Truck Driver Scheduling in the United States

Asvin Goel[1,2] and Leendert Kok[3]

[1] MIT-Zaragoza International Logistics Program

Zaragoza Logistics Center, Spain

asvin@mit.edu

[2] Applied Telematics/e-Business Group,

Department of Computer Science, University of Leipzig, Germany

asvin.goel@uni-leipzig.de

[3] ORTEC, Algorithmic R & D

P.O. Box 490, 2800 AL Gouda, The Netherlands

leendert.kok@ortec.com

October 25, 2010

Abstract

The U.S. Truck Driver Scheduling Problem (US-TDSP) is the problem of visiting a sequence of λ locations within given time windows in such a way that driving and working activities of truck drivers comply with U.S. hours of service regulations. In the case of single time windows it is known that the US-TDSP can be solved in $O(\lambda^3)$ time. In this paper we present a scheduling method for the US-TDSP which solves the single time window problem in $O(\lambda^2)$ time. Furthermore, we show that in the case of multiple time windows the same complexity can be achieved if the gap between subsequent time windows is at least 10 hours. This situation occurs, for example, if, because of opening hours of docks, handling operations can only be performed between 8.00 AM and 10.00 PM. Furthermore, we empirically show that for a wide range of other problem instances the computational effort is not much higher if multiple time windows are considered.

Keywords: Vehicle Scheduling, U.S. Hours of Service Regulations, Multiple Time Windows

1 Introduction

The Federal Motor Carrier Safety Agency has estimated that truck driver fatigue is a factor in 15 percent of large truck crashes. In November 2008, the Federal Motor Carrier Safety Agency adopted current hours of service regulations for truck drivers in the United States, see Federal Motor Carrier Safety Administration (2008). These regulations are merely identical with those adopted in 2003 and 2005 which were both overturned by the U.S. Court of Appeals for the D.C. Circuit. According to a survey conducted by McCartt et al. (2008), one out of six truck drivers admits to having dozed at wheel in the month prior to the survey. This value has significantly increased since the 2003 rule came into effect. The same survey revealed that less than one out of two truck drivers reported that delivery schedules are always realistic. Truck drivers who reported that they are sometimes or often given unrealistic delivery schedules are approximately three times as likely to violate the work rules as drivers who rarely or never have to deal with unrealistic delivery schedules. In March 2009 a lawsuit was filed again challenging the new regulations because of safety concerns and the fear that the regulations promote driver fatigue, see Stone et al. (2009). In a settlement agreement, the Federal Motor Carrier Safety Agency announced that it will reconsider and potentially change the regulation. The current regulation, however, will remain in effect during the rule making process.

One of the first research works explicitly considering break periods within vehicle routing and scheduling is presented by Savelsbergh and Sol (1998) who consider a problem in which lunch breaks and night breaks must be taken within fixed time intervals. Hours of service rules imposed by the U.S. Department of Transportation are first studied by Xu et al. (2003) who present a column generation approach for combined vehicle routing and scheduling. They conjecture that determining a minimal cost truck driver schedule for a given sequence of customer locations is NP-hard in the presence of multiple time windows. Archetti and Savelsbergh (2009) study a similar problem with single time windows, i.e. where each location has exactly one time window, and show that truck driver schedules complying with U.S. hours of service regulations for a sequence of λ locations to be visited can be found in $O(\lambda^3)$ time. Furthermore, they conclude that the conjecture made by Xu et al. (2003) may not be true. The approach presented by Archetti and Savelsbergh (2009) could be used to determine truck driver schedules in the presence of multiple time windows by solving a single time window problem for each combination of time windows. However, if each of the λ locations has τ time windows there are τ^λ combinations of time windows. Thus, knowing that truck driver schedules can be determined in polynomial time in the single time window case does not give any indication whether, in the presence of multiple time windows, truck driver schedules can be determined in polynomial time.

Recently, several works considering the generation of truck driver schedules complying with European Union regulations have been presented. However, none of these papers considers multiple time windows. Goel (2009), Kok et al. (2010), and Prescott-Gagnon et al. (2010) solve combined vehicle routing and truck driver scheduling problems in the European Union by heuristically determining truck driver schedules. Goel (2010) presents the first method that is guaranteed to find a truck driver schedule complying with European Union regulation if such a schedule exists. European Union regulations are more complex than U.S. hours of service regulation, because they require that in addition to rest periods, in which drivers can sleep, shorter breaks for recuperation must be scheduled after four and a half hours of driving. The provisions of the regulation concerning rest periods, however, share some similarity with U.S. hours of service regulations.

In this paper we study the U.S. Truck Driver Scheduling Problem with multiple time windows, which is the problem of visiting a sequence of λ locations within given time windows in such a way that driving and working activities of truck drivers comply with U.S. hours of service regulations. We provide a formal model and analyse some structural properties of the problem. We present a scheduling method which is guaranteed to find feasible truck driver schedules if such schedules exists. We show that, in the case of single time windows, this method can solve the U.S. Truck Driver Scheduling Problem in $O(\lambda^2)$ time. Furthermore, we show that the case of multiple time windows is not harder to solve if the gap between these time windows is at least 10 hours. This situation occurs, for example, if, because of opening hours of docks, handling operations can only be performed between 8.00 AM and 10.00 PM. Furthermore, we empirically show that for a wide range of problem instances which do not satisfy this property, the computational effort required by our scheduling algorithm does not increase significantly.

The remainder of this paper is organised as follows. Section 2 describes current hours of service regulations imposed by the U.S. Department of Transportation. In Section 3, a formal model of the U.S. Truck Driver Scheduling Problem (US-TDSP) is given. Section 4 introduces conditions for pseudo-feasibility which relax the conditions for feasibility presented in Section 3. This relaxation allows us to myopically construct truck driver schedules without considering future driver activities and constraints imposed on them. Section 4 furthermore gives dominance criteria, which help us in reducing the number of partial schedules that must be explored in order to solve the US-TDSP. Normality conditions are presented which guide us when solving the US-TDSP. Section 5 presents a scheduling method which solves the US-TDSP by constructing pseudo-feasible schedules satisfying these normality conditions. In Section 6, we analyse the performance of the method. Finally, Section 7 concludes this paper.

2 U.S. Hours of Service Regulations

Present hours of service regulations imposed by the U.S. Department of Transportation are comprehensively described by Federal Motor Carrier Safety Administration (2009). The regulation distinguishes between *on-duty time* and *off-duty time*. On-duty time refers to all time a driver is working and includes driving activities as well as other work such as loading and unloading. Off-duty time refers to any time during which a driver is not performing any work.

The regulation limits the maximum amount of accumulated driving time to 11 hours. After accumulating 11 hours of driving, the driver must be off duty for 10 consecutive hours before driving again. In the remainder of this paper we will denote a period of at least 10 consecutive hours of off-duty time as a *rest* period. Thus, a driver must not drive for more than 11 hours in between two rest periods.

