AUTHOR(S): REGNIER-COUDERT, O., MCCALL, J., AYODELE, M. and ANDERSON, S.

TITLE: Truck and trailer scheduling in a real world, dynamic and heterogeneous context.

YEAR: 2016

Publisher citation: REGNIER-COUDERT, O., MCCALL, J., AYODELE, M. and ANDERSON, S. 2016. Truck and trailer scheduling in a real world, dynamic and heterogeneous context. Transportation research part E: logistics and transportation review [online], 93, pages 389-408. Available from: http://dx.doi.org/10.1016/j.tre.2016.06.010

OpenAIR citation: REGNIER-COUDERT, O., MCCALL, J., AYODELE, M. and ANDERSON, S. 2016. Truck and trailer scheduling in a real world, dynamic and heterogeneous context. Transportation research part E: logistics and transportation review, 93, pages 389-408. Held on OpenAIR [online]. Available from: https://openair.rgu.ac.uk

Truck and Trailer Scheduling in a Real World, Dynamic and Heterogeneous Context

Olivier Regnier-Coudert[a], John McCall[a], Mayowa Ayodele[a], Steven Anderson[b]

[a]*Robert Gordon University, Aberdeen, Scotland*
[b]*ARR Craib Ltd., Aberdeen, Scotland*

Abstract

We present a new variant of the Vehicle Routing Problem based on a real industrial scenario. This VRP is dynamic and heavily constrained and uses time-windows, a heterogeneous vehicle fleet and multiple types of job. A constructive solver is developed and tested using dynamic simulation of real-world data from a leading Scottish haulier. Our experiments establish the efficiency and reliability of the method for this problem. Additionally, a methodology for evaluating policy changes through simulation is presented, showing that our technique supports operations and management. We establish that fleet size can be reduced or more jobs handled by the company.

Keywords: Logistics, Decision support systems, Scheduling, Heuristics, Simulation, Transportation

1. Introduction

Over the last few decades, an increasing number of studies focusing on optimization problems has arisen from the field of logistics, particularly with respect to vehicle routing and scheduling as recently reviewed in Berbeglia et al. (2007, 2010), Laporte (2009) and Schorpp (2011). This increasing interest is driven by the difficulties experienced on a daily basis by logistic companies such as the high numbers of tasks to be performed and dynamically-varying job information.

In this paper, we present a dynamic vehicle scheduling problem derived from the operations of ARR Craib Ltd. (ARRC), a UK haulage company based around Aberdeen harbour, the largest hub for oil and gas in Europe. The problem consists of assigning trucks and trailers to a set of jobs subject to a variety of constraints. Jobs can be received at any time during the day, so the problem must be repeatedly solved dynamically and in real time. The problem includes jobs with time windows, multiple customers, many types of job, and the use of rigid trucks, non-rigid units and trailers that may be dissociated. Use of subcontracted resources is permitted but costly to the company.

Instances of the Vehicle Routing Problem (VRP) (Golden et al., 2008) studied in the literature consider subsets of these problem features, however none of them deals with all features simultaneously. Therefore, the problem presented in this paper differs from previous research. Moreover, since this problem is common to haulage operations around industrial harbours across the globe, we argue that its study is of broad practical significance.

Hence, this paper contributes to the field of operational optimization by *a*) describing a new variant of VRP; *b*) explaining its relationship to existing VRPs; *c*) presenting a lightweight constructive solver capable of real-time dynamic optimization; *d*) assessing the operational impact of the optimized schedules for a large haulage company; and *e*) presenting a methodology for company policy-making through simulation.

The paper is organized as follows. In Section 2, the problem is described in detail and a mathematical formulation is established. In Section 3, background on vehicle routing problems is provided with reference to related work in the literature. Section 4 describes our proposed constructive solver. Experiments and results are presented and discussed in Section 5. Finally, conclusions and directions for further research are presented in Section 6.

2. Problem description

2.1. General description

ARRC is a leading truck haulage company based in Aberdeen, operating the largest haulage fleet in Scotland and generating an annual turnover of around £35m. Its activities are organized into two main areas: local and long-haul jobs, each managed from a separate office. Long-haul jobs involve transportation of goods across the UK whilst local jobs are all carried out within a fixed perimeter around Aberdeen. Local jobs present a significant scheduling and routing challenge compared with long-haul. Local jobs turn over quickly with trucks performing several jobs within a single day. Customer calls arrive throughout the day, and job length can be significantly affected by various delays, so the scheduling and routing problem varies dynamically throughout the day. This paper focusses exclusively on the problem of handling local jobs on a daily basis.

ARRC local customers are from a variety of industries. However, being located in Aberdeen, the vast bulk of the goods being transported are for the oil and gas industry. The local office of ARRC has an operations team of four members in charge of allocating trucks and trailers to the different jobs. The general policy of ARRC is to accept all jobs from their customers and to meet all deadlines. In order to make this possible, ARRC is able to subcontract resources from alternative haulage companies when needed. Naturally, this comes at a cost and so a general objective of operators at ARRC is to keep the number of subcontracted trucks, trailers and drivers as low as possible.

A key challenge is that jobs can be received or updated at any time. The level of dynamism is highly variable because the activity varies significantly between days. Some jobs require immediate action on notification while others are known, and can be planned, days in advance. For a more precise analysis, we refer the reader to Section 5, where metrics from the literature are applied to several real datasets obtained from ARRC.

A number of different job types exist. First, *ad-hoc* jobs are standard pick-up-and-drop jobs that do not require a change of trailer. Ad-hoc jobs may run between any pair of locations as required by the customer. Any type of truck and trailer may be used for ad-hoc jobs subject to customer requirements and internal policies.

Outbound shipping jobs are those transporting goods to the dock to be loaded onto vessels prior to their departure from the harbour to offshore oil installations. Outbound shipping jobs can only be performed by flatbed trailers, may be collected from any location and are always delivered to a quayside. Loads arriving at quaysides must wait to be loaded onto vessels by independent quayside workers. Hence, loaded trailers arriving at a quayside are detached and left there. Thus a truck that completes an outbound shipping job will detach its trailer.

Finally, *inbound shipping* jobs transport goods from the quayside to their final onshore destination. Inbound shipping jobs are always collected at a quayside location and can be delivered to any other location. Similarly to outbound shipping jobs, cargos are managed by independent quayside workers who load them onto empty trailers that can then be picked up by trucks. Thus, only a specific trailer type can be considered. A truck assigned to an inbound shipping job will attach a loaded trailer on collection. This may require detachment of an existing attached trailer. Inbound shipping jobs may be communicated by the quayside workforce at any time during the day.

The different types of job are allocated together and it is important that any compatible resource can be used for any job. There are approximately 100 vehicles and 150 trailers in the fleet, although numbers vary on a daily basis due to maintenance or unavailability of drivers. Trailers have different capacities and some vehicles are rigid and cannot be detached from their trailers (e.g. vans). Since the fleet is heterogeneous, some loads can only be handled by trailers of specific capacity. It may be acceptable to use trailers that are oversized for some jobs in order to meet customer deadlines. For example, a job defined for a van but performed by a 15-ton truck is preferable to hiring a subcontracted van. At the same time, operators normally wish to avoid using extremely large trailers for small loads. Using a flatbed trailer to perform the job of a van is a sign of misused resources. Table 1 shows a limited number of acceptable substitutions to perform a job for a specified type of trailer. In addition, Table 1 lists all types of trailers ranked by increasing capacity. Those trailer types that are not dissociable are indicated by an asterisk.

Due to the hazardous nature of the goods being transported, some jobs require specialized handling. Job allocations must comply with the European Agreement on international carriage of Dangerous goods by Road (ADR)[1]. The ADR scheme defines nine types of special goods, including for instance explosives, gases or radioactive materials.

[1] More details on ADR can be found on the Health and Safety Executive (HSE) website at http://www.hse.gov.uk/cdg/manual/classification.htm (accessed September 2013)

Table 1: Trailer type compatibility. The table shows the type of trailers that can be allocated to a job based on its required trailer type. For example, a job requiring a van could be performed by a van, a 5-ton or a 12-ton. Those trailer types that are not dissociable are indicated by a * symbols.

Requested \ Substitute	*Van**	*5-ton**	*12-ton**	*18-ton**	*26-ton**	*Flatbed*	*Extender*	*Step Frame*	*Tautliner*	*Skeletal*
Van*	✓	✓	✓							
5-ton*		✓	✓	✓	✓					
12-ton*			✓	✓	✓					
18-ton*				✓	✓	✓				
26-ton*					✓	✓				
Flatbed						✓				
Extender							✓			
Step Frame								✓		
Tautliner									✓	
Skeletal										✓

Thus, a job can only be allocated to a vehicle if its driver is trained for all of the ADR levels specified in the job description. Additional abilities may be required by specific jobs such as airport training. In this study, we consider a total of 10 different abilities, that is the 9 ADR levels and airport training.

The objectives of the study are to explore the benefits of automated scheduling and routing for this problem, in comparison to the existing scheduling at ARRC. The company is interested in understanding its fleet usage in order to reduce the number of subcontracted resources being hired and / or to increase the amount of business being undertaken. The next subsection gives a mathematical formulation of the problem and its constraints.

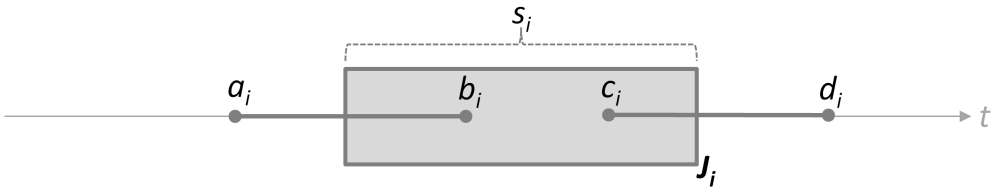


Figure 1: Representation of a job J_i and its time attributes

2.2. Mathematical formulation

In this section, the truck operations control problem is formalized.

Let $J = \{J_1, ..., J_n\}$ denote the set of all n jobs to be scheduled. Let $\lambda_i = 1$ if J_i is an outbound job, 0 otherwise. Similarly let $\mu_i = 1$ if J_i is an inbound job, 0 otherwise. Let ρ_i be the index of the loaded trailer of job J_i when $\mu_i = 1$, -1 otherwise.

Let T denote the set of all trailers and let $w_k = 0$ if T_k belongs to the company and 1 if T_k is a subcontractor trailer. Similarly, let S denote the set of all trucks and let $z_l = 0$ if S_l belongs to the company and 1 if S_l is a subcontractor truck. Let $y_l = 1$ if S_l is rigid, 0 otherwise and let τ_l be the index of the trailer associated with truck S_l if $y_l = 1$. $\tau_l = 1$ if $y_l = 0$.

