Combining Linear Programming and Automated Planning to Solve Intermodal Transportation Problems

Javier García[a], José E. Florez[a], Álvaro Torralba[a], Daniel Borrajo[a], Carlos Linares López[a], Ángel García-Olaya[a], Juan Sáenz[b]

[a]*Computer Science Department, Universidad Carlos III de Madrid*
Avenida de la Universidad 30, 28911 Leganés, Madrid, Spain
[b]*Acciona I+D+i / Acciona R&D*
Valportillo II 8, 28108 Alcobendas, Madrid, Spain

Abstract

When dealing with transportation problems Operational Research (OR), and related areas as Artificial Intelligence (AI), have focused mostly on uni-modal transport problems. Due to the current existence of bigger international logistics companies, transportation problems are becoming increasingly more complex. One of the complexities arises from the use of intermodal transportation. Intermodal transportation reflects the combination of at least two modes of transport in a single transport chain, without a change of container for the goods. In this paper, a new hybrid approach is described which addresses complex intermodal transport problems. It combines OR techniques with AI search methods in order to obtain good quality solutions, by exploiting the benefits of both kinds of techniques. The solution has been applied to a real world problem from one of the largest spanish companies using intermodal transportation, Acciona Transmediterránea Cargo.

Keywords: Logistics, Intermodal transport, Linear programming, Heuristic Planning

1. Introduction

Nowadays, intermodal transportation plays a key role in international logistics. Intermodal transport commonly refers to the combination of two or more modes of movement of goods, such as road, rail, or sea [1, 2]. In this type of task, the use of Operations Research (OR) is still limited. Research on logistic problems usually focuses

Email addresses: `fjgpolo@inf.uc3m.es` (Javier García), `jflorez@inf.uc3m.es` (José E. Florez), `atorralb@inf.uc3m.es` (Álvaro Torralba), `dborrajo@ia.uc3m.es` (Daniel Borrajo), `carlos.linares@uc3m.es` (Carlos Linares López), `agolaya@inf.uc3m.es` (Ángel García-Olaya), `jsaenz@acciona.es` (Juan Sáenz)

only on one mode of movement of goods, whether by road [3, 4, 5], rail [6], or sea [7]. However, there are two observations that make intermodal transportation problems more challenging than the uni-modal one: on one hand, the optimal path is not the shortest path anymore; instead, additional costs have to be considered at the nodes where a new transportation mean is applicable e.g., money and/or time. On the other hand, a new class of constraints has to be observed, which (to make things harder) is dependant on each node e.g., operating an exchange of transportation mean can actually involve other subproblems as it happens when moving goods from a truck to a ship. Thus, it provides very interesting and challenging tasks for researchers working in OR and related areas, such as heuristic search.

This paper deals with a real-world problem and in particular it focuses on the specific intermodal transportation problem of the spanish company Acciona Transmediternea Cargo. As presented in the Related work section, in the literature, we did not find any article describing the same intermodal transport problem. This particular problem fits into the *intermodal* chain of *container*-transportation services described in the literature [1]. This chain usually links the initial pick-up point to the final delivery point of the container, visiting in between different pick-up and delivery points. Transportation is provided by several carriers. Our contribution attempts to provide a new application, TIMIPLAN, to solve problems of this kind. The planning component of TIMIPLAN consists of two phases: in phase one, for each set of goods to be picked up and delivered, the containers and trucks with minimum estimated cost to complete the service are selected. In this phase, several assignment models are constructed and solved as linear programming problems. In phase two, an Artificial Intelligence (AI) planner is used to select the best (cheapest) plan to serve each service: from a first pick-up point to the last delivery point over the service. The plan should fulfill a given set of constraints (temporal and regulatory), and will include the sequence of the transportation modes to be used. Although some of the application areas addressed in AI and OR are very similar (e.g., planning, scheduling), the methods that are used to solve these problems are substantially different. This paper describes the application we have developed for a big logistics company, and provide some experimental results that evaluate the software in real situations extracted from the customer database.

The remainder of the paper is organized as follows. Section 2 gives a brief summary of the transportation problem in its uni-modal and intermodal versions, introducing some of the main approaches used to solve it. Section 3 describes the intermodal problem in detail. Section 4 presents the TIMIPLAN algorithm. In this section, two ways to tackle the assignment problem of containers and trucks to services are described. In addition, in this section the planning module to select the best transportation modes is explained in detail. Section 5 shows the experiments performed. Also, it describes some comparative results of the two versions of the TIMIPLAN algorithm. Lastly, Section 6 presents the conclusions and further research.

2. Related Work

Many approaches have dealt with the uni-modal transport problem. Coslovich et al. [5] focus on the container transportation problem taking into account the routing costs, the resources, and the container repositioning costs. Their approach has several similarities and analogies with our work (also in the solution approach, which uses the decomposition into subproblems). However, their approach focuses only on a fleet management problem that arises in a truck company, excluding the transport by ship or rail (significantly reducing the complexity of the problem). Powell et al. [3] tackle the problem of assigning drivers to loads in order to minimize the empty miles, but, again, only considering truck transportation. An interesting uni-modal solution for large transportation problems also is proposed by Sprenger et al. [8].

There has also been some work in the intermodal transport problem [1, 2]. Chang [9] focuses his study on how to select the best routes for shipments in an intermodal network. Because the entire problem is NP-hard [9], the original problem is broken into a set of smaller and easier subproblems, based on Lagrangian relaxation and decomposition techniques. However, in contrast with us, Chang assumes that the trucks drivers have no time constraints, and, additionally, Chang only reports results for problems carrying 10, 6, and 8 containers from the suppliers to the customer, over a small network graph of 112 nodes and only 407 links. Our biggest problem requires 300 containers movements over a network graph of 600 nodes and more than 3×10^5 links. Imai et al. [10] also propose to decompose the original problem into two different subproblems using a subgradient heuristic based on Lagrangian relaxation. However, they partially tackle the problem of intermodal transportation, as long as they only consider the problem of vehicle routing that arises in picking up and delivering full container load from/to an intermodal terminal (regardless of schedules of ships or trains). Bock [11] addresses a similar multimodal problem using LP techniques but with several important differences with the intermodal problem presented here. On one hand, Bock does not consider the containers in his multimodal problem and, hence, does not take into account the initial assignment of trucks to containers, which implies a significant reduction of the problem complexity. On the other hand, the biggest problem he considers has only 65 vehicles and 5 global transportation hubs. The biggest problem our approach deals with here has 300 trucks, 300 containers and more than 150 ship and train segments. Instead, Verma et al. [12] focus on the transport of hazardous material using truck-rail multimodal transportation (i.e. obviating the ship transportation mode) over a well-defined region of the US with only 37 shipper/receivers and 31 train segments and, additionally, assuming that the drivers have no time constraints. Gromicho et al. [13] propose an interesting and promising approach but considering only a single network to transport a container from a pick-up node to a delivery node. Therefore, they do not compute what container is the best to complete a service, and they not address the problem of solving multiple services sharing the same resources. Thus, all these approaches cannot be directly applied to our problem described in Section 3.

Also, there have already been some approaches that try to combine AI techniques with OR techniques. Bylander [14] uses linear programming as a heuristic that improves the search process in nonlinear planning; Kautz [15] uses linear programming formulations for planning problems with different resources, action costs, and difficult objective functions; Fernández [16] solves the clustered-oversubscription problem by performing an action selection pre-processing to help the planning task using linear programming. In comparison with these works, we propose to use linear programming combined with planning in a different way. In our case, linear programming is used first to compute the assignment cost of resources to services in order to find the assignment with the least estimated cost. Later, we formulate each task as a planning problem where the previously selected resources are taken into account.

