A label-setting algorithm for the truck driver scheduling problem in accordance with European Community social legislation

T. Garaix, P. Lacomme, I. Peña-Arenas

A Label-Setting Algorithm for the Truck Driver Scheduling Problem in accordance with European Community Social Legislation

Garaix, T.
Mines Saint-Etienne, Univ Clermont Auvergne, CNRS, UMR 6158 LIMOS, Centre CIS, Departement I4S, F - 42023 Saint-Etienne, France
e-mail: garaix@emse.fr

Lacomme, P., and Peña-Arenas. I.
Université Clermont Auvergne, LIMOS UMR 6158, 63178 Aubière, France
e-mail: placomme@isima.fr, exp684@yahoo.com (corresponding autor)

Abstract
In this paper, a label-setting algorithm is proposed in order to efficiently solve the weekly Truck Driver Scheduling Problem (TDSP), which can be seen as a single-machine problem that minimizes the completion time of a fixed sequence of jobs with release and due times and break periods. A focus is made on the night working rule that has only partially been considered in the literature. In the wide literature dedicated to Vehicle Routing Problems, some published heuristics and exact algorithms include constraints related to break/rest periods. But the night working rule defined by the European Community Social Legislation is only considered in some simplified and sub-optimal versions. This constraint is of interest, since it has a direct impact on the practical applicability of routes. Moreover, from a scheduling point of view, the night working rule enforces minimum and maximum time lag constraints, which makes challenging the development of an efficient solution algorithm. The method is benchmarked with a mixed integer linear programming model. Computational experiments show the efficiency of the proposed methods and the effect of the night working constraint on the feasibility of the schedules.

Keywords: driver scheduling, integer programming, label setting algorithms, road freight transport, EC Regulation 561/2006, Directive 2002/15/EC

1. Introduction

The European Union implemented European Community social legislation on drivers' working hours (hereafter, EC social legislation) to enhance road safety, address driver fatigue, improve working conditions, and regulate competition among overland transport modes. It consists of two legislative acts: European Community (EC) Regulation No 561/2006 on driving hours and Directive 2002/15/EC on working hours for road transportation personnel. These regulations apply to drivers of goods vehicles weighing over 3.5 tons (including trailers) and drivers of passenger vehicles with more than nine seats. Freight cargo companies are accountable for any violations committed by their employees. Consequently, this legislation has had a significant impact on transport companies' operations.

This paper focuses on the Truck Drivers Scheduling Problem (TDSP) and addresses the scheduling of breaks and rests while considering the EC social legislation. Previous studies by various authors, such as Goel (2009), Drexl & Prescott-Gagnon (2010), and Sartori et al. (2022), have made contributions to solving the problem. TDSP algorithms play a crucial role as subroutines in both exact and heuristic procedures for solving vehicle routing problems that involve driver's rules. One of the primary applications of a TDSP algorithm is its integration as a subroutine within a surrounding vehicle routing problem (meta)heuristic, necessitating the use of

fast and efficient heuristic algorithms (Drexl & Prescott-Gagnon, 2010; Goel, A. 2012a; Goel, et al. 2012b). A more advanced and efficient TDSP algorithm enables a more thorough search of the solution space, consequently enhancing the overall performance of the optimization process for vehicle routing problems with driver's rules (Goel, A. 2010). This is especially critical when the TDSP is solved within iterative methods for tackling vehicle routing problems with driver's rules, as computational efficiency becomes a key factor in the integrated solution schema.

However, none of the previous works have fully incorporated all the rules of the EC social legislation. Regarding the TDSP within the Vehicle Routing problem (VRTDSP), works by Kok et al. (2010), Prescott-Gagnon et al. (2010), and Goel & Vidal (2013) include most of the regulation's constraints, except for the night working constraint. Goel (2018) and Tilk & Goel (2020) propose labeling methods that find feasible solutions for the TDSP, assuming that drivers do not perform night work. Peña-Arenas et al. (2021) present a Mixed Integer Linear Programming (MILP) formulation that incorporates the complete set of rules from the EC social legislation, including the night working constraint, but call for more efficient solution methods.

Regarding solution methods, label-setting algorithms have demonstrated their efficacy in solving the TDSP (Goel, A., 2009; Drexl, M., & Prescott-Gagnon, E., 2010; Prescott-Gagnon, E. et al., 2010; Goel, A., & Vidal, T., 2013; Goel, A. & Irnich, S., 2016; Goel, A., 2018; Tilk, C., & Goel, A., 2020; Mayerle, S., et al., 2020; Vital, F. & Ioannou, P., 2021; Sartori, C., Smet, P., & Vanden-Berghe, G., 2022). When applied to the shortest path problems, TDSP is relatively straight forward since the sequence of activities to visit is fixed. However, the preemption of activities and the decision on breaks make it challenging task. On the complexity of the problem, there are two conjectures: the TDSP is Non-deterministic Polynomial-time Hard (NP-Hard) under the Hours Of Service regulation in the United States (Xu et al., 2003) and NP-Complete under the EC social legislation (Drexl & Prescott-Gagnon, 2010).

Additionally, concerning the problem's complexity, the night working rule restricts the daily working time to a maximum of ten hours within a 24-hour period when night work is involved. This rule presents two challenges for algorithmic solutions, especially for label-setting algorithms, which are commonly employed in solving the TDSP. The first challenge involves the need to update resources within a sliding 24-hour time window. The second challenge is the design of efficient dominance rules to achieve optimality within a reasonable computational time. The night working rule necessitates knowledge of the distribution of working time over the previous 24 hours, which extends the time required for comparisons and impacts the overall algorithm performance.

This paper introduces an exact label-setting algorithm for the Truck Drivers Scheduling Problem that optimizes the completion time of a customer sequence, considering all the rules in the EC social legislation, including the night working constraint, for a weekly planning period. Unlike previous studies that aim to find feasible solutions, our break scheduling method provides the optimal solution if one exists. Computational experiments demonstrate that our approach is faster than a MILP program solved by a commercial solver. Additionally, our results highlight the significant influence of the night working constraint on schedule feasibility.

The paper is structured as follows. Section 2 briefly reviews the literature related to the problem. Section 3 presents the set of rules from the EC social legislation. The mixed integer linear programming model is described in section 4. Section 5 presents the label-setting algorithm. Section 6 reports the results of the computational experiments, and section 7 summarizes our main contributions.

2. Literature review

Prior to the current EC social legislation taking effect in 2007, some attempts were made to incorporate breaks and rests within vehicle routing problems, as explored by Rochat & Semet (1994), Brandao & Mercer (1997), and Savelsbergh & Sol (1998). The Truck Drivers Scheduling Problem (TDSP) has been addressed in various studies, either independently or as a subproblem of vehicle routing procedures. Goel (2009) was among the first to consider scheduling breaks and rests in compliance with the EC social legislation after its implementation in April 2007. The problem includes regulations for full breaks based on driving time, maximum daily driving time, and daily rests within a 24-hour period. Goel (2010) presents an algorithm to calculate feasible schedules for truck drivers' working times and driving hours, considering European Union Regulation (EC) No. 561/2006. Specifically, the rules incorporated by Goel involve breaks for driving and daily rest periods within a 24-hour period. The possibility of splitting driving breaks and daily rests into two parts is also explored. Drexl & Prescott-Gagnon (2010) subsequently developed different versions of a label-setting algorithm. One version is exact, meaning it finds a legal schedule if one exists, while two heuristic approaches offer faster computational times but do not guarantee finding a feasible schedule even if one exists. The paper concludes with the suggestion that the TDSP is a Non-deterministic Polynomial-time complete (NP-complete) problem.

Kok et al. (2010) propose a Dynamic Programming heuristic to solve the Vehicle Routing Problem with Time Windows (VRPTW), considering both (EC) Regulation No 561/2006 on driving hours and Directive 2002/15/EC on working hours over a period of more than one week. However, their study does not account for the night working constraint mentioned in the working hours directive. Their results surpass those obtained by Goel (2009). Later, Prescott-Gagnon et al. (2010) develop a Large Neighborhood Search (LNS) algorithm for a VRPTW under the EC social legislation. Notably, the night working rule is not taken into account in this publication either. They compare their approach with Goel (2009) and Kok et al. (2010) using both the basic set of rules and the extended set of rules. The results demonstrate that their LNS algorithm outperforms the previous two methods.

Kok et al. (2011) tackle the VRTDSP using a hybrid approach that combines an insertion heuristic and a linear model. They incorporate (EC) Regulation No 561/2006 on driving hours into the Integer Linear Programming model to optimize departure times and use a continuous piecewise linear function to model driving time between customers. Their experiments demonstrate that incorporating time-dependent driving times and drivers' regulations in the formulations is crucial for practical use of VRP routes. Subsequently, Goel (2012c) also presents a generic Mixed Integer Programming (MIP) model for the TDSP, specifically designed to accommodate the EC social legislation and the United States Hours of Service regulations. Additionally, Goel develops a dynamic programming algorithm that outperforms the MIP solver in terms of speed.

Goel & Vidal (2013) present a hybrid genetic algorithm to solve the VRTDSP under different regulations governing drivers' hours in different regions such as the United States, Canada, the European Union, and Australia. They consider all rules of the EC social legislation, except for the night working rule. The study includes a comparison with the results obtained by Prescott-Gagnon et al. (2010) under two sets of rules referred to as No Split and All. By conducting a Wilcoxon test, they demonstrate that the mean of the solutions differs significantly, providing better results. However, their running times are 4.9 and 2.6 times longer, respectively. Furthermore, the study reveals that the European Union regulations lead to the highest safety level, while the Canadian regulations are the most economically competitive.

The first exact algorithm for solving the VRTDSP is introduced by Goel & Irnich (2016). The regulations considered include the United States Hours of Service and EC social legislation.

However, the EC social legislation rules related to the working time directive, daily driving time extensions, and daily rest reductions are not included in their work. Goel (2018) builds upon this research by addressing two additional aspects in VRTDSP problems: night working and minimizing the number of working days. In this contribution night work is forbidden, thus night periods are covered by breaks/rests or waiting time. Subsequently, Tilk & Goel (2020) expand upon this work by considering both European Union social legislation and U.S. Hours of Service regulations. They improve the average computational time by implementing a bidirectional label-setting algorithm instead of the previous forward label-setting version.

Recent contributions consider basic set of rules from the truck driver's legislations to include additional features or restrictions to their models. Mayerle et al. (2020), Cordieri et al. (2022) and Mor et al. (2022) focus on the problem to find a path between an origin-destination pair, while considering a set of potential stop locations to schedule rests or breaks and refuel. Instead of considering refueling stops, Vital, F. & Ioannou, P. (2021) under the United States Hours of Service driver's legislation (US-HOS) present the Shortest Path and Truck Driver Scheduling Problem with Parking Availability constraints (SPTDSP-PA), which studies the interaction between US-HOS-compliant scheduling, path planning and time-dependent parking availability. Later, Sartori et al. (2022) introduced the Truck Driver Scheduling Problem with Interdependent Routes, focusing on European Union social legislation. Their objective was to create feasible schedules that adhere to customer time windows and EU social legislation.

As shown in Table 1, the TDSP is mostly considered as a sub-problem of a VRTDSP. In the papers presented, the main objective is to find feasible or legal schedules; only in Kok et al. (2011), Goel (2012) and Peña-Arenas et al. (2021) present MILP formulations to find optimal schedules. In this regard, no other type of optimal method has been proposed to solve the TDSP. Finally, only three contributions have tackled the night working rule.

Table 1

Papers on the TDSP and VRTDSP under EC social legislation.

