Exact and heuristic solution methods to the vehicle routing problem with pickup and delivery and truck driver scheduling

Ole Johannes Lindseth[a], Simen H. Sørum[a], Magnus Stålhane[a,*]

[a] *Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, Alfred Getz veg 3, N-7491, Trondheim, Norway*

Abstract

This paper presents a new vehicle routing problem (VRP) to the literature which extends the pickup and delivery variant of the VRP to also consider the hours of service regulations stipulated by the European Union. To solve this problem, we present both an exact and a heuristic solution method. The exact method is a branch-and-price (B&P) algorithm, where the subproblem combines both routing of the vehicles and scheduling of the truck drivers rests and breaks. Further, we propose several problem dependent pre-processing techniques which significantly reduce the computational time. The heuristic proposed is an adaptive large neighborhood search (ALNS) heuristic, with several new destroy and repair operators. To test the two methodologies, a set of 648 new benchmark instances are proposed based on the geography and demographics of Norway. The results of these tests show that the B&P algorithm can solve instances with up to 50 requests within 2 hours, while the ALNS can find near optimal solution to all instances within 20 minutes. We further use the ALNS heuristic to investigate the potential increase in profits for a long haul transportation company by allowing trucks to service several transportation requests simultaneously.

Keywords: truck driver scheduling, vehicle routing , pickup and delivery, branch-and-price, adaptive large neighborhood search

1. Introduction

Vehicle routing problems are some of the most heavily studied problems in the operations research literature over the last 60 years (Cordeau et al., 2007). However, there still exists many gaps between academic research and requirements from real-world applications. One such gap is the implementation of hours of service (HoS) regulations in the planning of routes for truck drivers, which only recently has received attention from the research community (Goel and Irnich, 2017). In Norway, as in many other countries, the population is geographically dispersed, and great distances are traveled to deliver cargo. An increased focus on short lead times in supply chains,

*E-mail: magnus.staalhane@ntnu.no

combined with long distances, puts pressure on drivers, which in turn may lead to accidents related to fatigue. To avoid accidents and make work less stressful for truck drivers, governments world wide have introduced HoS regulations.

In Europe, the HoS regulations are enforced by the European Union through Regulation (EC) No 561/2006 (European Union, 2006) regulating driving time, and Directive 2002/15/EC (European Union, 2006) regulating working time. The regulations define four driver activities: driving time, working time, rests, and breaks. Driving time is the actual time a truck driver operates the vehicle and is not restricted to when the vehicle is moving. Working time is the time the driver uses on non-driving activities such as loading and unloading the vehicle, in addition to the time spent driving. Waiting time is also included in the working time, unless it is part of a break or rest.

A break is a short time period used for recuperation, while a rest is a longer time period in which the truck driver manages his or her time. A daily rest must be completed within 24 hours after the end of the last daily or weekly rest period, and the duration must be at least 11 hours. Furthermore, a maximum of 9 hours of accumulated driving time since the last daily or weekly rest is allowed. The duration of a break is at least 45 minutes, and a driver must take a break after an accumulated working or driving time of at most 6 or 4.5 hours, respectively. Finally, the accumulated driving and working time within a week must not surpass 56 hours and 60 hours, respectively.

As can be seen in Section 2, the VRP literature has thus far only studied HoS regulations in the context where an operator is to distribute cargo from a centrally located depot to a set of geographically dispersed locations, using their own fleet of trucks. This is how many large businesses typically operate, however, many small- and medium-sized businesses have insufficient transportation needs to justify operating their own fleet of vehicles. Thus, they rely on third party transportation companies to transport their cargo. Such transportation companies usually fulfill the transportation requirements for many businesses simultaneously, and they therefore face a VRP where cargo is to be picked up and delivered at multiple locations.

The purpose of this paper is to present a new problem to the research community, based on a case of a transportation company serving multiple geographically dispersed businesses. The problem extends the well-studied pickup and delivery problem with time windows (PDPTW), by also considering the HoS regulations. The problem consists of transporting a profitable subset of available transportation requests (henceforth referred to as a *request*), dispersed over a large geographical area, using a fleet of trucks. A request consists of a pickup and a delivery location, an earliest and latest time to start service at each of these locations, and the weight and volume of the cargo. Each request pays a revenue, if transported, and the vehicles available to serve these requests have limited capacity both in terms of weight and volume. We further assume that there

is no central depot, so each vehicle has a (potentially) different starting location, and that the problem is open ended, i.e. that a vehicle can end its route at any location. The objective is to create one route for each vehicle, from its starting location, servicing a subset of the requests, such that the total profit of the vehicle fleet is maximized. Due to the large distances involved, the planning horizon spans several days, and thus HoS regulations have to be considered for the drivers of these vehicles.

The contributions of this paper is an exact and a heuristic solution method for solving the problem. The exact method is based on branch-and-price (BP), where the subproblem is solved using a labeling algorithm. Several pre-processing techniques, tailored to this problem, are also described. The heuristic method is an adaptive large neighborhood search (ALNS) heuristic, where several new destroy and repair operators are presented. Further, we also present a set of 486 benchmark instances for the problem, based on real geographical data. Optimal solutions or dual bounds for many instances are found using the exact solution method, while the ALNS heuristic provides best-known solutions for the remaining instances. Insights into the performance of the heuristic solution method are provided, including the contribution of each destroy and repair operator. Finally, we analyze the benefit of allowing cargo from multiple requests on-board the vehicle at the same time (Less-Than-Truckload transportation), over allowing only a single request (Full Truckload transportation) which is the standard mode of operation for many transportation companies today.

The remainder of this paper is structured as follows. In Section 2 we provide an overview of existing literature related to the problem studied, while in Section 3 we give a formal problem description and a route based mathematical model of the problem. Section 4 describes a BP algorithm, and Section 5 presents an ALNS heuristic to solve the problem. Section 6 presents the set of benchmark instances created for this problem, before Section 7 presents a computational study of the solution methods, and gives som managerial instights. Finally, some concluding remarks are given in Section 8.

2. Literature Review

The vehicle routing problem (VRP) is a thoroughly studied combinatorial optimization problem where a set of transportation requests are to be transported from a depot to a set of customers by a fleet of vehicles (Cordeau et al., 2007). The problem studied in this paper combines several variants and extensions of the VRP, each of which has a rich body of research literature.

According to the classification scheme presented by Berbeglia et al. (2007), the problem studied in this paper is a variant of the multi vehicle one-to-one pickup and delivery problem (m|1-1|PDP), since it has more than one vehicle and exactly one pickup location and one delivery location for all requests. The problem further extends the 1-1-PDP since it also incorporates several VRP

extentions: it is open-ended (Li et al., 2007), includes time windows (Desrosiers et al., 1995), multiple depots (Montoya-Torres et al., 2015), optional requests (Archetti et al., 2014), and truck driver scheduling (TDS) (Goel, 2010). The problem is also an extension of the ship routing and scheduling problem (SRSP) which can be modelled as a one-to-one open-ended PDP with time windows, optional requests and a heterogeneous fleet (Christiansen and Fagerholt, 2014). In the following we focus on the existing literature on the PDPTW and TDS, as these are the most prominent characteristics of the problem.

Several exact solution approaches have been proposed for the one-to-one PDPTW. A Branch-and-cut algorithm for the PDPTW was proposed by Cordeau (2006), and later improved by Ropke et al. (2007). Ropke and Cordeau (2009) propose a Branch-and-Price-and-Cut algorithm, while Baldacci et al. (2011) propose a set-partitioning-based integer formulation where a bounding procedure with two dual ascent heuristics and a cut-and-column generation procedure is used to solve the problem. Gschwind et al. (2018) use bidirectional labeling to solve the same problem and reveal that utilizing bidirectional labeling may reduce the computational time by more than 40 %. Finally, Homsi et al. (2020) propose an exact BP algorithm to solve the SRSP.

Due to the high complexity of the problem and often limited available computational time, heuristic solution methods have been widely used for the PDPTW. Early approaches include tabu search based metaheuristics by Li and Lim (2003) and Nanry and Barnes (2000), and Large Neighborhood Search (LNS) heuristics by Bent and Van Hentenryck (2003) and Curtois et al. (2018). Ropke and Pisinger (2006) introduce the Adaptive Large Neighborhood Search (ALNS) as an extension of LNS to solve a PDPTW with a heterogeneous fleet of vehicles. The ALNS introduces an adaptive layer facilitating for using several destroy and repair operators during the search while each operator is rewarded according to the contribution in terms of finding high-quality solutions or solutions which facilitate exploration. Gschwind and Drexl (2019) propose an ALNS heavily inspired by Ropke and Pisinger (2006) to solve the the Dial-a-Ride-Problem (DARP), which is a variant of the one-to-one PDPTW, focusing on the transportation of people (Doerner and Salazar-González, 2014).

The Truck Driver Scheduling Problem (TDSP) consists of finding a schedule for a given vehicle route which adheres to the HoS regulations. Methods solving the TDSP are needed as subroutines in both exact and heuristic approaches for VRPs where the route duration spans several days. The TDSP was first introduced by Archetti and Savelsbergh (2009) as the trip scheduling problem, which consisted of finding a feasible driver schedule for a given vehicle route, adhering to the US HoS regulations. The first study of the TDSP with the EU HoS regulations was that of Goel (2009). He introduced a labeling algorithm for the problem that, among other things, allowed for preemptive breaks and rests, and time windows. Goel (2010) extends the work by incorporating the flexible EU HoS rules that allow for breaks and rests to be split in two parts. To solve this extended

version of the problem he propose a Breadth-First-Search (BFS), which is guaranteed to find a feasible schedule if it exists for the route. Drexl and Prescott-Gagnon (2010) propose a dynamic programming approach in the context of an Elementary Shortest Path Problem with Resource Constraints (ESPPRC) for the same set of regulations. Furthermore, Goel (2012) proposes a dynamic scheduling method for both the EU and US HoS regulations where the duration of the route is minimized. Finally, methods for the Canadian and Australian HoS regulations are proposed by Goel and Rousseau (2012) and Goel et al. (2012), respectively.

Recently several extensions of the TDSP have been proposed. Goel and Kok (2012) consider both single and multiple time windows for the TDSP, and the proposed method solves the single time window case in $O(n^2)$ time. Furthermore, Rancourt et al. (2013) propose a TDS subroutine for multiple time windows used in a Tabu Search heuristic. Goel and Vidal (2014) proposes a method for minimizing the duration of truck driver schedules for the US regulations, based on Goel and Kok (2012). The minimization is done as described by Goel (2012). Finally, Goel (2018) proposes a method that restricts nighttime work while extending daily driving time and reducing rests.