For each job J_i we define its earliest collection time a_i, latest collection time b_i, its earliest delivery time c_i, its latest delivery time d_i and its execution time s_i as summarized in Figure 1. The company operates within a time interval represented by α and β, its opening and closing time respectively. The travel time between two successive jobs J_i and J_j is represented as θ_{ij}. The travel time between the original location of a trailer T_k and a job J_i is denoted as η_{ik} and the travel time between the original locations of a truck S_l and a trailer T_k as ϵ_{lk}.

More notation is introduced to represent the driver ability and trailer compatibility constraints. Let E be the set of all driver abilities. $\Delta_{iq} = 1$ if ability E_q is required for J_i, 0 otherwise. Similarly, let $\Gamma_{lq} = 1$ if the driver of the truck S_l has ability E_q, 0 otherwise. Finally, let $\varphi_{ik} = 1$ if trailer T_k can perform job J_i.

The following decision variables are defined. $A_{ik} = 1$ if trailer T_k is allocated to J_i, 0 otherwise. $B_{il} = 1$ if truck S_l is allocated to J_i, 0 otherwise. t_i represents the time job J_i starts.

The main objective of the optimization is to determine a set of allocations that minimizes the number of subcontracted jobs, that is jobs that are either performed by a subcontracted truck or a subcontracted trailer. Therefore we formulate the objective as follows:

Minimize:

$$\sum_{i=1}^{n} max(\sum_{k=1}^{|T|} A_{ik} w_k, \sum_{l=1}^{|S|} B_{il} z_l) \tag{1}$$

Subject to:

$$\sum_{k=1}^{|T|} A_{ik} = 1, \forall i \in J \tag{2}$$

$$\sum_{l=1}^{|S|} B_{il} = 1, \forall i \in J \tag{3}$$

$$a_i \leq t_i \leq b_i, \forall i \in J \tag{4}$$

$$c_i \leq t_i + s_i \leq d_i, \forall i \in J \tag{5}$$

$$\alpha \leq t_i \leq \beta - s_i, \forall i \in J \tag{6}$$

$$t_i + s_i + \theta_{ij} - t_j \leq (2 - A_{ik} - A_{jk})M, \forall i \neq j \in J, t_i < t_j, \forall k \in T \tag{7}$$

$$t_i + s_i + \theta_{ij} - t_j \leq (2 - B_{il} - B_{jl})M, \forall i \neq j \in J, t_i < t_j, \forall l \in S \tag{8}$$

$$\epsilon_{lk} + \eta_{ik} - t_i \leq (2 - A_{ik} - B_{il})M, \forall i \in J, \forall k \in T, \forall l \in S \tag{9}$$

$$B_{il}\Delta_{iq} \leq \Gamma_{lq}, \forall i \in J, \forall l \in S, \forall q \in E \tag{10}$$

$$A_{ik} - \varphi_{ik} \leq 0, \forall i \in J, \forall k \in T \tag{11}$$

$$t_i - \lambda_i t_j > (2 - A_{ik} - A_{jk})M, \forall i \neq j \in J, \forall k \in T \tag{12}$$

$$t_i - \mu_i t_j < (2 - A_{ik} - A_{jk})M, \forall i \neq j \in J, \forall k \in T \tag{13}$$

$$\mu_i(A_{ik} - A_{\rho_i}) = 0, \forall i \in J, \forall k \in T \tag{14}$$

$$B_{il} y_l \leq A_{i\tau_l} y_l, \forall i \in J, \forall l \in S \tag{15}$$

$$[J, S, T, E \subset \mathbb{Z}] \tag{16}$$

Constraints (2) and (3) state that a job has to be performed by one truck exactly and one trailer exactly.

Constraints (4) and (5) show that hard time windows are used to enforce the company policy that all jobs are collected and delivered on time. Constraint (6) states that all jobs are performed within day boundaries. Constraints

(7) and (8) ensure respectively that a given trailer can only perform one job at a given time and that a given truck can only perform one job at a given time. Constraint (9) ensures that enough time is allowed for moving trucks and trailers from their original positions to their first jobs.

Constraint (10) requires that the allocated resources match the job-specific constraints while constraint (11) guarantees that all jobs are performed by trailers compatible with the specified size.

Constraint (12) states that a trailer used for an outbound shipping job will not be available for other jobs following delivery and (13) states that a trailer with an inbound shipping job cannot perform any job prior to the inbound shipping job. Constraint (14) ensures that only the trailer loaded with an inbound job can perform this job and constraint (15) requires that all jobs performed by a rigid truck be also performed by its associated trailer. Finally, constraint (16) states that jobs, trucks, trailers and driver abilities are represented by their indices and so are integers. Note that in equations (7),(8),(9),(12) and (13), M is a sufficiently large constant whose value is set as follows:

$$M > \max\left(\max_{i,j\in J}(t_i + s_i + \theta_{ij}), \max_{i\in J,k\in T,l\in S}(\epsilon_{lk} + \eta_{ik})\right) \tag{17}$$

3. Related Work

VRPs and more precisely Pickup and Delivery Problems (PDPs), where entities need to be moved between two locations, are widely studied in the literature. Despite this, real-world applications often exhibit unique characteristics that motivate variants of PDPs to be defined and optimized. Because of the wide range of PDPs, effort has been put into classifying them, for example in (Berbeglia et al., 2007). In this study, the authors classify PDPs according to three criteria: the structure of the problem; the types of location visits and the number of vehicles available. Berbeglia et al. (2007) defines three types of structure, that are often cited in other works. The structure of PDPs can be one-to-one, one-to-many-to-one or many-to-many. In one-to-one problems, entities to transport are defined by single collection and delivery locations. Since one-to-one PDPs are the most common problems, there exist many applications, ranging from parcel delivery (Savelsbergh & Sol, 1998) to truck scheduling (Zhang et al., 2010). In one-to-many-to-one problems, entities can either be moved from a central location such as a depot to a range of locations or from the range of locations to the central location. In other words, the movement of an entity includes a central location precisely once. This structure has been used to represent real-world problems such as vessel scheduling for oil and gas platform servicing (Gribkovskaia et al., 2007). Finally, many-to-many problems consider that any request from any location can be handled by any other location. Based on the aforementioned categories, the problem faced at ARRC can be described as having a one-to-one structure, where each request needs to be served from specific collection and delivery points.

Within the classification of Berbeglia et al. (2007), there exist more specific families of PDP with one-to-one structures. The general problem is called the Vehicle Routing Problem with Pickups and Deliveries (VRPPD). A distinction is made when the entities to be transported are humans. In such cases, the problem is labeled as the Dial-A-Ride Problem (DARP) (Cordeau & Laporte, 2007), which differs from VRPPD in the sense that passenger comfort is also taken into consideration, as a constraint or in the objective function. Finally, there exists another distinct type of VRPPD, called the Stacker Crane Problem (SCP) (Frederickson et al., 1976). An instance of SCP is better described as a graph where nodes represent locations and directed edges between those nodes represent requests. Hence, a solution is a sequence of directed edges representing portions of a tour. Some of these edges are required in the solution while others are optional and only used in order for the solution to be a tour. SCP is also considered as a special case of Travelling Salesman Problem in which the aim is to minimize the empty travelled distance, modelled by means of optional edges. The ARRC problem belongs to the VRPPD category.

VRPPD has been approached in many ways. However, it is reasonable to split previous studies between those using Linear Programming (LP) to represent the problem and those that exploit advantages of heuristic search (Cordeau et al., 2008). Studies based on LP include Dumas et al. (1991), Savelsbergh & Sol (1998), Xu et al. (2003), Ropke & Cordeau (2009) and we refer the reader to these works for further details on LP. Within this literature, a common strategy is to use two-stage algorithms, in which a route construction stage is followed by a refinement stage. Simple insertion mechanisms are often used for the construction stage, while recombination is used to refine the solutions obtained (Nanry & Wesley Barnes, 2000; Mitrović-Minić et al., 2004; Mitrović-Minić & Laporte, 2004).

In addition to the general VRPs and more particularly VRPPDs, there exist many studies where additional constraints have been introduced in order to match some real-world contexts. These constraints do not affect the general problem definition but affect the number of feasible solutions. One of the most common special cases of VRPPD is the use of time windows and denoted as TW (Savelsbergh, 1985). In VRPPD-TW, the collection and delivery times are not always exactly known, but a time window for both is defined. An example of VRPPD-TW can be found in Mitrović-Minić & Laporte (2004). In the capacitated VRP, information is known about the physical features of items to be transported and about the maximum capacity of all available vehicles. Thus, jobs can only be assigned to a particular vehicle if the dimension of their loads is suitable for the vehicle (Ralphs et al., 2003). The possibility of splitting the load of a given job has also been investigated (Nowak et al., 2008). In Zhang et al. (2014), a dynamic PDP-TW involving containers is presented. Vertices representing deliveries and collections are ordered and their time windows discretized with specific granularity. Solutions are then created by picking one point from each job's time window. The authors show that their strategy is robust and that reasonable differences between predicted and actual travel times do not impact the optimized schedules.

Another important characteristic of VRP instances is the way in which vehicle capacity is handled. Whilst the general VRPPD has the concept of capacity allowing several loads to be handled by a single vehicle simultaneously, this is not the case with full-truckload problems. For these problems, a vehicle may only handle one load at a given time and the problem becomes a special case of the Travelling Salesman Problem with Time Window called the multiple TSP-TW (m-TSP-TW) (Savelsbergh & Sol, 1995). Several studies have focused on this variant (Jula et al., 2005; Zhang et al., 2010). However, despite some common features, the problem at hand does not contain the concept of a depot and, moreover contains different types of jobs and vehicles. Therefore our current problem is not an instance of m-TSP-TW.

Most of the aforementioned PDP variants consider only the scheduling of single-entity vehicles. In the Truck and Trailer Routing Problem (TTRP), the notion of trailers is introduced and dissociating trailers from trucks is made possible. In TTRP, first formalized in Chao et al. (2002), a set of requests needs to be handled that serves a set of locations. Locations can be of different types, namely those that can be served using a truck and an attached trailer, and those that can be served by truck only. Additionally, jobs can be combined on trailers in order to minimize the number of resources needed. These properties of the problem result in the creation of routes for trucks composed of route legs where a trailer is used and route legs where the truck only is used. Several studies have led to improvements to TTRP, the development of variations of the problem and applications to real-world cases as reviewed in Villegas et al. (2013). Because of its strict definition, only a few real-world applications fall into the TTRP category. These are mainly related to delivery tasks in rural areas (Semet & Taillard, 1993).

