Truck Scheduling at Container Terminals

Dylan Camus[1], Ryan Do[2], Fan Jia[3], Matheus Mathelhaes[4], and Sugumar Prabhakan[5]

[1] 1007441859
[2] 1001254117
[3] 1001297198
[4] 1002118291
[5] 994126815

University of Toronto, 27 King's College Circle,
Toronto, ON M5S 1A1 Canada

Abstract. Trucks are very popular transport method in most terminal ports around the world to move containers from different locations. This problem considers a variant of the truck scheduling problem in container terminals that minimizes the extra costs involved with unloading product from containers for transport further down the supply chain. Mixed Integer Programming (MIP) and Constraint Programming (CP) formulations are proposed and the results compared on 10 different problem instances. The CP formulation performs better on all instances compare to the MIP. However, other methods such as a heuristics need to be explored before solving actual instance sizes of the real-life problem.

Keywords: Truck Scheduling, Container Terminal, Mixed Integer Programming, Constraint Programming.

1 Introduction

Container ports are an integral part of international supply chains, used primarily to transship goods from international shipping carriers to domestic transport. Ocean containers are discharged from container ships at container terminals (seaport), and sent to a transloading facility, where product is transloaded or cross-docked into a domestic container truck or rail for delivery to distribution centers further down the supply chain. There are several soft deadlines on containers once arrived at the container terminals that need to be met, otherwise extra costs will be incurred.

The problem is inspired by a real decision-making scenario that a company supply chain analyst is currently facing at one of the container terminals in Halifax, NS and is defined in Section 2. In Section 3, the problem is formulated as a Mixed Integer Program using inspiration from the disjunctive formulation of the Job-Shop Scheduling Problem. In Section 4, a Constraint Programming formulation is proposed using a cumulative resource formulation. In Section 5, the 10 chosen problem instances are detailed. In Section 6 and 7, the performances of both models are presented and discussed.

Lastly, Section 8 includes conclusions about the project and applications of formulations are presented.

2 Problem Definition

This focus of this project is specifically on the sequencing of ocean containers being sent from the container terminal after being discharged from a container ship to the transloading facility. The transloading facility transfers goods from the ocean containers to domestic containers owned by a distributor in a process known as cross-docking, and then loads the container on rail or trucks towards its next destination. To avoid financial penalties at the terminal and account for the varying processing times at the transloading facility and the variance of ship arrivals; containers can be sent to a third external location called the stack for temporary storage at a fixed cost of entry per container but with a much lower daily storage cost, as seen below in Fig. 1.

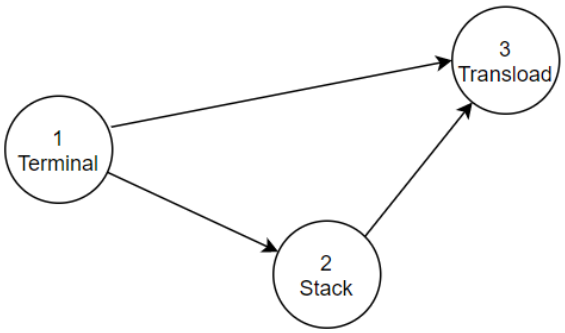

Fig. 1. Directed graph showing movement of containers between locations. Chassis may move bidirectionally between any two nodes.

Containers are moved between locations using a container bed called a chassis, which is attached to the back of a truck. An external crane is used to pick up a container and place it on chassis for transport. When a container is transported to the stack, it is then removed from the chassis, and placed in a container pile, and so the chassis is free to process other containers. However, when a container is being processed at the transloading facility, the chassis is required to sit at the dock until the ocean container is empty because the crane at the transloading facility is unable to lift an ocean container full of goods due to weight. Therefore, the chassis must remain at the transloading facility until the container is fully processed. This requirement is also known as non-preemptive scheduling, as an entity in process will not be interrupted until fully complete. The time associated with processing a container typically takes 1-5 days, with an average of 2 days, depending on the load of the container.

When the transloading process is finished, the chassis (with the initial ocean container) are both picked back up by the nearest truck and the ocean container is returned to the carrier at the terminal. Between successive transportation jobs, chassis need to

travel between their current container's drop off point to the next container's pickup point once their task is completed.

The objective of this project is to evaluate the effectiveness of two different scheduling methods to determine the optimal schedule of container transport with respect to minimizing total extra costs and a limited number of available chassis. Containers may either be transported directly from the terminal to the transload facility or be sent to the stack for a certain amount of time, and then onto the transloading facility.

As stated previously, depending on the international shipping carrier, ocean containers have a few free storage days at the terminal after being discharged from a ship known as demurrage costs, which is calculated as the tardiness at Terminal. After this free period, there is a cost incurred daily (E.g. For days 1-5, $0 is incurred. If the container remains past day 6+: $100/container/day). Like demurrage costs: depending on the carrier, ocean containers have a few weeks to be emptied after being discharged from the ship. After this free period, there is a detention cost incurred daily until the container is returned to the terminal (E.g. $200/container/day). At the external storage facility Stack, a one-time fixed cost is charged for loading and unloading containers into the yard, with a cheap cost incurred daily by container (E.g. Container x spends 4 days in the stack: $300 entrance fee + [4 days * $15/day/container] = $360 total)

