


Article

Connected and Autonomous Vehicle Scheduling Problems: Some Models and Algorithms

Evgeny R. Gafarov ^{1,†} and Frank Werner ^{2,*,†} 

¹ V.A. Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Profsoyuznaya St. 65, Moscow 117997, Russia; axel73@mail.ru

² Faculty of Mathematics, Otto-von-Guericke-Universität Magdeburg, PSF 4120, 39016 Magdeburg, Germany

* Correspondence: frank.werner@ovgu.de

† These authors contributed equally to this work.

Abstract: In this paper, we consider some problems that arise in connected and autonomous vehicle (CAV) systems. Their simplified variants can be formulated as scheduling problems. Therefore, scheduling solution algorithms can be used as a part of solution algorithms for real-world problems. For four variants of such problems, mathematical models and solution algorithms are presented. In particular, three polynomial algorithms and a branch and bound algorithm are developed. These CAV scheduling problems are considered in the literature for the first time. More complicated NP-hard scheduling problems related to CAVs can be considered in the future.

Keywords: scheduling; optimization; dynamic programming; connected and autonomous vehicle; precedence relations

MSC: 90 B 35; 90 C 27; 68 Q 25; 68 W 40

JEL Classification: D8; H51



Citation: Gafarov, E.R.; Werner, F. Connected and Autonomous Vehicle Scheduling Problems: Some Models and Algorithms. *Algorithms* **2024**, *17*, 421. <https://doi.org/10.3390/a17090421>

Academic Editor: Roberto Montemanni

Received: 23 August 2024

Revised: 15 September 2024

Accepted: 20 September 2024

Published: 21 September 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Connected and autonomous vehicle systems contain a set of vehicles that can automatically coordinate their routing, owing to vehicle-to-vehicle communications and a centralized scheduler (computer). The vehicles are controlled by the centralized scheduler residing in the network (e.g., a base station in the case of cellular systems) or a distributed scheduler, where the scheduler is autonomously selected by the vehicles. Such vehicle systems can coordinate the vehicles in order to speed up the traffic, minimize traffic jams, and they also take into account environment aspects by reducing emissions.

Vehicle routing and scheduling problems have been intensively investigated over the last decades. There exist a huge number of papers dealing with particular aspects or also covering relationships with related research fields, see, for instance, refs. [1–5] to name only a few papers. A complete special issue with 16 papers on different aspects of vehicle routing and scheduling has been edited by Repoussis and Gounaris [6]. For surveys in this field, the reader is referred to the papers [7–9]. During the last years, new challenges arose in connection with autonomous driving of vehicles.

Vehicle-to-vehicle (V2V) communications can be used as a potential solution to many problems arising in the area of traffic control. Several approaches have been developed including the modeling of a complete autonomous driving system as a multi-agent system, where the vehicles interact to ensure an autonomous functionality such that emergency braking and traffic jam are avoided as much as possible. Nowadays, vehicle systems are developed towards fully connected and fully autonomous systems. Vehicular communication technologies have been considered, for instance, in the papers [10,11].

In [12], the authors dealt with the optimization of the departure times, travel routes, and the longitudinal trajectories of connected and autonomous vehicles and also the control of the signal timings at the intersections to obtain a stable traffic flow in such a way that the vehicles do not need to stop before they enter an intersection. In addition, the vehicle queues before the intersections should not become too large. In this paper, all the data (i.e., the departure times, travel routes, and signal timings) are determined by a central controller, but the trajectories of the vehicles are fixed by distributed roadside processors, which constitute a hierarchical traffic management scheme together.

In [13], the authors considered the control of the required lane changes of a system of autonomous vehicles on a road segment with two lanes before the vehicles arrive at a given critical position. This paper presents an algorithm that realizes the lane change of an individual vehicle in the shortest possible time. Then, this algorithm is iteratively used to manage all the lane changes which are necessary on the road segment under consideration in such a way that traffic safety is maintained.

In the paper by [14], the problem of scheduling a CAV, which crosses an intersection, was considered with the goal to optimize the traffic flow at the intersection. In addition, a solution algorithm was presented.

In [15], the problem of scheduling the time phases of a traffic light was considered with the objective to improve the traffic flow by reducing the waiting time of the traveling vehicles at the indicated road intersections.

To the best of our knowledge, in this paper, simplified CAV problems are formulated as classical scheduling problems for the first time. Scheduling solution algorithms can be used as a part of solution algorithms for real-world CAV problems. In the future, more sophisticated NP-hard scheduling models can be considered.

The aim of the paper is to encourage the researchers to consider CAV problems as scheduling problems and to use classical methods of scheduling theory to investigate the complexity and to solve them.

Although the solution algorithms presented have a large running time and are somehow still useless in practice, we have shown that some CAV problems are polynomially solvable. Solution algorithms can be advanced in future research.

In this paper, we consider four scheduling problems that arise in connection with CAVs. The remainder of this paper is as follows. In each of the Sections 2–5, we consider one of these problems. The problem with a road of two lanes and one lane closure is considered in Section 2. Section 3 deals with the case of a turn to a main road. Section 4 considers the case of a road with three lanes and a lane closure on the middle lane. Section 5 deals with a crossroad having dividing lanes. For each of these cases, an appropriate scheduling problem is formulated and a solution algorithm is given. Finally, Section 6 gives a few concluding remarks.

2. A Road with Two Lanes and One Lane Closure

In this section, we consider a road with two lanes, where two sets of CAVs, N_1 and N_2 , are given. The CAVs from the set N_1 go on lane 1, and the CAVs from the set N_2 go on lane 2. Both lanes have the same direction. On lane 2, there is a lane closure and the CAVs from the set N_2 have to move to lane 1, see Figure 1.

