;\ i = 2, \dots, p_h. \qquad \square$$

Proposition 3.2. *Optimal arrival times* t^*_{hi} *for problem* $\mathcal{P}^{(q)}$ *can be determined by the following recursive formula*

$$\mathrm{t}^*_{hi} = \max\{\mathrm{t}^*_{h,i-1} + \tau_{k,i-1}, \max_{(k,j)\in\gamma^i_h} \{\mathrm{t}^*_{k,j+1} - \tau_{kj} + 1\}\}, \tag{8}$$

for any $i = 2, \dots, p_h$, *with the initialization*

$$\mathrm{t}^*_{h1} = \bar{\mathrm{t}}^{(q-1)}_{h1} + d^{(q)}_h$$

for each vehicle $h \in K$.

Proof. We start by defining the projection of the polyhedron (4)–(7) in the space of variables t_{hi}, obtained by expressing waiting times in terms of arrival times as follows

$$w^{(q)}_{hi} := \mathrm{t}^{(q)}_{h,i+1} - \mathrm{t}^{(q)}_{hi} - \tau_{hi} \tag{9}$$

with $h \in K$ and $i = 1, \dots, p_h - 1$. Then model (3)–(7) can be reformulated as:

$$\min \quad \sum_{h\in K} \sum_{i=1}^{p_h} (\mathrm{t}^{(q)}_{hi} - \bar{\mathrm{t}}^{(q-1)}_{hi}) \tag{10}$$

s.t.

$$\mathrm{t}^{(q)}_{h,i+1} \geq \mathrm{t}^{(q)}_{hi} + \tau_{hi} \qquad \forall\ h \in K;\ i = 1, \dots, p_h - 1 \tag{11}$$

$$\mathrm{t}^{(q)}_{hi} \geq \mathrm{t}^{(q)}_{k,j+1} - \tau_{kj} + 1 \qquad \forall\ h \in K;\ i = 1, \dots, p_h,\ (k,j) \in \gamma^i_h \tag{12}$$

$$\mathrm{t}^{(q)}_{h1} = \bar{\mathrm{t}}^{(q-1)}_{h1} + d^{(q)}_h \qquad \forall\ h \in K \tag{13}$$

$$\mathrm{t}^{(q)}_{hi} \geq 0 \qquad \forall\ h \in K;\ i = 1, \dots, p_h \tag{14}$$

It is worth noting that the inequalities (12) has been obtained by substituting in (5) the following definition of waiting times $w^{(q)}_{kj}$ in terms of the arrival times $t^{(q)}_{kj}$ and $t^{(q)}_{k,j+1}$, that is:

$$w^{(q)}_{kj} := \mathrm{t}^{(q)}_{k,j+1} - \mathrm{t}^{(q)}_{kj} - \tau_{kj}$$

with $h \in K;\ i = 1, \dots, p_h,\ (k,j) \in \gamma^i_h$. We start by observing that a compact formulation of (11) and (12) is provided by the following inequalities

$$\mathrm{t}^{(q)}_{hi} \geq \max\{\mathrm{t}^{(q)}_{h,i-1} + \tau_{h,i-1}, \max_{(k,j)\in\gamma^i_h} \{\mathrm{t}^{(q)}_{k,j+1} - \tau_{kj} + 1\}\}, \tag{15}$$

with $i = 2, \dots, p_h$ and $h \in K$. Given a pair (h,i), the minimum value of $\mathrm{t}^{(q)}_{hi}$ satisfies by equality at least one of the inequalities (11) and (12). In particular we have that

$$\tilde{t}^{(q)}_{hi} = \max\{\tilde{t}^{(q)}_{h,i-1} + \tau_{h,i-1}, \max_{(k,j)\in\gamma^i_h} \{\tilde{t}^{(q)}_{k,j+1} - \tau_{kj} + 1\}\},$$

with $\tilde{t}^{(q)}_{hi} = \min(\mathrm{t}^{(q)}_{hi}, \quad s.\ t. \quad (11)–(14))$. This implies the decomposability of LP (10)–(14), i.e.,

$$t^*_{hi} = \min(\mathrm{t}^{(q)}_{hi}, \quad s.\ t. \quad (11)–(14)) \quad \forall\ h \in K,\ i = 2, \dots, p_h,$$

which proves the thesis. $\square$

One implication of the previous proposition is that (8) determines the minimum value of each arrival time $\mathrm{t}^{(q)}_{hi}$. This implies that the recursive formula (8) determines the optimal solution for any LP problem defined on the polyhedron (11)–(14), where the objective function is a non-decreasing function in the arrival times. This leads to the following proposition.

Proposition 3.3. *The recursive formula* (8) *can be used to make decisions using any policy based on the solution of an LP problem defined on the polyhedron* (11)–(14)*, where the objective function is a non-decreasing function in the arrival times, with $h \in K$ and $i = 2, \dots, p_h$.*

4. A polynomial time optimal solution algorithm

Optimality conditions (8) suggest to utilize a dynamic programming approach as a solution strategy. To this aim, we define an auxiliary directed graph $\tilde{G}^{(q)} = (\tilde{V}, \tilde{A}, \tilde{\tau}^{(q)})$, where the vertex set $\tilde{V}$ is made up of all the nodes of the original temporal network $\mathcal{N}_{\mathcal{T}}$, plus a dummy vertex $(\cdot, 0)$, i.e. $\tilde{V} = V_{\mathcal{T}} \cup \{(\cdot, 0)\}$. The arc set $\tilde{A} \subseteq \tilde{V} \times \tilde{V}$ represents the precedence relationships modeled by (*conjunctive*) constraints (11) and (*disjunctive*) constraints (12).

As a result, all the conjunctive arcs of the original temporal network are included in $\tilde{A}$, i.e. $((h,i),(h,i+1))$, with $h \in K$; $i = 1, \dots, p_h - 1$, whilst if $((k,j),(h,i))$ is a disjunctive arc in the original temporal network, i.e. $(k,j) \in \gamma^i_h$, then the arc $((k,j+1),(h,i))$ is inserted in $\tilde{A}$, with $k, h \in K$ and $k \neq h$. Finally $\tilde{A}$ also includes all *dummy* arcs outgoing from the dummy node, i.e. $((\cdot,0),(h,1))$, with $h \in K$. Fig. 6 shows the set of arcs $\tilde{A}$ and vertices $\tilde{V}$ of the auxiliary graphs associated to the temporal network of Fig. 2(b). It is worth noting that all auxiliary graphs $\tilde{G}^{(1)}$, $\tilde{G}^{(2)}$, …, $\tilde{G}^{(q)}$ share a common topological structure, given that the vertex set $\tilde{V}$ and the arc set $\tilde{A}$ describe the precedence relationships of the initial nominal plan. Indeed, between two consecutive stages what has to be updated is only the cost function $\tilde{\tau}^{(q)} : \tilde{A} \to \mathbb{Z}$ defined as follows:

$$\tilde{\tau}^{(q)}((\cdot,0),(h,1)) = \mathrm{t}^{(q-1)}_{h1} + d^{(q)}_h$$

$$\tilde{\tau}^{(q)}((k,j),(h,i)) = \begin{cases} \tau^{(q)}_{kj} & \text{if } h = k,\ j = i-1 \\ 1 - \tau^{(q)}_{k,j-1} & \text{if } (k, j-1) \in \gamma^i_h \\ +\infty & \text{otherwise.} \end{cases}$$

for $h, k \in K; i = 1, \dots, p_h; j = 1, \dots, p_k$.

Let $\mathcal{W}^{(q)}$ be the one-to-all longest path problem on the directed acyclic graph $\tilde{G}^{(q)}$, with $(\cdot, 0)$ as a source node.

