Dynamic approach to solve the daily drayage problem with travel time uncertainty

Alejandro Escudero; Jesús Muñuzuri; José Guadix; Carlos Arango

Organization Engineering Group. Department of Industrial Management and Business Administration. University of Seville

Abstract

The intermodal transport chain can become more efficient by means of a good organization of drayage movements. Drayage in intermodal container terminals involves the pick up and 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. This scheduling has traditionally been done once a day and, under these conditions, any unexpected event could cause timetable delays. We propose to use the real-time knowledge about vehicle position to solve this problem, which permanently allows the planner to reassign tasks in case the problem conditions change. This exact knowledge of the position of the vehicles is possible using 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: Intermodality, Drayage, Heuristics, Real-time

1. INTRODUCTION

Road transport has always been prevalent in the movement of freight. However, the increasing road congestion and the necessity to find more sustainable means of transport have encouraged different governments to promote intermodality as an alternative. This combines the cost effectiveness of rail or ship with the flexibility of trucks. Even so, intermodal transport has several difficulties to overcome in order to become viable. One of them is a high fixed cost, which is why intermodality is unsuitable for trips shorter than a threshold distance (about 700 km) [34].

In an intermodal transport chain, the initial and final trips, also called drayage operations, represent 40% of total transport cost. Drayage operations move loaded or empty containers between terminals, shippers and consignees. According to several researches [34, 33, 39], a proper planning of drayage operations has the potential to reduce the threshold distance, thus raising the viability of intermodal transport.

The daily drayage problem involves the routing of vehicles to complete a set of drayage tasks. Its objective is to minimize the total cost of all the operations carried out by a company in the hinterland of a terminal. When there are time window restrictions, the problem is called daily drayage problem with time windows (DDPTW).

The DDPTW could be considered a specific case of the vehicle routing problem with time windows, VRPTW, [2, 38, 13, 14], where the status of the truck would be a binary variable: the truck can be loaded, with a container, or bobtail, without any container. Then, one may think that

the different approaches used to solve the VRPTW would be directly applied to the DDPTW. However, experimental results prove that good VRPTW algorithms might not be the optimal method to solve particular cases [15].

So, the DDPTW has been formulated as an asymmetric multi-traveling salesman problem with time windows, am-TSPTW, [12, 3, 26, 37, 19, 44] or as a full-load pickup and delivery problem with time windows, FLPDPTW, [22, 9, 10, 25, 7]. In the am-TSPTW every task is represented by a node, while in the FLPDPTW the tasks are represented by an arc. The different techniques used to solve both the FLPDPTW and the am-TSPTW could also be applied to solve the DDPTW. Jula et al. [26] shows how a FLPDPTW could be transformed into a am-TSPTW.

In the last years, a series of studies have focused on the DDPTW, but most of them consider the problem only from a static and deterministic perspective. Gronalt et al. [22] develop four heuristics based on cost saving to solve the FLPDPTW. This same problem is solved through Lagrangian relaxation by Imai et al. [25] in the hinterland of a terminal. Caris and Janssens [7] propose a two-phase insertion heuristic to build an initial solution that is then improved by a local search heuristic. In a posterior work [6] a deterministic annealing algorithm is also included. Jula et al. [26] models the problem with time windows in origin and destination as an am-TSPTW, and suggest three solution approaches: (a) an exact method based on dynamic programming, (b) a hybrid methodology consisting of dynamic programming for generating feasible solutions in conjunction with genetic algorithms and (c) a heuristic insertion method. Namboothiri and Erera [35] develop an understanding of the potential productivity impacts of appointment-based access control systems on drayage firms

Special attention has been paid to integrate the allocation of empty containers and routing in drayage operations. Smilowitz [37] models the problem as an am-TSPTW with flexible tasks, which are used to model potential empty trailer movements. The number of possible executions of flexible tasks is limited by the distance between nodes, upon which a region of geographic feasibility is defined. The solution approach developed to solve this problem includes column generation embedded in a branch-and-bound process. This work is improved by Francis et al. [19], which introduces a variable radius method of defining possible executions of flexible tasks. Ileri et al. [24] use column generation, too. Different kind of drivers and complex drayage tasks are studied. A improvement of the approach suggested by Caris and Janssens [6] is proposed by Braekers et al.[4], which consider empty container movements. Two solution approaches are proposed: a sequential and an integrated approach.

Zhang et al. [44] formulate the problem as an am-TSPTW with multiple depots and a single terminal, and use a reactive tabu search to solve it. This problem is expanded in Zhang et al. [43], which uses a windows-partition based method inspired by Wang and Regan [41]. Zhang et al. [45] limit the number of empty containers in the depot and even although this problem only considers a depot and a terminal, it is computationally more complex.

The different versions of the deterministic and static problems are defined in Erera et al. [16], which gives a general revision of scientific contributions to the deterministic drayage problem. However, there are not many examples in the literature of stochastic versions of the DDPTW. Cheung and Hang [8] 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. Mahr et al. [30] develop a comparison between an on-line method and a agent-based method. Uncertainty on both service time and arrived time are considered. Zhang et al. [42] introduce a dynamical approach in a problem where the tasks are not known a priori. It is uncommon to find randomness in trip times [27], which is particularly appropriate when the intermodal terminal requiring drayage opera-

tions is close to a large urban centre. This approach is followed by Escudero et Al. [17], who deal with transit time uncertainty through a dynamical re-optimization. The real-time knowledge of the position of the vehicle is assumed, and the problem is solved using a dynamical two-phases heuristic.

Our work considers the real-time knowledge about the position of the vehicles, which permanently allows the planner to dynamically reassign tasks, as described in Escudero et al. [17]. This exact knowledge of the position of vehicles is possible thanks to the use of a geographic positioning system by satellite (GPS, Galileo, Glonass), which can provide real-time data for dynamic recalculations. We propose an approach based on re-optimization is proposed, taking a snapshot of the state of each task and the position of each vehicle before recalculating. We then solve a DDPTW with stochastic transit times by means of an iterative optimization algorithm, modifying if necessary the scheduling of the drayage fleet. We calculate estimations of transit times in every re-optimization. To do it, we have implemented two algorithms: (a) an improved two-phases heuristic as shown in Escudero et al. [17] and (b) a genetic algorithm. Two are the main differences between Escudero et al. [17] and the improved two-phases heuristic developed in this paper: the *priority tasks*, which are the first ones to be inserted in the routes, and the *improvement factor*. If a vehicle is on its way to the origin of a given tasks and a re-optimization occurs, the driver will only change this destination (origin of the task) if the improvement factor is higher than a certain threshold level. Also, we propose series of benchmark tests based on Solomon's VRPTW instances.

The paper is structured as follows. Section 2 presents the problem definition and its formulation. A description of the dynamic optimization procedure is given in Section 3. Section 4 explains the algorithms used in each iteration. Test problems and results are shown in Section 5. Finally, the paper is concluded in Section 6.