We have to find a sequence of passing the lane closure by the CAVs from the sets N_1 and N_2 in order to minimize a given objective function, e.g., the total passing time.

We assume that

- a maximal feasible speed of the CAVs is given. The CAVs either go with the maximal feasible speed or brake in order to let another CAV change the lane.
- an acceleration is not taken into account.
- the time needed to change the lane is not taken into account, i.e., it is equal to zero.
- the CAVs have the same length.
- the safe distance between two CAVs is the same for all vehicles.

Since the problems investigated in this paper have not been considered in the literature so far, these assumptions are made for the first time to obtain simplified models. In real-world problems, these assumptions can be false, although solution algorithms for simplified models can be used to solve sub-problems or to obtain lower or upper bounds.

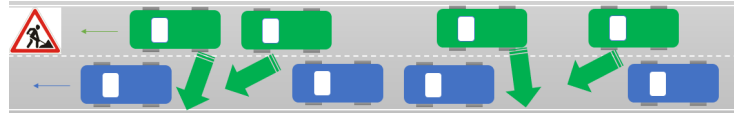


Figure 1. A road with two lanes and one lane closure.

The same problem arises, e.g., on railway sections and in automated warehouses of logistics companies with autonomous robot transporters. This simplified problem can be formulated as a single machine scheduling problem as follows.

Given a set $N = N_1 \cup N_2$ of n jobs that have to be processed on a single machine from time 0 on. For each job j , a processing time $p_j = p > 0$, a release date $r_j \geq 0$, a due date $d_j \geq 0$, and a weight $w_j > 0$ are given. The machine can process no more than one job at a time. The processing time p can be computed from the maximal feasible speed and the length of a CAV. The value r_j corresponds to the earliest time when the processing of the vehicle can start, resulting from the position of the CAV j on the road.

For example, let the speed of each car be 60 km per hour and the length of the closed road section be 100 m. Thus, a car needs 6 s to pass the road closure. Therefore, we assume $p = 6$ (in seconds). Let a car j be 0 m from the beginning of the road closure and a car i be 200 m behind. So, we assume $r_j = 0$ and $r_i = 12$ according to their speed and the distances. Moreover, let the starting time of any schedule computed be 05:00:00 a.m. which we consider as time 0. Let the due date for car i be 05:00:15, then we assume $d_i = 15$. It is obvious that the tardiness of car i is no less than $6 + 12 - 15 = 3$ seconds in any feasible schedule.

We call a feasible schedule active if one cannot reduce the objective function value by shifting a single job to an earlier time without violating the constraints. Without loss of generality, we consider only active schedules in this paper.

A schedule is uniquely determined by a permutation π of the CAVs of the set N . Let $S_j(\pi)$ be the starting time of job j in the schedule π . Then, $C_j(\pi) = S_j(\pi) + p$ is the completion time of job j in the schedule π . A precedence relation can be defined as follows. For the jobs from the set $N_1 = \{j_1, j_2, \dots, j_{n_1}\}$, we have $j_1 \rightarrow j_2 \rightarrow \dots \rightarrow j_{n_1}$, where $n_1 = |N_1|$ and $j \rightarrow i$ means that the processing of job j precedes the processing of job i . Thus, there is a chain of jobs on lane 1. Analogously, a chain of jobs is defined for the set $N_2 = \{i_1, i_2, \dots, i_{n_2}\}$.

For the single machine scheduling problem of minimizing total completion time, the goal is to find an optimal schedule π^* that minimizes

$$\sum C_j = \sum_{j \in N} C_j(\pi). \tag{1}$$

Here, the completion time of a job is equal to the time when the vehicle passes the closed lane segment. We denote this problem by $1|2 \text{ chains}, p_j = p, r_j| \sum C_j$ according to the traditional three-field notation $\alpha|\beta|\gamma$ for scheduling problems proposed by [16], where α describes the machine environment, β gives the job characteristics and further constraints, and γ describes the optimization criterion.

Let

$$T_j(\pi) = \max\{0, C_j(\pi) - d_j\}$$

be the tardiness of job j in the schedule π . If $C_j(\pi) > d_j$, then job j is tardy and we have $U_j = 1$; otherwise, $U_j = 0$.

Subsequently, we consider also the following objective functions:

$$\begin{aligned} \sum w_j C_j &= \sum w_j C_j(\pi) \text{ — total weighted completion time,} \\ \sum T_j &= \sum T_j(\pi) \text{ — total tardiness,} \\ \sum w_j T_j &= \sum w_j T_j(\pi) \text{ — total weighted tardiness;} \\ \sum w_j U_j &= \sum w_j U_j(\pi) \text{ — weighted number of tardy jobs,} \\ C_{max} &= \max\{C_j(\pi)\} \text{ — makespan} \end{aligned}$$

It is known that the problems $1|chains, p_j = p, r_j|\sum w_j C_j$ and $1|chains, p_j = p, r_j|\sum w_j T_j$ with an arbitrary number of chains are NP-hard, see [17]. This has been proven by a reduction from the 3-partition problem.

In [18], polynomial time dynamic programming algorithms have been presented to solve the problems $1|p_j = p, r_j|\sum T_j$ and $1|p_j = p, r_j|\sum w_j U_j$.

In an optimal schedule for the problem $1|2\ chains, p_j = p, r_j|\sum C_j$, the jobs are processed in non-decreasing order of the values r_j . This can be easily proven by contradiction. Assume that we have an optimal schedule $\pi = (\dots, j_2, j_1, \dots)$, where $r_{j_1} < r_{j_2}$. Then, for the schedule $\pi' = (\dots, j_1, j_2, \dots)$, we have $\sum C_j(\pi) \geq \sum C_j(\pi')$. For an illustration of the concepts introduced above, we consider the following small Example 1.