While OR techniques solve problems that can be modeled with linear constraints very efficiently, we advocate their combined usage with other general techniques like Automated Planning (AP) [17]. One of the main reasons is that Automated Planning actually starts by considering a very expressive language which usually overcomes some of the difficulties found when modeling a problem with linear constraints. Also, the standard language considered in AP (most likely PDDL [18] but also many other variants) is very well suited to represent a wider class of problems. These differences do not only apply to constraints, but also to the objective function, since AP can use non-linear optimization functions.

3. Problem Description

This paper focuses on the container-based intermodal transportation problem (transportation of containerized cargo by a combination of truck, rail, and ocean shipping, to move massive quantities of containers) [1], but with the requirements of the company Acciona Transmediterránea Cargo. Formally, our intermodal transportation problem can be defined as the tuple $< G, F, C, R, B, S >$ where G is the network graph, F, C, R and B are the sets of trucks, containers, trains and ships respectively and S the services that should be fulfilled. Let $G = (P, E)$ be a directed graph, where P and E are, respectively the set of nodes and set of edges representing a direction in which the corresponding arc can be traversed. In intermodal transportation there are different kinds of nodes: all nodes in P represent nodes that can be reached by trucks and containers, and $M \subseteq P$, represents intermodal nodes where one has to select between continuing with the current mode or changing it (as ports or train stations). Also, there are different kinds of edges representing roads, rails or ship routes.

Table 1: Truck and Container features.

Truck	t_f	Time counter of truck f
	p_f	Starting point, $p_f \in P$, where the vehicle is located at time t_f
	s_f	Mean speed denoted
	e_f	Cost per kilometer when truck f travels without any container or the container is empty
	l_f	Cost per kilometer when truck f travels with a loaded container
	cs_f	Cost per hour when truck f is stopped at a location
Container	t_c	Time counter of container c
	p_c	Starting point, $p_c \in P$, where the container is located at time t_c

Let F be a set of trucks, with $|F| = n_f$, and let C be a set of containers, with $|C| = n_c$. Each truck $f \in F$ and container $c \in C$ is characterized by a series of parameters as shown in Table 1. The existing temporal restrictions in the problem (each pick-up and delivery is scheduled according to the service time of each place) imply that an explicit management of the current time is needed. If a truck arrived early to a pick-up or delivery point, it must wait, and when it arrives late a penalty cost is applied. In addition, a container must wait at the station or port for the next departure of the train or ship. So, associated to each truck and container, there is a time counter, t_f and t_c. The truck and container movements conveniently increase the value of these time counters. Let also R be a set of trains, and let B denote a set of container ships. Each train $r \in R$ and $b \in B$ are characterized as shown in Table 2.

Table 2: Train and Ship features.

$m_0, m_1, ..., m_k$	Any predefined train/ship route is given by the ordered sequence of nodes $m_0, m_1, ..., m_k$ where $m_i \in M$ and $k > 0$. Stations/ports $m_0, m_1, ..., m_k$ form the train/ship route if train/ship starts its journey at m_0 and visits consecutively $m_1, ..., m_k$
P_{ij}	The time at which the train or ship leaves the node (station or port) $m_i \in M$ in direction to node (station or port) $m_j \in M$
D_{ij}	The time spent to go from m_i to m_j
ut	Time spent loading a container in the train or ship
dt	Time spent unloading a container from the train or ship

Let S be a set of services, with $|S| = n_s$. For each service $s \in S$ a predefined route is given by the ordered sequence of points $U_s = (u_1, u_2, ..., u_j)$, where u_1 is a pick-up point, u_j is a delivery point, and u_k, $1 < k < j$, is a delivery or pick-up point. The pick-up and delivery order is set by the Acciona's client requesting the service. In addition, each $u \in U_s$ is located in one node $p \in P$. Each pick-up or delivery point is characterized in Table 3.

Table 3: Pick-up or Delivery points.

pt_u	The pickup or delivery time, pt_u, which indicates the time at which the corresponding point u is available for the pick-up or delivery service.
st_u	The service time, or the time spent to complete the pick-up or delivery service
pc_u	The penalty cost per hour, applied when the pick-up or delivery is delayed.

In intermodal transportation, several trucks are usually needed. For example, Figure 1 shows how, in order to complete the service, there are five available trucks, one container, two trains and two ships. The first truck with the container picks the shipment up from *Pick–Up*$_1$ and transports it to *Pick–Up*$_2$ using either road or train. If the train option is selected, another truck will be necessary to transport the container to *Pick–Up*$_2$. Also, there are two other decision points related to the use of *Ship*$_1$ and *Train*$_2$. The use of *Ship*$_2$ and *Truck*$_4$ is mandatory for reaching the *Pick–Up*$_3$ point.

Notice that in the particular intermodal transportation problem of Acciona, only one container is used for each service, because it is sufficient to transport the amount of goods associated to the service. The planner is executed every day. A daily problem has approximately 600 locations (summing up all pick-up and delivery locations, as well as initial positions of trucks, containers, ships, and trains), more than 3×10^5 edges among those locations, 300 trucks, 300 containers, 300 services, 50 train segments and 150 ship

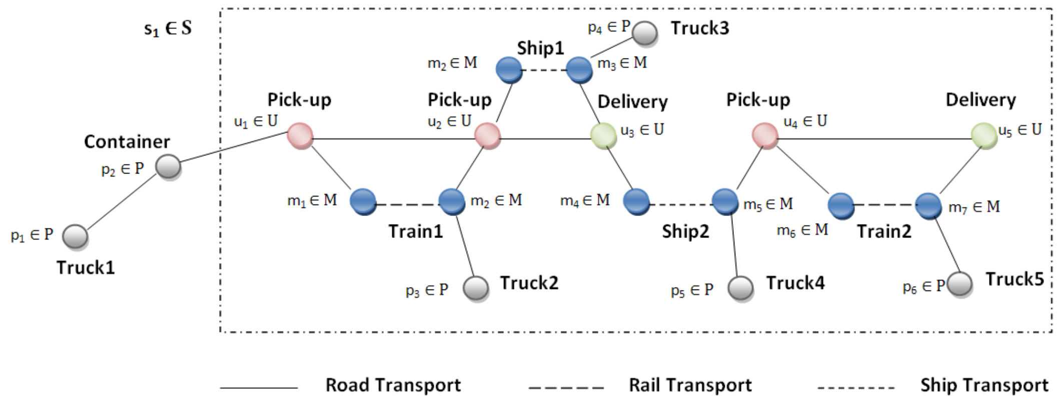

Figure 1: Example of Intermodal transportation graph.

segments. The company imposes a time limit of 2 hours for computing the daily plan.

4. The TIMIPLAN Algorithm

The first approach we followed consisted on trying to solve the complete problem by using an automated planner. Automated Planning is concerned with the automatic generation of a plan to solve a problem within a particular domain. At its simplest, a plan is a sequence of actions. Given an initial state, the planner tries to find the actions such that their ordered execution from the initial state achieves some goal conditions. Planners can be domain-dependent or domain-independent. In domain-dependent planning, there is no division between the solver and the domain knowledge. In general terms, they are very efficient. However, they are usually difficult to build and they have to be rewritten for each domain. On the other hand, domain-independent planners are not tied to one particular domain (they can solve problems in a variety of different domains, given a model of that domain in a suitable input language). The domain model specifies the actions available to the planner. Each action may be executed only in some set of world states (defined by its preconditions), and has some particular set of effects on the states where they are applied. A planning problem consists of a domain model together with an initial state of the world, and a desired goal state (or set of goal states). A planner solves a planning problem by producing a sequence of instantiated actions (plan), which takes the initial state to a goal state. Each action takes variable parameters and an instantiated action is obtained only when those parameters are assigned constant values from the state. Usually, these variables are prefixed with a question mark: *?truck, ?container*.