2. PROBLEM FORMULATION

The DDPTW seeks to find the best assignment of tasks to vehicles and the order in which they will be carried out to minimize the total cost of completing all the tasks. These tasks involve the movement of containers from an intermodal terminal to a specific customer or vice versa, fulfilling some temporal requirements. The operative costs include the number of vehicle, the travelled distance and a series of penalty costs in case temporal requirements are not accomplished.

In our problem there is uncertainty with respect to the transit times, so it is impossible to foresee the exact duration of the trips and the arrival time of the vehicles to every destination. Due to the high computational costs involved, it is unfeasible to work with stochastic times [27], which led us to develop an interactive re-optimization approach. The formulation is focused on solving snapshots of this problem.

A snapshot of the DDPTW, at a given instant, can be formulated as an asymmetric multi-traveling salesman problem with time windows (m-ATSPTW). This is defined as a graph that symbolizes the state of vehicles and tasks, $G_{time} = (T, A)$, where T represents the set of tasks and A is the set of links between tasks.

All the tasks can be classified into two groups: real and pseudo tasks. The set of real tasks, T_r, includes all the unfinished tasks, while pseudo tasks, T_p, are created to store vehicle information at the re-optimization instants: position, tasks in process, destination, etc.

$$T = T_r \cup T_p$$

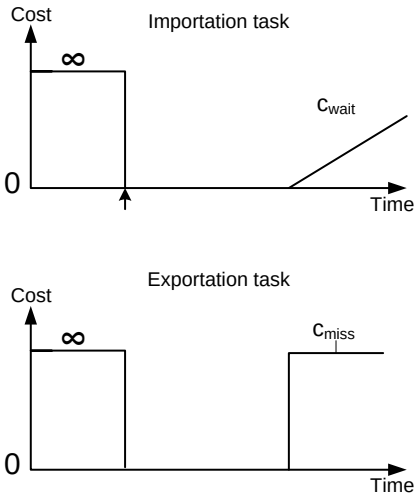

Figure 1: Soft Time Windows

$$T_r = T_r^I \cup T_r^E$$
$$T_p = T_p^{ini} \cup T_p^{end}$$

Each task is represented by an origin, a destination, service times in the origin and destination, the distance and expected 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. All these data are aggregated into a single node that represents the entire movement with all the characteristics of the task.

A task, $i \in T_r^I$, with origin in the terminal is called an importation task, whereas an exportation task, $j \in T_r^E$, has the origin at a customer and the destination in the terminal. The time window is always associated to the terminal, so importation tasks have the time window in their origin and exportation tasks have it in their destination. Both time windows are soft (Fig. 1), which means that tasks can be completed with a delay, but a penalty cost will be charged. If the consequence of the delay is that the container need to spend time waiting at the terminal, then the penalty cost must be proportional to this time. However, if due to the delay, the container misses its train or vessel, the penalty cost will be higher and fixed.

The vehicle fleet is known and is assumed to be homogeneous. Due to re-optimization process, in a snapshot the vehicle could be anywhere; to overcome this difficulty, the vehicle state is represented by pseudo tasks, T_p^{ini}, which contain all the vehicle information. A pseudo task could include information about tasks IN PROCESS, too. If the vehicle is FREE or ASSIGNED, the origin and destination of its pseudo tasks is its current position in the instant *time*, and the services times are 0. However, if the vehicle is BUSY, since it must finish its task IN PROCESS, the pseudo task must contain information about the remainder of that tasks. In that case, the origin will be the current position while the destination will be the destination of the task IN PROCESS.To force the system to go on with the tasks without interruption, the origin time win-

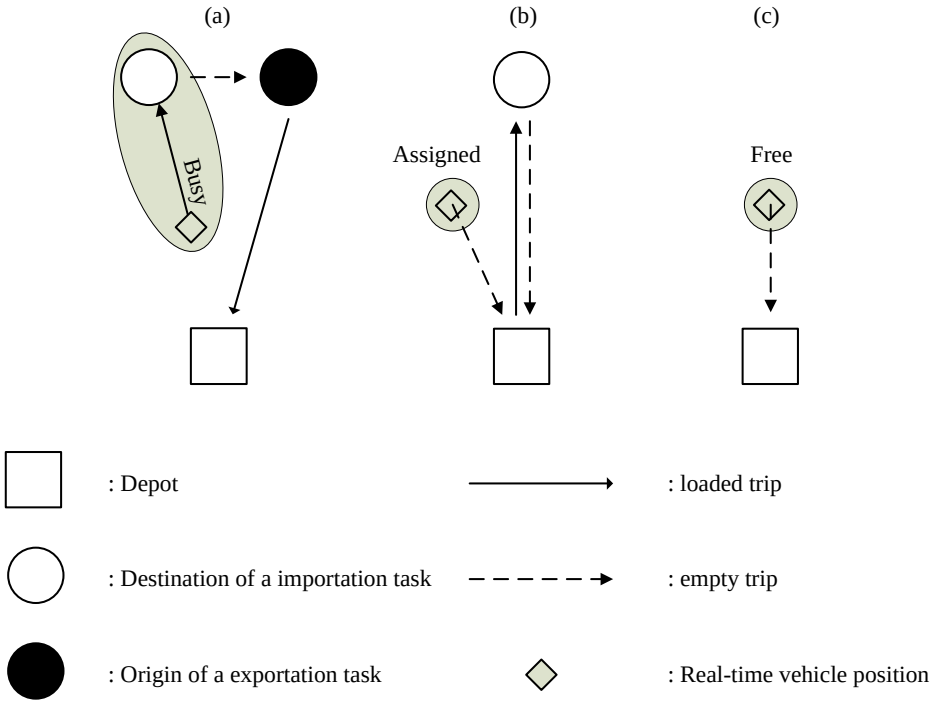

Figure 2: Vehicle states and pseudo tasks

dows in pseudo tasks are identified with the current *time* instant. An exemplary case is shown in Fig 2 where shady ovals represent the pseudo tasks.

Other pseudo tasks are also defined to determine the depot time window of every truck, T_p^{end}. Their origin and destination correspond to the depot position, their service times are 0, and their time windows match the depot time window. If the driver arrives late to the depot, a given amount will have to be paid for this time, c_{depot} per hours.

