A new drayage problem with different customer services and container requirements

Federica Bomboi*, Massimo Di Francesco*, Jonas Pruente**
*Department of Mathematics and Computer Science, University of Cagliari, Italy
**Fakultät für Mathematik, TU Dortmund, Germany
fbomboi@unica.it, mdifrance@unica.it, jonas.pruente@math.tu-dortmund.de

July 27, 2018

Abstract

This paper investigates a drayage problem generalizing a previously proposed, which is motivated by a case study of a real maritime carrier. To serve export and import customer requests in the hinterland of a port, a fleet of trucks able to carry one or two containers of the same size is adopted. The aim of this study is to enlarge the supply of the types of service to fulfill additional needs of the customers by considering two main aspects. As first, there exist overweight and special commodities, i.e. flammable goods, that cannot be carried together with other loads. Secondly, the customers want to choose whether the arriving container is immediately loaded or unloaded (stay-with service) or left at their location (drop & pick service). It is pointed out how the number of feasible routes is affected by the number and type of transportation requests. A Set Partitioning formulation and two scalable heuristics of route selecting are proposed. We present a way to estimate the running time of the algorithms so that the user can choose which algorithm to apply. The experimentation shows that the proposed enlargement of services can be managed to the optimum or with a small error.

Keywords: Vehicle Routing Problem, Drayage, Container transportation, Set Partitioning.

I. Introduction

Drayage is a popular research area in the field of transportation and logistics. It concerns the distribution of empty and loaded containers between intermodal facilities (e.g. ports or railway terminals), export and import customers. This paper is motivated by the case study of a maritime medium-sized carrier which adopts a fleet of trucks and containers based at a port to provide door-to-door transportation customer services in the landside, Ghezelsoflu et al. (2018) [3]. The aim of this study is to generalize the characteristics of the drayage problem stated in Ghezelsoflu et al. (2018) [3]. The main advance in their work was testing a new distribution policy for the carrier in which exporters can also be served after importers other than the street-turn methodology currently adopted by the carrier. The first additional characteristic considered in this paper is motivated by the reality of some special requests. It occurs that some container loads are not allowed to be carried with other loaded containers at the same time, e.g. containers containing flammable commodities like tobacco and alcohol or commodities with too large weight. Therefore, it is important for the carrier to meet these customer requests. For the sake of clarity, in this paper containers which are carrying these special loads are referred to as "special"
and the other containers are called "ordinary". Secondly, in this study the customer is allowed to choose between two ways of service, "stay-with" and "drop & pick". In most of the related literature, containers are left at customer locations by trucks bypassing packing and unpacking works. Since drivers do not wait for containers during loading and unloading operations, they can serve additional customers in the meanwhile. This type of service is known as drop & pick. In the stay-with service, drivers wait for containers during packing and unpacking operations and trucks carry the same container before and after the customer service. In Ghezelsoflu et al. (2018) [3], only the stay-with service is adopted. A paper in which stay-with and drop & pick are adopted together is Funke and Kopfer (2016) [2]. In their study, the decision of the way of service is not up to the customer like in this paper, but made by the algorithm. However, the method we present is flexible to manage instances in which a subset of customers require a fixed service type and for the others the algorithm can choose their service type. There exist various reasons for the customers to choose one of the two service types. Drop & pick services increase the flexibility of the customers because they can load or unload the container when it can be integrated optimally in their working schedule. Since drop & pick services last less time and can be integrated easier into existing routes, they can be reserved with less advance than stay-with services. Nevertheless, stay-with services need to be considered also because not all customers want to invest in a more expensive equipment that is necessary for drop & pick services. In line with the standards in many countries, the fleet of vehicles consists of trucks carrying one or two 24.5 ft containers. Empty containers need to be provided to and collected from customers in order to meet their transportation requests. Two ways to provide empty containers are regarded: they are picked up in the port or directly carried from importers to exporters according to the street-turn policy, Jula et al. (2006) [5], Deidda et al. (2008) [1]. In the next section, the most related literature is briefly presented and the main differences are pointed out. In section III, we propose a mathematical model for the problem statement and a Set Partitioning formulation is presented. The section ends with an analysis on how the types of request influence the number of feasible routes and therefore the running time. Two heuristics with scalable parameters to find either faster or more precise solutions are proposed in section IV. In section V, we compare the two heuristics with the exact solution in an extensive experimental evaluation. Section VI concludes.

