A SATELLITE NAVIGATION SYSTEM TO IMPROVE THE MANAGEMENT OF INTERMODAL DRAYAGE

Alejandro Escudero Santana. University of Seville, Spain. aescudero@esi.us.es
Jesús Muñuzuri Sanz. University of Seville, Spain. munuzuri@esi.us.es
Carlos Arango Pastrana. University of Seville, Spain. cap@esi.us.es
Luis Onieva Giménez. University of Seville, Spain. onieva@esi.us.es

ABSTRACT

The intermodal transport chain can become more efficient by means of a good organization of the drayage movements. Drayage in intermodal container terminals involves the pick up or delivery of containers at customer locations, and the main objective is normally the assignment of transportation tasks to the different vehicles, often with the presence of time windows. The literature shows some works on centralised drayage management, but most of them consider the problem only from a static and deterministic perspective, whereas the work we present here incorporates the knowledge of the real-time position of the vehicles, which permanently enables the planner to reassign tasks in case the problem conditions change. This exact knowledge of position of the vehicles is possible thanks to a geographic positioning system by satellite (GPS, Galileo, Glonass), and the results show that this additional data can be used to dynamically improve the solution.

Keywords: Intermodal transport, drayage, real time assignment, stochastic transit time

INTRODUCTION

Road transport has been and continues to be prevalent for the movement of freight. However, increasing road congestion and the necessity to find more sustainable means of transport have pushed different governments to promote inter-modality as an alternative. However, for inter-modality to become viable for trips shorter than 700 km, a cost reduction is necessary, and the interest focuses on the road segment inherent to all the intermodal chains. These initial and final road trips, also called drayage operations, represent 40% of the intermodal transport costs. There is therefore potential to overcome this disadvantage and make intermodal transport more competitive through proper planning of drayage operations (Spasovic and Morlok, 1993). The drayage problem (DP) can be considered a multi-resource routing problem (MRRP), which involves the routing of multiple resources to perform a series of tasks. MRRPs can also be modelled as pick up and delivery problems (PDP) with vehicle capacity equal to one. A review of the general pickup and delivery problem (PDP) is presented in Savelsbergh and Sol (1995).

An interesting work is shown in Ball et al (1983), transforming the problem of the allocation of trailers for a chemical company into a vehicle routing problem (VRP). The origin and destination of a movement are considered a single node that represents the entire movement with all the characteristics of the movement (duration, origin, destination and time windows). De Meulemeester et al (1997) and Bodin et al (2000) apply this transformation to a VRP with full load trailers. Following this path, the number of references on centralised drayage management has increased significantly over the last years, but most of them consider the problem only from a static and deterministic perspective. The main objective is normally the assignment of transportation tasks to the different vehicles, often with the presence of time windows (Wang and Regan, 2002). The first part of the work by Cheung and Hang (2003) develops a deterministic model with time windows, which is then solved by means of the discretization of time, and by incorporating the concept of fictitious tasks for the beginning and the end of the day for the vehicle. Ileri et al (2006) cover a large number of task types, both simple and combined, and of costs involved in drayage operations, and solve the problem with a column generation method. Smilovik (2006) and Francis et al (2007) incorporate flexible tasks where either only the origin or the destination is precisely known. Caris and Janssens (2009) propose a two-phase insertion heuristic to construct an initial solution that is then improved with a local search heuristic. A general revision of scientific contributions to the drayage problem can be found in Erera et al (2008).

Many works also include randomness in the generation of tasks (Bent and Van Hentenryck, 2004; Bertsimas, 1992; Gendreau et al, 1995) or dynamism in their assignment (Bent and Van Hentenryck, 2004; Psaraftis, 1995; Wang et al, 2007). Cheung and Hang (2003) and Cheung et al (2005) do consider the dynamic and stochastic characteristics of the drayage problem and solve it with a rolling window heuristic, but this randomness only affects the duration of the task, and not the displacement time between different tasks. It is uncommon to find randomness in trip times (Laporte et al, 1992), which is particularly appropriate when the intermodal terminal requiring drayage operations is close to a large urban centre.

Our work considers the real-time knowledge of the position of the vehicles, which permanently enables the planner to dynamically reassign tasks. This exact knowledge of position of the vehicles is possible thanks to a geographic positioning system by satellite (GPS, Galileo, Glonass), which can provide real-time data for dynamic recalculations. An approach based on re-optimization is proposed, where a snapshot of the state of each task and the position of each vehicle will be taken before recalculating. A DP with stochastic transit times is then solved by means of an iterative optimization algorithm, modifying if necessary the scheduling of the drayage fleet.

The paper is organized as follows. Section 2 is the problem definition, and a description of the real-time optimization procedure is given in section 3. Section 4 presents an insertion heuristic to solve a snapshot of the problem, and the results of a series of numerical experiments are shown in section 5.

PROBLEM DEFINITION

This paper discusses the drayage problem with time windows and stochastic transit times. This problem seeks to optimise the routing and scheduling of vehicles where the tasks involve the pick up or delivery of containers that need to be carried between an intermodal terminal and a set of customers or vice versa. The intermodal terminal is open during a specified time window, and all the tasks, also with time windows associated, are known in advance. However, the stochastic transit times make it impossible to foresee the exact duration of the trips related to the completion of tasks. A glossary of the notation used in this work is shown in the appendix.