Proposition 4.1. *Problems $\mathcal{P}^{(q)}$ and $\mathcal{W}^{(q)}$ form a strong-dual pair.*

Proof. It is worth noting that each arc of $\tilde{A}$ represents either a conjunctive or a disjunctive precedence relationship in the nominal plan. Since the nominal plan is conflict-free and the graph $\tilde{G}^{(q)}$ is acyclic, then the thesis is proved by observing that the value t^*_{hi}, computed with the recursive formula (8), represents the weight of the longest path from source vertex $(\cdot, 0)$ to vertex (h,i) in $\tilde{G}^{(q)}$, for $i = 1, \dots, p_h$ and $h \in K$. □

The longest path problem is well-known to be NP-hard on general directed graphs (Schrijver et al., 2003). In contrast, it can be efficiently solved in linear time on directed acyclic graphs (see, e.g., Sedgewick and Wayne (2011)). In this work, we compute the longest path from the source vertex to all the other vertices in $\tilde{G}^{(q)}$ as a shortest path on $-\tilde{G}^{(q)} = (\tilde{V}, \tilde{A}, -\tilde{\tau}^{(q)})$, i.e. the graph obtained by reversing the sign of the weight for each arc. First, we run a topological sorting procedure based on depth-first search. Subsequently, the search algorithm processes each vertex in topological order and updates the distances from neighboring vertices. The computational complexity of the algorithm is $\mathcal{O}(|\tilde{V}| + |\tilde{A}|)$.

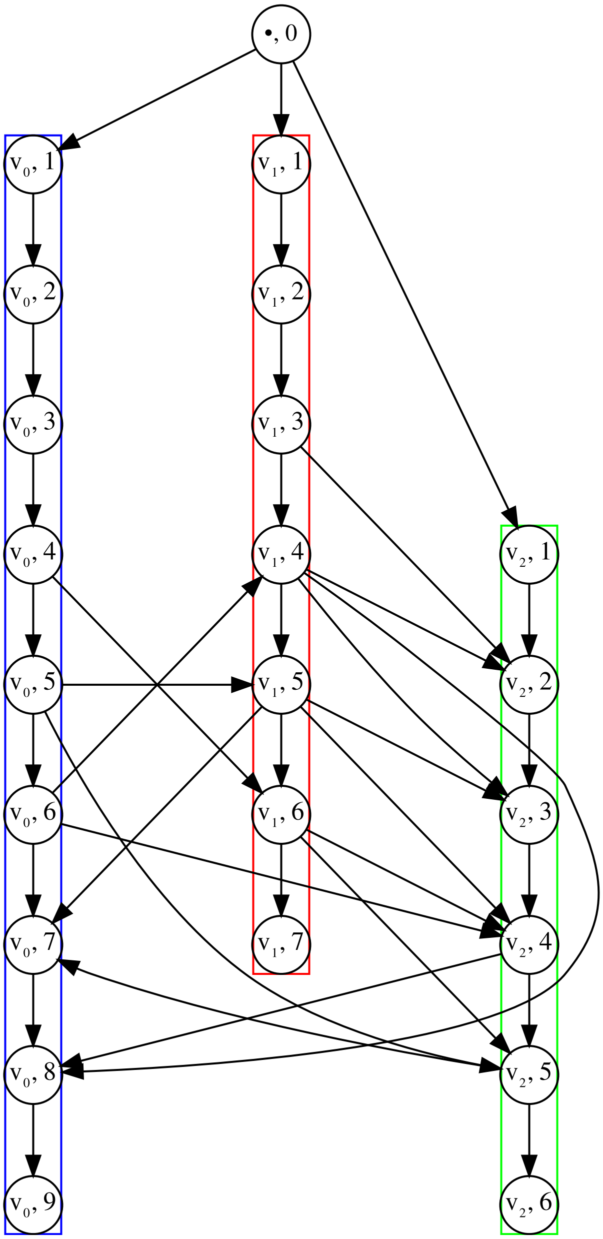


Fig. 6. The auxiliary graph of the temporal network reported in Fig. 2(b).

5. A graph reduction procedure

A further improvement in the computations of the optimal solution of problem $\mathcal{W}^{(q)}$ can be obtained by dominance rules aiming to determine a reduced arc set $\tilde{A}' \subseteq \tilde{A}$. We start by observing that, apart from the arcs outgoing the dummy node, all remaining arcs of $\tilde{A}$ can be univocally associated to a *conjunctive/disjunctive* constraint of problem $\mathcal{P}^{(q)}$. From Proposition 4.1 it descends that each arc belonging to an optimal path in the (*dual*) problem $\mathcal{W}^{(q)}$ corresponds to either a *conjunctive* constraint (11) or a *disjunctive* constraint (12) satisfied as an equality by an optimal solution of the (*primal*) problem $\mathcal{P}^{(q)}$. This implies the following remark.

Remark 1. If a *conjunctive/disjunctive* constraint of $\mathcal{P}^{(q)}$ is redundant then all paths traversing the corresponding arc in $\tilde{A}$ are dominated by a path on $\tilde{G}^{(q)}$ optimal for $\mathcal{W}^{(q)}$.

In the following, we propose two alternative polynomial-time procedures to check whether sufficient redundancy conditions are satisfied for a disjunctive constraint. The main underlying idea is that all auxiliary graphs share a common topological structure $(\tilde{V}, \tilde{A})$ inherited from the original temporal network $\mathcal{N}_\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{A}_\mathcal{T})$, where each arc in $\mathcal{A}_\mathcal{T}$ generates a disjunctive/conjunctive constraint and therefore an arc in $\tilde{A}$. The proposed procedures aim to reduce $\tilde{A}$ by determining a *reduced* temporal network $\mathcal{N}'_\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{A}'_\mathcal{T})$, with $\mathcal{A}'_\mathcal{T} \subseteq \mathcal{A}_\mathcal{T}$. In particular, Algorithm 1 starts by including in $\mathcal{A}'_\mathcal{T}$ all conjunctive arcs (lines 4 and 5). As an example, consider the disjunctive arcs $((0,5),(1,4))$ and $((0,3),(1,6))$ of the temporal network reported in Fig. 7(a). Arc $((0,5),(1,4))$ corresponds to a precedence on the physical vertex 6, i.e. $s_{0,5} = s_{1,4} = 6$. The corresponding disjunctive constraint requires that: $\mathrm{t}_{0,5} < \mathrm{t}_{1,4}$. Arc $((0,3),(1,6))$ corresponds to a disjunctive constraint on the physical vertex 2 and requires that $\mathrm{t}_{0,3} < \mathrm{t}_{1,6}$. Nevertheless, there also exists a path from $(0,3)$ to $(1,6)$, i.e. the path $(0,3)-(0,4)-(0,5)-(1,4)-(1,5)-(1,6)$, which implies that the disjunctive constraint associated to $((0,3),(1,6))$ is redundant, i.e.

$$\mathrm{t}_{0,3} < \mathrm{t}_{0,4} < \mathrm{t}_{0,5} < \mathrm{t}_{1,4} < \mathrm{t}_{1,5} < \mathrm{t}_{1,6} \Rightarrow \mathrm{t}_{0,3} < \mathrm{t}_{1,6}.$$

The following proposition generalizes the considered case by defining sufficient conditions for checking the redundancy of a constraint.

Proposition 5.1. *An arc* $((k,j),(h,i)) \in \mathcal{A}_\mathcal{T}$ *makes redundant any disjunctive constraint associated to* $((k,j'),(h,i')) \in \mathcal{A}_\mathcal{T}$ *such that*

$$i' \le i \wedge j \le j' \wedge (i \ne i' \vee j \ne j') \wedge k \ne h, \tag{16}$$

with $i, i' = 1, \dots, p_h$ *and* $j, j' = 1, \dots, p_k$.