In recent years, combinations of the VRP and TDS, often referred to as the Vehicle Routing and Truck Driver Scheduling Problem (VRTDSP) has received increased attention. The first exact solution method for the VRTDSP was proposed by Goel and Irnich (2017), which solves the problem with a Branch-and-Price algorithm where the pricing problem is solved using a modified network, where additional nodes are inserted between each customer node, facilitating for uniquely computing the length of each driver activity. Furthermore, Tilk and Goel (2020) propose a Branch-and-Price-and-Cut algorithm using bidirectional labeling for solving the pricing problem based on modified network proposed by Goel and Irnich (2017). Goel (2018) performs an experimental analysis on the impact of hours of service regulations on feasibility, route length, costs, and road safety in addition to reviewing the state-of-the-art in vehicle routing literature.

There also exist several heuristics for the VRTDSP. Zäpfel and Bögl (2008) propose a Genetic Algorithm incorporating a Tabu Search heuristic for solving a VRPTW before personnel is assigned to each route using a simple heuristic respecting hours of service regulations. Furthermore, Prescott-Gagnon et al. (2010) proposed a hybrid of an LNS algorithm and a Branch-and-Price heuristic where new solutions are constructed by a Tabu Search based column generator, while respecting the EU hours of service regulations. Goel (2009) considers the absolute EU regulations and presents two methods for scheduling truck driver activities, and embed them into a LNS algorithm. Kok et al. (2010) consider the flexibility rules from the EU regulations while also considering working hour rules which are often ignored in the literature. It combines a restricted dynamic programming heuristic and a scheduling heuristic. Furthermore, Rancourt et al. (2013) enhance the search space of their unified Tabu Search heuristic by allowing intermediate infeasible

solutions by using an objective function with self-adjusting penalties.

Goel and Vidal (2014) propose a Hybrid Genetic Search with Advanced Diversity Control (HGSADC) based on the method proposed by Vidal et al. (2012). The former uses an extension of the method proposed by Goel (2010) for checking feasibility of the vehicle routes with regards to the EU hours of service regulations, while also considering the US, Canadian and Australian hours of service regulations. Each individual is given a fitness value based on a penalized cost of violating capacity and time window restrictions and a contribution score of the individual to the diversity of the population. A set of speed-up techniques are proposed to solve the problem. Koç et al. (2017) propose a multi-start heuristic with a combination of ALNS and MIP for solving a VRTDSP with idling options and a more comprehensive objective function. Furthermore, Goel et al. (2020) investigate the advantages of having a mixed fleet of single- and team drivers by proposing a Hybrid Genetic Algorithm based on the method proposed by Goel and Vidal (2014). Using team driving may be profitable as it opens the opportunity of the vehicle not staying idle while one of the drivers is doing a rest. They conclude that considerable cost savings may be achieved.

In conclusion, the problem may be classified as a one-to-one open-ended pickup and delivery problem with time windows, multiple depots, optional requests, and truck driver scheduling. Nevertheless, as the most prominent attributes are PDPTW and TDS, the problem is henceforth referred to as the pickup and delivery and truck driver scheduling problem (PDTDSP). To the best of our knowledge, this problem has not been studied previously in the literature, and thus no exact nor heuristic solution methods have been proposed for solving this problem.

3. Problem Description and Mathematical Model

The PDTDSP consists of n requests that may be transported by a set of $\mathcal{V}$ vehicles. The problem may be defined on a graph $G = (N, A)$, where N is the set of nodes and $A \subset N \times N$ is the set of arcs connecting the nodes. The set $N = N^P \cup N^D \cup_{v \in V} \{o(v), d(v)\}$ may be divided into the disjoint subsets $N^P = \{1, ..., n\}$, $N^D = \{n+1, ..., 2n\}$, and one origin and destination node pair ($o(v)$,$d(v)$) for each vehicle. Each request is modeled with two nodes, node i representing the pickup location and $n+i$ representing the delivery location. A node i includes the information of the weight and volume, D_i^W and D_i^V, of the cargo, time window for service, $[\underline{T}_i, \overline{T}_i]$, duration of service at the node, S_i, and the revenue earned, R_i. The revenue at the delivery nodes is set to 0, while the cargo weight and volume is the negative of the corresponding pickup node. Furthermore, each arc holds information of the driving time, T_{ij}, and cost, C_{ij}, for traversing the arc. The capacity of each vehicle in terms of weight and volume is given as W^C and V^C.

A feasible route for vehicle v is a path through the network from $o(v)$ to $d(v)$, where each node is visited at most once, delivery nodes are visited after their corresponding pickup node, the

vehicle's capacity is not violated, and the hours of service regulations are respected. For the latter, $T^{drive|R}$ and $T^{elapsed|R}$, and $T^{drive|B}$ and $T^{elapsed|B}$, describe the maximum allowed accumulated driving time and elapsed time since the last rest or break, respectively. In addition, the maximum allowable weekly accumulated driving time and working time is denoted as $T^{drive|W}$ and $T^{work|W}$, respectively. The horizon of the problem is given as $T^{horizon}$, while the minimum length of a rest and a break are given as T^{rest} and T^{break}, respectively.

A route based formulation of the problem is stated below. The set R_v denotes the set of routes driven by vehicle $v \in \mathcal{V}$ and is indexed by r. The parameter P_{vr} denotes the profit from utilizing vehicle $v \in \mathcal{V}$ and route $r \in \mathcal{R}_v$, and is calculated as the total revenue of the requests serviced minus the total transportation cost of the route. A_{ivr} takes the value 1 if pickup $i \in \mathcal{P}$ is served by vehicle $v \in \mathcal{V}$ on route $r \in \mathcal{R}_v$, 0 otherwise. A variable λ_{vr} is equal to 1 if vehicle $v \in \mathcal{V}$ drives route $r \in \mathcal{R}_v$, 0 otherwise.

$$\max \quad z = \sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_v} P_{vr} \lambda_{vr}, \tag{1}$$

$$\sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_v} A_{ivr} \lambda_{vr} \leq 1, \qquad i \in N^P, \tag{2}$$

$$\sum_{r \in \mathcal{R}_v} \lambda_{vr} = 1, \qquad v \in \mathcal{V}, \tag{3}$$

$$\lambda_{vr} \in \{0, 1\}, \qquad \forall v \in \mathcal{V}, r \in \mathcal{R}_v. \tag{4}$$

The objective function (1) computes the total profit from assigning at most one route to each of the vehicles in the problem. Constraints (2) ensure that each transportation request is served at most once, while constraints (3) ensure that only one route can be assigned to each vehicle. Constraints (4) ensure that λ_{vr} is binary.

4. Branch and Price

Each λ-variable, henceforth referred to as a *column*, in the mathematical model described above, may be described by a path from $o(v)$ to $d(v)$ through the graph G. The number of feasible paths through the graph grows exponentially with the number of nodes and arcs. Therefore it is impractical, or even impossible, to generate all of them for large problem instances. To circumvent this problem, we propose a solution method based on Branch-and-Price (Barnhart et al., 1998), where only a subset of all columns are explicitly considered, while implicitly accounting for the remaining columns.

Branch-and-Price is a branch-and-bound (B&B) approach, where the linear relaxation of the mathematical model is solved in each B&B-node using column generation (Lübbecke and Desrosiers, 2005). In column generation, a linear program (LP), referred to as the master problem, is solved by iterating between a restricted master problem (RMP) consisting of a subset of columns, and

a subproblem (SP) tasked with finding columns with a positive reduced cost (for a maximization problem) that may improve the solution to the RMP. In a given iteration, the RMP is solved as a LP to obtain a dual solution which is used to formulate the objective of the SP. The SP is usually solved using dynamic programming, often formulated as labeling algorithms. If the SP finds a column with positive reduced cost, it is added to the RMP, initiating a new iteration. Otherwise there exist no column that improves the RMP, and the current solution is thus optimal for the full LP. Normal branch-and-bound rules are used to branch, prune and search the tree. For the problem solved in this paper, the RMP is defined by the model (1)–(4), but where the binary requirements are relaxed, and $\mathcal{R}_v$ only contains a small subset of the feasible routes for vehicle v.

Section 4.1 describes the labeling algorithm, while Section 4.2 presents the definition of a label with resources and resource windows. Furthermore, Section 4.3 introduces the resource extension functions of the algorithm, and the dominance criterion is described in Section 4.4. Finally, we present some tailored pre-processing techniques in Section 4.5, and describe some implementation details in Section 4.6.

4.1. Labeling Algorithm

The SP is solved as a labeling algorithm, following the description given by Irnich and Desaulniers (2005), where labels are used to represent a feasible path from the start node s to a node i in the network, and the resources accumulated along that path. A label $L(i, T, L^*)$, contains the last node on the path, i, a vector of resource values, T, and a pointer to the parent label, L^* used to retrace the full path. A pseudo code for a generic labeling algorithm is given in Algorithm 1. The labeling algorithm starts by initiating a set of unprocessed labels, U, only consisting of one label $L(o(v), T_0, NULL)$ for each vehicle, where T_0 is the initial resource vector, and the parent label pointer is a zero pointer, $NULL$. Note that since the only difference between the vehicles are their starting node ($o(v)$), we can handle all of them in a single subproblem.

In the algorithm, a selected label $L = (i, T, *)$, representing a partial path from node s to i with accumulated resources T, is removed from U. This partial path is extended along all arcs $(i, j) \in A$ to create new partial paths, each one represented by a label L'. The resource consumption of label L' is calculated according to so-called *resource extension functions* (REF), given by $f_{ij}(T)$. The resource consumption of each extended label is then checked to see if it is within the *resource window* $[a_j, b_j]$ at node j. If it is, then it is checked for dominance against other labels ending at node j, stored in the set $\mathcal{L}_j$. If L' is not dominated it is added to $\mathcal{L}_j$ and U, and the algorithm removes all labels dominated by L'. The algorithm terminates when there are no unprocessed labels left in U, returning all non-dominated labels who has reached the destination node $d(v)$. In the following we describe which resources are stored in T, what their resource window is, and the REFs for each arc.

Algorithm 1 Label algorithm

$U = \cup_{v \in \mathcal{V}} \{L(o(v), \mathcal{R}_0, NULL)\}$
while $U \neq \emptyset$ **do**
  $L = (i, T, *) =$ RemoveFirst(U),
  for $(i, j) \in A$ **do**
    $L' = (j, f_{ij}(T), L)$
    if $a_j \leq f_{ij}(\mathcal{R}) \leq b_j$ **then**
      if no label in $\mathcal{L}_j$ dominates L' **then**
        $\mathcal{L}_j = \mathcal{L}_j \cup \{L'\}$
        $\mathcal{U} = \mathcal{U} \cup \{L'\}$
        remove labels dominated by L' from $\mathcal{L}_j$ and $\mathcal{U}$
      end if
    end if
  end for
end while
return paths in $\cup_{v \in \mathcal{V}} \mathcal{L}_{d(v)}$

4.2. Label Resources & Resource Windows

Table 1 presents the resources used in defining a label representing a partial path ending at node i with notation, description, and resource window. The resources are based on those proposed by Ropke and Cordeau (2009) and Goel and Irnich (2017) for the PDPTW and VRPTDS, respectively. Furthermore, resources for handling accumulated weekly driving and working time are added.