II. Literature Review

This section is an overview of the most related literature. For a detailed survey about the literature on drayage problems, we suggest the recent papers Song et al. (2017) [8] and Ghezelsoflu et al. (2018) [3]. This work can be seen as an important extension of the paper of Ghezelsoflu et al. (2018) [3]. They regarded a drayage problem that is motivated by the case study of a real carrier. In their problem statement, the trucks are able to carry more than one container and all customers are served according to the stay-with service type. They showed some improvements in the current carrier policy solving the problem in which importers do not have the priority to be served before all exporters. The two additional characteristics of this study are the inclusion of drop&pick services and the introduction of special loads. The problem statement formulated by Funke and Kopfer (2016) [2] has a lot of similarities to ours because they use trucks which can carry more than one container and they consider both types of services. Despite this, there are some major differences. The most relevant difference is that they assume that every customer is prepared for a drop & pick service and therefore the cargo transportation company can decide during the optimization process which service they want to use. In our problem, to reach a more realistic setting, the customers are free to decide in advance if they want a stay-with or a drop & pick service, so that all the types of request are fixed before the optimization process. We consider
homogeneous containers of 24.5 ft size, instead they use 20 ft and 40 ft containers, which can not be coupled with any other container. Hence, it is not possible to model 40 ft containers as special containers because special containers can be coupled with empty containers. In addition to this, an other main difference is that they distinguish also between 20 ft and 40 ft empty containers. One more difference is that they introduce one depot for the storage of empty containers when necessary. In contrast to our drop & pick rules, they assume to end this kind of service on the same day by visiting the customer twice, either with the same truck or another. They solved the problem with a node-arc model for small instances. Ileri et al. (2006) considered stay-with and drop & pick orders that are fixed from the beginning like in our problem statement. They assumed that drivers start and end the working day without a trailer. Additionally, they introduced trailer pools, in which infinitely many empty containers are stored. Drivers can be one of two types-company driver (CD) and third party (TP). This influences the objective function that is minimizing the total cost for the company. The main difference to this paper is that they have only one container per truck. They used a Set Partitioning formulation and solved the relaxed problem with a column generation approach. Also Xue et al. (2014) regarded both types of service and assumed that the customer decides which service to take. They have different additional depots and only one container per truck is allowed. They solved the problem with an algorithm based on window partitions. In conclusion, in best of our knowledge, our problem statement is innovative and not directly comparable with the other problem statements in the literature. It integrates many realistic characteristics from different papers and is therefore worth to be regarded. In particular, the presented Set Partitioning formulation is very flexible, it can handle a lot of different constraints and can be used for various problem statements. Especially for this problem statement, it is more effective than other methods presented in the literature.

III. Modelling

In the first subsection, the sets of customer requests are defined and the feasible routes are illustrated. Next, the Set Partitioning model is presented and a discussion on the size of the realistic problem instances is provided.

i. Feasibility of routes

The set of customer service types can be defined as follows:

- $PE$: set of requests in which an empty container must be picked up from an importer during drop & pick service;
- $DE$: set of requests in which an empty container must be dropped to an exporter during drop & pick service.
- $PLO$: set of requests in which an ordinary loaded container must be picked up from an exporter during drop & pick service;
- $PLS$: set of requests in which a special loaded container must be picked up from an exporter during drop & pick service;
- $DLO$: set of requests in which an ordinary loaded container must be delivered to an importer during drop & pick service;
- $DLS$: set of requests in which a special loaded container must be delivered to an importer during drop & pick service.
- $PLSWO$: set of requests of exporters with stay-with services and ordinary containers;
• **PLSW S**: set of requests of exporters with stay-with services and special containers;
• **DL SWO**: set of requests of importers with stay-with services and ordinary containers;
• **D LSW S**: set of requests of importers with stay-with services and special containers.

In order to define the feasible sequences of visits among these types of requests, we describe the state of the truck slots after each type of customer request and the types of requests that can be served for each possible state. Let \( s_{rj}(i) \) be the state of truck slot \( j \) after visiting customer request \( i \) in route \( r \). Since trucks carry up to two containers, \( j \in \{1, 2\} \). The state of any container slot is one of the following entries: \{**Absent**, **ClosedO**, **ClosedS**, **Empty**, **InitialO**, **InitialS**\}, which are defined hereafter:

- \( s_{rj}(i) = Absent \) means that no container is placed in slot \( j \). It can be an initial state of the truck slot or the state after serving customer request \( i \in DE \cup DLS \cup DLO \). Next, truck slot \( j \) can be used to serve a request in \( PE \cup PLO \cup PLS \) in route \( r \);
- \( s_{rj}(i) = ClosedO \) means that an ordinary loaded container is put in slot \( j \) after serving an export request \( i \in PLSWO \cup PLO \). In this case, this slot cannot change the state any longer and cannot be used to serve any additional customer request in route \( r \);
- \( s_{rj}(i) = ClosedS \) means that a special loaded container is put in slot \( j \) after serving an export request \( i \in PLSWS \cup PLS \). Even in this case, this slot cannot change the state any longer along the route and cannot be used to serve any additional customer request;
- \( s_{rj}(i) = Empty \) means that the container in slot \( j \) is empty. It can be an initial state of the truck slot or the state after serving a customer request \( i \in DLSWO \cup DLSWS \cup PE \). Next, this truck slot \( j \) can be used to serve a request of set \( DE \cup PLSWO \cup PLSWS \) in route \( r \);
- \( s_{rj}(i) = InitialO \) means that an ordinary container in slot \( j \) is loaded to satisfy an importer request in set \( DLSWO \cup DLO \) in route \( r \). Next, truck slot \( j \) will be denoted by the state **Empty** or **Absent**;
- \( s_{rj}(i) = InitialS \) means that a special container in slot \( j \) is loaded to satisfy an importer request in set \( DLSWS \cup DLS \). Next, truck slot \( j \) will be denoted by the state **Empty** or **Absent**.

According to this notation, the possible initializations of the two container slots in a truck leaving the port are:

- InitialO & InitialO
- InitialO & Empty
- InitialO & Absent
- InitialS & Empty
- InitialS & Absent
- Empty & Empty
- Empty & Absent
- Absent & Absent

In the list of possible truck initializations it is taken into account that the special loaded containers can be carried alone in a truck or be coupled only with empty containers. Table summarizes the acceptable sequences of visit for each type of request, which is reported in the central column and denoted by **Type of request**. Column **Previous** shows which types of request can be visited before the considered one and the slot state that can be used to meet the considered type of request. Column **Next** shows which types of requests can be visited after the considered one or shows that the slot state is **ClosedO** or **ClosedS** after serving the considered type of request. For example, the first row shows the possible sequences \{**Absent**, **PLO**, **ClosedO**\}, \{**DLO**, **PLO**, **ClosedO**\},

In the list of possible truck initializations it is taken into account that the special loaded containers can be carried alone in a truck or be coupled only with empty containers. Table summarizes the acceptable sequences of visit for each type of request, which is reported in the central column and denoted by **Type of request**. Column **Previous** shows which types of request can be visited before the considered one and the slot state that can be used to meet the considered type of request. Column **Next** shows which types of requests can be visited after the considered one or shows that the slot state is **ClosedO** or **ClosedS** after serving the considered type of request. For example, the first row shows the possible sequences \{**Absent**, **PLO**, **ClosedO**\}, \{**DLO**, **PLO**, **ClosedO**\},
Table 1: Possible sequences of visits for each truck slot

<table>
<thead>
<tr>
<th>Previous</th>
<th>Type of request</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absent</td>
<td>PLO</td>
<td>ClosedO</td>
</tr>
<tr>
<td>DLO</td>
<td>(PLS)</td>
<td>(ClosedS)</td>
</tr>
<tr>
<td>DLS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>InitialO</td>
<td>(InitialS)</td>
<td>PLO</td>
</tr>
<tr>
<td>(InitialS)</td>
<td>(DLS)</td>
<td>PLS</td>
</tr>
<tr>
<td>Empty</td>
<td>DLSWO</td>
<td>ClosedO</td>
</tr>
<tr>
<td></td>
<td>(PLSWS)</td>
<td>(ClosedS)</td>
</tr>
<tr>
<td></td>
<td>PE</td>
<td></td>
</tr>
<tr>
<td>InitialO</td>
<td>(InitialS)</td>
<td>DE</td>
</tr>
<tr>
<td>(InitialS)</td>
<td>(DLSWS)</td>
<td>PLSWO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PLSWS</td>
</tr>
<tr>
<td>Absent</td>
<td>DLO</td>
<td>PLSWO</td>
</tr>
<tr>
<td>DLO</td>
<td>PE</td>
<td>PLSWS</td>
</tr>
<tr>
<td>DLS</td>
<td>DE</td>
<td></td>
</tr>
<tr>
<td>Empty</td>
<td>DLSWO</td>
<td>PLO</td>
</tr>
<tr>
<td>DLSWS</td>
<td>DE</td>
<td>PLS</td>
</tr>
<tr>
<td>PE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Changes in slot states