To formulate the problem we use the following notation:

T set of tasks.
T_r set of real tasks.
T_p set of pseudo tasks.
T_r^I set of importation tasks
T_r^E set of exportation tasks.
T_p^{ini} set of origin pseudo tasks.
T_p^{end} set of destination pseudo tasks.
c_v : fixed cost per vehicle.
c_{km}: cost per unit of distance travelled.
c_{miss}: high fixed penalty cost if the delay results in the miss of the train or ship.
c_{wait}: cost per time unit due to waiting situations.

c_{depot}: cost per time unit due to delays in the return to the depot.
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.
s_i^O: service time at the origin of task i.
s_i^D: service time at the destination of task i.
d_i: distance of task i.
$\overline{t_i}$: expected transit time of task i.
d_{ij}: distance on arc connecting task i with task j.
$\overline{t_{ij}}$: expected transit time on arc connecting task i with task j.
uv_i: 1 if vehicle i was used previously. 0 in any other case.

The decision variables are defined as:

x_{ij}: 1 if task i and task j are served consecutively, 0 in any other case.
nv_i: 1 if vehicle i is used for the first time, 0 in any other case.
l_i^O: 1 if origin time window of task i is served late, 0 in any other case.
l_j^D: 1 if destination time window of task j is served late, 0 in any other case.
l_i^{depot}: 1 if vehicle i arrives late to depot, 0 in any other case.
st_i: time that task i begins.
ft_i: time that task i finishes.

The following lines explain the formulation of the problem described above. The objective function minimizes total costs of serving all tasks: distance, new vehicle used and penalty costs (See equation (1)). Constraints (2) and (3) ensure that each customer is visited exactly once. Constraints (4) enforce flow conservation. Limitation in the time windows is forced by constraints (5) and (6). The consistency of time variables st_i and ft_i is enforced by (7) and (8). Expression (9) indicates which vehicles are used for the first time. Expressions (10), (11), and (12) point out which tasks are served with delays, where M is a sufficiently large number. Finally, (13)-(19) define the domain of the decision variables.

$$OF : \min\left(\sum_{i\in T_p^{ini}\cup T_r} \sum_{j\in T_r\cup T_p^{end}} d_{ij}x_{ij} + c_v \sum_{i\in T_p^{ini}} nv_i + c_{wait} \sum_{i\in T_r^P} (st_{i-L_i^O}) l_i^O + c_{miss} \sum_{j\in T_r^D} l_j^D + c_{depot} \sum_{i\in T_p^{end}} (st_{i-L_i^O}) l_i^{depot} \right) \tag{1}$$

subject to:

$$\sum_{i \in T_p^{ini} \cup T_r} x_{ij} = 1 \qquad \forall j \in T_r \cup T_p^{end}, \tag{2}$$

$$\sum_{j \in T_r \cup T_p^{end}} x_{ij} = 1 \qquad \forall i \in T_p^{ini}, \tag{3}$$

$$\sum_{i \in T_p^{ini} \cup T_r} x_{ij} - \sum_{i \in T_r \cup T_p^{end}} x_{ji} = 0 \qquad \forall j \in T_r, \tag{4}$$

$$st_i \geq E_i^O \qquad \forall i \in T, \tag{5}$$

$$ft_i \geq E_i^D \qquad \forall i \in T, \tag{6}$$

$$ft_i \geq st_i + s_i^O + \overline{t_i} \qquad \forall i \in T, \tag{7}$$

$$x_{ij}(ft_i + s_i^D + \overline{t_{ij}} - st_j) \leq 0 \qquad \forall i, j \in T, \tag{8}$$

$$\left(\sum_{j \in T_r} x_{ij} \right) - uv_i \leq nv_i \qquad \forall i \in T_p^{ini}, \tag{9}$$

$$st_i - L_i^O \leq Ml_i^O \qquad \forall i \in T_r^P, \tag{10}$$

$$ft_j - L_j^D \leq Ml_j^D \qquad \forall j \in T_r^D, \tag{11}$$

$$st_i - L_i^O \leq Ml_i^{depot} \qquad \forall i \in T_p^{end}, \tag{12}$$

$$st_i \geq 0 \qquad \forall i \in T, \tag{13}$$

$$ft_i \geq 0 \qquad \forall i \in T, \tag{14}$$

$$x_{ij} \in 0, 1 \qquad \forall i \in T_p^{ini} \cup T_r^P, j \in T_r^D \cup T_p^{end}, \tag{15}$$

$$nv_i \in 0, 1 \qquad \forall i \in T_p^{ini}, \tag{16}$$

$$l_i^O \in 0, 1 \qquad \forall i \in T_r^P, \tag{17}$$

$$l_j^D \in 0, 1 \qquad \forall j \in T_r^D, \tag{18}$$

$$l_i^{depot} \in 0, 1 \qquad \forall i \in T_p^{end}, \tag{19}$$

3. DYNAMIC OPTIMIZATION PROCEDURE

We run the optimization algorithm at the beginning of the day (See Section 4), thus calculating the expected values of the transit times and obtaining an initial solution. However, since transit times are stochastic it is impossible to know the exact time required to complete a task beforehand, and any unexpected event (congestion, accidents, etc.) could cause delays on the timetable. To overcome this situation, a dynamic procedure like the one shown in Escudero et al [17] is used to improve the solution. This procedure tries to adapt the solution to the circumstances at all times, improving the solution iteratively. Hence, every fixed time or every time a task is accomplished, the algorithm is run again considering the updated data only for the pending tasks. Besides, a re-optimization process is run if the system detects the CRITICAL INSTANT of a task. This is the precise moment in which a vehicle need to leave the depot to perform the corresponding task on time. It only is taken into account if there are not any vehicles assigned to that task.

When a re-optimization is run, a snapshot is taken. Information about both the current position of the vehicles and the degree of completion of the tasks is verified. This information lets

the system know if a vehicle is FREE, BUSY or ASSIGNED. A truck is busy if it is performing a task at 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.

Since the cost structure in DDPTW depends on the number of vehicles used, it is important to know which vehicles have already been used and which have not, USEDVEH and NONUSEDVEH. Information about vehicles in the depot is known as well.

Besides the vehicle state, the degree of completion of the task must also be updated in each recalculation. There are tree possible values: PENDING, IN PROCESS and FINISHED. A pending task is defined as the task which has not yet begun to be performed; when a truck is carrying out a task, this is considered a task in process; and finally, finished tasks are those which have already been completed.

In a re-optimization process, every BUSY vehicle must finish the corresponding task IN PROCESS which it is engaged in. Vehicle assigned to PENDING tasks may be re-routed, and if it is necessary, new vehicles from the depot could entry on the new routes.

Due to the transit times being stochastic, some tasks may only be performed by a specific vehicle. These tasks are called PRIORITY TASKS and the re-optimization will insert them first to avoid the impossibility of carrying them out.

The pseudocode of all this dynamic methodology can be written as shown in Algorithm 1. This improves the dynamic methodology described in Escudero et al. [17] due to the analysis of both CRITICAL INSTANTS and PRIORITY TASKS.