The remaining characteristic of VRPs to be considered in this section is dynamism. Routing and scheduling problems can be considered either static or dynamic. Dynamic problems arise from applications where the state of the information changes over time, for example where traffic information is considered. Dynamism may also be linked to the way jobs are received. In static problems, all jobs are known prior to starting the optimization process while in dynamic problems, jobs can be added at anytime. Hence, in a static problem, the aim of the optimization is to produce one schedule. However, in a dynamic problem, the aim is to produce a series of schedules over time. Because dynamism can vary considerably across problems, metrics have been developed that measure the dynamism present in any particular problem. Although there are a wide range of existing metrics, we focus here on two of them that allow comparison of instances of the problem at hand. The most basic metric for dynamism is called the degree of dynamism (*dod*) (Lund et al., 1996). *dod* measures the ratio of dynamic job requests R_d over all job requests R and is defined as in (18).

$$dod = \frac{|R_d|}{|R|} \tag{18}$$

In order to calculate *dod*, it is necessary to define what a dynamic request is. A request may be considered as dynamic if received at short notice. Often this means on the day preceding the solver run. Here we define dynamism according to the problem under study in order to capture those job requests that are received at short notice. In current practice at ARRC, controllers only start considering requests four hours before their collection time. Because of this, we define a concept of *dod* based on a short-notice threshold. We denote this as dod_t, where t is a threshold time length between the time that a job is input f_i and the latest collection time for the job b_i, determining whether a job

request is considered to be dynamic. Note that b_i represents the latest time at which a job can be collected in order to meet the time requirements. Thus, we specify that a job request $r_i \in R_d$ if $b_i - f_i \leq t$.

Note that more metrics for dynamism can be found in the literature and we refer the reader to Schorpp (2011) for further information on this topic.

Most of the aforementioned VRPs and VRPPDs have been looked at from a dynamic perspective (Berbeglia et al., 2010; Pillac et al., 2013). The first characteristic that may differ in solution strategies for dynamic problems is the choice between offline and online handling of dynamics. When offline strategies are used, the problem is re-solved as a static problem each time problem information is updated. Online strategies take into account new job requests by adapting existing solutions rather than producing entirely new ones. Despite their apparent lack of suitability for dynamic problems, offline strategies have shown that they can result in efficient schedules that outperform some generated by online strategies (Yang et al., 2004; Pankratz, 2005). Some online strategies are based on modeling the time distribution of requests in order to produce solutions that leave room for future requests (Yang et al., 2004). Other studies focused on the implementation of waiting strategies that impact the way in which future job requests can be inserted in existing solutions (Pureza & Laporte, 2008; Mitrović-Minić & Laporte, 2004; Larsen et al., 2004). More generally, different types of algorithms are commonly used in online scheduling. Similarly to static problems, insertion-based methods are popular due to their simplicity (Yang et al., 2004; Pankratz, 2005; Xiang et al., 2008).

Metaheuristics such as Genetic Algorithms (GAs) (Holland, 1973) have also been applied to dynamic VRPPD as seen in (Pankratz, 2005). In the latter study however, despite presenting relative benefits in terms of solution quality, the advantage of using a GA over simple heuristics based on request insertion deteriorates as the level of dynamism increases. This is due to the time available for optimization being reduced when the frequency of new job requests increases. Xiang et al. (2008) summarizes the difficulty of choosing a suitable algorithm for dynamic VRPPD as finding the threshold between efficiency, simplicity and optimality, with respect to the problem at hand and its properties. In Xiang et al. (2008), the task of allocating requests of different types to a heterogeneous fleet of vehicles is handled in three steps. First, day schedules, called trips, are created. Each trip is a sequence of destinations and starts and ends at a central depot. Two arc and reachability matrices are created that respectively define distances between locations and possible connections between them. In this way, specific trip legs can be avoided, which reduces the search space and improves feasibility of solutions. In the second step, trips are recombined, while the third step assigns trips to vehicles, using a Maximum Spare Time Block (MSTB) strategy. MSTB calculates the spare time blocks that would result in a trip insertion on each vehicle schedule. These are compared and the vehicle exhibiting the block of maximum length is chosen for the trip, with the rationale that this vehicle will be more likely to handle future requests. The dynamism of the problem is handled through insertion of new requests in trips alongside the use of MSTB. Although the problem presented in Xiang et al. (2008) has many similarities with the situation at ARRC, there are important differences, including the ability to reject job requests, the restriction to non-dissociable vehicles and the use of a unique depot.

An extra challenge which arises from dynamic VRPs is the difficulty in measuring the performance of dynamic algorithms in comparison with their static counterparts. As a result, operational simulations must be run and appropriate test platforms need to be carefully designed. Simulation may imply job request generation as seen in Yang et al. (2004) and Xiang et al. (2008), but also requires a process that simulates the arrival of job requests and updates to their status. This is mainly done through epoch-based procedures (Yang et al., 2004).

In this paper, an offline strategy is adopted and the notion of trips from Xiang et al. (2008) is used with significant adaptations as described in the next section.

4. Solution Approach

4.1. User interface

Although the main focus of this paper is not related to the user system, it is important to briefly describe the graphical interface we designed for ARRC to understand the quality of the data collected and so the quality and relevance of the solver's recommendations.

The first challenge arising from the problem at ARRC was the absence of an existing computerized system to record jobs. Therefore, it was not possible to solely focus our efforts on the solver: a graphical interface had to be created to collect data from the daily operations at ARRC. This was designed to allow staff to input the essential job

data required by the solver. Care was taken to minimize the amount of time each member of staff would have to spend in typing the job information. Following user feedback, a final web-based interface was developed that allowed input of resources, customer and job data. In addition, the interface allows for the state of information of each job to be updated as jobs are being collected, completed or cancelled, through a series of simple action buttons.

Feedback from the sales team, using mainly the job input interface was positive, with only a few seconds and a few clicks needed to input jobs. This was very important as it means that jobs could be input dynamically as they were received at ARRC. As a consequence, accurate input times were recorded for all jobs. On the other hand, tests of the interface with the operations team showed that tracking actions on jobs was very challenging because of the fast pace at which the job information is updated. However, this issue is minor with respect to the objectives of the present study since the interface made it possible to record accurate real job information.

4.2. Solver

In this section, the different components that are part of the solver are presented. Although the problem's objective is to minimize the number of subcontracted jobs, the solver minimizes the number of allocated trucks. The decision to refine the objective was motivated by the algorithm used which borrows ideas from other VRP studies, commonly optimizing the number of vehicles. In addition, by minimizing the number of trucks, the approach is able to suggest possible reduction in fleet size. In fact, because the solver always prefers company-owner trucks to subcontracted trucks, minimizing the number of allocated trucks also minimizes the number of subcontracted trucks. However, we continue to report the number of subcontracted jobs as a key performance indicator for assessing the value of solutions.

4.2.1. Distance and time estimates

An ideal way to perform this calculation would be to use recorded data on past jobs to produce time estimates for similar jobs. However, because no data was accessible prior to the study, it was decided to design a simple solution to yield approximate time estimates. Latitude and longitude of the locations were used to calculate straight line distances as shown in (19) for two locations a and b with respective pairs of {latitute,longitude} denoted as $\{lat_a, lon_a\}$ and $\{lat_b, lon_b\}$ and R' representing the Earth radius.

$$d = \arccos(\sin(lat_a)\sin(lat_b) + \cos(lat_a)\cos(lat_b)\cos(lon_a - lon_b)) * R' \tag{19}$$

From the straight line distance d calculated between two points, the time estimate needs to be computed. The challenge in doing so is that the activity at ARRC involves urban, semi-urban and rural travel. Because of this property, it is difficult to fit a simple model based on distance and provided time estimates. Based on preliminary tests, we decided to use a linear function in order to estimate travel time, assuming that short journeys are performed in an urban environment while the longest journeys will be mostly through rural areas. The final equation shown in (20) uses details of the shortest and longest distances between two locations. $minT$ and $minD$ respectively denote the provided travel time and distance for the shortest possible journey a truck can perform. Similarly, $maxT$ and $maxD$ gives the same information for the longest journey.

$$tt = \frac{maxT - minT}{maxD - minD} * (d - minD) + minT \tag{20}$$

In addition to the travel time tt, the time estimate for a job's execution also needs to take into consideration the time required for handling the loads at collection and delivery. In practice, it is assumed that 30 minutes is spent loading and 30 minutes spent unloading. Similar values are used in the solver. These times are added to the travel time in our approach.

In order to evaluate the proposed time estimation method, estimates were generated for all pairs of locations. These were then compared with estimates obtained from the controllers at ARRC, based on experience. Estimates using GoogleMaps were also generated for comparison purposes. Figure 2 shows the time estimates obtained using the three approaches. Linear regression was used to obtain a trend line from the operators' numbers. It is clear from Figure 2, that the estimation method presented in (20) is a good match to reported travel times, while results obtained using GoogleMaps are not. This is to be expected because GoogleMaps estimates does not consider heavy vehicles. The difference between the estimation methods was discussed with ARRC and led to the decision to use (20) to estimate travel times until a better approach based on mining historical collected data becomes possible.

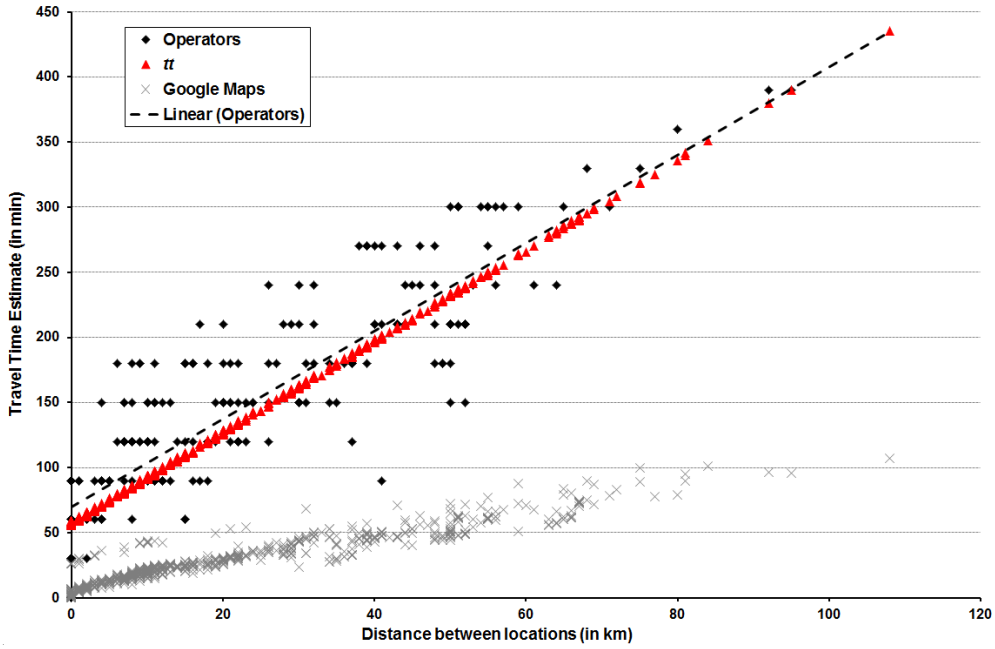


Figure 2: Travel time estimates for all pairs of locations in the data and for three different estimation methods: operators' knowledge; *tt* obtained from (20); and Google Maps. The dotted line shows the linear regression curve representing the operators' estimates

.