<table>
<thead>
<tr>
<th>Type</th>
<th>Slot Pre-condition</th>
<th>Pre-condition other slot</th>
<th>Slot Post-condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE</td>
<td>Absent</td>
<td>Irrelevant</td>
<td>Empty</td>
</tr>
<tr>
<td>DE</td>
<td>Empty</td>
<td>Irrelevant</td>
<td>Absent</td>
</tr>
<tr>
<td>PLO</td>
<td>Absent</td>
<td>not ClosedS - not InitialS</td>
<td>ClosedO</td>
</tr>
<tr>
<td>PLS</td>
<td>Absent</td>
<td>not ClosedO - not ClosedS - not InitialO - not InitialS</td>
<td>ClosedS</td>
</tr>
<tr>
<td>DLO</td>
<td>InitialO</td>
<td>Irrelevant</td>
<td>Absent</td>
</tr>
<tr>
<td>DLS</td>
<td>InitialS</td>
<td>Irrelevant</td>
<td>Absent</td>
</tr>
<tr>
<td>PLSWO</td>
<td>Empty</td>
<td>not ClosedS - not InitialS</td>
<td>ClosedO</td>
</tr>
<tr>
<td>PLSWS</td>
<td>Empty</td>
<td>not ClosedO - not ClosedS - not InitialO - not InitialS</td>
<td>ClosedS</td>
</tr>
<tr>
<td>DLSWO</td>
<td>InitialO</td>
<td>Irrelevant</td>
<td>Empty</td>
</tr>
<tr>
<td>DLSWS</td>
<td>InitialS</td>
<td>Irrelevant</td>
<td>Empty</td>
</tr>
</tbody>
</table>

{DLS, PLO, ClosedO}, {DE, PLO, ClosedO} for PLO.

To determine feasible routes for two-containers trucks, both container slots must be taken into account. Table 2 shows in column Slot pre-condition the required state of a slot in order to serve the type of request reported in the first column Type. For example, the state of a slot must be “Absent” in order to pick an empty container (i.e. serving a request of set PE) and put the container in that slot. In the column Pre-condition other slot the status of the other slot in the truck is indicated in order to accept the type of request reported in the first column. The string “Irrelevant” shows that the state of the other slot is independent from the ability to serve a request of set PE. Column Slot post-condition indicates the state of the used slot after the service of a customer request reported in the corresponding line. For example, the state of a slot must be “Empty” after picking up an empty container (i.e. serving a request of set PE). The status of the other slot does not change. To clarify, one can serve a request of set PLO by a truck slot if the state of the other slot is different from ClosedS and InitialS.

The set of feasible routes is obtained by connecting acceptable sequences of customer requests.
according to Table 1 and Table 2. Therefore a dynamic program over the number of customers in a route is used. To obtain routes with \( k+1 \) customers, all routes with \( k \) customers are regarded and for every customer it is checked if adding this customer is feasible. This has to be done for all possible truck initializations because there exist routes that are only feasible for one of the initializations.

ii. Mathematical model

We prefer the Set Partitioning model over the node-arc formulation for the presented problem statement because the problem is highly constrained and with a growing number of constraints the running time of the route generation and of the Set Partitioning model decreases in general, as opposed to node-arc formulations that become harder to solve. Given the previous set of feasible routes, we present a Set Partitioning model using the following notation:

\( R \): Set of all feasible routes

\( C \): Set of customer requests = \( PE \cup DE \cup PLS \cup PLO \cup DLS \cup DLO \cup PLSWS \cup PLSWO \cup DLSWS \cup DLSWO \)

\( d_r \): Cost route \( r \in R \);

\( a_{ir} \): Coefficient with value 1 if route \( r \in R \) covers \( i \in C \), 0 otherwise;

\( x_r \): Binary variable which is 1 if route \( r \in R \) is selected, 0 otherwise.