Example 1. Let $N_1 = \{1, 2\}$, $N_2 = \{3, 4\}$. Moreover, the values $p = 2$, $r_1 = 0$, $r_2 = 3$, $r_3 = 1$, $r_4 = 4$, and $d_1 = d_2 = 10$, $d_3 = 3$, $d_4 = 6$ are given. For the chosen job sequence $\pi = (1, 3, 2, 4)$, we obtain

$$S_1(\pi) = 0, S_3(\pi) = 2, S_2(\pi) = 4, S_4(\pi) = 6$$

and

$$C_1(\pi) = 2, C_3(\pi) = 4, C_2(\pi) = 6, C_4(\pi) = 8.$$

Thus, we obtain

$$\sum_{j=1}^4 C_j(\pi) = 20 \quad \text{and} \quad \sum_{j=1}^4 T_j(\pi) = 1 + 2 = 3.$$

For the job sequence $\pi' = (3, 4, 1, 2)$, we obtain

$$\sum_{j=1}^4 T_j(\pi') = 0.$$

We note that there exists a set Θ of possible completion times of all jobs with $|\Theta| \leq n^2$ since:

- without loss of generality, we consider only active schedules, where no job can be processed earlier without loss of feasibility;
- there are no more than n different values r_j ;
- all processing times are equal to p and thus, for any job $j \in N$, its completion time is equal to $r_i + lp$, $i \in N, l \leq n$.

The problems $1|2\ chains, p_j = p, r_j|f$, $f \in \{\sum w_j C_j, \sum w_j T_j, \sum w_j U_j\}$ can be solved by a dynamic program (DP). In the DP, we consider the jobs $i_1, i_2, \dots, i_{n_2} \in N_2$ one by one, where $i_1 \rightarrow i_2 \rightarrow \dots \rightarrow i_{n_2}$. Thus, at each stage k of the dynamic program, we consider a single job i_k , $k = 1, 2, \dots, n_2$. Moreover, at each stage $k > 1$ we consider all states $(f^{k-1}, C_{max}^{k-1}, pos^{k-1})$ and the corresponding best partial solutions (sequences of jobs) stored at the previous stage. The meaning of the above triplet is as follows. Here, f^{k-1} is the value of the considered objective function for the partial solution. $C_{max}^{k-1} = C_{i_{k-1}} \in \Theta$ denotes the completion time of job i_{k-1} in the corresponding partial solution. Finally, $pos^{k-1} \in \{0, 1, 2, \dots, n_1\}$ describes the position of a job, and this means that job i_{k-1} is processed between the jobs $j_{pos} \in N_1$ and $j_{pos+1} \in N_1$, $0 < pos < n_1$. For each job i_k and a state $(f^{k-1}, C_{max}^{k-1}, pos^{k-1})$, we compute

new states (f', C'_{max}, pos') , where $pos' \geq pos$, and C'_{max} is the completion time of job i_k in a new partial solution, where job i_k is scheduled after job $j_{pos'} \in N_1$. If, at any stage, there are two states (f', C'_{max}, pos') and (f'', C''_{max}, pos') with $f' \leq f''$ and $C'_{max} \leq C''_{max}$, we only keep the state (f', C'_{max}, pos') . After the last stage, we have to select the best found complete solution among all states generated.

Let us explain a state (f^1, C^1_{max}, pos^1) by means of Example 1. In Algorithm 1, at stage $k = 1$, we have $i_k = 3$. Thus, we consider the partial schedule $\pi = (1, 3)$ and the corresponding state (f^1, C^1_{max}, pos^1) . For the objective function $\sum C_j$, we have

$$(f^1, C^1_{max}, pos^1) = (C_1(\pi) + C_3(\pi), C_3(\pi), 1) = (6, 4, 1).$$

For the objective function $\sum T_j$, we have

$$(f^1, C^1_{max}, pos^1) = (T_1(\pi) + T_3(\pi), C_3(\pi), 1) = (1, 4, 1).$$

Algorithm 1: A pseudo-code of Algorithm 1 is presented below.

1. $StatesSet = \{(0, 0, 0)\}$;
 2. FOR EACH $i_k \in N_2$ DO
 - 2.1 $NewStatesSet = \{\}$;
 - 2.2 FOR EACH $(f^{k-1}, C^{k-1}_{max}, pos^{k-1}) \in StatesSet$ DO
 - 2.2.1 Let $PositionsList = \{pos^{k-1}, pos^{k-1} + 1, \dots, n_1\}$;
 - 2.2.2 FOR EACH $pos' \in PositionsList$ DO
 - 2.2.2.1 Calculate f' for the resulting partial solution, if job i_k is processed after $j_{pos'}$, according to the partial solution corresponding to state $(f^{k-1}, C^{k-1}_{max}, pos^{k-1})$;
 - 2.2.2.2 Add (f', C'_{max}, pos') to $NewStatesSet$. If in $NewStatesSet$, there is a state (f'', C''_{max}, pos') with $f' \leq f''$ and $C'_{max} \leq C''_{max}$, then exclude the state (f'', C''_{max}, pos') from $NewStatesSet$;
 - 2.2.2.3 If i_k is the last job in the set N_2 , then schedule all unscheduled jobs from the set N_1 at the earliest possible time.
 - 2.3 $StatesSet := NewStatesSet$;
 3. Select the best found complete solution among all states generated.
-

Theorem 1. *The problems 1|2 chains, $p_j = p, r_j|f$, $f \in \{\sum w_j C_j, \sum w_j T_j, \sum w_j U_j\}$ can be solved in $O(n^5)$ time by a dynamic program.*

Proof. Dynamic programming is a mathematical optimization method, where a complicated problem is split into simpler sub-problems in a recursive manner.