Algorithm 1 Dynamic Methodology

1: **if** Re-Optimization Event happens (fixed time, accomplished task and critical instants) **then**
2: Check the vehicle state (FREE, BUSY, and ASSIGNED).
3: Check the task state (PENDING, IN PROCESS and FINISHED).
4: Check PRIORITY TASKS
5: RE-OPTIMIZATION taking into account the new information
6: Update all the variables about the vehicle state (FREE, BUSY, ASSIGNED, USEDVEH, NONUSEDVEH, and DEPOTVEH).
7: **end if**

4. DYNAMIC ALGORITHM

The above problem requires a fast solution algorithm. The planner has to provide real-time information, so algorithms with a high computational costs would be unfeasible. In fact, only heuristics and meta-heuristic methods can fulfill the time requirements.

We analyzed different heuristics [38, 9, 22, 7], and chose the two-phase insertion heuristic (2PIH) shown in Caris and Janssen [7] due to its balance between high quality solutions and acceptable computational time. This heuristic was adapted to the daily drayage problem with travel time uncertainty (D-2PIH) as show in [17], but it is improved by preventing the existence of tasks that cannot be carried out on time by any vehicle

At the beginning of the day, we run the two-phase insertion heuristic (2PIH). Every time that an event takes place, two options appear:

- Run the real-time two-phase insertion heuristics, D-2PIH.

- Run a genetic algorithm to improve the last solution found, GA.

The real-time insertion heuristic and the genetic algorithm are explained below. Since the vehicle could be anywhere when the re-optimization process starts, we defined the corresponding pseudo tasks.

4.1. Dynamic two phase insertion heuristic

The D-2PIH is an adaptation of the 2PIH described by Caris and Janssens [7]. It follows the principles used in Escudero et al. [17] but is improved to prevent that some task could no be carried out on time by any vehicle. The improved algorithm is shown in Algorithm 2. The real-time insertion heuristic is divided into two phases. In the first one, all the possible combinations of the tasks (including pseudo tasks) are analyzed. In the second phase, the combined tasks are inserted into routes, attending firstly to the priority tasks, and selecting the best pairs later.

Algorithm 2 Dynamic Insertion Heuristic

```
1: Check possible merged tasks.
2: Evaluation individual and merged tasks.
3: ListPairingTasks ←Generate list of possible pairing
4: while ListPairingTasks is not empty do
5:    Check PRIORITY TASK
6:    if There are PRIORITY TASKs then
7:       Choose the correct pairing tasks to perform in time this task.
8:       Deleted from ListPairingTasks the other pairing that contain this PRIORITY TASK
9:    end if
10:   Selection of best pairing from ListPairingTasks
11:   Insertion in the routes
12: end while
```

The heuristic procedure is based on the savings obtained through the merging of single importation and exportation tasks. When tasks are completed individually, a 50% of the journey is wasted in empty trips, as shown in Fig. 3a. Meanwhile, the combination of tasks may lead to a significant reduction in the total distance travelled, as represented in Fig. 3b. pseudo tasks may also be included in the pairs. A vehicle that is BUSY and performing an importation tasks, could save costs if it is merged with an exportation tasks, as is shown in Fig. 3c.

Not any of tasks can be combined into a feasible merged task mainly due to the temporal requirements of the different tasks. Every task must be completed inside the corresponding time windows, keeping in mind that when two task are merged, the second task of the pair must be reached after the completion of the first task. Also, the waiting time between merged tasks has been limited to a maximun amount, $MAXWAIT$.

Then, the first steps of the improved algorithm are similar to Escudero et al. [17] to check the feasible pairs. Once these are found, they are evaluated and ranked according to the savings in expected travel time generated, according to the expression $\overline{t_i} + \overline{t_j} - \overline{t_{ij}}$. [7] uses the savings in travel time but, since the transit time is uncertain in our case, we chose expected travel time instead. When a previous pairing involves certain vehicle on its way to the origin of a given tasks, this pairing is favoured with a *improvement factor*. So, the driver will only change this destination (origin of the task) if the improvement is enough high. Single tasks are included in this "List of possible pairing tasks" although its saving cost will be zero.

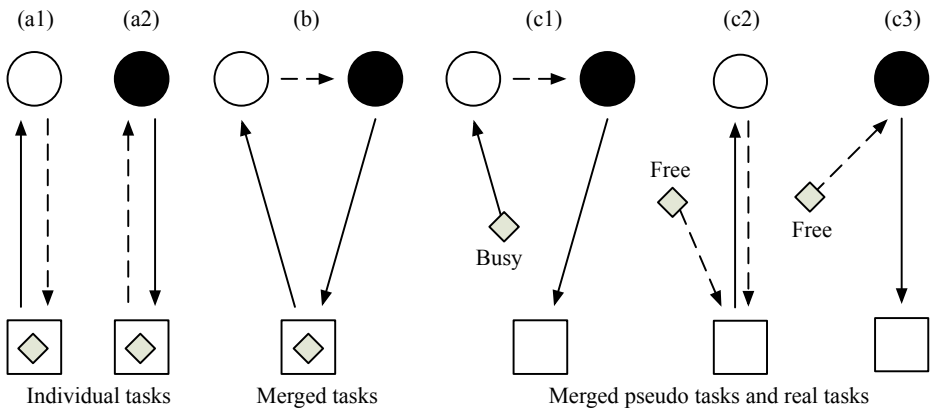

Figure 3: Merging tasks

Once the "list of possible pairing tasks" is defined, the route construction differs from Escudero et al. [17]. Now, the algorithm search first some PRIORITY TASKS, following the stages shown in Algorithm 3. These are tasks that can only be performed by a specific vehicle. If this task is not assigned to this specific vehicle, then it would not fulfill time window requirements.

If PRIORITY TASKS are detected, an extra step is added. The pair which contains the pseudo task of the adequate vehicle and the priority tasks is formed, and all the other rest pairs that contain either of these tasks are deleted from the list of possible pairing tasks.This guarantees to execute this pairing which contains the vehicle and the priority tasks.

Algorithm 3 Check PRIORITY TASK

1: Check instants vehicles will be finishing their IN PROCESS tasks.
2: Check L_i^O and L_i^D in each PENDING task.
3: Analyze how many vehicles could do each PENDING task.
4: **if** less than one vehicle could perform the task **then**
5: This task is marked like PRIORITY TASK.
6: **end if**

Now, the best pairs are selected from the list attending to their savings property. Since there is only one pair which contains the priority task, it will always be selected. Also, these selected pairs can not contain repeat tasks.

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 pseudo task assigned to the corresponding vehicle. The other pairs are assigned sequentially. First, the priority tasks, which can only be attended by a specific vehicle, are inserted. After that, the other tasks are inserted into routes in increasing order of their latest start time.

A pair can be inserted into a route if two conditions are met: the first condition is that the vehicle must be able to begin to complete the pair before its latest start time, and the second condition establishes that the vehicle must be able to return to the depot within its depot window.

