Computers & Industrial Engineering 183 (2023) 109437

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Recovering feasibility in real-time 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:
Logistics
Automated Guided Vehicles
Conflict-free Vehicle Routing and Scheduling
Real-time optimization

ABSTRACT

Conflict-Free Vehicle Routing Problems (CF-VRPs) arise in manufacturing, transportation and logistics facilities where *Automated Guided Vehicles* (AGVs) are utilized to move loads. Unlike *Vehicle Routing Problems* arising in distribution management, CF-VRPs explicitly consider the limited capacity of the arcs of the guide path network to avoid collisions among vehicles. AGV applications have two peculiar features. First, the uncertainty affecting both travel times and machine ready times may result in vehicle delays or anticipations with respect to the fleet nominal plan. Second, the relatively high vehicle speed (in the order of one or two meters per second) requires vehicle plans to be revised in a very short amount of time (usually few milliseconds) in order to avoid collisions. In this paper we present fast exact algorithms to recover plan feasibility in real-time. In particular, we identify two corrective actions that can be implemented in real-time and formulate the problem as a linear program with the aim to optimize four common performance measures (total vehicle delay, total weighted delay, maximum route duration and total lateness). Moreover, we develop tailored algorithms which, tested on randomly generated instances of various sizes, prove to be three orders of magnitude faster than using off-the-shelf solvers.

1. Introduction

Conflict-Free Vehicle Routing and Scheduling Problems (CF-VRPs) arise in manufacturing plants (Ullrich, 2015), warehouses (Van den Berg & Zijm, 1999) and automated port terminals (Schwientek, Lange, & Jahn, 2017; Stahlbock & Voß, 2008) where driverless vehicles (usually referred to as Automated Guided Vehicles, AGVs) are used to move materials, components and finished products, often in the form of palletized and containerized loads. Independently of the guidance system (e.g., magnet spot, laser or GPS navigation), an AGV may be assumed to move along the arcs of a graph whose vertex set represents loading/unloading stations, storage positions as well as intersections of segments of the guide path network. A distinctive feature of these systems is that, in order to avoid collisions among vehicles, a number of non-overlapping constraints have to be imposed. For instance, one may require that at most one vehicle occupies an arc at any given time *or* that vehicles must keep a minimum distance, etc. Relevant contributions to the solution of CF-VRPs have been proposed by Corréa, Langevin, and Rousseau (2007), Desaulniers, Langevin, Riopel, and Villeneuve (2003), Krishnamurthy, Batta, and Karwan (1993) and Miyamoto and Inoue (2016). More recently, Adamo, Bektaş, Ghiani, Guerriero, and Manni (2018) have studied the problem of determining vehicle routes (and vehicle speeds on routes segments) in such a way that no conflict arise, time windows are met, and the total energy consumption is minimized.

In most industrial applications, arc travel times and machine ready times are not known exactly in advance since they depend, to some extent, on a number of variables that cannot be controlled (e.g., enable signals from automatic machines, battery levels, dirt on the floor, etc.) or on unpredictable events (e.g., a worker cutting the road to an AGV, a machine breakdown). Consequently, it may happen that an initially feasible CF-VRP solution (fleet *nominal plan*) becomes unfeasible as a result of the delays (or anticipations) accumulated by some vehicles. The problem studied in this paper is how to recover plan feasibility after such variations have occurred. The objective is to avoid conflicts by implementing some *corrective actions*. We consider two corrective actions, namely (a) imposing a delay to some vehicles and (b) accelerating some vehicles. The aim is to minimize four of the most common performance measures: (1) total vehicle delay, (2) total weighted delay, (3) maximum route duration (*makespan*) and (4) total lateness w.r.t. specified *due dates*.

Before delving into the description of our contribution, we position our paper towards a number of related research lines. The need to coordinate the routes and schedules of a fleet of vehicles to maintain a minimum distance from each other at any time arises in several sectors, including air traffic control and railway traffic management. In *air traffic control*, aircraft move in a shared airspace and cannot get closer to each other than a given safety distance in order 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.cie.2023.109437
Received 24 January 2023; Received in revised form 8 May 2023; Accepted 9 July 2023
Available online 17 July 2023

Table 1
Relevant literature: main contributions.

Application domain	Main references
Air traffic control	Pallottino, Feron, and Bicchi (2002) and Ribeiro, Ellerbroek, and Hoekstra (2020)
Railway traffic management	Cacchiani et al. (2014)
Robot coordination	Akella and Hutchinson (2002), Peng and Akella (2005) and Spensieri, Åblad, Bohlin, Carlson, and Söderberg (2021)
Automated manufacturing plants	Fragapane, De Koster, Sgarbossa, and Strandhagen (2021)

avoid possible conflicts. Ribeiro et al. (2020) review conflict resolution methods for manned and unmanned aviation. In this line of research, a significant contribution is proposed by Pallottino et al. (2002) that consider the path planning problem among given waypoints avoiding all possible conflicts with the objective to minimize the total flight time. The authors propose two formulations as a mixed-integer linear program (MILP): in the first, only velocity changes are admissible maneuvers while in the second only heading angle changes are allowed. Solutions were obtained with standard optimization software. Unlike our contribution, this line of research assumes that aircraft move in a continuous (Euclidean) airspace. Another difference is that several seconds are usually allowed to generate a feasible solution.

In *railway traffic management*, initial relatively small delays of some trains may propagate and generate conflicts. As a result, especially during congested traffic situations where the infrastructure capacity is completely exploited for trains circulation, some trains must be stopped or slowed down to ensure safety, thus generating further delays. If deviations are relatively small, they can be handled by modifying the timetable, with no changes to the duties for rolling stock and crew. On the other hand, massive disruptions of service (due, e.g., to strikes or technical issues) require both the timetable and the duties for rolling stock and crew to be modified. See Cacchiani et al. (2014) for a review. Unlike the AGV applications our paper is motivated by, several minutes are usually allowed to generate a feasible solution which often makes feasible to use standard optimization software.

Finally, our paper is related to the problem of *coordinating the motions of multiple robots* operating in a shared workspace (e.g., robotic arms working in welding and painting workcells) without collisions. In this line of research, Akella and Hutchinson (2002) study the problem when only the robot start times can be varied. They show that, even when the robot trajectories are specified, minimum time coordination of multiple robots is NP-hard, and define a Mixed Integer Linear Programming formulation. This work has been subsequently extended by Peng and Akella (2005) that determine velocity profiles for given paths that obey the kinematic constraints of robots and avoid collisions while minimizing makespan. Unlike the fast exact algorithms presented in this paper, these authors use standard optimization software which make their approach impractical in large-scale real-time AGV applications. For an up-to-date state-of-the-art of robot coordination, see Spensieri et al. (2021). Table 1 provides a summary of the relevant literature.

The main contribution of this paper is the development of tailored algorithms suitable to recover feasibility in *Conflict-Free Vehicle Routing Problems* in industrial settings, like those arising in intralogistics applications (Fragapane et al., 2021). In this application domain, vehicles (AGVs) move at high speed (up to two meters per second) along relatively short “segments” (e.g., aisles) of the guide path network. As a result, when deviations are observed, plans have to be revised in a few milliseconds to avoid collisions among vehicles. In this paper we identify two corrective actions and present fast exact algorithms to recover plan feasibility in real-time under four different objectives. In Section 2, we define the notation used throughout the paper and model the problem. In Section 3, we study the problem of recovering feasibility with proper vehicle corrective delays. In particular, we first present a model and then show that it can be cast as a shortest path problem on a suitably defined auxiliary graph for four distinct objective functions. In Section 4, we extend the previous results to recovering feasibility with both vehicle delays and anticipations. In particular, we show that, under mild hypotheses, the problem can be solved with a two-stage approach in which a shortest path problem is solved at each stage. Finally, in Section 5, we compare experimentally our tailored exact algorithms with solving the proposed formulations with off-the-shelf solvers.

2. Problem definition and notation

Let $G(V, A)$ be the graph representing the internal transportation network of a facility served by a fleet of vehicles. Vertex set V represents loading/unloading stations, storage positions as well as intersections while A describes the segments of the material handling network. In practice, some complicating issues may be present to account for peculiar features of the vehicles (e.g., orientation as it is the case of automated forklifts). However, we neglect these aspects since they do not affect our findings.

