Dynamic Routing and Scheduling for Fleet and Containers with Stochastic Demand in Maritime Container Transport

Author: Runqi Hong[1] and Supervisor: Takashi Irohara[2]
[1, 2] Sophia University
[1] rh.runqihong@gmail.com
[2] irohara@sophia.ac.jp

Abstract. Maritime transport is the most energy-efficient method for transporting large volumes of goods and plays a key role in the global supply chain. However, the overall efficiency is still restricted by the quality of planning, human factors, and level of market integration. This research proposed a novel planning algorithm for maritime container transport, which aims to establish an efficient and flexible container transport network with high scalability. The proposed planning algorithm decides routing and scheduling for ships and containers in epochs over a planning horizon. Empirical data is used to confront uncertain demand and decisions are made by three consecutive processes in each epoch: Container routing; departure scheduling; and ship inventory rebalancing. The effectiveness of the algorithm is evaluated by computational simulation and sensitivity analysis.

Keywords: routing and scheduling; metaheuristic; computer simulation

1 Introduction

Container transport is the most common method to transport commodities. Emerging of container shipping is the result of containerization from the last century [1] and maritime container shipping is currently servicing about 90% of non-bulk cargo transport globally.

Two mainstream business models for maritime container transport are liner shipping and tramp shipping. Liner shipping services work in a way that is like bus services. Liner shipping companies research the global transport pattern and schedule the commercial operating routes to provide transport services [2]. The schedules of commercial routes are then released, and customers can book available shipping capacity along the routes. The lesser common tramp shipping relies on broker markets to get access to transport contracts. Tramp shipping services, therefore, are more flexible and provide on-demand services to customers. Despite being the most energy efficient among all means, the overall efficiency, in terms of time and cost, of container transport is still far from optimal. Many reasons are leading to the low efficiency of maritime container transports.

Liner shipping has the largest sector in container transport in terms of market share. The busiest major commercial routes commonly have multiple port calls along their routes and therefore prone to introducing excessive travel distance more than needed for individual containers. The predetermined schedule also made it less flexible compared to on-demand transport services. The advantage of liner shipping is a result of economies of scale which comes from the high efficiency of large vessels, equipment, and facilities in this regard.

The biggest issue of tramp shipping is that most businesses run independently in small sizes and therefore causes segmentation issue in the planning phase. Segmented resources and demands drive the unit costs higher than how much are necessary. This problem exists not only in the tramp shipping business but also for every liner shipping company because each agent shares the same market. Meanwhile, the qualities of planning of transport service providers differ dramatically. This further impairs the overall efficiency of container transport.

Human factors cause difficulties in maritime container transport planning. The vast majority of container shipping businesses rely on human crews to operate ships for transport, and humans are incapable of working around the clock as machines do and are vulnerable to accidents [3][4]. The result is that transport schedules are dependent on human schedules. Other miscellaneous reasons such as bloated organizations and excessive intermediaries also prevent unit costs from getting lower.

With the recent development in autonomous container ships, there is hope that the difficulty caused by human factors in planning will be alleviated for more cost-efficient operating strategies to be implemented in the future [5]. The tendency of integration in market and planning also facilitate improving of operational efficiency in time and cost [6][7][8].

2 Literature Review

In the general field of maritime transport research, many problems have been modeled and a variety of methods have been proposed to solve related problems. Lee et al. [9] proposed a single trip route planning and speed optimization model to minimize fuel consumption. Homsi et al. [10] proposed an exact branch and price algorithm and a hybrid metaheuristic for solving the tramp ship routing problem. Wu et al. [11] proposed a mixed-integer linear programming model to solve the empty container reposition problem with consideration of the container life stage. Hemmati et al. [12] proposed a heuristic for solving combined cargo and inventory routing problems for service providers who provide VMI and regular shipping service simultaneously. Zhen et al. [13] formulated a model for integrated planning of ship scheduling and container routing problem in liner shipping and proposed three heuristics for solving different sizes of problem. Wetzel et al. [14] proposed an integrated model for solving liner shipping fleet deployment and repositioning problem and used real-world data to show benefit of simultaneous optimization. Liu et al. [15] proposed a two stage distributionally robust optimization model for inventory routing problems in maritime transportation.

In summary, there are two common types of problems: Fleet scheduling problems for liner shipping; and integrated routing/scheduling problems for fleet/containers. However, the latter is commonly restricted to solve small-scale problems which involve hundreds of containers and dozens of ships. Meanwhile, not many studies consider a dynamic system which's demand will be introduced over the planning horizon or uncertain demand. The problem that considers large-scale integrated routing and scheduling of ships and containers with consideration of uncertain demand has not been well studied. Both scopes of studies and solution methods are restricting such studies from appearing. First off, the problem has not been encountered in the real world. Also, the conventional way that uses a single mathematical model to model a problem and solves it with exact or heuristic requires exponentially high computational power to solve large-scale problems. Integrated routing and scheduling of ships and containers globally are expected to be the tendency of future maritime container logistics; therefore, this research intends to do an advanced expedition into that field. To solve large-scale problems effectively, a novel method for modeling and solving problems is urgently needed.

3 Problem Definition

A new business model for maritime container transport is proposed as a basic framework for more efficient ship/container planning strategies to be implemented. In this model, a single planning system is responsible for all ship and container managements. This promotes full integration of demand and resources to takes full advantage of economies of scale. Ports are used as nodes for ship anchorage, container storage, and ship-container operations(loading/unloading). Customers can initialize contracts that request one or many containers to get transported from original nodes to destination nodes. However, no guaranteed delivery date will be given to a shipping contract. Container ships will enlist as service providers and transport assigned containers with instructed routes and schedules. There are three types of physical entities in the system. They include containers with stochastic origin/destination and time of appearance, a fleet of heterogenous ship models, and multiple ports for ship anchorage and container storage, which located on a directed graph. The planning horizon, T, is divided into multiple epochs, which $T = \{1, 2, 3, ..., o\}$. Every epoch, t for $t \in T$, is a short period of time, and is defined to be 1 hour in this research.

3.1 Physical Entities

Ports. Ports are nodes on a directed graph, $G = (V, E)$, which consists of all nodes $V = \{1, 2, 3, ..., l\}$ and edges of all unique pairs of origin and destination nodes, $E = \{\left(V_i, V_j\right)\} \forall i \in V, j \in V, i \neq j$. Ports are used for ship anchorage and container storage. Port nodes are where ship-container operations are conducted.

Ships. Ships come with multiple ship models and each ship model differs in terms of lightweight, capacity, and speed range. Ships are carriers for containers and can travel in between port nodes through edges. The movement of ships generates fuel consumption. Fuel consumption is dependent on ship model, displacement, and speed. Set of all ships, $S = \{1, 2, 3, ..., m\}$.

Containers. Containers will appear stochastically over a planning horizon. A container has stochastic origin and destination. Containers can be stored at port nodes and can be loaded onto and transported by any container ship with available capacity. Every container is identical in weight and will take up 1 unit of capacity of a ship when loaded. A container is not revealed until appearance. Set of all containers, $C = \{1, 2, 3, ..., n\}$.

3.2 Decision Variables

Starting from beginning of the planning horizon with initial conditions of ports and ships, containers start to appear. Decisions for ships' and containers' routing/scheduling are needed to instruct transport operations. Decision variables are series of movements of every ship and container over planning horizon. Decision variables define routes and schedules of movements. The movements of containers need to be compatible with ship movements. Compatibility between movements of containers and ships includes but is not limited to ship capacity and departure/arrival time.

3.3 Measurements/Objectives

The general objective is to reduce unit cost and unit time span for every unit of value of service provided. In other words, the objective is to maximize cost efficiency and time efficiency. Regarding the cost, this research only considers fuel cost generated by ship movement since it is the major contributor to operational cost in maritime transport. Therefore, necessary measurements are quantity of service delivered, cumulative fuel consumption, and cumulative time span.

Throughput. Value of service is only delivered when demand is satisfied. At any moment during planning horizon, the throughput of shipping service delivered is equivalent to the summation of shortest distance from the origin to destination of every container that has been delivered. It does not involve actual travel distance to avoid bias, because containers can get routed to intermediary nodes.