This algorithm is iterative, so after a pair is inserted in a route, it will check again if there are

other PRIORITY TASKS.

4.2. Genetic algorithm

The genetic algorithm (GA) is a stochastic metaheuristec based on evolution theory proposed by [23], who developed the basic concepts. [21] demonstrated the use of the GA to solve complex optimization problem. GAs have been widely used to solve VRPTW [5], including stochastic problems [31]

The GA used for solving the problem is as follows:

Algorithm 4 Genetic algorithm

```
1:  population ← POPULATION
2:  fitness ← EVALUATION (population)
3:  while stopcondition == False do
4:     parents ← SELECTION (population, fitness)
5:     if RAND(0, 1) < Pcross then
6:        offsprings ← CROSSOVER (parents)
7:     else
8:        offsprings ← MUTATION (parents)
9:     end if
10:    poputation ← population ∪ offsprings
11:    fitness ← EVALUATION (population)
12:    population ← REJECT (population, fitness)
13: end while
```

GA works with a population of individuals (possible solutions), encoded as chromosomes (vector of tasks). New individuals (offsprings) are created using CROSSOVER and MUTATION operators over existing individuals of the population. The use of these operators is managed according to a preset probability, $Pcross$. Through an iterative process new individuals are generated and some individuals (the worse ones) are rejected, until some convergence criteria are met. By doing this, the individuals in the population will be well adapted to the problem, and the best chromosome generated is then decoded, providing the corresponding solution to the problem.

A GA is a very adaptive and flexible optimization procedure, but the effect on its performance of a correct selection of values for its different parameters is critical. We developed a set of three experiments to determine these values. Every experiment was tested over the 12 instances of the R1 class (See Section 5), considering 25 and 50 customers.

The different aspect of the GA are described next:

4.2.1. Population and chromosome

In a genetic algorithm, a solution is represented with a chromosome (see Fig. 4). Each chromosome is composed of a number of genes, with each gene being a task to complete. These tasks can correspond either to pseudo or real tasks, with every pseudo task being the representation of a vehicle and its state. Each vehicle carries out its pseudo task and all tasks until the next pseudo one, using a circular chromosome design. So, the solution shown in Fig. 3 represents the following routes: $A(1 \rightarrow 2 \rightarrow 4), B(3 \rightarrow 7 \rightarrow 8 \rightarrow 10), C(6 \rightarrow 9 \rightarrow 5)$.

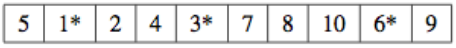

Figure 4: Chromosome

The population size was set to 60 individuals. The initial population includes 59 randomly generated chromosomes and 1 elaborated chromosome. This one is created through 2PIH [7] in the first optimization, and it is the best solution of the previous optimization when any other re-optimization process is launched.

in any other re-optimization.

4.3. Crossover and mutation operator

These operator try to find individuals better adapted through the combination of two existing individuals (Crossover) or the modification of one (mutation). Five crossover operators and four mutation operator were analyzed, and a detail explanation of all these operator is shown in [28]

- Partially mapped crossover (PMX) [20].
- Cycle Crossover (CX) [36].
- Order Crossover (OX1) [11].
- Order-based Crossover (OX2) [40].
- Position-based crossover (POS) [40]
- Displacement mutation (DM) [32]
- Exchange mutation operator (EM) [1]
- Insertion mutation (ISM) [18, 32]
- Scramble mutation (EM) [40]

All the combination of five crossover operators (PMX, CX, OX1, OX2, POS) and four mutation operator (SM, EM, ISM, DM) were tested, setting the probability of crossover, $Pcross$, between 1 and 0.75 in steps of 0,5. Every instance was tested 20 times for every possible combination of crossover and mutation operator, and for every possible $Pcross$, and the best pairs of operator found was PMX-ISM.

Once the operators have been chosen, a more specific experiment was done to choose the best value for $Pcross$. Using only the PMX-ISM operator, we tested values of $Pcross$ between 1 and 0 in steps of 0.5, running 100 times every problem instance. In view of the results, we chose $Pcross = 0.9$.

4.3.1. Selection procedure and repopulation mechanism

In every iteration, the GA selects two individuals. We tested three different selection procedures:

- Roulette wheel with probability of selection of each individual proportional to the objective function.

- Roulette wheel with probability of selection of each individual proportional to its ranking inside the population.
- Tournament method with two tournaments of three individuals each.

The selected individual are then crossed and mutated to obtain new individuals. One or both of these new individuals might be an exact match of one of the already existing individuals, in which case we analyzed four repopulation possibilities:

- Do nothing, and allow the repetition to happen.
- If a new individual is repeated, then it is deleted and a new individual is randomly generated.
- If the best individual fitness and the average fitness of the population are close enough, then the best individual is preserved and the rest of the population is reset.
- If the best individual?s fitness is repeated N times, then the best individual is conserved and the rest of the population is reset.

We tested different combinations of the selection procedures and the reoptimization mechanism. The best combination, which was thus selected for the simulations, was a roulette wheel where the probability to select an individual is proportional to its fitness, the fitness value being the equal to the inverse of the cost of the solution. When a new individual that is already in the population appears (repetition case), it is deleted and a new individual is randomly generated. Finally, the reject operator consists of two tournaments of three individuals each, thus guaranteeing the survival of the best solution.

5. TEST PROBLEM AND RESULTS

Heuristics applicable to the standard VRPTW are compared by applying them to the set of benchmark problems presented in Solomon [38] and to the PDPTW through the set shown in Li and Lim [29]. However, The DDPTW with transit time uncertainty is different to these standard benchmark problems because in the drayage problem the vehicle is either empty or fully loaded, and time windows could be defined in both origin and destination. We thus adapted the benchmark data in [29] to our case. Three classes of problems were defined (R1, C1 and RC1). In R1 problems, the geographical data are randomly generated by a random uniform distribution; in C1 problem the customers are clustered, while in RC1 they are semi-clustered. The test problems are created generating time windows of various widths (20, 60, and random) for different percentages of customers with time windows (25, 50, 75 and 100%). A summary of every tested instance is shown in the Table 1. Problems with 25 and 50 customers were tested, in each case. The cost per kilometer was 1, the fixed cost per vehicle was 10, the waiting cost was 10 per hours, and the cost of failing to complete a task was 100.

To simulate the stochastic transit time, the overall area was divided into 1x1 squares. Each square had a speed distribution assigned where the average speed in every square is known, $\overline{v_{xy}}$, but not the real-time speed, v_{xy}. Since the transit time is stochastic, one hundred different speed patterns were contemplated in every instance. The expected transit time, $\bar{t}$, of a displacement is the sum of the expected transit time of every stretch, the same for the real transit time, t.