Figure 2 shows an example of planning process where the truck *Truck1* is driven from location *L1* to location *L4*.

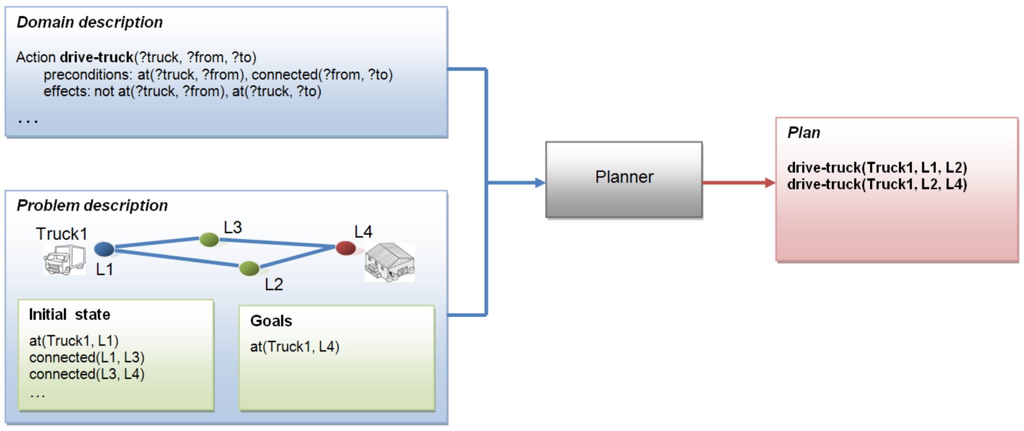
Domain description
Action drive-truck(?truck, ?from, ?to)
preconditions: at(?truck, ?from), connected(?from, ?to)
effects: not at(?truck, ?from), at(?truck, ?to)
...
Problem description
Truck1
L1
L3
L2
L4
Initial state
at(Truck1, L1)
connected(L1, L3)
connected(L3, L4)
...
Goals
at(Truck1, L4)
Planner
Plan
drive-truck(Truck1, L1, L2)
drive-truck(Truck1, L2, L4)

Figure 2: Example of a planning process.

Given that planning is the process of finding a set of instantiated actions that transforms an initial state into a goal state, many possible instantiations of actions make planning a hard combinatorial problem. Unfortunately, in the case of intermodal transportation, given the size of the problem, no domain-independent planner can go beyond an initial instantiation phase, common in most of the state of the art planners. For instance, the load-truck action has three parameters (truck, container, location). In our problems, there are around 300 trucks and containers, and around 600 locations. Thus, there are about $300 \times 300 \times 600 = 54$ million potential instantiations of that action solely. Table 4 shows the mean time (in seconds) to solve problems of size *N trucks* $\times$ *N containers* $\times$ *N services* with different planners: SGPLAN [19], LPG [20] and SAYPHI [21]. In all cases, the time grows exponentially with the size of the problem, and the problem $3 \times 3 \times 3$ is not solved by any planner within a time limit of 500 seconds. However, SAYPHI is able to explicitly reason about the cost of the solutions, so we prefer to use SAYPHI to perform the experiments in Section 5. In a similar way, linear programming experiences a combinatorial explosion due to the huge number of resources involved. In this case, because the intermodal transportation problem is NP-Hard, some type of decomposition is required [5, 10, 9].

Our approach sequentially solves the problem, using three different steps for each service. In step one, the container and truck/s with minimum cost estimated to complete the service are selected (using LP to solve a classical assignment problem). In step

Problem Size	**SGPLAN522**	**LPG1.2**	**SAYPHI**
$1 \times 1 \times 1$	0.0 *s*	4.272 *s*	0.12 *s*
$2 \times 2 \times 2$	0.57 *s*	-	17.55 *s*
$3 \times 3 \times 3$	-	-	-

Table 4: The mean times to solve problems of different size by SGPLAN, LPG and SAYPHI. Each mean time has been computed from 5 different executions. The symbol "-" means that the planner is not able to obtain a solution within a time limit of 500 seconds. The experiments were conducted on a 2,4 GHz quadcore processor with 4 GB RAM, running Linux.

two, a planning module is used to select the *best* path from a first pick-up point to the last delivery point over the service. In this case, *best* means that the path fulfills the given set of constraints, including the sequence of the transportation modes used (where several trains and/or ships can be used) with the minimum cost. This two-step approach balances the total cost obtained and the time required to compute the plan. The high level algorithm has been depicted in Table 5. The network graph is the graph defined by the locations (pick-up and delivery nodes, positions of trucks, containers, train stations and ports) and edges (roads, rails and ship lines). In step three, TIMIPLAN updates the assignment of trucks and containers to services taking into account the final position of the trucks and containers used to complete the last planned service. In this third step, it uses the same LP approach again.

```
TIMIPLAN (G, F, C, R, B, S)
;; Inputs: the graph (G), the set of trucks (F), containers (C), trains (R), ships (B) and services (S)
$plan = \emptyset$
;; Compute the initial assignment of trucks and containers to services (A)
$A$ = solveAssignmentProblem(G, F, C, R, B, S)
For each $s \in S$
    ;; Select the truck/s and container to complete the service
    selectedTrucks,selectedContainer= getServiceAssigment(A, s)
    ;; Plan the service with the truck/s and container selected. Select the best transportation modes
    $plan = \cup$ {solvePlanningProblem(selectedTrucks, selectedContainer, R, B, s)}
    ;; Updates assignment with the new cost of selectedTrucks and selectedContainer
    $A$ = updateAssignmentProblem(G, F, C, R, B, S)
Return $plan$
```

Table 5: Top level algorithm of TIMIPLAN.

4.1. Assignment Problem

The goal is to minimize the total cost of the assignments of truck/s and container to complete the set of services. However, this problem is quite complex because there are a great number of trucks, containers and services involved. Consider the simplified version of the intermodal transportation problem of assigning an initial pair truck-container to each service, such that the total cost of all assignments is minimal. In this assignment problem, the cost matrix has size $M \times N$, where M represents all possible combinations of trucks with containers, N is the number of services, and each cell in the cost matrix is the cost of associating the pair truck-container in a row with a service in a column.

Cost Matrix Size $M \times N$	**GLPSOL4.25**	**MINIZINC1.3.1**
$2500_{(50\ trucks \times 50\ containers)} \times 50$	80.0 s	834.2 s
$10000_{(100\ trucks \times 100\ containers)} \times 100$	396.4 s	-
$22500_{(150\ trucks \times 150\ containers)} \times 150$	-	-

Table 6: Mean time to solve assignment problems with cost matrix of different sizes by GLPSOL, and MINIZINC. Each mean time has been computed from 5 different executions. The symbol "-" means that the LP solver produces a memory fault. The experiments were conducted on a 2,4 GHz quadcore processor with 4 GB RAM, running Linux.

Table 6 shows the mean time to solve the assignment problem with different cost matrix sizes, $M \times N$, using two LP solvers: GLPSOL [22] and MINIZINC [23]. The LP solvers are not able to solve the assignment problem of 22500 $(pairs\ truck{-}container) \times 150\ (services)$, i.e. the assignment problem of $150\ trucks \times 150\ containers \times 150\ services$. However, our biggest problem is composed of 300 trucks, 300 containers and 300 services. In any case, in the experiments performed in Section 5 the LP solver used was GLPSOL.