Articles	TDSP	VRTDSP	Feasible TDSP	MILP TDSP	Nigh rule
(Rochat and Semet, 1994)		X	X		
(Branda and Mercer, 1997)		X	X		
(Savalsbergh and sol, 1998)		X	X		
(Goel, 2009)		X	X		
(Kok, Meyer, Kopfer and Schuttn, 2010)		X	X		
(Prescott-Gagnon, Desaulniers, Drexl and Rousseau, 2010)	X		X		
(Kok, Hans and Schutten, 2011)		X		X	
(Goel, 2012)	X			X	
(Goel and Vidal, 2014)		X			
(Goel and Irnich, 2016)		X	X		X
(Tilk and Goel, 2020)		X	X		X
(Pena-Arenas, Garaix, Lacomme and Tchernev, 2021)	X			X	X
(Sartori, Smet and vanden, 2022).	X		X		
(Cordieri, S.A., Fumero, F., Jabali, O. and Malucelli, F., 2022)		X	X	X	
(Mor, A., Archetti, C., Jabali, O., Simonetto, A. and Speranza, G., 2022)		X	X		

3. EC social legislation and assumptions

A break is a period in which the driver does not have to deliver a service (referred to as "other work" in the official documents) or a driving activity, and uses it to recuperate, while a rest is a break of at least nine hours. Working time includes driving and service activities (loading/unloading, cleaning the vehicle, etc.). In addition, there are Periods of Availability (POA), meaning that the driver is neither working nor taking a break or a rest. A shift is a period that starts at the end of a rest and finishes at the beginning of the next rest.

3.1 Rules

Using a planning horizon of one week, the set of rules in EC social legislation considered here are as follows:

R1. A driver cannot drive more than $\overline{D} = 4.5h$ without a full break. A full break is a break of more than 45 min. that can be split into two breaks, a first break of at least 15 min. (and less than 45 min.) and a second break of at least of 30 min.; breaks should be taken in this order.

R2. A driver cannot work more than $\overline{W} = 6h$ without a break of at least 15 min..

R3. The maximum driving time in a shift is $\overline{SD} = 9h$, which may be extended to $\overline{\overline{SD}} = 10h$ twice $E^D = 2$ per week.

R4. The minimum total duration of breaks to schedule in a shift depends on the total working time in the shift w and the maximum working time limits $\overline{SW^k}$, as detailed in Table 2.

Table 2

Conditions regarding Shift Breaks.

	Total break duration b	Total working time w
a.	$15\ min \leq b$	$\overline{SW^1} = 6h = w$
b.	$30\ min \leq b$	$\overline{SW^1} = 6h < w < \overline{SW^2} = 9h$
c.	$45\ min \leq b$	$\overline{SW^2} = 9h \leq w$

R5. Three types of rests are allowed under the conditions of Table 3 on the shift spread. Only three rest reductions $E^R = 3$ are allowed per week, condition b.

Table 3

Conditions regarding Rest Patterns.

	Rest duration r	Shift spread s
a.	$11\ h \leq r$	$s \leq 13h$
b.	$9h \leq r < 11h$	$s \leq 15h$ if $E^R \leq 3$
c.	A split rest with a first period during the shift of at least 3 h and the final rest of at least 9 h	$s \leq 15h$

R6. If some night working time is performed in an interval of $\overline{N} = 24h$, then the total working time in this interval is limited to $NW = 10h$.

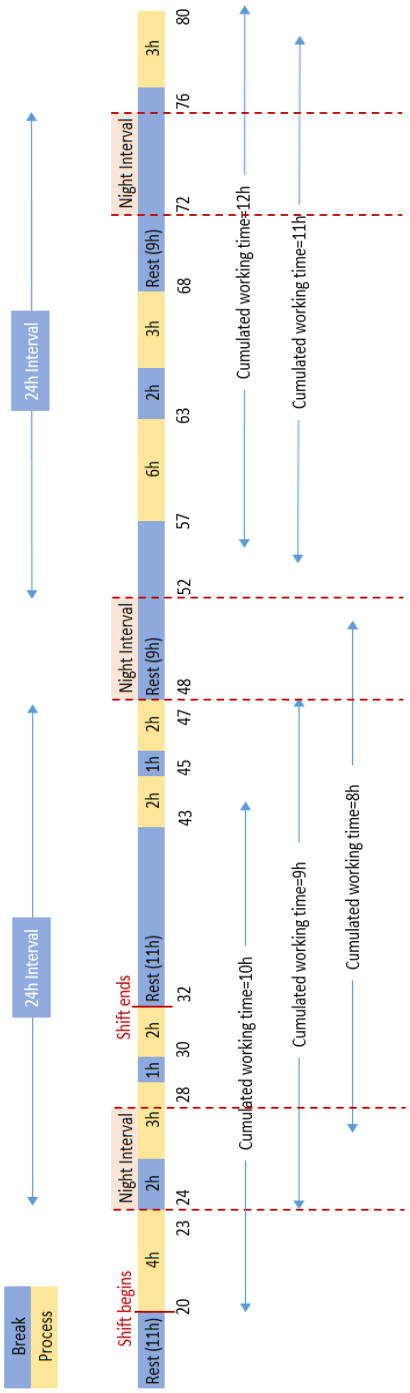

Figure 1

Gantt diagram of activities.

Within a planning horizon of one week, most of the rules apply during a shift, so the resources related to them, such as working time, daily driving time, shift duration, etc., reset their values to zero when a shift ends and a rest/break of 9 h or more takes place.

In addition, R5 implies that the time elapsed between the beginning of a shift and the minimum rest duration related to this shift has to be less than 24 h.

Figure 1. shows a solution were there are some breaks and rest to perform. Unlike other resources, night resources do not reset their values to zero at the end of a shift (after a rest). Hence, for any interval or time window of 24h, the total working time and the indication of whether night work has been performed or not should be dynamically updated. For instance, during intervals [20, 44] and [24, 48] night work is performed, and the total working time is 10h and 9h, respectively. It is worth noting that there are several other intervals between these two, and rule R6 as well applies to them. On the contrary, for the last two depicted intervals, as no night work is performed, there are no restrictions on the working time. Consequently, adding this ‘‘sliding window’’ of 24 h in set of rules, introduces a ‘‘history dependence’’ that was not previously present, making far more difficult to solve the problem.

The EC rules are not clearly defined, mathematically, and different settings can match the rule’s descriptions, especially when pre-emption applies.

3.2 Assumptions

We list the assumptions we made. They may differ to the settings adopted in other papers (Kok et al., 2010; Prescott-Gagnon et al., 2010; Goel, A., 2018), but these changes are not significant from a solver’s point of view.

In order to cover every rule, the following set B of types of breaks and rests is used: $\{15min., 30min., 45min., 3h., 9h., 11h.\}$. The 9-hr and 11-hr rests are extended until the beginning of the next shift, but the duration of other breaks is not extended. Almost all authors enforce this setting, since they assume that driving activities are always larger than working (Goel, A., 2009; Goel, A., 2010).

We found a single case where driving activities are less than 4.5h during the shift and the planning may take advantage of extended break durations. A shift working time greater than 9h implies a total break duration in the shift of at least 45min.

This total can be reached by two breaks of 20min. and 25min. that cannot be replaced by 15min. and 30min., for instance. Such occasional situations can be managed in our MILP and in our label setting, but with additional features that we discard in this work.

In the case of pre-emption, this setting allows an activity to be started for a very short period, and potentially finished very late. To avoid such unwanted behavior, we implement the earliest starting time and the latest completion time constraints on activities. In this sense, the latest starting time equals the latest completion time minus the service time.

We also considered that some breaks or rests may be scheduled contiguously in two cases: (i) depending on the definition of a shift – the interval between the end and the beginning of two consecutive rests – a shift may start or end with a break; (ii) the first part of a 'split-full-break' or a 'split-rest' can be scheduled before the driving or working time involved in the rule. Allowing such schedules avoids enforcing optimal solutions to schedule a very short driving or working time before the first part of the split break or rest.

4. TDSP definition and a mixed-integer linear program

In order to formally describe the optimization problem addressed in this paper, we present a MILP (mixed-integer linear program) which is similar to the MILP used in (Peña-Arenas et al., 2021).

4.1 TDSP description

A set of a fixed sequence of activities A, numbered from 1 to N, must be scheduled on a single machine (the driver) under the preemptive setting. Each activity $a \in A$ has a processing time P_a, an earliest starting time E_a, a latest completion time L_a and a type of activity, driving or other work (service). The driving type is given by attribute δ_a, equal to 1 for driving and 0 otherwise. A feasible solution of the TDSP is defined by a schedule of working periods which ensure that all activities are performed completely, in the correct order, within their time windows, and that all the regulation rules R1-R6 are satisfied.

The objective function is the completion time of the sequence, assuming that the first activity cannot start before 0 ($E_1 \geq 0$), and the sequence starts as if a daily rest of 11h finishes at time 0. The completion time is defined as the end of the last rest required to schedule all the activities. In general, the sequence of activities alternates services at customer premises and driving periods to reach the next customer. If they exist, the initial and terminal depots are assimilated with customers.

4.2 MILP formulation

In order to take into account preemption, each activity $a \in A$ is decomposed into a sequence of $N(a)$ nodes. The first (resp. last) node related to activity a is denoted $FN(a)$ (resp. $LN(a)$). A node $u \in U$ refers to activity $A(u)$ and receives all the attributes of the activity except the processing time.

The main idea of the model is to assign starting times and processing times to nodes, ensuring that the sum of processing times of nodes equals the processing time of the activity. After each node, a break, a rest or a POA can be scheduled.

As Remark 3 states, night rule R6 may significantly increase the number of splits – so nodes – to account for in an activity. Computing tight upper bounds on this number for each activity

becomes burdensome and is not investigated in this paper. Instead, a heuristic approach is used to conduct numerical experiments.

4.3 Data and variables

All the variables and parameters of the model are given in the list below, except for the activities' attributes of Section 5.1. and the parameters' notations given in the definitions of the regulation's rules.

The real starting time of the schedule is translated to time 0.

Sets:

A: the N activities to schedule.

B: the possible types of breaks and rests.

U: the complete set of nodes for all activities.

I: the night intervals.

Data:

$BD_b = \{0, 0.25, 0.50, 0.75, 3, 9, 11\}$: Minimal break and rest durations in hours.

$LB_i = \{0, 24, \dots, 144\}$: Starting time in hours of night interval i.

$UB_i = \{4, 28, \dots, 148\}$: Ending time in hours of night interval i.

$\bar{S}$ = 24 h: the maximal gap between the beginning of a shift and the end of the minimal rest required after this shift.

$\overline{N} = 24h$: Time lag between the beginning of two consecutive night intervals. Note that $\overline{N}$ could be different to 24h depending on changes of time zone, which is why it is stored in a different parameter to $\bar{S}$.

M: a large enough generic number.

ϵ: a small enough generic number used to model strict inequalities.

Decision variables:

x_u: the starting time of process at node u.

p_u: the processing time at node u.

y_{ub}: a binary set to 1 if at node u a break of type b takes place after the process; 0 otherwise.

Other variables:

$w_{uv} = \sum_{w=u}^{v} p_w$: the working time between nodes u and v.

$d_{uv} = \sum_{w=u|\delta_w=1}^{v} p_w$: the driving time between nodes u and v.

f_u: binary variable set to 1 if a full break is scheduled at node u; 0 otherwise.

g_u: a binary set to 1 if a driving extension is scheduled at node u; 0 otherwise.

sw_{uv}^1: binary variable set to 1 if the working time between u and v strictly exceeds $\overline{SW}^1 = 6h$; 0 otherwise.

sw_{uv}^2: binary variable set to 1 if the working time between u and v strictly exceeds $\overline{SW}^2 = 9h$; 0 otherwise.

h_u: a binary set to 1 if a reduced daily rest is scheduled at node u; 0 otherwise.