This problem can be formulated in terms of a vehicle routing problem with time windows (VRPTW) with full container loads. Assuming homogeneous container types and sizes, the problem is to find the assignment of tasks to vehicles, in order to minimize the total cost of completing all the tasks. All vehicles $v \in V$ have to return to the terminal before the end of their depot windows, where the depot window corresponds to the working day. A fixed vehicle cost c_v is included to penalize the use of additional vehicles, and travelling costs c_{km} per unit of distance travelled are also considered. Also, since transit times are stochastic there exists the possibility to arrive outside the time window, and the corresponding penalty costs are applied. This penalty cost can be proportional to the waiting time, with the cost per time unit equal to c_w, or much higher, c_l, if the delay results in missing the train or ship. The total cost is the sum of fixed, travelling and penalty costs.

Since the procedure to solve the drayage problem is based on re-optimization, a snapshot of the real situation of the problem is taken prior to each recalculation. In each re-optimization instant $time$, a graph is defined, $G^{time} = (T, A)$, where T represents the set of tasks and A is the set of links between tasks. The $time$ variable represents an instant in the day, the existing scenario at the point where the problem solution is re-optimized, and this graph symbolizes the state of vehicles and tasks. Each task is composed of an origin, a destination, service times in the origin and destination, the transit time between origin and destination, the time windows at the origin and destination, and the penalty cost in case the time window is missed. Tasks can be either delivery or pickup operations.

Let $i \in T^D$ be a delivery task. These tasks involve picking up a container in the terminal (origin) and delivering it at a customer location (destination). The distance between origin and destination is d_i, and the transit time is τ_i. Service times apply both at the origin and destination, s_i^O and s_i^D, and the time window is associated to the origin $[E_i^O, L_i^O]$, with the destination time window permanently open. This time window, shown in the upper side of Figure 1, is hard at the start to represent that a delivery task can never start before the arrival of the train or vessel (red interval), and soft at the end to represent that if the drayage driver is late the task can still be completed, but a given amount will have to be paid for the time the container spends waiting at the terminal (yellow interval), c_w per minute.

On the other hand, let $j \in T^P$ be a pickup task, which consists of picking up a container at a customer location (origin) and carrying it to the terminal (destination). The distance between origin and destination and the transit time are d_j and τ_j, respectively. The service time in the origin and destination are s_j^O and s_j^D, and the time window $[E_j^D, L_j^D]$ is now associated to the destination. As shown in the lower side of Figure 1, this time window is hard at the end, since, if the task is completed after the established time window, the container misses its train or vessel, incurring in the corresponding penalty cost (orange interval). However, the time window can be soft at the start, given that, if the container is delivered in the terminal before the established time window, a waiting cost fee (yellow interval) will be charged for the extra time the container spends in the terminal.

The set A is defined as the set of links between the destination of each task and the origin of the other tasks, where each link is characterized by a distance d_{ij} and a transit time τ_{ij}. In our model the origin and destination of a task are aggregated into a single node that represents the entire movement with all the characteristics of the task (origin, destination, time windows, transit time,…), so the set A is asymmetric, that is, distance d_{ij} is generally different from distance d_{ji}.

Attention must be paid to the fact that, when re-optimizing, each vehicle in the fleet could be either in the depot, in the middle of completing a task, travelling towards the origin of a task, or idle. To account for this fact, the algorithm generates fictitious tasks T_f^{ini} to represent every vehicle state, and each vehicle is assigned to a fictitious task before re-optimization. For example, if in a re-optimization instant a vehicle is performing a task, its fictitious task is characterized by: an origin, which is the current position of the vehicle; a destination, which is the destination of the task being performed; an origin service time equal to zero and a destination service time equal to the one for the task being performed; a time window $[E_i^O, L_i^O] = [time, time]$ at the origin, and another one at the destination, which is the same destination time window of the task that is being performed. Since $[E_i^O, L_i^O] = [time, time]$, with $time$ being the current re-optimization instant, the new solution is forced to go on with the task without pause. If, on the other hand, a vehicle is travelling towards the origin of a task, its fictitious task is characterized by the same origin and destination, which correspond to the current position of the vehicle, service times equal to zero and time windows that force the truck to leave its current position after the recalculation. If $[E_i^O, L_i^O] = [time, time]$ and $[E_i^D, L_i^D] = [time, time]$, the vehicle will complete this fictitious task immediately after the recalculation and be able to proceed to another task, which will be the same one it was travelling to initially, or a different one.

Other fictitious tasks are also defined to determine the depot time window of every truck, T_f^{end}. Their origin and destination correspond to the depot position, their service times are 0, and their time windows match the depot time window.

$$T = T^D \cup T^P \cup T_f^{ini} \cup T_f^{end}$$
$$T^D \cap T^P = \varnothing$$

The set T is therefore composed of four different sub-sets, which really represent only two kinds of tasks, because fictitious tasks are only created to store vehicle information at the re-optimization instant.

Here is Figure 1

Figure 1 – Time Windows.
Above: Delivery Task. The arrow means the arrival of the train or vessel to the terminal.
Below: Pick-Up Task. The arrow means the departure of the train or vessel from the terminal.
On both figures: The green interval corresponds to an open time window, the yellow interval is a soft penalty period, the orange interval is a hard penalty period and the red interval is the impossible period.

REAL-TIME VEHICLE ROUTING

We use a dynamic methodology to improve the solution iteratively. We run an optimization algorithm at the beginning of the day fixing the expected values of the transit times, and obtain an initial solution. However, since transit times are stochastic it is impossible to know beforehand the exact time required to complete a task. Thus, every time that an event takes place the algorithm is run again considering the updated data only for the remaining, pending tasks. A simplified scheme of the process is represented in Figure 2.