This scheduling problem can be thought as similar in some respects to the Unrelated Parallel Machine Scheduling Problem (UPM) and the Vehicle Routing Problem with Pickup and Deliveries. Alongside the regular UPM constraints of job release dates and non-overlaps on the chassis resource; there are the additional considerations of assigning containers to travel legs between locations, the sequence-dependent chassis travel time between jobs and the priority of the container must also be considered. Each container also has a priority level assigned by the company, to prevent stock-outs of certain products further up the supply chain. These containers can have a priority of low, medium, or high priority, as defined by the intended distributor. Thus, if a container has high priority, it would be preferable to avoid sending the container to the stack for temporary storage. Problem instances will consider a mix of priorities, such as a scenario of a batch of high priority containers arriving on the same ship. A cost penalty is assigned to each container at the time of departure from terminal which is scaled by the priority and container arrival time, to process high priorities first.

We consider containers as released when they are discharged from a container ship at the container terminal. Ships have an interarrival time typically between 1-3 days and with a random quantity of ocean containers between 20-80 containers as cargo (E.g. Ship 1 arrives on day 0 with 80 containers, Ship 2 arrives on day 3 with 54 containers, etc.). We assume containers from a ship are all discharged on the same day.

3 Mixed Integer Programming Formulation

3.1 MIP Model

Sets:

C	$Set\ of\ chassis, c \in C$
K	$Set\ of\ containers, k \in K$
J	$Set\ of\ carriers, j \in J$
$L = \{1, 2, 3\}$	$Set\ of\ travel\ legs, l \in L.\ 1: terminal\ to\ transload, process, and\ back\ to\ terminal,$ $2: terminal\ to\ stack,$ $3: stack\ to\ transload, process, and\ back\ to\ terminal$

Parameters:

Φ_l	$Time\ duration\ of\ leg\ l\ [days]\ l \epsilon L$
$D_{ll'}$	$Delay\ required\ for\ a\ chassis\ [days] to\ start\ leg\ l'\ after\ completing\ l\ (includes\ travel)$
M	$some\ large\ number$
R_k	$Release\ time\ of\ container\ k\ [days]$
Γ_j	$Demurrage\ cost\ at\ terminal\ per\ unit\ time\ for\ carrier\ j\ after\ free\ period\ [^{\$}/_{container * days}]$
Γ''_j	$Free\ period\ before\ demurrage\ cost\ for\ carrier\ j\ at\ terminal\ [days]$
S	$fixed\ cost\ at\ stack\ [^{\$}/_{container}]$
S'	$unit\ cost\ at\ stack\ per\ unit\ time\ [^{\$}/_{container * days}]$
δ''_j	$Free\ period\ before\ detention\ cost\ for\ some\ container\ of\ carrier\ j\ [days]$
δ_j	$Detention\ cost\ per\ unit\ time\ for\ some\ container\ of\ carrier\ j\ [^{\$}/_{container * days}]$
Z_{jk}	$1\ if\ container\ k\ belongs\ to\ carrier\ j$
ρ_k	$Priority\ factor\ of\ container\ k, higher\ value\ is\ higher\ priority$

Variables:

x_{klc}	$1\ if\ container\ k\ is\ transported\ across\ leg\ l\ using\ chassis\ c, 0\ otherwise$
s_{klc}	$Start\ time\ of\ transporting\ container\ k\ across\ leg\ l\ using\ chassis\ c.$
$z_{klk'l'c}$	$1\ if\ the\ transport\ job\ of\ container\ k' for\ chassis\ c\ along\ leg\ l' is\ scheduled\ after\ the\ transport\ job\ of\ container\ k\ for\ chassis\ c\ along\ leg\ l.$

$min \sum_{k\epsilon K} \sum_{c\epsilon C} \sum_{l\epsilon\{1,2\}} \sum_{j\epsilon J} \Gamma_j x_{klc} Z_{jk} * \max\{(s_{klc} - R_k - \Gamma_j^{"}), 0\}$	Demurrage cost
$+ \sum_{k\epsilon K} \sum_{c\epsilon C} \sum_{l\epsilon\{1,3\}} \sum_{j\epsilon J} \delta_j x_{klc} Z_{jk} * \max\{(s_{klc} + \phi_{kl} - (R_k + \delta^{"}_j)), 0\}$	Detention cost
$+ \sum_{k\epsilon K} \sum_{c\epsilon C} x_{k2c} (S + S'(s_{k3c} - (s_{k2c} + \phi_{k2})))$	Stack cost
$+ \sum_{k\epsilon K} \sum_{c\epsilon C} \sum_{l\epsilon\{1,2\}} \sum_{j\epsilon J} \rho_k x_{klc} (s_{klc} - R_k)$	Priority cost