The additional variables bn_{ui}, an_{ui}, nw_{uv}, se_{uv}, and ss_{uv} are used locally to model certain constraints. Their definitions are included in the descriptions of constraints.

4.4 Objective function and constraints

We present the objective function and constraints to model the set of rules R1 to R6 from the EC social legislation, with focus on the night working rule constraints.

$$Min\ z = x_{|U|} + p_{|U|} \tag{1}$$

$$\sum_{b=1}^{|B|} y_{ub} \leq 1 \qquad \forall\, u = 1,..,|U| \tag{2}$$

$$x_u + p_u + \sum_{b=1}^{|B|} BD_b \times y_{ub} \leq x_{u+1} \qquad \forall\, u = 1,..,|U|-1 \tag{3}$$

$$\sum_{u=FN(a)}^{LN(a)} p_u = P_a \qquad \forall\, a = 1,..|A| \tag{4}$$

$$E_u \leq x_u \leq L_u - p_u \qquad \forall\, u = 1,..,|U|-1 \tag{5}$$

$$d_{uv} \leq \overline{D} + M \sum_{w=u}^{v-1} f_w \qquad \forall\, u, v\ \in U, u \leq v \tag{6}$$

$$\sum_{b=3}^{|B|} y_{ub} \leq f_u \qquad \forall\, u = 1,..,|U| \tag{7}$$

$$\sum_{b=2}^{|B|} y_{ub} \geq f_u \qquad \forall\, u = 1,..,|U| \tag{8}$$

$$(1 - y_{u2}) + (1 - f_v) + \sum_{w=v+1}^{u-1} \sum_{b=1}^{2} y_{wb} \geq f_u$$

$$\forall\, u = 1,..,|U|; \forall\, v = 1,\dots,u; \tag{9}$$

$$w_{uv} \leq \overline{W} + M \sum_{w=u}^{v-1} \sum_{b=1}^{|B|} BD_b \times y_{wb} \qquad \forall\, u, v = 1,..,|U|, u \leq v \tag{10}$$

$$d_{uv} \leq \overline{SD} + g_u \times (\overline{\overline{SD}} - \overline{SD}) + M \left(1 - \sum_{b=5}^{|B|} y_{u-1,b} + \sum_{w=u}^{v-1} \sum_{b=5}^{|B|} y_{wb} \right)$$

$$\forall\, u, v = 1,..,|U|, u \leq v \tag{11}$$

$$\sum_{u=1}^{|U|} g_u \leq E^D \tag{12}$$

$$M \times sw_{uv}^1 \geq w_{uv} - \overline{SW}^1 \qquad \forall\, u, v = 1, .., |U|, u \leq v \tag{13}$$

$$M \times sw_{uv}^2 \geq w_{uv} - \overline{SW}^2 \qquad \forall\, u, v = 1, .., |U|, u \leq v \tag{14}$$

$$M\left(2 - \sum_{b=5}^{|B|}(y_{vb} + y_{u-1b})\right) + \sum_{w=u}^{v-1}\sum_{b=1}^{|B|} BD_b \times y_{wb} + M(1 - sw_{uv}^1) \geq BD_2$$

$$\forall\, u, v = 1, .., |U|, u \leq v \tag{15}$$

$$M\left(2 - \sum_{b=5}^{|B|}(y_{vb} + y_{u-1b})\right) + \sum_{w=u}^{v-1}\sum_{b=1}^{|B|} BD_b \times y_{wb} + M(1 - sw_{uv}^2) \geq BD_3$$

$$\forall\, u, v = 1, .., |U|, u \leq v \tag{16}$$

$$x_v + p_v + \sum_{b=1}^{|B|} BD_b \times y_{vb} - x_u \leq \bar{S} + M \sum_{w=u}^{v-1}\sum_{b=5}^{|B|} y_{wb} \qquad \forall\, u, v = 1, .., |U|, u \leq v \tag{17}$$

$$y_{v5} + \sum_{b=5}^{|B|} y_{ub} - \sum_{w=u+1}^{v-1} y_{w4} \leq h_v + 1 \qquad \forall\, u, v = 1, .., |U|, u \leq v \tag{18}$$

$$\sum_{u=1}^{|U|} h_u \leq E^R \tag{19}$$

The objective function (1) is to minimize the completion time, i.e., the finishing time of the last node $|U|$. A dummy node is added at the end to include the rest scheduled after the end of the previous node. Only one type of break can take place at each node and is enforced by Constraints (2). Constraints (3) to (5) enforce the sequence between activities and nodes including breaks/rests. Constraints (6)-(9) determine if the break at node u can be considered as a full break. Constraints (10) ensure that the working time w_{uv} between u and v is less than $\overline{W} = 6h$ if there is no break in between them. The driving time during a shift is limited to $\overline{SD} = 9h$ in Constraints (11) with a possible extension of one hour $\overline{\overline{SD}} - \overline{SD} = 1h$; constraints (12) limit such an extension at $E^D = 2$ times per week. Binary variables sw_{uv}^1 and sw_{uv}^2 are set to 1 by Constraints (13)-(14) if the working time between u and v strictly exceeds $\overline{SW}^1 = 6h$ and $\overline{SW}^2 = 9h$, respectively. In Constraints (15), all pairs of nodes u and v that bound a shift ($\sum_{b=5}^{|B|}(y_{vb} + y_{u-1b}) = 2$) need to compute more than BD_2h of break ($\sum_{w=u}^{v-1}\sum_{b=1}^{|B|} BD_b \times y_{wb} \geq BD_2$) if more than $\overline{SW}^1$ h are worked in the shift ($sw_{uv}^1 = 1$). Constraints (16) are similar to (15) for condition (c) on shift breaks (Table 2), where a break of $BD_3 = 0.75$h has to be scheduled if the working time in a shift exceeds $\overline{SW}^2 = 9$h. Constraints (17) determine the shift spread durations related with rule 5 from Table 3. Constraints (18) establish the conditions for using reduced rests of 9h, which are restricted to E^R times per week by Constraint (19).

R6 - Night-working

In order to implement this rule, two issues have to be addressed: to identify intervals where night working is performed and to identify intervals of more or less than 24h between nodes.

We need to add the binary variable nw_{uv} that is set to 1, if some night working is performed between u and v.

In order to compute the value of nw_{uv}, we introduce two families of binary variables bn_{ui} and an_{ui} that show whether the node u is processed, respectively before or after, the night-interval i, as described by Constraints (20)-(23).

$$M \times bn_{ui} \geq LB_i - x_u \qquad \forall\, u = 1,..,|U|; \forall\, i = 1,..,|I| \qquad (20)$$

$$M(1 - bn_{ui}) \geq x_u + p_u - LB_i \qquad \forall\, u = 1,..,|U|; \forall\, i = 1,..,|I| \qquad (21)$$

$$M \times an_{ui} \geq x_u + p_u - UB_i \qquad \forall\, u = 1,..,|U|; \forall\, i = 1,..,|I| \qquad (22)$$

$$M(1 - an_{ui}) \geq UB_i - x_u \qquad \forall\, u = 1,..,|U|; \forall\, i = 1,..,|I| \qquad (23)$$

This set of constraints forbids the processing time of each node from overlapping day and night.

With Constraints (24), variable nw_{uv} is set to 1 when there is at least one working node $k \in [u, v]$ that performs a process ($p_k > 0$) during a night interval i, *i.e.*, k is neither before nor after i.

$$nw_{uv} \geq \frac{p_k}{P_{A(k)}} - bn_{ki} - an_{ki} \qquad \forall\, u, v = 1,..,|U|; \forall\, k \in [u, v]; \; P_{A(k)} > 0; \forall\, i = 1,..,|I| \qquad (24)$$

The binary variable se_{uv} is introduced to indicate if the time elapsed between the starting time of u and the ending time of the process of v is less than or equal to $\overline{N} = 24h$. Constraints (25) and (26) set the value se_{uv} to 1 in this case and 0 otherwise.

$$M \times se_{uv} \geq \overline{N} - (x_v + p_v - x_u) + \epsilon \qquad \forall\, u, v = 1,..,|U|, u \leq v \qquad (25)$$

$$M(1 - se_{uv}) \geq (x_v + p_v - x_u) - \overline{N} \qquad \forall\, u, v = 1,..,|U|, u \leq v \qquad (26)$$

With all these elements, the main Constraints (27) bound the working time to $NW = 10h$. if some night-work is performed during a $\overline{N}$-interval.

$$w_{uv} \leq NW + M\,(2 - se_{uv} - nw_{uv}) \qquad \forall\, u, v = 1,..,|U|, u \leq v \qquad (27)$$

However, due to the definition of se_{uv} , Constraints (27) do not cover the case where the starting time of u plus $\overline{N}$ computes a value in the middle of the process of v, *i.e.*, $x_v \leq x_u + \overline{N} < x_v + p_v$. Both situations are described in Figure 2.

The additional binary variables ss_{uv} are set to 1 by Constraints (28) if the gap between the starting times of u and v is less or equal than $\overline{N}$.

$$M \times ss_{uv} \geq \overline{N} - (x_v - x_u) + \epsilon \qquad \forall\, u, v = 1,..,|U|, u \leq v \qquad (28)$$

Constraints (29) ensure that the case of a $\overline{N}$-interval finishing in the middle of v is covered.

$$w_{uv} - [x_v + p_v - x_u - \overline{N}] \leq NW + M\,(2 - ss_{uv} + se_{uv} - nw_{uv}) \qquad \forall\, u, v = 1,..,|U|, u \leq v \qquad (29)$$

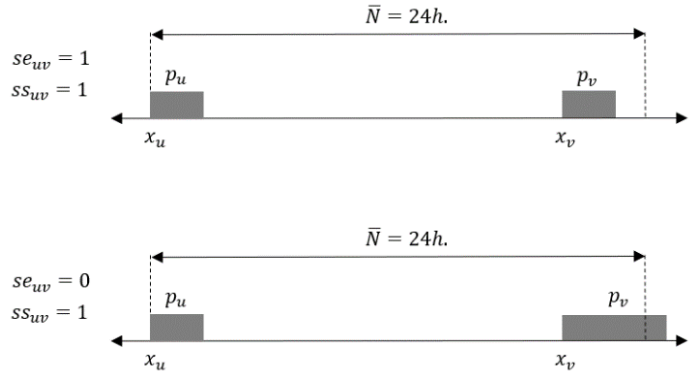

Figure 2

Night constraint between activities u and v.

Note that only one part of p_v is included in the $\bar{N}$-interval. As processing times are all included in or excluded from (Constraints (20)-(24)) night-intervals, the value of nw_{uv} is consistent for any interval of size $\bar{N}$ that cuts p_v.

In order to keep the equations simple, we discarded the following usual improvements in MILP: (i) the reduction of the domain of some constraints, for instance some pairs of nodes may be discarded from Constraints (29) if the working time between these nodes cannot be less than $\bar{N}$. (ii) the replacement of value *M* by a tight upper bound computed for each constraint. (iii) the calculation of tight upper bounds on some parameters such as the earliest starting times of nodes. However, the code shared at https://emse.fr/~garaix/TruckDriverSchedule/Expert_Systems/ implements these improvements of the model.

However, the main issue of this model is the number of nodes used for each activity, which are difficult to compute and are highly correlated to the performance of the model. This point is discussed in section 7. We propose a discrete label-setting algorithm to overcome this difficulty and to provide an alternative solution method that could be used in an integrated vehicle routing schema.