In Algorithm 1, each state (f, C_{max}, pos) (here we skip the upper index for simplicity of the notations), calculated at stage $k - 1$, divides the problem into two sub-problems. In the first sub-problem, all jobs $j \in \{j_1, j_2, \dots, j_{pos}\}$ and all jobs $i \in \{i_1, \dots, i_{k-2}\}$ are considered. In the second sub-problem, all jobs $j \in \{j_{pos+1}, \dots, j_{n_1}\}$ and all jobs $i \in \{i_k, \dots, i_{n_2}\}$ are considered. Let π' be an optimal solution for the first sub-problem and π'' be an optimal solution for the second one. Then $\pi = (\pi', i_{k-1}, \pi'')$ is an optimal solution corresponding to state (f, C_{max}, pos) .

The proof of optimality of Algorithm 1 can be performed by induction. Let, for an instance of a problem, $\pi^* = (\pi_1, i_k, \pi_2)$ be an optimal solution and f^* be the optimal value

of the considered objective function. Moreover, let $pos_k \in \{0, 1, \dots, n_1, n_1 + 1\}$, $k = 1, \dots, n$, be the position of job $i_k \in N_2$ in the job sequence. This means that job j_{pos_k} is processed before job i_k and job j_{pos_k+1} is processed after it, where j_0 means that job i_k is processed before job j_1 and j_{n_1+1} means that job i_k is processed after job j_{n_1} . Let f' be the objective function value for the job sub-sequence (π_1, i_k) .

Next, we prove the following Property (*).

Property (*): At each stage k of Algorithm 1, for each job i_k the state (f_k, C_{i_k}, pos_k) and the corresponding partial job sequence (π^k, i_k) will be considered, where $C_{i_k}(\pi^*) = C_{i_k}$ and $f' = f_k$.

For stage 1, Property (*) holds since we consider and save in the set of states all $n_1 + 1$ possible positions for job i_1 . Let Property (*) hold for stage $k - 1$, i.e., the state $(f_{k-1}, C_{i_{k-1}}, pos_{k-1})$ is contained in the set of states.

Then, according to step [2.2.2] of Algorithm 1, the state (f_k, C_{i_k}, pos_k) will be considered and stored in the set of states.

So, according to step [3.] of Algorithm 1, the job sequence π will be constructed with the objective function value $f = f^*$ and the problems $1|2 \text{ chains}, p_j = p, r_j|f, f \in \{\sum w_j C_j, \sum w_j T_j, \sum w_j U_j\}$ can be solved by a dynamic program.

There are $O(n)$ stages and $O(n^3)$ states (f, C_{max}, pos) at each stage, since $C_{max} \in \Theta$ with $|\Theta| \leq n^2$, and $pos \in \{0, 1, \dots, n_1\}$. At the next stage, for each state generated at the previous stage in step [2.2.2], we need to consider $O(n)$ new states.

To perform all steps [2.2.2.1] in [2.2.2], we need $O(n)$ operations. In step [2.2.2.1], to construct a partial solution, we need to schedule the jobs $j_{pos^{k-1}+1}, j_{pos^{k-1}+2}, \dots, j_{pos^k}$ into the partial solution, i.e., to calculate their starting times according to C_{max}^{k-1} , the release dates, and the processing time. We need $O(1)$ operations to calculate the starting time of each job, and we calculate the starting time for a job only once in step [2.2.2], e.g., the starting time of job $j_{pos^{k-1}+1}$ is calculated only once in [2.2.2]. So, to perform all steps [2.2.2.1] in the cycle [2.2.2], we need $O(n)$ operations, and to perform all steps [2.2.2.1] in Algorithm 1, we need $O(n^5)$ operations.

To perform step [2.2.2.2] in $O(1)$ time, we additionally keep *PositionsList* in a 2-dimensional array with $O(n^2)$ rows corresponding to all possible values C_{max} and $O(n)$ columns corresponding to all possible values pos . We initiate the array in step [2.1]. So, to check a dominated value, we need $O(1)$ time. Thus, we need $O(n^5)$ operations to perform all steps [2.2.2.1] in Algorithm DP.

To perform all steps [2.2.2.3] in Algorithm 1, we need $O(n^5)$ time, since it is performed only for the last job in the set N_2 .

So, the running time of Algorithm 1 is $O(n^5)$.

The theorem has been proven. \square

We conjecture that there are other solution algorithms with a running time less than $O(n^5)$. Here, our motivation is only to show that the problems are polynomially solvable. For real-world problems, there are more parameters and constraints that have to be taken into consideration, and they can be possibly solved by other methods than dynamic programming.

3. Turn to a Main Road

There is a set N_1 of CAVs going along a main road and a set N_2 of CAVs turning into the main road from a side road (see Figure 2). In contrast to the problems $1|2 \text{ chains}, p_j = p|f, f \in \{C_{max}, \sum w_j C_j, \sum w_j T_j, \sum w_j U_j\}$, we have now $p_j = p^1, j \in N_1$ and $p_j = p^2, j \in N_2$. We denote these problems by $1|2 \text{ chains}, p_j \in \{p^1, p^2\}, r_j|f, f \in \{C_{max}, \sum w_j C_j, \sum w_j T_j, \sum w_j U_j\}$.

These problems can be solved by the same Algorithm 1. In Algorithm 1, we describe the states in the same way: (f, C_{max}, pos) . For any job $j \in N$ in an active schedule for these problems, its completion time is equal to $r_i + lp^1 + vp^2, i \in N, l \leq n, v \leq n$. Thus, we have $|\Theta| \leq n^3$, and the running time of Algorithm 1 is $O(n^6)$.