Table 1: Instances

Problem Class	Instancias	Width of time windows		Percentages of customers
		Mean	Desviation	with time windows
R1	01	20	0	100%
R1	02	20	0	75%
R1	03	20	0	50%
R1	04	20	0	25%
R1	05	60	0	100%
R1	06	60	0	75%
R1	07	60	0	50%
R1	08	60	0	25%
R1	09	120	15	100%
R1	10	170	65	100%
R1	11	180	95	100%
R1	12	230	25	100%
C1	01	20	0	100%
C1	02	20	0	75%
C1	03	20	0	50%
C1	04	20	0	25%
C1	05	60	0	100%
C1	06	60	0	75%
C1	07	60	0	50%
C1	08	60	0	25%
C1	09	120	15	25%
RC1	01	20	0	100%
RC1	02	20	0	75%
RC1	03	20	0	50%
RC1	04	20	0	25%
RC1	05	60	0	100%
RC1	06	60	0	75%
RC1	07	60	0	50%
RC1	08	60	0	25%

Following the example of the Fig. 5 the expected transit time, $\overline{t_{AB}}$, and the real transit time, t_{AB}, are calculated with the formulates 20 and 21.

$$\overline{t_i} = d_i/\overline{v_{xy}} \qquad \overline{t_{AB}} = \sum_{i=1:4} \overline{t_i} \tag{20}$$

$$t_i = d_i/v_{xy} \qquad t_{AB} = \sum_{i=1:4} t_i \tag{21}$$

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 [7] procedure as our benchmark reference. We also selected this benchmark algorithm due to its good properties with respect to computational and solution quality, considering that the authors compare their algorithm with an exact one, obtaining very satisfactory results.

Table 2 represents the average total cost of each approach and the average improvement of the dynamic approach with respect to the static one. The dynamic methodologies show improvements in all the problem classes and in almost all instances. In general, our methodology achieves better results when customers are distributed randomly. Both methodologies show similar properties respect to the quality of the solution, but depending of the instance one or other method is better. However, RT-2PIH is better in terms of the running time of the algorithm. A single run of the RT-2PIH algorithm usually takes a few seconds on an Inter Core Duo 2,4 GHz, so the computational cost of that heuristic is low enough for the problem. However, the genetic algorithm takes about 40-50 seconds.

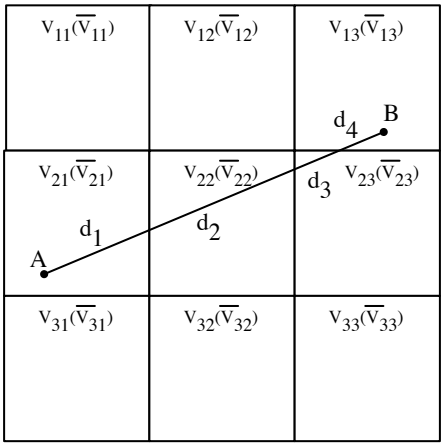

Figure 5: Chromosome

Table 3 presents an overview of every component of the objective function and Table 4 shows the average number of times that vehicles arrive late to the origin or destination of a task. Class R1 has been taken as an example. The problem class and the instance are shown in the first two columns. Distance, number of vehicles, and penalty costs are shown for every approach: two-phase insertion heuristic (2PIH), real-time two-phase insertion heuristic (RT-2PIH), and two-phase insertion heuristic improved in every re-optimization with a genetic algorithm (2PIH-GA). The results show that the number of occasions in which time windows are broken decrease using the dynamic methods. The RT-2PIH usually is more effective respect to the number of time windows broken, but the 2PIH+GA usually uses a smaller number of vehicles.

6. CONCLUSION

In this paper we have shown the importance of obtaining real-time data of vehicle locations in a drayage fleet through the use of a satellite positioning system. This knowledge, together with an optimization algorithm based on heuristics and metaheuristics, enables the dynamic management of the fleet in a changing environment, which reduces average operation costs by as much as 5%, sometimes achieving 13% improvements. These results are especially valuable for intermodal operations in congested metropolitan areas, where travel times are stochastic due to congestion.

This dynamic method reduces drastically the number of time windows broken and the penalty costs due to it. This is very important in the DDPTW, due to the strict timetable in the terminal, but when the time windows are not very restrictive, the availability of real-time information is less relevant.

To solve the drayage problem, we developed a real-time optimization model based on either a genetic algorithm or an adaptation of the heuristic presented in [7]. We operated with cost estimations, and tested our algorithms with a series of drayage problems generated through an adaptation of the Li and Lim Benchmark. The best results in terms of speed and flexibility of adaptation were obtained by the dynamic two-phase insertion heuristic.