Here is Figure 2

Figure 2 –Dynamism scheme

There are some options related to when this re-optimization search should be run:

1. Every fixed time.
2. When a task is accomplished.
3. When a vehicle is delayed of its expected position.

In this work, the re-optimization process is run both when a task is accomplished and every fixed time. When this re-optimization takes place, a snapshot of the system is taken and the state of every vehicle is verified, providing the system with information about the current position of the vehicles. This information lets the system know if a vehicle is FREE, BUSY or ASSIGNED. A truck is busy if it is performing a task in that moment; it is assigned if it is going to the origin of a task; and it is free if it is already in the depot or going there. Furthermore, the system has information about the trucks that have already been used and those that have not, USEDVEH and NONUSEDVEH; and about those that are in the depot, DEPOTVEH. Once the state of the vehicles state is verified, the state of the tasks is also updated, with three possible values: PENDING, IN PROCESS and FINISHED. A pending task is defined as the task which has not begun to be performed; when a truck is carrying out a task, it is considered a task in process; and finally, finished tasks are the ones that have already been completed.

After some time, given the stochastic nature of transit times, the operating scheduling program might not be the best one anymore, and the re-optimization process allows the system to correct delays in the execution of tasks, considering the new modified scenario. The new solution could change the allocation of tasks to trucks, keeping in mind that the busy vehicles must finish their tasks in process to be able to begin with a new task. This issue is solved with the definition of fictitious tasks, which contain all the information about tasks in process, as explained above. As an example, let us consider the case of two vehicles and six tasks, with the initial scheduling of vehicles as follows:

VEH 1: Ini (Depot) → 1 → 2 → 4 → 6 → End (Depot)
VEH 2: Ini (Depot) → 3 → 5 → End (Depot)

Vehicles start the day at the depot and they have to complete their allocated tasks and return to the depot. Then, let us assume that, when the first re-optimization is due, the first vehicle is performing the first task and the second one is still travelling towards the origin of the third task, which has not begun yet. But the traffic conditions have changed, and the first vehicle is delayed to the extent that it will be difficult for it to perform the second task on time. The re-optimization process might then result in a new solution like this one:

VEH 1: Ini (in 1) → 3 → 4 → 6 → End (Depot)
VEH 2: Ini (-) → 2 → 5 → End (Depot)

Now, as the first vehicle was busy when the re-optimization was done, it must finish the task in progress, and then go to the third task. The second vehicle, which was going to the third task, will change its destination an go to the second task.

The pseudo-code of all this dynamic methodology can be written as follows:

```
Dynamic Methodology
    1. If re-optimization event happens.
        1.1. Yes, then go to 2.
        1.2. No, then go to 7.
    2. Read the position of the vehicle.
    3. Check the task (PENDING, IN PROCESS, and FINISHED) and vehicle state (FREE,
       BUSY, and ASSIGNED).
    4. Update the initial fictitious tasks with the acquired information in 2 and 3.
    5. Re-optimization (Assignment algorithm).
    6. Update all the variables about the vehicle state (FREE, BUSY, ASSIGNED, USEDVEH,
       NONUSEDVEH, and DEPOTVEH).
    7. Go to 1.
```

The assignment algorithm, which is used to solve every re-optimization snapshot of the system, is an adaptation of the insertion heuristic developed by Caris and Janssens (2009) to the real-time drayage problem. Although the transit time is a stochastic variable, our algorithm uses its expected value (t_i, t_j and t_{ij}) to obtain fast convergence. The idea is to solve the stochastic transit time problem through iterative optimization, based on the knowledge of the position of the vehicle fleet. The necessary changes to adapt this heuristic to a stochastic environment are explained in the next section.

REAL-TIME INSERTION HEURISTIC

In this work, we have adapted the insertion heuristic of Caris and Janssens (2009) to the real-time drayage problem. Although the foundations are similar, there are important differences between both approaches. While Caris and Janssens (2009) define time windows at the customers, we considered it more appropriate to define time windows for the terminal. Also, they solve the problem from a static and deterministic point of view, where all the vehicles start in the depot and their starting time corresponds to the beginning of the day, while in our case the vehicle could be anywhere where the re-optimization process starts, and the time could be any time, and if a vehicle is carrying out a task when a re-optimization starts, it has to complete it before moving on to a new one. The fact that our algorithm must satisfy these conditions forced us to introduce fictitious tasks.

The heuristic procedure is based on the savings obtained through the merging of single pickup and delivery tasks. If single tasks are completed individually, 50% of the journey is wasted in empty trips, as shown in Figure 3a. However, the combination of delivery and pickup tasks may lead to a significant reduction in the total distance travelled, as represented in Figure 3b. Given that in the real-time drayage problem the assignment of tasks to vehicles is dynamic, vehicles could be anywhere, which is why their position is regarded as a fictitious task that could be combined with real tasks looking for cost savings. Nevertheless, in the case of fictitious tasks, savings will be obtained only if they are merged with a pickup task (see Figure 3c).

Here is Figure 3

Figure 3 – Merging trips

The insertion heuristic is divided into two phases. In the first one, all the possible combinations between tasks (including fictitious tasks) are analyzed, and the best combinations are selected. In the second phase, the combined tasks are inserted into routes. The heuristic procedure thus finds an initial solution, which is later improved by three local searches.