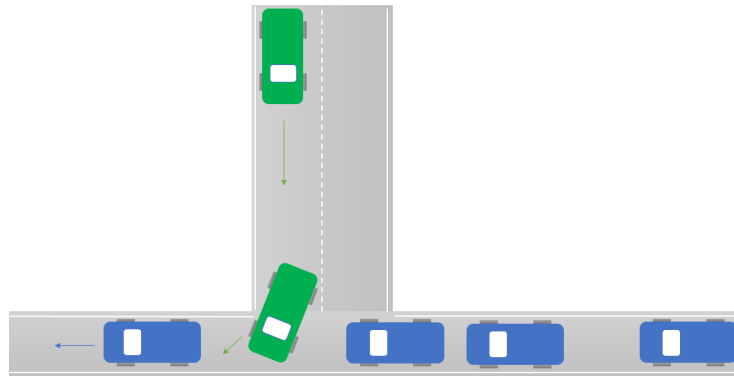


Figure 2. Turn to a main road.

4. A Road with Three Lanes and a Road Closure on the Middle Lane

In addition to the problems $1|2\ chains, p_j = p, r_j|f, f \in \{C_{max}, \sum w_j C_j, \sum w_j T_j, \sum w_j U_j\}$, there are an additional lane 3 and a subset N_3 of jobs (see Figure 3). The jobs of the set N_1 should be processed on machine M_1 , and the jobs of the set N_3 should be processed on machine M_3 . The jobs of the set N_2 can be processed on any of these two machines. Precedence relations among the jobs of the set N_3 can be defined as a chain of jobs.

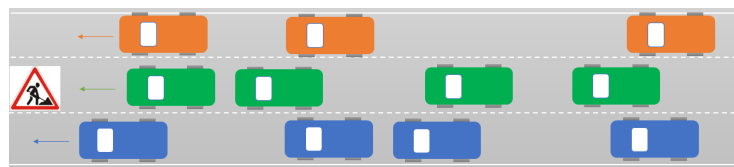


Figure 3. A road with three lanes and a closure on the middle lane.

We denote these problems by $P2|dedicated, 3\ chains, p_j = p, r_j|f, f \in \{C_{max}, \sum w_j C_j, \sum w_j T_j, \sum w_j U_j\}$. These problems can be solved by a modified Algorithm 1, where we consider the positions pos between the jobs of the set N_1 and between the jobs of the set N_3 .

We illustrate the dynamic programming algorithm for this problem by the following Example 2.

Example 2. Let $N_1 = \{1,2\}$, $N_2 = \{3,4\}$, $N_3 = \{5,6\}$. Moreover, the values $p = 2$, $r_1 = 0$, $r_2 = 3$, $r_3 = 1$, $r_4 = 4$, $r_5 = 1$, $r_6 = 4$, and $d_1 = 2$, $d_2 = 5$, $d_3 = 3$, $d_4 = 6$, $d_5 = 3$, $d_6 = 8$ are given. We consider the minimization of total tardiness, i.e., $f = \sum T_j$. Denote by M_1 the machine, where the jobs from the set N_1 have to be processed and by M_3 the machine, where the jobs from the set N_3 have to be processed. The initial positions of the jobs and an optimal schedule are presented in Figure 4.

In Algorithm 1, at each stage we consider all states

$$(pos^1, C_{max}^1, pos^3, C_{max}^3, f)$$

stored at the previous stage. Here, pos^1 is the first possible position for the current job on machine M_1 , C_{max}^1 denotes the completion time of the last job scheduled on machine M_1 , pos^3 gives the first possible position for the current job on machine M_3 , and C_{max}^3 denotes the completion time of the last job scheduled on machine M_3 .

In the following Table 1, all states computed in the two stages for the jobs $j = 3$ and $j = 4$ are presented. In the first column S1, the index numbers of the states are given. In the second column S2, we present the original state from which the current state is computed. P^1 represents pos^1 , C^1 denotes C_{max}^1 , P^3 represents pos^3 , and C^3 denotes C_{max}^3 . π^1 gives the corresponding job sequence on machine M_1 and π^3 describes the corresponding job sequence on machine M_3 . In the columns C_j , $j = 1, \dots, 6$, the corresponding completion times are given. In the columns T_j , $j = 1, \dots, 6$, the corresponding tardiness values are given. In the last column, the resulting total tardiness values $f = \sum T_j$ are presented.

Table 1. Calculations of Algorithm 1 for Example 2.