At any time, t, during planning horizon, T, there will be a set of containers that have been delivered, $C_t^d = \{1, 2, 3, ..., u\}$, which is a subset of all containers, C. At any time, for any container in C_t^d, $C_{at}^d \quad \forall a \in C^d,\ t \in T$, there is a shortest distance from its origin to destination, D_{at}^d. At any given time during the planning horizon, throughput, ϵ_t, can be represented as

$\epsilon_t = \sum_{a \in C_t^d} D_{at}^d$.

Cumulative fuel consumption. Fuel consumption is measured and recorded every time a ship arrives at a port node. A fuel consumption formula is used to estimate fuel consumption generated from ship movements based on ship model, deadweight, and speed during travel. By any moment during planning horizon, the summation of fuel consumption generated by finished ship movements is cumulative fuel consumption.

At any time, t, during time horizon, T, any ship $s \in S$, will have a recorded cumulative fuel consumption, $F_{st} \quad \forall s \in S, t \in T$. At any given time during the planning horizon, overall cumulative fuel consumption, γ_t, can be represented as $\gamma_t = \sum_{s \in S} F_{st}$

Cumulative time span. Time span is calculated and recorded when a container arrives at its destination. Time span of transport is the duration from the initialization of a container to the time of its final delivery. By any moment during planning horizon, the summation of time span of every delivered container is cumulative time span.

At any time, t, during planning horizon, T, there will be a set of containers that have been delivered, $C_t^d = \{1, 2, 3, ..., u\}$, which is a subset of all containers, C. At any time, for any container in C_t^d, $C_{at}^d \quad \forall a \in C^d,\ t \in T$, there is a recorded time duration from its spawn to delivery, T_{at}^d. At any given time during the planning horizon, cumulative time span, β_t, can be represented as $\beta_t = \sum_{a \in C_t^d} T_{at}^d$.

Cost efficiency is calculated in terms of throughput per fuel consumption. Time efficiency is calculated in terms of throughput per time span. The objective is to maximize cost efficiency and time efficiency. Fuel consumption per throughput at any given time is $\theta_t = \frac{\epsilon_t}{\gamma_t}$. Delivery time span per throughput at any given time is $\lambda_t = \frac{\epsilon_t}{\beta_t}$.

4 Problem Analysis

Uncertainty and dynamic. The uncertainty in this problem comes from the uncertain demand over the planning horizon. Even though every container is the same in weight and compatibility to ships, the stochastic spawn time, origin, and destination make it hard to utilize common methods used for solving static routing and scheduling problems. Meanwhile, the decision can be made throughout the planning horizon which means former decisions will transform the problem along with uncertain demand. As a result, it is necessary to implement dynamic programming.

Fuel efficiency. Fuel efficiency is different from cost efficiency. Fuel efficiency refers to transport capacity provided by per unit of fuel cost. Transport capacity can be measured in the unit, transport unit of container for unit of distance. For any transport operation, the product of number of containers onboard and travel distance is equivalent to quantity of transport capacity provided. According to Barras [16], the formula for daily fuel consumption of a ship is:

$$F(v, \nabla) = \lambda \cdot v^{3} \cdot \nabla^{\frac{2}{3}} \#\#(1)$$

F is daily fuel consumption. v is ship speed in hours. ∇ is displacement of ship. λ is constant coefficient dependent to ship engine. If the number of containers onboard is z, the fuel consumption per container per distance, F^{u}, can be derived:

$$F^{u}(v, \nabla) = \frac{\lambda \cdot v^{2} \cdot \nabla^{\frac{2}{3}}}{24 \cdot z} \#(2)$$

Knowing that each ship model has a feasible speed range, lower ship speed will cost less fuel compared to higher ship speed because the exponent on v is greater than 1. Also, because the displacement of a ship is the sum of the ship's lightweight and weight onboard, each additional container loaded will generate diminishing extra fuel consumption compared to formers do supposing each container has the same weight. Therefore, the minimum unit fuel consumption can be achieved by using ship model with the best possible fuel-efficient, sailing at minimum speed, and using all capacity. To further achieve maximum cost efficiency, it is necessary to transport containers directly from origin to destination, so that minimum fuel is used for every container. If building a maritime container network for maximum cost efficiency: allow only direct transport and put containers with the same origins and destinations into same container queues when they initially appear; always ensure availability of most fuel-efficient ship model at all ports; when a container queue's size reaches or exceeds the maximum capacity of the most fuel-efficient ship, depart the ship fully loaded to containers' destination at minimum speed. This ensures the best cost efficiency but time span for delivering containers will be long and uncertain due to biased policy and uncertain demand.

Shipping time span. The total shipping time span of a container consists of two parts, the travel time onboard along the route and waiting time at ports. container loading/unloading time is not considered in this research. To minimize travel duration, it is necessary to limit containers to only direct transport, so total travel distance is minimized. Then, using ship model with the highest possible max speed to transport containers. To minimize waiting time at ports, make ship model with the highest possible max speed always available at all ports and transport a container as soon as it appears.

Summary. To achieve highest cost efficiency, it requires ship model with best possible fuel efficiency and always use it to full utilization at minimum speed. Moreover, it is necessary to directly transport all containers so cumulative travel distance can be minimized. To achieve highest time efficiency, it requires cargos get transported as soon as they appear in network to minimize waiting time at nodes. Also, it requires using ship model with highest max speed at full speed. Also, it is important to use direct transport to minimize travel distance. There is a trade-off between cost and time efficiency in ship and container planning but achieving both objectives prefer containers to get direct transport, and they both demand availability of optimal ship model. The controversy are decisions of compatible ship/container movements on shipping legs, such as ship model selection, departure time, ship speed, and container section for departure. To tackle this problem of designing an efficient maritime container transport network, an algorithm that divides the parent problem into three sub-problems is proposed.

5 Planning Algorithm

The planning algorithm divides the planning horizon into epochs. With initial conditions and uncertain demands appears over time, decisions are made within each epoch. optimization in epochs are iterations of three optimization algorithms for deciding ship and container movements.

First algorithm routes containers based on customer preference when containers appear or arrive at intermediary ports. It routes containers directly to destinations by default but will route containers to the fastest routes by estimation if requested. It also set tentative deadlines for containers when they get routed. The second algorithm runs parallelly for each shipping leg. It keeps doing matchmaking between each container queue and idle ships at the origin node of the queue, then decides ship model, time, speed, container selection for the next departure. The objective of this sub-problem is to reduce unit cost for transport capacity provided. Both fuel consumption and penalties from lateness contribute to the unit cost. The third algorithm runs parallelly for each ship model. It rebalances ship inventories to target levels for all ports. The purpose is to ensure the availability of ship models for transport. The objective of this sub-problem is to reduce the total travel distance for rebalancing.

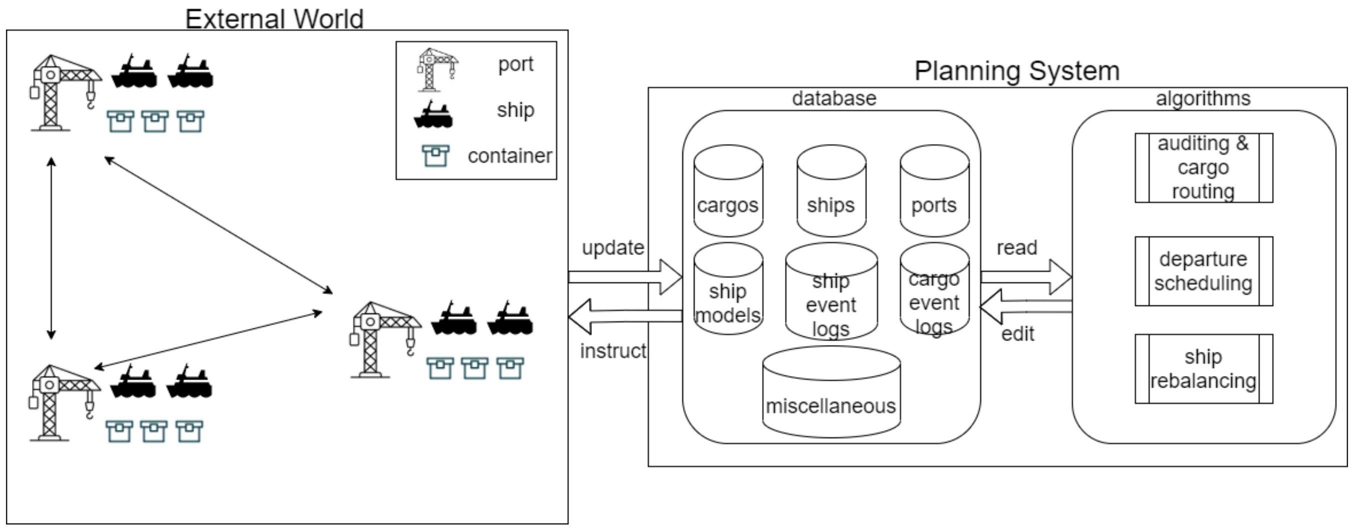


Fig. 1. Conceptual diagram of planning system interacting with external world

Fig. 1. is a diagram that shows general interactions between external world and planning system. During planning horizon, planning system constantly monitors physical entities in external world and update physical entities' status. Data is stored and planning algorithm can read data. After using algorithms to make calculations and decisions, data will be edited. decisions for ships and cargos will be stored and sent to external world as instructions.