Thus, we consider three easier assignment subproblems. This decomposition is based on the three decisions that a human planner faces when planning new services in order to minimize the cost and fulfill the constraints: i) What container is assigned to each truck?, ii) What pair truck-container is assigned to each service?, and iii) What truck is assigned to a container (in order to continue the service) when this one arrives to a destination port or rail-station? This decomposition provides a natural reduction of the problem complexity, more comprehensive than those based on stronger formulations or Lagrangian relaxations [5, 10, 9]. Therefore, the first subproblem solves the assignment of empty containers to trucks. The second subproblem solves the assignment of trucks with containers to services, using the assignments computed in the previous subproblem. These operations involve the provision of an empty truck and container to the service. The truck and container are used in the subsequent transportation until they arrive to the last delivery point or until they arrive to an intermodal node in the service. In intermodal transportation, additional trucks are needed in order to complete a service. These trucks pick-up the containers from the destination station/port and transport it to complete the service, or until they arrive to the next intermodal node. So in the third assignment subproblem, the method selects the best truck to pick-up the container from the destination station/port to continue the service, taking into account again the previous assignments. The algorithm is shown in Table 7.

```
solveAssignmentProblem(G, F, C, R , B, S, s)
;; Inputs: the network graph (G), the set of trucks (F), containers (C), trains (R), ships (B), services (S), and
the service to be completed (s)
;; Build the set of assignments of trucks to containers
setAssignmentsTrucksContainers=solveAssignTrucksContainers(G, F, C, R);
;; Select an empty truck and container to the service
setTrucks,container=solveAssignTrucksService(setAssignmentsTrucksContainers, R, B, S, s);
For each ($u_i$, $u_{i+1}$) ∈ $U_{tr}$
    ;; If there is a route of train or ship that starts in $m_k$, ends in $m_{k+1}$, $u_i$ is connected to $m_k$ and $u_{i+1}$ is connected to $m_{k+1}$
    If ($u_i, m_k$) and ($u_{i+1}, m_{k+1}$)
      ;; Add a truck to the set of trucks that pick-up the container from $m_{k+1}$ and continue the transportation route
      setTrucks=solveAssignTrucksIntermodalNodes(setTrucks,container, R, B, S, s);
End
Return setTrucks,container
```

Table 7: Top level algorithm to *solveAssignmentProblem*.

To tackle these three assignment subproblems we use two different approaches. The first one is a greedy approach that selects at each step the container and truck/s with the least estimated cost. In the second approach, the three assignment problems are solved using linear programming. These two approaches are described in detail in the next sections.

4.1.1. Solving the Assignment Problem using a Greedy Strategy

In this case, the truck/s and container to complete a service are selected following a greedy strategy. Next the different assignment processes following this strategy are formally described.

Assigning Containers to Trucks

First, the assignment problem of containers to trucks is solved. The algorithm iterates for each container and selects the truck with least cost. If a truck f is selected for a container c in the previous iteration, the truck f is removed from the list of available trucks in the following iterations. The cost of assigning a truck $f \in F$ to container $c \in C$ is denoted as c_{fc} and it is computed in the following way. The truck f is located in node $p_f \in P$ at time t_f. The container c is located in node $p_c \in P$ at time t_c. The truck has to travel from p_f to p_c to pick-up the container with a cost $cpu_{fc} = dist(p_f, p_c) \times e_f$, where $dist(p_f, p_c)$ is the distance between p_f and p_c, and e_f is the cost per kilometer when the truck travels empty. The distance is computed over the graph given that the *(x,y)* location of each node is known. In addition, the time t_a at which the truck f arrives to location p_c is estimated as $t_a = t_f + \frac{dist(p_f,p_c)}{s_f}$ where s_f is the mean speed of the truck. The waiting cost of a truck for a container is computed in Equation 1.

$$wc_{fc} = \begin{cases} (t_c - t_a) \times cs_f & \text{if } t_a < t_c, \\ 0 & \text{If } t_a \geq t_c. \end{cases} \tag{1}$$

where cs_f is the penalty cost when truck f is stopped at a location. Thus, Equation 2 computes the final cost of the assignment of truck f to container c.

$$c_{fc} = cpu_{fc} + wc_{fc} \tag{2}$$

The output of this assignment process is a set $W = (fc_1, fc_2, ..., fc_n)$ where each fc_i is a pair (f, c) of a truck and container. This set W is used in the next assignment process.

Assigning Trucks and Containers to Transportation Routes

Second, the algorithm iterates for each service and selects the pair $(f, c) \in W$ with least cost. If a pair (f, c) is selected for a service s in the previous iteration, it takes into account the final position and time of the pair *truck-container*, (f, c), used to complete the service s in the following iterations. The service $s \in S$ is given by the sequence of points $U_s = (u_1, u_2, ..., u_j)$. In the previous assignment process it already computed the cost to assign the truck $f \in F$ to container $c \in C$ located in node $p_c \in P$. Now, first it computes the cost to complete the first pick-up point u_1. Initially, the truck f with the container c travels to node u_1 with a cost $ct_{fc}^{u_1} = dist(p_c, u_1) \times e_f$.

The time t_{u_1} at which the truck f arrives to $u_1 \in U$ is estimated as $t_{u_1} = t_a + \frac{dist(p_c,u_1)}{s_f}$ where s_f is the mean speed of the truck and t_a was computed in the previous assignment process. The pick-up cost is computed following Equation 3

$$pu_{fc}^{u_1} = \begin{cases} (pt_{u_1} - t_{u_1}) \times cs_f & \text{if } t_{u_1} < pt_{u_1}, \\ (t_{u_1} - pt_{u_1}) \times pc_{u_1} & \text{if } t_{u_1} > pt_{u_1}, \\ 0 & \text{If } t_{u_1} = pt_{u_1}. \end{cases} \tag{3}$$

where pt_{u_1} indicates the time at which u_1 is available for the pick-up service, and pc_{u_1} is the penalty cost applied when the pick-up is delayed. Now, the truck and container with the shipment loaded in u_1 drive to the pick-up or delivery point u_2. If there is only a transportation mode between u_1 and u_2, and this is a road mode, the algorithm proceeds as follows. The truck f with the container c travels from node u_1 to node u_2 with cost $ct_{fc}^{u_2} = dist(u_1, u_2) \times l_f$, where l_f is the cost per kilometer when the truck travels loaded. The time t_{u_2} at which the truck f arrives to u_2 and the pick-up or delivery cost $pu_{fc}^{u_2}$ is computed in a similar way than previously.

If the road mode is the only transportation mode between points in $s \in S$, the algorithm proceeds in the same way with the remainder of delivery or pick-up points. In this case, Equation 4 computes the cost of the assignment of a truck with a container to a service.

$$c_{fc-s} = c_{fc} + (ct_{fc}^{u_1} + pu_{fc}^{u_1}) + \sum_{i=2}^{j} (ct_{fc}^{u_i} + pu_{fc}^{u_i}) \tag{4}$$

If a transportation mode between points u_k and u_{k+1} different to road mode (i.e. rail mode or ship mode) is possible, the algorithm proceeds differently. Suppose the rail mode or ship mode between u_k and u_{k+1} starts in $m_s \in M$, finishes in $m_e \in M$, u_k is connected with $m_s \in M$ by road, and $m_e \in M$ is connected with u_{k+1} by road. The truck and container with the shipment drive from u_k to node $m_s \in M$ with cost $ct_{fc}^{m_s} = dist(u_k, m_s) \times l_f$. At this point, the cost assignment process between truck (f, c) and s finishes because in m_e the container is transported by another truck in the assignment process, as described in the next section. Equation 5 computes the cost of the assignment of a truck with a container to a service when different transportation modes are presented in s.

$$c_{fc-s} = c_{fc} + (ct^{u_1}_{fc} + pu^{u_1}_{fc}) + \sum_{i=2}^{k}(ct^{u_i}_{fc} + pu^{u_i}_{fc}) + ct^{m_s}_{fc} \tag{5}$$

Assigning Trucks to Multimodal Nodes

In the case of intermodal nodes in the graph, the computation of the estimated cost of the assignment of trucks to intermodal nodes is different. Let $s \in S$ be the service that is being computed. Let $h \in F$ be the last truck that transported the container assigned to tr from u_k to m_s. u_k is the last pick-up or delivery point visited in s, and m_s is the start node for the last rail or ship transportation connected with u_k. Let $o \in C$ be the container assigned to s located in m_e, where m_e is the end node for transportation that starts in m_s. Let u_{k+1} be the next delivery or pick-up point to visit in s, where $m_e \in M$ is connected with u_{k+1}.