5. Optimal solutions analysis

In this section, we present a series of properties of optimal solutions used to design the two algorithms that we propose. Because of the night rule, several properties from the literature (Drexl & Prescott-Gagnon, 2010) do not hold. Therefore, we describe a new set of properties based on some dominant sets definitions. The properties proposed in this section are not verified by all optimal solutions.

Property 1. *The rest periods can be extended forward to the next break or working period or rest.*

Proof. If a rest is followed by a time period that is neither work, break, rest nor POA, the extension of this rest does not impact any rule. If POA is scheduled after the rest, the extension of this rest changes the starting time of the shift (started by the POA) to the end of the POA. The shift spread is reduced, and that cannot violate any rule. Note that two contiguous rests are considered as a single big rest. □

Property 2. *Any time period that is neither work nor break in the middle of a shift can be replaced by POA.*

Proof. As the POAs are not involved in the working time, no rule can be violated by this change. □

Any time period can be considered as either work, break, rest or POA without losing optimality, as a result of Properties 1 and 2. Actually, such time periods can be replaced by POA or rest extensions if they are in the middle or at the beginning of the shift. In addition, according to Property 1 it is possible to find optimal solutions that do not start by POA.

Property 3) *The number of dominant patterns for breaks in a shift is large, but they all satisfy the following rules:*

1. At most one first part of a split rest of 3h.
2. At most three breaks before a 15-min. break.
3. A 45-min. break is scheduled only after a driving period if a split-full break is not opened.
4. A 30-min. break is considered only as the first break of the shift or as the last part of a split-full break.

Proof. The first rule is obvious. After four breaks, enough driving and working time has been scheduled to guarantee that R2 (which enforces breaks of 15min.) does not apply at this point, as the second rule states. The statement of rules 3 and 4 are related to R1. □

Property 4) *Dominant sequences of breaks/rests at the end of a shift are limited to the thirty patterns resulting from the union of the two Cartesian products* $\{-;\ 15min.;\ 30min.\} \times \{9h\ ;\ 11h\} \times \{-;\ 15min.;\ 30min.;\ 3h\} \cup \{-;\ 15min.;\ 30min.\} \times \{9h\ ;\ 11h\} \times \{3h\} \times \{15min.\}$.

Proof. Breaks of 15min. and 30min at the beginning or the end of a shift could be used to satisfy R4. A break of 45min. can satisfy rule R4 with shift work greater than 9h, but rule R2 enforces a break of 15min. in the middle of the shift with more than 9h (6h, indeed) of working, and therefore a break of 30min. dominates a break of 45min. Therefore, 45min. is not considered at the end or beginning of a shift. A split rest (3h + 9h = 12h) scheduled at the end of a shift is dominated by a rest of 11h. Therefore, the 3-hr break can be discarded from the set of dominant breaks before a rest. A dominant start for a shift is to schedule the first part of a split-full break and/or a split rest. This remark defines the sets that follow the rest. □

Property 5) *Dominant sequences of breaks in the middle of a shift are limited to the Cartesian product* $\{-;\ 15min.;\ 30min.;\ 45min.;\ 3h\} \times \{-\ ;\ 15min.\}$.

Proof. If there are multiple rules to satisfy at the same time, the longest break required is selected. Only one case does not match this situation, when the first part of a split-full-break is scheduled just after a full-break. □

Property 6) *Split breaks/rests are useful only when waiting time can be filled by the first part of the split break/rest.*

Proof. See Prescott-Gagnon et al. (2010). □

Remark 1. Because of possible waiting times during a night, the 3h of a split-rest can be scheduled in the middle of an activity despite Property 5).

Definition 1. A Near-Active-Schedule (NAS) is a solution where POA neither precedes nor follows a rest and can be scheduled only before a working period, under at least one of the following conditions:

1. Just before the earliest starting time of an activity.
2. Just before the beginning of the day; at the end of the night interval.

3. Just before the end of a $\overline{N}$-interval where night rule R6 is saturated, i.e., some night-work has been performed and the NW (=10h) of accumulated working time has been reached.
4. Just before the end of a $\overline{N}$-interval with more than or equal to NW of accumulated working time and which ends during night.

Property 7) *NAS are dominant for the TDSP.*

Proof. We show that an optimal no NAS S can be transformed to an optimal NAS S'.

Any rest can be extended to cover the neighbors/adjacent POA periods without violating any constraint and any increase in the completion time, since a reduction of the spread of the preceding and following shifts are expected after such an extension.

If a break is scheduled after a POA period, then the swap of the two periods does not impact some working periods or shit spreads, and therefore no rule may become unfeasible. Let S be an optimal schedule that does not satisfy NAS. This means that there are some POA scheduled between time t and t plus the size of the POA d $[t, t+d]$ outside the NAS conditions (1)-(4), as in Figure 3. After this POA a break/rest or working is scheduled.

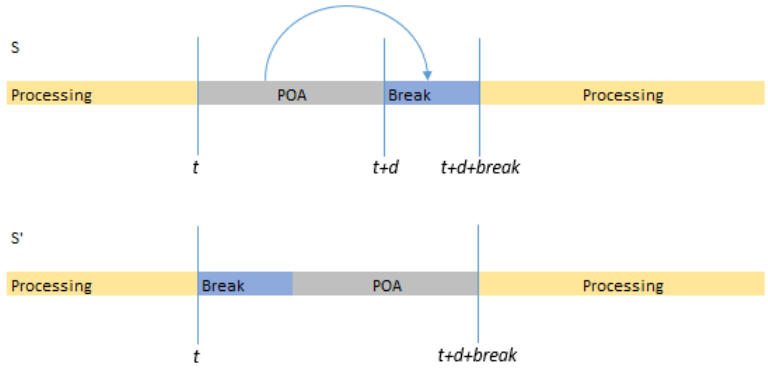

Figure 3

Modifying a No NAS Solution with POA Before Break.

If the POA is followed by a break/rest in optimal schedule S, it is easy to check that the move forward of the POA does not impact any rule, since neither the accumulated working and driving times involved in any rule nor the shift spread (remember that POA cannot end a shift) are modified.

Let us consider the last case where a working period WP of duration w follows a POA AP of duration d, and none of the NAS conditions (1)-(4) are satisfied for AP. We assume that AP starts at t, and therefore AP and WP cover respectively the intervals $[t, t+d]$ and $[t+d, t+d+w]$ as in Figure 4. We derive an alternative schedule S' from S by moving backwards the first e time units of W with $0 < e \leq min(d, w)$. In S', AP covers $[t, t+d-e] \cup [t+d, t+d+e]$ and WP covers $[t+d-e, t+d] \cup [t+d+e, t+d+w]$. Obviously, this change does not increase the completion time; we have to show that is S' feasible. If S' is not NAS, this transformation can be repeated until the NAS conditions are met by the resulting schedule S'.

The earliest starting time of the activity related to working time e cannot forbid the move; since S is not NAS, the NAS condition (1) cannot be satisfied by S.

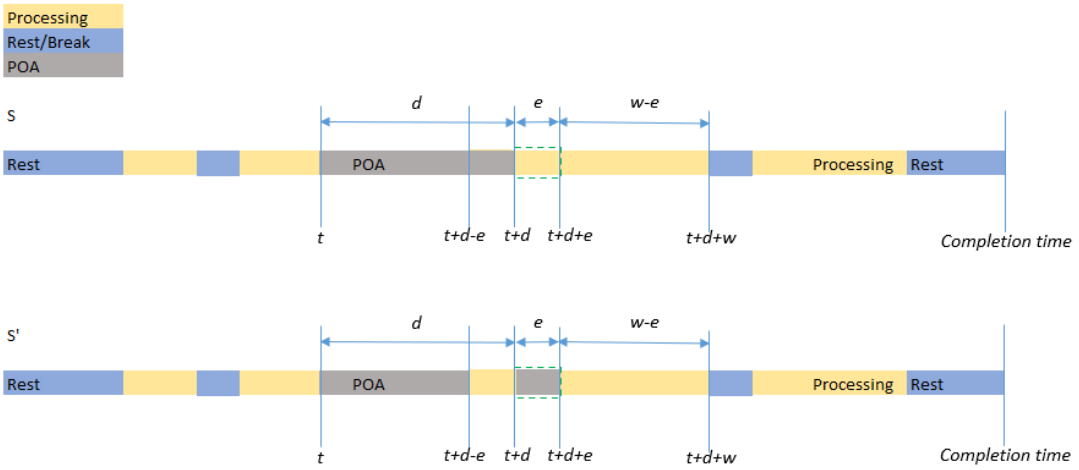

Figure 4

Moving backwards e working time within a no NAS solution.

By construction, S' does not modify any quantity of working or driving time involved in satisfying rules R1-R4, since the working time moved remains in the same break interval (the period between two consecutive breaks) and shift. Also, no shift spread is modified since shifts start and ends with rests only (see Property 2). Therefore, the rule R5 is also satisfied by S'.

Rule R6 can be violated in S' only in one 24-hr interval that includes the time slot $[t + d - e, t + d]$ where working time is added.

First, we consider interval $I = [t + d - 24h, t + d]$ under different cases which combine three components: W the accumulated working time during I; if night working is performed or not during I; the night status of $[t + d - e, t + d]$. The feasibility conditions related to the NAS definition are displayed in Table 4.

Table 4

R6 Satisfaction Conditions.

W	Night working in I	$[t + d - e; t + d]$ is night	Feasibility	NAS cond.
>10 h	No	Yes	Unfeasible	4
≥10 h	No	No	Feasible	
=10 h	No	Yes	Unfeasible	4
=10 h	Yes	-	Unfeasible	3
<10 h	-	-	Feasible	

Second, we consider all the other intervals $[j, j + 24h]$ with $t + d - 24h < j \leq t + d$. These intervals also include the original position of the moved work of duration e. As S is feasible, R6 satisfaction in S' only depends on a change on the status of night working during one of these $\bar{N}$-intervals where the accumulated working time remains constant. Therefore, infeasibility can be brought only if $[t + d, t + d + e]$ does not cover night and $[t + d - e, t + d]$ does; this case is rejected by the second NAS condition.

Finally, repeating the move from S to S' builds a new NAS feasible solution without increasing the completion from any non-NAS solution. □

Definition 2. A Near-Semi-Active-Schedule, NSAS, is a NAS where a not working period B (a POA, a break or a rest) is scheduled at t before a working period WP iff at least one of the following conditions is met:

1. The earliest starting time of the first activity related to WP is greater than t.
2. The limit of accumulated working or driving times of at least one of rules R1-R4 is reached at t.
3. B is a rest and the limit of at least one of the conditions of rule R5 on the spread of the shift finished by B is saturated.
4. The accumulated working time in $I = [t - 24h, t]$ is greater or equal than NW (=10h). and t starts or is in the middle of a night period.
5. B covers some periods during the night and the following working time WP is during the day.
6. Some night-work is performed during $I = [t - 24h, t]$, the cumulated working time in I equals NW, and I does not start with a working period.