The regulation prohibits a driver from driving after 14 hours have elapsed since the end of the last rest period. However, a driver may conduct other work after 14 hours have elapsed since the end of the last rest period.

Further regulations prohibit driving after a driver has accumulated 60 or 70 hours of on-duty time within a period of 7 or 8 days. Furthermore, a driver may restart counting duty times after taking 34 or more consecutive hours off duty. Like Archetti and Savelsbergh (2009), we will assume in the remainder of this paper that no more than 60 or 70 hours of on-duty time are assigned to a driver within the planning horizon. Furthermore, we assume that, at the beginning of the planning horizon, the driver returns from a rest period which is long enough such that previous driving and working activities do not have any influence on driving and working hours within the planning horizon.

3 The Truck Driver Scheduling Problem

In this section we describe the U.S. Truck Driver Scheduling Problem with multiple time windows. Let us consider a sequence of locations denoted by $n_1, n_2, \ldots, n_\lambda$ which shall be visited by a truck driver. At each location n_μ some stationary work of duration w_μ shall be conducted. This work shall begin within one of multiple disjunct time windows. The number of time windows at location n_μ shall be denoted by τ_μ and the time windows by $T_\mu^1, T_\mu^2, \ldots, T_\mu^{\tau_\mu}$. Let $T_\mu := T_\mu^1 \cup T_\mu^2 \cup \ldots \cup T_\mu^{\tau_\mu}$ denote the set of all feasible start times at location n_μ. The (positive) driving time required for moving from node n_μ to node $n_{\mu+1}$ shall be denoted by $\delta_{\mu,\mu+1}$. The *U.S. Truck Driver Scheduling Problem* is the problem of determining whether driving and working hours of a truck driver can be scheduled in such a way that all work activities begin within one of the corresponding time windows and that U.S. hours of service regulations are complied with.

In order to give a formal model of the problem, let us denote with `DRIVE` any period during which the driver is driving, with `WORK` any on-duty time in which the driver is not driving, with `REST` any period of 10 consecutive hours or more of off-duty time, and with `IDLE` any other off-duty time.

A truck driver schedule can be specified by a sequence of activities to be performed by the driver. Let $\mathcal{A} := \big\{a = (a^{\text{type}}, a^{\text{length}}) \mid a^{\text{type}} \in \{\texttt{DRIVE}, \texttt{WORK}, \texttt{REST}, \texttt{IDLE}\}, a^{\text{length}} > 0\big\}$ denote the set of driver activities to be scheduled. Let « . » be an operator which concatenates different activities. Thus, $a_1.a_2.\ \ldots\ .a_k$ denotes a schedule in which for each $i \in \{1, 2, \ldots, k-1\}$ activity a_{i+1} is performed immediately after activity a_i. For a given schedule $s := a_1.a_2.\ \ldots\ .a_k$ and $1 \leq i \leq k$ let $s_{1,i} := a_1.a_2.\ \ldots\ .a_i$ denote the partial schedule composed of activities a_1 to a_i. Recall that we assume that at the beginning of the planning horizon, the driver returns from a rest period which is long enough such that previous driving and working activities do not have any influence on the driving and working hours within the planning horizon. We will thus only consider schedules $s := a_1.a_2.\ \ldots\ .a_k$ which begin with a rest period of at least 34 hours, i.e. $a_1^{\text{type}} = \texttt{REST}$ and $a_1^{\text{length}} \geq 34$.

Notation	Value	Description
t^{rest}	10 hours	The minimum duration of a rest period
t^{drive}	11 hours	The maximum accumulated driving time between two consecutive rest periods
t^{elapsed}	14 hours	The maximum time since the end of the last rest period until which a driver may drive

Table 1: Parameters imposed by the regulation

Table 1 gives an overview of the parameters imposed by the regulation. We use the following notation for determining whether a schedule complies with the regulation. For each schedule $s := a_1.a_2. \ \ldots \ .a_k$ with $a_1^{\text{type}} = \texttt{REST}$ we denote the completion time of the schedule by l_s^{end}, the time of completion of the last rest period by $l_s^{\text{last\_rest}}$, and the accumulated driving time since completion of the last rest period by l_s^{drive}. These values can be recursively computed during schedule generation by

$$
\begin{aligned}
l_{s_{1,1}}^{\text{end}} &:= a_1^{\text{length}} \quad \text{and} \quad l_{s.a}^{\text{end}} := l_s^{\text{end}} + a^{\text{length}},\\
l_{s_{1,1}}^{\text{last\_rest}} &:= l_{s_{1,1}}^{\text{end}} \quad \text{and} \quad l_{s.a}^{\text{last\_rest}} := \begin{cases} l_{s.a}^{\text{end}} & \text{if } a^{\text{type}} = \texttt{REST}\\ l_s^{\text{last\_rest}} & \text{otherwise,} \end{cases}\\
l_{s_{1,1}}^{\text{drive}} &:= 0 \quad \text{and} \quad l_{s.a}^{\text{drive}} := \begin{cases} 0 & \text{if } a^{\text{type}} = \texttt{REST}\\ l_s^{\text{drive}} + a^{\text{length}} & \text{if } a^{\text{type}} = \texttt{DRIVE}\\ l_s^{\text{drive}} & \text{otherwise.} \end{cases}
\end{aligned}
$$

For a given sequence of locations $n_1, n_2, \ldots, n_\lambda$ and a schedule $s = a_1.a_2. \ \ldots \ .a_k$, let us denote with $i(\mu)$ the index in s corresponding to the μth stationary work period, i.e. $a_{i(\mu)}$ corresponds to the work performed at location n_μ. Analogously, for each $1 < i \leq k$ with $a_i^{\text{type}} = \texttt{WORK}$ let us denote with $\mu(i)$ the index of the respective location in $n_1, n_2, \ldots, n_\lambda$. With this notation we can now give a formal model of the problem.