#	***subject to***:		***description***
1	$M(1 - x_{klc}) + M(1 - x_{k'l'c}) + (s_{klc} - s_{k'l'c}) \geq \Phi_{k'l'} + D_{l'l} - M * z_{klk'l'c}$	$for\ all\ c \epsilon C,$ $for\ all\ k, k' \epsilon K$ $for\ all\ l, l' \epsilon L$ $s.t.$ $NOT\ AND$ $(k = k', l = l')$ $AND\ (k \leq k')$	For each chassis, jobs can't overlap, and chassis may have to travel between jobs. For klc scheduled after k'l'c.
1	$M(1 - x_{klc}) + M(1 - x_{k'l'c}) + (s_{k'l'c} - s_{klc}) \geq \Phi_{kl} + D_{ll'} - M * (1 - z_{klk'l'c})$	$for\ all\ c \epsilon C,$ $for\ all\ k, k' \epsilon K$ $for\ all\ l, l' \epsilon L$ $s.t.$ $NOT\ AND$ $(k = k', l = l')$ $AND\ (k \leq k')$	For each chassis, jobs can't overlap, and chassis may have to travel between jobs. For k'l'c scheduled after klc.
2	$s_{klc} \geq R_k x_{klc}$	$for\ all\ k \epsilon K,$ $for\ all\ l \epsilon L,$ $for\ all\ c \epsilon C$	Release time constraint
3	$\sum_{c\epsilon C} x_{k2c} = \sum_{c\epsilon C} x_{k3c}$	$for\ all\ k \epsilon K$	Containers going through leg 2 must also go through leg 3
4	$\sum_{c\epsilon C} x_{k2c} + \sum_{c\epsilon C} x_{k1c} = 1$	$for\ all\ k \epsilon K$	Containers go through leg 1 OR leg 2/3
5	$M \sum_{c\epsilon C} x_{k1c} + \sum_{c\epsilon C} s_{k3c} \geq \sum_{c\epsilon C} s_{k2c} + \Phi_{k2}$	$for\ all\ k \epsilon K$	leg 2 to 3 precedence constraint

6	$Mx_{klc} \geq s_{klc}$	$for\ all\ k \in K,$ $for\ all\ l \in L,$ $for\ all\ c \in C$	When x is 0, s is 0. Necessary for the precedence constraint
7	$s_{klc} \geq 0$	$for\ all\ k \in K$ $for\ all\ l \in L$ $for\ all\ c \in C$	Domain of s
8	$x_{klc} = [0,1]$	$for\ all\ k \in K$ $for\ all\ l \in L$ $for\ all\ c \in C$	Domain of x

3.2 MIP Formulation Rationale

The MIP model described in 3.1 aims to solve a scheduling-assignment hybrid problem that is to determine optimal start times for each container transport job while simultaneously assigning containers to chassis (resources) and determining which route to take for each chassis-container assignment (leg 1 vs. leg 2/3). The objective function for the optimization is to minimize demurrage costs (time containers spend at terminal), detention costs (time containers spend before being emptied and sent back to terminal), stack cost (time containers spend at stack), and priority cost (time containers spend at terminal scaled by priority factor). Demurrage costs and detention costs are incurred after a free period ($\Gamma_j^{"}$ and $\delta"_j$ where j is in the set of carriers) respectively. The stack cost includes two components: a fixed cost of going to the stack, and a smaller periodic cost relative to the demurrage and detention periodic costs.

To re-iterate, at the transload facility chassis must stay with containers before returning them to the carrier at the terminal. As such, the legs are defined as the following (below, left):

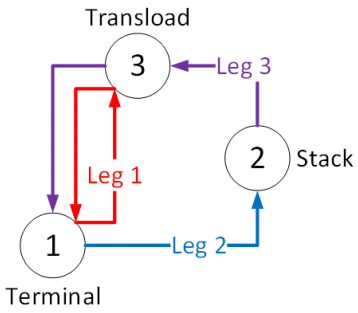

Fig. 2. Transport leg definitions.

The decision variables include the start times of each container-chassis-leg combination, s_{klc}. If a container-chassis-leg combination is not in the set of assignments, decision variable $x_{klc} = 0$. In this case, s_{klc} is also constrained to be 0 (3.1, Constraint 6). This makes Constraint 5 to be valid.

Containers assigned to a leg 2 transport job must also be assigned to a leg 3 transport job. (3.1, Constraint 3). It follows that the start time of each container leg 2 transport job must be at least Φ_{k2} time (travel time from (1) to (2) in Fig. 2) before its leg 3 transport job start time (3.1, Constraint 5). The big-M term on the left in constraint 5 switches off the constraint if the container is routed through leg 1 instead. Constraint 4 ensures that containers only take one route or the other (leg 1 vs. leg2/3). Containers also have a release time at which they are released for transport from their

respective carriers. Start times of all transport jobs must be at or after the respective release times of each container (3.1, Constraint 2).

Finally, for each individual chassis, they can only take 1 container at a time and must complete the leg-container combination before traveling to the next job. As such, this resource constraint is formulated as Constraint 1 where D represents the travel time between completion of leg l and start of leg l'. This behavior can be likened to an extension of the resource constraints in the typical disjunctive JSP model where not only can jobs allocated to a chassis not overlap, but the constraint needs to account for the fact that a travel delay is required between the end of one transport job and the start of the next ($D_{ll'}$). This inter-leg travel time is not modeled in the constraint programming model.

To complete the discussion on the comparison with the classical disjunctive JSP model, the z variable returns here in the pair of chassis constraints as $z_{klk'l'c}$. Similarly, to the disjunctive JSP, it also is used in a big-M formulation in each constraint in the pair which accounts for the two different possible orderings of jobs constrained against one another. Φ_{kl} is equivalent to the typical processing time in the classical JSP but here it is a function of both the container k and leg l and represents the total duration of leg l which includes transport time and also processing at the transload in the case of leg 1 and 3 (Fig. 2). A summary of the differences between the formulation here in 3.1 and the classical disjunctive JSP is as follows:

1. A job is represented by both the container k and leg l.
2. The parameter $D_{ll'}$ is introduced to represent a travel delay between successive jobs, where the first job is leg l and the second job is leg l'.
3. The left hand side of constraint 1 has extra big M terms to 'switch off' the constraint when $x_{klc} = 0$.