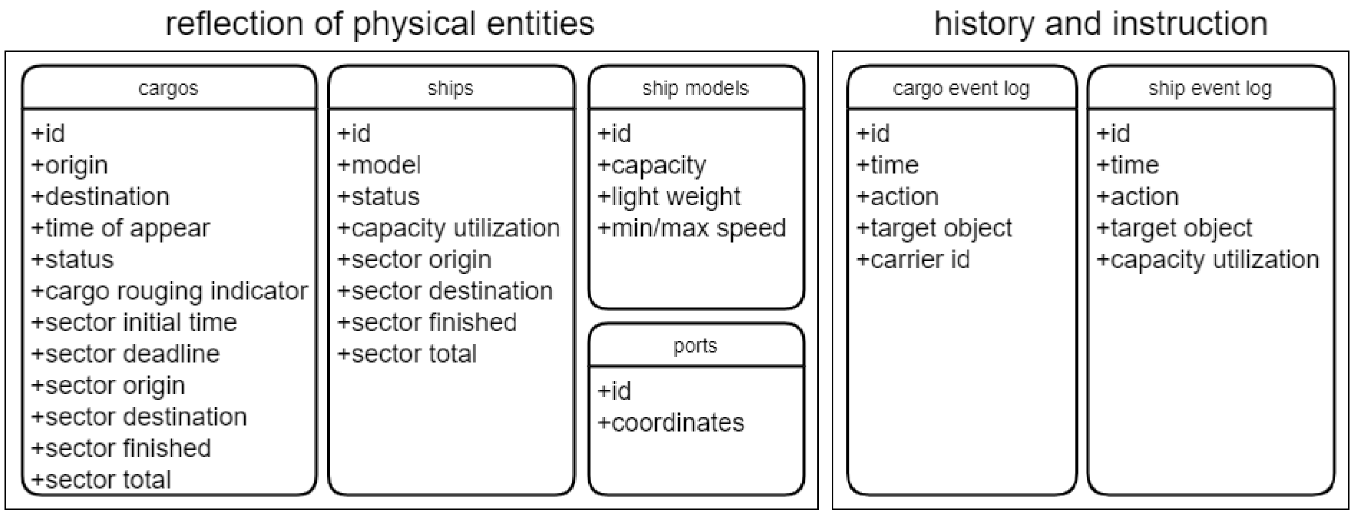


Fig. 2. data modeling of physical entities and event logs

Fig. 2. shows data modeling of data that are stored and used. These data can be classified into two categories: data that are reflections of physical entities and event log that record ship/container movement history and instruction for future. While there are some miscellaneous datasets, such as distance matrix of ports and routing table. Miscellaneous datasets are crated and updated throughout simulation process.

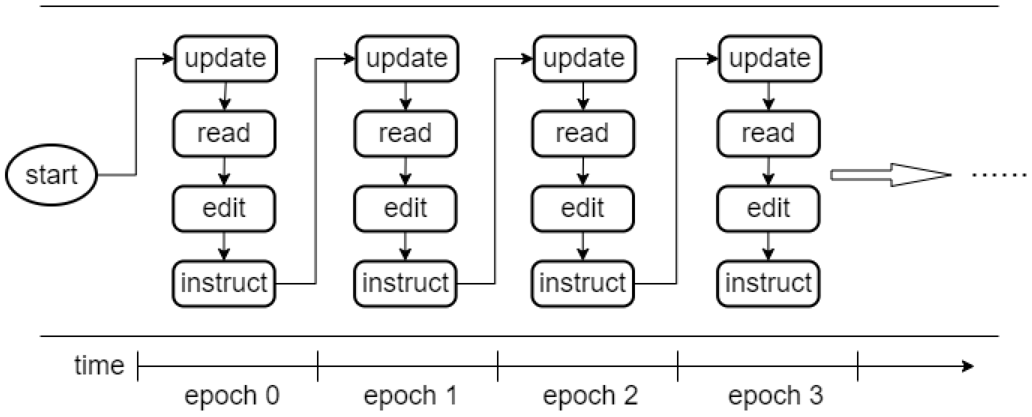
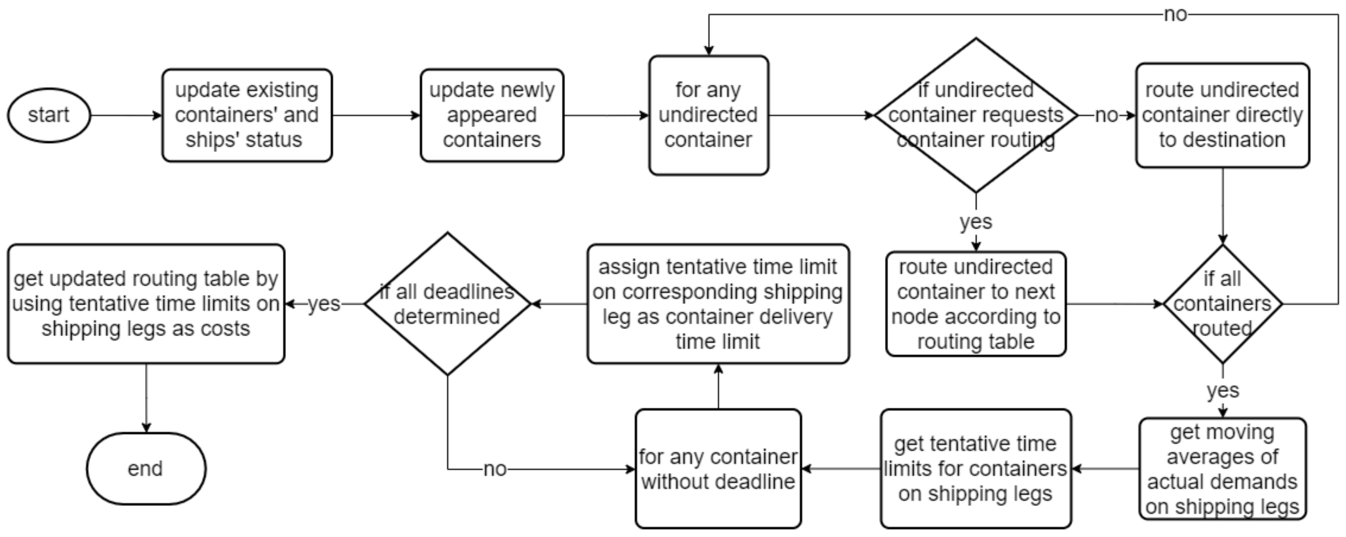


Fig. 3. Diagram of iterations of planning algorithm in epochs

Fig. 3. shows repeated processes of planning system update information, make decisions, and instruct ship/container operations. For every epoch, planning system goes through consecutive processes of update, read, edit, and instruct, mentioned in Fig. 1.

5.1 Container Routing Algorithm



start
update existing containers' and ships' status
update newly appeared containers
for any undirected container
if undirected container requests container routing
no
route undirected container directly to destination
no
yes
route undirected container to next node according to routing table
if all containers routed
yes
get moving averages of actual demands on shipping legs
get tentative time limits for containers on shipping legs
for any container without deadline
assign tentative time limit on corresponding shipping leg as container delivery time limit
if all deadlines determined
no
yes
get updated routing table by using tentative time limits on shipping legs as costs
end

Fig. 4. Container routing algorithm flowchart

The purpose of this algorithm is to simulate the routing of containers under default direct transport policy with customer customizability. The algorithm will route a container based on routing table and customer preference. If a container request routing, it will be routed along the route that is estimated to ensure earliest final delivery. Otherwise, a container will be routed directly to its destination. Also, this algorithm is responsible for assigning a tentative deadline for every routed container on its next shipping leg.