The U.S. Truck Driver Scheduling Problem (US-TDSP) with multiple time windows is the problem of determining for a given sequence of locations $n_1, n_2, \ldots, n_\lambda$, whether a schedule $s := a_1.a_2. \ \ldots \ .a_k$ with $a_1^{\text{type}} = \texttt{REST}$ and $a_1^{\text{length}} \geq 34$ exists which satisfies

$$\sum_{\substack{i(1) \leq j \leq i(\lambda) \\ a_j^{\text{type}} = \texttt{DRIVE}}} a_j^{\text{length}} = \sum_{\substack{1 \leq j \leq k \\ a_j^{\text{type}} = \texttt{DRIVE}}} a_j^{\text{length}} \tag{1}$$

$$a_{i(\mu)}^{\text{length}} = w_\mu \text{ for each } \mu \in \{1, 2, \ldots, \lambda\} \tag{2}$$

$$l_{s_{1,i(\mu)-1}}^{\text{end}} \in T_\mu \text{ for each } \mu \in \{1, 2, \ldots, \lambda\} \tag{3}$$

$$\sum_{\substack{i(\mu) \leq j \leq i(\mu+1) \\ a_j^{\text{type}} = \texttt{DRIVE}}} a_j^{\text{length}} = \delta_{\mu,\mu+1} \text{ for each } \mu \in \{1, 2, \ldots, \lambda - 1\} \tag{4}$$

$$l_{s_{1,i}}^{\text{drive}} \leq t^{\text{drive}} \text{ for each } 1 < i \leq k \tag{5}$$

$$a_i^{\text{length}} \geq t^{\text{rest}} \text{ for each } 1 < i \leq k \text{ with } a_i^{\text{type}} = \texttt{REST} \tag{6}$$

$$l_{s_{1,i}}^{\text{end}} \leq l_{s_{1,i}}^{\text{last\_rest}} + t^{\text{elapsed}} \text{ for each } 1 < i \leq k \text{ with } a_i^{\text{type}} = \texttt{DRIVE} \tag{7}$$

Condition (1) demands that all driving is conducted between the first and the last work activity. Condition (2) demands that the duration of the μth work activity matches the specified work duration at location n_μ. Condition (3) demands that each work activity begins within one of the corresponding time windows. Condition (4) demands that the accumulated driving time between two work activities matches the driving time required to move from one location to the other. Condition (5) demands that the maximum amount of driving between two rest periods does not exceed the limit given by the regulation. Condition (6) demands that each rest period has the minimum duration required by the regulation. Condition (7) demands that no driving is conducted after 14 hours have elapsed since returning from the last rest period. In the remainder of this paper, we will say that a schedule $s := a_1.a_2. \ \ldots \ .a_k$ with $a_1^{\text{type}} = \texttt{REST}$ and $a_1^{\text{length}} \geq 34$ is *feasible* if and only if it satisfies conditions (1) to (7).

Figure 1 illustrates five different feasible schedules for the truck driver scheduling problem given by $\lambda = 5$ and $T_\mu = [t_\mu^{\text{min}}, t_\mu^{\text{max}}]$, $\delta_{\mu,\mu+1} = 1$, and $w_\mu = \frac{1}{2}$ for all $1 \leq \mu \leq 5$. Each of these schedules has different characteristics and different values for l_s^{end}, $l_s^{\text{last\_rest}}$, and l_s^{drive}. Obviously, many other feasible schedules exist for the truck driver scheduling problem. In order to efficiently solve the truck

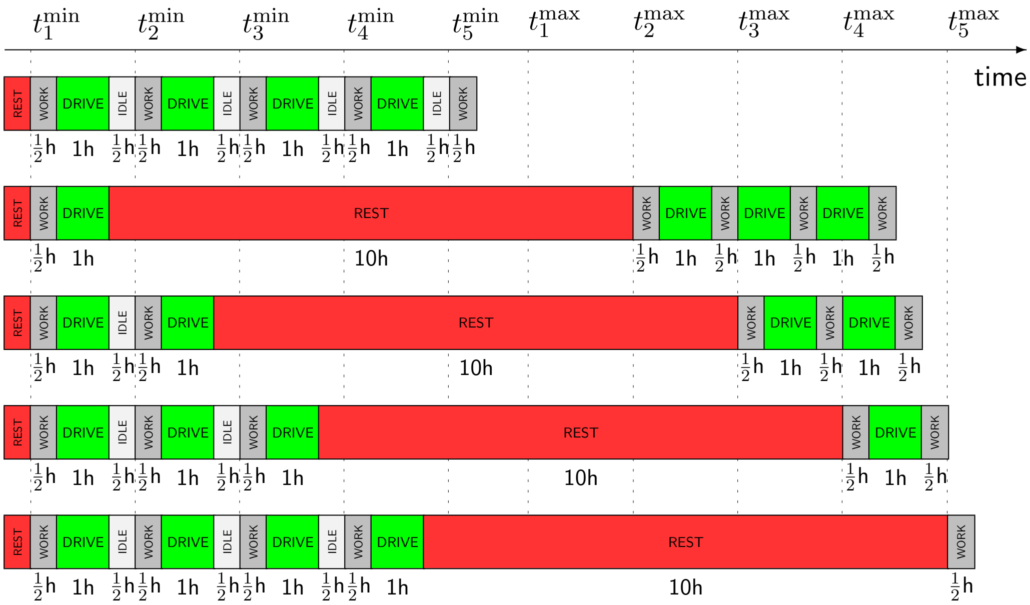

Figure 1: Five feasible schedules for a truck driver scheduling problem with five locations

driver scheduling problem, we thus need to identify problem characteristics which help us reducing the search space as much as possible.

4 Pseudo-feasibility and Normality

For single time window truck driver scheduling problems considering European Union regulations, Goel (2010) defined criteria for pseudo-feasibility and a normal form which provides guidance when solving the problem. In this section, we will adopt these criteria and develop a normal form for truck driver schedules for the U.S. Truck Driver Scheduling Problem. Furtermore, we generalise the concepts of pseudo-feasibility and the normal form to the case of multiple time windows.

In order to be able to reduce the search space to schedules in which each rest period has a duration of minimal length, we replace constraint (7) of the US-TDSP by a relaxed constraint (7'). The relaxed constraint will guarantee that compliance with constraint (7) can be achieved in a post processing step. To formulate the relaxed constraint, we have to determine the amount of time by which each rest period can be extended without violating any other constraint of the truck driver scheduling problem. For any schedule s let us denote the accumulated slack time since completion of the last rest period by

l_s^{slack}. This value constrains the amount by which the duration of the last rest period can be extended without increasing the completion time of the schedule. It can be recursively computed by

$$l_{s_{1,1}}^{\text{slack}} := 0 \text{ and } l_{s.a}^{\text{slack}} := \begin{cases} 0 & \text{if } a^{\text{type}} = \texttt{REST} \\ l_s^{\text{slack}} + a^{\text{length}} & \text{if } a^{\text{type}} = \texttt{IDLE} \\ l_s^{\text{slack}} & \text{otherwise.} \end{cases}$$