A result of this comparison is that since the job shop scheduling problem is NP-hard, the problem presented in this paper is also NP-hard since the job shop scheduling problem is a simplification of this problem. If legs 2 and 3 (Fig. 2) are eliminated, the problem becomes the JSP problem since there is no longer travel between jobs, and a job is defined solely by the container since there is no assignment of containers to legs (only leg 1 remains).

3.3 Alternative MIP formulations

The problem presented in this paper is very clearly in the class of job shop scheduling (more specifically parallel machine scheduling/ flexible job shop) problems due to the presence of a planning horizon, finite resources, and jobs (containers) arriving in a time-dependent manner (release times). However, on a first glance this problem may resemble a variant of the vehicle routing problem (VRP) which is a multi-salesman generalization of the travelling salesman problem (TSP). The reason why this is a completely different problem with little to no parallels is due to the fact that the VRP does not have a time component and solely deals with a static set of travel demands. The minimization of cost in the VRP is concerned with the travel cost along arcs whereas in the problem presented in this paper, the minimization of cost is concerned with the waiting cost of

containers over the time dimension. The scheduling MIP formulation makes this even more clear.

An alternative MIP formulation can potentially be produced as a variant of the time-indexed JSP model. The modification to include travel between transport jobs likens this potential formulation to the operating room scheduling MIP model formulated by Marques et. Al. [1], where the operating rooms must be cleaned between operations, with a cleaning delay γ:

$$\sum_{c \in C} \sum_{\substack{t'=t-p_c+1-\gamma \\ t' \in T_c}}^{I} x_{crt'd} \leq 1 \ , \ \forall r \in R, \ t \in T, \ d \in D \qquad (5)$$

In both cases, there is a delay that resources must undergo between taking jobs. The time-indexed JSP MIP also features precedence constraints that are necessary for this problem with the precedence of leg 2 before leg 3. The modification that needs to be made to the time-indexed model is similar to the one made to the disjunctive model that produced the formulation in 3.1: Jobs are to be defined as an assignment of containers to legs simultaneously to the scheduling of containers to chassis. This approach was considered however due to historic results of the disjunctive model outperforming the time-indexed model, the disjunctive model was decided to be the better starting point.

4 Constraint Programming Formulation

As a concept, interval variables in Constraint Programming are useful for modeling scheduling problems, having a start date, end date, and length of time which an activity was performed. If specified, interval variables can be excluded from a model. As such, the main idea behind the CP model was to represent the time in the transload facility as an interval that must be included, and an optional interval of time to represent the time spent in the stack, if needed.

The truck scheduling problem of the chassis resource is very similar to the unrelated parallel machine scheduling problem. There are 2 methods to model resources in scheduling, using Cumulative or Unary resources [2][3]. In the MIP model, the chassis resources are modeled as Unary resources, with the movement between each task for a given resource all being tracked sequentially, much like a sequence of jobs on a machine in JSP. One of the simplifications made compared to the MIP, is that the CP model the resources are modeled as a pool of combined resources using a elementary cumulative resource function called Pulse, in constraints (5)(6), which seizes a specified number of resources from a pool (Chassis_usage) for the length of the interval variable, and automatically released once completed. There is a single additional constraint which caps the number of actively used chassis below a specified threshold (7). The number of legs considered has reduced to the distance between terminal-transload and terminal-stack, which both seize a chassis resource for the container to be moved.

The objective function is very similar to the MIP version: summing over all demurrage penalty costs for tardiness of leaving the shipping terminal, the detention costs to

minimize the tardiness of transload completion for all containers, the optional stack costs simplified by the interval variable presence values included in the optimal solution, and the priority costs being a function of completion time at transload. Constraints (1)-(3) specify the interval lengths at transload using P_k, start times greater than the arrival time of the ships arriving in port, and specifying the length of the leg trip. The End_Before_Start global constraints in (4) ensure that precedence constraints, regardless of the path that a container decides to take, are met. The constraint programming formulation is implemented using IBM ILOG CPLEX CP Optimizer using the Python Docplex library.

4.1 CP Model

Sets:

K	$Set\ of\ containers, k \epsilon K$
C	$Set\ of\ chassis, c \in C$
J	$Set\ of\ carriers, j \in J$

Parameters:

R_k	$Release\ time\ of\ container\ k\ [days]$
Γ_j	$Demurrage\ cost\ at\ terminal\ per\ unit\ time\ for\ carrier\ j\ after\ free\ period\ [^{\$}/_{container * days}]$
Γ''_j	$Free\ period\ before\ demurrage\ cost\ for\ carrier\ j\ at\ terminal\ [days]$
S	$fixed\ cost\ at\ stack\ [^{\$}/_{container}]$
S'	$unit\ cost\ at\ stack\ per\ unit\ time\ [^{\$}/_{container * days}]$
δ''_j	$Free\ period\ before\ detention\ cost\ for\ some\ container\ of\ carrier\ j\ [days]$
δ_j	$Detention\ cost\ per\ unit\ time\ for\ some\ container\ of\ carrier\ j\ [^{\$}/_{container * days}]$
L	$Duration\ os\ leg\ to\ both\ stack\ and\ transload$
ρ_k	$Priority\ factor\ of\ container\ k, higher\ value\ is\ higher\ priority$