Property 8) *NSAS are dominant for the TDSP.*

Proof. Let S be an optimal schedule that does not satisfy NSAS. Therefore S schedules this sequence outside the NSAS conditions (1)-(6): a not working period of duration d in $[t, t + d]$ and a working period WP during $[t + d, t + d + w]$, as in Figure 5.□

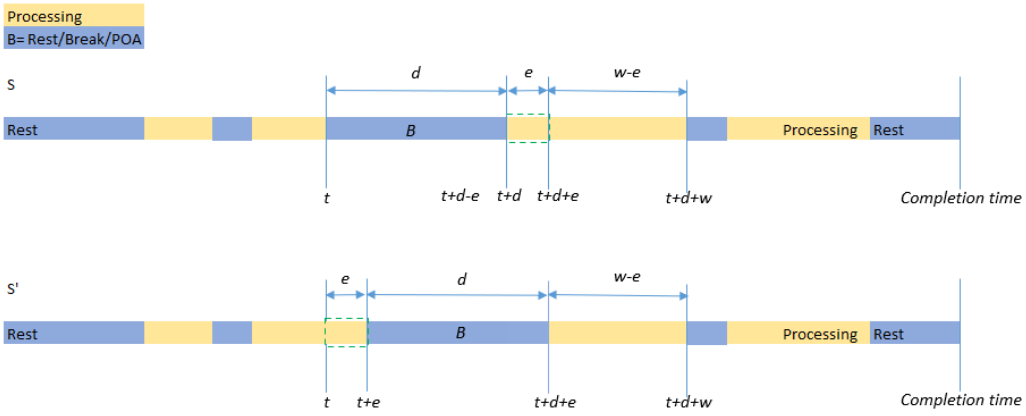

Figure 5

Moving backwards e working time before POA/Break/Rest within a no NSAS solution.

Let us consider the solution S' obtained by swapping B and $0 < e \leq w$ working time in S, in such a way B starts at $t + e$. It is obvious that the completion time of S' is not greater than the completion time of S. There is also no doubt about the feasibility of S' considering rules which include time intervals after $t + d$, since some working/driving time is replaced by not working time.

As the NSAS conditions (1), (2) and (3) are violated in S, a small enough value e can be found to make S' feasible regarding the rules R1-R5.

In order to check R6, the conditions (4), (5) and (6) are considered. The proof considers several cases and subcases. In this section we denote $I = [t - 24h, t]$ and W the total working time in I.

Let us consider two opposite cases: (i) there exists $f > 0$ such that $[t, t + f]$ is a night-interval, and (ii) there exists $f > 0$ such that $[t, t + f]$ is a day-interval.

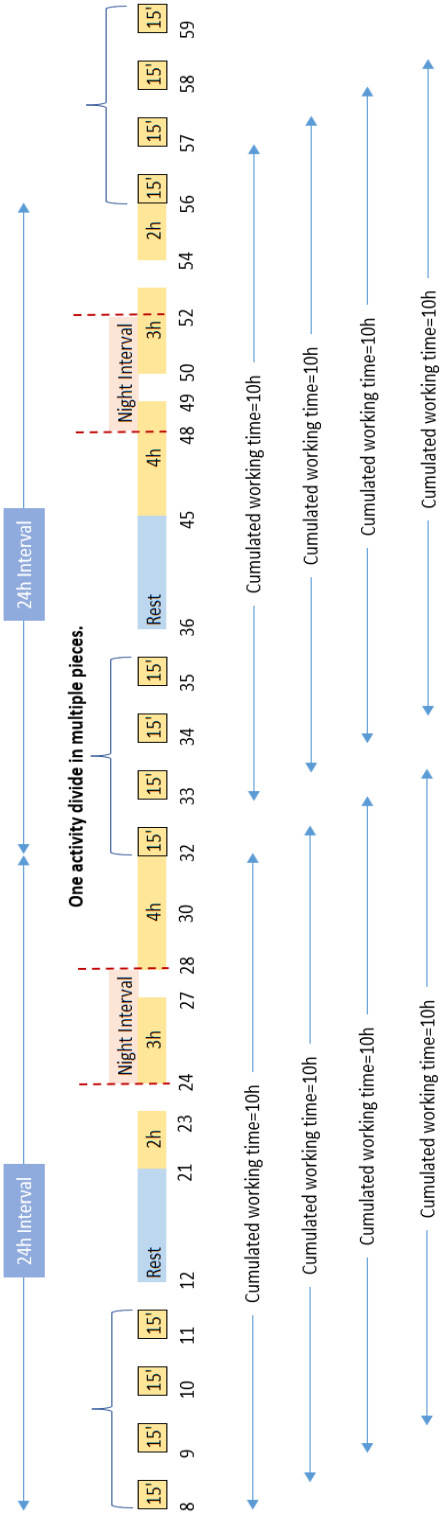

Figure 6

One activity divided in multiple pieces due to the night working constraint.

In Case (i), the violation of the NSAS condition (4) implies that W is strictly less than NW(=10h). Then, the selection of $0 < e \leq \min(d, w, f, NW - W)$ ensures that S' satisfies R6 over $[t + e - 24h, t + e]$. Thus, all the 24h. intervals ending between $t + e$ and $t + d + e$ also satisfy R6, since their total working time is less or equal than NW. The violation of the NSAS condition (5) by S in Case (i) implies that $[t + d, t + d + e]$ covers some night. Then, any interval in S that includes $[t + d, t + d + e]$ cumulates at most NW of working time. As, the transformation to S' does not increase the total working times of these intervals, we can conclude that the feasibility of R6 in S' is ensured in Case (i).

In Case(ii), we consider the backward move of a working period $0 < e \leq \min(d, w, f)$ from $t + d$ to t, to create S'. By construction $[t, t + e]$ is in day and this move does not increase the total night-work in any $\bar{N}$-interval. In order to satisfy R6 in S', we have to check that the total working time of any $\bar{N}$-interval that includes some night-work does not exceed NW. Only $\bar{N}$-intervals that end between t and $t + e$ have an increased total working time.

As e can be selected as small as possible, we can limit this study to $J = [t + e - 24h, t + e]$, without loss of generality. Let us consider the three subcases that violate the NSAS condition (6). First, if no night-work is performed during I, R6 does not apply for J in Case(ii). A value of W different from NW is the second way to violate the condition (6). A greater value implies that no night-work is performed during I and the previous argument can be used. Otherwise, with $W < NW$, a value of e small enough to satisfy $e \leq \min(d, w, f, NW - W)$ ensures the satisfaction of R6 for J. The third option to violate condition (6) implies that I starts with a working period of duration g>0. Then e has to be selected lower or equal than g. Thus, the total working time of J equals W and R6 cannot be violated in S'.

By successive backward moves of $e > 0$ working time we show that it is possible to build a feasible NSAS solution from any feasible non-NSAS solution without increasing the completion time. □

Remark 2. Because of NSAS conditions (1) and (5), the sequences of breaks/rest at the beginning of a shift (see Property 4)) only occur when the schedule is ahead compared to the earliest starting time of the next activity, or when some night period is covered by the sequence.

Remark 3. As the night-working rule R6 applies on every interval $\bar{\boldsymbol{N}}$, one can imagine solutions where an activity is split into many pieces of work separated by POA in order to satisfy the $\boldsymbol{NW}$ limit on the working time, see

Figure **6**. In the case of the MILP formulation, this kind of solutions make difficult to compute the optimal number of nodes for a given activity.

6. Label-setting algorithm

In this section we describe an optimal label-setting algorithm that can deal with activity preemption and the night-working rule R6. Basically, label-setting algorithms extend partial paths step by step from the origin to the destination. As in a dynamic program, all feasible possibilities are considered for extension. Some dominance rules serve to discard some partial paths. In theory, the computational complexity of the algorithm is given by the number of partial paths generated. The complexity of our algorithm is discussed at the end of the section.

Due to preemption, activities can be stopped at any time. Our algorithm only explores solutions that satisfy the NSAS conditions as described in Properties 7 and 8, that drastically limits the solution space described by the preemptive assumption. Especially, the night working constraints may imply many POA as in Figure 6. NSAS conditions limit the situations where POA have to be considered. The idea is that an activity is not stopped if no rule can be broken if no pause is performed at this time. The Properties 1 to 6 allow to limit the number of types of not working periods (POA/break/rest) to consider at the end of a working or not working period.

The discrete-time model used here gives an original design to our algorithm compared to the literature. In addition, specific procedures are required to control the computational complexity of extension and dominance, key procedures of the algorithm. For all these reasons this section describes in detail the model and the algorithm.

6.1 Algorithm description

A label describes a schedule from the origin (let us say the depot) to an activity. The schedule of breaks and rests taken is part of the path. Because of preemption, a path can be stopped in the middle of an activity. The discrete-time model is used to control the decision on where to cut activities; δ denote the time step.

All the labels generated are managed in $\mathcal{L}$. At each iteration a working or idle time, denoted p, is scheduled followed by a break or a rest or nothing, denoted b. The idle time is POA or an extension of the current rest. The three cases (C_i) described in Table 6 are considered to generate valid pairs (p, b) at Steps 4 to 6. The optimal working time that can be scheduled to extend each label l is computed by Algorithm 2 (in Section 6.3) at Step 3. Based on the NSAS conditions defined in Properties 7 and 8, the duration p is computed as long as possible.

Algorithm 1: LabelSetting

```
STEP 0:  ℒ ≔ {An initial label};
STEP 1:  While ℒ ≠ ∅ do
STEP 2:    l ≔ pop(ℒ)
STEP 3:    p ≔ computeP(l)
STEP 4:    forall i = 1, …, 3 do
STEP 5:      if the condition for C_i is verified then
STEP 6:        forall (p, b) ∈ C_i .
STEP 7:          Create the label m s.t. ∀ resource r, r(m) = f^r(l, p, b)
STEP 8:            if m satisfies all rules R1 to R6 then
STEP 9:              if m is not dominated in ℒ then
STEP 10:               remove from ℒ the labels dominated by m
STEP 11:               if a(m) is different from the last activity then
STEP 12:                 ℒ ← m
STEP 13:               else
STEP 14:                 Best ← m
STEP 15:                                                          Return
the solution computed from one of the label in Best with the lowest completion time.
```

We do not exploit any analytical result to control the length of the idle time extension. Therefore, the idle time extension is set to δ. The choice of a discrete model for this algorithm is motivated by this extension of δ. The choice between POA and rest is made according to Properties 1 and 2. Thanks to Properties 3 to 6 the set of breaks and rests to consider is limited depending on the structure of the current label.

For each valid pair (p, b) defined in Table 6, a new label is created at Step 7 through the resource extension function given in Section 6.3. For all feasible labels, the dominance rules of Section 6.4 are applied to eliminate some labels in Steps 9 and 10. Finally, the function $a(m)$ verifies the activity related to label m, and the new label can be added to $\mathcal{L}$ or to the list $\mathcal{Best}$ of non-dominated final solutions.

The attributes of each label are detailed in the next section. The initial starting label models the current situation of the driver before starting the first activity to schedule. In all instances tested, the initial situation is at the depot after a rest.

6.2 Label definition

The following are the set of resources R used to describe a label l.

t: Current time; the lower bound to schedule next process, or break/rest/POA.

a: Terminal activity of the label; the activity can be running or finished.

q: Cumulated processing time of the current activity.

ss: Starting time of the current shift.

d: Current driving time without a full break.

ds: Current driving time during the shift.

wc: Current working time without a break of at least 0.25h

ws: Current working time during the shift.

bs: Accumulated break time during the shift, if lower than 0.75h

sf: Binary marker if the first part of a split-full-break can be used.

sr: Binary marker if the first part of a split-rest can be used.

de: Number of shift driving extensions already performed.

rr: Number of reduced daily rests already performed.

f: Link to the father label.

p: Processing or idle time (POA or rest extension) to schedule at $t(f)$.

b: Type of break/rest to schedule at $t(f) + p$.

fw: Link to the first predecessor label m (in a backward search) with $p(m) > 0$.

tw: Total working time during $[t - \bar{N}, t]$.

wb: Duration of the continuous working time that starts at $t - \bar{N}$ and ends at the beginning of the first break or POA.

n: Finishing time of the last period where night-work is performed in the label.