By extending the duration of the last rest period in the schedule, the start times of subsequent work activities may be pushed to a later point in time. Thus, we need to know by how much the duration of the last rest period can be increased without violating time window constraints. For the European Union Truck Driver Scheduling Problem with single time windows, Goel (2010) determined the maximum value by which the rest period can be increased without violating time window constraints. Any extension by smaller amount maintains compliance with time window constraints and any extension by a larger amount results in a violation of time window constraints. Furthermore, any extension of the last rest period increases the start times of subsequent work activities by at most the amount of the extension. For the problem with multiple time windows studied in this paper, this is no longer the case. Figure 2, which will be discussed in more detail later, illustrates an example in which we can extend the last rest period in the last schedule by up to 3 hours. However, if we extend the rest period by only 1 hour and 1 minute, the start time of the first work activity must be increased by 2 hours and the start time of the second work activity must be increased by 3 hours. Calculating and updating the maximum amount by which the last rest period can be extended can be a tedious task in the presence of multiple time windows, because it may require to examine the impact of an extension on the start times of all subsequent work activities. A value that can be easily calculated and updated during schedule construction is the maximum amount by which the last rest period can be extended without pushing the start time of any subsequent work activity out of its current time window. Let us denote this value by l_s^{push}. For any schedule $s := a_1.a_2. \ \ldots \ .a_k$ with $a_k^{\text{type}} = \texttt{WORK}$ let us denote with u_s the closing time of the corresponding time window during which the work activity a_k starts, i.e. $u_s := \max T_{\mu(k)}^j$ where j is the index with $l_{s_{1,k-1}}^{\text{end}} \in T_{\mu(k)}^j$. For any schedule $s := a_1.a_2. \ \ldots \ .a_k$ with $a_k^{\text{type}} = \texttt{WORK}$, the last rest period in s can be increased by $l_{s_{1,k-1}}^{\text{slack}}$ without increasing the start time of activity a_k because the amount of idle time preceding the activity can be reduced by $l_{s_{1,k-1}}^{\text{slack}}$.

We can further extend the extend the last rest period by $u_s - l^{\text{end}}_{s_{1,k-1}}$ without pushing the start time of a_k out of its current time window. The value of l^{push}_s can thus be recursively computed by

$$l^{\text{push}}_{s_{1,1}} := \infty \text{ and } l^{\text{push}}_{s.a} := \begin{cases} \infty & \text{if } a^{\text{type}} = \texttt{REST} \\ \min\{l^{\text{push}}_s, l^{\text{slack}}_s + u_{s.a} - l^{\text{end}}_s\} & \text{if } a^{\text{type}} = \texttt{WORK} \\ l^{\text{push}}_s & \text{otherwise.} \end{cases}$$

The duration of the last rest period in schedule s may be extended by

$$l^{\text{extend}}_s := \min\{l^{\text{slack}}_s, l^{\text{push}}_s\}$$

without violating time window constraints or increasing the completion time. Now, let us replace condition (7) of the US-TDSP by condition

$$l^{\text{end}}_{s_{1,i}} \leq l^{\text{last\_rest}}_{s_{1,i}} + l^{\text{extend}}_{s_{1,i}} + t^{\text{elapsed}} \text{ for each } 1 < i \leq k \text{ with } a^{\text{type}}_i = \texttt{DRIVE} \tag{7'}$$

and let us say that a schedule $s := a_1.a_2. \ \ldots \ .a_k$ with $a^{\text{type}}_1 = \texttt{REST}$ and $a^{\text{length}}_1 \geq 34$ is *pseudo-feasible* if and only if it satisfies conditions (1) to (6) and (7').

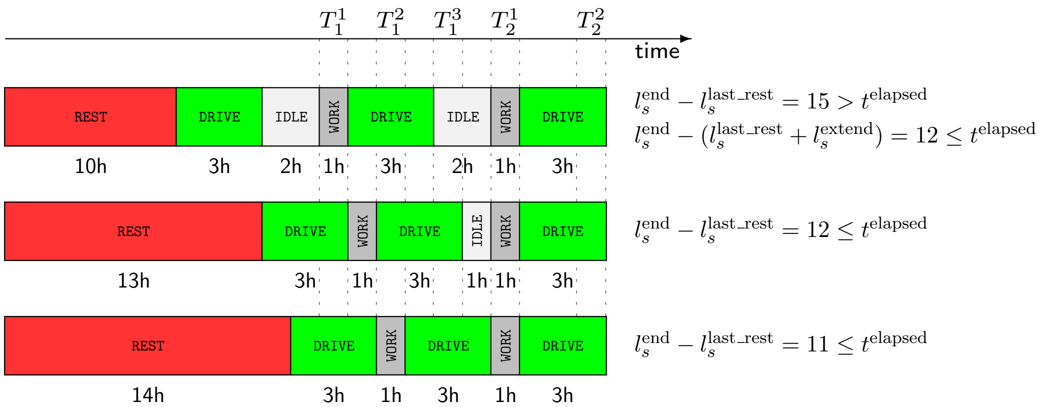

Figure 2: Feasible and pseudo-feasible schedules

Figure 2 illustrates the idea behind replacing condition (7) by condition (7'). The first schedule in Figure 2 is infeasible because the last driving activity finishes 15 hours after the end of the last rest period, and thus condition (7) is violated. As $l^{\text{slack}}_s = 4$ and $l^{\text{push}}_s = 3$ we know that we can increase the last rest period in the schedule by $l^{\text{extend}}_s = 3$. The schedule is pseudo-feasible because condition (7') is satisfied. The second schedule in Figure 2 is a feasible schedule obtained by increasing the duration of the last rest period in the first schedule by $l_s^{\text{extend}} = 3$ (and reducing the duration of subsequent idle periods). By increasing the duration of the last rest period, the arrival time at the first location is increased to the end of time window T_1^1. Note, that we can also extend the duration of the last rest period by a smaller value than $l_s^{\text{extend}} = 3$ to achieve feasibility. The increase, however, must be at least $l_s^{\text{end}} - l_s^{\text{last\_rest}} - t^{\text{elapsed}} = 1$ to achieve compliance with condition (7). The third schedule in Figure 2 is a feasible schedule that can be obtained by increasing the duration of the last rest period by 4 hours, i.e. a value larger than $l_s^{\text{extend}} = 3$. In this schedule the work activity at the first location can no longer start within time window T_1^1. By only considering extensions of a rest period which do not exceed l_s^{extend} we make sure that no extension pushes the start time of some work activity from one time window to the next.

As $l_s^{\text{extend}} \geq 0$ for any feasible schedule s we know that each feasible schedule is pseudo-feasible. As we can transform every pseudo-feasible schedule into a feasible schedule by extending the duration of rest periods, the U.S. Truck Driver Scheduling Problem (US-TDSP) is equivalent to the problem of determining whether a pseudo-feasible schedule $s := a_1.a_2. \ \ldots \ .a_k$ with $a_1^{\text{type}} = \texttt{REST}$ and $a_1^{\text{length}} \geq 34$ exists.

The advantage of searching for pseudo-feasible schedules is that we do not need to know the best duration of rest periods during schedule generation. Instead, we can start by scheduling rest periods which are as short as possible. In a post-processing step, the duration of these rest periods can be extended to the required length.