Table 2: Average total costs and improvement in problem instances

Problem		Total Cost			Improvement (%)	
Instance	Customers	2PIH	RT-2PIH	2PIH+GA	RT-2PIH	2PIH-GA
R1-1	25	1315,1	1234,9	1219,4	6,10%	7,28%
R1-2	25	1147,6	1072,3	1086,0	6,56%	5,37%
R1-3	25	954,6	886,9	897,5	7,09%	5,99%
R1-4	25	875,3	862,1	868,0	1,50%	0,83%
R1-5	25	1246,7	1167,2	1180,0	6,38%	5,35%
R1-6	25	1175,3	1127,6	1041,6	4,06%	11,37%
R1-7	25	1110,9	996,8	994,9	10,27%	10,44%
R1-8	25	1064,4	975,5	1011,4	8,35%	4,98%
R1-9	25	1130,4	1043,7	1097,6	7,67%	2,90%
R1-10	25	1029,3	950,7	978,8	7,64%	4,90%
R1-11	25	1173,6	1039,4	1030,1	11,43%	12,23%
R1-12	25	960,8	963,9	951,1	-0,32%	1,01%
C1-1	25	926,7	888,2	810,1	4,16%	12,58%
C1-2	25	831,6	790,2	768,6	4,98%	7,58%
C1-3	25	676,6	673,5	673,2	0,47%	0,50%
C1-4	25	680,6	675,8	673,7	0,70%	1,01%
C1-5	25	813,7	766,2	743,7	5,83%	8,60%
C1-6	25	942,3	920,4	821,7	2,32%	12,79%
C1-7	25	762,0	757,2	721,6	0,64%	5,31%
C1-8	25	653,0	662,4	650,6	-1,44%	0,36%
C1-9	25	702,4	694,9	684,5	1,08%	2,56%
RC1-1	25	1764,9	1693,4	1690,0	4,05%	4,24%
RC1-2	25	1458,7	1344,8	1329,9	7,81%	8,83%
RC1-3	25	1363,2	1294,1	1244,2	5,06%	8,73%
RC1-4	25	1189,2	1136,3	1165,7	4,45%	1,98%
RC1-5	25	1683,1	1585,9	1585,5	5,78%	5,80%
RC1-6	25	1649,1	1611,7	1574,9	2,27%	4,50%
RC1-7	25	1371,1	1359,2	1355,7	0,87%	1,13%
RC1-8	25	1433,9	1312,7	1374,5	8,45%	4,14%
R1-1	50	2360,5	2356,5	2299,5	0,17%	2,58%
R1-2	50	2064,4	2018,5	2014,4	2,22%	2,42%
R1-3	50	1938,7	1891,5	1895,2	2,43%	2,24%
R1-4	50	1817,5	1773,7	1769,1	2,41%	2,67%
R1-5	50	2327,3	2322,1	2261,6	0,22%	2,82%
R1-6	50	2089,0	2039,8	2013,0	2,35%	3,64%
R1-7	50	2007,6	1866,2	1880,0	7,04%	6,36%
R1-8	50	1929,1	1745,9	1797,7	9,50%	6,81%
R1-9	50	2259,7	2139,4	2174,7	5,32%	3,76%
R1-10	50	2067,0	1994,2	1990,5	3,52%	3,70%
R1-11	50	2103,7	1998,7	1972,5	4,99%	6,24%
R1-12	50	2065,6	1988,5	1865,9	3,73%	9,67%
C1-1	50	1805,3	1771,7	1699,4	1,86%	5,87%
C1-2	50	1729,2	1650,2	1618,4	4,57%	6,41%
C1-3	50	1587,1	1544,5	1465,0	2,69%	7,70%
C1-4	50	1525,3	1474,0	1473,7	3,36%	3,39%
C1-5	50	1614,6	1656,9	1571,4	-2,62%	2,67%
C1-6	50	1793,7	1742,3	1653,1	2,87%	7,84%
C1-7	50	1547,7	1530,7	1528,3	1,10%	1,26%
C1-8	50	1436,1	1408,0	1400,8	1,96%	2,46%
C1-9	50	1570,8	1513,5	1476,9	3,65%	5,98%
RC1-1	50	3680,4	3622,9	3490,2	1,56%	5,17%
RC1-2	50	3015,0	2930,0	2860,4	2,82%	5,13%
RC1-3	50	2708,5	2584,9	2610,2	4,56%	3,63%
RC1-4	50	2530,4	2339,0	2414,3	7,56%	4,59%
RC1-5	50	3408,3	3283,8	3260,1	3,65%	4,35%
RC1-6	50	3438,0	3373,4	3308,2	1,88%	3,78%
RC1-7	50	2802,2	2637,1	2712,3	5,89%	3,21%
RC1-8	50	2871,5	2667,8	2744,4	7,09%	4,43%

This work opens the door to future research considering stochastic transit time and dynamic optimization, and the use of multi-objective functions or rolling windows are also in progress to be considered in the future.

Table 3: Average costs in problem instances

Problem		Distance			Number of Vehicles			Penalty Cost		
Class	Instance	2P IH	RT-2P IH	2PIH+GA	2PIH	RT-2PIH	2PIH+GA	2PIH	RT-2PIH	2PIH+GA
R1	1	978,1	999,5	994,9	11,0	11,7	10,4	227,1	118,2	120,8
R1	2	961,4	961,4	961,4	7,0	7,5	7,0	116,2	36,4	54,5
R1	3	821,7	821,4	820,0	7,0	6,5	6,8	63,0	0,1	9,3
R1	4	810,3	810,3	808,9	5,0	5,2	5,0	14,9	0,0	9,1
R1	5	974,3	967,2	970,9	8,0	9,1	8,2	192,5	109,1	127,3
R1	6	940,6	950,3	941,4	7,0	7,7	7,3	164,7	100,0	27,5
R1	7	925,3	925,9	924,0	8,0	7,1	7,1	105,6	0,0	0,0
R1	8	898,3	898,3	894,4	7,0	5,9	6,2	96,1	18,2	55,2
R1	9	885,2	900,1	896,7	6,0	7,1	6,5	185,2	72,7	136,4
R1	10	837,7	841,6	849,7	7,0	7,3	7,5	121,6	36,4	54,5
R1	11	892,5	906,7	900,5	7,0	8,7	7,5	211,0	45,5	55,0
R1	12	891,1	891,1	889,8	6,0	6,4	6,1	9,7	9,1	0,4

Table 4: Average number of times when time windows are broken.

Problem		Late arrive to origin			Late arrive to destination		
Class	Instance	2P IH	RT-2P IH	2PIH+GA	2PIH	RT-2PIH	2PIH+GA
R1	1	1,4	0,0	0,9	2,1	1,2	1,2
R1	2	0,9	0,0	0,0	1,0	0,4	0,5
R1	3	0,5	0,2	0,2	0,5	0,0	0,1
R1	4	0,0	0,0	0,0	0,1	0,0	0,1
R1	5	0,2	0,0	0,0	1,8	1,1	1,3
R1	6	0,1	0,0	0,1	1,5	1,0	0,3
R1	7	0,0	0,0	0,0	1,0	0,0	0,0
R1	8	0,5	0,0	0,5	0,9	0,2	0,5
R1	9	0,5	0,0	0,1	1,8	0,7	1,4
R1	10	0,0	0,0	0,0	1,2	0,4	0,5
R1	11	0,0	0,0	0,1	2,1	0,5	0,5
R1	12	0,5	0,1	0,4	0,1	0,1	0,0

References

[1] Banzhaf, W., 1990. The molecular traveling salesman. Biological Cybernetics 64, 7–14.

[2] Bodin, L., Golden, B., Assad, A., Ball, M., 1983. Routing and scheduling of vehicles and crews: The state of the art. Computers and Operations Research 10 (2), 63–211.

[3] Bodin, L., Mingozzi, A., Baldacci, R., Ball, M., 2000. The rollon-rolloff vehicle routing problem. Transportation Science 34 (3), 271.

[4] Braekers, K., Caris, A., Janssens, G., 2012. Integrated planning of loaded and empty container movements. Or Spectrum, 1–22.

[5] Braysy, O., Gendreau, M., 2005. Vehicle routing problem with time windows, part ii: Metaheuristics. Transportation Science 39 (1), 119–139.

[6] Caris, A., Janssens, G., 2010. A deterministic annealing algorithm for the pre-and end-haulage of intermodal container terminals. International Journal of Computer Aided Engineering and Technology 2 (4), 340–355.

[7] Caris, A., Janssens, G. K., 2009. A local search heuristic for the pre-and end-haulage of intermodal container terminals. Computers and Operations Research 36 (10), 2763–2772.

[8] Cheung, R. K., Hang, D. D., 2003. A time-window sliding procedure for driver-task assignment with random service times. IIE Transactions 35 (5), 433–444.

[9] Currie, R. H., Salhi, S., 2003. Exact and heuristic methods for a full-load, multi-terminal, vehicle scheduling problem with backhauling and time windows. Journal of the Operational Research Society, 390–400.