								r_1	r_2	r_3	r_4	r_5	r_6	d_1	d_2	d_3	d_4	d_5	d_6		
								0	3	1	4	1	5	2	5	3	6	3	8		
S1	S2	P^1	C^1	P^3	C^3	π_1	π_3	C_1	C_2	C_3	C_4	C_5	C_6	T_1	T_2	T_3	T_4	T_5	T_6	f	
Stage $j = 3$																					
1		0	3			3				3				0	0	0	0	0	0	0	
2		1	4			1, 3		2		4				0	0	1	0	0	0	1	
3		2	7			1, 2, 3		2	5	7				0	0	4	0	0	0	4	
4				0	3		3			3				0	0	0	0	0	0	0	
5				5	7		5, 3			5		3		0	0	2	0	0	0	2	
6				6	9		5, 6, 3			9		3	7	0	0	6	0	0	0	6	
Stage $j = 4$																					
7	1	0	6			3, 4, 1, 2	5, 6	8	10	3	6	3	7	6	5	0	0	0	0	11	
8	1	1	7			3, 1, 4, 2	5, 6	5	9	3	7	3	7	3	4	0	1	0	0	8	
9	1	2	9			3, 1, 2, 4	5, 6	5	7	3	9	3	7	3	2	0	3	0	0	8	
10	1	0		0	5	3, 1, 2	4, 5, 6	5	7	3	6	8	10	3	2	0	0	5	2	12	
11	1	0		5	6	3, 1, 2	5, 4, 6	5	7	3	6	3	8	3	2	0	0	0	0	5	
12	1	0		6	9	3, 1, 2	5, 6, 4	5	7	3	9	3	7	3	2	0	3	0	0	8	
13	2	1				1, 3, 4, 2	5, 6	2	8	4	6	3	7	0	3	1	0	0	0	4	
14	2	2				1, 3, 2, 4	5, 6	2	6	4	8	3	7	0	1	1	2	0	0	4	
15	2	1		0	6	1, 3, 2	4, 5, 6	2	6	4	6	8	10	0	1	1	0	5	2	9	
16	2	1		5	6	1, 3, 2	5, 4, 6	2	6	4	6	3	8	0	1	1	0	0	0	2	
17	2	1		6	9	1, 3, 2	5, 6, 4	2	6	4	9	3	7	0	1	1	3	0	0	5	
18	3	2		2	9	1, 2, 3, 4	5, 6	2	5	7	9	3	7	0	0	4	3	0	0	7	
19	3	2		0	5	1, 2, 3	4, 5, 6	2	5	7	9	11	13	0	0	4	3	8	5	20	
20	3	2		5	6	1, 2, 3	5, 4, 6	2	5	7	9	3	11	0	0	4	3	0	3	10	
21	3	2		6	9	1, 2, 3	5, 6, 4	2	5	7	9	3	7	0	0	4	3	0	0	7	
22	4	0	6			4, 1, 2	3, 5, 6	8	10	3	6	5	7	6	5	0	0	2	0	13	
23	4	1	6			1, 4, 2	3, 5, 6	2	8	3	6	5	7	0	3	0	0	2	0	5	
24	4	2	7			1, 2, 4	3, 5, 6	2	5	3	7	5	7	0	0	0	1	2	0	3	
25	4			0	6	1, 2	3, 4, 5, 6	2	5	3	6	8	10	0	0	0	0	5	2	7	
26	4			5	7	1, 2	3, 5, 4, 6	2	5	3	7	5	9	0	0	0	1	2	1	4	
27	4			6	9	1, 2	3, 5, 6, 4	2	5	3	9	5	7	0	0	0	3	2	0	5	
28	5	0	6	5		4, 1, 2	5, 3, 6	9	11	5	7	3	7	7	6	2	1	0	0	16	
29	5	1	7	5		1, 4, 2	5, 3, 6	2	9	5	7	3	7	0	4	2	1	0	0	7	
30	5	2	7	5		1, 2, 4	5, 3, 6	2	5	5	7	3	7	0	0	2	1	0	0	3	
31	5			5		1, 2	5, 3, 4, 6	2	5	5	7	3	9	0	0	2	1	0	1	4	
32	5			6		1, 2	5, 3, 6, 4	2	5	5	9	3	7	0	0	2	3	0	0	5	
33	6	0	11	6		4, 1, 2	5, 6, 3	13	15	9	11	3	7	11	10	6	5	0	0	32	
34	6	1	6	6		1, 4, 2	5, 6, 3	2	13	9	11	3	7	0	8	6	5	0	0	19	
35	6	2	11	6		1, 2, 4	5, 6, 3	2	5	9	11	3	7	0	0	6	5	0	0	11	
36	6			6	11	1, 2	5, 6, 3, 4	2	5	9	11	3	7	0	0	6	5	0	0	11	

An optimal solution is found in state 16. Let us consider an extended instance with an additional job 7 in the set N_2 , where $3 \rightarrow 4 \rightarrow 7$. Then, in Algorithm DP, we will have 3 stages. At the state 15, we will have $(pos^1, C_{max}^1, pos^3, C_{max}^3, f) = (1, 4, 0, 6, 2)$, $\pi^1 = (1, 3)$ and $\pi^2 = (4)$. At the state 16, we will have $(pos^1, C_{max}^1, pos^3, C_{max}^3, f) = (1, 4, 1, 6, 2)$, $\pi^1 = (1, 3)$ and $\pi^2 = (5, 4)$. So, we only keep the state 16, since pos^3 is larger for state 16 and the other parameters of the states are the same.

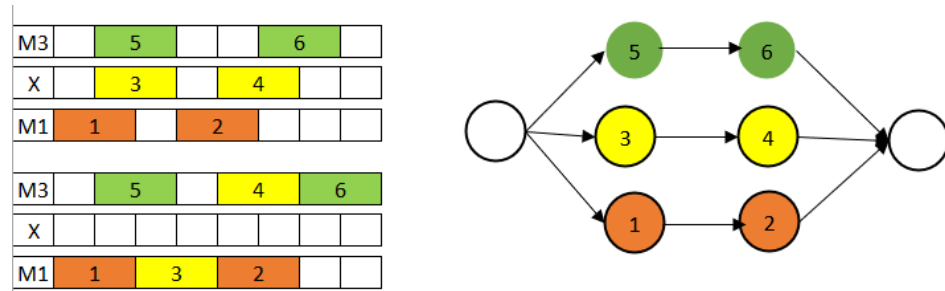


Figure 4. The initial positions of the jobs and an optimal schedule for Example 2.

5. A Crossroad with Dividing Lines

In this section, we consider a crossroad with dividing lines and four sets, N_1, N_2, N_3, N_4 , of CAVs. They share four sectors of a crossroad denoted by M_1, M_2, M_3, M_4 (see Figure 5). We have to find an optimal sequence of passing these sectors.