In general, the set of all pseudo-feasible schedules is too large to be enumerated. Let us now define criteria indicating which schedules can safely be ignored when solving the US-TDSP.

Definition A schedule s' *dominates* another schedule s'' if some schedule $\tilde{s}'$ exists such that $s'.\tilde{s}'.\tilde{s}''$ is pseudo-feasible for all $\tilde{s}''$ for which $s''.\tilde{s}''$ is pseudo-feasible.

Thus, if the schedule s'' is dominated by schedule s' and a pseudo-feasible schedule for tour $n_1, n_2, \ldots, n_\lambda$ exists which begins with the activities in schedule s'', then there also exists a pseudo-feasible schedule for tour $n_1, n_2, \ldots, n_\lambda$ which begins with the activities in schedule s'. Therefore, s'' is not required for solving the US-TDSP. Criteria for dominance are given in the following two lemmata.

Lemma 1 Let s', s'' be pseudo-feasible schedules for the partial tour $n_1, \ldots, n_\mu$ with $1 \leq \mu < \lambda$. Schedule s' dominates s'' if

$$l^{\text{end}}_{s'} \leq l^{\text{end}}_{s''} \text{ and } l^{\text{drive}}_{s'} \leq l^{\text{drive}}_{s''} \text{ and}$$

$$l^{\text{last\_rest}}_{s'} + l^{\text{slack}}_{s'} \geq l^{\text{last\_rest}}_{s''} + l^{\text{slack}}_{s''} \text{ and } l^{\text{last\_rest}}_{s'} + l^{\text{push}}_{s'} \geq l^{\text{last\_rest}}_{s''} + l^{\text{push}}_{s''}.$$

Proof. If $l^{\text{end}}_{s'} = l^{\text{end}}_{s''}$, any activity a that can be appended to the schedule s'' without violating constraints (1) to (6) and (7') can be appended to the schedule s' without violating constraints (1) to (6) and (7'). After appending a, the conditions of the lemma equally hold for $s'.a$ and $s''.a$. Thus, any sequence of activities that can be appended to s'' can be appended to s' without violating the conditions for pseudo-feasibility. If $l^{\text{end}}_{s'} < l^{\text{end}}_{s''}$ we can set $\tilde{s}' := (\texttt{IDLE}, l^{\text{end}}_{s''} - l^{\text{end}}_{s'})$. We have $l^{\text{end}}_{s'.\tilde{s}'} = l^{\text{end}}_{s''}$, $l^{\text{drive}}_{s'.\tilde{s}'} = l^{\text{drive}}_{s'} \leq l^{\text{drive}}_{s''}$, $l^{\text{last\_rest}}_{s'.\tilde{s}'} + l^{\text{slack}}_{s'.\tilde{s}'} \geq l^{\text{last\_rest}}_{s'} + l^{\text{slack}}_{s'} \geq l^{\text{last\_rest}}_{s''} + l^{\text{slack}}_{s''}$ and $l^{\text{last\_rest}}_{s'.\tilde{s}'} + l^{\text{push}}_{s'.\tilde{s}'} = l^{\text{last\_rest}}_{s'} + l^{\text{push}}_{s'} \geq l^{\text{last\_rest}}_{s''} + l^{\text{push}}_{s''}$. Thus, the conditions of the lemma hold for $s'.\tilde{s}'$ and s''. Dominance of $s'.\tilde{s}'$ over s'' can be shown analogously to the first case. Thus, any sequence of activities that can be appended to s'' can be appended to $s'.\tilde{s}'$. □

Lemma 2 Let s', s'' be pseudo-feasible schedules for the partial tour $n_1, \ldots, n_\mu$ with $1 \leq \mu < \lambda$. Schedule s' dominates s'' if

$$l^{\text{end}}_{s'} + t^{\text{rest}} \leq l^{\text{end}}_{s''}$$

Proof. If $l^{\text{end}}_{s'} + t^{\text{rest}} = l^{\text{end}}_{s''}$ let $\tilde{s}' := (\texttt{REST}, t^{\text{rest}})$. If $l^{\text{end}}_{s'} + t^{\text{rest}} < l^{\text{end}}_{s''}$ let $\tilde{s}' := (\texttt{REST}, t^{\text{rest}}).(\texttt{IDLE}, l^{\text{end}}_{s''} - l^{\text{end}}_{s'} - t^{\text{rest}})$. Then, dominance of $s'.\tilde{s}'$ over s'' can be shown analogously to the previous lemma. □

In order to be able to efficiently solve the US-TDSP by subsequently constructing pseudo-feasible schedules for partial tours $n_1, \ldots, n_\mu$, we now define several normality conditions. For notational reasons, we can require that each pseudo-feasible schedule $s = a_1.a_2. \ \ldots \ .a_k$ satisfies condition

$$\text{for each } 1 < i \leq k : a^{\text{type}}_{i-1} \neq a^{\text{type}}_{i}. \tag{N1}$$

If we have a pseudo-feasible schedule violating (N1), we can simply transform the schedule into a pseudo-feasible schedule satisfying (N1) by merging all subsequent activities of the same type.

We can demand that the duration of each rest period is as short as possible, i.e. that any schedule $s = a_1.a_2. \ \ldots \ .a_k$ satisfies

$$\text{for each } 1 < i \leq k \text{ with } a^{\text{type}}_{i} = \texttt{REST} : a^{\text{length}}_{i} = t^{\text{rest}} \tag{N2}$$

Any schedule violating (N2) can be transformed by shortening the rest period and inserting an idle period of appropriate length. This reduces the time at which the rest period ends, but increases the amount by which the rest period can be extended by the same value.

We can require that all driving time is conducted as early as possible, i.e. that each schedule $s = a_1.a_2. \ \ldots \ .a_k$ satisfies

$$\begin{gathered}\text{for each } 1 < i < k \text{ with } a_i^{\text{type}} = \texttt{REST} \text{ and } a_{i+1}^{\text{type}} = \texttt{DRIVE}: \\ l_{s_{1,i-1}}^{\text{drive}} = t^{\text{drive}} \text{ or } l_{s_{1,i-1}}^{\text{end}} \geq l_{s_{1,i-1}}^{\text{last\_rest}} + l_{s_{1,i-1}}^{\text{extend}} + t^{\text{elapsed}}.\end{gathered} \tag{N3}$$

If a pseudo-feasible schedule does not satisfy condition (N3), we can move some of the driving time just before the rest period.

As idle activities represent unproductive periods, they should only be scheduled if they are required due to an early arrival at a work location. We can demand that idle periods are only scheduled immediately before work periods, i.e. that each schedule $s = a_1.a_2. \ \ldots \ .a_k$ satisfies

$$\text{for each } 1 < i \leq k \text{ with } a_{i-1}^{\text{type}} = \texttt{IDLE} : a_i^{\text{type}} = \texttt{WORK}. \tag{N4}$$

If a pseudo-feasible schedule does not satisfy condition (N4), we can remove the idle activity and insert it immediately before the next work period.