The two main decisions made at each label extension are the values of p and b, which define processing or idle time followed or not by a break or a rest. Note that sequences of breaks are generated by consecutive extensions with breaks, but $p = 0$.

The attribute fw is used to get the working periods over the interval $[t - \bar{N}, t]$ through a backward parsing of the labels. The last three attributes n, tw and wb are used to speed up the dominance and feasibility checking procedures related to R6 and described in the following sections.

Compared to the label definition in Goel & Irnich (2016), who modeled labels through 'times to each type of break/rest', we instead used attributes that describe the different working times and breaks of the partial solution. Our formulation does not improve the efficiency of the algorithm, but we think this formulation is easier to link to the rules and the decisions when the final solution has to be retrieved. At all events, it is easy to switch from one formulation to another one. Of course, the last three attributes are specific to our model since they control the night rule, which is not implemented in Goel & Irnich (2016).

6.3 Resource extension functions

A label l is extended to a label l' according to the (p, b) and the functions given in Table 5. The resource extension functions are non-decreasing until some discontinuity values that correspond to a rule violation (and so to break/rest/POA periods). Thus, the label-setting algorithm converges to an optimal solution when all the possible extensions are considered at each discontinuity value of each resource.

Table 5

Resource-Extension Functions.

$\boldsymbol{l'}$	**= *Resource-extension functions*** $\boldsymbol{f^r(l,p,b)}$						
b	$0h.$	$0.25h.$	$0.5h.$	$0.75h.$	$3h.$	$9h.$	$11h.$
a'	a, if $p+q<P_a$ and $a+1$, otherwise.						
t'	$t+p+BD_b$						
q'	$q+p$, if $a'=a$ and 0, otherwise.						
ss'	ss					t'	
d'	$d+p\times\delta_{a'}$		$0, if\ (sf=1\ \wedge\ sf'=0)$ and $d+p\times\delta_{a'}$, otherwise.	0			
ds'	$ds+p\times\delta_{a'}$					0	
wc'	$wc+p$	0					
ws'	$ws+p$						
bs'	$min(BD_3=0.75h\,;\ bs+b)$					0	
sf'	sf	1	1, $if\ sf$=0 and (0 ∧ 1), $if\ sf$=1	0			
sr'	sr				1	0	
de'	de					de+1, if $ds+p\times\delta_{a'}>9h$ and de, otherwise.	
rr'	rr					$rr+1$, if sr=0 and rr, otherwise.	rr
fw'	$fw'(l)$						
tw'	tw-working time in $[t-\bar{N};t-\bar{N}+(t'-t)]+p$ = working in $[t'-\bar{N},t']$						
wb'	the length of the working period that starts at $t'-\bar{N}$ and ends at the beginning of the first break or POA, directly computed as $wb-(t'-t)$, if $t'-t\ \leq wb$						
n'	n, when p is entirely day-work, and the minimum between $t+p$ and the end of the last night period covered by the extension of p, otherwise.						

Each feasible and non-dominated label is extended according to the following three possibilities (see Table 6) that generate a NSAS:

Table 6

Extension Cases.

	Cases	***Condition***
C1	$(p > 0, b \in B \cup \emptyset)$	$ComputeP(\,) > 0$
C2	$(p = 0, b \in B)$	$ComputeP(\,) = 0 \ \vee a\ dominant\ sequence\ of\ breaks\ is\ possible$
C3	$(Rest\ Period\ or\ POA > 0, \emptyset)$	$t\ is\ between\ two\ shifts \ \vee t\ is\ during\ the\ night$ $\vee\ the\ earliest\ starting\ time\ of\ the\ activity\ is\ greater\ than\ t$

Each possible extension describes two consecutive items to schedule. Several cases can be developed at the same iteration to generate different sons of the same label.

Case C1 may schedule a working period followed by nothing. This means that at the next iteration this label may be extended with a break or a rest only (C2) or with work (C1) or with POA (C3). The two first possibilities are required to consider soft constraints of rules on shift driving time extensions (see R3) and shift spread reductions (see R5). The computation of the length of $p > 0$ is described in the next subsection.

The schedule of a single break or rest, like in C2, is useful in the situation described above. It also generates the sequences of breaks/rests of Properties 4) and 5).

In case C3, extensions of idle time are performed in steps of size $p = \delta$, according to Properties 1 and 2, becoming rest or POA. Following the NAS definition, POA is not followed by a break or a rest. A rest can be followed by a break. This possibility is considered by the case C2 of the son of the current label. Therefore, no break or rest is considered in C3.

Due to Properties 7 and 8 as shown in the analysis of optimal solutions, the processing time (p) to be scheduled is computed as the maximal value that could be performed without violating any rule, taking into account the current label and the remaining time to finish the current activity. Therefore, the calculation of p is limited to the lowest value that may cause a rule violation, as described in Algorithm 2 $ComputeP(l)$ that assumes a feasible input label. This function is a key to generating extensions longer than step δ, and so to improving the efficiency of the algorithm.

Algorithm 2: ComputeP(l)

input:
l: Current label to extend;
output:
p: The processing time for the extension;

1. $p_1 = p_a - q$;
2. $p_2 = \overline{W} - wc$;
3. $p_3 = \overline{D} - d$;
4. $p_4 = \overline{SW}^2 - ws$ **if** $bs < BD_3$; $\infty +$ **otherwise**;
5. $p_5 = \overline{\overline{SD}} - ds$ **if** $ds > \overline{SD}$; $\overline{SD} - ds$ **otherwise**;
6. $p_6 = 15h - (t - ss)$ **if** $t - ss > 13h$; $13h - (t - ss)$ **otherwise**;
7. $p_7 = \min_{i \in I}\{LB_i : LB_i > t\} - t$;
8. $p_8 = \max\{NW + wb - tw,\ 0\}$ **if** $\exists\, i \in I\ where\ t \in [LB_i, UB_i] \vee n \geq t - \overline{N}$; $\infty +$ **otherwise**;
9. **Return** $\min_{\{k=1,\dots,8\}} p_k$;

Bounds p_1, p_2, p_3 and p_4 are easily computed. The values of p_5 and p_6 depend on whether the current schedule already exceeds the soft limit on the shift driving time extension or the rest

reduction. The three groups of expressions p_1, p_2 to p_4, and p_5 to p_6 are respectively related to NSAS conditions 1, 2, and 3.

The values of p_7 and p_8 are related to night constraints and NSAS conditions 4, 5 and 6. The expression of p_7 means that the extension is stopped at the beginning of the next night interval. Thus, two further extensions can be considered: start the night with work or not. First, we consider that the total working time in the last $\bar{N}$-interval of l does not exceed the limit imposed by R6, *i.e.* $NW \geq tw$. Let us now consider the two possible cases for p: p covers night or not. If p covers night, some processing time can be scheduled just after l, under the limit NW. As the 24h. window to consider for R6 is sliding, the total working time remains stable if the $\bar{N}$-interval $[t - \bar{N}, t]$ starts by a working period. Therefore, p is bounded by $NW - tw + wb$ in this case. In the second case, where p does not cover night, there is no limitation on p due to R6 if there is no night-work in the last $\bar{N}$-interval of l. Otherwise, p is bounded by $NW - tw + wb$, as if p covers the night.

The second case $NW < tw$ is easy to treat, since no extension that generates night work can be considered. Because of $p7$, p covers night only if $\exists\, i \in I$ such that $p \in [LB_i, UB_i]$. The presence of night work in l is determined by the equation $n \geq t - \bar{N}$. Thus, the bound $p8$ allows to cover all the cases where R6 is involved. Note, to compute p_8 the night constraint attributes fw, n, tw and wb must be updated for the last 24 hours time window at each extension. As we mentioned before in section 3, this is the history dependence that makes the algorithm more challenging.

We remark that a label l with an active night constraint R6 may be extended to a label where R6 is released; if p does not cover a night. The bound $p8$ could be relaxed in this case. Because of the regular spread of nights every 24h., such situation cannot occur, and $p8$ is tight.

6.4 Dominance rules

Any label can be discarded from the search without loss of optimality if it is dominated by another label. Label l dominates m if it can be stated that if m can be extended to reach an optimal solution, then l can be extended to a feasible label that is also optimal. Our implementation is more conservative and ensures that any feasible extension of m is feasible for l. It is interpreted at the resource level in Table 7, where all the constraints on the resource consumptions must be less restrictive for l than for m, as the comparison of their current completion times.

Table 7

The Dominance Conditions of Label l over m.

Resource	***Dominance condition on the generic resource x***
$t, d, ds, wc, ws, de, rr, n$	$x(l) \leq x(m)$
a, ss, bs, sf, sr	$x(l) \geq x(m)$
q	$a(l) > a(m) \vee q(l) \geq q(m)$
tw	$tw(l) \leq tw(m) \vee tw(m) \geq NW$
$\forall\, \bar{N} - t(m) \leq u < t(l),\ w(l,[u,t(l)]) \leq w(m,[u,t(l)])$, where $w(l,[x,y])$ is the accumulated working time in the label l during the interval $[x,y]$.	

The dominance rules described in the four first rows of Table 7 are straightforward and model an independent dominance on each attribute. The last row introduces a dominance rule associated with the night work rule, which is not specific to any resource of label l. This rule involves comparing the working time distribution in the last $\bar{N}$-interval for labels m and l. The total working time in each sub-interval between $[\bar{N} - t, t]$ must be lower for the dominant label. To achieve this, it is necessary to backtrack along each label (using the pointer f to the parent label)

and calculate the total working time for the two labels being compared. This rule ensures that any extension of label l will result in less total working time in its last $\overline{N}$-interval.

In order to improve the performance of the dominance procedure when a new label is generated, this label is not compared against all the non-dominated labels already created. Actually, few labels have a chance of dominating or being dominated by a new label, because of the contradictory dominance conditions of the current finishing time (smaller is better), the amount of work already performed (larger is better), and the break/rest time provisioned (larger is better). In our implementation, only labels with the same values for t are compared.

6.5 Algorithm convergence and complexity

A basic algorithm considers three possible extensions with a duration δ for each label: (i) process, (ii) piece of break/rest and (iii) POA. It is obvious that if all parameters of one instance are multiples of δ, this procedure can generate the optimal schedule. During the procedure, the infeasible schedules may be removed without any risk of missing the optimal solution. Our algorithm improves the basic one by considering some extensions larger than δ without taking any risk to miss the optimal solution. This is guarantee by the set of properties demonstrated on optimal solutions.

The worst-case complexity of our algorithm is upper bounded the basic algorithm that can be easily estimated as follows. Each label is extended by at least δ minutes from the earliest starting time of the earliest activity, E_1, to the latest completion time of the last activity,L_N; according to three possibilities: working, POA or break/rest. The dominance function may require one to parse backward the father labels over the last 24h, i.e., at most $24h./\delta$ labels. If we consider that a label is compared with all the labels for dominance, then the overall complexity of the algorithm is $O\left(3^{\{(L_N - E_1)/\delta\}} \times \overline{N}/\delta\right)$.

7. Computational experiments

We conducted numerical experiments with quantitative and qualitative objectives. First, we evaluated the setting of the parametrized models we provided: the number of nodes per activity for the MILP and the time step for the Label-Setting Algorithm (LS for short). Afterwards, the performance of the two models was compared. The second series of experiments investigated the effect of night rule R6. We present a comparison between the problem-setting proposed and previous approaches for the night working rule. In addition, we explore how the night rule affects the feasibility of the schedules.