Variables:

$transload_k$	$Interval\ variable\ representing\ time\ traveled\ to\ and\ processed\ at\ trasnload\ of\ container\ k$
$trip\_leg_k$	$Optional\ interval\ variable\ representing\ time\ spent\ on\ leg\ to\ stack$
$stack_k$	$Optional\ interval\ variable\ representing\ time\ spent\ in\ stack\ of\ container\ k$
$chassis\_usage$	$Elementary\ cumulative\ function\ variable, represneting\ pooled\ chassis\ usage$

Model formulation:

$$min \sum_{k \epsilon K} \sum_{j \epsilon J} \Gamma_j * \max\{(Start\_of(trip\_leg_k) - R_k - \Gamma_j^{"}), 0\}$$ — Demurrage Costs

$$+ \sum_{k \epsilon K} \sum_{j \epsilon J} \delta_j * \max\{(End\_of(transload_k) + Length\_of(stack_k) - R_k - \delta"_j), 0\}$$ — Detention Costs

$$+ \sum_{k \epsilon K} S * Presence\_of(stack_k) + S' * Length\_of(stack_k)$$ — Stack Costs

$$+ \sum_{k \epsilon K} \sum_{c \epsilon C} \rho_k * End\_of(transload_k)$$ — Priority Penalty Costs

#	***subject to***:		***Description***
1	$Length\_of(transload_k) = P_k$	$for\ all\ k\ \epsilon\ K$	For each container, the length of the transload interval is P_k
2	$Start\_of(trip\_leg_k) \geq R_k$	$for\ all\ k\ \epsilon\ K$	Release time constraint denoting arrival time of ships
3	$Length\_of(trip\_leg_k) = L$	$for\ all\ k\ \epsilon\ K$	Length of trips are predefined as L
4	$End\_Before\_Start(trip\_leg_k, stack_k)$ $End\_Before\_Start(trip\_leg_k, transload_k)$ $End\_Before\_Start(stack_k, transload_k)$	$for\ all\ k\ \epsilon\ K$	End_Before_Start global constraints to respect precedences on container k
5	$Chassis\_usage\ +=\ Pulse(transload_k, 1)$	$for\ all\ k\ \epsilon\ K$	Cumulation resource expression, where transload_k seizes 1 chassis resource for the interval duration
6	$Chassis\_usage\ +=\ Pulse(trip\_leg_k, 1)$	$for\ all\ k\ \epsilon\ K$	Seize 1 chassis resource for trip length
7	$Chassis\_usage \leq \lvert C \rvert$		Chassis usage must always be below the number of total available chassis

4.2 The CP formulation rationale

The CP model in section 4.1 uses interval variables to solve for this truck/container scheduling problem with limited number of parallel resources. The interval variables represent the time spent in transload, stack and in travelling through the legs. The optional variables in Constraint Programming which are useful in this scheduling case to represent the optional path to the stack for temporary storage. The goal is to reduce the

overall cost of assigning containers to different facilities with different priority levels of containers.

5 Problem Instances

Problem instances were created with a Python instance generator notebook that makes use of the uniform probability distribution numpy.random function to generate arrays of random values within defined bounds. This is seen in Fig. 2, which represents the arrival of containers at the terminal for a particular instance. The containers arrive on ships on random days and are allocated randomly to a carrier. Similarly, Fig. 3 depicts the random distribution, typically triangular distribution of 1 to 5 days, with the most common being 2 days, of how many containers correspond to a particular processing time at the transload facility. Detailed instance tables can be found in Appendix 1 with explanations on how the instances are organized.

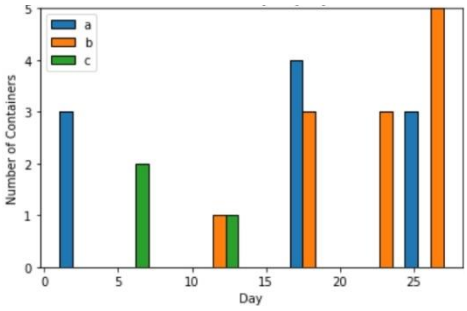

Fig. 2. Arrival of containers by carrier by day

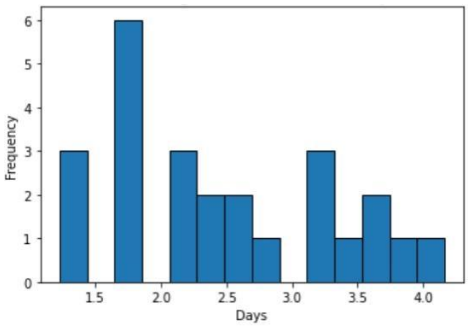

Fig. 3. Number of containers with a given processing time at Transload Facility

To generate the actual instances, several key parameters were adjusted based on realistic scenarios. Table 1 below provides a summary of the differences between instances. For example, instance_3 attempts to view the effects of potentially using the Stack more because fewer chassis are available and costs at the terminal are much

higher. In this instance, keeping several chassis with the container during the entire transload process would not work because the objective function would incur very high costs at the terminal. Shuttling remaining containers to the Stack is necessary for an optimal solution in this instance when compared to the base instance.