We can formulate the following job shop scheduling problem with four machines. There are four sets, N_1, N_2, N_3, N_4 , of jobs and four machines corresponding to the sectors M_1, M_2, M_3, M_4 . Each job j consists of two operations. For each job $j \in N_1$, its first operation has to be processed on machine M_1 and its second one has to be processed on machine M_2 . For each job $j \in N_2$, its first operation has to be processed on machine M_2 and its second one has to be processed on machine M_4 . For each job $j \in N_3$, its first operation has to be processed on machine M_3 and its second one has to be processed on machine M_1 . For each job $j \in N_4$, its first operation has to be processed on machine M_4 and its second one has to be processed on machine M_3 . The processing times of the operations are equal to p . Precedence relations can be given as chains of jobs.

If the lengths of the dividing lines are equal to 0, then the second operation of a job j should be processed immediately after the first one. Otherwise, for each of the sets N_1, N_2, N_3, N_4 , there are four buffers of limited capacities, namely b_1, b_2, b_3, b_4 jobs for the corresponding set of jobs. At any moment, for the set N_1 , there can be up to b_1 jobs for which the first operation is completed and the second one is not yet started. We denote these problems by $J4|4 chains, p_j = p, r_j|f, f \in \{C_{max}, \sum w_j C_j, \sum w_j T_j, \sum w_j U_j\}$.

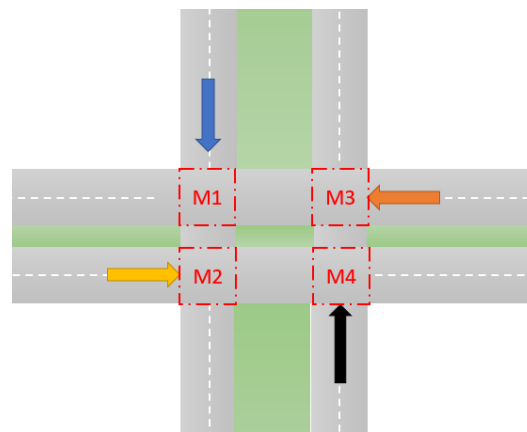


Figure 5. A crossroad with dividing lines.

The problems $J4|4 chains, p_j = p, r_j|f, f \in \{C_{max}, \sum w_j C_j, \sum w_j T_j, \sum w_j U_j\}$ can be solved by a branch-and-bound (B&B) algorithm. The search (rooted) tree is constructed by the following branching rule. For any node of the tree, we consider the following 8 possible branches:

- Schedule the first unscheduled possible operation for a job $j \in N_1$ on machine M_1 at the earliest possible starting time. If there is no such an operation, skip this branch.
- Schedule the first unscheduled possible operation for a job $j \in N_3$ on machine M_1 at the earliest possible starting time.

- Schedule the first unscheduled possible operation for a job $j \in N_1$ on machine M_2 at the earliest possible starting time.
- Schedule the first unscheduled possible operation for a job $j \in N_2$ on machine M_2 at the earliest possible starting time.
- Schedule the first unscheduled possible operation for a job $j \in N_3$ on machine M_3 at the earliest possible starting time.
- Schedule the first unscheduled possible operation for a job $j \in N_4$ on machine M_3 at the earliest possible starting time.
- Schedule the first unscheduled possible operation for a job $j \in N_2$ on machine M_4 at the earliest possible starting time.
- Schedule the first unscheduled possible operation for a job $j \in N_4$ on machine M_4 at the earliest possible starting time.

Thus, there are up to $2^3 = 8$ branches for each node to be considered. Since there are $2n$ operations, where $n = |N_1 \cup N_2 \cup N_3 \cup N_4|$, there are no more than $2n$ levels in the search tree. Thus, we have no more than $(2^3)^{2n} = 2^{6n}$ nodes to be considered. If some of the values b_1, b_2, b_3, b_4 are equal to 0, we have fewer nodes. As an example, if each of them is equal to 0, then we have only 2^{3n} nodes.

Moreover, we can use the following trivial upper and lower bounds for the problem $J4|4 \text{ chains}, p_j = p, r_j|C_{max}$.

Upper bound. To construct a feasible solution, we use a list scheduling algorithm. In this algorithm, we consider the unscheduled operations one-by-one according to a non-decreasing order of the release dates of the corresponding jobs. We schedule the next unscheduled operation at the earliest possible starting time according to the current partial schedule. To order the set of jobs, we need $O(n \log n)$ operations. In addition, we need $O(n)$ operations to construct a feasible solution.

Lower bound. Consider a set of unscheduled operations N' . For each of them, we calculate the earliest possible starting time according to the current partial schedule without taking into account the other unscheduled operations. In such a way, we obtain a schedule π that can be infeasible. Let $C_{M_1}(\pi)$ be the makespan (i.e., the maximal completion time of an operation assigned to the machine) for machine M_1 , $IT_{M_1}(\pi)$ be the idle time on machine M_1 between the operations of the set N' , and $OT_{M_1}(\pi)$ be the total overlap time, where more than one operation is processed at the same time. Moreover, let

$$C'_{M_1}(\pi) = C_{M_1}(\pi) + \max\{0, OT_{M_1}(\pi) - IT_{M_1}(\pi)\}.$$

Similarly, we can define $C'_{M_j}(\pi)$ for $j = 2, 3, 4$. Then,

$$LB1 = \max\{C'_{M_1}(\pi), C'_{M_2}(\pi), C'_{M_3}(\pi), C'_{M_4}(\pi)\}$$

is a lower bound. It is easy to check that we need $O(n)$ operations to calculate this bound.

If we use an upper bound and lower bound, then the B&B algorithm requires $O(n2^{6n})$ operations.