The Set Partitioning formulation of the model is:

\[
\begin{align*}
\min & \quad \sum_{r \in R} d_r x_r \\
\text{s.t.} & \quad \sum_{r \in R} a_{ir} x_r = 1 \quad \forall i \in C \\
& \quad x_r \in \{0, 1\} \quad \forall r \in R
\end{align*}
\]

In (1) the cost of the selected routes are minimized. Constraints (2) ensure that all container requests are met by exactly one route. Finally, (3) defines the domain of the decision variable.

iii. Parameters affecting the number of feasible routes

The ability of solving the previous model by a standard mixed-integer programming solver depends on the number of feasible routes. A general formula is derived to determine \( |R| \) as function of the types of requests, but it is too large to be reported in the paper. In the basic problem statement it is possible to compute the exact number of routes in less than 1 second. For the problem with time window constraints, 8 hours working day constraints and other special constraints deriving from the heuristics (section IV), we estimate the number of routes using sampling. The knowledge of the number of routes for a given instance is important to decide which algorithm to apply. More details about these formulas are provided in Appendix A. Nevertheless, the maximization of this function shows that, for a fixed number of customers, the number of feasible routes is maximum when half of the customers requests belongs to the set \( PE \) and the other half belongs to \( DE \). Indeed, a problem with a huge number of customers and less \( PE \) and \( DE \) requests could have a lower number of feasible routes than a problem with less customers...
but more PE and DE requests. Table 3 shows the number of the feasible routes for four instances with special constraints on the types of requests, reported in the column Type of request, in the case of up to 8 customers per route. In the first column the number of customers $|C|$ for every instance is indicated. The column $|R|_{\text{max}}$ represents the maximum number of feasible routes. In the first and third instances, the maximum is reached when half of the customers requests are PLO and the other half are DLO. In the second and fourth instances, the maximum is reached when half of the customers requests are PE and the other half are DE. This relevant role of PE and DE requests is also confirmed by Table 1. It shows that PE and DE requests result in a higher number of acceptable sequences of visits as opposed to the other types of requests.

### IV. Heuristics

In this section, two heuristics are proposed to select a subset of routes from the overall set $R$ of feasible routes. This subset of routes will replace $R$ in the model in section III and result in a restricted Set Partitioning problem. Therefore, these heuristics can be seen as rules to delete from $R$ routes which are not likely to appear in the optimal solution. The heuristics are described by the example in Table 4 where only 4 customer requests are considered. They are denoted by C1, C2, C3 and C4.

| $|C|$ | Type of request | $|R|_{\text{max}}$ |
|------|----------------|-----------------|
| 20   | without PE - DE | 20440           |
| 20   | half PE half DE  | 1.4 billion     |
| 100  | without PE - DE  | 12.5 million    |
| 100  | half PE half DE  | 50 trillion     |

Table 3: Example for number of feasible routes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type of request</th>
<th>Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>-</td>
<td>(0,0)</td>
</tr>
<tr>
<td>C1</td>
<td>PE</td>
<td>(0,1)</td>
</tr>
<tr>
<td>C2</td>
<td>DE</td>
<td>(1,1)</td>
</tr>
<tr>
<td>C3</td>
<td>DE</td>
<td>(1,0)</td>
</tr>
<tr>
<td>C4</td>
<td>PLS</td>
<td>(2,0)</td>
</tr>
</tbody>
</table>

Table 4: Example for the heuristics

- Heuristic 1: since customers of type PE and DE highly increase the total number of routes, the first heuristic requires a limit $\text{lim}$ on the number of customers of these types in each route. For instance, if $\text{lim} = 2$, only routes with at most 2 customers which have type PE or DE are regarded. Following this idea, one eliminated route is \{Port, C1, C2, C3, Port\} because this route contains three customers with type PE or DE, which is more than the allowed limit $\text{lim}$. A larger value of $\text{lim}$ leads to more precision but to a higher running time.

- Heuristic 2: for every customer request $i \in C$, we make a list of the closest $m \leq |C| - 2$ requests. Then, in the routes generation process, a candidate can only be added to the route only if it is in the list of the previous customer in the route. This is motivated by the fact that a high number of long distance routes, which are not likely in the optimal solution, are not regarded. To ensure feasibility, the port can always be added to a route and from the port it is allowed to reach every customer. In our example, see Table 4 if we set $m = 2$, then
one eliminated route is \{Port, C1, C4, Port\} because C4 is not one of the \(m\) closest neighbors of C1. A larger value of \(m\) leads to more precision but to a higher running time.

- Combination of Heuristic 1 and Heuristic 2: it applies both additional rules.