Pairing single tasks

As shown in Figure 3, the combination of two single tasks in merged tasks could represent large cost savings. These pairs will be formed by a first initial fictitious or delivery task, $i \in T^D \cup T_f^{ini}$, and a second pick up or end fictitious task, $j \in T^P \cup T_f^{end}$. Any other combination of tasks entails no cost savings.

It is necessary to consider that not all pairs of tasks can be combined into feasible merged tasks. On one hand, due to the existence of time windows, and since the second task of the pair must be reached after the completion of the first task, not all the pairs are possible. And on the other hand, the waiting time between the two single tasks is limited to a maximum amount $MAXWAIT$, and a pair is refused if its minimum waiting time $MINWAIT_{ij}$ is higher than $MAXWAIT$. This limit is related to the inefficiency of large waiting times between tasks: if a vehicle must wait too long between the completion of the first task in a pair and the beginning of the second task in the same pair, too much time from the working day of this vehicle will be wasted, increasing the need for more vehicles in the operating fleet. $MINWAIT_{ij}$ is defined as the minimal waiting time between the two tasks of pair (i,j):

$$MINWAIT_{ij} = \max\left(0, E_j^D - \left(\max(time, L_i^O) + s_i^O + t_i + s_i^D + t_{ij} + s_j^O + t_j\right)\right)$$

These two conditions limit the composition of feasible pairs. The corresponding equations are shown below, the first one related to the time windows restriction, and the second one to the maximum waiting time restriction.

$$\max(time, E_i^O) + s_i^O + t_i + s_i^D + t_{ij} + s_j^O + t_j \leq L_j^D$$
$$E_j^D - \left(\max(time, L_i^O) + s_i^O + t_i + s_i^D + t_{ij} + s_j^O + t_j\right) \leq MAXWAIT$$

Once the feasible pairs are found, they are evaluated and ranked according to the savings in travel distance generated, according to the expression $d_i + d_j - d_{ij}$. Caris and Janssens (2009) use the saving in travel time but, since the transit time is random in our case, we chose distance instead.

The pairs of feasible tasks generated are then selected in descending order of the savings produced, and all the pairs containing either of the tasks of the selected pair is deleted from the list. The process of pairing tasks up is repeated until no feasible combination exists in the list, and the remaining tasks are assigned to individual trips and form an imaginary pair with a dummy customer.

Every pair has an earliest and a latest start time, E_{ij} and L_{ij}. The earliest starting time E_{ij} is defined as the earliest time a vehicle can start to complete a pair (i, j) without unnecessary waiting between tasks *i* and *j*, and the latest start time is the latest instant in which a vehicle could begin to complete the pair (i, j), given that, if this pair is started later, then the tasks will not be reached on time.

$$E_{ij} = \begin{cases} L_i^O & \text{if } E_j^D - t_j - s_j^O - t_{ij} - s_i^D - t_i - s_i^O > L_i^O \\ E_j^D - t_j - s_j^O - t_{ij} - s_i^D - t_i - s_i^O & \text{if } E_i^O \leq E_j^D - t_j - s_j^O - t_{ij} - s_i^D - t_i - s_i^O \leq L_i^O \\ E_i^O & \text{if } E_j^D - t_j - s_j^O - t_{ij} - s_i^D - t_i - s_i^O \leq E_i^O \end{cases}$$

$$L_{ij} = \min\left(L_i^O, L_j^D - t_j - s_j^O - t_{ij} - s_i^D - t_i - s_i^O\right)$$

The necessary time to complete a pair, RS_{ij}, is the sum of travel times, service times and the minimum waiting time. For this evaluation, the values of travel times are considered deterministic and average values are used.

$$RS_{ij} = s_i^O + t_i + s_i^D + t_{ij} + s_j^O + t_j + s_j^D + MINWAIT_{ij}$$

Route Construction

A route is created for each vehicle, composed of pairs of tasks, where the first pair of every route is the one containing the initial fictitious task assigned to the corresponding vehicle. The other pairs are assigned sequentially, with vehicles with lower costs used first and with pairs of tasks selected to be inserted into routes in increasing order of their latest start time L_{ij}. A pair can be inserted into a route $v \in V$ if two conditions are complied: the first condition is that vehicle v must be able to begin to complete the pair before L_{ij}, and the second condition establishes that vehicle v must return to the depot within its depot window.

$$\max(time, RS_v) \leq L_{ij}$$

$$\max(time, RS_v, E_i^O) + RS_{ij} \leq T_v$$

The route service time RS_v is initially set to $RS_v^{fictitious}$. If a pair can be inserted into a route of a USEDVEH, the route is updated and its route service time is increased. A new vehicle will be used otherwise. It is possible that no vehicle can serve the pair on time, even using the idle vehicles remaining in the depot, in which case the pair will be assigned to the vehicle that can arrive to start the pair with the smallest delay.

$$RS_v \leftarrow \max(time, RS_v, E_{ij}) + RS_{ij}$$

If a route only consists of a fictitious task after the construction procedure and $RS_v^{fictitious}$ is 0, this means that the route is really unused, and the vehicle will be in the depot all time. If the route is only composed by a fictitious task but $RS_v^{fictitious}$ is not 0, then the vehicle is performing a task, has not any other one assigned, and will return to the depot afterwards.

Local Search Algorithm

Three local searches are proposed to improve the initial solution obtained by the aforementioned heuristic: the CROSS, COMBINE and INSERT operators. The sequence in which these operators are implemented is CROSS → COMBINE → INSERT. First, the CROSS operator is applied to find the best combinations of pairs. Then, the COMBINE and INSERT operators try to reduce the number of new vehicles to incorporate. These COMBINE and INSERTION operators must try to reduce only the routes assigned to vehicles that have not been used yet, since, if the vehicle was used previously, the fact of deleting its route will not result in any cost savings.