In this article, we assume that a *nominal* plan has been previously generated for a given *Conflict-Free Vehicle Routing and Scheduling Problem* on G. As a result, each vehicle has been assigned a route and a schedule, including possible stops at some vertices in V. The individual vehicle plans constitute a nominal plan which is *conflict-free*, and hopefully optimizes some performance measure. We also assume that the position of the vehicles is monitored periodically to identify possible delays or anticipations w.r.t. the nominal plan. If one or more vehicles come up to be early or late, the nominal plan may become unfeasible and some corrective actions need to be promptly identified and implemented in order to avoid collisions.

In large-scale AGV applications, where computing time cannot exceed few milliseconds, feasibility recovery cannot include any route change and the only corrective actions are delaying some vehicles and possibly (depending on AGV technology) speeding up some vehicles.

Let $V_c = \{1, \dots, n\}$ be the set of vehicles and $\xi_h(t)$ the *planned* position of vehicle $h \in V_c$ in G at time t. Moreover, let $\chi_h(t)$ be the *observed* position of vehicle $h \in V_c$ in G at time t. Then, the deviation observed at time t is

$$d_h(t) = t - \xi_h^{-1}(\chi_h(t)).$$

It is worth noting that $d_h(t)$ can be either positive, negative or zero: if $d_h(t) > 0$, vehicle h is late w.r.t. the *nominal* plan; if $d_h(t) < 0$, it is ahead of its schedule; otherwise, it is on time. In what follows, for the sake of simplicity, we omit the time instant t at which the AGV positions were last observed.

For every pair h, k of vehicles, let $s_{h,k}$ (≥ 0) be the maximum delay (*slack*) that vehicle h may accumulate, w.r.t. the nominal plan, without conflicting with vehicle k. Equivalently, $s_{h,k}$ represents the maximum conflict-free anticipation of vehicle k on vehicle h allowed by the current nominal plan. Slacks can be easily computed starting from the nominal plan. Finally, it is worth noting that slacks are not symmetric: e.g., $s_{h,k} = 5$ and $s_{k,h} = +\infty$.

In order to model the relationship among vehicles in the nominal plan, we define a *conflict graph* $G_c(V_c, A_c)$ where vertex $h \in V_c$ represents vehicle h and an arc (h, k) exists in A_c if and only if $s_{h,k} < +\infty$. In this case the arc is assigned a weight equal to $s_{h,k}$. It is worth noting that graph G_c may be cyclic and/or disconnected as shown in Fig. 1.

In order to avoid that deviations d_h may cause vehicle conflicts, we impose to each vehicle $h \in V_c$ a delay or anticipation at the beginning

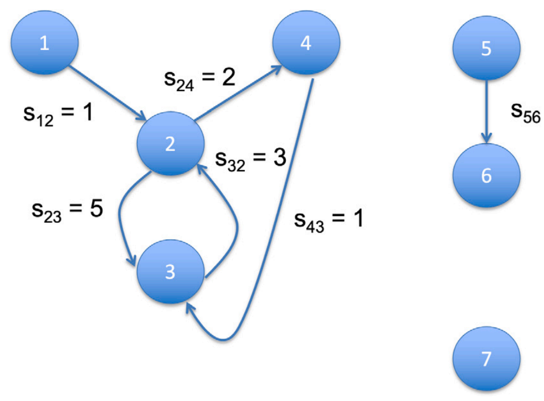


Fig. 1. A conflict graph.

of its route. Let $u_h \in \mathbb{R}$ be the delay/anticipation of vehicle h w.r.t. the nominal plan after feasibility has been recovered. Hence,

$$\delta_h = u_h - d_h$$

represents the *corrective* action imposed to vehicle h. If $\delta_h > 0$, a stop of duration δ_h will be inserted in the plan of the vehicle; if $\delta_h < 0$, vehicle h will be accelerated in such a way as to generate an anticipation of $|\delta_h|$. When a vehicle h does not respect its nominal plan ($d_h \neq 0$), it might be necessary to delay or anticipate other vehicles in order to recover feasibility. For instance, in Fig. 1, if $d_1 = 5$, $d_2 = 1$ and $d_3 = \cdots = d_7 = 0$, the initial delay of vehicles 1 and 2 may be compensated by further delaying vehicle 2 of 3 time units ($\delta_2 = 3$, $u_2 = 4$) which in turns requires vehicle 4 to be delayed of 2 time units ($\delta_4 = u_4 = 2$) which ultimately requires vehicle 3 to be delayed of 1 time units ($\delta_3 = u_3 = 1$). It is worth noting that vehicle 3 is delayed because of the interference between its route and that of vehicle 4 (not vehicle 2 because $s_{23} = 5$ was greater than the difference between $u_2 = 4$ and $u_3 = 0$). Finally, vehicle 5, 6 and 7 were not affected by the initial plan perturbation.

3. Recovering feasibility with corrective delays

We first model the problem of finding an optimal recovery plan under the hypothesis that only delays can be imposed to vehicles. Assuming that the objective is to minimize the sum of vehicle delays, the problem can be formulated as follows:

$$(P_1^D) \quad \min z_1 = \sum_{h \in V_c} u_h \tag{1}$$

s.t.

$$u_h - u_k \leq s_{h,k} \qquad (h,k) \in A_c, \tag{2}$$

$$u_h \geq d_h \qquad h \in V_c, \tag{3}$$

where constraints (2) impose that collisions between pairs of vehicles are avoided, while inequalities (3) require that the total delay of a vehicle is no less that its observed deviation.

A second objective is the minimization of a weighted sum of the vehicle delays. Let $w_h \geq 0$ be the cost of having vehicle h delayed by a time unit. Then, the model can be reformulated as:

$$(P_2^D) \quad \min z_2 = \sum_{h \in V_c} w_h u_h$$

s.t. (2)–(3).

A third objective is the minimization of the longest vehicle route (*makespan*). Let c_h be the completion time of vehicle h in the nominal plan. Then, the model can be rewritten as:

$$(P_3^D) \quad \min z_3 = \max_{h \in V_c} \left(c_h + u_h\right)$$

s.t. (2)–(3),

or equivalently:

$$(P_3^D) \quad \min z_3$$

s.t. (2)–(3), and

$$z_3 \geq c_h + u_h \qquad h \in V_c.$$

Finally, we consider the case in which a *due date* ρ_h is specified for each vehicle $h \in V_c$ and the objective is the minimization of the total *lateness* of the vehicles w.r.t. their *due dates*:

$$(P_4^D) \quad \min z_4 = \sum_{h \in V_c} \max\left(0, u_h - \rho_h\right)$$

s.t. (2)–(3),

or equivalently:

$$(P_4^D) \quad \min z_4 = \sum_{h \in V_c} y_h$$

s.t. (2)–(3), and

$$y_h \geq u_h - \rho_h \qquad h \in V_c,$$

$$y_h \geq 0 \qquad h \in V_c.$$

We now prove some propositions that will be helpful to derive tailored fast exact algorithms for recovering plan feasibility.

Proposition 1. *Problems (P_1^D)-(P_4^D) are feasible for any $d_h \in \mathbb{R}$ and $s_{h,k} \geq 0 \;\; h, k \in V_c$.*

Proof. We prove this proposition for problem (P_1^D). Solution

$$u_h = \max_{i \in V_c} d_i \qquad h \in V_c, \tag{4}$$

makes the left-hand sides of constraints (2) null and satisfies constraints (3). For problems (P_2^D)-(P_4^D), the proof is similar. □

Solution (4) is equivalent to impose to vehicle h an additional delay equal to

$$\delta_h = \max_{i \in V_c} d_i - d_h. \tag{5}$$

For instance, if the first vehicle is 1 unit of time late and the other vehicles are on time ($d_1 = 1$ and $d_h = 0$ for $h = 2, \ldots, n$), then a feasible solution (4) is: $u_1 - d_1 = 0$ and $u_h - d_h = 1$ for $h = 2, \ldots, n$. On the other hand, if the first vehicle is one unit of time ahead and the other vehicles are on time ($d_1 = -1$ and $d_h = 0$ for $h = 2, \ldots, n$), then feasible solution (4) is: $u_1 - d_1 = 1$ and $u_h - d_h = 0$ for $h = 2, \ldots, n$.