Demand of transport capacity. When a container has its next shipping leg determined during an epoch, it will be counted towards the demand on next shipping leg in same epoch. The planning algorithm records container demand patterns over the planning horizon on all shipping legs. The information is used to predict future demand. Current approach is to use linear regression or moving averages of demand over a period for demand forecasting.

Estimated cost-in-time. After getting demand pattern, an estimated cost-in-time for a container to get through a shipping leg can be calculated with the following information: Travel distance of shipping leg i to j, $D_{ij}\ \forall i \in V,\ j \in V$; Forecasted quantity of demand appears on a shipping leg i to j, $Q_{ij}^{MA}\ \forall i \in V,\ j \in V$; Target value of number of containers onboard for departures on shipping leg i to j,$Q_{ij}^{d}\ \forall i \in V,\ j \in V$; Target value of ship speed for departures on shipping leg i to j, $v_{ij}\ \forall i \in V,\ j \in V$. If container queue never exceeds capacity of largest ship at the end of each epoch and ships are always available, expected value of maximum time span for container to go through shipping leg i to j, $T_{ij}\ \forall i \in V,\ j \in V$, is:

$$T_{ij} = \frac{D_{ij}}{v_{ij}} + \frac{Q_{ij}^{d}}{Q_{ij}^{MA}} \#(3)$$

First term is average travel duration, and second term is average interval between departures. The values of v_{ij} and Q_{ij}^{d} can be selected freely within feasible range to explore effects of different speed/utilization settings on time efficiency. In this research, the minimum speed and maximum capacity of most fuel-efficient ship model are used for this estimation.

Routing table. The routing table records the shortest path for the lowest cost-in-time for each pair of directed origin and destination. The routing table is used to determine the routing of containers that request container routing. In general, a routing table requires using the Dijkstra algorithm to find the shortest paths on directed or undirected graphs with costs on edges. In this research, estimated maximum durations to get through shipping legs are used as costs. Afterwards, algorithm decides where a container should be routed to when it initially appears at its origin or is delivered at an intermediary node. When a container refuse routing, only direct transport will be considered for the container. When a container request routing, the algorithm will look up for the shortest path which estimates the earliest delivery of the container in worst case scenario. After deciding the next shipping leg for a container, the algorithm sets a tentative time limit to transport the container on the next shipping leg. The estimated maximum duration on a shipping leg mentioned in the auditing process will be used as the tentative time limit**.**

Initialization. At the beginning of planning horizon, a dummy routing table will be generated. It only indicates direct transport for every origin-destination pair because no historical data can be used to estimate cost-in-time for transport a container on shipping legs.

The following is pseudocode for container routing algorithm after initialization phrase.

```
Container routing algorithm
update existing ship/container status
update newly appeared containers status
for every undirected container:
    if undirected container requests container routing:
        route undirected container to next node according to routing table
    else:
        route undirected container directly to destination
get forecasted demand of actual demands on shipping legs
get tentative time limits for containers on shipping legs
for every container without deadline:
    assign tentative time limit on corresponding shipping leg as container delivery time limit
get updated routing table by using tentative time limits on shipping legs as costs
```

5.2 Departure Scheduling Algorithm

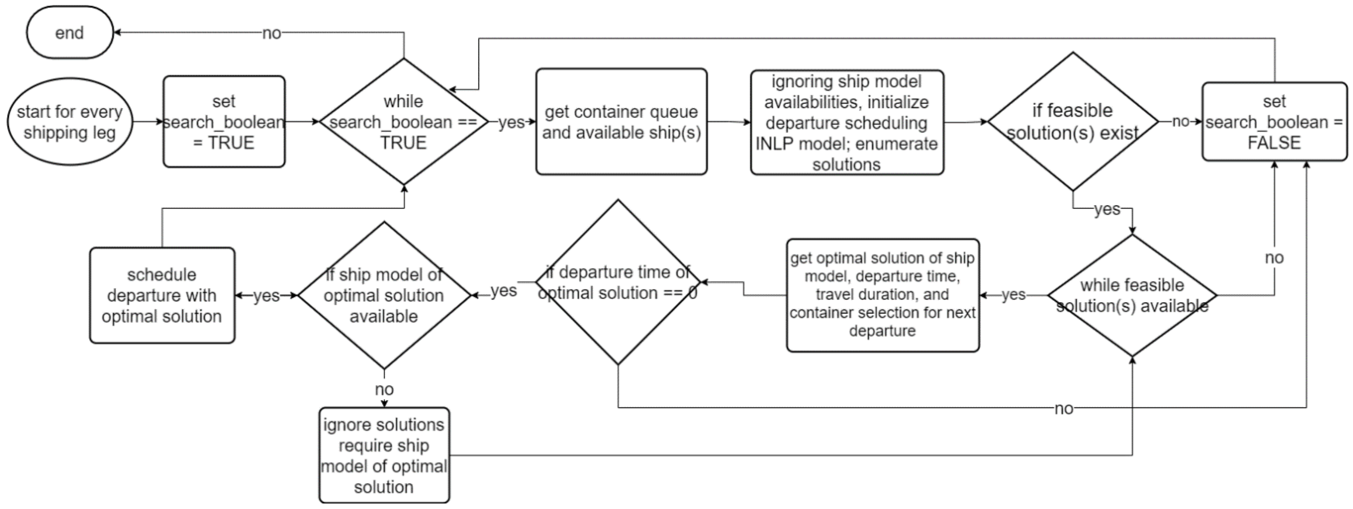

end
no
start for every shipping leg
set search_boolean = TRUE
while search_boolean == TRUE
yes
get container queue and available ship(s)
ignoring ship model availabilities, initialize departure scheduling INLP model; enumerate solutions
if feasible solution(s) exist
no
set search_boolean = FALSE
yes
while feasible solution(s) available
no
yes
get optimal solution of ship model, departure time, travel duration, and container selection for next departure
if departure time of optimal solution == 0
yes
if ship model of optimal solution available
yes
schedule departure with optimal solution
no
ignore solutions require ship model of optimal solution
no

Fig. 5. Departure scheduling algorithm flowchart

This algorithm runs parallelly for each shipping leg in every epoch. For every shipping leg, matchmaking will be made between containers waiting for transport and ships available at the shipping leg's origin node. The problem is modeled as an Integer Nonlinear Programming (INLP) model. The objective of the model is to minimize the unit cost generated by the next transport. The total cost consists of fuel cost and penalty for the lateness in delivery. Lateness in one transport is measured in the total quantity of lateness of every container transported. The decision variables are a combination of ship model, departure time, travel duration, speed, and container selection for the next departure.

Set	
C	set of all containers in current container queue, which $C = \{1, 2, 3, ..., n\}$
V	set of all ports, which $V = \{1, 2, 3, ..., l\}$
M	set of all ship models, which $M = \{0, 1, 2\}$
Parameter	
f	Constant coefficient for fuel consumption
D_{ij}	distance between two target port nodes i and j $\forall i \in V,\ j \in V$
P	penalty for per quantity of lateness
C_m	capacity of ship model m $\forall m \in M$
T_c	time limit for container c $\forall c \in C$
W	weight of container constant
W_m	lightweight of ship model m $\forall m \in M$
Q	container queue size at the epoch
Q_{ij}^{MA}	forecasted number of containers added to container queue i to j $\forall i \in V,\ j \in V$

D_{mij}^{max}	maximum number of epochs for ship model m to travel from i to j $\forall m \in M,\ i \ \in V,\ j \ \in V$
D_{mij}^{min}	minimum number of epochs for ship model m to travel from i to j $\forall m \in M,\ i \ \in V,\ j \ \in V$
Decision Variable	
t_c^-, t_c^+	integer variables for linearization $\forall c \ \in C$
x_m	binary decision variable, which $x_m = 1$ if ship model i is used, $x_m = 0$ otherwise $\forall m \in M$
y_c	binary decision variable, which $y_j = 1$ if container$j$ loaded, $y_j = 0$ otherwise $\forall c \ \in C$
d	integer decision variable, number of epochs for duration of next transport
t	integer decision variable, number of epochs from next departure

$$minimize \frac{f\cdot \sum_{m\in M} x_m \cdot \left(\frac{D_{ij}}{d}\right)^3 \cdot \left(\sum_{c\in C} y_c \cdot W + W_m \sum_{m\in M} x_m\right)^{2/3}}{\sum_{c\in C} y_c \cdot \frac{D_{ij}}{d} \cdot 24} + \frac{\sum_{c\in C} (P \cdot t_c^+)}{\sum_{c\in C} (y_c \cdot D_{ij})} \#(4)$$

s.t.