Given two distinct vehicles $h \in K$ and $k \in K$, Algorithm 1 aims to (iteratively) determine pairs of disjunctive arcs not satisfying (16). Let us denote with $((k,j_\ell),(h,i_\ell)) \in \mathcal{A}_\mathcal{T}$ the ℓ-th disjunctive arc *selected* by the Algorithm 1, with $\ell \ge 1$, for vehicles k and h. The pair of arcs $((k,j_{\ell-1}),(h,i_{\ell-1}))$ and $((k,j_\ell),(h,i_\ell))$ satisfies the following condition

$$j_{\ell-1} < j_\ell \wedge i_{\ell-1} < i_\ell, \tag{17}$$

with $2 \le \ell \le \min\{p_h, p_k\}$. It is worth noting that (17) defines a lexicographic order between arcs, i.e. $((k,j_{\ell-1}),(h,i_{\ell-1})) \prec ((k,j_\ell),(h,i_\ell))$. This implies that (17) holds between arc $((k,j_\ell),(h,i_\ell))$ and either any previous selected arc $\ell' \le \ell-1$ or any subsequent selected arc $\ell'' \ge \ell+1$, that is:

$$((k,j_{\ell'}),(h,i_{\ell'})) \prec ((k,j_\ell),(h,i_\ell)) \prec ((k,j_{\ell''}),(h,i_{\ell''})).$$

In particular, Algorithm 1 includes in $\mathcal{A}'_\mathcal{T}$ a disjunctive arc $((k,j_\ell),(h,i_\ell))$ if it satisfies the following conditions.

$$i_\ell = \min_{i_{\ell-1}+1 \le b \le p_h} \{b \mid j_{\ell-1}+1 \le a \le p_k \wedge (k,a) \in \gamma_h^b\}, \tag{18}$$

$$j_\ell = \max_{j_{\ell-1}+1 \le a \le p_k} \{a \mid (k,a) \in \gamma_h^{i_\ell}\} \tag{19}$$

with the initialization $i_0 = j_0 = 0$.

The min and max operators in (18) and (19) are coded by scanning vertices in $\mathcal{V}_\mathcal{T}$ associated to vehicles k and h backwardly and forwardly, respectively (lines 8–13). In particular, Algorithm 1 scans forwardly vertices (h,i) associated to vehicle h from $i = 1$ to $i = p_h$. During the ith (outer) iteration the algorithm scans backwardly vertices (k,j) associated to vehicle k from $j = p_k$ to $j = j_{\ell-1}+1$. If the current pair of vertices $((k,j),(h,i))$ corresponds to an arc of $\mathcal{A}_\mathcal{T}$, then $((k,j),(h,i))$ is included in $\mathcal{A}'_\mathcal{T}$. Then the inner loop is stopped after having updated its stopping condition (line 12).

Time complexity for Algorithm 1 is $\mathcal{O}(|\mathcal{V}|^2)$: the worst case occurs when vehicles plans are completely disjoint, i.e.,

$$\gamma_h^i = \emptyset \qquad h \in K,\ i = 1, \dots, p_h.$$

Algorithm 1 Graph reduction

```
 1: function REDUCE($\mathcal{N}_\mathcal{T}$)
 2:     $\mathcal{A}'_\mathcal{T} \leftarrow \emptyset$                              ▷ initially empty set
 3:     for $h \in K$ do
 4:         for $i \leftarrow 2$ to $p_h$ do
 5:             add $((h,i-1),(h,i))$ to $\mathcal{A}'_\mathcal{T}$                  ▷ conjunctive arc
 6:         for $k \in K : k \neq h$ do
 7:             $prev\_stop \leftarrow 1$
 8:             for $i \leftarrow 1$ to $p_h$ do
 9:                 for $j \leftarrow p_k$ to $prev\_stop$, step $-1$ do
10:                     if $((k,j),(h,i)) \in \mathcal{A}_\mathcal{T}$ then
11:                         add $((k,j),(h,i))$ to $\mathcal{A}'_\mathcal{T}$        ▷ arc satisfying (16)–(17)
12:                         $prev\_stop \leftarrow j+1$
13:                         break
14:     return $\mathcal{A}'_\mathcal{T}$
```

Fig. 7(b) shows an example of graph reduction for the temporal network in Fig. 7(a). The original directed acyclic graph in Fig. 7(a) contains 35 arcs, i.e., 16 disjunctive arcs and 19 conjunctive arcs. Algorithm 1 selects 7 disjunctive arcs out of 16. Nevertheless, Algorithm 1 does not exclude red-dashed arcs, although they are both redundant. This is the case, for example, of path $(0,5)-(1,4)-(2,3)-(2,4)$ resulting from the arrivals of the three vehicles at the physical vertex 6, i.e. $s_{0,5} = s_{1,4} = s_{2,4} = 6$, at times $\bar{\mathrm{t}}_{0,5} = 4$, $\bar{\mathrm{t}}_{1,4} = 7$, $\bar{\mathrm{t}}_{2,4} = 10$ (see the nominal plan of Fig. 1). Such a path makes redundant the disjunctive constraint associated to arc $((0,5),(2,4))$, that is:

$$\mathrm{t}_{0,5} < \mathrm{t}_{1,4} < \mathrm{t}_{2,3} < \mathrm{t}_{2,4} \Rightarrow \mathrm{t}_{0,5} < \mathrm{t}_{2,4}.$$

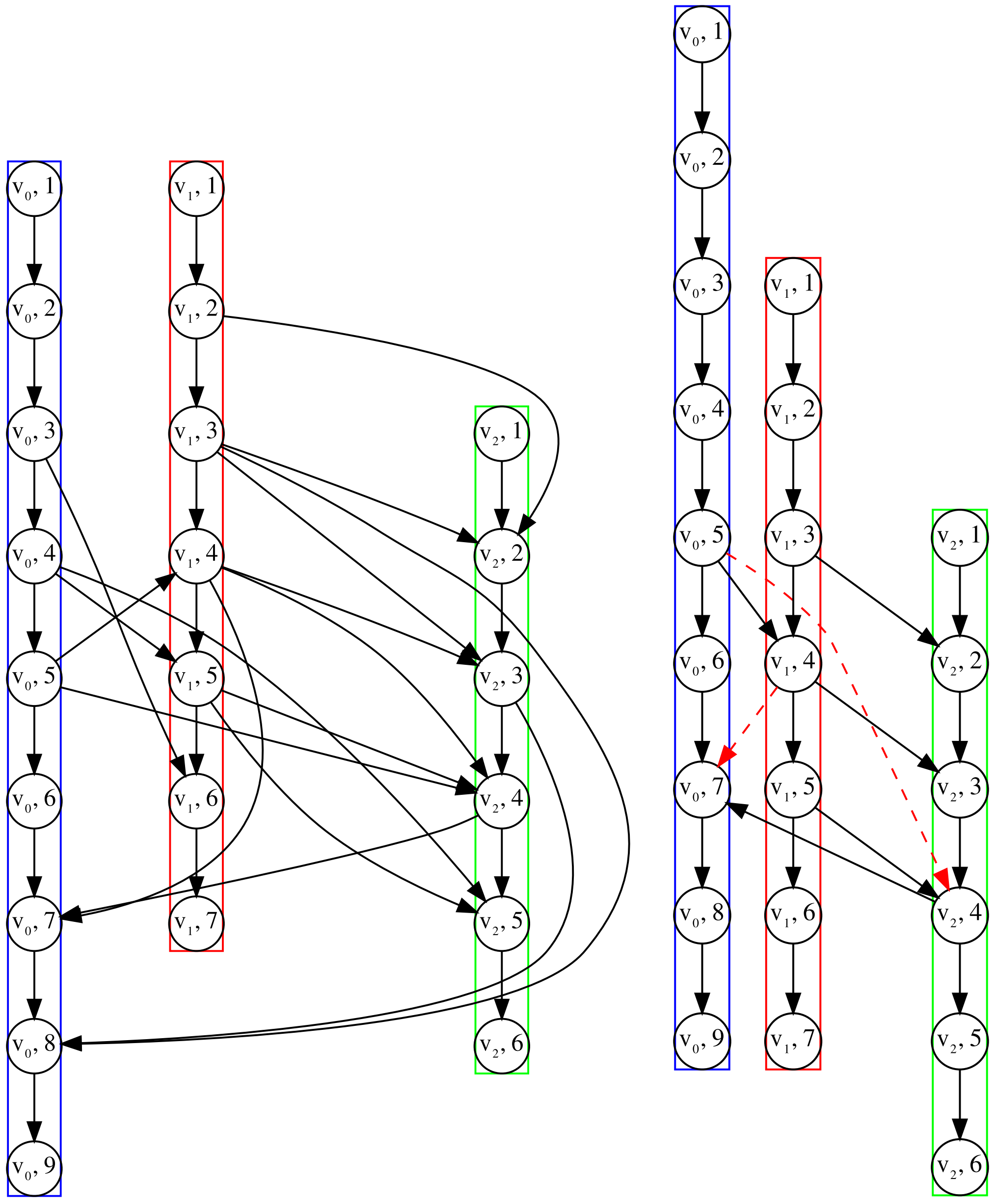
Nevertheless, Algorithm 1 selects the disjunctive arc $((0,5),(2,4))$, since there does not exist a path from $(0,5)$ to $(2,4)$ involving only the two vehicles 0 and 2. To overcome this limit, Algorithm 2 exploits set $\mathcal{A}''_\mathcal{T}$. At the initial step, $\mathcal{A}''_\mathcal{T}$ and $\mathcal{A}'_\mathcal{T}$ include all arcs of $\mathcal{A}_\mathcal{T}$. At a generic iteration, Algorithm 2 can add a (*new*) arc to $\mathcal{A}''_\mathcal{T}$, if it *discovers* a path connecting its endpoints (lines 4–8). On the other hand, an arc $((k,a),(h,b))$ is removed from $\mathcal{A}'_\mathcal{T}$, if during previous iterations a path from (k,a) to (h,b) involving (ℓ,r) has been *discovered* (lines 9–14). As a results, the preprocessing phase is able to generate an equivalent graph with only 24 arcs out of the 35 arcs of the original temporal network, with a selection of 5 disjunctive arcs out of the 16 original ones. More generally, Algorithm 2 selects a (lower) number of disjunctive arcs with a (higher) time complexity of $\mathcal{O}(|\mathcal{V}|^3)$.