In the single time window problem studied by Goel (2010) idle periods are only appended to a schedule if the arrival at location μ is not in T_μ. The duration of the idle period is set to the smallest value allowing the work activity to be scheduled in T_μ. In the US-TDSP with multiple time windows we may have to schedule idle periods even if the arrival time at location μ is in T_μ. We can, however, require that no idle period is longer than required to reach the opening time of one of the time windows, i.e. we can demand that each schedule $s = a_1.a_2. \ \ldots \ .a_k$ satisfies

$$\begin{gathered}\text{for each } 1 < i \leq k \text{ with } a_{i-1}^{\text{type}} = \texttt{IDLE} \text{ and } a_i^{\text{type}} = \texttt{WORK}: \\ l_{s_{1,i-1}}^{\text{end}} \in \{t \mid t = \min T_{\mu(i)}^j, 1 \leq j \leq \tau_{\mu(i)}\}.\end{gathered} \tag{N5}$$

If a pseudo-feasible schedule does not satisfy condition (N5), we can reduce the length of the idle activity period in such a way that the start time of the work activity is set to the smallest possible value within the same time window and insert an idle period of appropriate length after the work period.

All schedules obtained by modifying a schedule as described above are pseudo-feasible and the modified schedule dominates the original schedule with respect to the conditions of Lemma 1. Obviously, some of the modifications may create a violation of another normality condition. However, by iteratively transforming a pseudo-feasible schedule, we can achieve a pseudo-feasible schedule satisfying conditions (N1) to (N5). We will say that a schedule satisfying (N1) to (N5) is in *normal form*. As we can transform any pseudo-feasible schedule into a pseudo-feasible schedule in normal form, the US-TDSP is equivalent to the problem of determining whether a pseudo-feasible schedule $s := a_1.a_2. \ \ldots \ .a_k$ with $a_1^{\text{type}} = \texttt{REST}$ and $a_1^{\text{length}} \geq 34$ exists which is in normal form.

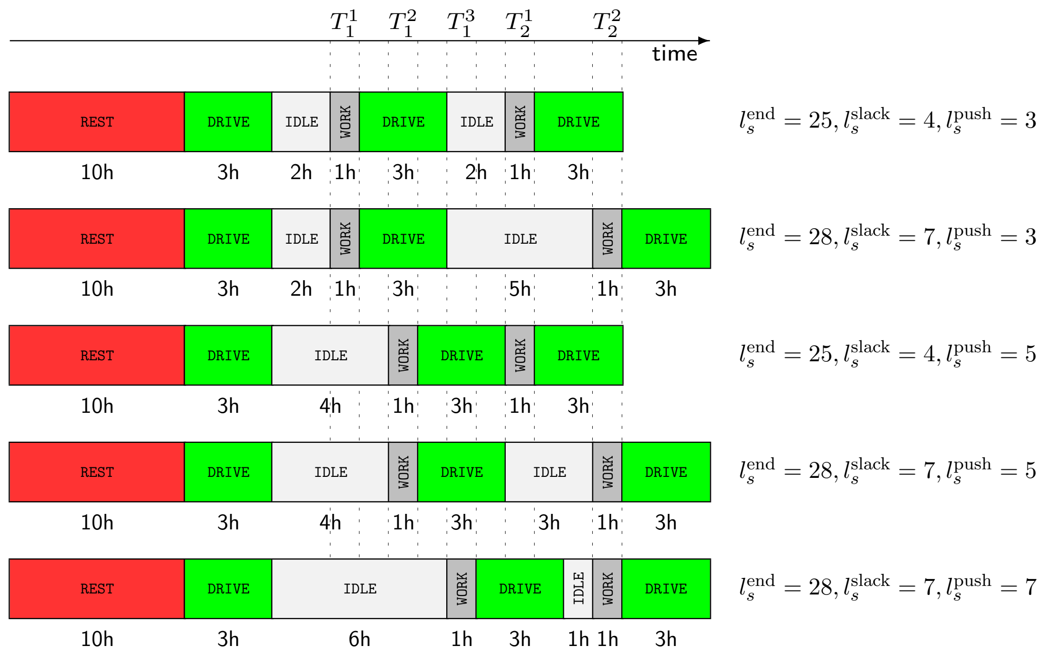

Figure 3: Pseudo-feasible schedules in normal form

Figure 3 illustrates five pseudo-feasible schedules which are in normal form. The only difference in the schedules is that the work activities begin in different time windows and that a different amount of idle time is scheduled before the work activities. We can observe that the third schedule in Figure 3 dominates the first schedule and the fifth schedule dominates the second and the fourth schedule. If we increase the last rest period in the third schedule by l_s^{extend} we obtain the third schedule of Figure 2. Note that the start time of the first work activity in the dominating schedules is in a later time window than in the dominated schedules. This illustrates that while it appears to be attractive to schedule work activities as early as possible, a later start time may be beneficial as it allows the last rest in the schedule to be increased by a larger value. The number of pseudo-feasible schedules in normal form may be thus be significantly higher in the case of multiple time windows than in the case of single time windows. In the next section we will present a scheduling method which constructs pseudo-feasible schedules in normal form until the US-TDSP is solved.

5 Scheduling Method

The normal form presented in the previous section guides us in solving the US-TDSP. Because of (N3) and (N4) each work activity must be followed by a driving activity, unless conditions (5) or (7') do not allow further driving. The duration of each driving activity must be the minimum of the remaining driving time to the next location, the maximum amount of driving that can be conducted with respect to constraint (5), and the maximum amount of driving that can be conducted with respect to constraint (7'). If the next work location is not reached after the end of a driving period, a rest period must be scheduled because of (N4). This rest period must have duration t^{rest} because of (N2). Because of (N1) and (N4) the rest period must be followed by another driving period. When the next work location is reached we have alternative possibilities of scheduling further activities. If the next location is reached within one of the time windows, the work activity can be appended to the schedule. If the arrival time at the work location increased by t^{rest} lies within one of the time windows, a rest period of duration t^{rest} followed by the next work activity can be appended to the schedule. Furthermore, an idle period followed by the work period may be appended to the schedule and a rest period of duration t^{rest} followed by an idle period and the work period may be appended to the schedule. Because of (N5), the idle period must have a duration such that the work begins at the start of one of its time windows.