V. EXPERIMENTATION

The presented model is implemented in Java 1.8.0_171 using IBM ILOG CPLEX Studio 12.6.0.0. Tests have been run on an Intel(R) Xeon(R) E7340 processor with 2.4 GHz. The time limit is set to 2 hours and memory limit to 32 GB. Artificial instances are built by a square of 1000x1000 units, where we generate random coordinates of the customers according to the Uniform distribution. The port is placed at \((0,0)\). The first customers requests are labeled with type \(PE\) or \(DE\) according to how many \(PE\) or \(DE\) we want to have in the instances. Next, we set the request type of all other customers by the Uniform distribution. The service time is supposed to be 30 minutes for stay-with requests and 15 minutes for drop & pick requests.

The time-windows range is always 6 hours long and can start from 6:00 am till 2:00 pm. Their initial time is generated by the Uniform distribution. Drivers can start their routes at every time point, therefore, they start exactly at the time point that is necessary to fulfill time windows constraints. We think that letting the drivers have flexible starting time is more realistic. After their first stop, the drivers are not allowed to wait to be in time at the customer location.

Each route including the time of the service cannot take longer than 8 hours, which is a reasonable duration for a driver shift. In order to serve a stay-with customer with coordinates \((1000, 1000)\) within 8 hours, 1 minute must be equal to at least 6.29 units. Since this setting cuts too many routes and makes the problem too easy, we double it and set 1 minute to 12.58 units to obtain reasonable settings for the experimentation. According to this data, up to 8 customers can be visited in each route.

At first, instances with time-windows are considered. Although time-windows make more realistic instances, they decrease the number of feasible routes and make the Set Partitioning problem easier to solve. Next, harder problem instances without time-windows are tested. The results with and without time-windows are reported in Table 5 and Table 6 respectively. Every row concerns one group of 10 random instances. The first two columns describe the problem data: the number of customers requests \(|C|\), the number of requests of type \(PE\) and \(DE\). For example the first row stands for the setting with 20 customers in which one of them asks for a \(DE\) request, no customer requires a \(PE\) service and the other customers have random request types among the remaining 8 request types.

The results are summarized in five groups of columns. The first group of columns is denoted by All Routes and reports the case of complete enumeration of the overall set of feasible routes. The other groups of columns report the results obtained by four reasonable calibrations of the heuristics proposed in Section IV. More precisely, the second group of columns is denoted by \(lim = 1\) because it refers to the case of a restricted Set Partitioning problem in which each feasible route has one \(PE\) or \(DE\) request at most. Its associated number of routes is denoted by \(|R_1|\). The second group of columns is denoted by \(lim = 2\) because it refers to the case of a restricted Set Partitioning problem in which each feasible route has two \(PE\) or \(DE\) requests at most. Its associated number of routes is denoted by \(|R_2|\). The third group of columns is denoted by \(m = 0.75|C|\) because it refers to the case of a restricted Set Partitioning problem in which each customer request is connected to 75 percent of the other requests. Its associated number of routes
is denoted by $|R_3|$. Finally, the fourth group of columns is denoted by $lim = 2 \& m = 0.75|C|$ because it refers to the case of a restricted Set Partitioning problem in which each feasible route has two PE or DE requests at most and each customer request is connected to 75% of the other requests. Its associated number of routes is denoted by $|R_4|$.

The following results are reported in the case of enumeration of all feasible routes: the average number $|R|$ of routes, the average sum of the route generation time and the running time (it is denoted by time and measured in seconds), the average percentage of the route generation time out of the previous sum (it is denoted by gen. %) and number of solved instances out of ten (it is denoted by s.i.). For each heuristic, we report the average number of routes, the average ratio between the solution obtained by that heuristic and the solution determined with all feasible routes if possible or with the best heuristic otherwise, the average sum of route generation time and the running time (it is denoted by time and measured in seconds) and number of solved instances out of ten (it is denoted by s.i.). The notation uses k for 1000 and — for the cases in which it was not possible to obtain a solution.

In this experimentation 3000 instances were run. In 27 instances the proposed methods were not able to determine a feasible solution: 5 of these instances have time-windows, 22 do not. More precisely, 21 instances were not solved because of a lack of memory: in 9 of these instances it was not possible to generate all the routes, in the remaining 12 the lack of memory was observed during the running of the solver. Because of the time limit, 6 instances were not solved: 1 during the generation of the routes, 5 during the solver execution. According to this experimentation, 69.1% of the running time was used by Cplex and only 30.9% for the route generation. The percentage of route generation time was higher for the heuristics because of the additional time spent to check the feasibility of the routes.