4.2.2. A 4-step constructive approach for allocation

The problem at hand contains several constraints that must all be satisfied for a solution to be valid. As the problem needs to be solved dynamically, a lightweight approach with short computation time is needed. For these reasons, simple constructive methods were assessed rather than more advanced heuristic approaches such as GA. Another reason behind this choice is the difficulty created by the trailer dissociation constraint, which requires changes of trailers when performing inbound and outbound shipping jobs. Preliminary work showed that this constraint, that only applies to certain types of jobs, creates either a representational complexity or solution-repair overhead when applying heuristic search.

Our chosen strategy consists of four distinct steps as illustrated in Figure 3. In the first step, jobs are gathered together in order to create a set of trips. At the end of this process, each job is part of a trip. When building trips, jobs are selected in such a way that all jobs in a single trip can be performed by a single trailer. This means that enough time for travel between jobs is allowed, that all jobs are compatible with each other and that there exists an ordering of these jobs that allows them to be performed without breaking constraints. In addition, inbound shipping jobs can only occur at the beginning of a trip as they require a new trailer to be collected. Similarly, outbound shipping jobs can only occur at the end of a trip, where a trailer would need to be left at the delivery location. Note that there is no limit placed on the number of trips. Once trips have been created, the minimum size of trailer is determined along with its ADR and airport training requirements. Based on the minimum trailer size, a trailer from a list of available resource is allocated to each trip. This is followed by the truck allocation step in which trucks from a list of available trucks are allocated to all trailers previously assigned. If an allocated trailer is already attached to a truck, this unit is automatically assigned to the trip. In other cases, trucks are selected based on their location. Finally, the fourth step of the process attempts to minimize the number of trailer changes. If a truck assigned to a trip is already attached to a trailer that differs from the one assigned in the second step, these trailers are swapped. This ensures that if a truck with a trailer attached is used, it will use the same trailer unless it is assigned to a trip involving an inbound shipping job. The proposed approach is similar in many ways to the one chosen in Xiang et al. (2008), where trips are first created and vehicles allocated in a second step. However, due to the higher number of constraints involved here, the solution from Xiang et al. (2008) is only used as a reference and many additional features had to be developed for the

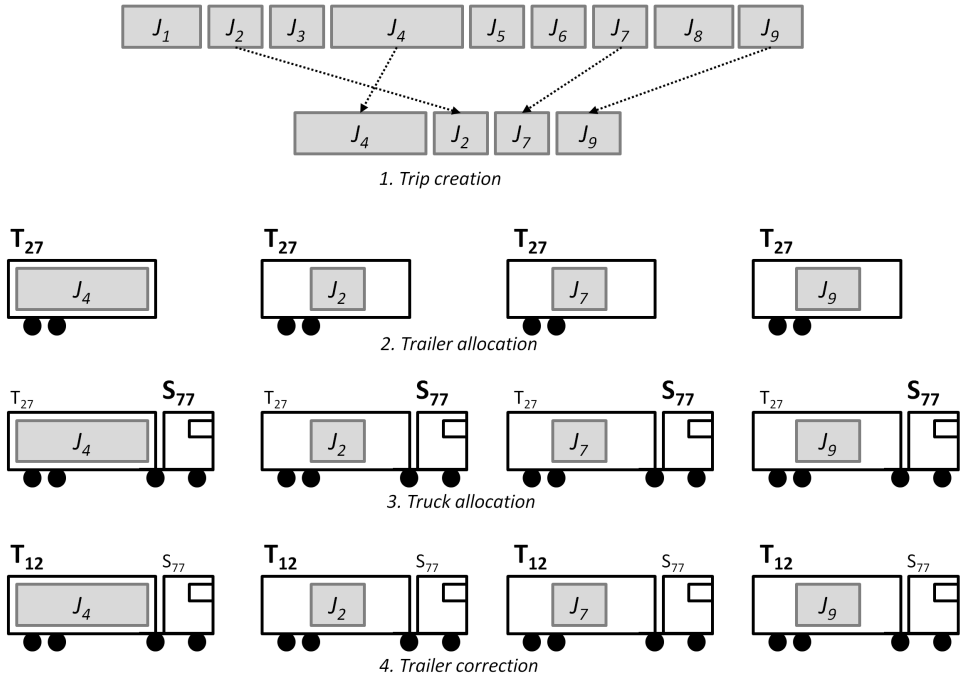


Figure 3: The 4 steps of the allocation process. This example shows the creation of a trip containing jobs J_2, J_4, J_7 and J_9. This trip is initially assigned to trailer T_{27} and truck S_{77}. In the correction step, the trailer is changed to the one already combined with truck S_{77}, that is trailer T_{12}. Note that jobs will be performed successively by the allocated resources.

problem at hand.

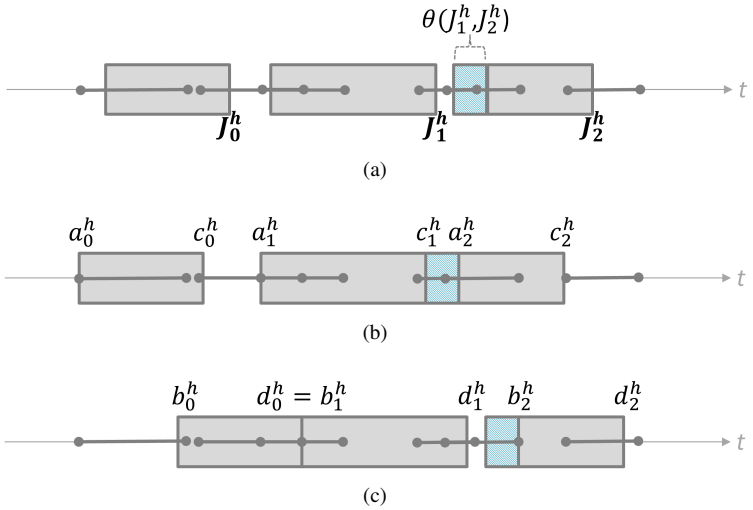


Figure 4: Representation of a sequence of jobs within a trip and its attributes. a) Base representation showing 3 jobs J_0^h, J_1^h and J_2^h, in the trip ω_h. Travel time between jobs is also represented. Travel time between J_1^h and J_2^h is shown and denoted as $\theta(J_1^h, J_2^h)$. Note that in this example, no travel time is required between J_0^h and J_1^h, highlighting an ideal connection. b) Representation of the jobs' earliest possible collections when in the trip. The earliest collection time for the job at the p-th position is denoted as a_p^h and its earliest delivery time as c_p^h. c) Representation of the jobs' latest possible deliveries, showing their latest collection time b_p^h and latest delivery time d_p^h.Note that empty spaces represent some slack between jobs, allowing for their start and end time to be flexible, or allowing for new jobs to be inserted.

Trip creation. We denote the set of all trips that are generated by Ω and the h-th trip of Ω by ω_h. Additional notation is used to define characteristics of jobs within a trip. Figure 4 shows this for three jobs in a trip. For a job at the p-th position in a trip ω_h, the earliest collection time within this trip is defined as a_p^h, its latest collection time as b_p^h, its earliest delivery time as c_p^h and its latest delivery time as d_p^h. The actual collection time of a job at the p-th position in a trip ω_h is denoted by t_p^h. We adopt the convention that t_p^h and t_i are set to ∞ when the relevant job is not collected. Similar notation is used to define whether a job within a trip is outbound shipping or inbound shipping by defining $\lambda_p^h = 1$ if the p-th job in trip ω_h is outbound shipping, 0 otherwise and $\mu_p^h = 1$ if the p-th job in trip ω_h is inbound shipping, 0 otherwise. Travel time between the delivery location of a job J_i and the collection location of a job J_j is denoted as $\theta(J_i, J_j)$. This time may include cargo handling. In the proposed approach, time windows are kept in their continuous form.

In order to manage trailer compatibilities within trips, we define $\tau(J_i)$ as the trailer assigned to a job J_i and M as the set of all possible trailer types (i.e. specifications). We introduce functions denoted ϕ which assign a trailer type to a job or to a trailer. The particular meaning of ϕ will be made unambiguous by context. Thus, $\phi(J_i)$ refers to the trailer type requested for J_i, and $\phi \circ \tau(J_i)$ the trailer type of the trailer allocated to J_i. We represent compatible trailer types by a relation $\varsigma \subset M \times M$, where $(m_1, m_2) \in \varsigma$ signifies that a trailer of type m_1 can be substituted by a trailer of type m_2. Note that ς does not have to be symmetric or transitive. Table 1 shows the specific example of ς used in this paper.

Algorithm 1 summarizes the trip creation process. The set of trips is denoted as Ω. Jobs are handled one at a time. For each existing trip of Ω, a possible insertion position *bestPosition* is generated for the selected job. This is done through the $getInsertionPositionForJobInTrip()$ process described in Algorithm 2. For a given job and trip, this process returns the position at which the job should be inserted in the trip. If insertion is not possible due to time or capacity constraint violations, the returned insertion position is -1. A job is inserted in the first trip in which insertion is possible. If none of all possible insertion positions for all existing trips is suitable for a given job, a new trip is created in Ω and the job inserted.