We can now formulate a scheduling method which takes a set of pseudo-feasible schedules in normal form for a partial tour $n_1, n_2, \ldots, n_\mu$ and extends each schedule to construct pseudo-feasible

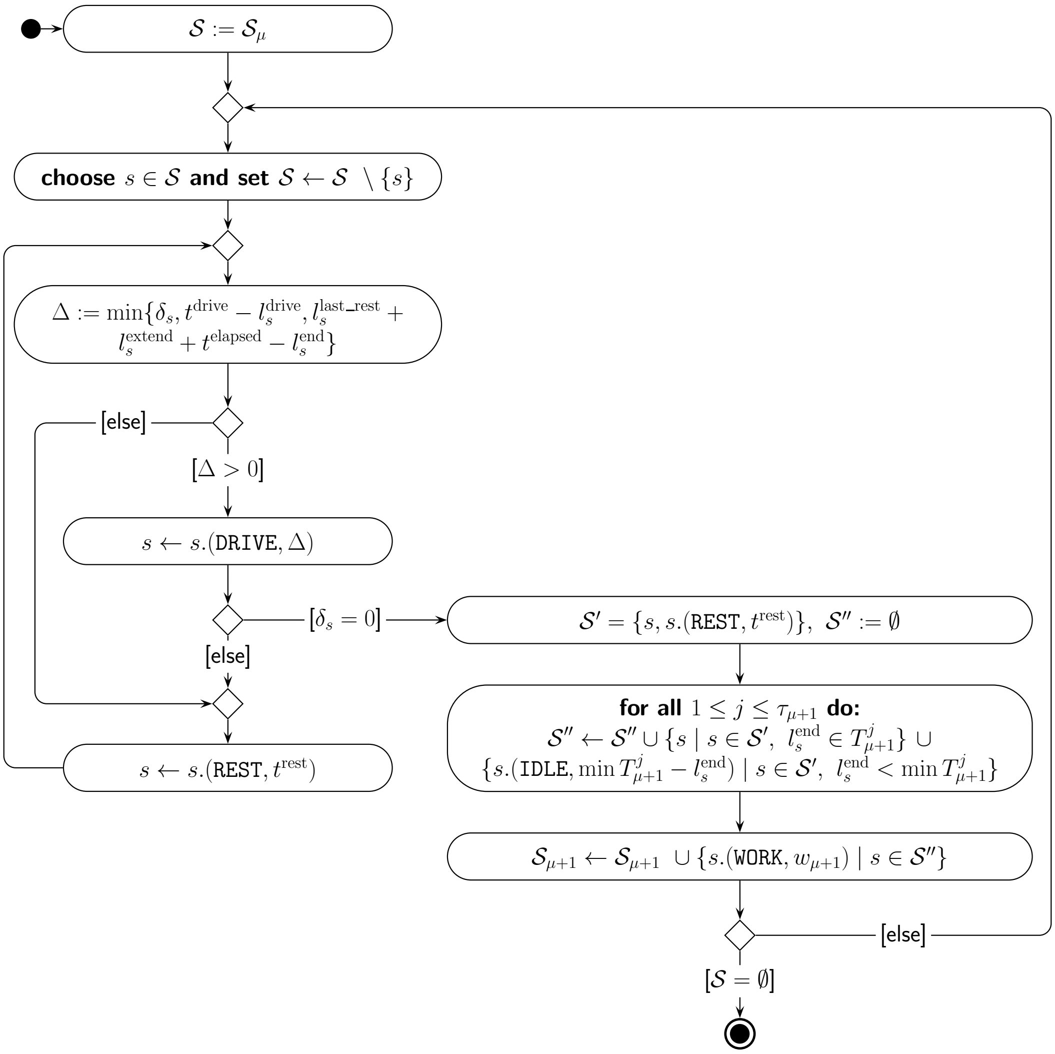

Figure 4: Scheduling method

schedules in normal form for tour $n_1, n_2, \ldots, n_\mu, n_{\mu+1}$. This process is repeated until the US-TDSP for tour $n_1, n_2, \ldots, n_\lambda$ is solved. Before invoking the scheduling method, we set

$$\mathcal{S}_1 := \big\{ s \mid s = (\texttt{REST}, \min T_1^j).(\texttt{WORK}, w_1), 1 \leq j \leq \tau_1 \big\}$$

and

$$\mathcal{S}_\mu := \emptyset \text{ for all } 1 < \mu \leq \lambda.$$

The method is then invoked with $\mu = 1$. Note, that any other schedule for the initial tour n_1 is dominated by one of the τ_1 schedules in $\mathcal{S}_1$. The scheduling method is illustrated in Figure 4. Within the scheduling method, δ_s denotes for each partial schedule s the remaining driving time required to reach the next location $n_{\mu+1}$. The scheduling method starts by setting $\mathcal{S} := \mathcal{S}_\mu$. Then it choses a partial schedule $s \in \mathcal{S}$ and removes it from $\mathcal{S}$. Now, it determines the maximum duration of the next driving activity. If this value, which is denoted by Δ, is larger than zero, a driving period of duration Δ is scheduled.

If $\Delta = 0$ or $\delta_s > 0$ a rest period is required before another driving activity may be scheduled. The method schedules a rest period of duration t^{rest} and continues with determining the maximum duration of the next driving activity. If $\delta_s = 0$ after scheduling a driving activity, the next location is reached. The method creates a set $\mathcal{S}'$ containing the schedule determined so far and an additional schedule created by appending a rest period. For each time window in $T_{\mu+1}$ and each schedule in $\mathcal{S}'$ the method adds the schedule to the set $\mathcal{S}''$ if its completion time is within the time window. For each time window in $T_{\mu+1}$ and each schedule in $\mathcal{S}'$ the method adds a new schedule to the set $\mathcal{S}''$ which is generated by adding some idle time to the schedule in $\mathcal{S}'$ if its completion time is before the opening of the time window. The method adds the work activity to each of these copies and includes them in the set $\mathcal{S}_{\mu+1}$. If $\mathcal{S} = \emptyset$, the method terminates. Otherwise, the method continues by choosing the next partial schedule in $\mathcal{S}$.

Note, that if no dominated schedules are removed, the method creates the same schedules which would be generated by solving a single time window problem for all combinations of time windows. Without removing dominated schedules from $\mathcal{S}_\mu$, the number of partial schedules generated by this approach may grow exponentially. If we remove all dominated schedules in $\mathcal{S}_\mu$ each time before we invoke the scheduling method we can, however, significantly reduce the number of partial schedules which need to be considered.