In the case of complete enumeration of all routes, the instances with all feasible routes were always solved to the optimality or considered as not solved instances, thus they are a valid benchmark for the quality of solutions determined by the heuristics. According to Table 5 and Table 6, all the heuristics are much faster as opposed to the case of complete enumeration of all feasible routes, without losing much precision. The comparison between $lim = 1$ and $lim = 2$ shows that the first is faster and does not lead to better results in all instances. This is not
The table shows results for all methods without time windows. Surprising because the first heuristic uses a subset of routes of the second one. In the row with 100 customers and 2PE 2DE of Table 5, the solution time of 411 seconds for the heuristic with lim = 1 is obtained, even if 3632 seconds are spent to solve an outlier instance. The comparison between the heuristic with lim = 2 and the heuristic with m = 0.75|C| is more interesting: the last one is in general slower than the first one if there are more than 3 PE or DE customers in the instance and in the other cases is faster. Yet, this result is somehow expected because an increase in PE or DE customer requests does not affect the heuristic with lim = 2 crucially. Nevertheless, both heuristics lead to optimal and near-optimal solutions. Since the last heuristic incorporates the two heuristics, the solutions cannot be better than those obtained by the previous two heuristics, but the time is clearly lower of the time of both. Nevertheless, the solution quality keeps being good. Table 6 shows the results for the same instances without time windows. For all settings, the number of routes increases, but the same comments made in the case of time-windows still hold.

Finally, during the 3000 tests we tested the formula mentioned in Section III. The main goal of this formula is to predict the running time and the memory of a solution method for a given instance before running it by estimating the number of feasible routes for that instance. The number of routes estimated by the formula is compared to the number of generated routes. The average quotient between these two numbers was 1.000 in all tests. The average time of the estimation over all tests was 4.18 seconds and, in the case of instances with 10 million or more routes, the average time was 47.169 seconds. The maximum time the formula needed was 88 seconds. Since the correlation coefficient between the running time and the estimated number of routes was 0.813 (0.945 if we delete the outlier instance) and 0.991 between the memory and the estimated number of routes, we conclude that the formula is very effective. For the correlation coefficient about time, we considered only instances with at least 1 second running time because the other instances were so fast that they did not need a prediction and, because of the measurement in seconds, they result in distorted statistics.
VI. Conclusion

The distribution of containers between customers and ports is not a new issue but this paper faces a drayage problem with a number of new characteristics that are relevant for practical applications. Although trucks can carry up to two 24.5 ft ordinary containers, no paper has focused on both ordinary and special containers, which are requested to be carried one at a time. Moreover, most of the research concerns stay-with or drop & pick services, but limited attention has been devoted to their integration, as customers can typically choose between both service types. We explicitly enumerated the set of feasible routes in a short time and used a Set Partitioning formulation to determine the routes serving each customer request and minimizing routing costs. The presented method is very flexible and can be used for all problem statements in which the feasibility of a route can be decided fast. Since the derived number of feasible routes may be too large for a Mixed-Integer programming solver, two heuristics are proposed to select subsets of promising routes and solve restricted Set Partitioning models. The heuristics can be tuned: one could cut few routes in order to obtain higher quality solutions in longer running times, or cut many routes in order to determine lower quality solutions in a shorter running time or set possible tradeoffs. The experimentation shows that the proposed heuristics are always able to determine optimal and near-optimal solutions much faster than the exact solution method. A formula to estimate the overall number of feasible routes precisely in short time is presented. It can be adapted to various problem statements with different constraints. The number of feasible routes highly correlates with the running time of the algorithm and therefore with this information the user can decide quickly which method to apply.
A. Estimating the number of feasible routes

The possibility to solve the Set Partitioning problem by the algorithms of Mixed-Integer Programming solvers mostly depends on the number of feasible routes. Therefore, it is very useful to determine this number before solving any problem instance, in order to tune the heuristics and control the problem size. In this study, a formula is derived to determine the number of feasible routes. Generally speaking, the computation of the number of routes by this formula is very fast. The formula has 74 summands. The choice not to sum some similar summands (for examples the summands in bold) is due to the fact that these summands are derived from different routes.