Table 1. Summary of all instances

Inst-ance	Description	Contain-ers (k)	Carri-ers (j)	Chassis (c)	Time Horizon	Ships	Demurrage Cost Range	Stack Cost (fixed/ variable)	Leg Length
1	Base - All others compared to this	10	3	5	7	3	$100-$200	300/15	standard
2	High number of containers for MIP	20	3	5	7	3	$100-$200	300/15	standard
3	Fewer chassis and higher demur-rage cost	10	3	2	7	3	$1000-$3400	300/15	standard
4	More containers but over a longer period	15	3	5	10	3	$100-$200	300/15	standard
5	More carriers and more ships over a month	15	5	5	30	7	$100-$200	300/15	standard
6	Lower stack cost	15	3	5	7	3	$100-$200	100/10	standard
7	short travel, lower stack cost and high dem.	14	3	5	7	3	$1000-$3400	100/10	closer stack
8	Carriers split over many ships with high dem.	15	3	5	30	10	$1000-$3400	300/15	standard
9	More containers over a longer pe-riod of time	25	3	5	30	10	$100-$200	300/15	standard
10	More containers with high pri. and high dem.	25	3	5	30	10	$1000-$3400	300/15	standard

6 Results

The running time of the models and the optimality gap for each instance were measured to compare each models' performances. Table 2 displays the results for each instance. Note that for the sake of this project, the team decided to set a time limit of 1 minute for both models. The reason for this is to allow the user to run the model multiple times in a short period of time, which is more efficient in case the user wants to try different scenarios.

Table 2. Running time and optimality gap for each model and instance.

Instances	MIP (Time and Gap)		CP (Time and Gap)	
1	1-minute limit	9.2%	0.51 seconds	Optimal
2	1-minute limit	100%	1-minute limit	38%
3	38.98 seconds	Optimal	10.19 seconds	Optimal
4	31.20 seconds	Optimal	1.57 seconds	Optimal
5	1-minute limit	23.1%	0.56 seconds	Optimal
6	1-minute limit	100%	2.02 seconds	Optimal
7	1-minute limit	100%	27.37 seconds	Optimal

8	11.54 seconds	Optimal	0.72 seconds	Optimal
9	36.62 seconds	Optimal	2.36 seconds	Optimal
10	37.39 seconds	Optimal	5.12 seconds	Optimal

Based on the information on Table 2, the CP Model was far superior to the MIP model. It found an optimal solution 9 out of 10 instances, while the MIP model found a solution 5 out of 10 instances. Another factor that demonstrates the CP model's superiority was the run time. It was significantly faster on all runs except on Instance 2, which was the only instance where the CP model did not reach an optimal solution. However, the CP model still decreased the gap to 38% while the MIP model continued at 100%.

7 Discussion

Looking at Table 2 as well as the Problem Instance Section, there is a pattern between the models' performances and the instances.

Whenever the time horizon of the instances was shorter, the performances of the models were negatively affected by it. This happened because the proportion of containers per day ended up being higher as the time horizon decreased. For example, when the time horizon was 7 days (the shortest time frame), there was an average of around 14 containers for those days. In other words, an average of 2 containers a day. When the time horizon was 30 days (the longest time frame), there was an average of around 24 containers. In other words, less than a container a day. When the average of containers of day was higher, the instance became more complex for the models and their performances worsened.

Another aspect of the instances that affected the results was the number of chassis. When the number of chassis was smaller, the performances of the models improved. The chassis are the resources of this problem. They need to be assigned to a container and a leg. By having more resources available to use, the problem becomes more difficult to solve because there are more options, which increases the number of iterations for the models to run.

The number of containers also affected the performance of the models. As the number of containers increased, the performances decreased. The containers represent the operations the resources (chassis) must do. By having more operations, the problem becomes more complex.

There were also other variables that did not seem to affect the performance of the models. Changing the demurrage costs and the stack costs did not have any effect on the models as much as the other 3 variables. There was no significant difference when in the time to run the models or in the optimality gap.

It is also important to talk about the limitations of the models. Both models' performances are worsened due to symmetry. Symmetry makes both programs take significant more time since the solution space is too large. The MIP model suffered the most from this performance damper. The model required some variables to have many

dimensions because the chassis behaved independently of each other and they could move around without having a container with them (as explained in section 3.2 and 3.3).

8 Conclusions

For this truck scheduling problem, the CP model is preferred over MIP model because it had a better performance throughout all instances.

On the other hand, the instances used for this project do not fully depict reality. There are days where there are 80 containers arriving at the terminal, while our instances would be much smaller by having 20 containers in 7 days. It would be almost impossible for any of the models to run if the instances fully resembled reality.

For future research, it is recommended to use some sort of greedy heuristic because it would most likely provide a solution to a real-life instance within a reasonable time using reasonable processing power to solve it. The final solution would most likely not be optimal, but it would be better than assigning a random schedule.