Algorithm 1 Greedy method for trip creation

```
1:  Initialize stop = false, Ω = ∅
2:  for each job J_i, i ∈ [1, n] do
3:      while ¬stop do
4:          for each trip ω_h of Ω, h ∈ [1, |Ω|] do
5:              bestPosition = getInsertionPositionForJobInTrip(J_i, ω_h)
6:              if bestPosition ≠ −1 then
7:                  insert J_i in ω_h at position bestPosition
8:                  stop = true
9:              end if
10:         end for
11:         if ¬stop then
12:             add new trip to Ω
13:             insert J_i in ω_|Ω| at position 0
14:             stop = true
15:         end if
16:     end while
17:     stop = false
18: end for
```

As mentioned previously, one of the key components of the trip creation is the process used to define an insertion position for a given job J_i in a given trip ω_h. This is described in detail in Algorithm 2. Following an initialization step, the compatibility of J_i and all other jobs of ω_h is checked. This is done through using the $compatibleTrailerTypes()$ function specified in Algorithm 3 and is required to ensure that recommendations provided by the solver respect the company's policies as described in Section 2. If J_i is not compatible with all other jobs already in ω_h, J_i cannot be inserted in ω_h and the position returned by $getInsertionPositionForJobInTrip()$ is -1. In the opposite case, further steps are used to determine a possible insertion position. Insertion position is mainly constrained by the job type of J_i represented by μ_i and λ_i and its current collection status. If J_i is ad-hoc and not yet collected, any position in the

Algorithm 2 *getInsertionPositionForJobInTrip*(J_i, ω_h)

Initialize $position = -1$
Initialize $compatible = true$
for each job J_p^h in ω_h, $p \in [0, |\omega_h|[$ **do**
  if $\overline{compatibleTrailerTypes(\phi(J_i), \phi(J_p^h))}$ **then**
    $compatible = false$
  end if
end for
if $compatible$ **then**
  if $\lambda_i + \mu_i = 0$ and $t_i = \infty$ **then**
    for each position p in ω_h, $p \in [0, |\omega_h|]$ and $p \neq -1$ **do**
      if $p = 0$ **then**
        if $\mu_p^h = 0$ and $t_p^h = \infty$ and $(b_p^h > c_i + \theta(J_i, J_p^h))$ **then**
          $position = p$
        end if
      else if $p = |\omega_h|$ **then**
        if $\lambda_p^h = 0$ and $(c_p^h + \theta(J_{p-1}^h, J_i) < b_i)$ **then**
          $position = p$
        end if
      else
        if $(s_i + \theta(J_{p-1}^h, J_i) + \theta(J_i, J_p^h) < b_p^h - c_{p-1}^h)$ and $(c_i < b_p^h + \theta(J_i, J_p^h))$ and $(b_i > c_{p-1}^h + \theta(J_{p-1}^h, J_i))$ **then**
          $position = p$
        end if
      end if
    end for
  else if $\lambda_i = 0$ and $(\tau(J_i) \neq 0$ or $t_i \neq \infty)$ **then**
    if $\mu_0^h = 0$ and $t_0^h = \infty$ and $(b_0^h > c_i + \theta(J_i, J_0^h))$ **then**
      $position = 0$
    end if
  else if $\lambda_i = 1$ and $\tau(J_i) = 0$ **then**
    if $\lambda_{|\omega_h|}^h = 0$ and $(c_{|\omega_h|}^h + \theta(J_{|\omega_h|}^h, J_i) < b_i)$ **then**
      $position = |\omega_h|$
    end if
  end if
end if

trip has a chance to be used for insertion. In the case that J_i has type outbound shipping and has no allocated trailer, only the last position in the trip is suitable. For the remaining cases, that is inbound shipping jobs or any other jobs with either a trailer already assigned or already collected, only the first position can potentially be used. In any case, insertion of J_i is assessed by estimating its execution time and the travel times between J_i and its neighbours in ω_h, that is its predecessor and successor jobs, if any. Inserting J_i at a specific position p should ensure that its predecessor job, denoted J^h_{p-1}, (if any) can be performed and enough time allowed for travelling between jobs without time overlap. For this purpose, the earliest delivery time c^h_{p-1} of J^h_{p-1} is used. Similarly, for a successor job, denoted J^h_p, (if any) the latest collection time b^h_p is used as a reference.

The final component of the trip creation is the compatibility function between two jobs defined in Algorithm 3. This function simply checks if the largest of the two specified trailer types $\phi(J_i)$ and $\phi(J_j)$ of two jobs J_i and J_j belongs to the set of compatible trailers of the smallest one obtained from the relation ς. If the above is true, the jobs are compatible.

Algorithm 3 $compatibleTrailerTypes(\phi(J_i), \phi(J_j))$

Initialize $compatible = false$
if $\phi(J_i) > \phi(J_j)$ **then**
  if $\phi(J_i) \in \varsigma(\phi(J_j))$ **then**
    $compatible = true$
  end if
else
  if $\phi(J_j) \in \varsigma(\phi(J_i))$ **then**
    $compatible = true$
  end if
end if

Algorithm 4 Trailer allocation

Initialize set of available company trailers $Z = \{T_k | T_k \in T, w_k = 0\}$
Initialize set of available subcontractor trailers $Y = \{T_k | T_k \in T, w_k = 1\}$
for each trip ω_h, $h \in [1, |\Omega|]$ **do**
  if $\tau(J^h_0) \neq \varnothing$ **then**
    $\tau(\omega_h) = \tau(J^h_0)$
    $Z = Z \setminus \tau(\omega_h)$
  end if
end for
for each trip ω_h, $h \in [1, |\Omega|]$ **do**
  if $\tau(J^h_0) = \varnothing$ **then**
    if $getClosestCompatibleTrailer() \neq \varnothing$ **then**
      $\tau(\omega_h) = getClosestCompatibleTrailer()$
      $Z = Z \setminus \tau(\omega_h)$
    end if
  else
    $\tau(\omega_h) = getClosestCompatibleSubcontractorTrailer()$
    $Y = Y \setminus \tau(\omega_h)$
  end if
end for

Trailer allocation. Once a set of trips has been generated, a first attempt is made at allocating trailers to each. Because trips are never re-ordered during the trip creation procedure, the first trips created (i.e. with low h values) tends to have more jobs and less idle time than trips created later. For both trailer and truck allocation, resource is allocated to trips following the order in which they have been created, which tends to prevent the allocation of subcontracted resources to trips with a lot of jobs. For trailer allocation, we extend the τ notation previously defined. Consequently,

$\tau(\omega_h)$ represents the trailer allocated to the trip ω_h and therefore, for each job J_p^h of ω_h, $\tau(J_p^h) = \tau(\omega_h)$. The initial trailer allocation process shown in Algorithm 4 begins with the initialization of the sets Z and Y of all trailers available from the company or from subcontractors. The rest of the allocation is done in two main steps. First, trips whose first job, i.e. J_0^h, already has a trailer allocated are set with the same trailer as their first job. This situation is seen when the first job of a trip is either inbound shipping or already collected. Once allocated to the trip, the trailer is removed from Z. In the second step, the remaining trips ordered by time of creation are handled by allocating the closest compatible trailer. This is done through the *getClosestCompatibleTrailer*() and *getClosestCompatibleSubcontractorTrailer*() functions. These functions essentially order available trailers by increasing distance to the collection location of the first job in trip. They then check compatibility of the trailers with the trip details. In addition to the sole trailer information, they also consider information about the truck currently attached to the trailer, if any. This means that when the closest trailer is currently attached to a truck, the truck details, including the set of constraints E, need to match the trip details for the trailer to be allocated. This strategy aims at reducing the number of trailer changes for trucks, which is generally a time consuming operation in practice.

Algorithm 5 Truck allocation

Initialize set of available company trucks $V = \{S_l | S_l \in S, z_l = 0\}$
Initialize set of available subcontractor trucks $U = \{S_l | S_l \in S, z_l = 1\}$
for each trip ω_h, $h \in [1, |\Omega|]$ **do**
  if $t_0^h \neq \infty$ **then**
    $\sigma(\omega_h) = \sigma(J_0^h)$
    $V = V \setminus \sigma(\omega_h)$
  end if
end for
for each trip ω_h, $h \in [1, |\Omega|]$ **do**
  if $\tau(\omega_h) \neq \varnothing$ and $\sigma(\omega_h) = \varnothing$ **then**
    if $\sigma(\tau(\omega_h)) \neq \varnothing$ **then**
      $\sigma(\omega_h) = \sigma(\tau(\omega_h))$
      $V = V \setminus \sigma(\omega_h)$
    end if
  end if
end for
$V = V \setminus \{S_l | S_l \in V, y_l = 1\}$
$U = U \setminus \{S_l | S_l \in U, y_l = 1\}$
for each trip ω_h, $h \in [1, |\Omega|]$ **do**
  if $\tau(\omega_h) \neq \varnothing$ and $\sigma(\omega_h) = \varnothing$ **then**
    if *getClosestCompatibleTruck*() $\neq \varnothing$ **then**
      $\sigma(\omega_h) =$ *getClosestCompatibleTruck*()
      $V = V \setminus \sigma(\omega_h)$
    end if
  else
    $\sigma(\omega_h) =$ *getClosestCompatibleSubcontractedTruck*()
    $U = U \setminus \sigma(\omega_h)$
  end if
end for

Truck allocation. The third step of the proposed algorithm focuses on allocating trucks to the trips. We introduce the notation $\sigma(\omega_h)$ to denote the truck allocated to all jobs in ω_h. This means that for each job of ω_h, $\sigma(J_p^h) = \sigma(\omega_h)$. From the previous steps, trips have been assigned trailers. Algorithm 5 describes the truck allocation process. It starts with the initialization of all available company and subcontracted trucks represented by the sets V and U respectively. On an initial pass through all trips, the algorithm allocates trucks to trips whose first job has already been collected as there is no choice on which truck can be used. These trucks are removed from V and U. The algorithm makes a

second pass through all trips and allocates trucks to trips whose allocated trailers are currently attached to a truck. For this purpose, we overload the notation $\sigma(\tau(\omega_h))$ and define σ to assign the truck attached to a particular trailer. We also allow $\sigma(\tau(\omega_h))$ to take a $\varnothing$ value if no truck is attached. After this point, the only trips with no trucks allocated are trips with unattached allocated trailers. For this reason, the only trucks that can now be allocated are non-rigid. Hence, all rigid trucks are removed from V and U. In the final step, trips whose trailers are not attached are assigned a truck based on the $getClosestCompatibleTruck()$ and $getClosestCompatibleSubcontractedTruck()$ functions, which are similar in nature to the $getClosestCompatibleTrailer()$ function already described.

Trailer correction. The fourth and final step of the constructive solver compares for all trips whose first job is not collected and not of type inbound shipping, the allocated trailer and the trailer currently attached to the allocated truck. If these trailers are not the same and the trailer attached to the truck is compatible with all jobs of the trip, the allocated trailer is replaced with the trailer currently attached to the truck. Because of the order in which each step of the algorithm is run, there is no risk of assigning a trailer to many trips at this stage. Also, this correction step further reduces the number of trailer changes required. The outcome of this last step is a set of trips with compatible trucks and trailers allocated to each of them.

4.2.3. Solver optimization

The 4-step constructive approach presents features that can be optimized. For each, there are several strategies that can be employed. Here, we present those features and different optimization options. Options are compared empirically and results are presented in Section 5.

Job insertion mechanisms. There exist different ways to insert unscheduled jobs in existing trips. Here we investigate two options. In Algorithm 2, a job is inserted in the first suitable trip. We call this insertion procedure *first insertion*. This procedure is initially chosen for its simplicity resulting in fast insertion of jobs. However, it may come at the cost of a sub-optimal final set of trips. In order to understand the impact of the insertion method, a *best insertion* alternative was defined. The *best insertion* procedure aims to insert the job in the best possible existing trip. Prior to a job insertion, trips are compared with each other based on the idle time that would result in inserting the job. The trip with the shortest potential idle time is chosen as the best trip.