CROSS operator

Two pairs are selected, (g, h) and (i, j). These pairs are crossed, resulting in two new pairs, (g, j) and (i, h), and the feasibility of these new pairs is checked. If both are feasible, the algorithm checks whether the new pairs can be reinserted into the routes. Then two possible combinations are analyzed: (g, j) inserted into the first route and (i, h) into the second route; or (i, h) inserted into the first route and (g, j) into the second route. If a combination is viable, this is added to the list of possible CROSS movements. The improvement produced by a CROSS movement is:

$$I_{ghij} = d_{gh} + d_{ij} - d_{gj} - d_{ih} - VR_{ghij}$$

where VR_{ghij} is the improvement due to the reduction of the number of necessary trucks. This improvement is feasible if a resulting route only contains dummy tasks and its allocated vehicle has never been used before, which means that it is registered in NONUSEDVEH.

The CROSS local search stops after a number of iterations without reduction in the total expected cost.

COMBINE operator

The algorithm also checks whether the pairs served by two different vehicles can be combined into a single route. The operator tries to combine the routes of the trucks that have not begun to work, NOUSEDVEH, with other routes, in order to seek a reduction in the number of new trucks to be used. Two routes can be combined if the last pair of the route to combine can be completed before the latest starting time of the route it is combined with.

INSERT operator

This operator removes pairs from certain routes and reinserts them into other routes. Like the COMBINE operator, the INSERT operator tries to reduce the number of new vehicles, and thus the potential routes to remove and reinsert are the routes assigned to the vehicles that have not been used yet.

TESTING AND RESULTS

Heuristics applicable to the standard VRPTW are compared by making use of the set of benchmark problems presented in Solomon (1987) or other test sets. This problem, however, is different to these standard benchmark problems because in the drayage problem the vehicle is either empty of fully loaded, which calls for a new experimental design. Two problem characteristics are selected: the dispersion of customers and the width of the customer time windows. For each characteristic we selected a low and high level, and generated ten problems instances in each problem class, which meant that a total forty instances were tested. Each test was composed of 30 tasks to complete, of which 15 were delivery tasks and 15 pickup tasks, and the customers and depot were randomly distributed in a 50x50 area or a 100x100 area. Time windows at customer locations were either narrow or wide. Narrow time windows were randomly distributed between 60 and 120 minutes, whereas wide time windows were distributed between 90 and 240 minutes. The cost per kilometre was 1, the fixed cost per vehicle was 10, the waiting cost was 10 per hours, the cost of failing to complete a task was 100, and the MAXWAIT parameter was fixed to 30 minutes.

To simulate the stochastic transit time, the overall area was divided into 5x5 squares, resulting in one hundred squares in the 50x50 area (see Figure 4) and four hundred squares in the 100x100 area. Each square had a speed distribution assigned, ss, where the average speed in every square is known but not the real-time speed. Since the transit time is stochastic, one hundred different speed patterns were contemplated in every instance. Forty instances tested one hundred times each meant a total set of four thousand test problems.

Here is Figure 4

Figure 4 – Example of simulated hinterland: 50x50 area.

Under these conditions of stochastic transit times, we compared the classical approach, which schedules all the tasks at the beginning of the day, and our dynamic approach, which solves the problem through iterative optimization and the knowledge of the real-time position of the vehicles. We chose the Caris and Janssens (2009) procedure as our benchmark reference so we could perfectly assess the influence of real-time information and recalculations. We also selected this benchmark algorithm due to its good properties regarding speed and solution quality, considering that the authors compare their algorithm with an exact one, obtaining very satisfactory results. We want to quantify the effects of incorporating real-time information, and so compared this algorithm with ours, which has been adapted to a stochastic environment.

Table 1 presents an overview of the results. The problem class and the value of its characteristics are shown in the first three columns. The fourth and fifth columns represent the average improvement and the maximum improvements of the transportation cost with respect to the classical approach if the re-optimizations are run every time a task is completed, and the last two columns show the average and maximum improvements of the transportation cost if the re-optimizations are run every 15 minutes. Our methodology shows improvements in all the problem classes, showing also some interesting results, like the fact that the periodic re-optimization obtains better results on average. In general, our methodology achieves better results when the width of the customer time windows is narrow, which is very logical because narrow time windows are easier to be affected by delays over the planned scheduling. Results for all the 40 problem instances are given in Table 2, where the results shown contain the average of the 100 different speed patterns.

Table 1 – Results overview.

Problem Class	Hinterland Size	Width of time windows	Average Improvement (%)	Maximum Improvement (%)	Average Improvement (%)	Maximum improvement (%)
			Finished Tasks	Finished Tasks	Fixed Time 15 minutes	Fixed Time 15 minutes
1	50	60-120	9.66	33.85	12.17	37.98
2	50	90-240	5.38	35.19	6.95	35.82
3	100	60-120	3.91	33.23	5.05	35.80
4	100	90-240	4.25	35.19	4.69	35.82

Table 2 – Results of the problem instances.