[10] Currie, R. H., Salhi, S., 2004. A tabu search heuristic for a full-load, multi-terminal, vehicle scheduling problem with backhauling and time windows. Journal of Mathematical Modelling and Algorithms 3 (3), 225–243.

[11] Davis, L., 1985. Applying adaptive algorithms to epistatic domains. In: Proceedings of the International Joint Conference on Artificial Intelligence. pp. 162–164.

[12] De Meulemeester, L., Laporte, G., Louveaux, F. V., Semet, F., 1997. Optimal sequencing of skip collections and deliveries. The Journal of the Operational Research Society 48 (1), 57–64.

[13] Desrochers, M., Desrosiers, J., Solomon, M. M., 1992. A new optimization algorithm for the vehicle routing problem with time windows. Operations Research 40 (2), 342–354.

[14] Desrosiers, J., Dumas, Y., Solomon, M. M., Soumis, F., Ball, M., Magnati, T. L., Monma, C. L., Nemhauser, G. L., 1995. Time constrained routing and scheduling. In: Network routing, Handbook in Operation Research and management Science. Vol. 8. INFORMS Elsevier Science, pp. 35–130.

[15] Dumas, Y., Desrosiers, J., Gelinas, E., Solomon, M. M., 1995. An optimal algorithm for the traveling salesman problem with time windows. Operations Research, 367–371.

[16] Erera, A. L., Smilowitz, K. R., Ioannou, P., 2008. Intermodal drayage routing and scheduling. In: Intelligent Freight Transportation. CRC Press. Taylor & Francis Group, pp. 171–188.
[17] Escudero, A., Munuzuri, J., Arango, C., Onieva, L., 2011. A satellite navigation system to improve the management of intermodal drayage. Advanced Engineering Informatics 25 (3), 427–434.
[18] Fogel, D. B., 1988. An evolutionary approach to the traveling salesman problem. Biological Cybernetics 60, 139–144.
[19] Francis, P., Zhang, G., Smilowitz, K. R., 2007. Improved modeling and solution methods for the multi-resource routing problem. European Journal of Operational Research 180 (3), 1045–1059.
[20] Goldberg, D., Lingle Jr, R., 1985. Alleles, loci and the traveling salesman problem. In: Proceedings of the 1st International Conference on Genetic Algorithms. L. Erlbaum Associates Inc., pp. 154–159.
[21] Goldberg, D. E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley.
[22] Gronalt, M., Hartl, R. F., Reimann, M., 2003. New savings based algorithms for time constrained pickup and delivery of full truckloads. European Journal of Operational Research 151 (3), 520–535.
[23] Holland, J., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Michigan.
[24] Ileri, Y., Bazaraa, M., Gifford, T., Nemhauser, G. L., Sokol, J., Wikum, E., 2006. An optimization approach for planning daily drayage operations. Central European Journal of Operations Research 14 (2), 141–156.
[25] Imai, A., Nishimura, E., Current, J., 2007. A lagrangian relaxation-based heuristic for the vehicle routing with full container load. European Journal of Operational Research 176 (1), 87–105.
[26] Jula, H., Dessouky, M. M., Ioannou, P., Chassiakos, A., 2005. Container movement by trucks in metropolitan networks: modeling and optimization. Transportation Research Part E 41 (3), 235–259.
[27] Laporte, G., 1992. The vehicle routing problem: An overview of exact and approximate algorithms. European Journal of Operational Research 59 (3), 345–358.
[28] Larrañaga, P., Kuijpers, C., Murga, R., Inza, I., Dizdarevich, S., 1999. Evolutionary algorithms for the travelling salesman problem: A review of representations and operators. Artificial Intelligence Review 13 (2), 129–170.
[29] Li, H., Lim, A., 2001. A metaheuristic for the pickup and delivery problem with time windows. In: Tools with Artificial Intelligence, Proceedings of the 13th International Conference on. IEEE, pp. 160–167.
[30] Máhr, T., Srour, J., de Weerdt, M., Zuidwijk, R., 2010. Can agents measure up? a comparative study of an agent-based and on-line optimization approach for a drayage problem with uncertainty. Transportation Research Part C: Emerging Technologies 18 (1), 99–119.
[31] Mak, K., Guo, Z., 2004. A genetic algorithm for vehicle routing problems with stochastic demand and soft time windows. In: Systems and Information Engineering Design Symposium. IEEE, pp. 183–190.
[32] Michalewicz, Z., 1992. Genetic algorithms+ data structures. Springer.
[33] Morlok, E., Spasovic, L., 1995. Approaches to improving drayage in rail-truck intermodal service. In: IEEE TransTech Conference, 30 Jul - 2 Aug. IEEE, Seattle, WA , USA, pp. 74–80.
[34] Morlok, E. K., Spasovic, L. N., 1994. Redesigning rail-truck intermodal drayage operations for enhanced service and cost performance. Journal of Transportation Research Forum 34 (1), 16–31.
[35] Namboothiri, R., Erera, A. L., 2008. Planning local container drayage operations given a port access appointment system. Transportation Research Part E. 44 (2), 185–202.
[36] Oliver, I., Smith, D., Holland, J., 1987. A study of permutation crossover operators on the tsp. In: Grefenstette, J. J. (Ed.), Genetic Algorithms and Their Applications: Proceedings of the Second International Conference. Lawrence Erlbaum, Hillsdale, New Jersey, pp. 224–230.
[37] Smilowitz, K. R., 2006. Multi-resource routing with flexible tasks: an application in drayage operations. IIE Transactions 38 (7), 577–590.
[38] Solomon, M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35 (2), 254–265.
[39] Spasovic, L. N., Morlok, E. K., 1993. Using marginal costs to evaluate drayage rates in rail-truck intermodal service. Transportation Research Record 1383 (1), 8–16.
[40] Syswerda, G., 1991. Schedule optimization using genetic algorithms. In: David, L. (Ed.), Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York, pp. 332–349.
[41] Wang, X., Regan, A. C., 2002. Local truckload pickup and delivery with hard time window constraints. Transportation Research Part B 36 (2), 97–112.
[42] Zhang, G. M., Smilowitz, K., Erera, A., 2011. Dynamic planning for urban drayage operations. Transportation Research Part E-Logistics and Transportation Review 47 (5), 764–777.
[43] Zhang, R. Y., Yun, W., Kopfer, H., 2010. Heuristic-based truck scheduling for inland container transportation. Or Spectrum 32 (3), 787–808.
[44] Zhang, R. Y., Yun, W. Y., Moon, I., 2009. A reactive tabu search algorithm for the multi-depot container truck transportation problem. Transportation Research Part E-Logistics and Transportation Review 45 (6), 904–914.
[45] Zhang, R. Y., Yun, W. Y., Moon, I. K., 2011. Modeling and optimization of a container drayage problem with resource constraints. International Journal of Production Economics 133 (1), 351–359.