Job ordering. The solver presented in this paper is a greedy constructive method. Based on a given ordering of jobs, the solver returns a set of trips. It appears clearly that the job ordering used as an input in Algorithm 1 influences the configuration of the final set of trips. Thus, it is legitimate to try optimizing the job ordering, which is a permutation. Although the ordering of jobs could be sorted by using job attributes such as time information or type of jobs, Local Search (LS) methods were preferred as a way to understand if improvements can be gained with reasonable computational efforts. This is motivated by the fact that it is not clear what attribute should be used to sort jobs and thus choices in this respect could bias the optimization process. Hill Climber (HC) and Simulated Annealing (SA) (Kirkpatrick, 1984) are chosen for their simplicity and their good performance on many problems. Both algorithms start from an initial solution generated at random and perform several iterations where the solution is modified. While HC only keeps changes if they improve the quality of the solution, SA allows some degrading changes with a certain probability which depends on the degree of deterioration and the number of iterations performed. For a fair analysis of SA on different problem instances and to reduce the efforts dedicated to parameter tuning, SA is set with the naive parameters that showed best overall results in preliminary runs, that is an initial temperature of 1 and a cooling rate of 0.5. We refer the reader to Aarts & Lenstra (2003) and Eglese (1990) for more in-depth analysis of HC and SA. For both HC and SA, the swap mutation was used as local operator. As suggested by its name, a job ordering is modified by swapping the position of two randomly selected jobs. Although alternative operators can be used, swap is generally a good operator to use to get an initial insight on what LS can do (Serpell & Smith, 2010). At each iteration of the LS, the greedy procedure is run using the job ordering produced and thus the optimized job ordering is the ordering produced after the last iteration.

4.3. Handling the dynamism

The constructive solver is an offline method. It re-runs after each event, adapting to the dynamic nature of the problem using updated problem information. Collected jobs, that is those with $t_i \neq \infty$, can only be scheduled at the

start of a trip. In addition, if a job is already collected, the resource physically assigned to the job will automatically be allocated to the trip in which it is placed. Finally, completed jobs are removed from the list of jobs to schedule and new jobs added prior to re-running the solver.

4.4. Simulation

The simulator designed for experimental purposes works by mimicking the job inflow, a requirement needed for a realistic assessment of the solver. Real life jobs collected at ARRC are introduced based on their recorded input time as the solver iterates, simulating the dynamic change to problem information. In addition to job input, the simulation considers the other actions that affect the state of jobs such as collection and delivery events. By considering that some events trigger others, the simulation can faithfully reproduce the changes to problem information covering a full day of operation.

The simulation manages an ordered list of events. For each job, there are 3 events that are *input*, *collection* and *delivery* events. Each of these events has an associated time at which it is performed. For a job J_i, the time of the *input* event is set to f_i. The time of the *collection* event is t_i and is generated by the solver. Finally, the time of the *delivery* event is $t_i + s_i$. The simulation uses a fourth kind of event in which the solver is run for the first time. It is typically set to α, the company's opening time. Throughout the simulation, events are executed in time order and triggers addition of events. For example, when an *input* event is performed, a *collection* event is added in which the trailer and truck recommended by the solver are assigned to the job. Similarly, when a *collection* event is performed, a *delivery* event is added. Following its first run, the solver runs each time the information is updated, that is after each event. This also updates the time of all successive events. The simulation terminates when all events have been executed and business-related KPIs can be measured, typically, number of subcontracted jobs, used trucks and used trailers over the simulation period. By means of this simulation, performance of the solver can be compared with real-world operational numbers reported by ARRC.

5. Experiments, Results and Discussion

5.1. Data

5.1.1. Data collection

In order to evaluate the quality of the recommendations obtained from the solver, the simulation framework described in Section 4 needs to use data that represents real operations at ARRC as accurately as possible. Four days were dedicated to data collection at ARRC, aiming at capturing details on all jobs during each of these days. As part of this information, the input times of all jobs were recorded, which allows the dynamic simulation to faithfully reproduce the dynamic occurrence of new job information.

The data collection process is a challenging and time consuming task for staff members at ARRC. It involves adding extra work to a team which is already under a lot of pressure. Despite the final collected data being a close approximation to the actual days' information, it is essential to report some of the difficulties and assumptions that were made before presenting our results. It is important to understand that the data collection task, although handled seriously by ARRC, could not be prioritized over other tasks that involve customer satisfaction or unexpected problem solving. The list below describes some of the approximations related to data collection under these conditions.

- Recording the exact details of available trucks and drivers at the beginning of each day is highly time consuming and could not be carried out at ARRC. Instead, we assume that all trucks start at their respective base at the start of each day. This also means that the number of ADR and airport trained drivers is not known prior to the data collection exercise. We were able to collect data on the number of trucks of each type for one of the four days on which data was recorded, 23rd October - see Table 5.

- The prototype version of the data collection software does not prevent or correct for erroneous data entry. The data was manually reviewed but some errors in precise job times or destinations may remain.

- Activity at ARRC is unpredictable and each day offers unique situations that may require a lot of resources for the day. Hence, we cannot assume that the days we have selected are typical.

Table 2: Data characteristics

	6Mar	23Oct	24Oct	5Nov
AH jobs	125	76	82	80
OS jobs	18	28	38	34
IS jobs	60	48	14	41
Total jobs	203	152	134	155
ADR jobs	44	35	30	27
dod	49%	32%	13%	50%

5.1.2. Data characteristics

We collected four distinct datasets at ARRC on 6th March, 23rd October, 24th October and 5th November. These are labelled *6Mar*, *23Oct*, *24Oct* and *5Nov* respectively for the remainder of this paper. Table 2 describes the job characteristics of the four datasets, that is the number of jobs per type and the number of AH, OS and IS jobs that are ADR-constrained. It is important to note that the data collection for *24Oct* was interrupted during the day and therefore reported numbers only represent partial data for that day. However, it was decided to retain this data to the study, mainly because the interruption did not prevent the busiest part of the day from being captured.

Ad-hoc jobs are most common, accounting for over 50% of all jobs in all datasets. *23Oct* and *5Nov* have a similar job distribution. *6Mar* has a lower proportion of OS jobs (9%) but a higher ratio of AH jobs (62%). Finally, *24Oct* exhibits a high proportion of OS jobs (28%) but a lower proportion of IS jobs (10%). With higher proportions of shipping jobs, *23Oct* and *5Nov* require more trailer changes than the other two datasets. Finally, ADR jobs represent between 17% and 23 % of the jobs, depending on the dataset. Note that there were no job requiring a driver with airport training, one constraint of the problem.

Because scheduling at ARRC is dynamic, the above numbers are not sufficient to describe the difficulty of the task. Dynamism measures were recorded for each dataset. The basic *dod*, which considers that a job is dynamic if its input and latest handling times are within the same day, is presented for each day in Table 2. As a consequence to the interuption in the data collection related to *24Oct*, this dataset is mostly static, with only 13% of dynamic requests. *23Oct* displays a *dod* of 32%, while both *6Mar* and *5Nov* have same proportion of static and dynamic jobs, that is 49%.

As discussed in Section 3, the basic dod measure does not capture different levels of dynamism among dynamic requests. For this reason, dod_t is used. Because t cannot be chosen arbitrarily, its value was altered between 1 and 24 hours. We recall that dod_1 only considers as dynamic, jobs whose input and latest handling times are within the same hour. On the other hand, dod_{24} requires at least a 24-hour difference between input and handling times for a job to be static, hence classifying most jobs as dynamic in our case. dod_t is presented in Figure 5 for all datasets, under variation of t. Note that t excludes the time at which the company's premises are closed, that is between 9pm and 7am.

Out of the four datasets, we observe that three of them, *23Oct*, *24Oct* and *5Nov* display similar patterns of dynamism. Although they display differences when t is large, this does not have much impact in practice because operators do not normally allocate jobs more than 10 hours in advance. On the other hand, the dataset *6 March* displays a relatively high degree of dynamism when 4 hours is chosen as a dynamism threshold. This reflects the fact that 40% of the jobs from this dataset are input with a maximum notice of 4 hours. This threshold is meaningful in practice and this type of short notice can lead to serious scheduling challenges.

Finally, data on actual truck usage at ARRC and subcontracted jobs is available for *23Oct*, allowing comparison with the proposed approach on this instance. These numbers are presented in Table 5 along with the results obtained by the simulation.

5.2. Experiments and Results

The present study includes two sets of experiments. First, we present a set of preliminary experiments, aimed at ensuring that the solver used for the main set of experiments is running with optimized settings. In the main set of experiments, the solver is run as a dynamic offline strategy using events generated by simulation over the four ARRC

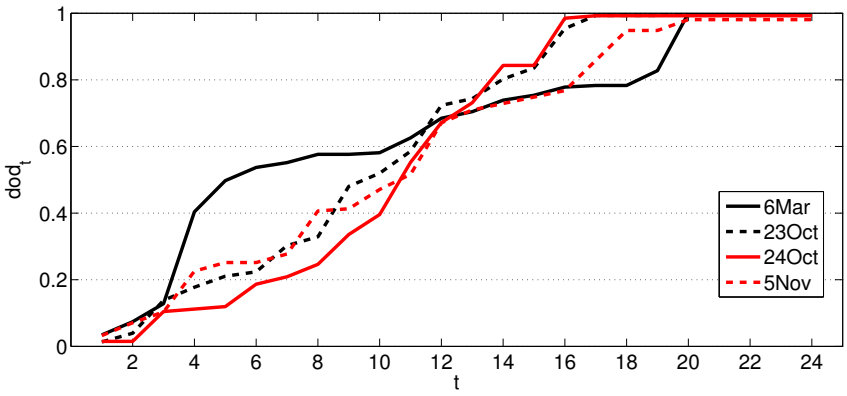


Figure 5: dod_t under variation of t for the four datasets

Table 3: Number of trips obtained by the solver under the different insertion methods following 100 runs using static data. No statistically significant difference is found between the two methods.

	6Mar	23Oct	24Oct	5Nov
first insertion	117.4 (1.1)	95.5 (0.7)	81.5 (1.2)	95.2 (0.9)
best insertion	117.6 (1.2)	95.6 (0.8)	81.2 (1.4)	95.2 (0.8)

datasets. Results obtained from these simulations are analysed in comparison to current practice and to evaluate the benefit of the solver in supporting management decisions.

Solver settings. In this section we describe preliminary experiments conducted to identify the best settings of the algorithm presented in Section 4 in terms of insertion method and job ordering optimization strategy. Because several algorithm parameters needed testing, it was not sensible to use the simulation on the four datasets available. Instead, the data was made static for the purpose of this preliminary experiment only, that is all jobs were considered as known at the start of the day, when running the solver.