Algorithm 2 Topological reduction

```
 1: function TOP-REDUCE($\mathcal{N}_\mathcal{T}$)
 2:     $\mathcal{A}'_\mathcal{T} \leftarrow \mathcal{A}_\mathcal{T}$
 3:     $\mathcal{A}''_\mathcal{T} \leftarrow \mathcal{A}_\mathcal{T}$
 4:     for $(\ell,r) \in \mathcal{V}_\mathcal{T}$ do
 5:         for $(k,a) \in \mathcal{V}_\mathcal{T}$ do
 6:             for $(h,b) \in \mathcal{V}_\mathcal{T}$ do
 7:                 if $((k,a),(\ell,r)) \in \mathcal{A}''_\mathcal{T} \wedge ((\ell,r),(h,b)) \in \mathcal{A}''_\mathcal{T}$ then
 8:                     $\mathcal{A}''_\mathcal{T} \leftarrow \mathcal{A}''_\mathcal{T} \cup \{((k,a),(h,b))\}$
 9:     for $(\ell,r) \in \mathcal{V}_\mathcal{T}$ do
10:         for $(k,a) \in \mathcal{V}_\mathcal{T}$ do
11:             for $(h,b) \in \mathcal{V}_\mathcal{T}$ do
12:                 if $k \neq h$ then                    ▷ remove only disjunctive arcs
13:                     if $((k,a),(\ell,r)) \in \mathcal{A}''_\mathcal{T} \wedge ((\ell,r),(h,b)) \in \mathcal{A}''_\mathcal{T}$ then
14:                         $\mathcal{A}'_\mathcal{T} \leftarrow \mathcal{A}'_\mathcal{T} \setminus \{((k,a),(h,b))\}$
15:     return $\mathcal{A}'_\mathcal{T}$
```

6. Computational results

This section presents the results of computational experiments carried out to evaluate the performance of our method and its applicability



(a) Original temporal network

(b) Reduced temporal network

Fig. 7. An example of graph reduction for the original nominal plan of Fig. 1.

in real-time contexts. In details, we compared the proposed solution procedure for the base model of SA-CFVRP, with the more conservative approach introduced in Adamo et al. (2023) and also with respect to an off-the-shelf solver. Experiments were performed on a Linux machine featuring an Intel Core i7 processor with 4 cores operating at 2.5 GHz, and 16 GB of RAM. The algorithms were implemented in C++. The linear program (3)–(7) has been solved with IBM ILOG CPLEX 22.1.1 (IBM, 2023).

We tested two sets of benchmark instances: the former refers to Grid WareHouse (GWH) networks taken from the literature, whilst the latter (RND) has been randomly generated. The GWH set has been composed by using four different physical networks that mimic real-world automated warehouses (Ma et al., 2017, Stern et al. (2019)).

Each network is characterized by a set of homogeneous rectangular obstacles. The placement of obstacles follows a pattern consisting of equally spaced rows and equally spaced columns. The instance name *warehouse-A-B-C-D-E* encodes the features of the network as follows:

- A: number of obstacles columns;
- B: number of obstacles rows;
- C: obstacle width;
- D: obstacle height;
- E: distance between rows/columns of consecutive obstacles.

Table 1 describes the main features of the GWH test instances. Column headings have the following meaning:

Table 1
Features of GWH physical networks.

Id	Name	Width	Height	Unreachable locations
W_1	warehouse-10-20-10-2-1	161	63	4444
W_2	warehouse-10-20-10-2-2	170	84	4504
W_3	warehouse-20-40-10-2-1	321	123	16 884
W_4	warehouse-20-40-10-2-2	340	164	17 004

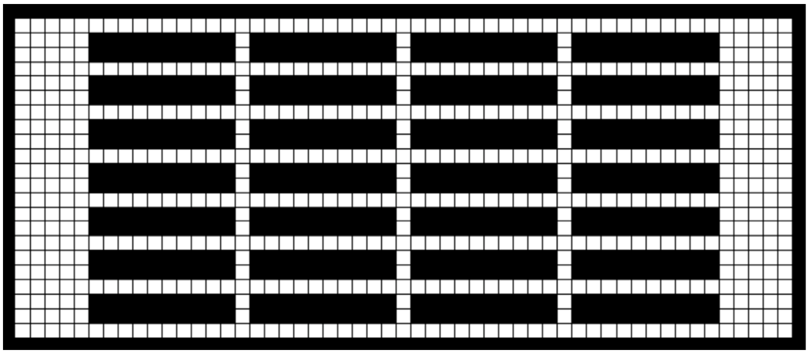


Fig. 8. Example of a GWH warehouse grid (Cohen et al., 2018).

Table 2
Computational results — GWH instances.

Network	|K|	T_0 [ms]	T_1 [ms]	T_2 [ms]	T_3 [ms]	DEV_1 [%]	DEV_2 [%]	DEV_3 [%]	DEV_4 [%]
W_1	50	437.06	0.01	0.08	0.05	13.2	14.3	0.0	6.2
	100	1574.70	0.04	0.21	0.16	16.6	17.1	0.6	16.7
	150	3707.35	0.08	0.36	0.37	17.7	17.7	0.5	22.3
	200	7414.27	0.13	0.61	0.67	19.1	18.9	0.4	26.5
	250	13 359.15	0.20	0.98	1.11	18.8	18.7	0.5	27.9
	300	21 513.42	0.28	1.53	1.67	17.7	17.7	0.6	26.6
W_2	50	425.17	0.01	0.08	0.04	13.6	14.1	0.1	6.5
	100	1215.36	0.03	0.19	0.12	18.5	18.6	0.2	16.5
	150	2466.87	0.08	0.32	0.25	18.1	17.6	0.3	21.0
	200	4338.21	0.13	0.50	0.42	18.6	18.7	0.4	23.7
	250	7079.06	0.20	0.78	0.66	18.0	18.4	0.5	24.5
	300	10 529.45	0.29	1.05	0.97	17.4	17.2	0.5	24.7
W_3	50	754.58	0.01	0.13	0.06	9.1	6.1	0.1	1.3
	100	2135.55	0.04	0.30	0.19	13.6	14.1	0.1	7.4
	150	4459.01	0.07	0.56	0.38	16.7	16.4	0.2	13.4
	200	7761.37	0.13	0.83	0.65	16.4	16.6	0.1	15.9
	250	12 322.44	0.20	1.37	1.14	17.6	17.6	0.1	20.2
	300	18 430.17	0.29	1.94	1.64	17.4	17.0	0.4	22.4
W_4	50	708.70	0.01	0.14	0.06	5.6	7.5	0.0	1.4
	100	1852.06	0.03	0.30	0.15	13.5	14.0	0.1	5.7
	150	3176.26	0.07	0.53	0.29	17.0	16.1	0.2	12.0
	200	5335.99	0.13	0.87	0.50	17.3	17.1	0.1	14.6
	250	8269.27	0.20	1.23	0.79	17.8	17.8	0.1	18.1
	300	13 030.09	0.29	2.08	1.33	20.0	20.2	0.2	24.3
AVG		**6345.65**	**0.12**	**0.71**	**0.57**	**16.2**	**16.2**	**0.3**	**16.7**

- **width** of the grid expressed in terms of number of locations;
- **height** of the grid expressed in terms of number of locations;
- **unreachable locations**: the total number of locations occupied by rectangular obstacles plus the edge locations of the grid.