Table 1: Label definition with corresponding resources and resource windows.

Resource	**Resource Description**	**Resource Window**
$\overline{p}$	Accumulated reduced cost after visiting node i	$[-\infty, \infty]$
l^W	Load of the vehicle after visiting node i in terms of weight	$[0, W^C]$
l^V	Load of the vehicle after visiting node i in terms of volume	$[0, V^C]$
t^{time}	Time elapsed since start of route	$[\underline{T_i}, \overline{T_i}]$
t^{dist}	Remaining driving time to the next node, j	$[0, 0]$
$t^{drive\|R}$	Accumulated driving time since the end of the last rest	$[0, T^{drive\|R}]$
$t^{elapsed\|R}$	Time elapsed since the end of the last rest	$[0, T^{elapsed\|R}]$
$t^{latest\|R}$	Latest time for when a rest must end	$[0, \infty]$
$t^{drive\|B}$	Accumulated driving time since the last break or rest	$[0, T^{drive\|B}]$
$t^{elapsed\|B}$	Time elapsed since the end of the last break or rest	$[0, T^{elapsed\|B}]$
$t^{latest\|B}$	Latest time for when a break must end	$[0, \infty]$
$t^{drive\|W}$	Total accumulated weekly driving time	$[0, T^{drive\|W}]$
$t^{work\|W}$	Total accumulated weekly working time	$[0, T^{work\|W}]$
$\mathcal{U}$	Set of unreachable nodes on the route	$\mathcal{U} \subseteq N^P$
$\mathcal{O}$	Set of requests started but not completed on this route	$\mathcal{O} \subseteq N^P$

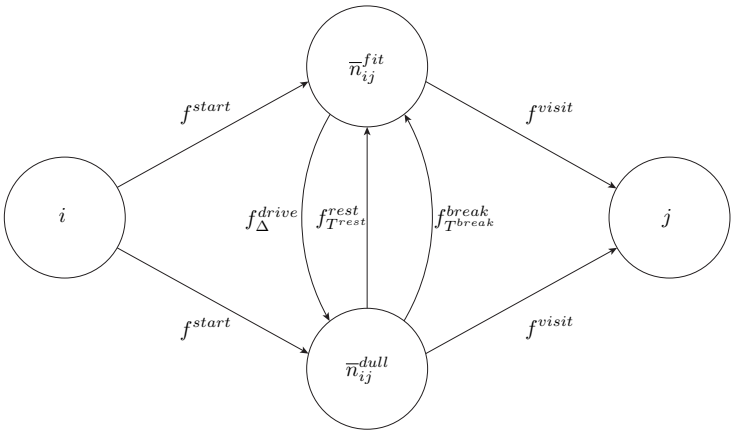


Figure 1: Auxiliary Network proposed by Goel and Irnich (2017). The network describes the possible extensions of a label by traversing the arc (i, j).

The last node on a partial path represented by label L is denoted $i(L)$ with a a similar notation is used for the resources. Thus, the entire resource vector of a label is:

$$T(L) = [\overline{p}(L), l^W(L), l^V(L), t^{time}(L), t^{dist}(L), t^{drive|R}(L), t^{elapsed|R}(L), t^{latest|R}(L),$$
$$t^{drive|B}(L), t^{elapsed|B}(L), t^{latest|B}(L), t^{drive|W}(L), t^{work|W(L)}, \mathcal{U}(L), \mathcal{O}(L)]$$

For any extension of a label, L, along the arc (i, j) to be feasible, the resources must stay within their respective resource windows stated in Table 1.

4.3. Resource Extension Functions

To account for the HoS regulations, Goel and Irnich (2017) presented a modified network to handle the driver scheduling constraints within the labeling framework of Irnich and Desaulniers (2005). Between each pair of nodes (i, j), the arc connecting them is replaced with a small network, as shown in Figure 1. Instead of a single REF, this network has five REFs: f^{start}, f^{drive}, f^{rest}, f^{break}, and f^{visit}.

After ending the service at node i, f^{start} initializes the start of driving towards node j, by setting $t^{dist}(L') = T_{ij}$. The REF f^{drive} uses a pre-calculated maximum driving time which guarantees that the driving activity does not violate any constraints. The maximum driving time is stated in Equation (5) and is given as the minimum of the remaining driving to reach the destination node, and the remaining allowed driving and working time before a rest or break must be taken, respectively. Furthermore, the REFs f^{rest} and f^{break} are used to model taking a rest of T^{rest} hours or a break of T^{break} hours, respectively. Finally, f^{visit} handles the service at node j, while extending the last rest or break to avoid unnecessary waiting time.

$$\Delta = min[t^{dist}, T^{drive|R} - t^{drive|R}, T^{drive|B} - t^{drive|B},$$
$$T^{elapsed|R} - t^{elapsed|R}, T^{elapsed|B} - t^{elapsed|B}] \tag{5}$$

Table 2 summarizes the resource extension functions f^{drive}, f^{break}, f^{rest}, and f^{visit}. The resources in the resource column indicate the new value of the resource in label L' after being extended by the REF from label L along arc (i, j). Furthermore, the REF columns f^{drive}, f^{break}, f^{rest}, and f^{visit} state how the resource is updated based on the previous resource value and the changes being made by performing the driver activity modeled by the REF. For ease of exposition we have left all entries where the resource value stays unchanged, as blank in the table.

Table 2: Summary of the extensions of resources handled by the resource extension functions f^{drive}, f^{break}, f^{rest} and f^{visit}

Resource	**Resource Extension Function**			
	f^{drive}	f^{break}	f^{rest}	f^{visit}
$\overline{p}(L')$				$\overline{p}(L) + \overline{C}_{ij}$
$l^W(L')$				$l^W(L) + D_j^W$
$l^V(L')$				$l^V(L) + D_j^V$
$t^{time}(L')$	$t^{time}(L) + \Delta$	$t^{time}(L) + T^{break}$	$t^{time}(L) + T^{rest}$	$max\{t^{time}(L), \underline{T}_j\} + S_j$
$t^{dist}(L')$	$t^{dist}(L) - \Delta$			
$t^{drive\|R}(L')$	$t^{drive\|R}(L) + \Delta$		0	
$t^{elapsed\|R}(L')$	$t^{elapsed\|R}(L) + \Delta$	$t^{elapsed\|R}(L) + T^{break}$	0	$max\{t^{elapsed\|R}(L), \underline{T}_j - t^{latest\|R}(L)\} + S_j$
$t^{latest\|R}(L')$			∞	$min\{t^{latest\|R}(L), \overline{T}_j + S_j - t^{elapsed\|R}(L)\}$
$t^{drive\|B}(L')$	$t^{drive\|B}(L) + \Delta$	0	0	
$t^{elapsed\|B}(L')$	$t^{elapsed\|B}(L) + \Delta$	0	0	$max\{t^{elapsed\|B}(L), \underline{T}_j - t^{latest\|B}(L)\} + S_j$
$t^{latest\|B}(L')$	$min\{t^{latest\|B}(L), t^{latest\|R}(L) + t^{elapsed\|R}(L) - t^{elapsed\|B}(L) - \Delta\}$	∞	∞	$min\{t^{latest\|B}(L), \overline{T}_j + S_j - t^{elapsed\|B}(L)\}$
$t^{drive\|W}(L')$	$t^{drive\|W}(L) + \Delta$			
$t^{work\|W}(L')$	$t^{work\|W}(L) + \Delta$			$t^{work\|W}(L) + S_j$
$\mathcal{U}(L')$				$\mathcal{U}(L) \cup \{j\} \cup \mathcal{U}_{j,t}$
$\mathcal{O}(L')$				$\begin{cases} \mathcal{O}(L) \cup \{j\} & \text{if } j \in \mathcal{P} \\ \mathcal{O}(L) \backslash \{j - N^P\} & \text{if } j \in \mathcal{D} \end{cases}$

Equation (6) states that the modified cost along arc (i, j), $\overline{C}_{ij}$, where π_j is the value of the dual variable from constraints (2) and α_v is the value of the dual variable from constraints (3).

$$\overline{C}_{ij} = R_j - C_{ij} - \pi_j - \begin{cases} \alpha_v & \text{if } i = o(v) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

4.4. Dominance Criteria

The dominance criterion is a pairwise comparison of the resources and are based on the dominance criteria of Ropke and Cordeau (2009) and Goel and Irnich (2017). Furthermore, dominance criteria are added for $t^{drive|W}$ and $t^{work|W}$. For a label, L_1, to dominate another label, L_2 the following conditions must apply:

- $i(L_1) = i(L_2)$

- $\overline{p}(L_1) \geq \overline{p}(L_2)$
- $\mathcal{V}(L_1) \subseteq \mathcal{V}(L_2)$
- $\mathcal{O}(L_1) \subseteq \mathcal{O}(L_2)$
- $t^{time}(L_1) \leq t^{time}(L_2)$
- $t^{dist}(L_1) \leq t^{dist}(L_2)$
- $t^{drive|R}(L_1) \leq t^{drive|R}(L_2)$
- $t^{elapsed|R}(L_1) \leq t^{elapsed|R}(L_2)$
- $t^{latest|R}(L_1) \geq t^{latest|R}(L_2)$
- $t^{drive|B}(L_1) \leq t^{drive|B}(L_2)$
- $t^{elapsed|B}(L_1) \leq t^{elapsed|B}(L_2)$
- $t^{latest|B}(L_1) \geq t^{latest|B}(L_2)$
- $t^{drive|W}(L_1) \leq t^{drive|W}(L_2)$
- $t^{work|W}(L_1) \leq t^{work|W}(L_2)$

The dominance criteria state that the current node of the label must be the same, the accumulated reduced cost must be larger or equal, and that the elapsed time and the remaining driving time to node the next node must be smaller or equal. Furthermore, the accumulated driving time since the last rest or break and the elapsed time since the last rest or break must be smaller or equal, while the latest end of the last rest or break must be larger or equal. Finally, the total accumulated weekly driving and working time must be smaller or equal. It was shown by Desaulniers et al. (1998) that as long as all resources are non-decreasing, doing a pair-wise comparison like above, is sufficient to establish a dominance relation.

4.5. Data Preprocessing

The solution time of the labeling algorithm increases with the number of labels created. Consequently, it is desirable to reduce the number of labels, which may be achieved by using certain preprocessing techniques. To discard labels that cannot lead to feasible vehicle routes as early as possible in the labeling process, we solve a simplified version of the PDTDSP problem, where we only consider a subset of the resources, on a subset of nodes W. The goal is to determine the latest possible time we can start service at node k, given that we subsequently have to visit all the nodes in W.