The time at which the container arrives to m_e can be estimated. The time t_{m_s} at which the truck h and container o arrived to m_s is estimated as $t_{m_s} = t_{u_k} + \frac{dist(u_k,m_s)}{s_h}$. In the station or port m_s, the container is disengaged from the truck. Then, the estimated time at which the container arrived to m_e, t_{m_e} is computed as $t_{m_e} = t_{m_s} + ut + D_{se} + dt$, where ut is the time spent to load the container on the train or ship, D_{se} is the time spent moving the container between the node m_s and m_e, and dt is the time spent to unload the container from the train or ship.

For each truck $f \in F$, the cost c_{fo-s} is recomputed using Equation 4 (if no more intermodal nodes are presented in s) or Equation 5 (if more intermodal nodes are presented in s). For costs where truck h is involved, it is taken into account that h is located in m_s at time t_{m_s}. For costs where container o is involved, it is taken into account that it is located in m_e at time t_{m_e}. In addition, it is taken into account that u_{k+1} is the next delivery or pick-up node to visit for service s. The new truck selected will be the truck with least c_{fo-s} computed.

If more intermodal nodes are found in s, a new process to assign a truck to intermodal node is started, and a new truck is selected.

4.1.2. Assignment Problem using Linear Programming

The assignment problem can be formulated as a linear programming problem. For each assignment problem described previously, a cost matrix is constructed and a linear programming model is designed in order to minimize the total assignment cost. In the next sections the different cost matrix and linear programming models are described in detail.

Assigning Containers to Trucks

The algorithm for solving the problem is based on the cost matrix M_{fc}. Cost matrix M_{fc} has a row for each truck $f \in F$ and a column for each container $c \in C$. Each position in the cost matrix is computed according to Equation 2. The linear programming model generated to solve the assignment problem is defined as follows:

$$Minimize \quad \sum_{i=1}^{n_f} \sum_{j=1}^{nc} c_{ij} x_{ij} \tag{6}$$

$$subject\ to \quad \sum_{j=1}^{n_c} x_{ij} = 1 \qquad (i = 1, 2, ..., n_f), \tag{7}$$

$$\sum_{i=1}^{n_f} x_{ij} = 1 \qquad (j = 1, 2, ..., n_c), \tag{8}$$

$$x_{ij} \in \{0, 1\} \qquad (i = 1, 2, ..., n_f)\,(j = 1, 2, ..., n_c) \tag{9}$$

where n_f is the number of trucks and nc is the number of containers. If there are less trucks than containers, $n_f < n_c$, $(n_c - n_f)$ containers do not have a truck assigned at the end of the assignment process. In this case, Equation 8 is modeled as $\sum_{i=1}^{n_f} x_{ij} = 1$ and Equation 7 as $\sum_{j=1}^{nc} x_{ij} \leq 1$. If $n_c < n_f$, $(n_f - n_c)$ trucks do not have a container assigned at the end of the process. In this case, Equation 7 is modeled as $\sum_{j=1}^{nc} x_{ij} = 1$ and Equation 8 as $\sum_{i=1}^{n_f} x_{ij} \leq 1$. If the number of trucks is equal to the number of containers, $n_f = n_c$, Equation 7 and Equation 8 are modeled as a strict equality. The GLPSOL solver [22] was used to solve this assignment problem. The return of this assignment process is a set $S = (fc_1, fc_2, ..., fc_n)$ where each fc_i is a pair (f, c) of a truck and container.

Assigning Trucks and Containers to Transportation Routes

The algorithm for solving the problem is based on the cost matrix M_{fc-s}, that has a row for each pair $(f, c) \in W$, and a column for each $s \in S$. Each position in the cost matrix is computed as Equation 4 (if no intermodal nodes are presented) or Equation 5 (if intermodal nodes are presented). In this case, the assignment model is defined as follows:

$$Minimize \quad \sum_{i=1}^{n} \sum_{j=1}^{n_s} c_{ij} x_{ij} \tag{10}$$

$$subject\ to \quad \sum_{j=1}^{n_s} x_{ij} = 1 \qquad (i = 1, 2, ..., n), \tag{11}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1, 2, ..., n_s), \tag{12}$$

$$x_{ij} \in \{0, 1\} \qquad (i = 1, 2, ..., n)\ (j = 1, 2, ..., n_s) \tag{13}$$

where n is the number of pairs (f, c) of trucks and containers computed previously, and n_s is the number of services. If there are less pairs (f, c) than services, $n < n_s$, $(n_s - n)$ services do not have a truck and container assigned at the end of the assignment process. In this case, the Equation 12 is modeled as $\sum_{i=1}^{n} x_{ij} = 1$ and Equation 11 as $\sum_{j=1}^{n_s} x_{ij} \leq 1$. If $n_s < n$, $(n - n_s)$ pairs of trucks and containers do not have a service assigned at the end of the process. In this case, Equation 11 is modeled as $\sum_{j=1}^{n_s} x_{ij} = 1$ and Equation 12 as $\sum_{i=1}^{n} x_{ij} \leq 1$. If the number of pairs of trucks and containers is equal to the number of services, $n_s = n$, Equation 11 and Equation 12 are modeled as a strict equality.

Assigning Trucks to Intermodal Nodes

The costs are computed in similarly as when the greedy strategy was used. In this case, the costs of the matrix cost M_{fc-s} corresponding to s are recomputed (service that is being computed), c_{hc-s} where $h \in F$ is the last truck used in s located in m_s, and c_{fo-s} where $o \in C$ is the container used in s located in m_e (Figure 3).

$$M_{fc-tr} = \begin{pmatrix} c_{fc_1-tr_1} & \cdots & \boxed{c_{fo-tr}} & \cdots & c_{fc_1-tr_{nr}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \boxed{c_{hc-tr_1}} & \cdots & \boxed{c_{ho-tr}} & \cdots & \boxed{c_{hc-tr_{nr}}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ c_{fc_{na}-tr_1} & \cdots & \boxed{c_{fo-tr}} & \cdots & c_{fc_{na}-tr_{nr}} \end{pmatrix}$$

Figure 3: *Cost matrix M_{fc-tr} recomputed.*

The costs c_{fo-s} and c_{hc-s} are recomputed using Equation 4 (if no more intermodal nodes are present) or Equation 5 (if more intermodal nodes are present). For costs where truck h is involved, it is taken into account that h is located in m_s at time p_{m_s}. For costs where container o is involved, it is taken into account that it is located in m_e at time p_{m_e}. For costs where s is involved, it is taken into account that u_{k+1} is the next delivery or pick-up node to visit. The assignment model is built in the same way that in the previous section. If more intermodal nodes are found in s, a new assignment truck to intermodal node process is started, and a new truck is selected.

4.2. Planning Problem

Once the truck/s and container are selected for the current service, a planning problem is built in order to find the best transportation modes to complete it.