$$\sum_{m\in M} x_m = 1\#(5)$$

$$t \ \geq 0\#(6)$$

$$d \geq \sum_{m\in M} x_m D_{mij}^{min}\#(7)$$

$$d \leq \sum_{m\in M} x_m D_{mij}^{max}\#(8)$$

$$\sum_{c\in C} y_c \leq \sum_{m\in M} C_m x_m\#(9)$$

$$\sum_{c\in C} y_c \leq Q + t * Q_{ij}^{MA}\#(10)$$

$$y_c \geq y_{c+1}\#(11)$$

$$d + t - T_c = t_c^+ - t_c^-\#(12)$$

$$x_m \in \{0, 1\}\#(13)$$

$$y_c \in \{0, 1\}\#(14)$$

$$t_c^-, t_c^+ \in Z\#(15)$$

For constraints: (5) only one ship model used for a departure. (6) departure time is now or later. (7 - 8) travel duration constraints for every ship model. (9) ship capacity constraint. (10) container availability constraint (11) container processing order constraint. (12) linearization constraint. Due to the difficulty of solving INLP model, it is proposed to solve it by a metaheuristic algorithm that partially enumerate all combinations of decision variables. It enumerates all possible combinations of ship model, departure time, and travel duration. The containers in same container queue are processed in a way that a container can be processed for transport if no container with earlier deadline exists. Therefore, when decision variables other than container selection are decided, the sub-problem of container selection can be converted into a problem of determining number of containers to load onboard. Further, because every next selected container will have later or same deadline and equal weight, additional cost added by next container will diminish one after another. Therefore, it is always the optimal strategy to load as many containers as possible when other decision variables are decided for a combination.

After getting optimal solution, if optimal departure time is current epoch, algorithm will schedule a departure with optimal solution. If the target ship model is not available, ignore combinations related to target ship model and find next best option in current epoch. This process will continue until departure can be processed or no solution is available. Otherwise, if optimal departure time is in the future for a shipping leg, algorithm skip scheduling for this shipping leg. The following is pseudocode for departure scheduling algorithm.

Departure scheduling algorithm

```
for every shipping leg:
    search_boolean = TRUE
    while search_boolean == TRUE:
        get container queue, get available ship
        initialize departure scheduling
        for every ship model:
            estimate maximum time duration until contain queue exceed ship capacity (enumerate feasible departure times)
                for every integer departure time:
                    estimate possible travel durations according to speed range (enumerate feasible travel durations)
                    for every travel duration:
                        estimate # of containers onboard (get feasible # of containers onboard)
                        estimate unit cost
        if feasible solution(s) empty:
            search_boolean = FALSE
        while feasible solution(s) available:
            if optimal departure is now:
                if optimal ship model available:
                    schedule departure with optimal solution
                else:
                    delete solutions require optimal ship model
                    if feasible solution(s) empty:
                        search_boolean = FALSE
            else:
```

```
search_boolean = FALSE
delete all solutions
```

5.3 Ship Inventory Rebalance Algorithm

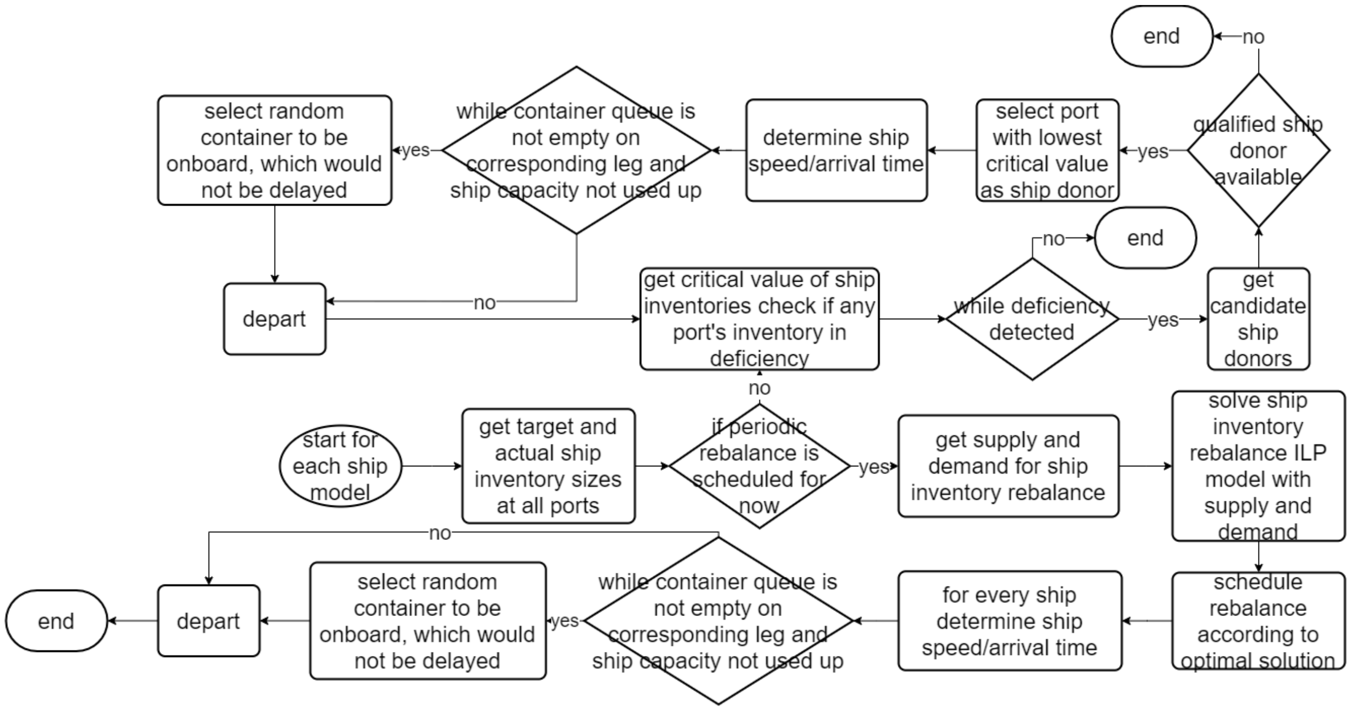

start for each ship model
get target and actual ship inventory sizes at all ports
if periodic rebalance is scheduled for now
yes
get supply and demand for ship inventory rebalance
solve ship inventory rebalance ILP model with supply and demand
schedule rebalance according to optimal solution
for every ship determine ship speed/arrival time
while container queue is not empty on corresponding leg and ship capacity not used up
yes
select random container to be onboard, which would not be delayed
depart
end
no
no
get critical value of ship inventories check if any port's inventory in deficiency
while deficiency detected
no
end
yes
get candidate ship donors
qualified ship donor available
no
end
yes
select port with lowest critical value as ship donor
determine ship speed/arrival time
while container queue is not empty on corresponding leg and ship capacity not used up
yes
select random container to be onboard, which would not be delayed
depart
no

Fig. 6. Ship inventory rebalance algorithm flowchart

Ship inventory is collection of ships that belong to same port. If a ship is idle, it is counted towards the ship inventory of where it is at; if ship is traveling, it is counted towards the ship inventory of the node where it is heading towards. Ship inventory rebalance algorithm rebalances the ship inventories of each ship model to maintain the availability of all ship models at all ports. The algorithm decides the target ship inventory level of all ship models for each port based on demand originated from every port and total number of each ship model, so the ratio of demand and ratio of target ship inventory sizes are as equal as possible. The algorithm creates an inventory rebalancing problem on a bipartite network consists of supply and demand nodes. The supply and demand are determined by actual/target inventory sizes of all ports and number of idle ships that can be repositioned. This sub-problem is modeled as an Integer Linear Programming (ILP) model with objective to minimize total travel distance. ILP model is implemented with Python MIP package and solved by Gurobi 9.1.2 for exact solution.