We solve this problem on a graph with $N = \{k\} \cup W \cup d(v)$ and use the following subset of the resources described above: t^{time}, t^{dist}, $t^{drive|R}$, $t^{elapsed|R}$, $t^{latest|R}$, and $\mathcal{O}$, remove the break related terms from the calculations of Δ and omit the f^{break} REF. In addition, we introduce two new resources, t^{slack} keeping track of how much we can postpone the arrival at node k and still have a feasible schedule, and t^{wait} which is the accumulated waiting time. These resources are initiated as $t^{slack}(L_0) = \overline{T}_j - \underline{T}_j$ and $t^{wait}(L_0) = 0$, and are updated using the f^{visit} REF as:

$$t^{wait}(L') = t^{wait}(L) + \max\{\underline{T}_j - t^{time}(L), 0\}, \tag{7}$$

$$t^{slack}(L') = \min\{t^{slack}(L), \overline{T}_j - t^{time}(L) + t^{wait}(L)\}. \tag{8}$$

We also introduce a resource v_i for $i \in W$ which is increased to 1 when node i is visited. Since this problem will only be solved on small networks, we also omit the dominance tests from the labeling algorithm. The latest time it is possible to arrive at node k, we denote $t_k^{late}(W) = \underline{T}_k + t^{slack}(L)$, is the partial path at node $d(v)$ with the smallest value of $t^{slack}(L)$ given that $v_i = 1, \forall i \in W$.

This can be used both to discard paths the cannot lead to feasible routes, and to increase the size of the unreachable set. We define the following two sets:

$$\mathcal{U}_k(t) = \left\{ i \in N^P | t_k^{late}(W) < t, W = \begin{cases} \{i, i+n, k+n\}, \text{ if } k \in N^P \\ \{i, i+n\} \text{ if } k \in N^D \end{cases} \right\}, \tag{9}$$

$$\mathcal{F}_k(t) = \left\{ W \subset N^P \middle| |W| \leq 3, W^D = \{i \in N^D | i - n \in W\}, t_k^{late}(W^D) < t \right\}. \tag{10}$$

where $\mathcal{U}_k(t)$ is the set of all requests where it is not possible to service both the pickup and delivery node within their time windows, by extending a partial path at node k arriving later than time t. The set $\mathcal{F}_k(t)$ contain all subsets of pickup nodes with a cardinality less than 4 where is it not possible to service all the corresponding delivery nodes by extending a partial path at node k arriving later than time t.

Then we can re-define the f^{visit} for $\mathcal{U}(L') = \mathcal{U}(L) \cup \{j\} \cup \mathcal{U}_j(t^{time}(L'))$ to increase the number of unreachable nodes, which may lead to more dominance. Further, we may discard any label L if $\exists W \subseteq \mathcal{O}(L), W \in \mathcal{F}_{i(L)}(t^{time}(L))$. In this case we know that it is not possible to deliver all the open requests within their time windows, and thus label L may be discarded.

4.6. Implementation details

We use several strategies to reduce the time spent solving the SP. The first strategy is to solve the SP without the break-related resources and REFs. By doing this we solve a relaxation of the original SP, and then check if it is possible to insert breaks into the optimal solution to this relaxed SP. Three situations may occur. 1) We find a reduced cost column where it is feasible to insert breaks, 2) the optimal solution to the relaxed SP has a reduced cost of 0, and 3) we find positive reduced cost columns, but itis not possible to insert breaks into any of them. In the first two cases, we either initiate a new iteration of CG, or stop as we have proven optimality. Only in case 3, do we have to solve the full SP with the break resources added.

The second strategy is to solve the labeling algorithm on a reduced graph, to quickly find positive reduced cost columns. Here we modify the description of Desaulniers et al. (2008) to a pickup and delivery context. In each CG iteration we select a subset of arcs $(i, j) \in A : j \in N^P$ based on their reduced cost ($\overline{c}_{ij} = P_j - \alpha_j^* - C_{ij}$). For each node $j \in N^P$ we select the n incoming arcs with the highest reduced cost, while for $i \in N$ we choose the n outgoing arcs with highest reduced costs. Note that we keep all incoming arcs into delivery nodes (N^D) since extensions along

these arcs are limited by the set $\mathcal{O}$. We then create networks with only the n best and 10 best arcs into/out of each node are added to the network. The exception is that We keep all arcs into delivery nodes, as feasible extensions into delivery nodes are severely limited by the set $\mathcal{O}$.

The acceleration strategies are applied in a hierarchical order, where each new CG iterations starts from the top, and stops when at least one feasible reduced cost column is found. The order is the following:

1. Relax break resources and reduced arc set with $n = 5$
2. Relax break resources and reduced arc set with $n = 10$
3. Relax break resources
4. Full subproblem (only if 3. finds a non-feasible column with positive reduced cost)

For handling fractional solutions while solving the RMP, a Branch-and-Bound (BB) tree is implemented. A best-first strategy is implemented to ensure that a minimum number of nodes is investigated. Branching is performed on the most fractional arc, where the one branch requires the arc to be used, while the arc cannot be used in the second branch. For the PDPTW, Ropke and Cordeau (2009) noted that their dominance criterion requires the so-called delivery triangle inequality to be fulfilled, and this may no longer hold after branching decisions are added. They handled this by altering the arc cost matrix, and we have taken the same approach in our implementation.

5. Adaptive large neighborhood search heuristic

In this section we present an adaptive large neighborhood search heuristic for the PDPTDS. The heuristic follows the main framework presented by Ropke and Pisinger (2006), and use operators inspired by Ropke and Pisinger (2006) and Demir et al. (2012). In addition, we present several new operators tailored to the PDPTDS. In Section 5.1 we give an overview of the ALNS framework, before presenting the destroy and repair operators used in Section 5.2.

5.1. Overview of the Heuristic

An overview of the ALNS, using a set of destroy operators Γ and a set of repair operators Υ, is given in Algorithm 2. First, an initial solution S^0 is constructed by a greedy construction heuristic and set as the best known (S^B) and current solution (S), respectively. The initial weights of the removal and insertion operators, w^Γ and w^Υ are set, before setting the initial temperature T and the cooling rate c of the simulated annealing. As long as the stop-criterion is not met, the algorithm continues to iterate by destroying and repairing solutions. In each iteration of the algorithm, a removal operator, γ, and an insertion operator, υ, is drawn randomly based on the weighed probability of each operator w^Γ and w^Υ. The solution is altered by the removal and

Algorithm 2 ALNS

S^0 = greedyConstruction()
$S \leftarrow S^0, S^B \leftarrow S^0$
Initialize weights: w^Γ and w^Υ
Initialize cooling rate c
$T = -\frac{\tau \cdot f(S^0)}{\ln(0.5)}$
$k = 0$
while stop-criterion not met **do**
  $k = k + 1$
  Select removal $\gamma \in \Gamma$ and insertion $\upsilon \in \Upsilon$ operator using w^Γ and w^Υ
  $S^C = \upsilon(\gamma(S))$
  if $e^{\frac{f(S^C)-f(S)}{T}} > U(0,1)$ **then**
    $S \leftarrow S^C$
    update reward for γ and υ
    if $f(S^C) > f(S^B)$ **then**
      $S^B \leftarrow S^C$
    end if
  end if
  if $k \bmod \mu = 0$ **then**
    Update weights w^Γ and w^Υ
  end if
  $T = T \cdot c$
end while
Return S^B

insertion operators, and the solution is accepted according to a simulated annealing acceptance criterion. If the altered solution is accepted, a reward is given to the current removal and insertion operators based on if the solution is a new best-known solution, improving solution or a worse but accepted solution, respectively. Furthermore, the best-known solution is updated if the altered solution is a new best-known solution. Additionally, every μ-iteration, the weights of the operators are updated. Once the stopping criteria is met, the best solution found during the search, S^B is returned.

5.2. Operators

In this section, we present the removeal and insertion operators used in the sets Γ and Υ. In the following, whenever it is stated that a request is removed from the solution, both the pickup and delivery node are removed. Additionally, the insertion of a request indicates inserting the pickup and delivery node into the best route and position according to the operator's criterion.

The implemented removal operators are:

- **Random Removal** removes one random request from the solution and is implemented as described by Ropke and Pisinger (2006).
- **Route Removal** removes all requests in one random route and is implemented as described by Gschwind and Drexl (2019).

- **Shaw Removal** is implemented as described by Ropke and Pisinger (2006) and is based on a relatedness measure between requests. The relatedness measure is stated in Equation (11) and is a weighted sum of the shortest time between two pickup nodes and two delivery nodes, the difference in the start of time windows of the two requests, and the difference in demand in terms of weight and volume. The weights are given as $\alpha_{T^{min}}$, α_T, α_{D^W} and α_{D^V}, respectively. Following the implementation of Ropke and Pisinger (2006), a randomness factor, p_{shaw} is introduced to increase the exploration of the operator.

$$\begin{aligned} R(i,j) = \alpha_{T^{min}}(T_{ij}^{min} + T_{(i+n)(j+n)}^{min}) + \alpha_T(|\underline{T}_i - \underline{T}_j| + |\underline{T}_{i+n} - \underline{T}_{j+n}|) + \\ \alpha_{D^W}(|D_i^W - D_j^W|) + \alpha_{D^V}(|D_i^V - D_j^V|) \end{aligned} \tag{11}$$

  where

$$T_{ij}^{min} = T_{ij} + \left\lceil \frac{T_{ij}}{T^{drive|R}} - 1 \right\rceil T^{rest} \tag{12}$$

- **Shaw Removal Scheduling** is similar to the Shaw Removal operator. However, the relatedness measure is designed to exploit information from truck driver scheduling, and is calculated as stated in Equation (13). This states a weighted sum of the spatial difference between the requests and the difference in time of finished service at both pickup and delivery node. Furthermore, at the end of service, the relatedness in terms of normalized driving time since the last rest or break and the normalized elapsed time since the last rest or break is calculated. Requests which are similar in terms of these attributes are expected to be easily relocatable. A randomization factor, $p_{scheduling}$ is used to randomize the operator in a similar manner to the Shaw Removal operator.

$$\begin{aligned} R(i,j) = \alpha_1(D_{ij} + D_{i+N^P,j+N^P}) + \alpha_2((t_i^{time} - t_j^{time}) + (t_{i+n}^{time} - t_{j+n}^{time})) \\ \alpha_3\left(\frac{t_i^{drive|R} - t_j^{drive|R}}{T^{drive|R}}\right) + \alpha_4\left(\frac{t_i^{elapsed|R} - t_j^{elapsed|R}}{T^{elapsed|R}}\right) + \\ \alpha_5\left(\frac{t_i^{drive|B} - t_j^{drive|B}}{T^{drive|B}}\right) + \alpha_6\left(\frac{t_i^{elapsed|B} - t_j^{elapsed|B}}{T^{elapsed|B}}\right) \end{aligned} \tag{13}$$

- **Worst Removal** removes requests that give the highest difference in the total profit of a route with and without a specific request, and is implemented as described by Ropke and Pisinger (2006). As with Shaw Removal, a randomness factor, p_{worst}, is included in the removal process.