For example, Fig. 8 shows an illustration taken from Cohen et al. (2018) of instance *warehouse-4-7-10-2-1*, having a width of 55 locations, a height of 24 locations and 714 unreachable locations.

There are 25 scenarios for any physical network. Each scenario specifies a list of start/goal locations evenly distributed. We found a feasible conflict-free nominal plan using a suboptimal variant of the conflict-based search algorithm (Barer et al., 2014, Sharon et al. (2015)) with a number of vehicles $|K| \in \{50, 100, 150, 200, 250, 300\}$. Therefore, we tested 600 GWH instances in total.

With regards to the RND set, we randomly generated a set of nominal plans, with a number of vehicles $|K| \in \{50, 100, 150, 200, 250, 300\}$. Each vehicle is assigned a path consisting of a number of vertices randomly generated according to a uniform distribution, i.e. $p_h \sim \mathcal{U}_{[70,100]}$, with $h \in K$. Similarly, we randomly generated the traversal times with uniformly distributed values, i.e. $\tau_h^i \sim \mathcal{U}_{[10,30]}$, with $h \in K$, $i = 1, \ldots, p_h - 1$. For each nominal plan, the disjunctive arcs were generated based on a conflict graph, which is a directed graph where each node corresponds to a vehicle and the arc (h, k) represents a pair of vehicles such that there exists at least one disjunctive arc from vehicle h to vehicle k, for $h, k \in K$. A parameter λ models the sparsity of the conflict graph, taking values in $\{0\%, 25\%, 50\%, 75\%\}$, where 0% corresponds to a complete graph. To generate the set of disjunctive arcs associated with each arc (h, k) in the conflict graph, the following procedure was applied. Initially, a zero matrix with dimensions of p_h rows and p_k columns was considered. Iteratively, a null element (i, j) was randomly selected from the matrix, with $i = 1, \ldots, p_h$ and $j = 1, \ldots, p_k$. Once a null element (i, j) was chosen, it was set to 1, along with all the elements (i', j') that satisfied condition (16). Additionally, the chosen element (i, j) was used to add the disjunctive constraint $((h, i), (k, j))$ to the nominal plan. The procedure stops when the sparsity of the matrix gets greater than 30%. It is worth noting that setting elements (i', j') satisfying condition (16) to 1 allowed us to avoid generating "trivial" redundancies. For each pair $(|K|, \lambda)$ we generated 15 instances resulting in a total number of 360 RND instances.

For both sets of benchmark instances (GWH and RND), misalignments of vehicles with respect to the nominal plan were also uniformly distributed, with $d_h \sim \mathcal{U}_{[-20,20]}$ and $h \in K$. We tested the proposed approaches with respect to four objective functions z_1, z_2, z_3 and z_4. In particular, z_1 measures the total corrective delay of each vehicle w.r.t. the nominal plan, i.e.

$$z_1 = \min \sum_{h \in K} \left(\mathrm{t}_{h,p_h}^{(q)} - \bar{\mathrm{t}}_{h,p_h}^{(q-1)} \right);$$

z_2 measures the total weighted corrective delay w.r.t nominal timetable, i.e.

$$z_2 = \min \sum_{h \in K} w_h \left(\mathrm{t}_{h,p_h}^{(q)} - \bar{\mathrm{t}}_{h,p_h}^{(q-1)} \right)$$

with $w_h \geq 0$ for $h \in K$; z_3 measures the makespan, i.e.

$$z_3 = \min \max_{h \in K} \mathrm{t}_{h,p_h}^{(q)}.$$

Finally, z_4 measures the total lateness of the vehicles with respect to their due dates $\bar{\mathrm{t}}_{h,p_h}^{(0)} + \rho_h$ for $h \in K$, i.e.

$$z_4 = \min \sum_{h \in K} \max \{0, \mathrm{t}_{h,p_h}^{(q)} - \bar{\mathrm{t}}_{h,p_h}^{(0)} - \rho_h\}$$

Weights w_h and parameters ρ_h were generated uniformly as:

$$w_h \sim \mathcal{U}_{[0,20]}, \quad \rho_h \sim \mathcal{U}_{[0,20]},$$

with $h \in K$.

Tables 2 and 3 aim to compare the performance of the proposed approach with the performance of both algorithm by Adamo et al. (2023) and the linear program (3)–(7) solved by IBM ILOG CPLEX 22.1.1. The computational results are reported in Tables 2 and 3 under the following headings:

- T_0: average time in milliseconds required to solve linear program (3)–(7);
- T_1: average time in milliseconds spent by the algorithm proposed in Adamo et al. (2023);

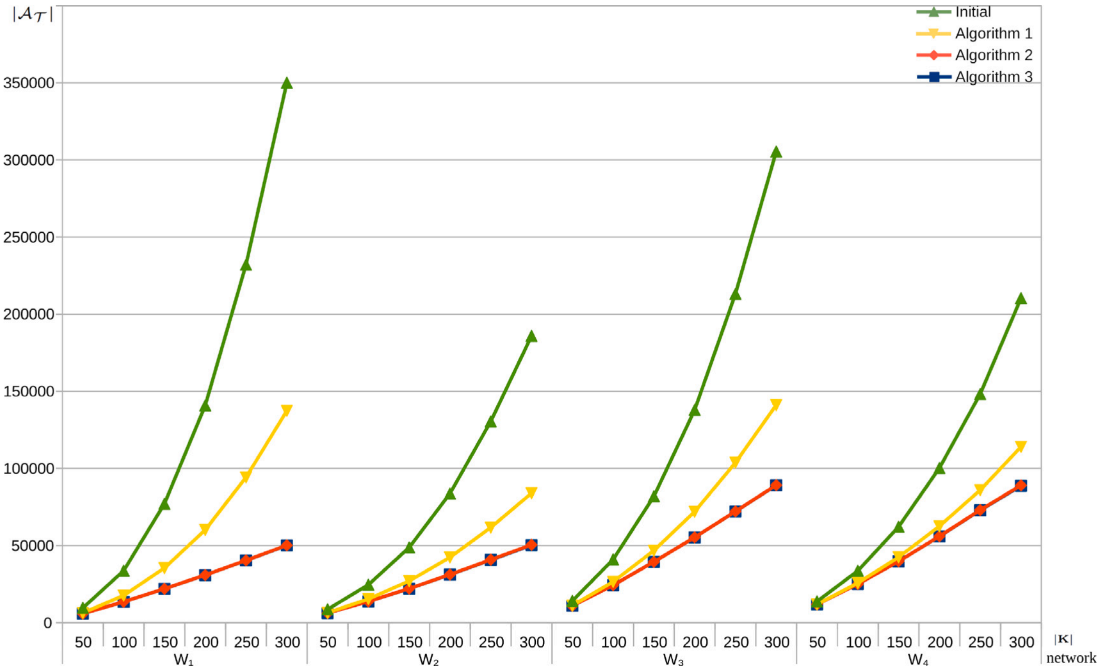


Fig. 9. Arcs reduction — GWH instances.

Table 3
Computational results — RND instances.