The most efficient current domain-independent planners are based on heuristic search. Heuristic planners [24] are composed of a heuristic search algorithm guided by a domain-independent heuristic function. We use the SAYPHI [21] planner. The high level algorithm to solve the transportation planning problem is depicted in Table 8. The algorithm receives as input the truck/s and container selected in the previous phase to complete the service. First the algorithm selects the trains and ships that can be used to complete the service. Later the problem is constructed taking into account the trains, ships and the truck/s and container selected to complete the service. The problem is modeled using the PDDL [18] language which is a standard planning domain and problem description language. Lastly TIMIPLAN executes the planner, which searches in the space of combinations and decides about the best transportation modes in order to complete the service.

4.2.1. Domain Description

In this section, the planning domain will be described using PDDL notation. PDDL allows the users to represent the predicates and functions that can be used to describe the states, what actions can be executed (transportation actions in our case), and what the conditions and effects of actions are.

Predicates are the properties of the objects interesting for this application. In our case, several predicates are used, but here only the most relevant ones are described:

- *(at ?goods - goods ?node- node)*: goods *?goods* is at node *?node*. The node *?node* corresponds to a pick-up point.
- *(in ?goods - goods ?c - container)*: goods *?goods* is inside container *?c*.

```
**solvePlanningProblem**(selectedTrucks, selectedContainer, R, B, s)
;; Inputs: the truck/s selected to solve the service (selectedTrucks), the selectedContainer (selectedContainer), trains (R), ships (B), and the transportation
;; service to be completed (s). It selects the trains and ships present in the service
For each $(u_i, u_{i+1}) \in U_s$
    For each $r \in R$
    ;; If there is a route of the train that starts in $m^r_k$, ends in $m^r_{k+1}$, $u_i$ is connected to $m^r_k$ and $u_{i+1}$ is connected to $m^r_{k+1}$
    If $dist(u_i, m^r_k) + dist(u_{i+1}, m^r_{k+1}) < dist(u_i, u_{i+1})$
        ;; Add the train to the transportation route
        setTrains=addTrain(r);
    For each $b \in B$
    ;; If there is a route of the ship that starts in $m^b_k$, ends in $m^b_{k+1}$, $u_i$ is connected to $m^b_k$ and $u_{i+1}$ is connected to $m^b_{k+1}$
    If $dist(u_i, m^b_k) + dist(u_{i+1}, m^b_{k+1}) < dist(u_i, u_{i+1})$
        ;; Add the ship to the transportation route
        setShips=addShip(b);
;; Write the pddl problem
pddlProblem=writePDDLProblem(selectedTrucks, selectedContainer, setTrains, setShips, s);
;; Plan the pddl problem
plan=callSAYPHI(pddlProblem);
End
Return **plan**
```

Table 8: Top level algorithm to *solvePlanningProblem*.

- *(road ?node1 - node ?node2 - node)*: nodes $?node1$ and $?node2$ are connected by a road.

Numeric functions are frequently useful in real domains. They allow the user handling numerical values in PDDL. The functions can be used in actions preconditions or effects and their initial value is given in the problem file. In our case, these are some relevant functions:

- *(distance ?node1 - node ?node2 - node)*: distance in kilometers between node $?node1$ and node $?node2$.
- *(pickup-time ?goods - goods ?node - node)*: time at which the corresponding node $?node$ is available for the pick-up service.
- *(penalization-pickup-delayed ?goods - goods ?node - node)*: penalty cost per hour applied when the pick-up is delayed.

Actions change the state of the world. An action is defined by a list of parameters, a list of conditions and a list of effects. As an example, Figure 4 shows the description of action *LOAD-TRAIN*. An action is applicable if its preconditions are true within the current world state. In this case, the container and train should be in the same location, and there should be time to load the container inside the train before the departure time. This planning action also describes the changes (effects) applied to the world state after its execution. In this case, the container is inside the train in the destination station. As seen, actions are declaratively defined, so it is relatively easy for end users to understand what they perform or even modify them when problem constraints change.

```
(:action load-train
 :parameters (?t - train ?c - container ?station1 ?station2 - localization)
 :precondition (and (in-town ?c ?station1)
                    (in-town ?t ?station1)
                    (<= (+ (time-container ?o) (time-load-train ?t ?station1))
                        (departure-time-train ?t ?station1 ?station2)))
 :effect (and (not (in-town ?c ?station1))
              (in-train ?c ?t)
              (assign (time-container ?c)
                      (departure-time-train ?t ?station1 ?station2))))
```

Figure 4: Action *LOAD-TRAIN* described in PDDL.

4.2.2. Problem Description

A problem specifies an initial state, and a set of goals to achieve. The initial state is specified as a list of literals assumed to be true in the initial state. A literal is an instantiated predicate (i.e. *literal = (predicate argument-value*)*). As an example, *(road n5 n2)* is a literal where *road* is the predicate, and *n5* and *n2* are its instantiated arguments. The goals specify the end condition of the search problem. The solution to a problem is an ordered set of actions (usually a sequence) that, if applied to the initial state, transforms it into a state where all goals are true. In our case, this initial state can be expressed using again PDDL as in Figure 5. In this case the initial state describes the resources used in the problem (trucks, containers, ships, trains), the roads, and the different pick-up and delivery points.

The goals in the problem are to reach the state where both pick-up and delivery points have been served. This goal can be defined as shown in Figure 6. In this case, the goal is to pick-up goods from node *n1* and deliver it to node *n6*.

```
(:init
  ;; Trucks
  (in-town truck1 n1)
  (= (cost-per-kilometer-empty truck1) 0.5)
  ...
  ;; Roads
  (road n5 n2)
  (= (distance n5 n2) 347.0)
  ...
  ;; Containers
  (in-town container1 n1)
  (empty container1)
  ...
  ;; Goods
  (in-town goods1 n5)
  (= (pickup-time goods1 n5) 100)
  (= (penalization-pickup-delayed goods1 n5) 0.1)
  ...
  ;; Ships
  (in-town ship1 n0)
  (ship-trip n0 n3)
  (= (departure-time-ship ship1 n0 n3) 200)
  ...
```

Figure 5: Example of an initial state of a problem described in PDDL.

```
(:goal (and (picked goods1 n1)
            (delivered goods1 n6)))
```

Figure 6: Goals to achieve.

5. Empirical Evaluation of TIMIPLAN

To evaluate the TIMIPLAN algorithm, we use a set of representative problems, based on real data gathered by the company. The problems were generated using ship routes and pick-up and delivery points gathered from real problems. There has been a positive qualitative evaluation from users. However, direct comparison against the current solutions adopted by Acciona is not possible at this point. First, their databases, handled by humans, have many inconsistencies (bad written addresses, same company with different names, ...), and it is necessary to estimate values which are not provided in the databases (time spent for each pick-up or delivery service, driving hours/rest periods for the drivers, ...). Second, our application was developed by the central offices to address the loss of solutions quality due to the decentralized planning (resources) among the branches, as it is currently done. Thus, currently there is no human solving a 300 services assignment (each branch considers instead a smaller problem). There are no plans of such size to compare against, and also coming up with a plan for 300 services is a hard task for humans. However, they already examined the generated plans and considered them to be in the range they would generate.

Two versions of the TIMIPLAN algorithm are used to solve problems of different sizes.

Both versions differ on how they perform the first step of the algorithm: the assignment of truck/s and container to services. In the first one, a greedy strategy is used to select the *best* container and truck/s to complete the service as explained in Section 4.1.1. In the second one, an assignment model is constructed as a linear programming problem in order to find the best assignment of the containers and trucks to the services as explained in Section 4.1.2. The greedy approach provides a reasonable simple solution to compare the LP approach and also a baseline reference to determine how much improvement (in terms of cost) is produced when using the LP approach, although obviously the LP approach will require more computing time. All the experiments we report here were conducted on a 2,4 GHz quadcore processor with 4 GB RAM, running Linux.