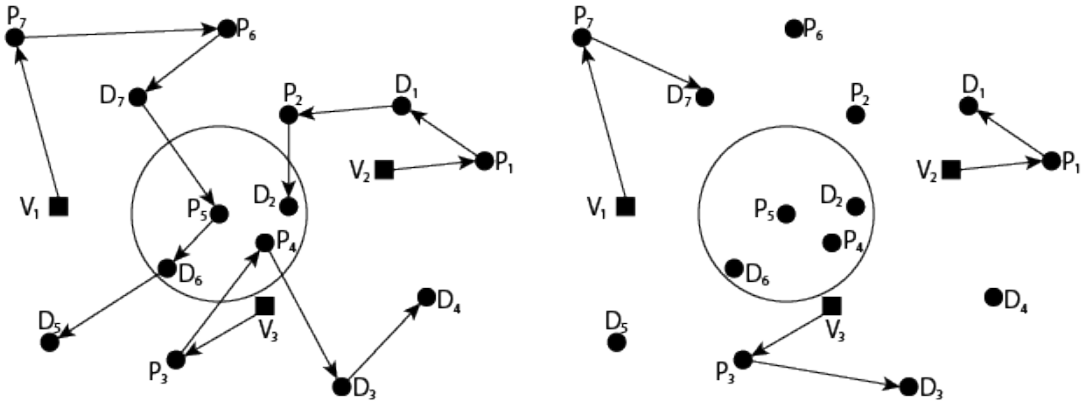
- **Outlier Removal** calculates the average time of a route, and removes the request which makes the total time of the route deviate the most from the average time when the request is removed from the solution. This is implemented following the Neighborhood Removal described by Demir et al. (2012) and is adjusted to a PDP structure.

- **Historical Knowledge Removal** removes requests with a position cost farthest from their optimal historical recorded position cost. Thus, removing requests with the presumed highest

negative impact on the solution. The position cost of request $i \in N^P$ is defined as stated in Equation (14) where $p(i)$ and $f(i)$ is the arc preceeding and following node i on the route, respectively. The implementation follows the description by Demir et al. (2012) and is adjusted to incorporate a PDP structure by adding the cost of both the pickup and delivery node. A randomization factor, $p_{historical}$ is used to randomize the operator similarly to the Shaw Removal operator.

$$c_i = C_{p(i)} + C_{f(i)} + C_{p(i+n)} + C_{f(i+n)} \tag{14}$$

- **Node Neighborhood Removal** chooses a random request and removes all requests with a pickup or delivery node within a given radius of the pickup node of the removed request. The radius is increased until more than or equal to a specific number of removals have been executed. This is implemented as described by Demir et al. (2012) and is adjusted to incorporate a PDP structure. An example is given in Figure 2.



(a) Before removal. (b) After removal.

Figure 2: Illustration of Node Neighborhood Removal. P_5 and all pickup and delivery nodes within a given radius from pickup node P_5 are removed from the solution.

In each removal operation, except Route Removal and Node Neighborhood Removal, a percentage in the interval $[\underline{\xi}, \overline{\xi}]$ of the requests handled in the current solution is removed.

The implemented insertion operators are:

- **Greedy Insertion** inserts requests, according to a list sorted on descending revenue, at the position yielding the largest increase in profit. This is repeated until no more insertions are profitable/feasible.

- **k-Regret** inserts the request with the largest accumulated difference between the best insertion and the $k-1$ best insertions. 2-regret, 3-regret, and $|\mathcal{V}|$- regret is used. The implementation of the operator follows the description given by Ropke and Pisinger (2006).

- **GRASP** operator (Feo and Resende, 1989) that builds a restricted candidate list holding the subset of feasible request/vehicle combinations that gives the largest increase (or smallest decrease) in profit. From this list, one request/vehicle combination is chosen at random and inserted into its best position. Two variants of the insertion is created: one only allowing profitable insertions, and one allowing both profitable and non-profitable insertions.

To add more randomization to the search, a noise parameter, κ, is implemented as described by Ropke and Pisinger (2006) for Greedy Insertion and k-regret. The insertion cost of a request is calculated as the increase in the objective value of inserting the request at its best position in the route according to the criterion of the insertion operator.

5.3. Scheduling Algorithm

To check that a route is respecting the hours of service regulations, Algorithm 1 is used on a modified network consisting of a path representation of the route, with the intermediate nodes (dull and fit) inserted between each node. Since, we only care about checking feasibility of the route with respect to the hours of service regulations, the resource vector is reduced to:

$$R(L) = [t^{time}(L), t^{dist}(L), t^{drive|R}(L), t^{elapsed|R}(L), t^{latest|R}(L), t^{drive|B}(L), t^{elapsed|B}(L), t^{latest|B}(L), T^{drive|W}(L), T^{work|W}(L)] \tag{15}$$

6. Instance Generation

This section describes the generation of the benchmark instances for the PDPTDS. The instances are generated from time and distance matrices between 132 locations in the southern, and central, parts of Norway (see Figure 3). In addition, clusters of locations (regions) are defined around some of the largest cities in Norway (Oslo, Bergen, Stavanger, Kristiansand and Trondheim). This is done to enable a balance between short- and long-distance haulage, to ensure that there is a flow of transportation requests out of the Oslo region as this is the hub for international import, and a net flow into the other large regions which are considered industry hubs. A cabinet volume of 96 cubic meters (m^3) and a maximum weight capacity of 30 tons are used for the vehicles (ColliCare). However, the latter may easily be changed to 25 tons for complying with the capacity restrictions of the European Union.

A transportation request specifies the pickup and delivery location of the cargo, the weight, and volume of the cargo, the time window for starting service at each location, the revenue received for transporting the cargo, and the time required for servicing at each location. The generation of pickup and delivery locations is done randomly based on a weighted probability distribution for the set of possible locations. We assume a positive correlation between the population of a area and the number of transportation requests into/out of that area, and the weighted probability of each location is therefore based on its proportion of the total population of all locations. Pickup

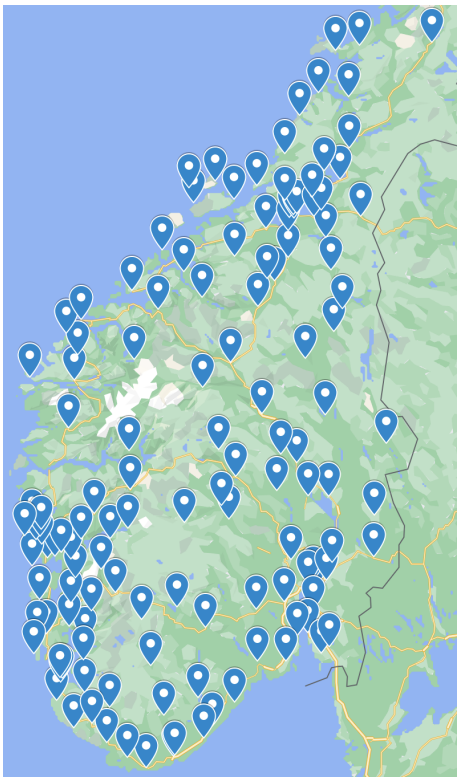


Figure 3: Overview of the 132 locations sampled from in the instance generation.

and delivery within the same region is allowed in 20 % of the requests, and the pickup and delivery location cannot be the same. To capture the role of the Oslo region as the main import hub of Norway, the population of all locations in the Oslo region is adjusted down by 30 % when calculating the probability of being a delivery location, while the population of the other regions is adjusted down by 30 % when calculating the probability of being a pickup location. Thus, the expected flow of transportation requests is expected out of the Oslo region, and into the other regions.

To create instances with different characteristics, different time window width is chosen for different subsets of instances. For a given instance all requests have time windows within the intervals, 12-24 hours, 24-48 hours, or 48-144 hours. A start time for the pickup location is randomly chosen from the interval 0-144 hours and is only accepted if it is possible to deliver the cargo to the delivery location within the end of the planning horizon. Furthermore, the time windows at the two locations must be of the required width. This is illustrated in Figure 4. Time windows are regenerated if the end of the time window at either the pickup or delivery location exceeds 144 hours. Furthermore, the time window start at the delivery location is set to the start time of the time window at the pickup location with the addition of the driving time between the locations.

The weight of the cargo of a transportation request is chosen in one of the two intervals, 1-10

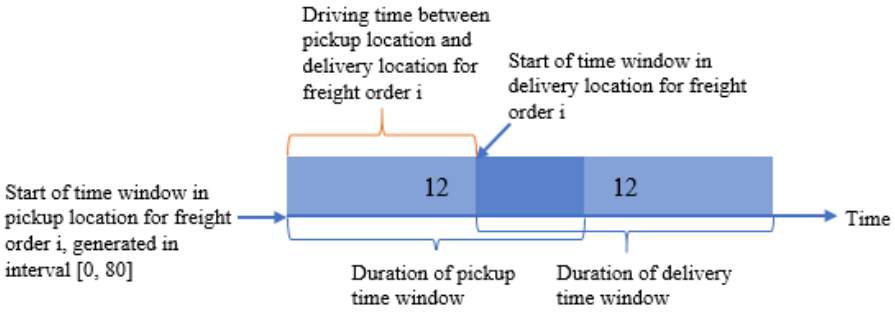


Figure 4: Generation of time windows of length 12 hours.

and 10-20 tons. Furthermore, the volume of the cargo is chosen randomly from an interval given as the weight of the cargo multiplied by 2 plus/minus 10. Furthermore, the service time at each location is given as 0.1 times the weight.

The cost of an arc C_{ij} is set to the total distance in kilometers between location i and j. The revenue of each cargo is then scaled based on the distance directly from pickup location i to delivery location $i+n$, $C_{i(i+n)}$ and, the weight of the cargo, D_i^W. Furthermore, a randomness factor, ϵ, on the interval $[0.9, 1.1]$ is included to randomize the revenue. A factor, h, is calculated as 1.5 divided by the expected cargo weight, $E[D^W]$, for the weight interval of the instance, and the profit function is given as stated in Equation (16). This profit function makes most transportation requests profitable to transport when going directly from pickup location to delivery location.

$$P_p = h \cdot D_p^W \cdot D_{pd} \cdot \epsilon \tag{16}$$

The number of available vehicles in a given instance is based on the number of transportation requests of the instance. For each number of transportation requests we create three vehicle fleets with $\lfloor \frac{|N^P|}{6} \rfloor$, $\frac{|N^P|}{5}$, and $\lceil \frac{|N^P|}{4} \rceil$ vehicles, respectively.

A dataset consists of all instances created when combining all numbers of transportation requests, time window intervals, weight intervals, and the number of vehicles. The number of transportation requests considered are 10, 15, 20, 25, 50, 75, 100, 150, 200. Each data set consists of 162 instances and a total of four data sets are created. An instance with 100 transportation requests, 20 vehicles, a time window interval of 12-24 hours, a weight interval of 1-10 tons from data set *1D* is denoted as *1D100R20V12-24T1-10W*. For each dataset, a pool of 200 transportation requests is created for every combination of the attributes. Thus, each instance with a lower number of requests consists of the first x requests, such that an instance having e.g. 150 transportation requests, the first 100 transportation requests are identical to an instance having 100 transportation requests when all other attributes are equal for the given dataset. The same is done for the starting location of each vehicle in each data set. The data sets can be found at http://axiomresearchproject.com/publications/.