λ [%]	$\lvert K\rvert$	T_0 [ms]	T_1 [ms]	T_2 [ms]	T_3 [ms]	DEV_1 [%]	DEV_2 [%]	DEV_3 [%]	DEV_4 [%]
0	50	608.29	0.01	0.10	0.12	79.3	79.4	0.4	64.9
	100	2274.64	0.02	0.25	0.43	66.9	67.0	0.5	65.0
	150	6569.55	0.04	0.48	0.98	83.0	83.4	0.5	79.1
	200	12 636.14	0.06	0.66	1.58	69.2	69.4	0.6	70.0
	250	21 503.70	0.09	0.90	2.34	70.8	70.2	0.7	74.8
	300	33 425.86	0.13	1.22	3.29	74.2	74.2	0.8	76.6
25	50	464.58	0.01	0.10	0.10	82.1	78.5	0.4	69.7
	100	1745.48	0.02	0.23	0.33	87.1	86.9	0.5	73.8
	150	4784.70	0.04	0.45	0.78	82.9	82.3	0.6	77.0
	200	9196.09	0.06	0.60	1.22	70.7	70.4	0.4	74.8
	250	15 466.61	0.09	0.82	1.88	69.0	68.9	0.5	71.6
	300	23 644.13	0.12	1.04	2.59	71.2	71.7	0.5	74.1
50	50	350.87	0.01	0.09	0.07	84.1	86.5	0.3	46.7
	100	1249.35	0.02	0.21	0.24	74.1	73.6	0.3	66.1
	150	3154.16	0.04	0.39	0.56	80.2	80.1	0.4	71.4
	200	5902.28	0.06	0.55	0.93	79.3	78.6	0.5	74.8
	250	9697.25	0.09	0.76	1.46	61.1	61.3	0.5	65.2
	300	14 639.01	0.13	0.87	1.86	67.2	67.2	0.5	70.5
75	50	214.06	0.01	0.08	0.05	64.4	64.0	0.0	36.0
	100	716.56	0.02	0.17	0.15	80.6	81.0	0.2	55.0
	150	1652.10	0.04	0.31	0.30	67.4	68.2	0.2	57.3
	200	2893.84	0.06	0.48	0.56	72.8	71.9	0.2	62.6
	250	4659.25	0.10	0.61	0.82	76.0	76.0	0.3	68.9
	300	6867.06	0.15	0.72	1.08	69.3	69.0	0.3	66.7
AVG		**7679.81**	**0.06**	**0.50**	**0.99**	**74.3**	**74.2**	**0.4**	**67.2**

- T_2: average time in milliseconds spent in the topological sorting proposed by Sedgewick and Wayne (2011);
- T_3: average time in milliseconds spent in longest path computations;
- DEV_i: the average percentage gap of the objective function values, i.e.

$$DEV_i = \frac{z_i^A - z_i}{z_i^A},$$

with $i = 1, \dots, 4$ and z_i^A denotes the objective function value determined by Adamo et al. (2023).

For each row we report averages across all 25 instances for the GWH set and 15 instances for the RND set. Tables 2 and 3 demonstrate that the proposed approach determines the optimal solution with an average computational time of approximately 1 millisecond. Above all, in terms of computing time, our exact algorithm outperformed the off-the-shelf solver even with a few vehicles. It is worth noting that, although the approach proposed by Adamo et al. (2023) may exhibit better time-efficiency, it yields remarkably worse solutions compared to our method, particularly in terms of total delay.

From a managerial point of view, allowing adjustments of the waiting times w.r.t. future time points in order to avoid conflicts was profitable across the four objective functions (especially on total corrective delay, total weighted corrective delay and total lateness). These enhancements directly translate into higher throughput within the physical layout.

Table 2 shows that GWH instances have better performance when considering more vehicles, probably due to a better utilization of the layout. Tables 4 and 5 show the impact of graph reduction algorithms on the real-time computation of the longest path on $\tilde{G}^{(q)}$. The columns headings are:

- $\Delta|\mathcal{A}_{\mathcal{T}}|$: average percentage reduction in the cardinality of arc set, that is:

  $$\Delta|\mathcal{A}_{\mathcal{T}}| = \frac{|\mathcal{A}_{\mathcal{T}}| - |\mathcal{A}'_{\mathcal{T}}|}{|\mathcal{A}_{\mathcal{T}}|}$$

- T_r: average preprocessing time spent in graph reduction;
- ΔT_3: average percentage reduction of computing time T_3, i.e.

  $$\Delta T_3 = \frac{T_3 - T'_3}{T_3},$$

  where T'_3 is the computing time on the reduced graph.

We also determined an upper bound for $\Delta|\mathcal{A}_{\mathcal{T}}|$ by executing the procedure (dubbed Algorithm 3) reported in Appendix, which is a slight variant of the Floyd–Warshall's all-pairs-shortest-paths algorithm (Floyd, 1962; Warshall, 1962). This procedure cannot be executed as a preprocessing step since it computes all-pairs-shortest-paths on the basis of the current auxiliary graph $\tilde{G}^{(q)}$. However, it identifies all the arcs

Table 4
Impact of graph reduction — GWH instances.

Instances				Algorithm 1			Algorithm 3			Algorithm 2		
Network	$\vert K\vert$	$\vert\mathcal{V}_\mathcal{T}\vert$	$\vert\mathcal{A}_\mathcal{T}\vert$	$\Delta\vert\mathcal{A}_\mathcal{T}\vert$ [%]	T_r [s]	ΔT_3 [%]	$\Delta\vert\mathcal{A}_\mathcal{T}\vert$ [%]	T_r [s]	ΔT_3 [%]	$\Delta\vert\mathcal{A}_\mathcal{T}\vert$ [%]	T_r [s]	ΔT_3 [%]
W_1	50	4675	9531	33.4	0.06	34.5	38.1	7.05	48.9	38.1	6.78	48.1
	100	9463	33 673	47.4	0.30	54.1	59.4	58.65	64.6	59.4	61.15	64.3
	150	14 260	77 054	53.9	0.75	60.1	71.2	231.99	73.1	71.2	250.27	72.5
	200	19 168	140 813	57.2	1.45	63.5	77.8	668.24	78.4	77.8	707.92	77.0
	250	24 276	232 115	59.3	2.49	67.5	82.5	1557.38	82.8	82.4	1597.46	81.9
	300	29 500	350 063	60.7	3.85	70.6	85.5	3043.24	84.5	85.5	3061.60	84.5
W_2	50	5349	8603	24.7	0.08	29.2	27.2	9.41	35.2	27.1	8.96	34.3
	100	10 623	24 558	37.5	0.34	43.0	43.9	61.60	51.9	43.8	64.87	50.6
	150	15 966	48 728	44.5	0.83	52.1	54.6	220.48	60.5	54.5	245.76	59.4
	200	21 373	83 688	49.4	1.58	54.4	62.8	583.42	66.6	62.6	660.22	65.0
	250	26 849	130 445	52.7	2.61	58.1	68.8	1302.22	71.1	68.7	1483.28	69.9
	300	32 286	185 926	54.8	3.93	59.2	72.9	2520.24	75.5	72.8	2861.51	74.4
W_3	50	9645	14 099	20.4	0.25	26.5	22.0	43.33	27.8	22.0	39.59	26.6
	100	19 226	40 941	34.8	1.10	44.4	40.1	301.05	49.3	40.1	306.80	49.0
	150	29 083	81 876	42.7	2.72	53.0	51.6	1095.09	59.7	51.6	1176.22	57.9
	200	38 757	137 902	47.7	5.15	58.1	59.7	2850.89	66.5	59.7	3162.54	66.4
	250	48 597	213 250	51.3	8.53	62.0	66.1	6469.46	70.1	66.0	7188.30	69.3
	300	58 289	305 486	53.8	12.87	63.1	70.8	12 419.66	71.3	70.7	14 140.22	69.1
W_4	50	11 038	13 740	12.9	0.31	19.0	13.6	64.64	19.9	13.6	63.03	19.2
	100	21 725	33 620	23.3	1.30	27.6	25.4	400.88	34.7	25.4	427.83	32.4
	150	32 510	62 175	31.6	3.07	36.0	36.0	1335.13	42.2	35.9	1518.22	36.7
	200	43 556	100 245	37.6	5.78	43.3	44.2	3431.52	45.9	44.1	4045.13	45.4
	250	54 570	148 207	42.0	9.49	42.2	50.8	8041.00	54.5	50.7	14 198.72	50.4
	300	65 648	210 372	45.8	14.12	30.9	57.8	27 971.07	51.4	57.6	30 765.78	48.5
AVG		**26 935**	**111 963**	**42.5**	**3.46**	**48.0**	**53.5**	**3111.99**	**57.8**	**53.4**	**3668.42**	**56.4**

Table 5
Impact of graph reduction — RND instances.