References

1. Marques, I. et. Al. An integer programming approach to elective surgery scheduling: Analysis and comparison based on a real case. OR Spectrum, 34:407–427 (2012).
2. Gökgür, B., Hnich, B., & Özpeynirci, S. Parallel machine scheduling with tool loading: A constraint programming approach. International Journal of Production Research, 56(16),5541–557. https://doi.org/10.1080/00207543.2017.1421781. (2018).
3. Baptiste, P., Laborie, P., Pape, C. L., & Nuijten, W. Constraint-based scheduling and planning. In Foundations of Artificial Intelligence (Vol. 2, pp. 761–799). Elsevier. https://doi.org/10.1016/S1574-6526(06)80026-X. (2006).

Appendices

Appendix A – Problem Instances

For each instance, the first row contains the file name, which in the sample above is instance_base. The second row (row1) contains summary data of the number of containers (k), number of carriers (j), number of chassis (c), the fixed cost at the Stack (S') and the variable cost at the Stack (S). In the above example, we have 10 containers split over 3 carriers with 5 chassis available as resources to move the containers. Rows 2 to 6 contain container specific data such as the carrier the container belongs to, the container release date, processing time at the trans-load facility and the priority of the container. Rows 7 to 10 contain carrier specific data such as the respective fixed and variable demurrage or detention costs. The last row contains the travel time of each of the three legs. All time durations are expressed in days.

Table A1. Instance format using instance_base.

	Info Descr.	**col_0**	**col_1**	**col_2**	**col_3**	**col_4**	**col_5**	**col_6**	**col_7**	**col_8**	**col_9**
row_0	**file name:**	instance_base									
row_1	**summary data:**	10	3	5	300	15					
row_2	**container (k):**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
row_3	**carrier (j):**	a	b	c	b	b	a	a	c	a	c
row_4	**release date (R_k):**	0	0	0	2	2	4	4	6	6	1
row_5	**transload proc. time (p_k):**	1	1	2	3	4	1	2	1	2	2
row_6	**priority (ρ_k):**	1	2	3	4	5	6	7	8	9	10
row_7	**demurrage free period (T''_j)**	**a:**	3	**b:**	2	**c:**	5				
row_8	**demurrage cost/day (T_j)**	**a:**	100	**b:**	14	**c:**	18				
row_9	**detention free period (δ''_j)**	**a:**	21	**b:**	14	**c:**	18				
row_{10}	**detention cost/day (δ''_j)**	**a:**	50	**b:**	75	**c:**	100				
row_{11}	**travel time/leg**	**leg_1:**	0.125	**leg_2:**	0.125	**leg_3:**	0.125				

Table A2. Instance_2 (first 10 columns)

instance_2 -- 20 containers -- 3 carriers -- 5 chassis -- 7-day horizon									
20	3	5	300	15					
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
a	b	a	c	c	a	c	b	b	c
0	5	0	0	0	0	0	5	5	0
2.4	1.3	2.6	2.4	2.4	2.2	1.9	2.9	2.1	2.0
13	5	19	1	10	6	4	11	2	18
a:	2	**b:**	5	**c:**	5				
a:	100	**b:**	100	**c:**	200				

a:	28	**b:**	28	**c:**	14				
a:	150	**b:**	100	**c:**	50				
leg_1:	0.125	**leg_2:**	0.125	**leg_3:**	0.125				

Table A3. Instance_2 continued (columns 10 to 19 of rows 2 to 6)

11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
a	a	a	b	c	a	c	c	c	b
0	0	0	5	0	0	0	0	0	5
2.8	2.6	1.7	2.5	1.9	3.4	3.7	2.5	3.6	1.6
8	17	15	3	12	7	20	14	16	9

Table A4. Instance_3

instance_3 -- 10 containers -- 3 carriers -- 2 chassis -- 7-day horizon									
10	3	2	300	15					
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
c	a	b	a	a	a	b	b	c	b
1	2	0	2	2	2	0	0	1	0
2.68	3.13	2.08	2.58	3.87	3.88	1.71	1.91	1.45	2.41
6	5	2	3	10	7	8	1	4	9
a:	4	**b:**	2	**c:**	5				
a:	3400	**b:**	1000	**c:**	2000				
a:	21	**b:**	21	**c:**	21				
a:	50	**b:**	100	**c:**	100				
leg_1:	0.125	**leg_2:**	0.125	**leg_3:**	0.125				

Table A5. Instance_4

instance_4 -- 15 containers -- 3 carriers -- 5 chassis -- 10-day horizon														
15	3	5	300	15										
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
c	c	b	b	a	a	c	b	c	a	b	c	b	a	a
5	5	1	1	7	7	5	1	5	7	1	5	1	7	7
4.37	2.67	4.43	3.15	3.1	1.93	4.47	1.16	2.01	2.4	3.37	1.89	3.72	4.55	1.81
13	1	7	4	5	10	12	3	15	14	9	2	6	8	11
a:	4	**b:**	4	**c:**	7									
a:	150	**b:**	150	**c:**	150									

a:	14	**b:**	28	**c:**	14									
a:	50	**b:**	50	**c:**	50									
leg_1:	0.13	**leg_2:**	0.13	**leg_3:**	0.13									