7. Computational study

This section presents the results obtained by running an implementation of the BP algorithm presented in Section 4 and the ALNS algorithm presented in Section 5 on the benchmark instances described in Section 6. First, an overview of the results of the Branch-and-Price algorithm is presented in Section 7.1. Then, the parameter tuning and selection process for the ALNS is described in Section 7.2 while Section 7.3 analyzes the performance of the ALNS algorithm compared to the exact BP results. Section 7.4 elaborates on the operator performance. Finally, Section 7.5 investigates the difference between Full Truckload and Less-Than-Truckload transportation.

The RMP of the BP algorithm is solved using Gurobi 9.0, while both the labeling algorithm of the BP and the ALNS algorithm is implemented using Java 11. All computational experiments are performed on a computer having a 16 core *4x2.2GHz AMD Opteron 6274* CPU and 128 Gb of RAM while using *CentOS 7.9.2009*. Due to the random nature of the ALNS method, the algorithm is run ten times on each instance to facilitate analysis of the performance using average values. The runtime limit for solving each instance is set to 20 minutes (1200 seconds).Throughout the analysis, the optimality gap is calculated as the difference between the primal solution of the given method and the best known dual solution, divided by the best known primal solution. Detailed results of all test instances, can be found at http://axiomresearchproject.com/publications/.

7.1. Results from the Branch-and-Price algorithm

To test the Branch-and-Price method, and investigate the contribution of the pre-process techniques suggested in Section 4.5, we tested all instances with up to 50 transportation requests. All instances were given a maximum computing time of 7200 seconds. Table 3 shows the number of instances, out of 72, where the optimal solution (# opt.) or a dual bound (# DB) is found, and the average computing time (Avg. time) both when solving with and without the pre-processing techniques.

The results show that the pre-processing techniques greatly increases the number of instances we are able to solve within the time limit, and that the average computing time decreases substantially. We further see, that this difference increases with the number of requests in the instances. Overall, the BP is able to solve almost twice the number of instances within the time limit, and cut the average computing time in half, when using the pre-processing techniques. A final comment is that even with just 10 requests, the BP is unable to solve all instances within 2 hours. These are all instances that combines the lowest weight with the widest time windows.

Table 3: The number of instances with a given number of requests where the BP method can find an optimal solution (# opt.) and a dual bound (# DB), and the average computing time (Avg. time) used, with and without the preprocessing techniques from Section 4.5.

	Without preprocess			With preprocess		
# requests	**# opt.**	**# DB**	**Avg. time**	**# opt.**	**# DB**	**Avg. time**
10	63	63	1218.79	66	68	631.99
15	46	47	3063.39	60	60	1217.78
20	28	29	4575.96	58	60	1679.74
25	23	24	4989.11	58	60	1801.29
50	6	6	6873.35	31	37	4357.43
Total	166	169	4144.12	273	285	1937.64

To further investigate when the pre-processing techniques work well, we have aggregated the results based on the width of the time windows in Table 4. As we can see from the table, the pre-processing is extremely effective when the time windows are narrow, as it is able to solve all the instances with time window width between $12-24$ hours, and all but 7 instances with time window width in the range $24-48$ hours. The computing time difference is also considerable, going from close to 2700 seconds to 40 seconds for the former, and reducing the computing time to about one third for the latter time window width. For the widest time window range the improvement is less considerable, although the number of instances solved within the time limit is more than doubled. The results are in line with what one would expect, since narrow time windows increases the number of infeasible (partial) paths, and therefore we are able to both remove partial paths earlier, and increase the set of unreachable nodes.

Table 4: The number of instances with a given time window width where the BP method can find an optimal solution (# opt.) and a dual bound (# DB), and the average computing time (Avg. time) used, with and without the preprocessing techniques from Section 4.5.

	Without preprocess			With preprocess		
Time Windows	**# opt.**	**# DB**	**Avg. time**	**# opt.**	**# DB**	**Avg. time**
12-24	81	81	2695.85	120	120	40.23
24-48	61	63	3775.15	103	109	1236.03
48-144	24	25	5961.36	50	56	4536.68

7.2. Parameter Tuning and Selection for the ALNS

The ALNS algorithm is dependent on the following parameters: w_I, σ_1, σ_2, σ_3, ϕ, μ, p_{shaw}, p_{worst}, $p_{scheduling}$, $p_{neighborgraph}$, $\underline{\xi}$, $\overline{\xi}$, κ, c and τ. With such an exhaustive list of parameters, finding the best combination of parameter values using parameter tuning was considered too time consuming. Thus, the following parameters were set according to the implementation of Ropke and Pisinger (2006): $(w_I, \sigma_1, \sigma_2, \sigma_3, \mu, c, \kappa, p_{shaw}, p_{worst}) = (1, 33, 13, 9, 100, 0.99975, 0.025, 3, 6)$. However, Ropke and Pisinger (2006) used 9 for σ_2 and 13 for σ_3, i.e., finding a new and worse solution was rewarded more than finding a new and improving. Furthermore, based on trial-

and-error during development, $\underline{\xi}$, was set to 5 % while $p_{historical}$ was set equal to p_{worst}, i.e., 3. The remaining parameters were tuned in the following order based on the expected influence of the parameter: $\overline{\xi}$, ϕ, τ and $p_{scheduling}$, resulting in the following parameter values: ($\overline{\xi}$, ϕ, τ, $p_{scheduling}$) = (0.4, 0.075, 0.075, 4). The details of the parameter tuning phase is given in Appendix A.

7.3. ALNS Performance

The ALNS was tested with the operators described in Section 5. However, four variants of the Shaw removal and Shaw scheduling removal operators were used, as they behave quite differently with different weight settings. For Shaw removal four operators were created with different the weight settings ($\alpha_{T^{min}}$, α_T, α_{D^W}, α_{D^V}): (0.45, 0.45, 0.05, 0.05), (0.2, 0.2, 0.3, 0.3), (0.8, 0.2, 0.0, 0.0) and (0.2, 0.8, 0.0, 0.0), henceforth referred to as Shaw 1, Shaw 2, Shaw 3 and Shaw 4. Similarly Shaw Scheduling $1-4$ were created with the weights (α_1, α_2, α_3, α_4, α_5, α_6): (0.05, 0.2, 0.05, 0.2, 0.3, 0.2), (0.2, 0.05, 0.2, 0.05, 0.3, 0.2), (0.45, 0.0, 0.05, 0.0, 0.3, 0.2), (0.0, 0.45, 0.0, 0.05, 0.3, 0.2). Thus, the ALNS can find the best weights for a given problem instance.

Table 5 shows the results, aggregated by the number of transportation requests, of running the ALNS algorithm ten times on each instance in the instance sets 1–3. We have excluded instance set 4 from this part of the test as some of these instances were used for parameter tuning in Appendix A. The maximum, minimum and average optimality gap is found over all ten runs for each instance. Furthermore, the average of the maximum, minimum, and average optimality gaps is found over all instances with the same number of transportation requests. For instances with up to 25 transportation requests, the ALNS finds close to optimal solutions with an average optimality gap of less than 0.3 %. However, for 50 transportation requests, a major leap is seen as the average optimality gap increases to 3.01 %. This is probably due to the increase in solution space as the number of transportation requests increases. Nevertheless, the average minimum gap of 2.08 % indicates that the ALNS finds better solutions, although not consistently.

Table 5: Average maximum, minimum and average gap over all instances after 10 runs for each instance based on the number of transportation requests.

	10R	**15R**	**20R**	**25R**	**50R**
Minimum Gap (%)	0.12	0.00	0.00	0.10	2.08
Average Gap (%)	0.12	0.00	0.01	0.28	3.01
Maximum Gap (%)	0.12	0.00	0.02	0.46	3.92

For the instances with 75-200 requests, we do not have any dual bound to compare the ALNS results with, so we limit ourselves to report on the stability of the solutions provided. Table 6 shows the average coefficient of variation (CoV) over all instances with the same number of transportation requests. The table shows that for instances up to 20 requests, the ALNS (almost) always finds the same solution, thus there is virtually no variance for those instances. For larger instances the

results indicate that the ALNS performs consistently and, on average, the CoV in solution value is not above 1 % for any number of customers.

Table 6: Average coefficient of variation over instances having the same number of transportation requests.

	10R	**15R**	**20R**	**25R**	**50R**	**75R**	**100R**	**150R**	**200R**
CoV (%)	0.00	0.00	0.01	0.11	0.67	0.82	0.98	0.90	0.87

7.4. Operator Performance

Each operator's performance in the ALNS algorithm is essential for the overall performance of the search process, and the algorithm consists of 14 removal operators and 10 insertion operators. To perform the analysis of operator performance, two different measures are introduced. First, the number of times an operator finds a new best solution of the total number of new best solutions found during testing. Second, the number of times an operator finds a new best solution or an improving solution out of the total number of such solutions found during testing.

Table 7 shows the results for each removal operator on the two measures as well as the average time usage of the operator over all operations. Worst Removal is the best performing operator over both measures. Thus, removing requests contributing to a significant increase in the traveled distance seems to work well over the given test instances. Outlier Removal is the worst performer over both measures. When not considering the best and the worst performer, the rest of the removal operators perform similarly, and it may be concluded that all operators contribute to the search process. This includes the problem-specific Shaw Removal Scheduling, which removes requests based on time usage, the distance between requests, and scheduling-specific measures. However, the time usage of the operator is considerably higher than for the rest of the operators. This time increase is expected to be due to the dependency on the scheduling algorithm for creating labels holding information about the route schedule, which holds the requests to be removed. Finally, Random Removal performs well as it is believed to contribute to the exploration of the solution space.

Table 7: The average score of each removal operator based on the two performance measures. The best score for each measure is marked in bold. Furthermore, the average time usage of the operators is shown.

Operator	Best (%)	Best or Improving (%)	T (ms)
Random	6.91	7.49	0.00
Worst	**13.34**	**8.97**	0.03
Route	5.99	6.65	0.00
Node Neighborhood	7.55	5.54	0.02
Outlier	3.42	4.99	0.01
Historical Knowledge	7.39	7.18	0.03
Shaw 1	7.07	7.35	0.01
Shaw 2	7.15	7.32	0.01
Shaw 3	7.10	7.49	0.01
Shaw 4	6.97	7.43	0.01
Shaw Scheduling 1	6.66	7.37	0.29
Shaw Scheduling 2	6.72	7.37	0.29
Shaw Scheduling 3	6.86	7.42	0.29
Shaw Scheduling 4	6.87	7.43	0.29

For the insertion operators, Table 8 shows the performance of each operator on the two measures as well as the average time usage of each operator. Greedy Insertion is the best performance over both measures and is, together with Greedy Insertion with Noise, the least time-consuming operator. However, also 2-regret and 3-regret with and without noise perform well, although being more time-consuming. GRASP Negative is the worst performer over both measures. However, allowing for inserting requests giving negative profit facilitates the inclusion of combinations of requests, giving a positive profit, although each request contributes with a negative profit in isolation. Thus, although performing poorly, the operator is expected to be essential for certain combinations of routes otherwise not accounted for by the other operators. This is a problem that must be accounted for when considering optional requests. Furthermore, $\mathcal{V}$-regret is time-consuming and contributes significantly less in the search process as compared to Greedy, 2-regret, and 3-regret. As expected, the insertion operators are more time-consuming than the removal operators.