Instances				Algorithm 1			Algorithm 3			Algorithm 2		
λ [%]	$\vert K\vert$	$\vert\mathcal{V}_\mathcal{T}\vert$	$\vert\mathcal{A}_\mathcal{T}\vert$	$\Delta\vert\mathcal{A}_\mathcal{T}\vert$ [%]	T_r [s]	ΔT_3 [%]	$\Delta\vert\mathcal{A}_\mathcal{T}\vert$ [%]	T_r [s]	ΔT_3 [%]	$\Delta\vert\mathcal{A}_\mathcal{T}\vert$ [%]	T_r [s]	ΔT_3 [%]
0	50	4261	32 229	57.8	0.03	43.2	71.2	11.40	54.9	69.9	12.40	54.9
	100	8525	121 614	61.9	0.16	58.0	79.2	114.23	66.1	77.6	120.93	62.8
	150	12 757	268 002	63.4	0.40	62.0	82.7	430.81	72.4	81.0	462.12	71.7
	200	17 081	471 663	64.1	0.78	60.1	84.7	1107.41	74.0	83.0	1200.63	71.3
	250	21 262	732 231	64.6	1.28	56.2	86.0	2234.09	75.3	84.3	2456.09	71.0
	300	25 413	1 049 747	64.9	1.92	56.4	87.0	3940.60	73.8	85.3	4374.14	73.5
25	50	4235	25 225	55.4	0.03	41.6	66.4	9.61	57.8	65.3	10.74	52.9
	100	8520	93 287	60.5	0.17	53.3	75.6	97.34	59.9	74.2	104.72	57.2
	150	12 783	204 201	62.4	0.41	61.7	79.7	383.83	68.6	78.1	411.62	67.7
	200	16 993	357 833	63.4	0.78	57.8	82.2	982.51	68.7	80.4	1059.20	68.2
	250	21 214	554 296	64.0	1.31	60.2	83.8	2007.71	72.1	82.0	2185.93	68.7
	300	25 465	793 762	64.4	1.95	58.2	85.0	3620.15	70.8	83.1	3970.99	69.7
50	50	4229	18 209	51.1	0.03	40.7	58.9	7.25	48.2	58.0	8.41	47.8
	100	8526	65 129	57.8	0.16	46.8	69.8	76.79	53.0	68.4	84.69	50.1
	150	12 782	140 333	60.5	0.40	58.0	74.8	310.56	61.0	73.3	336.38	60.7
	200	16 992	244 171	61.9	0.76	59.3	77.9	809.51	64.6	76.2	874.29	63.0
	250	21 220	376 813	62.8	1.26	61.3	79.9	1694.52	69.0	78.1	1836.75	66.0
	300	25 507	537 521	63.4	1.91	57.7	81.5	3114.96	65.9	79.6	3391.30	65.5
75	50	4219	11 230	41.5	0.03	45.6	44.6	4.16	49.4	44.2	5.03	47.3
	100	8492	36 771	51.2	0.14	39.1	57.5	46.36	45.2	56.6	54.21	44.3
	150	12 745	76 531	55.5	0.35	41.7	64.2	196.19	47.8	63.0	222.61	43.1
	200	16 980	130 765	57.9	0.67	51.4	68.3	529.94	53.3	66.9	592.38	51.6
	250	21 201	198 951	59.5	1.12	53.5	71.2	1142.42	56.6	69.7	1268.13	55.5
	300	25 515	281 449	60.5	1.70	53.0	73.4	2133.44	57.4	71.7	2330.70	53.4
AVG		**14 871**	**284 248**	**59.6**	**0.74**	**53.2**	**74.4**	**1041.91**	**61.8**	**72.9**	**1140.60**	**60.0**

that do not belong to any shortest path on $-\tilde{G}^{(q)}$, thereby providing an upper bound on $\Delta|\mathcal{A}_\mathcal{T}|$.

The results reported in Tables 4 and 5 empirically demonstrate how the computational times for finding the longest paths can be reduced to less than one millisecond by utilizing the reduction procedure presented in Section 5. Specifically, Algorithm 1 was capable of removing approximately 60% of the *redundant* arcs, resulting in a significant improvement in computational times of about 53%. Although Algorithm 2 incurred a higher computational burden, it achieved a substantial reduction in the number of arcs, approaching the upper bound determined by Algorithm 3. This trend is also illustrated in Figs. 9 and 10.

An increase of $|K|$ determines a remarkable growth of T_r, especially with respect to Algorithms 2 and 3. Finally, Tables 3 and 5 show that lower λ values (i.e. denser conflict graphs) correspond to higher computational times.

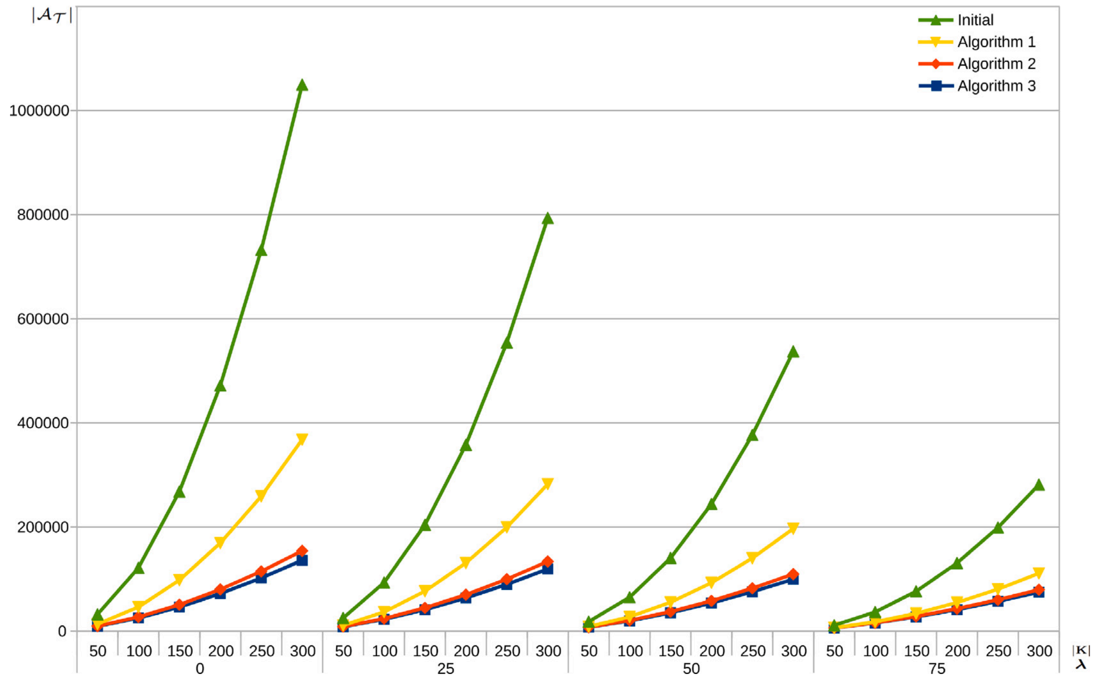


Fig. 10. Arcs reduction — RND instances.

7. Conclusions

In this paper, we have addressed the problem of determining schedule adjustments in Conflict-Free Vehicle Routing Problem when some vehicles deviate from a nominal plan. This problem is of the utmost importance in manufacturing, transportation, and logistics facilities that utilize AGVs to transport loads between stations. We have modeled the SA-CFVRP as a sequential decision problem and have proposed a fast exact algorithm to solve it. Through an extensive empirical study, we have demonstrated that our exact algorithm is much faster than the IBM ILOG CPLEX 22.1.1 solver. Results have also showed that the proposed approach have consistently outperformed the (Adamo et al., 2023) procedure in terms of objective function values, with computing times in the same order of magnitude (less than 3.3 milliseconds for instances with up to 300 vehicles).

Future research might focus on the exploitation of historical data to generate *robust plans* in order to reduce the need of corrective actions and increase the overall efficiency.

An additional research opportunity is based on the following observation: the proposed approach works well when delays and anticipations stem from minor (*endogenous*) uncertainties, primarily associated with the continuous time and space dynamics of the fleet of vehicles. However, this (*data-driven*) approach may encounter challenges in the face of relevant disruptions such as a vehicle blocking an arc because of a major mechanical failure. In this case, a new nominal plan has to be generated by considering that a vehicle is out of service and at least an arc of the path layout is not available. To this purpose fast re-optimization techniques might be valuable.