6 Performance Analysis

In order to analyse the performance of the algorithm let us partition each set $\mathcal{S}_\mu$ into the two subsets

$$\mathcal{S}_\mu^+ := \{s \in \mathcal{S}_\mu \mid l_s^{\text{drive}} > 0\} \text{ and } \mathcal{S}_\mu^0 := \{s \in \mathcal{S}_\mu \mid l_s^{\text{drive}} = 0\}.$$

The scheduling method starts with $|\mathcal{S}_1^+| = 0$ and $|\mathcal{S}_1^0| = \tau_1$. In each iteration the method creates for each schedule in $\mathcal{S}_\mu$ at most $\tau_{\mu+1}$ new schedules with $l_s^{\text{drive}} > 0$ and at most $\tau_{\mu+1}$ new schedules with $l_s^{\text{drive}} = 0$ to be included in $\mathcal{S}_{\mu+1}$. Thus, $|\mathcal{S}_{\mu+1}^+| \leq \tau_{\mu+1} \cdot |\mathcal{S}_\mu|$ and $|\mathcal{S}_{\mu+1}^0| \leq \tau_{\mu+1} \cdot |\mathcal{S}_\mu|$. For each time window $T_{\mu+1}^j$ with $1 \leq j \leq \tau_{\mu+1}$ there exists a schedule $s^j \in \mathcal{S}_{\mu+1}^0$ which dominates all other schedules in $\mathcal{S}_{\mu+1}^0$ in which the $(\mu+1)$st work activity begins in $T_{\mu+1}^j$. Thus, after removing dominated schedules we have $|\mathcal{S}_{\mu+1}^0| \leq \tau_{\mu+1}$. Unless additional assumptions are made, we doubt that a polynomial bound on the number of schedules generated can be given.

In the case of single time windows, however, we have $|\mathcal{S}_1^+| = 0$, $|\mathcal{S}_1^0| = 1$, $|\mathcal{S}_{\mu+1}^0| \leq 1$, and $|\mathcal{S}_{\mu+1}^+| \leq |\mathcal{S}_\mu|$. We can conclude that, after removing dominated schedules, we have $|\mathcal{S}_\mu| \leq \mu$ for each $1 \leq \mu \leq \lambda$. Thus, the method generates at most $\sum\limits_{1 \leq \mu \leq \lambda} \mu$ non-dominated schedules and has a worst case complexity of $O(\lambda^2)$. Now, let us again have a look at Figure 1. Each of the five schedules in Figure 1 is pseudo-feasible and in normal form and none dominates another. We can easily find other examples in which for each subproblem with $\mu \leq \lambda$ at least μ non-dominated pseudo-feasible schedules in normal form exist. Thus, the scheduling method is minimal because it does not generate more schedules in each iteration than required.

Let us now consider the US-TDSP with multiple time windows which satisfy the following condition

$$\max T_\mu^j + t^{\text{rest}} \leq \min T_\mu^{j+1} \text{ for all } 1 \leq \mu \leq \lambda \text{ and } 1 \leq j < \tau_\mu.$$

This condition states that in between subsequent time windows of the same location there are at least 10 hours during which the work must not begin. This situation occurs, for example, if, because of opening hours of docks, handling operations can only be performed between 8.00 AM and 10.00 PM. Under this condition, the US-TDSP with multiple time windows can be solved without generating more non-dominated schedules than in the single time window case. The reason for this is that, because of Lemma 2, the schedule with the smallest completion time dominates all schedules in which the last work activity is scheduled in a subsequent time window. Thus, for each location only one of the multiple time windows is actually relevant.

In order to analyse the performance in the case where handling activities must not be conducted during night or lunch time, we conducted computational experiments on several million instances in which between one and ten time windows, each from 8.00 AM to 1.00 PM or from 3 .00 PM to 8.00 PM, were associated to each work location. The duration of each work activity was set to $w_\mu = 1$, and the driving time from one location to another was set to $\delta_{\mu,\mu+1} \in \{4, 8, 12, 16\}$. The planning horizon was set to five days. The maximum number of non-dominated partial schedules generated throughout the course of the algorithm was less than twice as high in the case of two time windows compared to the case of single time windows. Increasing the number of time windows from two to ten did not bring a further increase in this number. We can thus conclude that the running time of the algorithm does not increase significantly for a wide range of problem instances with multiple time windows.

7 Conclusions

This paper studies the U.S. Truck Driver Scheduling Problem in which a sequence of λ locations must be visited within given time windows. In the case of single time windows, Archetti and Savelsbergh (2009) show that the problem can be solved in $O(\lambda^3)$. We present a scheduling method which, in the case of single time windows, solves the problem in $O(\lambda^2)$ time. The method is minimal because it does not generate more schedules in each iteration than required. Furthermore, we show that in the case of multiple time windows, we can also solve the US-TDSP in $O(\lambda^2)$ time if the gap between subsequent time windows of the same location is at least 10 hours. For problem instances in which the gap between subsequent time windows of the same location is less than 10 hours, we empirically show that the computational effort is not significantly higher.

It must be noted that the generalisations made in this paper to consider multiple time windows can also be adapted to the European Union Truck Driver Scheduling Problem studied in Goel (2010). However, no comparable complexity bound can be given due to additional provisions in European Union regulations which are not found in U.S. Hours of Service regulations.

References

C. Archetti and M. W. P. Savelsbergh. The trip scheduling problem. *Transportation Science*, 43(4): 417–431, 2009. doi: 10.1287/trsc.1090.0278.

Federal Motor Carrier Safety Administration. Hours of service of drivers. Federal Register Vol. 73, No. 224, p. 69569 - 69586, 2008.

Federal Motor Carrier Safety Administration. Interstate truck driver's guide to hours of service. http://www.fmcsa.dot.gov/rules-regulations/truck/driver/hos/fmcsa-guide-to-hos.pdf, 2009.

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

A. Goel. Truck Driver Scheduling in the European Union. *Transportation Science*, Published online ahead of print August 19, 2010, 2010. doi: 10.1287/trsc.1100.0330.

A. L. Kok, C. M. Meyer, H. Kopfer, and J. M. J. Schutten. A Dynamic Programming Heuristic for the Vehicle Routing Problem with Time Windows and European Community Social Legislation. Transportation Science (to appear), 2010.

A. T. McCartt, L. A. Hellinga, and M. G. Solomon. Work schedules of long-distance truck drivers before and after 2004 hours-of-service rule change. *Traffic Injury Prevention*, 9(3):201–210, 2008. doi: 10.1080/15389580802040287.

E. Prescott-Gagnon, G. Desaulniers, M. Drexl, and L.-M. Rousseau. European driver rules in vehicle routing with time windows. *Transportation Science*, Published online ahead of print August 19, 2010, 2010. doi: 10.1287/trsc.1100.0328.

M. W. P. Savelsbergh and M. Sol. DRIVE: dynamic routing of independent vehicles. *Operations Research*, 46:474–490, 1998.

J. L. Stone, S. M. Wolfe, J. Claybrook, J. P. Hoffa, J. Lannen, and D. Izer. Letter sent on March 9, 2009 to Raymond H. LaHood, Secretary of Transportation, U.S. Department of Transportation. http://www.citizen.org/documents/LahoodHOSLetter.pdf, 2009.

H. Xu, Z.-L. Chen, S. Rajagopal, and S. Arunapuram. Solving a practical pickup and delivery problem. *Transportation Science*, 37(3):347–364, 2003.