Table 8: The average score of each insertion operator on the two measures, and the average time usage of each insertion operator. The best score for each measure is marked in bold.

Operator	Best (%)	Best or Improving (%)	T (ms)
Greedy	**16.02**	**12.59**	0.25
Greedy with Noise	13.42	12.29	0.24
GRASP	5.02	8.88	0.36
GRASP Negative	1.07	1.15	1.82
2-Regret with Noise	12.35	12.05	0.82
2-Regret	12.72	11.37	0.85
3-Regret with Noise	10.48	11.32	1.31
3-Regret	12.60	11.21	1.38
v-regret with Noise	7.50	9.46	2.98
v-regret	8.82	9.67	3.03

Figure 5 and Figure 6 shows the development of selection probability of the removal and insertion operators when solving instance *3D100R16V24-48T10-20W*, respectively. As operator perfor-

mance is dependent on both randomness and the instance being solved, no conclusions can be made from the given plots. Nevertheless, note how Shaw Scheduling 2 contributes to the search process early on, while Worst Removal and Historical Knowledge Removal contribute later. For the latter, better performance is expected as it depends on the requests' historical position cost, which possibly becomes more accurate as the search proceeds. Node Neighborhood Removal performs poorly for the given instance, although being a good performer according to the two performance measures. For the insertion operators, the selection probability is more stable, and the well-performing operators for the given performance measures are also performing well for the given instance.

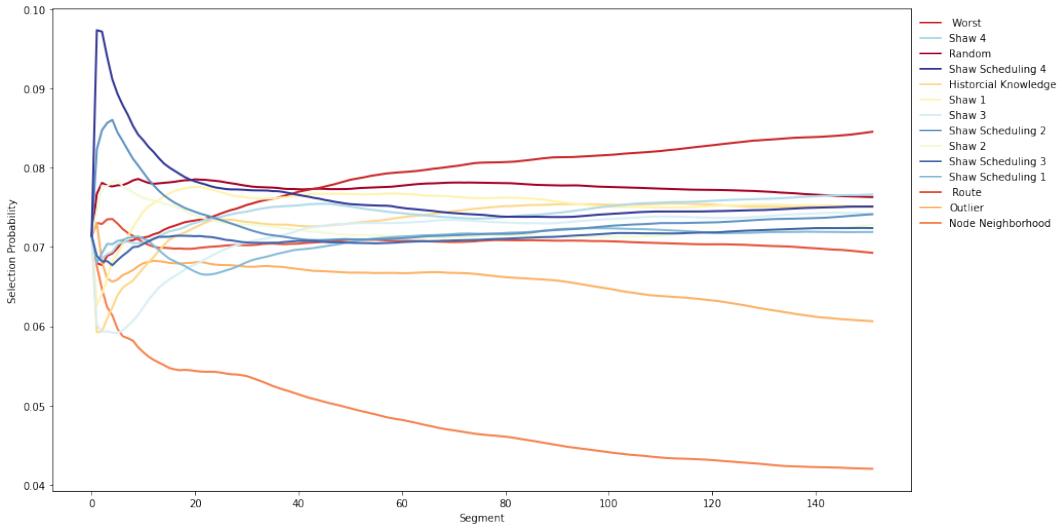


Figure 5: The development of the probability of selecting each of the removal operators as the search proceeds. The operator labels are sorted descending according to the last point on the line of their graph.

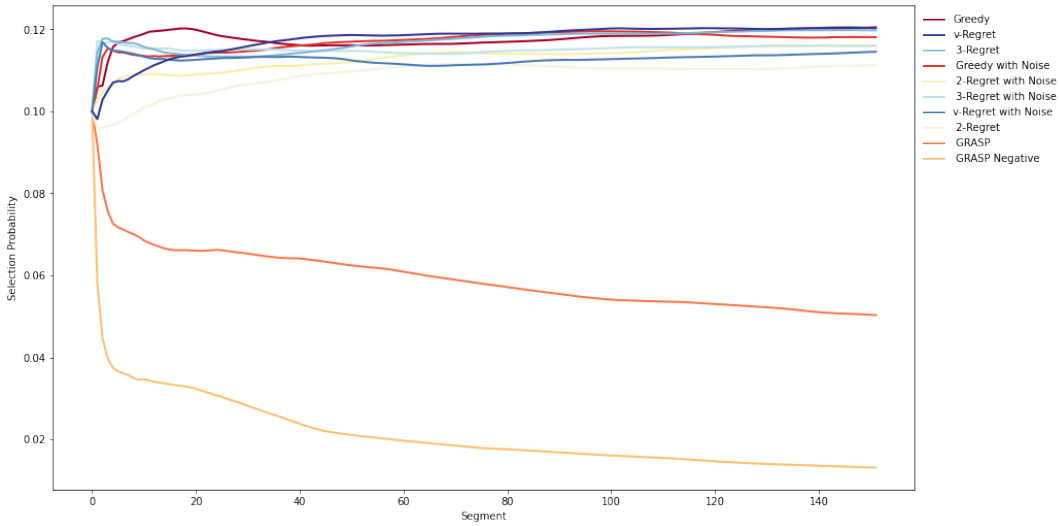


Figure 6: The development of the probability of selecting each of the insertion operators as the search proceeds. The operator labels are sorted descending according to the last point on the line of the graph.

7.5. Full Truckload vs Less-Than-Truckload

There are two main modes of operation within vehicle goods transportation (Hall, 2003): full truckload transportation (FT), where there is at most one transportation request onboard each vehicle at any time, and less-than-truckload transportation (LTT), where several request may be are carried at the same time in order to fill the truck capacity. From a theoretical point of view, it is clear that LTT leads to higher profits for the transportation company, as it is a relaxation of the FT mode of operation. A challenge with this, however, is that the transportation company will potentially move the goods of competing businesses together. Customers are then concerned that their rivals can reap the benefits of high capacity utilization at their expense. Many customers make contracts with transportation companies where they reserve an entire vehicle, although the cargo size is significantly less than a full truckload. This makes the transportation mode FT, although the size of the cargo give opportunities for LTT transportation. Such contracts lower the opportunity for transportation companies to achieve economies of scale.

It is therefore likely that a transportation company would need to reduce prices in order for customers to accept that their transported goods share a truck with those of other companies. In this section we investigate the difference in profits for a transportation company from operating their transportation network in FT and LTT mode, in order to see if the increase in profit is significant enough to allow for reduced freight rates to the customers.

To compare the profits of FT and LTT we have run all test instances presented in Section 6 in FT mode using the ALNS. In order to test the FT mode, we have modified the insertion operators so that it is only possible to insert the delivery node immediately after the pickup node. Tables 9 and 10 compare the average LTT and FT objective values, the number of handled requests, and the number of iterations of the ALNS, respectively, when each instance is run 10 times. The comparison is done by taking the average value of the LTT solution and dividing it by the corresponding value for the FT solution. Table 9 shows the difference aggregated according to the number of transportation requests, while Table 10 shows the differences aggregated according to the different weight and time window intervals.

As can be seen from Table 9, the differences both in terms of objective value and number of handled requests are quite stable across the number of requests. The objective value increases by roughly $50 - 70$ %, while the number of requests handled increase by $35 - 60$ %. The number of iterations are reported simply to show that the FT version of the ALNS have actually run two to four times as many iterations as the LTT, and thus it is likely that, if anything, the test results are skewed in favour of the FT version.

The results presented in Table 10 give some indication as to what characteristics of an instance leads to the biggest benefits from using LTT over FT. Unsurprisingly, small cargo sizes and wide time windows leads to a larger difference in both profit and the number of handled requests since

this allows for more combinations of transportation requests to be transported simultaneously.

Table 9: Comparison of objective value, handled requests, and iterations performed between LTT and FT with respect to the number of freight order requests.

	10R	**15R**	**20R**	**25R**	**50R**	**75R**	**100R**	**150R**	**200R**
Objective value (LTT/FT)	160.86%	168.69%	167.75%	152.19%	165.82%	162.85%	166.97%	170.79%	172.08%
Handled requests (LTT/FT)	160.76%	141.22%	143.23%	136.26%	137.85%	135.86%	139.31%	141.43%	145.17%
Iterations (LTT/FT)	50.03%	39.72%	33.96%	30.81%	25.41%	23.31%	22.73%	25.83%	27.30%

Table 10: Comparison of objective value, handled requests, and iterations performed between LTT and FT with respect to time window interval, weight interval, and overall.

	Overall	**1-10W**	**10-20W**	**12-24T**	**24-48T**	**48-144T**
Objective value (LTT/FT)	164.78%	174.20%	155.36%	133.49%	165.94%	194.91%
Handled requests (LTT/FT)	141.94%	151.45%	132.44%	132.54%	143.89%	149.41%

8. Concluding remarks

This paper has presented a new problem to the operations research community, where the vehicle routing problem with pickup and deliveries is combined with truck driver scheduling. The problem is relevant in many settings where third party transportation companies operate in long haul trucking. We present the problem, along with both an exact and heuristic solution method to solve it. The exact method is a branch-and-price (BP) method, where the subproblem is solved as a resource constrained shortest path problem. Several preprocessing techniques are proposed to accelerate the solution time. Further the heuristic solution method is an adaptive large neighborhood search (ALNS) where several new operators, tailored to this problem, is proposed.

We further present a set of 648 benchmark instances for the problem, based on the geography and road network of Norway. When testing the proposed methodologies on these instances we see that the BP method can solve instances with up to 50 transportation requests within a two hour time limit. We further see, that the proposed pre-processing techniques significantly improves the solution time and increases the number of instances solved from 166 to 273, while reducing the computing time by more than 50 %. The ALNS heuristic can find near optimal solutions to all instances solved to optimality, and solves instances with up to 200 requests within 20 minutes. Further analysis of the ALNS shows that the new operators proposed contributes to the search.

Finally, a comparison between full truckload, and less than truckload, mode of operation is studied. The results indicate that for long haul transportation companies there is a significant potential to increase profits by allowing a truck to handle several transportation requests simultaneously. The results further show, that the benefit of allowing this increases when the time windows are wide and/or the transported quantities are small.

References

Archetti, C., Savelsbergh, M., 2009. The trip scheduling problem. Transportation Science 43, 417–431.

Archetti, C., Speranza, M.G., Vigo, D., 2014. Chapter 10: Vehicle routing problems with profits, in: Vehicle Routing: Problems, Methods, and Applications, Second Edition. SIAM, pp. 273–297.