Problem Class	Instance	Cost of transportation			Improvement (%)	
		No dynamism	Finished Tasks	Fixed Time 15 minutes	Finished Tasks	Fixed Time 15 minutes
1	1	1483.09	1324.01	1284.18	10.73	13.41
1	2	1506.58	1381.14	1339.39	8.33	11.10
1	3	1512.88	1368.61	1313.00	9.54	13.21
1	4	1547.12	1371.37	1315.29	11.36	14.98
1	5	1514.00	1345.43	1325.48	11.13	12.45
1	6	1456.00	1310.23	1283.03	10.01	11.88
1	7	1441.70	1339.36	1294.49	7.10	10.21
1	8	1507.43	1331.23	1338.66	11.69	11.20
1	9	1470.03	1345.74	1296.31	8.46	11.82
1	10	1440.18	1320.47	1275.43	8.31	11.44
2	1	1224.14	1205.41	1199.71	1.53	2.00
2	2	1322.60	1201.96	1205.89	9.12	8.82
2	3	1326.42	1222.36	1200.91	7.85	9.46

2	4	1342.25	1254.93	1250.01	6.51	6.87
2	5	1236.23	1173.36	1174.77	5.09	4.97
2	6	1264.56	1203.53	1169.11	4.83	7.55
2	7	1268.37	1220.23	1208.77	3.79	4.70
2	8	1276.92	1198.72	1184.36	6.12	7.25
2	9	1383.90	1324.54	1261.60	4.29	8.84
2	10	1323.11	1260.34	1203.33	4.74	9.05
3	1	2222.83	2173.45	2127.17	2.22	4.30
3	2	2067.54	1974.52	1951.83	4.50	5.60
3	3	2046.83	1992.01	1947.94	2.68	4.83
3	4	2144.68	2037.35	2010.17	5.00	6.27
3	5	2043.62	1950.98	1961.45	4.53	4.02
3	6	2079.32	2068.11	1994.04	0.54	4.10
3	7	2045.35	1935.62	1940.83	5.36	5.11
3	8	2180.34	2028.55	2039.38	6.96	6.47
3	9	2051.66	1963.86	1954.52	4.28	4.73
3	10	2096.19	2033.80	1989.92	2.98	5.07
4	1	1961.68	1897.47	1891.05	3.27	3.60
4	2	1997.11	1895.86	1871.42	5.07	6.29
4	3	1957.24	1885.85	1888.88	3.65	3.49
4	4	2060.49	1963.98	1939.17	4.68	5.89
4	5	1947.93	1868.00	1859.89	0.41	4.52
4	6	2114.82	1984.98	1966.23	6.14	7.03
4	7	2032.95	1951.02	1950.49	4.03	4.06
4	8	2053.06	1974.39	1959.71	3.83	4.55
4	9	1973.83	1921.54	1931.05	2.65	2.17
4	10	1920.95	1821.75	1817.71	5.16	5.37

Table 3 and Table 4 show the average distance travelled and the average number of times that vehicles arrive late to the origin of a task. The occasions in which the time windows are broken decrease using the dynamic method, as shown in Table 3, but the overall distance travelled increases, as shown in Table 4.

Table 3 – Average number of times when time windows are broken.

Problem Class	Instance	Late arrival in origin			Early arrival in destination			Late arrival in destination		
		No dynamism	Finished Tasks	Fixed Time 15 minutes	No dynamism	Finished Tasks	Fixed Time 15 minutes	No dynamism	Finished Tasks	Fixed Time 15 minutes
1	1	2.5	0.1	0.0	1.4	1.2	0.9	2.6	0.9	0.4
1	2	2.6	0.2	0.0	0.6	0.7	0.5	2.8	1.2	0.9
1	3	2.3	0.2	0.0	1.4	1.0	0.9	2.9	1.2	0.8
1	4	3.3	0.2	0.0	0.6	0.5	0.4	3.2	1.0	0.6
1	5	2.7	0.1	0.0	1.8	1.4	1.2	2.9	0.9	0.8
1	6	2.2	0.0	0.0	2.0	1.7	1.4	2.3	0.7	0.5
1	7	2.2	0.2	0.0	1.3	1.2	0.8	2.2	0.8	0.5
1	8	3.0	0.2	0.0	1.4	1.3	1.0	2.8	0.9	0.7
1	9	2.5	0.2	0.0	1.5	1.4	1.2	2.5	0.8	0.7
1	10	1.9	0.1	0.0	1.5	1.4	0.9	2.2	0.8	0.4
2	1	0.7	0.2	0.0	1.5	1.3	1.4	1.2	0.5	0.5
2	2	1.8	0.1	0.0	0.6	0.6	0.4	2.1	0.6	0.7
2	3	1.7	0.0	0.0	0.7	0.5	0.3	2.2	0.7	0.6
2	4	1.8	0.0	0.0	0.7	0.6	0.6	2.3	0.9	0.9
2	5	1.2	0.0	0.0	1.0	1.0	1.0	1.3	0.5	0.4
2	6	1.3	0.0	0.0	1.1	0.8	0.8	1.6	0.7	0.4
2	7	1.0	0.0	0.0	1.3	1.0	0.7	1.6	0.7	0.7
2	8	1.4	0.0	0.0	0.9	0.8	0.6	1.7	0.5	0.4
2	9	2.0	0.1	0.0	0.8	0.6	0.6	2.7	1.2	1.0
2	10	1.7	0.0	0.0	1.4	0.8	0.7	2.1	1.0	0.7