Proposition 2. *Problem (P_1^D) is the dual of a one-to-all shortest path problem on a suitably defined auxiliary graph.*

Proof. We define an auxiliary graph $\hat{G}_c = (\hat{V}_c, \hat{A}_c)$ with an additional (source) vertex 0 and $|V_c|$ additional arcs from 0 to any other node $h \in V_c$ having weight equal to $-d_h$. In other terms:

$$\hat{V}_c = V_c \cup \{0\},$$

$$\hat{A}_c = A_c \cup \left\{(0,h) \mid h \in V_c\right\},$$

$$s_{0,h} = -d_h \qquad h \in V_c,$$

$$s_{h,0} = +\infty \qquad h \in V_c.$$

Given a vertex $k \in \hat{V}_c$, we denote with $\delta^-(k) = \{h : (h,k) \in \hat{A}_c\}$ and $\delta^+(k) = \{h : (k,h) \in \hat{A}_c\}$ the set of in-neighbors and out-neighbors of k, respectively. Obviously:

$$|\delta^-(0)| = 0,$$

$$|\delta^+(0)| = |V_c|,$$

$$u_0 = d_0 = 0.$$

Therefore, for any $k \in V_c$:

$$u_k \geq u_h - s_{h,k} \qquad h \in \delta^-(k), \qquad (6)$$

$$u_k \geq d_k. \qquad (7)$$

Constraints (6) and (7) imply

$$u_k \geq \max_{h\in\delta^-(k)} \left(u_h - s_{h,k}\right) \qquad k \in V_c. \qquad (8)$$

In order to minimize (1), the recursive inequality (8) must be satisfied as an equality, i.e.:

$$u^*_k = \max_{h\in\delta^-(k)} \left(u^*_h - s_{h,k}\right) \qquad k \in V_c, \qquad (9)$$

where the asterisk indicates an optimal solution. For $h \neq 0$ relationship (9) can be written as:

$$u^*_k = \max_{h\in\delta^-(k)} \left(\max_{\ell\in\delta^-(h)} \left(u^*_\ell - s_{\ell,h}\right) - s_{h,k}\right) \qquad k \in V_c.$$

By iterating until vertex 0 is reached, the following expression is obtained:

$$u^*_k = \max_{h\in V_c} \max_{p\in\Pi_{hk}} \left(d_h - \sum_{i=1}^{|p|-1} s_{v_i,v_{i+1}} \right) \qquad k \in V_c,$$

where Π_{hk} $(h, k \in V_c)$ denotes the set of all paths in G_c departing from $h \in V_c$ and arriving in $k \in V_c$, and $v_i \in V_c$ is the ith vertex along path $p \in \Pi_{hk}$. Hence, the optimal solution u^*_k corresponds to the shortest path between vertex 0 and vertex k on $\hat{G}_c$. □

Proposition 3. *Let u^*_k $(k \in V_c)$ be an optimal solution for problem (P^D_1). Then, u^*_k $(k \in V_c)$ is also optimal for problems (P^D_2)-(P^D_4) (albeit obviously with different objective function values).*

Proof. Let u^*_k be a feasible solution minimizing z_1.

Case z_2 - Because of (9), solution u^*_k remains optimal if weights w_h are any nonnegative numbers.

Case z_3 - We define a new graph $\bar{G}_c = (\bar{V}_c, \bar{A}_c)$ with an additional (sink) vertex $n+1$ as well as with $|V_c|$ additional arcs going from each vertex $h \in V_c$ to $n+1$ with weight equal to $\max_{k\in V_c} c_k - c_h$:

$$\bar{V}_c = V_c \cup \{n+1\},$$
$$\bar{A}_c = A_c \cup \left\{(h, n+1) \mid h \in V_c\right\},$$
$$s_{h,n+1} = \max_{k\in V_c} c_k - c_h \qquad h \in V_c,$$
$$s_{n+1,h} = +\infty \qquad h \in V_c.$$

By setting $d_{n+1} = 0$, an optimal solution corresponds to the shortest path tree on $\bar{G}_c$ originating in $n+1$. It is worth noting that u^*_{n+1} represents the minimum increase of the maximum completion time, i.e.

$$\min z_3 = \max_{k\in V_c} c_k + u^*_{n+1}.$$

Case z_4 - A new graph $\tilde{G}_c = (\tilde{V}_c, \tilde{A}_c)$ is defined by adding (to G_c) $|V_c|$ additional (sink) vertices and the corresponding incoming arcs as follows:

$$\tilde{V}_c = V_c \cup \{n+1, \ldots, 2n\},$$
$$\tilde{A}_c = A_c \cup \left\{(h, n+h) \mid h \in V_c\right\},$$
$$s_{h,n+h} = \rho_h \qquad h \in V_c,$$
$$s_{n+h,h} = +\infty \qquad h \in V_c.$$

Then, by setting $d_{n+h} = 0$ for $h \in V_c$, an optimal solution corresponds to a one-to-all shortest path tree on $\tilde{G}_c$. It is worth noting that u^*_{n+h} represents the optimal value of variable y_h in the definition of z_4, i.e.:

$$y^*_h = u^*_{n+h}. \quad \square$$

4. Recovering feasibility with corrective delays and anticipations

In the previous section we examined how to recover the feasibility of a nominal plan by imposing corrective delays to vehicles. In this section we consider the case that vehicles may also be accelerated, resulting in corrective anticipations with respect to the current plan. Indeed, we expect that faster vehicles might result in a shorter occupation of tracks, further improving the various performance measures. As before, the aim is to minimize four of the most common performance measures: (1) total vehicle delay, (2) total weighted delay, (3) maximum route duration (*makespan*) and (4) total lateness w.r.t. specified *due dates*.

For the sake of simplicity, we assume there are just two speed levels, a nominal speed v_1 and a higher speed

$$v_2 = k \cdot v_1,$$

with $k > 1$, which can be hold for at most T seconds (to limit energy consumption as well as to reduce the wear and tear of vehicles).

Let x_h be a nonnegative continuous variable representing the corrective anticipation imposed to vehicle h to recover plan feasibility. Two factors impose an upper bound on x_h: T and the time t^c_h after which a conflict would occur if vehicle h continued moving at speed v_1. As for the first factor, the maximum anticipation is achieved by using speed v_2 (instead of v_1) for T instants and amounts to

$$\frac{v_2 T}{v_1} - T = (k-1)T.$$

As for the second factor, the distance from the current position of vehicle h to the first conflict on its route is $v_1 t^c_h$. So speed v_2 can be kept for at most

$$\frac{v_1 t^c_h}{v_2} = \frac{t^c_h}{k}$$

instants, resulting into a maximum anticipation equal to:

$$t^c_h - \frac{t^c_h}{k} = \frac{k-1}{k} t^c_h.$$

Hence, the upper bound on x_h is

$$x_h \leq L_h,$$

where

$$L_h = \min((k-1)T, \frac{k-1}{k} t^c_h).$$

The objective function is defined as the weighted sum, with parameters $\alpha > 0$ and $\beta \geq 0$, of performance measure z_i $(i = 1, \ldots, 4)$ and the total anticipation $\sum_{h\in V_c} x_h$ imposed to vehicles. Problem P^D_i, $(i = 1, \ldots, 4)$ is then re-formulated as follows:

$$(P^{AD}_i) \quad \min \quad z'_i = \alpha z_i + \beta \sum_{h\in V_c} x_h \qquad (10)$$

s.t.

$$u_h - x_h - u_k + x_k \leq s_{h,k} \qquad (h,k) \in A_c, \qquad (11)$$
$$u_h \geq d_h \qquad h \in V_c, \qquad (12)$$
$$0 \leq x_h \leq L_h \qquad h \in V_c, \qquad (13)$$

where constraints (11) prevent collisions between pairs of vehicles, inequalities (12) impose the observed delays/anticipations and constraints (13) define upper bounds on vehicle anticipations. We now prove some propositions.