Baldacci, R., Bartolini, E., Mingozzi, A., 2011. An exact algorithm for the pickup and delivery problem with time windows. Operations research 59, 414–426.

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W., Vance, P.H., 1998. Branch-and-price: Column generation for solving huge integer programs. Operations research 46, 316–329.

Bent, R., Van Hentenryck, P., 2003. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows, in: International Conference on Principles and Practice of Constraint Programming, Springer. pp. 123–137.

Berbeglia, G., Cordeau, J.F., Gribkovskaia, I., Laporte, G., 2007. Static pickup and delivery problems: a classification scheme and survey. Top 15, 1–31.

Christiansen, M., Fagerholt, K., 2014. Chapter 13: Ship routing and scheduling in industrial and tramp shipping, in: Vehicle Routing: Problems, Methods, and Applications, Second Edition. SIAM, pp. 381–408.

ColliCare, . Lasteenheter for veitransport. URL: `https://www.collicare.no/kjekt-å-vite/lasteenheter/lasteenheter-veitransport`.

Cordeau, J.F., 2006. A branch-and-cut algorithm for the dial-a-ride problem. Operations Research 54, 573–586.

Cordeau, J.F., Laporte, G., Savelsbergh, M.W., Vigo, D., 2007. Chapter 6: Vehicle routing, in: Barnhart, C., Laporte, G. (Eds.), Transportation. Elsevier. volume 14 of *Handbooks in Operations Research and Management Science*, pp. 367 – 428.

Curtois, T., Landa-Silva, D., Qu, Y., Laesanklang, W., 2018. Large neighbourhood search with adaptive guided ejection search for the pickup and delivery problem with time windows. EURO Journal on Transportation and Logistics 7, 151–192.

Demir, E., Bektaş, T., Laporte, G., 2012. An adaptive large neighborhood search heuristic for the pollution-routing problem. European Journal of Operational Research 223, 346–359.

Desaulniers, G., Desrosiers, J., Solomon, M.M., Soumis, F., Villeneuve, D., et al., 1998. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems, in: Fleet management and logistics. Springer, pp. 57–93.

Desaulniers, G., Lessard, F., Hadjar, A., 2008. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. Transportation Science 42, 387–404.

Desrosiers, J., Dumas, Y., Solomon, M.M., Soumis, F., 1995. Time constrained routing and scheduling. Handbooks in operations research and management science 8, 35–139.

Doerner, K.F., Salazar-González, J.J., 2014. Chapter 7: Pickup-and-delivery problems for people transportation, in: Vehicle Routing: Problems, Methods, and Applications, Second Edition. SIAM, pp. 193–212.

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.

European Union, 2006. European union, 2006. regulation (ec) no 561/2006 of the european parliament and of the council of 15 march 2006 on the harmonisation of certain social legislation relating to road transport and amending council regulations (eec) no 3821/85 and (ec) no 2135/98 and repealing council regulation (eec) no 3820/85. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A02006R0561-20150302, last accessed June 8, 2021.

Feo, T.A., Resende, M.G., 1989. A probabilistic heuristic for a computationally difficult set covering problem. Operations research letters 8, 67–71.

Goel, A., 2009. Vehicle scheduling and routing with drivers' working hours. Transportation Science 43, 17–26.

Goel, A., 2010. Truck driver scheduling in the european union. Transportation Science 44, 429–441.

Goel, A., 2012. The minimum duration truck driver scheduling problem. EURO Journal on Transportation and Logistics 1, 285–306.

Goel, A., 2018. Legal aspects in road transport optimization in europe. Transportation research part E: logistics and transportation review 114, 144–162.

Goel, A., Archetti, C., Savelsbergh, M., 2012. Truck driver scheduling in australia. Computers & Operations Research 39, 1122–1132.

Goel, A., Irnich, S., 2017. An exact method for vehicle routing and truck driver scheduling problems. Transportation Science 51, 737–754.

Goel, A., Kok, L., 2012. Truck driver scheduling in the united states. Transportation science 46, 317–326.

Goel, A., Rousseau, L.M., 2012. Truck driver scheduling in canada. Journal of Scheduling 15, 783–799.

Goel, A., Vidal, T., 2014. Hours of service regulations in road freight transport: An optimization-based international assessment. Transportation science 48, 391–412.

Goel, A., Vidal, T., Kok, A.L., 2020. To team up or not: single versus team driving in european road freight transport. Flexible Services and Manufacturing Journal , 1–35.

Gschwind, T., Drexl, M., 2019. Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. Transportation Science 53, 480–491.

Gschwind, T., Irnich, S., Rothenbächer, A.K., Tilk, C., 2018. Bidirectional labeling in column-generation algorithms for pickup-and-delivery problems. European Journal of Operational Research 266, 521–530.

Hall, R.W., 2003. Supply Chains, in: Hall, R.W. (Ed.), Handbook of Transportation Science. Springer. chapter 15, pp. 561–597.

Homsi, G., Martinelli, R., Vidal, T., Fagerholt, K., 2020. Industrial and tramp ship routing problems: Closing the gap for real-scale instances. European Journal of Operational Research 283, 972–990.

Irnich, S., Desaulniers, G., 2005. Chapter 2: Shortest path problems with resource constraints, in: Column generation. Springer, pp. 33–65.

Koç, Ç., Jabali, O., Laporte, G., 2017. Long-haul vehicle routing and scheduling with idling options. Journal of the operational research society , 1–13.

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, 442–454.

Li, F., Golden, B., Wasil, E., 2007. The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. Computers & operations research 34, 2918–2930.

Li, H., Lim, A., 2003. A metaheuristic for the pickup and delivery problem with time windows. International Journal on Artificial Intelligence Tools 12, 173–186.

Lübbecke, M.E., Desrosiers, J., 2005. Selected topics in column generation. Operations research 53, 1007–1023.

Montoya-Torres, J.R., Franco, J.L., Isaza, S.N., Jiménez, H.F., Herazo-Padilla, N., 2015. A literature review on the vehicle routing problem with multiple depots. Computers & Industrial Engineering 79, 115–129.

Nanry, W.P., Barnes, J.W., 2000. Solving the pickup and delivery problem with time windows using reactive tabu search. Transportation Research Part B: Methodological 34, 107–121.

Prescott-Gagnon, E., Desaulniers, G., Drexl, M., Rousseau, L.M., 2010. European driver rules in vehicle routing with time windows. Transportation Science 44, 455–473.

Rancourt, M.E., Cordeau, J.F., Laporte, G., 2013. Long-haul vehicle routing and scheduling with working hour rules. Transportation Science 47, 81–107.

Ropke, S., Cordeau, J.F., 2009. Branch and cut and price for the pickup and delivery problem with time windows. Transportation Science 43, 267–286.

Ropke, S., Cordeau, J.F., Laporte, G., 2007. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. Networks: An International Journal 49, 258–272.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation science 40, 455–472.

Tilk, C., Goel, A., 2020. Bidirectional labeling for solving vehicle routing and truck driver scheduling problems. European Journal of Operational Research 283, 108–124.

Vidal, T., Crainic, T.G., Gendreau, M., Lahrichi, N., Rei, W., 2012. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Operations Research 60, 611–624.

Zäpfel, G., Bögl, M., 2008. Multi-period vehicle routing and crew scheduling with outsourcing options. International Journal of Production Economics 113, 980–996.

Appendix A. Parameter tuning for the ALNS

The Parameter tuning was performed on 12 instances from a $4D$ dataset, using an average normalized objective value. First, the average objective value over five runs on each instance for a given parameter value is calculated. This is performed for all parameter values for a given parameter. Second, the maximum and minimum average objective value of the instance when varying the parameter value is found and utilized to normalize the averages for the given instance over all parameter values of the parameter. Finally, the average over all normalized averages is calculated for each parameter value, and the parameter yielding the highest normalized average becomes the chosen parameter value.

With the minimal removal percentage, $\underline{\xi}$, set to 5 % based on trial-and-error, the maximal removal percentage, $\overline{\xi}$, is tuned according to the tuning strategy. It determines the upper limit of removals as a given percentage of the number of transportation requests currently transported in the solution. The maximal removal percentage should not be set too high as Ropke and Pisinger (2006) indicate that insertion operators are poor when construction solutions from nothing. Consequently, values ranging from 10% to 50% are tested for $\overline{\xi}$ and the normalized average objective values from the tuning process are shown in Table A.11. With an average normalized objective value of 68.63 %, a parameter value of 40 % is shown to perform best on the test instances and is thus chosen as the parameter value.

Table A.11: Normalized average of the objective value as the value of the maximal removal percentage, ξ, varies.

Parameter Value (%)	10	20	30	40	50
Normalized Average (%)	39.54	38.60	57.74	68.63	64.74

The balance between the historical and the current performance over the last segment when updating the weight of an operator is given by the momentum weight, ϕ. Thus, increasing the value of momentum weight increases the emphasis on the current performance over the historical performance of the operator. The parameter is tested for values ranging from 2.5 % to 12.5 %, and the results from Table A.12 indicate that a momentum weight of 7.5 % gives the best performance of the algorithm on the given test instances. Thus, this value is chosen as the final parameter value.

Table A.12: Normalized average of the objective value as the value of the momentum weight, ϕ, varies.

Parameter Value (%)	2.5	5.0	7.5	10.0	12.5
Normalized Average (%)	35.55	33.87	77.14	60.27	39.71

The parameter, τ, is the temperature control used to set the start temperature such that a τ% worse solution than the current solution is accepted with a probability of 50 %. A τ equal to 0

% indicates that simulated annealing only accepts improving solutions. The normalized average objective value over the parameter values 2.5%, 5%, 7.5%, 10.0% and 12.5% is shown in Table A.13, respectively. For the test instances, a parameter value of 7.5 % gives the best solutions, while also a parameter value of 2.5 % provides good solutions. Although achieving good performance with both parameter values, a value of 7.5 % is chosen due to achieving the best normalized average, and that it is expected that 2.5 % may give too little exploration of the search space.

Table A.13: Normalized average of the objective value as the value of the temperature control, τ, varies.

Parameter Value (%)	2.5	5.0	7.5	10.0	12.5
Normalized Average (%)	60.19	49.69	61.25	37.34	55.37.

Finally, the randomness of the Shaw Removal Scheduling operator is determined by the parameter $p_{scheduling}$ where a value of 1 gives an entirely random operator, and a higher value leads to less randomness. Table A.14 shows the normalized average objective values for the parameter tuning process over the values 3, 4, 5, 6, and 7, and the results indicate that a value of 4 achieves the best results on the test instances. Thus, $p_{scheduling}$ is set to 4.

Table A.14: Normalized average of the objective value as the value of the randomness in Shaw Removal Scheduling, $p_{scheduling}$, varies.

Parameter Value	3	4	5	6	7
Normalized Average (%)	45.43	64.43	51.84	40.94	41.12