3	1	2.8	0.7	0.2	0.9	0.6	0.6	3.6	1.7	1.6
3	2	2.2	0.0	0.1	2.2	1.6	1.0	2.2	0.9	0.6
3	3	2.1	0.1	0.0	1.3	0.9	0.8	2.0	0.7	0.6
3	4	2.4	0.2	0.1	1.4	1.3	1.1	2.9	1.0	1.0
3	5	2.4	0.1	0.0	1.0	0.6	0.5	2.0	0.7	0.7
3	6	1.9	0.3	0.1	1.3	1.1	1.0	2.6	1.2	0.8
3	7	2.1	0.2	0.0	1.2	0.7	0.6	2.3	0.8	0.6
3	8	2.6	0.3	0.2	1.0	0.6	0.4	3.6	1.3	1.2
3	9	1.8	0.1	0.1	1.3	1.3	0.8	2.4	0.8	0.6
3	10	2.0	0.5	0.1	0.6	0.6	0.6	2.8	1.0	1.0
4	1	1.5	0.0	0.0	0.8	0.7	0.6	1.8	0.6	0.6
4	2	1.4	0.2	0.0	1.0	0.9	0.9	2.4	0.9	0.6
4	3	1.5	0.2	0.0	0.8	0.6	0.4	2.0	0.9	0.7
4	4	2.0	0.2	0.2	1.1	0.8	0.5	3.0	1.1	0.8
4	5	1.4	0.1	0.0	1.2	1.2	0.7	2.0	0.6	0.6
4	6	2.4	0.2	0.1	0.6	0.6	0.4	3.5	1.3	1.1
4	7	1.9	0.2	0.0	0.6	0.5	0.3	2.8	0.9	0.8
4	8	1.7	0.4	0.1	0.6	0.5	0.6	3.0	1.1	0.8
4	9	1.8	0.1	0.0	1.3	1.0	0.8	2.2	0.9	0.9
4	10	1.3	0.1	0.0	2.1	1.6	1.1	1.6	0.6	0.5

Table 4 – Average distance travelled.

Problem Class	Instance	Late arrival in origin		
		No dynamism	Finished Tasks	Fixed Time 15 minutes
1	1	1149.2	1168.8	1180.4
1	2	1149.2	1189.6	1186.3
1	3	1149.2	1186.0	1170.2
1	4	1149.2	1197.1	1181.6
1	5	1149.2	1188.1	1172.6
1	6	1149.2	1172.2	1163.8
1	7	1149.2	1195.1	1181.1
1	8	1149.2	1174.5	1195.5
1	9	1149.2	1196.5	1162.1
1	10	1149.2	1174.7	1166.3
2	1	1052.3	1092.5	1091.1
2	2	1052.3	1077.4	1075.8
2	3	1052.3	1089.8	1082.5
2	4	1052.3	1100.0	1095.5
2	5	1052.3	1066.3	1073.8
2	6	1052.3	1078.1	1070.2
2	7	1052.3	1087.1	1080.6
2	8	1052.3	1086.9	1079.9
2	9	1052.3	1133.9	1096.7
2	10	1052.3	1102.5	1068.8
3	1	1748.3	1916.5	1887.8
3	2	1748.3	1810.8	1820.1
3	3	1748.3	1844.5	1808.9
3	4	1748.3	1856.4	1831.5
3	5	1748.3	1805.2	1817.4
3	6	1723.2	1872.6	1836.4
3	7	1723.2	1779.9	1800.7
3	8	1723.2	1817.0	1844.5
3	9	1723.2	1804.6	1815.5
3	10	1723.2	1852.4	1816.6
4	1	1705.0	1766.0	1754.6
4	2	1679.9	1737.6	1739.0
4	3	1679.9	1726.5	1749.6
4	4	1679.9	1776.9	1782.1
4	5	1679.9	1732.4	1733.8

4	6	1679.9	1781.6	1777.3
4	7	1679.9	1785.0	1794.5
4	8	1679.9	1790.7	1803.2
4	9	1149.2	1168.8	1180.4
4	10	1149.2	1189.6	1186.3

A single run of the assignment algorithm takes between a few seconds and one minute depending on the size of the problem on an Inter Core Duo 2,4GHz. So, the computational cost of that heuristic is short enough for the problem. However, due to the re-optimization, the simulation of a whole day usually takes between 1 and 10 minutes, but that time is not significant.

CONCLUSIONS

Through the adaptation of an algorithm to the dynamic drayage problem and the application to a series of problem instances, we have shown in this paper the beneficial effects of incorporating the knowledge of real-time vehicle locations in a drayage fleet. Although Caris and Janssens (2009) developed a fast heuristic and achieved good results with a static and deterministic approach, our work shows that this approach produces worse results than the dynamic procedure in 98% of the cases when tested in a stochastic environment. Our results prove that the use of real-time information helps to reduce operation costs an average 5-10%, also contributing to decrease the number of times that time windows are broken. We have analyzed two dynamic methodologies: re-optimization every time a task is completed or every fixed amount of time, with better results for the second option.

Further improvements of this methodology necessarily include the development of a faster algorithm for the allocation of tasks to vehicles every time a re-optimization is run. For instance, Laporte et al (1992) use an exact method to solve a vehicle routing problem with stochastic travel times, and the results they obtain prove that an exact method is only suitable for instances with a small number of tasks. Large computational times mean that the position of vehicles is likely to change significantly between the beginning of the re-optimization process and the implementation of the results, perhaps causing the new solution to be obsolete by the time it is implemented. Once the dynamic management of drayage fleets has proved successful, it could further benefit from the introduction of procedures based on insertion heuristics or other meta-heuristic approaches (Bell and McMuller, 2004).