Proposition 4. *Problems P^{AD}_i $(i = 1, \ldots, 4)$ are feasible for any $\alpha > 0$, $\beta \geq 0$, $d_h \in \mathbb{R}$, $s_{h,k} \geq 0$ and $L_h \geq 0$ $h, k \in V_c$.*

Proof. Solution $x_h = 0$ $(h \in V_c)$ and $u_h = \max_{i\in V_c} d_i$ $(h \in V_c)$ satisfies all constraints. □

Proposition 5. *Each basic feasible solution of formulation (10)–(13) has either $\delta_h = u_h - d_h = 0$ or $x_h = 0$ (or both) for any $h \in V_c$.*

Proof. Let $\mathbf{u}$ be the vector of u_h associated to vehicles $h \in V_c$. We prove the statement by contradiction. Let us assume that it exists an optimal solution with $u_h - d_h \geq x_h > 0$ for a vehicle $h \in V_c$. A new feasible solution can be defined as follows:

$$u'_h = u_h - x_h,$$
$$x'_h = 0,$$

Evaluating both solutions w.r.t. objective function z'_i $(i = 1, \dots, n)$, the following relationship is obtained:

$$\alpha \cdot z_i(\mathbf{u}') + \beta \cdot 0 < \alpha \cdot z_i(\mathbf{u}) + \beta x_h i = 1, \dots, 4.$$

Since the new feasible solution has a lower objective function value than the optimal one, the hypothesis is contradicted. On the other hand, if an optimal solution is such that $x_h \geq u_h - d_h > 0$, then a new feasible solution can be defined as follows:

$$u'_h = d_h,$$
$$x'_h = x_h - u_h + d_h.$$

Once again, evaluating both solutions w.r.t. objective function z'_i $(i = 1, \dots, n)$,

$$\alpha \cdot z_i(\mathbf{u}') + \beta \left(x_h - u_h + d_h\right) < \alpha \cdot z_i(\mathbf{u}) + \beta x_h i = 1, \dots, 4,$$

contradicts the hypothesis. □

Proposition 6. *If $L_h = +\infty$, problem P_i^{AD} $(i = 1, \dots, 4)$ with only anticipations is the dual of a one-to-all shortest path problem on a suitably defined graph.*

Proof. For $u_h = d_h$, formulation (10)–(13) becomes:

$$\alpha z_i + \beta \sum_{h \in V_c} d_h + \beta \min \sum_{h \in V_c} \bar{x}_h \tag{14}$$

s.t.

$$\bar{x}_k - \bar{x}_h \leq s_{h,k} \qquad (h,k) \in A_c \tag{15}$$

$$\bar{x}_h \geq -d_h \qquad h \in V_c \tag{16}$$

where $\bar{x}_h = x_h - d_h$. Therefore, formulation (14)–(16) is equivalent to (1)–(3) formulated on the reverse graph $G_c^{-1} = (V_c, A_c^{-1})$ with

$$A_c^{-1} = \{(k,h) \mid (h,k) \in A_c\},$$

and observed delays/anticipations $-d_h$. □

The following proposition allows to decompose problem P_i^{AD} $(i = 1, \dots, 4)$ in case minimizing z_i is more of a priority than minimizing the sum of the x_h $(h \in V_c)$ variables.

Proposition 7. *If $\alpha \gg \beta \geq 0$ (in particular, if $\beta = 0$), problem P_i^{AD} $(i = 1, \dots, 4)$ decomposes into two independent (one-to-all shortest path) subproblems.*

Proof. Initially, we prescribe that each vehicle is imposed an anticipation equal to the maximum allowed (e.g., $x_h = L_h$). Then formulation (10)–(13) becomes:

$$\alpha \min z_i + \beta \sum_{h \in V_c} L_h \tag{17}$$

s.t.

$$\bar{u}_h - \bar{u}_k \leq s_{h,k} \qquad (h,k) \in A_c \tag{18}$$

$$\bar{u}_h \geq d_h - L_h \qquad h \in V_c \tag{19}$$

where $\bar{u}_h = u_h - L_h$. Problem (17)–(19) is equivalent to problem (1)–(3) on a graph coincident with G_c, except that observed delays/anticipations are equal to $d_h - L_h$. Hence, it can be solved as a one-to-all shortest path problem. The corresponding optimal corrective delays $\delta_h = \bar{u}_h - d_h + L_h$ are feasible for (10)–(13). Moreover, if $\beta = 0$ they are also optimal. Otherwise, this solution has to be corrected in order to minimize the total anticipation. To do so, it has to be noted that, because of Proposition 5, the overall corrective delay determined by model (17)–(19) is the optimal one also for (10)–(13), i.e. $\delta_h^* = \max\{0, \bar{u}_h - d_h \cancel{+L_h} \cancel{-L_h}\}$, while the overall corrective anticipation will be $x_h = \max\{0, \cancel{L_h} - \bar{u}_h + d_h \cancel{-L_h}\}$.

However, because the anticipations x_h $(h \in V_c)$ prescribed in (10)–(13) can be larger than the optimal ones, this formulation with $u_h = \delta_h^* + d_h$ becomes:

$$\alpha z_i + \beta \sum_{h \in V_c} (\delta_h^* + d_h) + \beta \min \sum_{h \in V_c} \bar{x}_h \tag{20}$$

s.t.

$$\bar{x}_k - \bar{x}_h \leq s_{h,k} \qquad (h,k) \in A_c \tag{21}$$

$$\bar{x}_h \geq -\delta_h^* - d_h \qquad h \in V_c \tag{22}$$

where corrective actions are $\bar{x}_h + \delta_h^* + d_h = x_h$. Solution of model (17)–(19) ensures feasibility of model (20)–(22). Moreover,

$$\bar{x}_h + \delta_h^* + d_h \leq L_h$$

is surely satisfied by that solution. Then applying Proposition 6 the thesis is proved. □

Summing up, if $\beta = 0$, only (17)–(19) has to be solved. Otherwise, it is required to solve (20)–(22) in a second stage. Both problems are equivalent to solving a one-to-all shortest path problem.

Algorithms 2 and 3 summarize the tailored procedures used to solve problems P_i^D and P_i^{AD} $(i = 1, \dots, 4)$. In particular, Algorithm 2 solves problems P_i^D $(i = 1, \dots, 4)$: first, graph $\hat{G}_c$ is constructed as shown in the proof of Proposition 2; then, Dijkstra's shortest path procedure (Algorithm 1) is executed on $\hat{G}_c$. In contrast, as demonstrated in Proposition 7, Algorithm 3 is made up of two stages: the first executes Algorithm 2 on graph G_c, using the slack matrix and imposing both the observed delay and the maximum allowed anticipation; then, a second call to Algorithm 2 on the reverse graph G_c^{-1}, using the transpose of slack matrix is able to compute the optimal solution of problems P_i^{AD} $(i = 1, \dots, 4)$.

Algorithm 1 Dijkstra's Algorithm

```
1:  function ShortestPath($G_c$, $s$, $\sigma$)          ▷ $\sigma$ is the source node
2:      $Q \leftarrow \emptyset$                         ▷ queue of unvisited vertices
3:      for $h \in V_c$ do
4:          $\tau_h \leftarrow +\infty$                  ▷ distance from $\sigma$ to $h$
5:          add $h$ to $Q$
6:      $\tau_\sigma \leftarrow 0$
7:      while $Q$ is not empty do
8:          $h \leftarrow \arg\min_{k \in Q} \tau_k$
9:          $Q \leftarrow Q \setminus \{h\}$
10:         for $(h,k) \in A_c$ do                     ▷ go through each outgoing arc of $h$
11:             if $\tau_k > \tau_h + s_{h,k}$ then
12:                 $\tau_k \leftarrow \tau_h + s_{h,k}$
13:     return $\tau$
```

Algorithm 2 SP_D Algorithm

```
1:  function $SP_D$($G_c$, $s$, $d$)
2:      Build graph $\hat{G}_c = (\hat{V}_c, \hat{A}_c)$ using $d$
3:      $\tau \leftarrow$ ShortestPath($\hat{G}_c$, $s$, 0)
4:      for $h \in V_c$ do
5:          $u_h^* \leftarrow -\tau_h$
6:      return $u^*$
```

Table 2
Computational results — minimizing z_1 (total delay).