\[
|R| = 3|DLO| + 3|DLSWO| + 3|DLO| \cdot |PE| + 3|DLO| \cdot |PLO| + |DLO|(|DLO| - 1) + \\
|DLO| \cdot |DLSWO| + 3|DLSWO| \cdot |DE| + |DLSWO| \cdot |DLO| + 3|DLSWO| \cdot |PLSWO| + \\
|DLSWO|(|DLSWO| - 1) + 4|PE| + 3|PLO| + 4|PE| \cdot |DE| + |PE| \cdot |DLO| + \\
3|PE| \cdot |PLSWO| + |PE| \cdot |DLSWO| + |PLO| \cdot |DLO| + |PLO| \cdot |DLSWO| + \\
2|DLO| \cdot |PLS| + |DLSWO| \cdot |PE| + |DLSWO| \cdot |PLO| + |DLSWO| \cdot |PLS| + \\
2|DLSWO| \cdot |PLSWS| + 4|DE| + 3|PLSWO| + 4|DE| \cdot |PE| + 3|DE| \cdot |PLO| + \\
|DE| \cdot |DLO| + |DE| \cdot |DLSWO| + |DLO| \cdot |DE| + |DLO| \cdot |PLSWO| + |DLO| \cdot |PLSWS| + \\
|PLSWO| \cdot |DLO| + |PLSWO| \cdot |DLSWO| + 2|PLS| + |PE|(|PE| - 1) + |PE| \cdot |PLO| + \\
|PE| \cdot |PLS| + 2|PE| \cdot |PLSWS| + |PLO| \cdot |PE| + |PLO|(|PLO| - 1) + |PLS| \cdot |PE| + \\
2|PLSWS| + 2|DE| \cdot |PLS| + |PLO| \cdot |DE| + |PLO| \cdot |PLSWO| + |PLS| \cdot |DE| + \\
|PLSWO| \cdot |PE| + |PLSWO| \cdot |PLO| + |PLSWO| \cdot |PE| + |DE|(|DE| - 1) + \\
|DE| \cdot |PLSWO| + |DE| \cdot |PLSWS| + |PLSWO| \cdot |DE| + |PLSWO|(|PLSWO| - 1) + \\
|PLSWS| \cdot |DE| + 2|DLS| + 2|DLSWO| + |PE| \cdot |DLS| + |PE| \cdot |DLSW| + \\
2|DLS| \cdot |PE| + 2|DLS| \cdot |PLSWO| + 2|DLS| \cdot |PLS| + |DLSWO| \cdot |PE| + \\
2|DLSWO| \cdot |DE| + |DLSWO| \cdot |DLO| + |DLSWO| \cdot |PLS| + 2|DLSWO| \cdot |PLSWO| + \\
2|DLSWO| \cdot |PLSW| + |DE| \cdot |DLS| + |DE| \cdot |DLS| + |DLS| \cdot |DE| + \\
|DLS| \cdot |PLSWO| + |DLS| \cdot |PLSWS|.
\]

This formula has 74 summands. The choice not to sum some similar summands (for examples the summands in bold) is due to the fact that these summands are derived from different routes. Constraints depending explicitly or implicitly on the customer coordinates due to the time cannot be modelled exactly in the formula. In our problem statement we have three constraints of this type: the 8 hours working
day constraint, the time-windows constraints and the constraints of the second heuristic. For these constraints, we want to find a solution that give us a good estimation of the number of routes in a short time. To deal with this problem, we calibrate the probability $p_s$ that every summand $s$ appears in the formula. The calibration is performed in a sample of $k$ routes by the computation between the ratio between the number of routes meeting all additional requirements and the number of routes computed by the formula. If $k$ is larger, the estimation of $p_s$ is better. In our test we set $k$ to 10000. If the value of the summand $s$ is less than $k$, instead of simulating we just enumerate all the feasible routes. This makes the estimation better and decreases the running time. To solve this problem, we need to multiply every summand $s$ in the formula with the probability $p_s$ that a route made by the sequences of types represented by this summand is feasible. To make it clear we want to adapt the formula shown before for the first five summands to the new constraints:

$$|R| = 3p_1|DLO| + 3p_2|DLSWO| + 3p_3|DLO| \cdot |PE| + 3p_4|DLO| \cdot |PLO| + p_5|DLO|((|DLO| - 1) + \ldots$$

This formula can be used for all kinds of problem statements in which it is possible to decide the feasibility of a route fast.

**ACKNOWLEDGEMENT**

Federica Bomboi gratefully acknowledges Sardinia Regional Government for the financial support of her PhD scholarship (P.O.R. Sardegna F.S.E. Operational Programme of the Autonomous Region of Sardinia, European Social Fund 2007-2013 - Axis IV Human Resources, Objective l.3, Line of Activity l.3.1.).
REFERENCES