Set	
V	set of all ports, which $V = \{1, 2, 3, ..., l\}$
Parameter	
N_i	difference between the number of available ships and the target inventory level $\forall i \in V$

d_{ij}	distance from port *i* to port *j* $\forall i \in V,\ j \in V$
Decision variable	
x^{-}_{ij}, x^{+}_{ij}	Integer variable for linearization $\forall i \in V,\ j \in V$
x_{ij}	Integer decision variable, number of ships repositioned from port *i* to port *j* $\forall i \in V,\ j \in V$. x_{ij} takes positive value when port *i* is ship giver and takes negative value when port *i* is ship taker

$$minimize \sum_{i\in V}\sum_{j\in V} d_{ij} * \left(x^{+}_{ij} + x^{-}_{ij}\right)\#(16)$$

s.t.

$$x_{ij} + x_{ji} = 0 \quad \forall i \in V,\ j \in V\#(17)$$

$$x^{+}_{ij} - x^{-}_{ij} = x_{ij} \quad \forall i \in V,\ j \in V\#(18)$$

$$\sum_{j\in V} (x^{+}_{ij} - x^{-}_{ij}) = N_i \quad \forall i \in V\#(19)$$

$$x^{-}_{ij}, x^{+}_{ij}, x_{ij} \in Z \quad \forall i \in V, j \in V\#(20)$$

For constraints: (17) one's giving is others' taking. (18) linearization constraint (19) demand and supply satisfaction constraint

The ILP model is solved, and its solution is executed periodically. Another rebalance strategy will kick in during the gaps. The second strategy is a critical level detection method that constantly monitors inventory levels of all ship inventories. It will schedule a single ship paired reposition from ship inventory that has idle ship and with most surplus to ship inventory below critical level. This process will iterate until no critical level inventory is detectable or no suitable ship donor is available.

The following is pseudocode for ship inventory rebalance algorithm.

Ship inventory rebalance algorithm

```
for every ship model:
    get target_ship_inventory_sizes at all ports
    get actual_ship_inventory_sizes at all ports
    if periodic rebalance is scheduled for now:
        initialize getting supply and demand of ship inventories
        search_boolean = TRUE
        while search_boolean = TRUE:
            for every port:
            if target_ship_inventory_size - actual_ship_inventory_size >= 1:
                flag this port as ship_acceptor
            elif actual_ship_inventory_size - target_ship_inventory_size >= 1 and idle_ship_inventory size >= 2:
```

```
            flag this port as ship_donor
        **if** ship_donor not empty **and** ship_acceptor not empty:
            find random pair of ship_acceptor and ship_donor
            demand of ship_acceptor += 1; actual_ship_inventory_size of ship_acceptor + 1
            supply of ship_donor += 1; actual_ship_inventory_size of ship_donor - 1
        **else:**
            search_boolean = FALSE
    **if** supply and demand exit:
        solve ship inventory rebalance ILP with supply and demand
        execute ship rebalance plan
        **for** every repositioned ship:
            determine ship speed, calculate arrival time
            **while** container queue is not empty on corresponding leg **and** ship capacity not used up:
                select random container to be onboard, which would not be delayed
**else:**
    pairing_boolean = TRUE
    **while** pairing_boolean == TRUE
        get critical_value of ship inventories of all ports
        critical_value = (target_ship_inventory_size - actual_ship_inventory_size) / target_ship_inventory_size
        **for** every port:
            **if** critical_value > threshold and actual_ship_inventory_size **and** target_ship_inventory_size -
actual_ship_inventory_size >= 1:
                flag this port as ship_acceptor
            **elif** actual_ship_inventory_size - target_ship_inventory_size >= 1 **and** idle_ship_inventory size >= 2:
                flag this port as ship_donor
        **if** ship_acceptor(s) exist:
            **if** ship_donor(s) exist:
                select port with lowest critical value as ship_donor
                execute a paired single-ship rebalance from donor to acceptor
                determine ship speed, calculate arrival time
                **while** container queue is not empty on corresponding leg **and** ship capacity not used up:
                    select random container to be onboard, which would not be delayed
            **else:**
                pairing_boolean = FALSE
        **else:**
            pairing_boolean = FALSE
```

6 Computational Simulation

To testify the effectiveness of proposed algorithm, a computational simulation is conducted. Simulation program is implemented in Python and ran by a computer with AMD Ryzen 5 3600 CPU and 32GB of RAM. Ship and container movements are recorded, and performance metrics are measured throughout planning horizon. Planning horizon is set to be first 168 epochs of every instance. An instance defines numbers of ports, ships, and containers. A list of indexed ports is generated and randomly located on a graph of size 1000*1000. An instance with n ports will include first n ports from the list. Demand is defined by number of containers generated on each shipping leg in each epoch. 50% of randomly selected containers request container routing by default. Three ship models are available for simulation. Ship models of all ships are randomly distributed to three ship models. Ships are randomly located at nodes initially. Demands are defined with probability distribution with inputs.

Table 1. Ship information

Ship model	capacity	lightweight	min/max speed (distance/epoch)	Minimum unit cost per cargo per distance	Minimum unit time span per cargo per distance
0	15	5	8/20	7.50e-6	5.00e-2
1	30	10	9/22.5	7.54e-6	4.44e-2
2	45	15	9.33/23.33	**7.07e-6**	**4.29e-2**

Theoretical fuel consumption is lower bound of fuel consumption for transport container without violation of physics. It is equivalent to minimum unit fuel consumption in theory timed by throughput. Fuel consumption per throughput and shipping time span per throughput get measured starting from first delivery.

6.1 Performance and scalability benchmark

A standard group of instances of different number of ports and ships are used for standard simulations. Number of ships are scaled based on demand because more shipping legs introduce more demand. The purpose is to testify the scalability of algorithm for problem of different sizes. The algorithm should be able to output consistent high-quality decision that maintains time and cost efficiency of shipping network with increased number of shipping legs, ships, and more demand.

Table 2. Instance group A

Instances	Number of ports	Number of ships	Demand pattern
A1	5	400	uniform (0,10)
A2	10	1800	uniform (0,10)
A3	15	4200	uniform (0,10)
A4	20	7600	uniform (0,10)
A5	25	12000	uniform (0,10)

Table 3. Simulation results of instance group A

Instance	Throughput	Fuel consumption lower bound	Actual fuel consumption	Gap (%)	Unit cost in fuel	Unit cost in time	Computational time (s)
A1	5.15e6	3.65e1	4.05e1	11.0	7.87e-6	1.17e-1	257
A2	2.33e7	1.65e2	1.78e2	7.88	7.62e-6	1.16e-1	1263
A3	5.03e7	3.56e2	3.83e2	7.58	7.61e-6	1.18e-1	4786
A4	9.38e7	6.64e2	7.00e2	5.42	7.47e-6	1.17e-1	9643
A5	1.47e8	1.04e3	1.11e3	6.73	7.50e-6	1.17e-1	21677

Performance metrics are compared in epoch 168. The most important metrics are unit cost and unit time span which are most accurate measurement for efficiency in cost and time, respectively. The parameter settings used for instance group A are in favor of cost efficiency so the effectiveness can be judged from comparing the actual fuel consumption to theoretical lower bound for delivered containers. The theoretical lower bound of unit cost and unit time span are 7.07e-6 and 4.29e-2 in table 1, respectively. For largest and most difficult instance, A3, unit time span is 273% of the theoretical lower bound while its unit cost is 106.73% of its theoretical lower bound.

Results show good scalability in terms of giving solutions that maintain high efficiency in both cost and time. There are even noticeable improvements in unit cost with increase of shipping legs, ships, and more demand. However, the computational time grows rapidly with increasing number of ports. It has been confirmed that majority of the computational time are spent on enumerating decision variables for departure scheduling. Potential improvement from multiprocessing in enumeration is possible.