100 runs of the solver were performed using *first insertion* and *best insertion*.The number of trips resulting from these runs is presented in Table 3. Unpaired t-tests were performed based on these results and no statistical significance was found for all datasets between the insertion methods. Because *first insertion* is computationally cheaper than *best insertion*, *first insertion* is chosen for the main set of experiments.

To understand whether LS can benefit the performance of the solver, experiments are performed in which HC and SA are both run 30 times on each static dataset for different numbers of iterations. The results, which are presented in Table 4, shows that the choice of LS method impacts the performance, with SA reaching overall better results in less iterations (statistical significance reached by SA on *23Oct* and *5Nov* after 10 iterations only, while HC requires respectively 100 and 50 iterations). Although the number of iterations decides how much improvement is gained by LS, it does not always require many to show significant difference with the base greedy search. Using 1000 iterations, large differences are obtained. However, this is also correlated with an increased computational burden which would

Table 4: Number of trips obtained by the solver using local search following 30 runs using static data for different number of iterations. Configurations that are significantly better than the simple greedy search using *first insertion* are shown in bold.

	6Mar		23Oct		24Oct		5Nov	
Iterations	HC	SA	HC	SA	HC	SA	HC	SA
10	117.2 (1.0)	117.2 (0.9)	95.3 (0.4)	**95.2 (0.4)**	**80.8 (1.2)**	**80.8 (1.0)**	95.2 (0.9)	**94.8 (0.6)**
50	**116.7 (0.7)**	**116.5 (0.7)**	95.3 (0.5)	**95.2 (0.4)**	**79.6 (0.8)**	**80.1 (0.9)**	**94.8 (0.9)**	**94.5 (0.5)**
100	**116.3 (0.7)**	**116.4 (0.8)**	**95.0 (0.2)**	**95.0 (0.2)**	**79.1 (0.8)**	**79.0 (0.7)**	**94.4 (0.5)**	**94.4 (0.5)**
1000	**115.2 (0.4)**	**115.4 (0.5)**	**95.0 (0.0)**	**95.0 (0.0)**	**77.1 (0.5)**	**77.1 (0.4)**	**94.0 (0.0)**	**94.0 (0.0)**

not be acceptable to run dynamically after each event. Therefore, despite the apparent benefits of LS, we reject it in favour of the lightweight constructive approach without pre-ordering. We suggest that the use of LS and alternative metaheuristics be the subject of future work.

Solver performance. Our main experiments are designed to address the following questions:

- Can the proposed approach match or improve performance obtained by ARRC operators?
- Can the proposed approach be used to estimate the resources needed?
- Can the simulation framework be used to evaluate the importance of decision-making assumptions?
- What are the overall benefits the approach could bring to ARRC?

Several scenarios are simulated for each dataset. 10 runs of the dynamic simulation were performed for each experiment, allowing for mean and standard deviations to be computed. The initial location of each truck was set to the driver's base which is known and hence does not vary across runs. A random trailer was initially allocated to each non-rigid truck. Since there is a larger number of trailers than non-rigid trucks, the unassigned trailers were randomly located between the different docks, where empty trailers usually await in practice. This allocation is done in the initial phase of the simulation and so differs between successive runs. However, because of the large number of trailers available, these assumptions are very unlikely to lead to a significant variation in the solver's performance across runs.

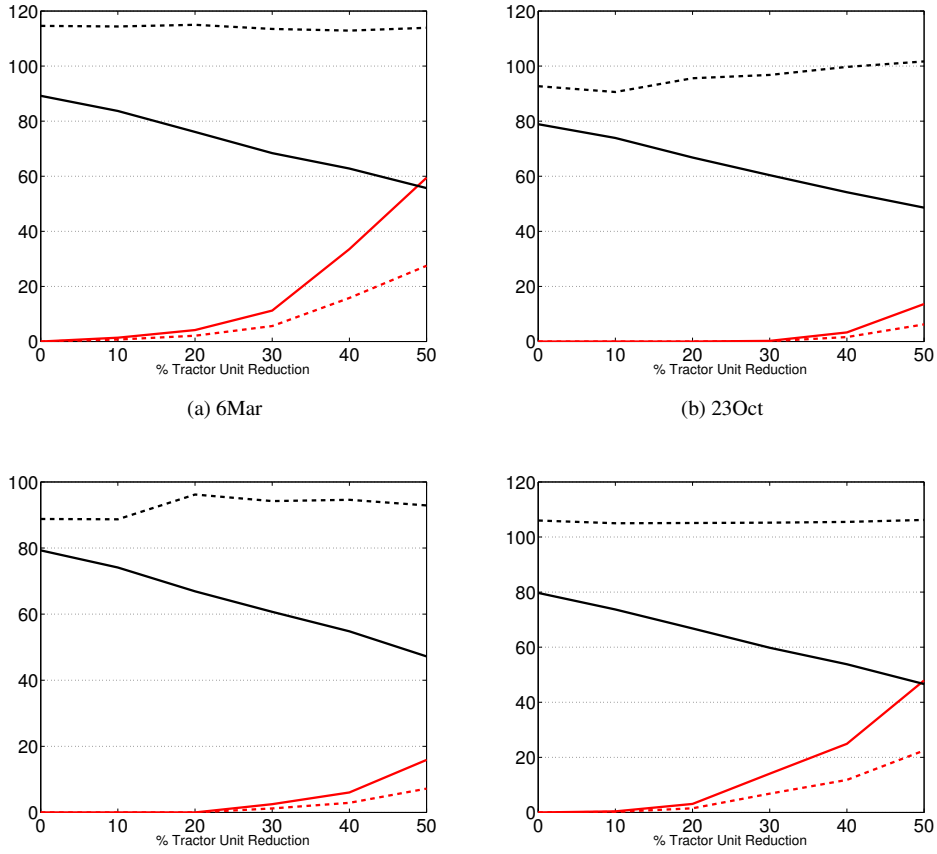
The performance of our approach is assessed by incrementally decreasing the number of non-rigid trucks available to the solver while the number of subcontracted jobs that results from each experimental setup is recorded. The reduction of non-rigid trucks is done in steps of 10%, starting from an initial number of trucks of 99, considered to be much larger than the required number. We recall that actual truck usage data is only known for one dataset. No reduction beyond 50% is considered as it was observed in preliminary runs that this value already leads to a significant number of subcontracted jobs being needed on all datasets.

Results for the four datasets are presented in Figure 6. This shows how the number of trucks and trailers used in the simulation varies as the total number of trucks available decreases. It also shows the number of subcontracted jobs resulting from the decisions of the solver in simulation. Figure 6 illustrates that in general the number of used trucks decreases with the number of available ones, unlike the number of trailers used which remains overall constant. The same figure also shows that the number of non-rigid trucks can be reduced without always leading to jobs being subcontracted. Simulation on *6Mar*, the biggest dataset, shows that the reduction in trucks quickly leads to subcontracted jobs, although their number remains low up to a 30% reduction. On the other hand, the number of trucks can be decreased significantly more on *23Oct*, with the first subcontracted jobs appearing beyond 30% reduction and remaining low until 50%. We observed through simulation that reducing the number of trucks available also affects the number of trucks used, which may seem counterintuitive at first. However, this can be explained. First, when reducing the fleet in our experiments, trucks are removed in a random manner without consideration of their locations. Consequently, some locations with a high demand may have a deficit of trucks, resulting in the solver allocating trucks that are based far away, which is likely to delay the start of a trip and increase the number of subcontracted resources needed. Second, because this observation is a result of the simulation, it is important to consider the dynamic aspect of the scheduling and the fact that scheduling is performed repeatedly throughout the simulation. At each iteration, trucks are allocated through the $getClosestCompatibleTruck()$ function which chooses trucks from a pool of available trucks. Chances of allocating a truck which has already been used on a trip are therefore higher if the pool is small rather than large. Smaller number of trucks at the end of a simulation run reflects a better usage of the trucks with more jobs allocated per truck.

Since the number of actual ARRC operations is available for *23Oct*, this dataset is used to evaluate the solver quality against current practice. Results in terms of subcontracted jobs, total trucks and trailers used with details of the types of trailer are presented in Table 5. The simulation scenarios considered cover truck reductions of 30%, 40% and 50%. Table 5 shows that at 40% non-rigid truck reduction, the number of subcontracted jobs needed by the solver in simulation is lower than that used in actual operations. The number of trucks used is also much lower, although the solver in simulation makes use of more vans and 18-ton trucks. At 50% reduction, the number of non-rigid unit used is close to half those used in practice. Yet, only 6.2 jobs are subcontracted on average.

Table 5: Comparison of average solver results with actual numbers for *23Oct*. Solver results are computed over 10 runs and standard deviations are shown in brackets.

	Actual Numbers	30% Red.	40% Red.	50% Red.
Total Trucks	71	60.4 (0.5)	54.2 (0.8)	48.6 (0.5)
Total Trailers	unknown	96.8 (1.0)	99.7 (1.2)	101.7 (2.0)
Non-rigid Trucks	61	45	39	32
Van	4	8.4	8.2	8.3
5-ton	3	3.6	3.7	3.7
10-ton	3	1	1	1
18-ton	0	2.4	2.3	3.6
26-ton	0	0	0	0
Sub. Jobs	3	0.1 (0.3)	1.6 (2.1)	6.2 (3.2)



(a) 6Mar (b) 23Oct

(c) 24Oct (d) 5Nov

Figure 6: Solver performance estimation under reduction of non-rigid truck number: subcontracted hours (——); subcontracted jobs (- - -); used trucks (——); used trailers (- - -)

Finally, in order to explore the importance of decision-making assumptions, we set two hypothetical scenarios. In the first scenario, labelled *H1*, the importance of the maximum capacity of a trailer that can handle a job is studied. To do so, the constraint is removed, the simulation run and results compared with those of the constrained simulation. By releasing the constraint, the solver allows a job to be performed by any trailer whose size is large enough. This allows for example transit jobs to be handled by flatbed trailers. The second scenario, labelled *H2*, explores the benefit of

Table 6: Average results with different algorithm assumptions on 23Oct over 10 runs. Standard deviations are shown in brackets. Hypothesis that are significantly better than the others are shown in bold. Note that none of them exhibits a significant improvement in terms of subcontracted jobs.

		all Res.	**30% Red.**	**40% Red.**	**50% Red.**
H0	Total Trucks	78.9 (1.1)	60.4 (0.5)	54.2 (0.8)	48.6 (0.5)
	Sub. Jobs	0 (0.0)	0.1 (0.3)	1.6 (2.1)	6.2 (3.2)
H1	Total Trucks	**66.2 (0.4)**	**46.9 (0.3)**	**40.9 (0.3)**	**34.2 (0.4)**
	Sub. Jobs	0.4 (0.7)	0.4 (1.0)	1.2 (1.2)	7.2 (5.5)
H2	Total Trucks	79.9 (1.2)	60.1 (0.6)	54.4 (0.5)	48.1 (0.6)
	Sub. Jobs	0 (0.0)	0 (0.0)	1.1 (1.2)	6.6 (3.6)

training more drivers to higher ADR standard. In this scenario, all drivers are initially set with full-ADR qualifications before running the simulation.