APPENDIX

Nomenclature used in this work:

V	Set of vehicles
T	Set of tasks
T^D	Set of delivery tasks
T^P	Set of pick up tasks
T_f^{ini}	Set of initial fictitious tasks
T_f^{end}	Set of final fictitious tasks
c_{Km}	Travelling cost per kilometre
c_v	Fixed cost of vehicle v

c_w	Waiting time cost per hour
c_l	Cost of missing a train
d_i	Distance of task i
d_{ij}	Distance on arc connecting task i with task j
τ_i	Transit time of task i
τ_{ij}	Transit time on arc connecting task i with task j
t_i	Expected transit time of task i
t_{ij}	Expected transit time on arc connecting task i with task j
s_i^O	Service time at the origin of task i
s_i^D	Service time at the destination of task i
E_i^O	Earliest start time of task i
L_i^O	Latest start time of task i
E_i^D	Earliest finish time of task i
L_i^D	Latest finish time of task i
E_{ij}	Earliest start time of pair (i,j)
L_{ij}	Latest start time of pair (i,j)
RS_{ij}	Total time to serve the pair (i,j)
RS_v	Route service time of vehicle v
I_{ghij}	Improvement produced by cross of pairs (i,j) and (g,h)
VR_{ghij}	Improvement due to the reduction of the necessary number of vehicles by cross of pairs (i,j) and (g,h)
$MAXWAIT$	Maximum waiting time
$MINWAIT_{ij}$	Minimal waiting time between tasks i and j

REFERENCES

Ball, M., Golden, B., Assad, A. & Bodin, L. (1983) Planning for truck fleet size in the presence of common-carrier options. Decision Sciences, vol. 14, no. 1, pp. 103–120.

Bell, J.E. and McMuller, P.R. (2004) Ant colony optimization techniques for the vehicle routing problem. Advanced Engineering Informatics, vol. 18, no. 1, pp. 41-48.

Bent, R. W. & Van Hentenryck, P. (2004). Scenario-Based Planning for Partially Dynamic Vehicle Routing with Stochastic Customers. Operations Research, vol. 52, no. 6, p. 977.

Bertsimas, D. J. (1992). A vehicle routing problem with stochastic demand. Operations Research, vol. 40, no. 3, pp. 574-585.

Bodin, L., Mingozzi, A., Baldacci, R., & Ball, M. (2000). The Rollon–Rolloff Vehicle Routing Problem. Transportation Science, vol. 34, no. 3, pp. 271-288.

Caris, A. & Janssens, G. K. (2009). A local search heuristic for the pre-and end-haulage of intermodal container terminals. Computers and Operations Research, vol. 36, no. 10, pp. 2763-2772.

Cheung, R. K. & Hang, D. D. (2003). A time-window sliding procedure for driver-task assignment with random service times. IIE Transactions, vol. 35, no. 5, pp. 433-444.

Cheung, R. K., Hang, D. D., & Shi, N. (2005). A labeling method for dynamic driver-task assignment with uncertain task durations. Operations Research Letters, vol. 33, no. 4, pp. 411-420.

De Meulemeester, L., Laporte, G., Louveaux, F. V., & Semet, F. (1997). Optimal sequencing of skip collections and deliveries. Journal of the Operational Research Society, vol. 48, no. 1, pp. 57-64.

Erera, A. L., Smilowitz, K. R. & P. Ioannou. Intermodal drayage routing and scheduling. In Intelligent freight transportation, CRC Press. Taylor & Francis Group, 2008, pp. 171-188.

Francis, P., Zhang, G., & Smilowitz, K. (2007). Improved modeling and solution methods for the multi-resource routing problem. European Journal of Operational Research, vol. 180, no. 3, pp. 1045-1059.

Gendreau, M., Laporte, G., & Seguin, R. (1995). An exact algorithm for the vehicle routing problem with stochastic demands and customers. Transportation Science, vol. 29, no. 2, pp. 143-155.

Ileri, Y., Bazaraa, M., Gifford, T., Nemhauser, G., Sokol, J., & Wikum, E. (2006). An optimization approach for planning daily drayage operations. Central European Journal of Operations Research, vol. 14, no. 2, pp. 141-156.

Laporte, G., Louveaux, F., & Mercure, H. (1992). The vehicle routing problem with stochastic travel times. Transportation Science, vol. 26, no. 3, pp. 161-170.

Psaraftis, H. N. (1995). Dynamic vehicle routing: Status and prospects. Annals of Operations Research, vol. 61, no. 1, pp. 143-164.

Savelsbergh, M. W. P. & Sol, M. (1995) The general pickup and delivery problem, Transportation Science, vol. 29, no. 1, pp. 17-29.

Smilowitz, K. (2006). Multi-resource routing with flexible tasks: an application in drayage operations. IIE Transactions, vol. 38, no. 7, pp. 577-590.

Solomon, M.M. (1987) Algorithm for the vehicle routing and scheduling problems with time windows contraints. Operational Research, vol. 35, no. 2, pp. 254-65

Spasovic, L. N. & Morlok, E. K. (1993). Using marginal costs to evaluate drayage rates in rail-truck intermodal service. Transportation Research Record, Vol. 1383, pp. 8-16.

Wang, J. Q., Tong, X. N., & Li, Z. M. (2007). An Improved Evolutionary Algorithm for Dynamic Vehicle Routing Problem with Time Windows. In Computational Science – ICCS 2007. Springer Berlin / Heidelberg, ed., pp. 1147-1154.

Wang, X. & Regan, A. C. (2002). Local truckload pickup and delivery with hard time window constraints. Transportation Research Part B, vol. 36, no. 2, pp. 97-112.