Instances		Corrective delays					Corrective anticipations and delays					DEV
p	n	SP_D [ms]	$CPLEX_D$ [ms]	GUR_D [ms]	$SCIP_D$ [ms]	z_1 [s]	SP_{AD} [ms]	$CPLEX_{AD}$ [ms]	GUR_{AD} [ms]	$SCIP_{AD}$ [ms]	z_1 [s]	[%]
0	50	0.01	150.02	92.94	85.84	493.3	0.03	204.89	94.53	127.87	398.2	19.0
	100	0.02	577.64	401.54	306.15	999.9	0.08	606.52	404.01	489.50	839.9	16.5
	150	0.05	1316.73	890.87	679.56	1500.0	0.13	1399.28	908.74	1476.53	1290.1	13.8
	200	0.08	2347.16	1618.81	1201.17	2000.0	0.18	2713.41	1723.72	3185.72	1740.0	12.8
	250	0.12	3928.97	2537.23	1901.89	2500.0	0.26	4598.07	2661.80	5718.10	2225.0	11.0
	300	0.16	6601.63	3630.84	2715.84	3000.0	0.37	6702.86	3875.14	8978.28	2700.0	10.0
0.25	50	0.01	106.05	71.84	70.99	485.8	0.03	146.69	74.86	94.45	370.7	23.5
	100	0.03	443.90	285.37	231.14	999.5	0.08	477.93	289.47	404.10	849.6	14.8
	150	0.05	949.17	667.73	523.53	1500.0	0.13	1049.86	718.26	1010.91	1320.0	11.7
	200	0.07	1728.66	1233.89	924.57	2000.0	0.19	1875.75	1338.95	1712.98	1760.0	11.8
	250	0.10	2729.27	1906.91	1452.90	2500.0	0.27	3043.46	2084.10	3495.32	2175.0	12.9
	300	0.15	4058.68	2768.89	2162.66	3000.0	0.39	4264.78	3125.10	4468.80	2700.0	9.9
0.50	50	0.01	90.15	45.42	53.89	468.9	0.03	84.00	47.21	67.71	345.6	26.4
	100	0.03	324.10	203.79	213.07	987.4	0.09	318.45	207.61	254.37	837.9	16.0
	150	0.06	683.22	485.03	374.80	1499.5	0.14	722.09	480.22	765.84	1334.7	10.9
	200	0.09	1193.58	910.87	647.02	1999.8	0.21	1297.74	925.53	1282.30	1739.8	13.0
	250	0.13	2056.36	1404.97	1111.71	2500.0	0.31	1937.82	1468.08	1888.44	2250.0	9.9
	300	0.16	2935.56	1940.26	1595.95	3000.0	0.43	2955.88	2182.70	2785.92	2640.1	11.8
0.75	50	0.01	45.33	24.71	30.39	398.5	0.03	51.35	29.23	42.16	298.7	25.8
	100	0.04	150.79	98.66	106.25	952.8	0.10	188.10	112.00	146.78	801.3	15.4
	150	0.07	342.28	223.98	230.57	1488.2	0.18	405.06	238.46	358.31	1249.1	15.8
	200	0.10	622.10	469.56	431.28	1994.0	0.25	729.30	478.98	813.81	1794.3	10.1
	250	0.17	1071.23	790.28	678.36	2496.9	0.38	1139.11	787.92	1299.99	2246.9	10.0
	300	0.19	1440.69	1183.42	945.51	2998.8	0.48	1556.97	1077.57	1921.71	2638.8	12.1
AVERAGE		**0.08**	**1495.55**	**995.33**	**778.13**	**1740.1**	**0.20**	**1602.88**	**1055.59**	**1782.91**	**1522.7**	**14.4**

Algorithm 3 SP_{AD} Algorithm

```
1: function $SP_{AD}(G_c, s, d, L)$
2:     for $h \in V_c$ do
3:         $d'_h \leftarrow d_h - L_h$
4:     $\bar{u} \leftarrow SP_D(G_c, s, d')$
5:     for $h \in V_c$ do
6:         $\delta^*_h \leftarrow \max_{h \in V_c}\{0, \bar{u}_h - d_h\}$
7:         $d''_h \leftarrow -\left(\delta^*_h + d_h\right)$
8:     $\bar{x} \leftarrow SP_D(G_c^{-1}, s^t, d'')$
9:     for $h \in V_c$ do
10:        if $\delta^*_h > 0$ then
11:            $u^\star_h \leftarrow \bar{u}_h$
12:            $x^\star_h \leftarrow 0$
13:        else
14:            $u^\star_h \leftarrow 0$
15:            $x^\star_h \leftarrow \bar{x}_h - d''_h$
16:    return $u^*, x^*$
```

5. Computational results

The aim of our computational experiments was to assess whether our approach can be valuable to recover plan feasibility in a real-time setting like those arising when using AGVs in manufacturing, transportation and logistics applications. In particular, we compared the performance of the tailored approach described in Sections 3 and 4 versus finding exact solutions to problems $P_1^D - P_4^D$ and $P_1^{AD} - P_4^{AD}$ with three off-the-shelf solvers: commercial solvers IBM ILOG CPLEX 22.1 (Nickel, Steinhardt, Schlenker, & Burkart, 2022) and Gurobi 10.0.1 (Gurobi Optimization, LLC, 2023) as well as open-source solver SCIP 8.0 (Bestuzheva et al., 2021).

All the experiments were run on a standalone Linux machine with an Intel Core i7 processor composed by 4 cores clocked at 2.5 GHz and equipped with 16 GB of RAM. All algorithms have been coded in C++. Shortest paths on auxiliary graphs were determined by the Dijkstra's algorithm (Dijkstra et al., 1959) with a Fibonacci heap min-priority queue.

We first describe the generation of input data, and then present the results. Instances have been randomly generated with a number of vehicles $n = |V_c| = 50, 100, 150, 200, 250, 300$. A parameter p, defined as

$$p = \frac{|A_c|}{n^2 - n} - 1,$$

was used to control the sparsity level of the conflict graph. In particular, p was set equal to $\{0, 0.25, 0.50, 0.75\}$. Vehicle (observed) deviations from the nominal plans d_h ($n = 1, \dots, n$) were randomly generated according to a uniform distribution in the $[-10, 10]$ range. Vehicle slacks $s_{h,k}$ ($(h,k) \in A_c$) were uniformly generated in interval $[0, 13]$. As far as z_2 is concerned, weights w_h ($n = 1, \dots, n$) were uniformly generated in $[0, 1]$. As for z_3, completion times c_h ($n = 1, \dots, n$) were derived from a uniform distribution in $[100, 110]$. Regarding objective function z_4, due dates ρ_h ($n = 1, \dots, n$) were uniformly generated in interval $[0, 10]$. Finally, when corrective anticipations are allowed, we set $\alpha = 1000$ and $\beta = 1$. For any pair (n, p) we generated 10 instances, for a total of 240 instances. All test files are available at https://tdrouting.com/cfpdp.

The results of our experiments are reported in Tables 2 to 5 for objective functions z_1 to z_4 (assumed to be expressed in seconds). In all tables, the first two columns are self-explanatory and the values reported are averaged across all instances. The remaining column headings are as follows:

- SP_D: time (in milliseconds) spent by the tailored approach whenever only corrective delays are allowed;
- $CPLEX_D$: time (in milliseconds) spent by the IBM ILOG CPLEX 22.1 solver on problem P_i^D ($i = 1, \dots, 4$);
- GUR_D: time (in milliseconds) spent by the Gurobi 10.0.1 solver on problem P_i^D ($i = 1, \dots, 4$);
- $SCIP_D$: time (in milliseconds) spent by the SCIP 8.0 solver on problem P_i^D ($i = 1, \dots, 4$);
- SP_{AD}: time (in milliseconds) spent by the tailored approach whenever both corrective anticipations and corrective delays are allowed;
- $CPLEX_{AD}$: time (in milliseconds) spent by the IBM ILOG CPLEX 22.1 solver on problem P_i^{AD} ($i = 1, \dots, 4$);

Table 3
Computational results — minimizing z_2 (total weighted delay).