Table 9: Types of problems used in the experimentation.

Problem Type	**Transportation Routes**	**Nodes**	**Edges**	**Trucks**	**Containers**	**Ship Segments**	**Train Segments**
1	75	150	21758	75	75	60	5
2	100	200	39008	100	100	70	10
3	125	250	61258	125	125	80	15
4	150	300	88508	150	150	90	20
5	175	350	120758	175	175	100	25
6	200	400	158008	200	200	110	30
7	225	450	200258	225	225	120	35
8	250	500	247508	250	250	130	40
9	275	550	299758	275	275	140	45
10	300	600	357008	300	300	150	50

In order to evaluate the TIMIPLAN algorithm we study the time and cost requirements of our algorithm defining ten types of problems in ascending order of size. Each problem has a linear increase in the number of services (between 75 and 300), nodes (between 150 and 600), trucks (between 75 and 300), containers (between 75 and 300), ships segments (between 60 and 150) and train segments (between 5 and 50). For each problem size, ten different problems are solved in order to obtain representative mean values and standard deviations. Given that the company started mainly as a ship transportation company, all problems contain locations on islands, so it is necessary to use ships. The number of elements for each problem size is shown in Table 9.

Figure 7 shows graphically the comparison of mean times to solve problems of the different type proposed using the two different versions of the TIMIPLAN algorithm.

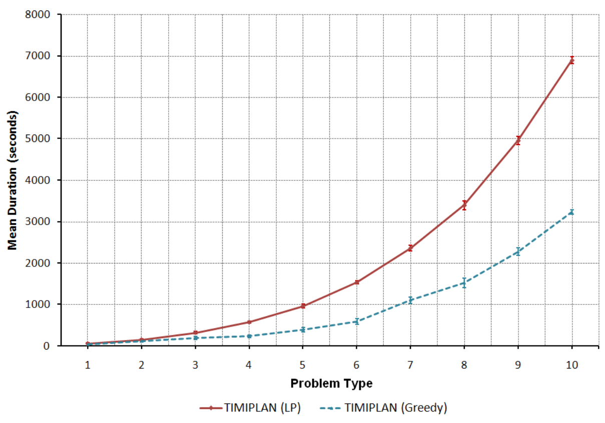

Figure 7: Mean solving time and standard deviations.

The solid red line shows the mean times and standard deviations spent by the TIMIPLAN algorithm when using linear programming. The dashed blue line shows the mean times and standard deviations needed by the TIMIPLAN algorithm when it uses the greedy strategy. In the case of TIMIPLAN *(LP)*, the mean time grows from 65.15 seconds (the mean time TIMIPLAN takes to solve the simplest problem) to 6896.09 seconds (mean time it takes to solve the most complex problem). Given that it performs a more complex assignment of trucks and containers to routes, this version of the TIMIPLAN algorithm needs more time to solve problems than the greedy approach. In the latter, the mean time ranges from 38.05 seconds to 3236.94 seconds. In both cases, time grows exponentially, with the curve of TIMIPLAN *(Greedy)* being less steep.

The sensitivity of both solutions to the cost schema defined by the company is analyzed with three different cost configurations, ordered in decreasing order of cost. Each configuration uses a cost per kilometer e_f when truck f travels without any container or the container is empty, and a cost per kilometer l_f when truck f travels with a loaded container. Acciona Transmediterránea Cargo tries to minimize the number of kilometers where each truck travels unloaded, so in all cost configurations $e_f > l_f$. In addition, each cost configuration uses a cost per hour cs_f when truck f is stopped in a location, a penalty cost pc_v per hour applied when a delivery is delayed, and a penalty cost pc_u per hour applied when a pick-up is delayed. Configuration 1 has higher costs than configuration 2, and configuration 3 is the cheapest one. Again, these cost settings are based on real data gathered from the company.

Figure 8 shows the comparison of quality (cost) of solutions of the same problems solved previously. Costs are expressed in millions of euros. In this case, the solid red line labeled as TIMIPLAN *(LP)* shows the mean costs and standard deviations obtained by the TIMIPLAN algorithm when it uses linear programming. The dashed blue line TIMIPLAN *(Greedy)* shows the same information when using the greedy strategy. Figure 8 shows the results from the three different cost configurations considered. The difference in costs between the *LP* and *Greedy* versions depends on the cost configuration. So, taking into account the cost configurations established at the beginning, the solution for the cost configuration 1 for the problem type 10 is approximately 1.53 millions of euros cheaper than the solution obtained by the greedy approach; it is approximately 0.91 millions of euros cheaper for the cost configuration 2; and it is 0.37 millions of euros cheaper for the cost configuration 3. Given these results, when higher cost penalties are used, the differences in the total cost (quality) between the *LP* and *Greedy* plans are more significant. Given that the main driver of Acciona is the reduction in cost, once the rest of constraints are fulfilled (specially as regards to the time limit imposed), the combined approach of OR and AI techniques is empirically proven to be better than a reasonable simpler *Greedy* approach.

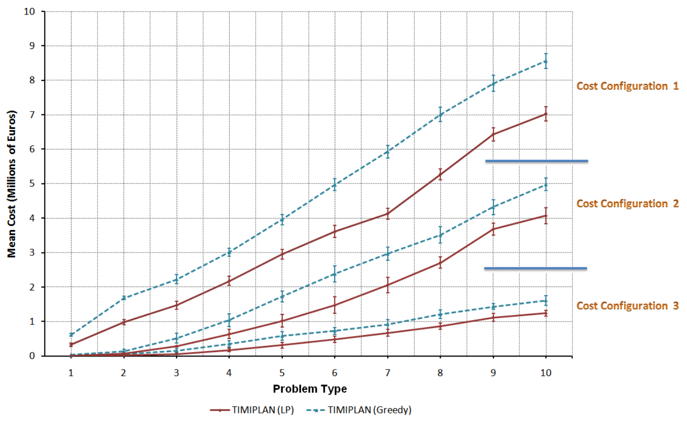

Figure 8: Mean costs (in million euros) and standard deviations for the three proposed cost configurations.

Although it is impossible to directly compare the TIMIPLAN solutions with the solutions adopted by Acciona for the reasons listed above, it can be stated with fully confidence what truck/s and container the Acciona planner selected for each particular service. This information can be supplied to the first step of TIMIPLAN. Thus, we can compare the performance of TIMIPLAN when it uses in the first step the assignment of truck/s and containers to services proposed by Acciona, and when it uses in the first step the assignment proposed by TIMIPLAN (Section 4.1.2). After the first step, the second step of TIMIPLAN (Section 4.2) is executed normally. Figure 9 shows the cost per day obtained by TIMIPLAN working in these two different ways for five real and consecutive daily problems extracted from the databases of Acciona (removing by hand the inconsistencies as the bad written addresses for these problems, which requires an enormous effort). In both cases, it used the cost configuration 3.

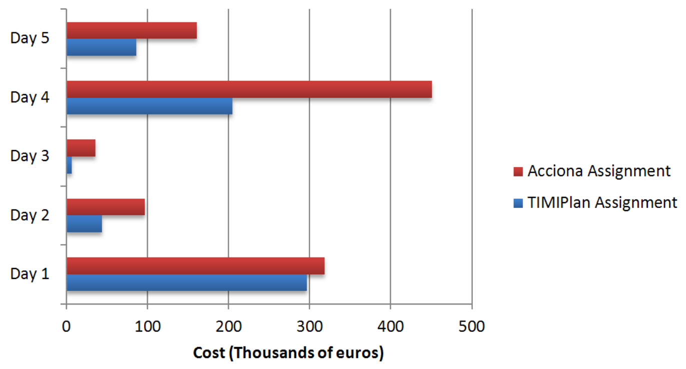

Figure 9: The costs per day obtained using the assignment of truck/s and containers to services proposed by TIMIPLAN and by the Acciona planner for five consecutive daily problems.