We used two original sets of instances in this section. The set referred to as GOEL corresponds to the 157 routes - visiting from four to nineteen customers - obtained from solutions of the Vehicle Routing and Truck Driver Scheduling Problem solved by Goel (2018). We also used the set PGLT of 40 instances, with two to fourteen customers proposed in Peña-Arenas et al. (2021). The creation of this second set was motivated by the application of all the EU rules. Thus, some instances are unfeasible. In all these instances, each customer corresponds to one service activity (other work). One driving activity separates two consecutive customers. The sequence of activities starts and ends with some service activities at the depot.

In GOEL instances, all values are in minutes. We derived three sets of instances with different rounding values, 1 (the original ones), 5 and 15 minutes. The rounding procedure makes the solutions of Goel (2018) feasible, and they are computed as the lower multiple for all values, except the latest service time which is rounded to the greater multiple. In PGLT, all data are multiples of 15 min. except in instance 29. For the sake of clarity, the new set of instances are denoted RX-GOEL or RX-PGLT, where X is the rounding value.

Note that LS with a time step Y is exact for all the instances RX when X is a multiple of Y.

All the experiments were conducted on a single processor of an Intel® Core™i5-8400 at 2.81 GHz under Windows 10, using C++ and Gurobi 8.1.1 and instances and results are available at https://emse.fr/~garaix/TruckDriverSchedule/Expert_Systems/. A limit on the computation time was imposed after 15 minutes.

7.1 MILP performance

Our MILP model is parametrized by the number of nodes per activity. These series of experiments were conducted on R15-GOEL instances, for which LS with time step $\delta = 15min.$ is able to provide optimal solutions.

As tight bounds on the number of nodes per activity are not easy to compute (see Remark 3 from Section 4), we performed experiments using an increasing number of nodes per activity. Three nodes per activity is not enough to solve to optimality the R15-GOEL instances and the running time exceeds 900 seconds for 17 instances among 157. Actually, at least two instances, TDS_C107_1 and TDS_R209_1, require four nodes to reach optimality. Four nodes per activity involve overly long running times and only a few activities need four nodes (see Figure 7). Therefore, we used an additional setting with a tight number of nodes (TN) for each activity. This number is derived from the optimal solution obtained by LS-15. On this set of instances using the TN setting, the minimum, the average, and the maximum number of nodes per instance are 1, 1.9 and 4, respectively. TN gives an approximation of the best possible setting for the MILP. The full set of files with the number of nodes per instance used for the experiments are available at the companion web page.

Activities	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Nodes	1	3	1	1	1	1	1	2	2	3	1	3	2	3	1	3	1	4	1	1

Figure 7

Number of nodes per activity for instance TDS_C107_1.

Table 8 presents the performance of the MILP model on R15-GOEL when the number of nodes per activity is changed from 1 to 3 and the setting TN is used. Columns present the average and maximum running times, the maximum and average GAP, and the number of instances where the algorithm finished and did not find the optimal solution. The statistics are computed over the instances where the maximum running time was not reached.

There is an increase in the running times and the number of optimal solutions obtained by the MILP related to the increment of the number of nodes per activity. Note that using one node per activity is equivalent to working under non-preemption assumptions. Since it is a more constrained model, it is faster, though it is more likely not to find optimal or even feasible solutions. Indeed, this is the fastest setting of the MILP model, with an average running time per instance of 0.4 seconds, but without finding any optimal solutions. With two and three nodes per activity, the average running time increases from 7.1 to 55.7 and the linear model does not find 9 and 2 optimal solutions, respectively. Moreover, the maximum running time is only reached for 17 instances when the linear model uses three nodes per activity. Finally, when a tight number of nodes is used, the model finds all the optimal solutions, with maximal and average running times of 13.5 seconds and 2.6 seconds, respectively.

Of the 17 instances removed from the sample, the setting TN finds all the optimal solutions using a maximum and average running times of 64.3 and 14.5 seconds. Hence, as mentioned above, the setting TN is close to the best performance that our MILP can reach, and it is used to evaluate the performance of the LS algorithm.

Table 8

MILP performance according to the number of nodes per activity on R15-GOEL.

	Running times (s)*		***GAP (min)****		
Number of nodes	***Max.***	***Avg.***	***Max.***	***Avg.***	***# Not optimal****
1	1.5	0.4	9105.0	4669.1	140
2	116.3	7.1	90.0	3.3	9
3	782.4	55.7	30.0	0.3	2
TN	13.5	2.6	0.0	0.0	0

* The statistics are computed over 140 instances where the time limit was not reached.

7.2 Label-Setting performance

The variants of LS algorithms with different values of parameter δ, the time step, are denoted LS-δ. Extensions with idle times have a fixed duration of size δ, so as this size diminishes the number of labels generated with idle/POA will increase in parallel with the running times of the algorithm. Figure 8 shows the running times of LS-15 and LS-5 on instances R15-PGLT and R15-GOEL. The statistics are computed over the instances where the maximum running time was not reached. Note that under this setting, both algorithms are optimal.

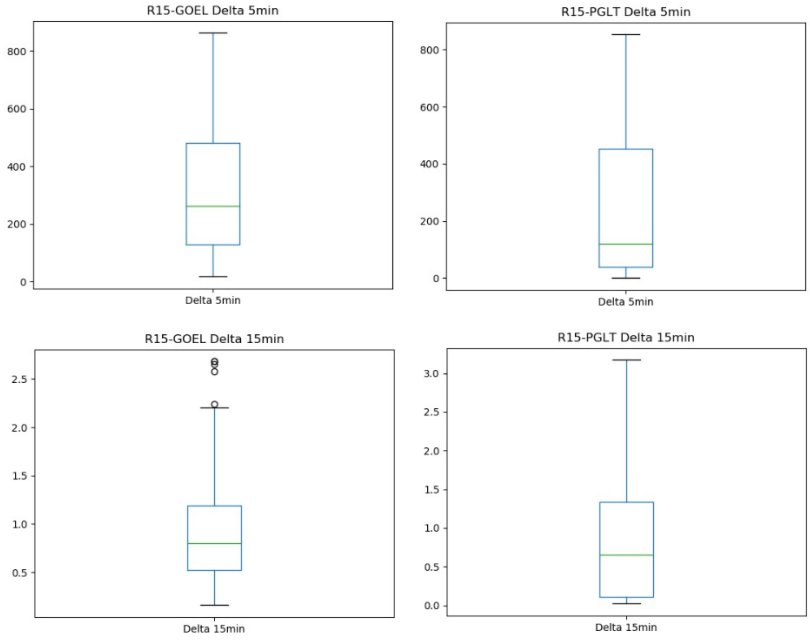

Figure 8

Effect of time step δ on the LS running times using instances R15-PGLT and R15-GOEL.

The statistics are in seconds and computed over 36 instances R15-PGLT and 116 instances R15-GOEL where the time limit was not reached.

On instances R15-PGLT and R15-GOEL, LS-15 solves an instance on average in 0.85 and 0.95 seconds, respectively. In the worst case, the maximal running time is 3.2 seconds on R15-PGLT instances. On the contrary, the LS with $\delta = 5min$ reaches a running time limit greater than 800 seconds for both set of instances. Hence, there is a steep increase in the running times due to a decrease in the time step δ; from the experiments the average running time increases 319 times for a decrease of δ from 15min to 5min on R15-GOEL instances. We anticipated this increase in

running times, as a smaller delta size leads to a greater number of extensions explored between activities. In this sense, the best performance of the LS is attained using $\delta = 15min$.

Next, the size of the rounding parameter on the instances is decreased from 15min. to 5min. to measure its effect on the performance of the algorithm. A massive number of labels is created when using a time step of $\delta = 1min$, so this prohibits the use of the LS under this setting. As a consequence, instances with the rounding parameter set to 1min. are not used because it is not possible to provide optimal solutions, either with the LS or with the MILP model.

Table 9 presents the running times and the GAP of LS-5 and LS-15 using the set of instances rounded to 5 minutes. When using LS-15 on R5-GOEL, the algorithm finishes without finding the optimal solution for 86 instances, however the differences are less than the minimum break duration (15min.), since the maximum and the average GAP are 10min. and 6.3min, respectively. LS-5 brings optimality, although on instances R5-GOEL it reaches the running time limit for 46 instances. In this sense, LS-15 keeps the same running time performance with respect to the results on instances rounded to 15 minutes, but there is a small reduction in the quality of the solutions, because when using this rounding parameter the algorithm cannot find the optimal solution for all the instances.

Table 9

Results LS using δ=15min and δ=5min on the set of instances R5-PGLT and R5-GOEL.

		Running times (s)*		**GAP (min)***		
δ		***Max.***	***Avg.***	***Max.***	***Avg.*** **	***# Not optimal***
15min.	*R5-PGLT*	3.0	0.8	0.0	0.0	0.0
	R5-GOEL	3.0	0.9	10.0	6.3	86
5min.	*R5-PGLT*	897.8	252.1	0.0	0.0	0.0
	R5-GOEL	895.9	342.2	0.0	0.0	0.0

* The statistics are computed over 40 instances R5-PGLT and 111 instances R5-GOEL where the time limit was not reached.

** For the 4 unfeasible R5-PGLT instances (among the 36), the GAP is set to 0.

To measure the effect on the quality of the solutions of the use of rounded instances, the optimal solutions found by LS-5 are compared on instances rounded to 15 and 5 minutes. Table 10 presents the number of instances where the completion time using a rounding parameter of 15 minutes is strictly less than completion time using 5 minutes, and the maximum and average difference between solutions. On PGLT instances there is only one instance with different completion times, and this difference is of 5 minutes, whereas there are 98 GOEL instances with differences due to the rounding parameter, where the maximum and average difference are 570 and 47.9 minutes, respectively. Therefore, on the set of instances evaluated there are at least 99 instances for which rounding to 15min. cannot give a feasible solution for the same instance with a rounding of 5min. In this regard, the rounding parameter affects both the quality of the solutions and the feasibility; the differences in completion times could be of one daily rest (570min.) and most of the instances (approx. $99/144 \cong 69\%$) could become infeasible. The same completion time could not be obtained if the rounding parameter is reduced from 15min. to 5min.

We also compare the performance of LS-15 with the linear model under the setting TN (MILP-TN). From Table 11 both methods find the optimal solutions without reaching the running time limit, though LS-15 is on average almost 4 and 2 times faster than the MILP model on R15-PGLT and R15-GOEL instances, respectively. MILP-TN reaches the maximum running time for one test of R15-PGLT instances, although the average running times are steady with respect to the

type of instance. Moreover, LS-15 increases the average running time by two and the maximum running time roughly by nine ($32.2/3.6 \cong 8.9$), when changing from R15-PGLT to R15-GOEL instances. In this sense LS-15 is sensitive to the type of instance.

Table 10

Comparison completion time of LS-5 on the set of instances PGLT and GOEL rounded to 15 and 5 minutes.

		Difference (min)	
Completion	***Number of instances***	***Max.***	***Avg.***
*R15-PGLT < R5-PGLT**	1	5	0.1
*R15-GOEL < R5-GOEL***	98	570	47.9

* The statistics are computed over 36 instances R15-PGLT and R5-PGLT where the time limit was not reached.
** The statistics are computed over 108 instances R15-GOEL and R5-GOEL where the time limit was not reached.

Table 11

Running times MILP-TN and LS-15 on the set of instances R15-PGLT and R15-GOEL.

	Running times (s)				***Completion times (min)***	
	R15-PGLT*		***R15-GOEL***		***R15-PGLT****	***R15-GOEL***
	Max.	***Avg.***	***Max.***	***Avg.***	***Avg.***	***Avg.***
MILP-TN	42.3	3.9	64.3	3.9	2393.8	6687.6
LS15	3.6	1.0	32.2	2.0	2393.8	6687.6

*The statistics are computed over 39 instances of set PGLT where the time limit was not reached.