Instances		Corrective delays					Corrective anticipations and delays					DEV
p	n	SP_D [ms]	$CPLEX_D$ [ms]	GUR_D [ms]	$SCIP_D$ [ms]	z_2 [s]	SP_{AD} [ms]	$CPLEX_{AD}$ [ms]	GUR_{AD} [ms]	$SCIP_{AD}$ [ms]	z_2 [s]	[%]
0	50	0.01	140.65	85.64	90.40	2146.7	0.03	151.53	87.72	121.86	1728.4	19.1
	100	0.02	570.53	395.23	295.62	4544.1	0.08	594.22	396.93	469.01	3799.3	16.0
	150	0.05	1306.04	895.74	656.59	6641.0	0.13	1359.35	922.85	1400.32	5708.4	14.0
	200	0.08	2367.73	1599.47	1175.33	8831.0	0.18	2455.25	1702.93	2536.37	7679.4	13.0
	250	0.12	3965.93	2529.78	1853.67	11 310.0	0.26	3901.06	2668.21	4474.88	10 066.8	11.0
	300	0.16	6656.24	3615.33	2673.13	13 580.0	0.37	5685.01	3866.96	7172.66	12 222.0	10.0
0.25	50	0.01	115.33	66.05	70.60	2196.0	0.03	119.47	68.88	87.59	1675.8	23.7
	100	0.03	439.93	301.90	232.99	4441.8	0.08	451.78	307.89	375.09	3781.7	15.0
	150	0.05	1015.91	689.58	507.37	7089.0	0.13	1047.64	741.48	938.06	6234.9	12.0
	200	0.07	1769.17	1235.85	910.15	9102.0	0.19	1836.91	1345.95	1597.90	8009.5	12.0
	250	0.10	2787.87	1895.44	1406.95	11 428.0	0.27	2892.88	2079.45	3031.46	9937.3	13.0
	300	0.15	4024.27	2784.84	2038.38	13 423.0	0.39	4159.17	3155.83	4226.66	12 080.7	10.0
0.50	50	0.01	108.66	45.16	52.40	2103.3	0.03	112.88	46.82	63.06	1565.1	26.3
	100	0.03	289.31	190.20	180.71	4353.5	0.09	302.22	196.87	248.92	3696.1	15.1
	150	0.06	652.11	486.64	361.32	6731.4	0.14	684.88	483.98	697.68	5993.9	11.0
	200	0.09	1169.77	914.51	616.04	8994.2	0.21	1228.86	932.04	1164.55	7835.3	13.0
	250	0.13	1844.79	1410.12	960.09	11 316.0	0.31	1938.11	1483.69	1830.03	10 184.4	10.0
	300	0.16	2670.26	1927.98	1382.90	13 566.0	0.43	2798.06	2186.93	2447.01	11 924.7	12.0
0.75	50	0.01	40.88	23.83	27.18	1755.7	0.03	43.21	29.27	36.93	1304.6	25.6
	100	0.04	139.51	97.55	100.16	4400.1	0.10	146.83	112.41	134.53	3692.0	15.9
	150	0.07	332.19	242.16	224.70	6668.7	0.18	349.67	256.54	312.36	5605.3	16.1
	200	0.10	587.61	477.33	409.05	9062.8	0.25	616.49	479.04	646.27	8155.7	10.0
	250	0.14	915.13	783.22	580.78	11 101.9	0.38	961.58	780.43	1133.95	9990.5	10.0
	300	0.19	1351.54	1234.12	818.13	13 507.4	0.48	1394.44	1156.14	1538.65	11 902.6	12.0
AVERAGE		**0.08**	**1469.22**	**996.99**	**734.36**	**7845.6**	**0.20**	**1467.98**	**1062.05**	**1528.57**	**6865.6**	**14.4**

Table 4
Computational results — minimizing z_3 (maximum route duration).

Instances		Corrective delays					Corrective anticipations and delays					DEV
p	n	SP_D [ms]	$CPLEX_D$ [ms]	GUR_D [ms]	$SCIP_D$ [ms]	z_3 [s]	SP_{AD} [ms]	$CPLEX_{AD}$ [ms]	GUR_{AD} [ms]	$SCIP_{AD}$ [ms]	z_3 [s]	[%]
0	50	0.01	141.80	87.54	87.19	118.9	0.03	157.64	87.55	92.15	117.1	1.5
	100	0.02	582.44	399.74	297.14	119.0	0.08	604.55	393.82	345.81	117.4	1.3
	150	0.05	1305.45	900.81	670.45	119.0	0.13	1356.92	887.16	954.05	117.6	1.2
	200	0.08	2357.89	1605.21	1184.54	119.0	0.18	2444.40	1678.63	1655.04	117.7	1.1
	250	0.12	3977.72	2542.16	1862.65	119.0	0.26	3848.72	2575.55	2558.27	117.9	0.9
	300	0.16	6697.59	3628.38	2692.51	119.0	0.37	5495.73	3787.91	3679.66	118.0	0.8
0.25	50	0.01	134.99	67.00	72.70	118.8	0.03	141.89	68.32	82.47	116.5	1.9
	100	0.03	440.30	297.18	238.24	119.0	0.08	446.43	297.49	316.78	117.5	1.3
	150	0.05	967.83	698.44	511.27	119.0	0.13	1006.07	744.86	895.67	117.8	1.0
	200	0.07	1769.53	1234.61	914.28	119.0	0.19	1834.56	1348.93	1488.75	117.8	1.0
	250	0.10	2776.47	1896.80	1421.74	119.0	0.27	2880.65	2065.35	2420.99	117.7	1.1
	300	0.15	4014.25	2770.62	2066.91	119.0	0.39	4173.16	3094.77	3453.27	118.0	0.8
0.50	50	0.01	74.39	46.78	54.60	118.6	0.03	77.33	47.41	64.33	116.3	1.9
	100	0.03	298.07	203.78	184.66	118.9	0.09	308.33	203.97	245.71	117.5	1.2
	150	0.06	659.08	557.61	358.32	119.0	0.14	687.23	544.56	659.39	117.9	0.9
	200	0.09	1180.71	903.99	620.85	119.0	0.21	1226.90	909.93	1110.35	117.7	1.1
	250	0.13	1866.16	1405.72	967.59	119.0	0.31	1939.64	1501.88	1695.69	118.0	0.8
	300	0.16	2672.27	1924.92	1391.01	119.0	0.43	2792.27	2183.45	2502.10	117.8	1.0
0.75	50	0.01	61.83	25.26	30.04	118.1	0.03	63.28	29.84	37.07	116.7	1.2
	100	0.04	196.10	107.55	105.34	118.8	0.10	205.19	121.72	138.50	117.3	1.3
	150	0.07	342.35	228.75	237.18	119.0	0.18	352.69	235.93	321.32	117.6	1.2
	200	0.10	596.75	474.68	398.47	119.0	0.25	617.89	484.01	655.02	118.0	0.8
	250	0.14	928.92	778.74	608.53	119.0	0.38	959.61	791.68	1052.39	118.0	0.8
	300	0.19	1366.83	1153.92	842.93	119.0	0.48	1387.70	1091.32	1650.93	117.8	1.0
AVERAGE		**0.08**	**1475.40**	**997.51**	**742.46**	**118.9**	**0.20**	**1458.70**	**1049.00**	**1169.82**	**117.6**	**1.1**

- GUR_{AD}: time (in milliseconds) spent by the Gurobi 10.0.1 solver on problem P_i^{AD} $(i = 1, \dots, 4)$;
- $SCIP_{AD}$: time (in milliseconds) spent by the SCIP 8.0 solver on problem P_i^{AD} $(i = 1, \dots, 4)$;
- DEV_i: percentage deviation between the z_i values of the solutions with and without corrective anticipations allowed, i.e.

$$DEV_i = 100 \cdot \frac{z_i^D - z_i^{AD}}{z_i^D},$$

where z_i^{AD} and z_i^D are the values of objective function z_i when recovering feasibility with and without corrective anticipations, respectively.

Above all, computational results show that, in terms of computing time, our tailored exact approach outperformed CPLEX, Gurobi and SCIP solvers, independently of the objective function. This trend is shown in Figs. 2 and 3. Even for small fleet sizes (n = 50), our procedures were able to recover feasibility in less than a millisecond while the CPLEX, Gurobi and SCIP solvers took something between 40

Table 5
Computational results — minimizing z_4 (total lateness).