Results for these scenarios showing total number of trucks used and number of subcontracted jobs are provided in Table 6 for *23Oct*. Statistical significance was tested by means of unpaired t-tests. *H0* refers to the original set of assumptions as presented in Section 2. Results show that releasing the constraint on maximum trailer size helps to reduce the overall number of trucks being used. For example, at 40% reduction, only 40.9 trucks are used against 54.2 with the original constraint, for a low number of subcontracted jobs (respectively 1.2 and 1.6). On the other hand, *H2* does not yield improved results over scenario *H0*.

5.3. Discussion

5.3.1. A new vehicle routing variant

The problem presented in this paper is a novel case of VRP. Because it models an existing industrial problem, it is composed of many constraints, is dynamic and involves a fleet of heterogeneous vehicles, some dissociable, others not. These aspects taken together make this a new problem and the proposed solver closer to the industry needs. Failure to incorporate realistic constraints such as these is a limitation of most published VRPs (Pigden, 2013). Although one may also think that such a heavily constrained problem is isolated in the landscape of VRPs, in fact it is related to many problems from the literature presented in Section 3, with the exception of a few such as capacitated VRPs and TTRPs.

As a result of the high definition of the problem, the solver defined here may also be applied to simpler problems. The problem definition allows for problems that considers fewer constraints to use the same formalism. For example the use of homogeneous vehicle fleets or vehicles that do not use trailers does not require any adaptation. Similarly, scenarios where only one type of job is defined or where all requests are static can be modelled as the VRP defined in this paper. This wide range of possible sub-problems that can be handled makes the novel problem definition a valuable formalism whose solutions can be re-used in other contexts, thereby overcoming a limitation of most VRPs, where introduced constraints often result in problem isolation.

5.3.2. Overall performance of the solver

As shown in Table 5, the offline constructive solver is able to effectively solve the dynamic problem. It has also revealed the potential to significantly reduce fleet size through repeated runs with simulated reduction of non-rigid truck numbers. Our results also suggest conversely that, with efficient use, the existing fleet can handle a substantially increased number of jobs.

The solver proves to be efficient at using the available resource. For example, when it has the choice among a wide selection of trucks, it utilizes the available fleet. When the number of available trucks decreases and the solver has fewer choices, it manages to also reduce the number of trucks it uses. This ability to spread or narrow down the effort to the fleet is a valuable feature that suits the need of transportation companies.

5.3.3. Importance of the maximum trailer size constraint

Our results showed that removal of the maximum trailer size constraint improved overall performance. The constraint is intended to prevent the use of large trailers to perform small tasks, motivated by the desire to use the heterogeneous characteristics of the fleet to full advantage.

Close analysis of the results obtained by the solver when this constraint is in use however, show that the number of trucks needed is much higher than when the constraint is relaxed, with similar outcomes in terms of number of subcontracted jobs. Without this constraint, we observe that only flatbed trailers are used, while all smaller vehicles appear not to be required. These results also suggest that flatbed trailers are not used at their maximum potential in the base simulation runs whose results are presented in Figure 6 and Table 5.

While strongly suggestive, these results alone are not enough to suggest changes in the fleet distribution. Extra aspects related to the cost, management and of other nature of such a decision are not considered here and these changes may not fit the company policy despite good simulated results. However this does show how the simulation framework could be a valuable tool in questioning business assumptions during strategic planning.

5.3.4. Overall benefit to the company

Experiments showed that use of the dynamic constructive solver could help reduce the fleet size or alternatively increase the workload, the original bottleneck in ARRC operations. Analysis of some decision-making hypotheses shows that some such as those relevant to trailer capacity may bring benefit to the company. The simulation framework allows such assumptions to be explored and presents opportunities to further develop features that can be used for fleet management including usage prediction and planning for example.

Although solver effectiveness is proved in this study, a packaged software system is needed to make best use of it. This represents the next development step at ARRC. Among other things, integration of the solver to operations will enable analysis of the solver perfomance from live trial, using different performance indicators related to qualitative aspects such as operator' satisfaction and operator trust in the tool.

6. Conclusion and Future Work

In this paper, we have introduced a new variant of VRP, inspired by a problem faced in the haulage industry. The problem is heavily constrained, relates to a heterogeneous vehicle fleet, considers different types of job with distinct requirements and is dynamic. The new problem also has simplified formulations that conform to many of the VRPs reported in the literature. A lightweight dynamic offline constructive solver is proposed to produce solutions to the problem, in which trips are created before successively assigning trailers and trucks to each. Four datasets were collected from ARRC that allow evaluation of the proposed approach on real data through the use of a simulation.

Experiments demonstrate that the solver can realize improved performance in terms of truck usage and subcontracted jobs in comparison to current operational practice. Investigation of different decision-making scenarios using the solver in simulation shows that strict constraints on trailer capacity inhibit solutions of better overall quality.

However, full benefit can only result from integration of the solver to the company's infrastructure. Development work is currently under way to achieve this. Future work will focus on developing and evaluating refinements to our approach including: more accurate estimation of timings using live data; incorporation of soft or partitioned time windows and investigation of suitable metaheuristics to improve solution quality at acceptable computational cost.

Acknowledgments

This work is funded by ARR Craib Ltd. and by the Scottish Informatics and Computer Science Alliance (SICSA) under a ECIF grant.

References

Aarts, E. H., & Lenstra, J. K. (2003). *Local search in combinatorial optimization*. Princeton University Press.

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

Berbeglia, G., Cordeau, J.-F., & Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research*, *202*, 8–15.

Chao, I. et al. (2002). A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, *29*, 33–51.

Cordeau, J.-F., & Laporte, G. (2007). The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, *153*, 29–46.

Cordeau, J.-F., Laporte, G., & Ropke, S. (2008). Recent models and algorithms for one-to-one pickup and delivery problems. In B. Golden, S. Raghavan, & E. Wasil (Eds.), *The vehicle routing problem: Latest advances and new challenges* (pp. 327–357). Springer.

Dumas, Y., Desrosiers, J., & Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, *54*, 7–22.

Eglese, R. (1990). Simulated annealing: a tool for operational research. *European journal of operational research*, *46*, 271–281.

Frederickson, G. N., Hecht, M. S., & Kim, C. E. (1976). Approximation algorithms for some routing problems. In *17th Annual Symposium on Foundations of Computer Science* (pp. 216–227). IEEE.

Golden, B. L., Raghavan, S., & Wasil, E. A. (2008). *The Vehicle Routing Problem: Latest Advances and New Challenges: latest advances and new challenges* volume 43. Springer Science & Business Media.

Gribkovskaia, I., Laporte, G., & Shlopak, A. (2007). A tabu search heuristic for a routing problem arising in servicing of offshore oil and gas platforms. *Journal of the Operational Research Society*, *59*, 1449–1459.

Holland, J. H. (1973). Genetic algorithms and the optimal allocation of trials. *SIAM Journal on Computing*, *2*, 88–105.

Jula, H., Dessouky, M., Ioannou, P., & Chassiakos, A. (2005). Container movement by trucks in metropolitan networks: modeling and optimization. *Transportation Research Part E: Logistics and Transportation Review*, *41*, 235–259.

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, *34*, 975–986.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, *43*, 408–416.

Larsen, A., Madsen, O. B., & Solomon, M. M. (2004). The a priori dynamic traveling salesman problem with time windows. *Transportation Science*, *38*, 459–472.

Lund, K., Madsen, O. B., & Rygaard, J. M. (1996). *Vehicle routing problems with varying degrees of dynamism*. Technical Report Institute of Mathematical Modelling.

Mitrović-Minić, S., Krishnamurti, R., & Laporte, G. (2004). Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, *38*, 669–685.

Mitrović-Minić, S., & Laporte, G. (2004). Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, *38*, 635–655.

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

Nowak, M., Ergun, Ö., & White, C. C. (2008). Pickup and delivery with split loads. *Transportation Science*, *42*, 32–43.

Pankratz, G. (2005). Dynamic vehicle routing by means of a genetic algorithm. *International Journal of Physical Distribution & Logistics Management*, *35*, 362–383.

Pigden, T. (2013). Missing from the model: orders, drivers, tractors and trailers and non-linear loading. In *Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion* (pp. 1079–1084). ACM.

Pillac, V., Gendreau, M., Guéret, C., & Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research*, *225*, 1–11.

Pureza, V., & Laporte, G. (2008). Waiting and buffering strategies for the dynamic pickup and delivery problem with time windows. *INFOR: Information Systems and Operational Research*, *46*, 165–176.

Ralphs, T. K., Kopman, L., Pulleyblank, W. R., & Trotter, L. E. (2003). On the capacitated vehicle routing problem. *Mathematical Programming*, *94*, 343–359.

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.

Savelsbergh, M., & Sol, M. (1998). Drive: Dynamic routing of independent vehicles. *Operations Research*, *46*, 474–490.

Savelsbergh, M. W. (1985). Local search in routing problems with time windows. *Annals of Operations research*, *4*, 285–305.

Savelsbergh, M. W., & Sol, M. (1995). The general pickup and delivery problem. *Transportation science*, *29*, 17–29.

Schorpp, S. (2011). *Dynamic Fleet Management for International Truck Transportation: Focusing on Occasional Transportation Tasks*. Springer.

Semet, F., & Taillard, E. (1993). Solving real-life vehicle routing problems efficiently using tabu search. *Annals of Operations research*, *41*, 469–488.

Serpell, M., & Smith, J. E. (2010). Self-adaptation of mutation operator and probability for permutation representations in genetic algorithms. *Evolutionary Computation*, *18*, 491–514.

Villegas, J. G., Prins, C., Prodhon, C., Medaglia, A. L., & Velasco, N. (2013). A matheuristic for the truck and trailer routing problem. *European Journal of Operational Research*, .

Xiang, Z., Chu, C., & Chen, H. (2008). The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments. *European Journal of Operational Research*, *185*, 534–551.

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

Yang, J., Jaillet, P., & Mahmassani, H. (2004). Real-time multivehicle truckload pickup and delivery problems. *Transportation Science*, *38*, 135–148.

Zhang, R., Lu, J.-C., & Wang, D. (2014). Container drayage problem with flexible orders and its near real-time solution strategies. *Transportation Research Part E: Logistics and Transportation Review*, *61*, 235–251.

Zhang, R., Yun, W. Y., & Kopfer, H. (2010). Heuristic-based truck scheduling for inland container transportation. *OR spectrum*, *32*, 787–808.