Table A6. Instance_5

instance_5 -- 15 containers -- 5 carriers -- 5 chassis -- 30-day horizon														
15	5	5	300	15										
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
c	b	d	e	c	c	a	b	b	d	b	c	b	e	e
10	13	20	15	10	10	9	10	13	20	13	10	10	3	3
3.87	2.17	3.54	1.41	1.66	2.8	2.62	2.36	2.18	2.87	2.4	3.22	2.6	2.75	2.94
12	14	7	8	3	9	6	13	2	1	15	5	4	10	11
a:	4	**b:**	7	**c:**	3	d	4	e	6					
a:	150	**b:**	200	**c:**	150	d	150	e	150					
a:	14	**b:**	14	**c:**	21	d	14	e	14					
a:	150	**b:**	150	**c:**	100	d	150	e	150					
leg_1:	0.13	**leg_2:**	0.13	**leg_3:**	0.13									

Table A7. Instance_6

instance_6 -- 15 containers -- 3 carriers -- 5 chassis -- 7-day horizon														
15	3	5	100	10										
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
a	b	c	a	b	c	a	b	c	c	c	a	c	a	a
4	6	1	4	6	1	4	6	1	1	1	4	1	4	4
1.55	3.37	2.4	3.18	4.49	2.65	2.55	1.54	2.04	2.55	3.04	3.47	2.27	1.51	2.08
9	6	11	3	13	1	2	15	14	8	12	4	7	10	5
a:	6	**b:**	3	**c:**	4									
a:	150	**b:**	100	**c:**	200									
a:	28	**b:**	21	**c:**	14									
a:	50	**b:**	150	**c:**	150									
leg_1	0.13	**leg_2:**	0.13	**leg_3:**	0.13									

Table A8. Instance_7

instance_7 -- 14 containers -- 3 carriers -- 5 chassis -- 7-day horizon

14	3	5	100	10									
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
a	b	b	b	c	a	a	a	b	b	a	b	c	a
1	3	3	3	4	1	1	1	3	3	1	3	4	1
3.77	4.39	3.75	2.63	1.96	1.21	2.39	2.32	2.61	2.5	2.69	2.66	3.31	3.14
11	8	3	12	13	7	1	14	9	6	10	2	5	4
a:	5	**b:**	7	**c:**	7								
a:	3000	**b:**	1800	**c:**	3100								
a:	21	**b:**	14	**c:**	14								
a:	100	**b:**	50	**c:**	100								
leg_1:	0.25	**leg_2:**	0.063	**leg_3:**	0.063								

Table A9. Instance_8

instance_8 -- 16 containers -- 3 carriers -- 5 chassis -- 30-day horizon															
16	3	5	300	15											
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
c	a	c	b	c	a	c	b	b	c	a	a	c	b	a	b
22	28	22	21	1	28	1	21	24	1	22	28	22	21	22	29
1.2	2.56	2.54	1.73	1.75	1.94	2.36	2	1.58	2.2	1.82	3.79	4.23	1.39	3.1	2.74
8	12	14	10	3	1	4	5	11	2	16	9	7	6	13	15
a:	6	**b:**	3	**c:**	3										
a:	100	**b:**	300	**c:**	100										
a:	28	**b:**	14	**c:**	21										
a:	100	**b:**	150	**c:**	100										
leg_1:	0.13	**leg_2:**	0.13	**leg_3:**	0.125										

Table A10. Instance_9

instance_9 -- 25 containers -- 3 carriers -- 5 chassis -- 30-day horizon												
25	3	5	300	15								
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
b	b	a	a	b	a	b	b	a	b	b	c	a
25	4	27	9	4	9	25	4	29	25	25	17	9

3.34	1.29	2.9	3.26	2.55	1.95	1.89	3.3	1.82	1.59	3.06	4.25	1.13
6	2	14	3	13	20	15	8	12	23	4	24	7
a:	3	**b:**	5	**c:**	2							
a:	300	**b:**	300	**c:**	150							
a:	14	**b:**	21	**c:**	28							
a:	100	**b:**	100	**c:**	50							
leg_1:	0.125	**leg_2:**	0.125	**leg_3:**	0.125							

Table A11. Instance_9 continued (columns 14 to 25

14	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**
b	a	c	a	c	a	a	a	c	a	c	c
4	27	17	29	28	0	0	0	17	27	17	29
2.58	3.5	2.84	3.17	2.08	4.01	3.15	2.66	1.75	2.26	3.02	2.41
19	11	22	25	9	21	17	18	1	16	5	10

instance_10 -- 25 containers -- 3 carriers -- 5 chassis -- 30-day horizon												
25	3	5	300	15								
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
b	a	b	b	a	b	c	a	b	a	a	b	a
27	17	27	12	25	23	7	17	23	25	17	18	25
1.85	1.28	2.46	3.18	2.36	2.52	1.23	2.52	4.16	3.66	3.2	1.66	3.87
11	20	7	15	4	10	12	9	6	21	3	5	22
a:	3	**b:**	6	**c:**	7							
a:	2600	**b:**	1100	**c:**	2600							
a:	21	**b:**	28	**c:**	21							
a:	100	**b:**	50	**c:**	50							
leg_1:	0.125	**leg_2:**	0.125	**leg_3:**	0.125							

14	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**
b	a	b	a	c	a	b	c	b	b	a	b
27	17	18	1	7	1	27	13	18	27	1	23
3.69	1.81	2.9	1.29	1.68	2.14	1.79	3.3	3.52	2.2	2.14	1.67
16	1	24	25	23	19	8	2	13	14	18	17