Instances		Corrective delays					Corrective anticipations and delays					DEV
p	n	SP_D [ms]	$CPLEX_D$ [ms]	GUR_D [ms]	$SCIP_D$ [ms]	z_4 [s]	SP_{AD} [ms]	$CPLEX_{AD}$ [ms]	GUR_{AD} [ms]	$SCIP_{AD}$ [ms]	z_4 [s]	[%]
0	50	0.01	153.23	88.16	87.77	274.7	0.03	163.05	88.30	95.94	187.9	31.5
	100	0.02	588.34	397.62	300.91	557.9	0.08	617.64	397.85	347.33	408.9	26.5
	150	0.05	1319.74	909.02	668.03	820.4	0.13	1380.71	902.03	947.73	616.2	25.0
	200	0.08	2373.19	1623.19	1192.34	1091.7	0.18	2467.96	1720.02	1663.02	838.9	23.1
	250	0.12	3984.24	2544.70	1871.59	1348.8	0.26	3871.13	2696.05	2572.47	1076.7	20.3
	300	0.16	6749.59	3614.15	2707.37	1643.7	0.37	5574.29	3852.75	3674.64	1343.7	18.3
0.25	50	0.01	113.05	67.37	71.92	259.4	0.03	117.086	68.89	87.87	156.2	39.2
	100	0.03	458.40	307.17	237.54	552.7	0.08	466.38	311.29	326.79	408.7	25.9
	150	0.05	1020.14	684.64	516.81	841.6	0.13	1056.35	765.90	908.94	665.3	20.9
	200	0.07	1757.49	1243.78	944.92	1097.0	0.19	1831.17	1417.05	1682.39	860.5	21.6
	250	0.10	2785.57	1890.69	1434.06	1379.2	0.27	2913.65	2115.10	3283.40	1060.0	23.2
	300	0.15	4020.14	2786.59	2070.30	1636.8	0.39	4194.53	3166.89	4055.17	1336.8	18.3
0.50	50	0.01	74.26	47.28	54.09	237.8	0.03	78.17	48.04	68.48	133.2	44.3
	100	0.03	295.42	202.25	184.98	538.9	0.09	306.28	204.48	266.42	397.7	26.2
	150	0.06	660.25	488.60	363.06	790.5	0.14	691.13	501.05	708.29	627.8	20.7
	200	0.09	1190.09	919.18	626.48	1102.5	0.21	1246.89	961.80	1186.81	847.5	23.2
	250	0.13	1849.15	1400.96	980.66	1374.9	0.31	1939.44	1555.88	1908.06	1124.9	18.2
	300	0.16	2696.32	1936.78	1405.35	1652.0	0.43	2829.06	2365.82	2654.74	1297.5	21.5
0.75	50	0.01	43.19	25.86	29.39	172.0	0.03	45.024	30.46	43.01	107.9	38.0
	100	0.04	143.70	101.61	102.76	494.6	0.10	151.06	116.56	151.51	358.9	27.1
	150	0.07	342.50	223.57	232.38	830.2	0.18	356.11	236.55	345.88	603.5	27.3
	200	0.10	600.56	478.75	414.04	1118.2	0.25	626.08	512.17	712.76	918.9	17.8
	250	0.14	934.83	799.88	602.25	1357.9	0.38	970.00	802.41	1143.79	1108.5	18.4
	300	0.19	1367.17	1245.75	835.19	1663.7	0.48	1401.90	1205.87	1708.89	1309.5	21.3
AVERAGE		**0.08**	**1480.02**	**1001.15**	**747.26**	**951.5**	**0.20**	**1470.63**	**1085.13**	**1272.68**	**741.5**	**24.9**

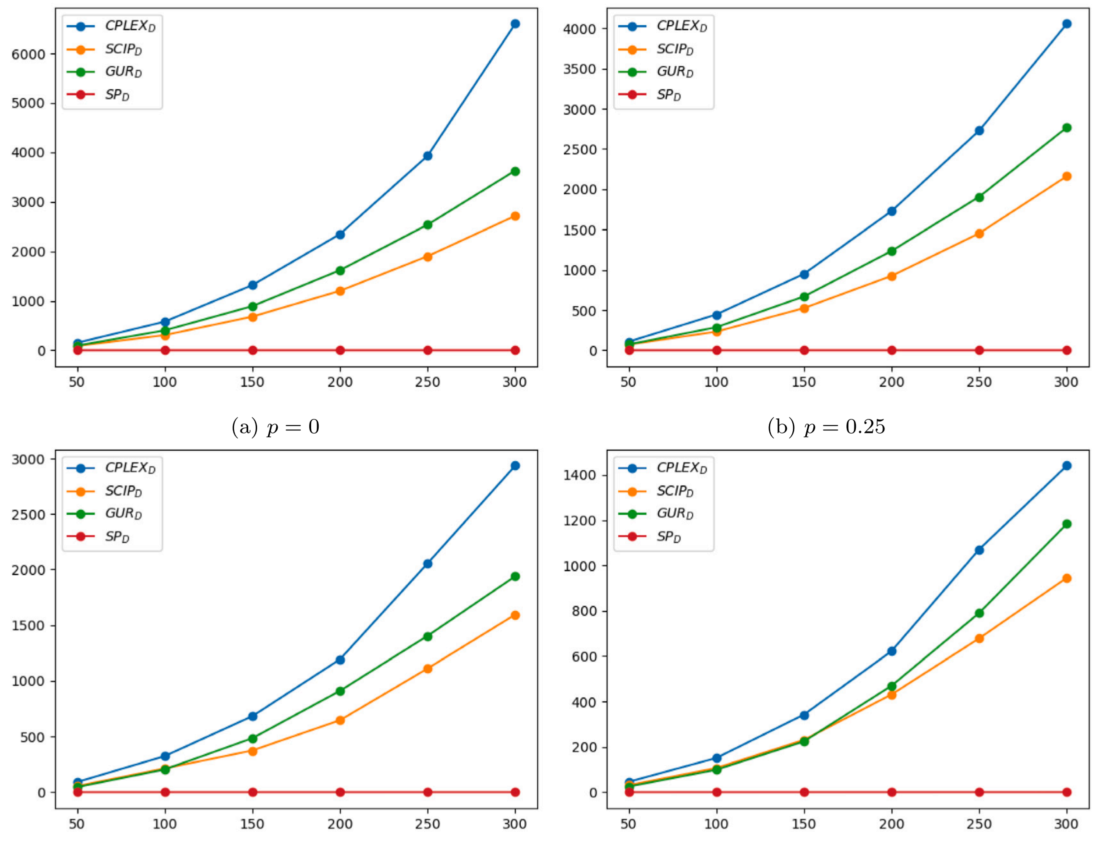
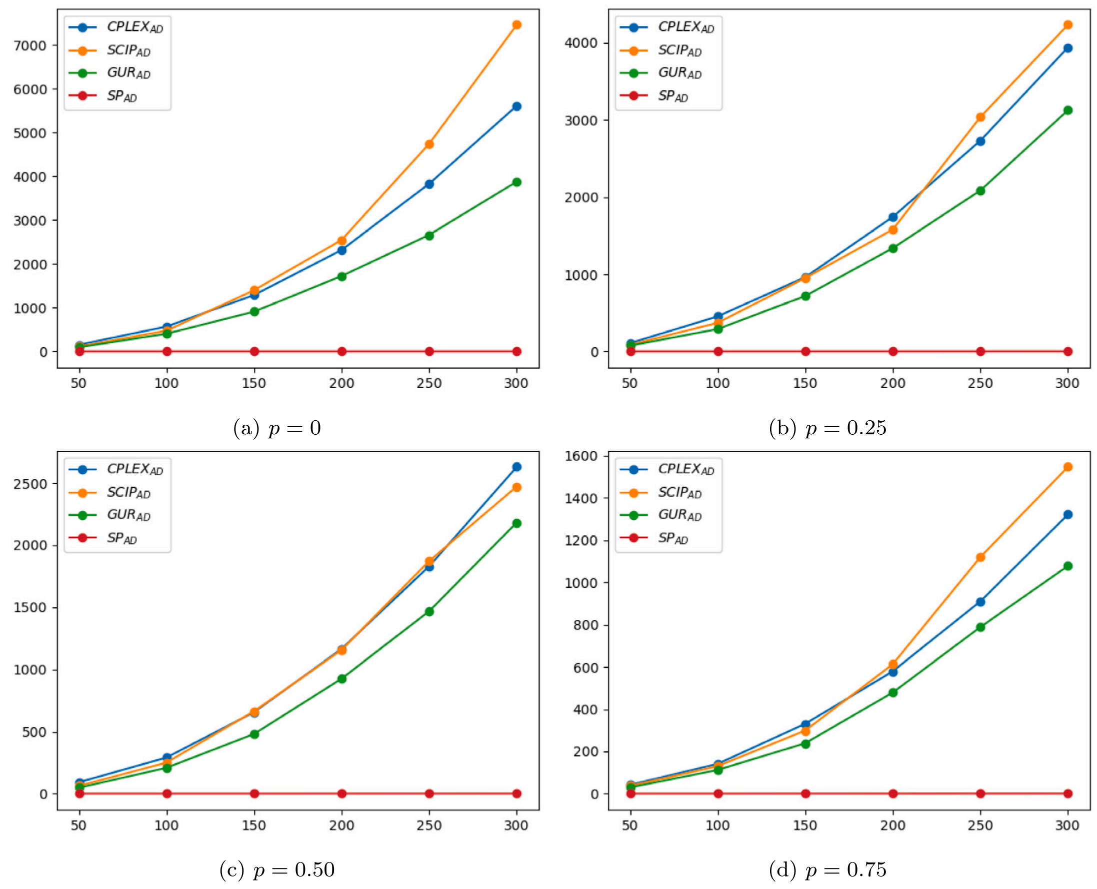