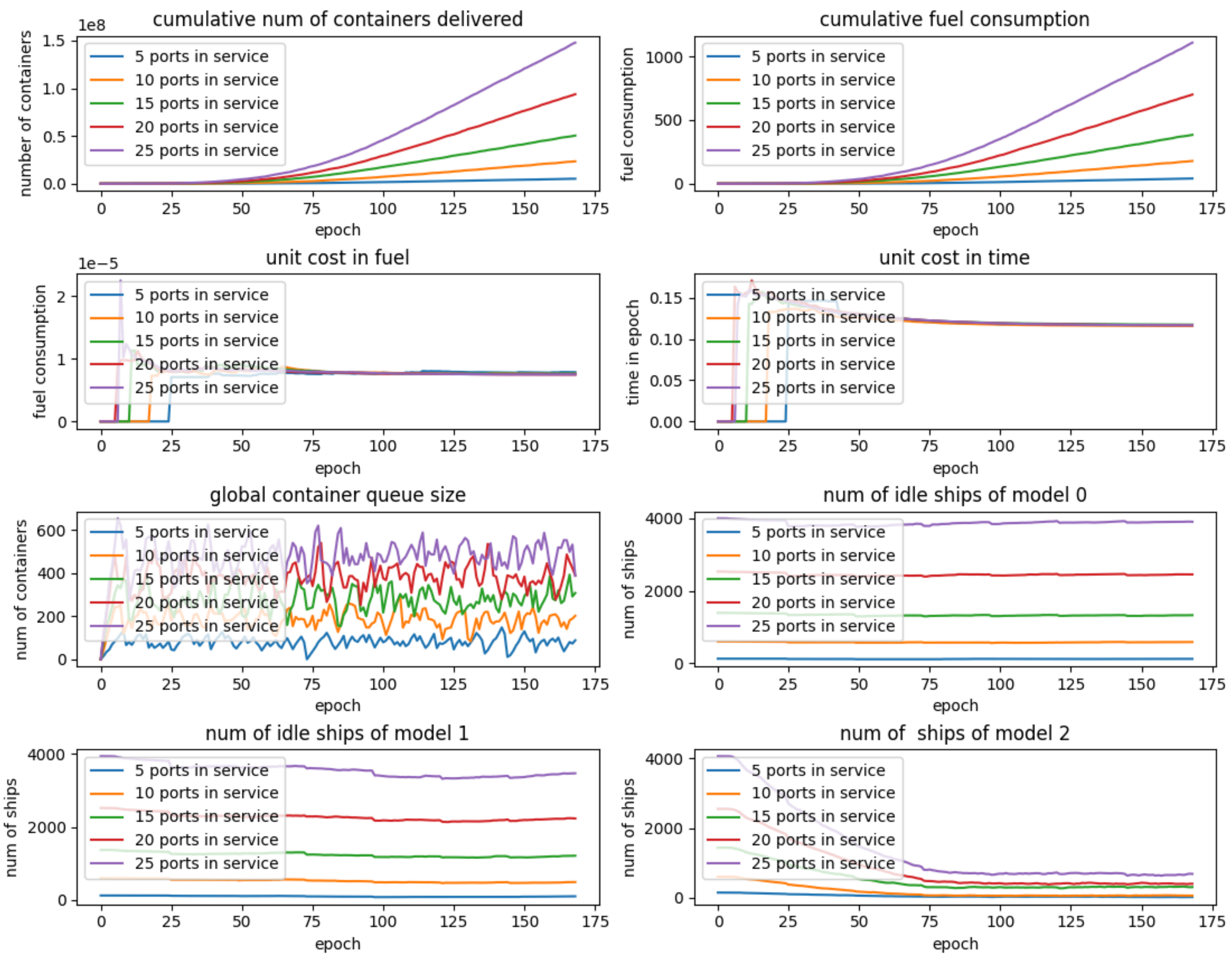


Fig. 7. performance metrics of three instances with 5, 10, 15, 20, 25 ports over planning horizon

6.2 Sensitivity Analysis – Container Routing

This section is to testify the effects of container routing on system performance. Five instances with different fractions of containers asking for container routing service are used for simulations. Instances have 0%, 25%, 50%, 75%, and 100% of containers requesting container routing service for instances B1, B2, B3, B3, B4, B5, respectively. Meanwhile locations of ports and demands on shipping legs are manually adjusted to magnify queue time in total shipping time span and encourage happening of container routing to intermediate nodes. Five ports are located at coordinates, (0,0), (0,1), (0,2), (0,3), (0,4), respectively from port 0 to port 4. 400 ships are generated for each instance. Demands on some shipping legs are manually scaled to encourage container routing.

Table 4. Demand pattern matrix for port 0-4

Demand matrix	0	1	2	3	4
0	0	uniform (0,10)	uniform (0,10)	uniform (0,10)	uniform (0,5)
1	uniform (0,10)	0	uniform (0,10)	uniform (0,5)	uniform (0,10)

2	uniform (0,10)	uniform (0,10)	0	uniform (0,10)	uniform (0,10)
3	uniform (0,10)	uniform (0,5)	uniform (0,10)	0	uniform (0,10)
4	uniform (0,5)	uniform (0,10)	uniform (0,10)	uniform (0,10)	0

Table 5. Simulation results of instance group B

Instance	Throughput	Fuel consumption lower bound	Actual fuel consumption	Gap (%)	Unit cost in fuel	Unit cost in time
B1	2.792e4	1.976e-1	2.176e-1	10.1	7.79e-6	2.85
B2	2.799e4	1.981e-1	2.217e-1	11.9	7.92e-6	2.90
B3	2.786e4	1.972e-1	2.222e-1	12.7	7.98e-6	2.82
B4	2.815e4	1.993e-1	2.256e-1	13.2	8.01e-6	2.80
B5	2.790e4	1.975e-1	2.255e-1	14.1	8.08e-6	2.74

Performance metrics are compared in epoch 168. With more containers in percentage request container routing, there is noticeable improvement in time efficiency with trade-off of cost efficiency.

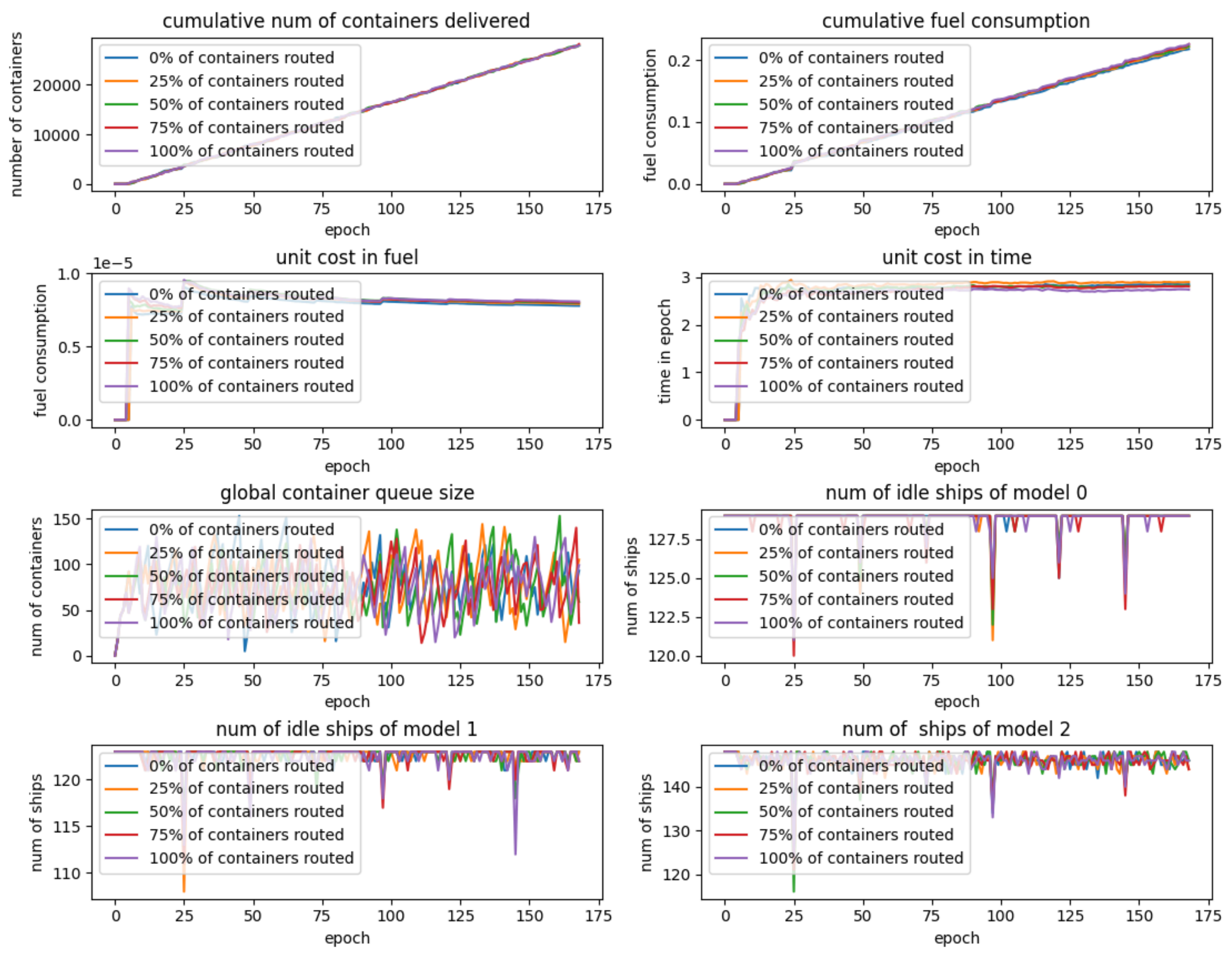


Fig. 8. performance metrics of three instances with 0%, 25%, 50%, 75%, 100% of containers accept routing over planning horizon

6.3 Sensitivity Analysis - Number of Ship in Service

This section is to learn the effects of number of ships in service on system performance. Five instances with 50, 100, 200, 400, and 800 ships were created, ship models are randomly distributed in 3 ship models.