CRediT authorship contribution statement

Tommaso Adamo: Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Conceptualization. **Gianpaolo Ghiani:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Investigation, Conceptualization. **Emanuela Guerriero:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no conflict of interest.

Data availability

Data will be made available on request.

Acknowledgments

This work was partly supported by Ministero dell'Università e della Ricerca (MUR) of Italy. This support is gratefully acknowledged ("Decreto Ministeriale n. 1062 del 10-08-2021. PON Ricerca e Innovazione 14-20 nuove risorse per contratti di ricerca su temi dell'innovazione" contract number 12-I-13147-10).

Appendix

In general, we can reduce graph $\tilde{G}^{(q)}$, without any topological consideration, using a slightly modified version of the well-known Floyd–Warshall's procedure (Floyd, 1962; Warshall, 1962) as reported in Algorithm 3.

Algorithm 3 removes every arc in $\tilde{A}$ not belonging to any longest path in $\tilde{G}^{(q)}$. In particular, we changed the original algorithm at line 17 where we introduced a minus sign in order to compute longest paths on $\tilde{G}^{(q)}$, and we added line 10 to remove every unnecessary arc. It is worth noting that completeness of the original Floyd–Warshall's algorithm in finding all-pairs shortest paths also ensures completeness of Algorithm 3. Time complexity is obviously $\mathcal{O}(|\tilde{V}|^3)$.

Algorithm 3 Floyd-Warshall reduction

```
function FW-REDUCE($\tilde{G}^{(q)}$)
    $\tilde{A}' \leftarrow \tilde{A}$
    $c \leftarrow$ INIT($\tilde{G}^{(q)}$)
    for $(\ell, r) \in \tilde{V}$ do
        for $(k, j) \in \tilde{V}$ do
            for $(h, i) \in \tilde{V}$ do
                if $c((k,j),(\ell,r)) \neq +\infty \ \wedge \ c((\ell,r),(h,i)) \neq +\infty$ then
                    if $c((k,j),(h,i)) > c((k,j),(\ell,r)) + c((\ell,r),(h,i))$ then
                        $c((k,j),(h,i)) \leftarrow c((k,j),(\ell,r)) + c((\ell,r),(h,i))$
                        remove $((k,j),(h,i))$ from $\tilde{A}'$
    return $\tilde{A}'$
function INIT($\tilde{G}^{(q)}$)
    $c \leftarrow$ new 2D array of size $|\tilde{V}| \times |\tilde{V}|$
    for $(k, j) \in \tilde{V}$ do
        for $(h, i) \in \tilde{V}$ do
            if $((k,j),(h,i)) \in \tilde{A}$ then
                $c((k,j),(h,i)) \leftarrow -\tilde{\tau}^{(q)}((k,j),(h,i))$
            else
                $c((k,j),(h,i)) \leftarrow +\infty$
    return $c$
```

References

Adamo, T., Bektaş, T., Ghiani, G., Guerriero, E., Manni, E., 2018. Path and speed optimization for conflict-free pickup and delivery under time windows. Transp. Sci. 52 (4), 739–755.

Adamo, T., Ghiani, G., Guerriero, E., 2023. Recovering feasibility in real-time conflict-free vehicle routing. Comput. Ind. Eng. 183, 109437.

Barer, M., Sharon, G., Stern, R., Felner, A., 2014. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In: Proceedings of the International Symposium on Combinatorial Search, vol. 5, (1), pp. 19–27.

Cao, Y., Yang, A., Liu, Y., Zeng, Q., Chen, Q., 2023. AGV dispatching and bidirectional conflict-free routing problem in automated container terminal. Comput. Ind. Eng. 184, 109611.

Chen, J., Zhang, X., Peng, X., Xu, D., Peng, J., 2022. Efficient routing for multi-AGV based on optimized ant-agent. Comput. Ind. Eng. 167, 108042.

Cohen, L., Koenig, S., Kumar, T.S., Wagner, G., Choset, H., Chan, D.M., Sturtevant, N.R., 2018. Rapid randomized restarts for multi-agent path finding: Preliminary results. In: AAMAS. pp. 1909–1911.

Corréa, A.I., Langevin, A., Rousseau, L.-M., 2007. Scheduling and routing of automated guided vehicles: A hybrid approach. Comput. Oper. Res. 34 (6), 1688–1707.

Desaulniers, G., Langevin, A., Riopel, D., Villeneuve, B., 2003. Dispatching and conflict-free routing of automated guided vehicles: An exact approach. Int. J. Flexible Manuf. Syst. 15 (4), 309–331.

Desrosiers, J., Pelletier, P., Soumis, F., 1983. Plus court chemin avec contraintes d'horaires. RAIRO Oper. Res. 17 (4), 357–377.

Floyd, R.W., 1962. Algorithm 97: shortest path. Commun. ACM 5 (6), 345.

Fragapane, G., De Koster, R., Sgarbossa, F., Strandhagen, J.O., 2021. Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. European J. Oper. Res. 294 (2), 405–426.

Gawrilow, E., Köhler, E., Möhring, R.H., Stenzel, B., 2008. Dynamic routing of automated guided vehicles in real-time. In: Mathematics–Key Technology for the Future. Springer, pp. 165–177.

Grand View Research, 2022. Automated guided vehicle market size, share & trends analysis report by vehicle type, by navigation technology, by application, by end use industry, by component, by battery type, and segment forecasts, 2022–2030. https://www.grandviewresearch.com/industry-analysis/automated-guided-vehicle-agv-market.

Hwang, I., Jang, Y.J., 2020. Q (λ) learning-based dynamic route guidance algorithm for overhead hoist transport systems in semiconductor fabs. Int. J. Prod. Res. 58 (4), 1199–1221.

IBM, 2023. V22.1.1: User's manual for CPLEX. https://www.ibm.com/docs/en/icos/22.1.1.

Kim, C.W., Tanchoco, J.M., 1991. Conflict-free shortest-time bidirectional AGV routeing. Int. J. Prod. Res. 29 (12), 2377–2391.

Krishnamurthy, N.N., Batta, R., Karwan, M.H., 1993. Developing conflict-free routes for automated guided vehicles. Oper. Res. 41 (6), 1077–1090.

Ma, H., Li, J., Kumar, T., Koenig, S., 2017. Lifelong multi-agent path finding for online pickup and delivery tasks. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems. pp. 837–845.

Miyamoto, T., Inoue, K., 2016. Local and random searches for dispatch and conflict-free routing problem of capacitated AGV systems. Comput. Ind. Eng. 91, 1–9.

Murakami, K., 2020. Time-space network model and MILP formulation of the conflict-free routing problem of a capacitated AGV system. Comput. Ind. Eng. 141, 106270.

Powell, W.B., 2021. Reinforcement Learning and Stochastic Optimization. John Wiley & Sons, Hoboken, NJ.

Schrijver, A., et al., 2003. Combinatorial optimization: polyhedra and efficiency, vol. 24, (2), Springer.

Sedgewick, R., Wayne, K., 2011. Algorithms, fourth ed. Addison-Wesley Professional.

Sharon, G., Stern, R., Felner, A., Sturtevant, N.R., 2015. Conflict-based search for optimal multi-agent pathfinding. Artificial Intelligence 219, 40–66.

Stern, R., Sturtevant, N., Felner, A., Koenig, S., Ma, H., Walker, T., Li, J., Atzmon, D., Cohen, L., Kumar, T., et al., 2019. Multi-agent pathfinding: Definitions, variants, and benchmarks. In: Proceedings of the International Symposium on Combinatorial Search, vol. 10, (1), pp. 151–158.

Vis, I.F., 2006. Survey of research in the design and control of automated guided vehicle systems. European J. Oper. Res. 170 (3), 677–709.

Warshall, S., 1962. A theorem on boolean matrices. J. ACM 9 (1), 11–12.

Zhong, M., Yang, Y., Dessouky, Y., Postolache, O., 2020. Multi-AGV scheduling for conflict-free path planning in automated container terminals. Comput. Ind. Eng. 142, 106371.