Fig. 2. Times comparison: recovering feasibility with corrective delays.



(a) $p = 0$ (b) $p = 0.25$ (c) $p = 0.50$ (d) $p = 0.75$

Fig. 3. Times comparison: recovering feasibility with corrective anticipations and delays.

and 200 ms, depending mainly on the sparsity of the conflict graph and less on the objective function. For moderate fleet sizes (n = 100) the approach based on off-the-shelf solvers became no more viable (with a computing time between 100 and 600 ms) while our approach was still able to recover feasibility in less than a millisecond. The performance gap became even more relevant for large fleets in which case the approach based on off-the-shelf solvers became impractical.

More in detail, Tables 2–5 show that, as a rule, computing times were lower for sparser conflict graphs (p = 0.75). For instance, when minimizing z_1 with corrective delays, CPLEX took 612.07 ms on average to obtain the optimal solution with p = 0.75 versus 2487.02 ms with $p = 0$. This trend was confirmed for z_2, z_3 and z_4, with or without corrective anticipations.

SCIP generally requires less time to solve models without anticipations; in contrast, Gurobi outperforms the other solvers when corrective anticipations are allowed.

Another aspect to be considered is that on average both SP_D and SP_{AD} were not affected by the objective function to be minimized. Moreover, SP_{AD} was 2.5 times greater than SP_D, while still remaining very moderate (at most 0.48 ms for n = 300).

From a managerial point of view, allowing vehicles to speed up to avoid conflicts was beneficial in terms of all the objective functions. However, while the reductions in total delay, total weighted delay and lateness were significant (14.4%, 14.4%, 24.90% on average, respectively), the impact on makespan minimization was negligible (1.1%). These reductions might result in comparable improvements of a number of key performance indicators such as throughput. However, it is difficult to quantify precisely these variations without any reference to a real application setting.

6. Conclusions

In this paper we have dealt with recovering plan feasibility in *Conflict-Free Vehicle Routing and Scheduling Problems* whenever some vehicles are ahead or behind of schedule. The problem is of the outmost importance in manufacturing, transportation and logistics facilities where AGVs are utilized to move loads between stations. In such settings, vehicles move at speeds in the order of one-two meter per second and feasibility has to be recovered in a few milliseconds. In this paper we have presented fast exact algorithms to solve this problem with respect to two corrective actions (introducing corrective delays/anticipations) with the objective to optimize four common performance measures (total vehicle delay, total weighted delay, maximum route duration and total lateness). An extensive empirical study has shown that, in terms of computing time, our tailored exact algorithms are at least three orders of magnitude faster than IBM ILOG CPLEX 22.1 (Nickel et al., 2022), Gurobi 10.0.1 (Gurobi Optimization, LLC, 2023) and SCIP 8.0 (Bestuzheva et al., 2021) solvers and are suitable for large intralogistics applications.

CRediT authorship contribution statement

Tommaso Adamo: Conceptualizaion, Methodology, Formal analysis, Writing – review & editing, Software. **Gianpaolo Ghiani:** Conceptualization, Methodology, Formal analysis, Writing – review & editing, Supervision. **Emanuela Guerriero:** Conceptualization, Methodology, Formal analysis, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

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).

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. *Transportation Science*, *52*(4), 739–755.

Akella, S., & Hutchinson, S. (2002). Coordinating the motions of multiple robots with specified trajectories. In *Proceedings 2002 IEEE international conference on robotics and automation (Cat. no. 02CH37292), Vol. 1* (pp. 624–631). IEEE.

Bestuzheva, K., Besançon, M., Chen, W.-K., Chmiela, A., Donkiewicz, T., van Doornmalen, J., et al. (2021). *The SCIP optimization suite 8.0*: *Technical Report*, Optimization Online.

Cacchiani, V., Huisman, D., Kidd, M., Kroon, L., Toth, P., Veelenturf, L., et al. (2014). An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research, Part B (Methodological)*, *63*, 15–37.

Corréa, A. I., Langevin, A., & Rousseau, L.-M. (2007). Scheduling and routing of automated guided vehicles: A hybrid approach. *Computers & Operations Research*, *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. *International Journal of Flexible Manufacturing Systems*, *15*(4), 309–331.

Dijkstra, E. W., et al. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269–271.

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 Journal of Operational Research*, *294*(2), 405–426.

Gurobi Optimization, LLC (2023). Gurobi optimizer reference manual.

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

Miyamoto, T., & Inoue, K. (2016). Local and random searches for dispatch and conflict-free routing problem of capacitated AGV systems. *Computers & Industrial Engineering*, *91*, 1–9.

Nickel, S., Steinhardt, C., Schlenker, H., & Burkart, W. (2022). IBM ILOG CPLEX Optimization Studio—A primer. In *Decision optimization with IBM ILOG CPLEX optimization studio: A hands-on introduction to modeling with the optimization programming language (OPL)* (pp. 9–21). Springer.

Pallottino, L., Feron, E. M., & Bicchi, A. (2002). Conflict resolution problems for air traffic management systems solved with mixed integer programming. *IEEE Transactions on Intelligent Transportation Systems*, *3*(1), 3–11.

Peng, J., & Akella, S. (2005). Coordinating multiple robots with kinodynamic constraints along specified paths. *International Journal of Robotics Research*, *24*(4), 295–310.

Ribeiro, M., Ellerbroek, J., & Hoekstra, J. (2020). Review of conflict resolution methods for manned and unmanned aviation. *Aerospace*, *7*(6), 79.

Schwientek, A. K., Lange, A.-K., & Jahn, C. (2017). Literature classification on dispatching of container terminal vehicles. In *Digitalization in maritime and sustainable logistics: City logistics, port logistics and sustainable supply chain management in the digital age. Proceedings of the Hamburg international conference of logistics (HICL), Vol. 24* (pp. 3–36). Berlin: epubli GmbH.

Spensieri, D., Åblad, E., Bohlin, R., Carlson, J. S., & Söderberg, R. (2021). Modeling and optimization of implementation aspects in industrial robot coordination. *Robotics and Computer-Integrated Manufacturing*, *69*, Article 102097.

Stahlbock, R., & Voß, S. (2008). Vehicle routing problems and container terminal operations–an update of research. In *The vehicle routing problem: Latest advances and new challenges* (pp. 551–589). Springer.

Ullrich, G. (2015). *Automated guided vehicle systems, Vol. 10*. Springer-Verlag Berlin Heidelberg, 978–3.

Van den Berg, J. P., & Zijm, W. H. (1999). Models for warehouse management: Classification and examples. *International Journal of Production Economics*, *59*(1–3), 519–528.