Figure 9 shows that when TIMIPLAN uses the assignment of truck/s and containers to services it proposes the cost per day is significantly reduced with respect to the one coming from Acciona's solution. This reduction is due to two main reasons: i) TIMIPLAN performs a centralized planning while Acciona human planners perform decentralized planning (resources) among the different branches. Therefore, the use of all available resources of the company for each problem is advantageous with respect to only using the resources of a single branch. ii) Linear programming is able to find the best assignment of truck/s and containers to services for all the available resources overcoming the logical limitations of a human planner. It is impossible for a human planner to address the total assignment problem of 300 $trucks \times 300$ $containers \times 300$ $routes$, and, additionally, take into account costs, ship/train segments,...

6. Conclusions

This paper has introduced TIMIPLAN, that provides a very good tradeoff between time to plan and quality of the solutions for the intermodal transportation problem from one of the largest spanish companies using intermodal transportation, Acciona Transmediterránea Cargo. However, clearly, the bottleneck in this problem is the combinatorial explosion which makes obtaining optimal solutions impossible in the time limit imposed by the company using only classical planning or only OR techniques. Given the size of the problem, existing domain-independent planners cannot solve it in a reasonable time. In a similar way, linear programming experiences a combinatorial explosion and some type of decomposition is required [5, 10, 9]. Instead, this paper presents a novel way to decompose the intermodal transportation problem, using the combination of OR and AP techniques. Our approach decomposes the problem into two different subproblems. In the first one, the assignment cost of resources (trucks and containers) to tasks (services) is computed. In the second one, each task is formulated as a planning problem where different actions are taken (selecting the best transportation modes) to achieve the goals (the different pick-up and delivery requests for each service) taking into account the resources (trucks and containers) selected in the previous phase. LP has shown to be effective to optimally solve the different assignment subproblems, and automated planning has solved successfully the selection of the best transportation modes. This novel way of combining linear programming and planning has allowed us to balance the total cost (quality) obtained, the time required to compute a solution and the time to model the different optimization problems.

Another key issue in relation to solving real world problems consists on the difficulty of modelling. In our case, we could have opted to spend much more time on modelling the whole problem as a LP or CSP problem, or to spend much more time on coming up with a solution to the grounding explosion problem for current planners. It might be possible that following those alternatives would have generated a better solution in terms of quality and/or time. We believe that our current solution is a viable solution that has also minimized the modelling time (programming effort) providing a good solution to the task. As a side effect, we have also separated the modelling difficulties, so that we deal with the best solution in terms of the multiple criteria problem of <*modelling time, quality of solution, time to solve*>. Empirical evaluation shows that this combination of techniques finds valid (but suboptimal) plans to complete all the daily services of the company within the imposed time limit, and outperforming the quality of plans obtained by a reasonable simpler approach. TIMIPLAN, provides several improvements to the company operations.

In order to finally deploy TIMIPLAN we have to pre-process the databases (or include some kind of robust input parsing). As future work, we consider combining LP and automated planning in a different way to find better solutions (lower cost) in less time.

Acknowledgements

This work has been partially supported by the CDTI (Spanish Ministry of Research) within the project CENIT2007-2002 (TIMI), the Spanish MICINN project TIN2008-06701-C03-03 and the UC3M-CAM project CCG08-UC3M/TIC-4141. We would like to thank the people from Acciona.

References

[1] T. G. Crainic, K. H. Kim, Intermodal transportation, Handbook in ORMS 14 (2007) 467–537.

[2] A. Caris, C. Macharis, G. K. Janssens, Planning problems in intermodal freight transport: accomplishments and prospects, Transportation Planning and Technology 31 (2008) 277–302.

[3] W. B. Powell, A. Marar, J. Gelfand, S. Bowers, Implementing real-time optimization models: A case application from the motor carrier industry, Oper. Res. 50 (2002) 571–581.

[4] S. Ropke, D. Pisinger, An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows, Transportation Science 40 (2006) 455–472.

[5] L. Coslovich, R. Pesenti, W. Ukovich, Minimizing fleet operating costs for a container transportation company, European Journal of Operational Research 171 (2006) 776–786.

[6] M. A. Salido, F. Barber, Mathematical Solutions for Solving Periodic Railway Transportation, Mathematical Problems in Engineering (2009).

[7] A. Imai, K. Shintani, S. Papadimitriou, Multi-port versus Hub-and-Spoke Port calls by Containerships, Transportation Research Part E: Logistics and Transportation Review (2009).

[8] R. Sprenger, L. Monch, A methodology to solve large-scale cooperative transportation planning problems, European Journal of Operational Research (2012).

[9] T.-S. Chang, Best routes selection in international intermodal networks, Comput. Oper. Res. 35 (2008) 2877–2891.

[10] A. Imai, E. Nishimura, J. Current, A lagrangian relaxation-based heuristic for the vehicle routing with full container load, European Journal of Operational Research 176 (2007) 87–105.

[11] S. Bock, Real-time control of freight forwarder transportation networks by integrating multimodal transport chains, European Journal of Operational Research 200 (2010) 733–746.

[12] M. Verma, V. Verter, N. Zufferey, A bi-objective model for planning and managing rail-truck intermodal transportation of hazardous materials, Transportation Research Part E: Logistics and Transportation Review (2011).

[13] J. A. Gromicho, E. Oudshoorn, G. Post, Generating price-effective intermodal routes, Statistica Neerlandica 65 (2011) 432–445.

[14] T. Bylander, A Linear Programming Heuristic for Optimal Planning, in: In AAAI97/IAAI-97 Proceedings, pp. 694–699.

[15] H. Kautz, J. P. Walser, State-space Planning by Integer Optimization, in: In Proceedings of the Sixteenth National Conference on Artificial Intelligence, AAAI Press, 1999, pp. 526–533.

[16] S. Fernndez, D. Borrajo, Solving clustered oversubscription problems for planning e-courses, in: Proceedings of SPARK, Scheduling and Planning Applications woRKshop, ICAPS'09, Thessaloniki (Greece).

[17] D. Nau, M. Ghallab, P. Traverso, Automated Planning: Theory & Practice, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[18] M. Fox, D. Long, PDDL2.1: An Extension to PDDL for Expressing Temporal Planning domains, J. Artif. Intell. Res. (JAIR) 20 (2003) 61–124.

[19] Y. Chen, C. wei Hsu, B. W. Wah, Sgplan: Subgoal partitioning and resolution in planning, in: In Edelkamp, pp. 30–32.

[20] A. Gerevini, A. Saetti, I. Serina, Planning through stochastic local search and temporal action graphs in lpg, J. Artif. Int. Res. 20 (2003) 239–290.

[21] T. La Rosa, A. García Olaya, D. Borrajo, Using Cases Utility for Heuristic Planning Improvement, in: ICCBR '07: Proceedings of the 7th international conference on Case-Based Reasoning, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 137–148.

[22] A. Makhorin, GLPK Linear Programming Kit: Implementation of the Revised Simplex Method, GLPK documentation, Moscow Aviation Institute, Moscow, Russia, 2001.

[23] N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, G. Tack, Minizinc: Towards a standard cp modelling language, in: In: Proc. of 13th International Conference on Principles and Practice of Constraint Programming, Springer, 2007, pp. 529–543.

[24] J. Hoffmann, FF: The Fast-Forward Planning System, AI Magazine 22 (2001) 57–62.