6. Concluding Remarks

In this note, four models of scheduling problems for CAVs have been given. Three of them can be solved by a dynamic programming algorithm in polynomial time. For the fourth problem, a B&B algorithm has been presented. The investigations in this paper are only a first step in the development of scheduling algorithms for CAV problems. The presented algorithms can handle only very special problems. However, they can be potentially used as a part of more complex algorithms for more general CAV problems (e.g., considering more lanes, different speeds of the cars, or a closure of more lanes). For real-world problems related to the scheduling of CAVs, fast metaheuristics and online algorithms can be developed in the future. Finally we give two specific research questions that are worth considering:

- Are the problems $J4|4 \text{ chains}, p_j = p, r_j|f, f \in \{C_{max}, \sum w_j C_j, \sum w_j T_j, \sum w_j U_j\}$ NP-hard or can they be solved in polynomial time?
- Are there problems with CAVs having equal processing times and a fixed number of chains of jobs, which is an NP-hard problem?

Author Contributions: Conceptualization, E.R.G. and F.W., investigation, E.R.G. and F.W., visualization, E.R.G.; writing—original draft preparation, F.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The data are available from the authors on request.

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

References

1. Ullrich, C. Integrated Machine Scheduling and Vehicle Routing with Time Windows. *Eur. J. Oper. Res.* **2012**, *1*, 152–165.
2. Afifi, S.; Dang, D.-C.; Moukrim, A. Heuristic Solutions for the Vehicle Routing Problem with Time Windows and Synchronized Visits. *Optim. Lett.* **2016**, *10*, 511–525. [[CrossRef](#)]
3. Norouzi, N.; Sadegh-Amalnick, M.; Tavakkoli-Moghaddam, R. Modified Particle Swarm Optimization in a Time-Dependent Vehicle Routing Problem: Minimizing Fuel Consumption. *Optim. Lett.* **2017**, *11*, 121–134. [[CrossRef](#)]
4. Nasiri, M.M.; Rahbari, A.; Werner, F.; Karimi, R. Incorporating Supplier Selection and Order Allocation into the Vehicle Routing and Multi-Cross-Dock Scheduling Problem. *Intern. J. Prod. Res.* **2018**, *56*, 6527–6552. [[CrossRef](#)]
5. Rahbari, A.; Nasiri, M.M.; Werner, F.; Musavi, M.; Jolai, F. The Vehicle Routing and Scheduling Problem with Cross-Docking for Perishable Products under Uncertainty: Two Robust Bi-objective Models. *Appl. Math. Modell.* **2018**, *70*, 605–625. [[CrossRef](#)]
6. Repoussis, P.P.; Gounaris, C.E. Special Issue on Vehicle Routing and Scheduling: Recent Trends and Advances. *Optim. Lett.* **2013**, *7*, 1399–1403. [[CrossRef](#)]
7. Bunte, S.; Kliewer, N. An Overview on Vehicle Scheduling Models. *Public Transport.* **2009**, *1*, 299–317. [[CrossRef](#)]
8. Han, M.; Wang, Y. A Survey for Vehicle Routing Problems and its Derivatives. *IOP Conf. Ser. Mater. Sci. Eng.* **2018**, *452*, 042024. [[CrossRef](#)]
9. Mor, A.; Speranza, M.G.: Vehicle Routing Problems over Time: A Survey. *4OR* **2020**, *18*, 129–149. [[CrossRef](#)]
10. Bazzal, M.; Krawczyk, L.; Govindarajan, R.P.; Wolff, C. Timing Analysis of Car-to-Car Communication Systems Using Real-Time Calculus: A Case Study. In Proceedings of the 2020 IEEE 5th International Symposium on Smart and Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), Dortmund, Germany, 17–18 September 2020; pp. 1–8.
11. Şahin, T.; Khalili, R.; Boban, M.; Wolisz, A. Reinforcement Learning Scheduler for Vehicle-to-Vehicle Communications Outside Coverage. In Proceedings of the 2018 IEEE Vehicular Networking Conference (VNC), Taipei, Taiwan, 5–7 December 2018; pp. 1–8.
12. Qian, G.; Guo, M.; Zhang, L.; Wang, Y.; Hu, S.; Wang, D. Traffic scheduling and control in fully connected and automated networks. *Transp. Res. Part C Emerg. Technol.* **2021**, *126*, 103011. [[CrossRef](#)]
13. Atagoziev, M.; Schmidt, E.G.; Schmidt, K.W. Lane change scheduling for connected and autonomous vehicles. *Transp. Res. Part C Emerg. Technol.* **2023**, *147*, 103985. [[CrossRef](#)]
14. Ma, M. Optimal Scheduling of Connected and Autonomous Vehicles at a Reservation-Based Intersection. Ph.D. Thesis, University of Louisville, Louisville, KY, USA, 2022.
15. Bani Younes, M.; Boukerche, A. An Intelligent Traffic Light scheduling algorithm through VANETs. In Proceedings of the 39th Annual IEEE Conference on Local Computer Networks Workshops, Edmonton, AB, Canada, 8–11 September 2014; pp. 637–642.
16. Graham, R.L.; Lawler, E.L.; Lenstra, J.K.; Rinnooy Kan, A.H.G. Optimization and Approximation in Deterministic Machine Scheduling: A Survey. *Ann. Discr. Math.* **1979**, *5*, 287–326.
17. Lenstra, J.K.; Rinnooy Kan, A.H.G. Complexity results for scheduling chains on a single machine. *Eur. J. Oper. Res.* **1980**, *4*, 270–275. [[CrossRef](#)]
18. Baptiste, P. Scheduling equal-length jobs on identical parallel machines. *Discret. Appl. Math.* **2000**, *103*, 21–32. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.