7.3 Night working rule effect on the schedules

Two sets of rules, namely Forbidding Night Work (FNW) and EC social legislation (EURULE), have been used previously to incorporate the night working rule in truck driver scheduling. However, EURULE introduces flexibility, allowing night work if the total working time within the last 24-hour interval is less than 10 hours. Both FNW and EURULE have a direct impact on the behavior and performance of the Label Setting (LS) algorithm, specifically the label propagation and dominance rules. To evaluate the effect of the night constraint on schedules and LS performance, the LS algorithm is executed without considering night work (referred to as NO-Night), and the results are compared with FNW and EURULE. The LS algorithm, utilizing a time increment of $\delta = 15min$ minutes, is modified to accommodate these three assumptions, and the comparison is conducted using the set of instances R15-PGLT and R15-GOEL.

In the case of the R15-PGLT instance set, the presence of time windows, customer distances, and the duration of the night interval can render some instances infeasible when the night work constraint is applied. For the experiment conducted on the R15-PGLT instance set, the statistics are calculated based on the feasible instances found by the LS-15 algorithm under the most constrained assumption for the night work, which is FNW. Table 12 provides a summary of the results from this experiment, including the average and maximum deviation from the LS solution with No-Night, the average and maximum running times, and the number of instances that were found to be infeasible.

Under the FNW assumption, the running times of the model are better than EURULE and comparable to NO-Night, though it finds 7 additional infeasible instances with respect to the EURULE assumption, for a total of 14 infeasible instances, and it increases the average completion time by 31.2 (231.9 - 200.8) minutes for other instances. When using the No-Night

assumption the model is still faster than or equal to the others, although the model finds at least 7 infeasible instances with respect to the EU night rule. As a result, 35% of the sequences that do not account for the night working rule become infeasible if night work is forbidden, whilst 17.5% of them are infeasible if the EU night working rule applies.

Table 12

Night effect on the LS performance and the quality of the solutions over R15-PGLT instances.

		*Deviation from NO-Night (min)**		*Running times (s)**		*Infeasible***	
	Avg. Completion (min)*	*Avg.*	*Max.*	*Avg.*	*Max.*	*Count*	*%*
NO-Night	2989.6	0.0	0.0	0.1	0.3	0	0.0
FNW	3221.5	231.9	555.0	0.1	0.4	14	35.0
EURULE	3190.4	200.8	555.0	1.7	8.3	7	17.5

*The statistics are computed over 26 feasible instances for the three night assumptions.
**The statistics are computed over the 40 instances of set PGLT.

Instances R15-GOEL from the literature, are feasible if the night work constraint applies. Table 13 presents the results for this experiment on instances R15-GOEL, using the same set of statistics of Table 12. Although, for this set of instances, the column infeasible shows how many solutions of the LS-NO-Night are not possible to achieve if the night working rule (FNW or EURULE) applies.

Table 13

Night effect on the LS performance and the quality of the solutions over R15-GOEL instances.

		Deviation from NO-Night (min)		*Running times (s)*		*Infeasible*	
	Avg. Completion (min)	*Avg.*	*Max.*	*Avg.*	*Max.*	*Count*	*%*
NO-Night	6563.2	0.0	0.0	0.1	0.3	0	0
FNW	6720.6	157.4	540.0	0.2	2.0	105	70
EURULE	6687.6	124.4	540.0	2.1	33.4	73	50

Regarding completion and running times, the results align with those observed in the R15-PGLT instances. Under EURULE, there is an improvement of 32.9 minutes (157.4 - 124.44), while the fastest version of the LS algorithm remains the one without considering night work (NO-Night). However, there is a significant rise in the number of infeasible solutions, reaching 70% and 50% for the FNW and EURULE rules, respectively.

The results demonstrate that forbidding night work accelerates the solution process compared to EURULE, resulting in similar running times to not considering the night at all. There are three key reasons for this outcome. Firstly, there is a reduction in the number of labels, since the set of possible extensions is reduced to rests or POA once a night period is reached. Secondly, there is no need to verify the total working time during the last 24 hours, which is a computationally intensive task. Lastly, the process of applying dominance rules becomes simpler as there is no

requirement to compare the distribution of working time during the last 24 hours when comparing two labels.

Finally, schedules without considering the night working rule are probably infeasible w.r.t the night provisions. Hence, dispatchers should modify the solutions by hand, increasing the costs of the schedules, either by avoiding the night work or by increasing additional time compensations for the driver due to night work. Finally, including the flexibility of the EURULE can bring cost savings by reducing the schedule durations, although there is an increase in the computational times due to the complexity added to the scheduling process.

8. Conclusion

Several previous contributions to the literature have worked on the TDSP, but most of them have neglected or simplified the night working rule. In this paper, we present a MILP model and a label-setting algorithm to solve the TDSP which minimizes the completion time of a given sequence of customers, while including all the rules in EC social legislation, for a weekly planning period including the night working constraint. The LS is compared against the MILP model. Both solution methods bring optimal solutions under a predefined setting, but the LS requires significantly lower computational times. The size of the rounding parameter affects the quality of the solutions, leading to considerably differences which, in half of the cases, result in infeasibility, for example when changing from 15min. to 5min.; however, LS-15 performs well under both rounding sizes 15min. and 5min., with only a slight increase of 6.3 minutes on the average completion time of the schedules when the rounding parameter is reduced.

The results show that forbidding or neglecting night work drastically diminishes the running times of the algorithm, although the EU night working rule yields schedules with equal or better completion times than the prohibition of night work. In addition, regarding the experiment's results, solution methods that do not take account of the night working rule are likely to find infeasible solutions. Finally, the LS exhibits a good performance in terms of both the quality of the solution and running times. This makes it a valuable tool for dispatchers to verify routes and schedules that adhere to the established weekly rules of the EC social legislation.

Future research related with the algorithms presented in this paper should consider, the solution of the VRTDSP, and the pick-up and delivery problem with time windows and driver's rules. This involves enhancing the scheduling algorithm's performance in terms of running times and investigating its integration into exact or heuristic methods as part of an integrated solution strategy. Furthermore, there is still a challenging open question about the complexity of the TDSP, and how it is affected by the inclusion of the night working rule.

References

Brandao, J., & Mercer, A. (1997). A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. *European Journal of Operational Research, 100*(1), 180-191. https://doi.org/10.1016/S0377-2217(97)00010-6.

Cordieri, S.A., Fumero, F., Jabali, O., Malucelli, F. (2022). The Long-Haul Transportation Problem with Refueling Deviations and Time-Dependent Travel Time. In: de Armas, J., Ramalhinho, H., Voß, S. (Eds). *Computational Logistics. ICCL 2022. Lecture Notes in Computer Science, 13557*. Springer, Cham. https://doi.org/10.1007/978-3-031-16579-5_17.

Drexl, M., & Prescott-Gagnon, E. (2010). Labelling algorithms for the elementary shortest path problem with resource constraints considering EU drivers' rules. *Logistics Research, 2*, 79–96. https://doi.org/10.1007/s12159-010-0022-9.

Goel, A. (2009). Vehicle scheduling and routing with drivers' working hours. *Transportation Science, 43*(1), 17–26. https://doi.org/10.1287/trsc.1070.0226.

Goel, A. (2010). Truck driver scheduling in the European Union. *Transportation Science, 44*(4), 429–441. https://doi.org/10.1287/trsc.1100.0330.

Goel, A. (2012a). The minimum duration truck driver scheduling problem. EURO *Journal on Transportation and Logistics, 1*, 285–306. https://doi.org/10.1007/s13676-012-0014-9.

Goel, A. (2012b). Truck driving scheduling in Australia. *Computers & Operations Research, 39*, 1122–1132. https://doi.org/10.1016/j.cor.2011.05.021.

Goel, A. (2012c). A mixed integer programming formulation and effective cuts for minimising schedule durations of Australian truck drivers. *Journal of Scheduling, 15*, 733–741. https://doi.org/10.1007/s10951-012-0282-0.

Goel, A. and Irnich, S. (2016). An exact method for vehicle routing and truck driver scheduling problems. *Transportation Science, 51(2)*, 737–754. https://doi.org/10.1287/trsc.2016.0678.

Goel, A. (2018). Legal aspects in road transport optimization in Europe. *Transportation Research Part E: Logistics and Transportation Review, 114*, 144–162. https://doi.org/10.1016/j.tre.2018.02.011.

Goel, A., & Vidal, T. (2013). Hours of Service Regulations in Road Freight Transport: An Optimization-Based International Assessment. *Transportation Science, 48*(3), 391-412. https://doi.org/10.1287/trsc.2013.0477.

Kok, A. L., Meyer, C. M., Kopfer, H, & Schutten, J. M. J. (2010). A Dynamic Programming Heuristic for the Vehicle Routing Problem with Time Windows and European Community Social Legislation. *Transportation Science, 44*(4), 442–454. https://doi.org/10.1287/trsc.1100.0331.

Kok, A.L., Hans, E.W., & Schutten, J.M.J. (2011). Optimizing departure times in vehicle routes. *Journal of Operational Research, 210*(3), 579-587. https://doi.org/10.1016/j.ejor.2010.10.017.

Mayerle, S., De Genaro, D., Neiva, J. & Ferreira H. (2020). The long-haul full-load vehicle routing and truck driver scheduling problem with intermediate stops: An economic impact evaluation of Brazilian policy. *Transportation Research Part A: Policy and Practice, 140*, pp. 36-51. https://doi.org/10.1016/j.tra.2020.07.021.

Mor, A., Archetti, C., Jabali, O., Simonetto, A. & Speranza, G. (2022). The Bi-objective Long-haul Transportation Problem on a Road Network. *Omega, 106, https://doi.org/10.1016/j.omega.2021.102522*.

Peña-Arenas. I., Garaix, T., Lacomme, P., & Tchernev, N. (2021). A mixed integer programming formulation for the truck drivers scheduling problem considering the European Union drivers rules. *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, 101-106, doi: 10.1109/CASE49439.2021.9551634.

Prescott-Gagnon, E., Desaulniers, G., Drexl, M., & Rousseau, L.M. (2010). European Driver Rules in Vehicle Routing with Time Windows. *Transportation Science, 44*(4), 455-473. https://doi.org/10.1287/trsc.1100.0328.

Rochat, Y., & Semet, F. (1994). A tabu search approach for delivering pet food and flour in Switzerland. *The Journal of the Operational Research Society, 45*(11), 1233-1246. https://doi.org/10.2307/2583852.

Sartori, C., Smet, P., & Vanden-Berghe, G. (2022). Scheduling truck drivers with interdependent routes under European Union regulations. *European Journal of Operational Research, 298*(1), 76–88. https://doi.org/10.1016/j.ejor.2021.06.019.

Savelsbergh, M., & Sol, M. (1998). Drive: dynamic routing of independent vehicles. *Operations Research, 46*(4), 474-490. https://doi.org/10.1287/opre.46.4.474.

Tilk, C., & Goel, A. (2020). "Bidirectional labeling for solving vehicle routing and truck driver scheduling problems", *European Journal of Operational Research, 283*(1), 108–124. https://doi.org/10.1016/j.ejor.2019.10.038.

Vital, F. & Ioannou, P. (2021). Scheduling and shortest path for trucks with working hours and parking availability constraints. *Transportation Research Part B: Methodological, 148*, 1-37. https://doi.org/10.1016/j.trb.2021.04.002.

Xu, H, Chen, Z., Rajagopal, S., & Arunapuram, S. (2003). Solving a Practical Pickup and Delivery Problem. *Transportation Science, 37*(3), 347–364. https://doi.org/10.1287/trsc.37.3.347.16044.