Table 6. Instance group C

Instances	Number of ports	Number of ships	Demand pattern
C1	5	50	uniform (0,10)
C2	5	100	uniform (0,10)
C3	5	200	uniform (0,10)
C4	5	400	uniform (0,10)
C5	5	800	uniform (0,10)

Table 7. Simulation results of instance group C

Instance	Throughput	Fuel consumption lower bound	Actual fuel consumption	Gap (%)	Unit cost in fuel	Unit cost in time
C1	2.13e6	1.51e1	3.94e1	161	18.5e-6	1.29e-1
C2	3.65e6	2.58e1	5.75e1	123	15.8e-6	1.27e-1
C3	5.12e6	3.63e1	4.18e1	15.1	8.16e-6	1.19e-1
C4	5.16e6	3.65e1	4.06e1	11.2	7.87e-6	1.17e-1
C5	5.15e-6	3.64e1	4.14e1	13.7	8.04e-6	1.17e-1

Performance metrics are compared in epoch 168. Results show importance of maintaining enough idle ships at each port. System performance is impaired when not enough ship or not enough ship of ideal model is available. When ship inventory is insufficient, it increases both fuel consumption and time span for transport. However, excessive number of ships seems to negatively affect the system performance. The hypothesis is that more rebalances have been scheduled in instances with more ships, therefore impair the overall efficiency. More experiments are required to testify this hypothesis and an extra module that optimize the total number of ships in the system can be added to the algorithm in the future works.

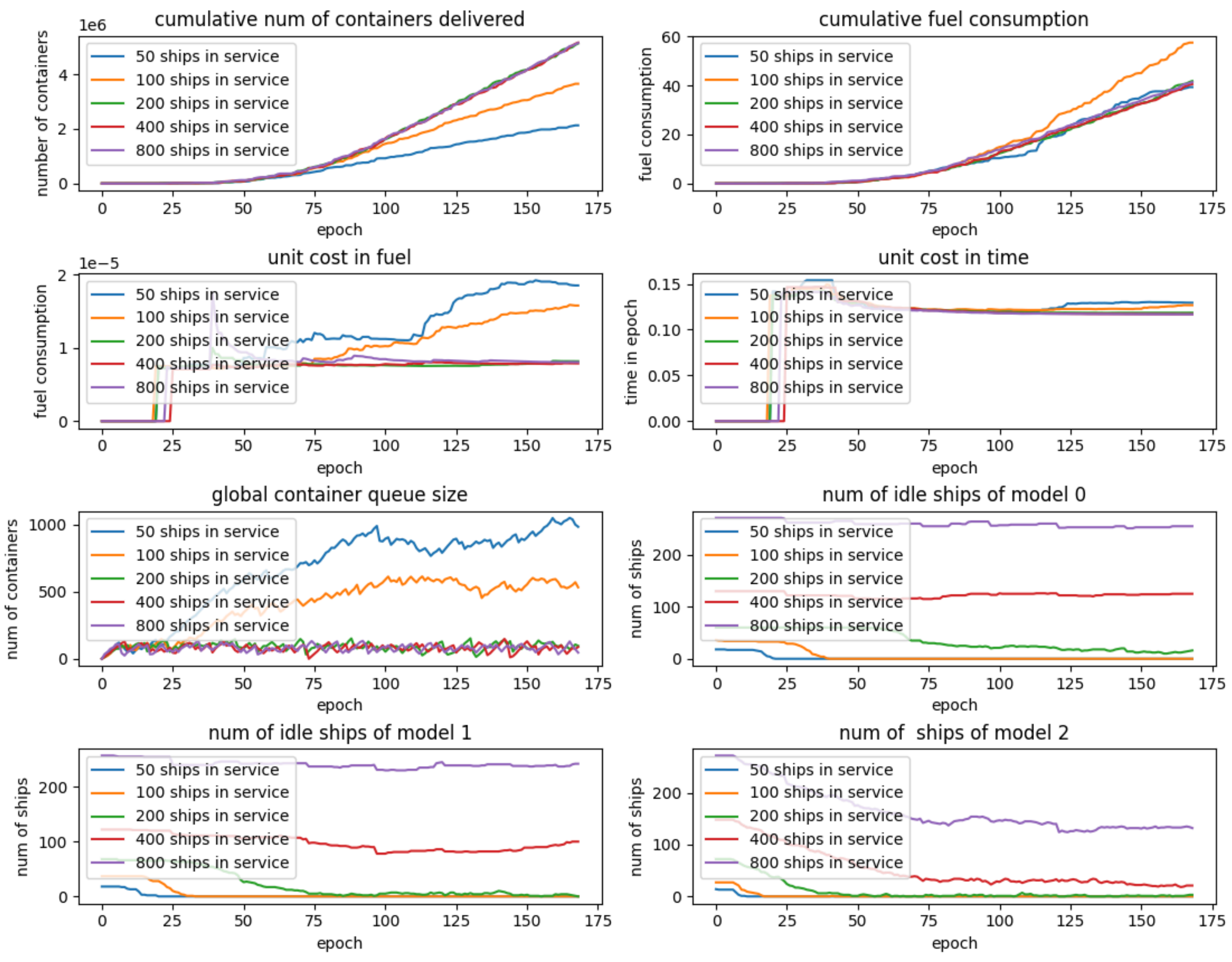


Fig. 9. performance metrics of five instances with 50, 100,200, 400, 800 ships in service over planning horizon

7 Conclusion

A planning algorithm is proposed to establish an efficient and flexible maritime container shipping network with high scalability by optimizing fleet and containers' movement. It uses empirical data in a dynamic environment to confront uncertainty in demand. The algorithm is epoch-based, so it is capable of continuously optimizing fleet and containers' movements. In each epoch, the algorithm divides parent problem into three sub-problems of container routing, departure scheduling, and ship inventory rebalancing. A hybrid approach which utilizes algorithm, metaheuristic, and solver is used to solve sub-problems. The planning algorithm has been tested to prove it maintains outputting solution with high efficiency in cost and time for problems of various sizes. Cargo routing is verified to have a positive impact on time efficiency with trade-off of cost efficiency. Planning algorithm is sensitive to number of ships. It requires enough ships, especially optimal ships, to maintain high efficiency in both time and cost. Compared to other studies in the topic of maritime routing and scheduling, this research attempted to solve the problem at a much larger scale by simplifying the parent problem in a dedicated way so that problems can be solved in reasonable amount of time. This could make real-time optimization possible.

To build a functional platform for customers and service providers, one feasible way is to let algorithm simultaneously decide dynamic freight rate and ship/container movements. Freight rate determines price for transport per container per distance for each ship model. Assume that the basics of transport service is potential customers can get benefit through using transport service to transport things to different locations. For every container, there are a quantified value of benefit and a cost which is dependent on travel distance and freight rate. A customer is expected to use the transport service if value of benefit is greater than cost of a transport. On the other hand, service providers get rewards for providing transport services while providing transport services generate operational cost. A service provider is expected to participate in the transport network if reward is greater than operational cost for providing service. Therefore, dynamically change pricing of freight rate can change the supply and demand in the market, and there is negative correlation between supply and demand. The goal of dynamic pricing is to maximize the throughput of transport network by finding equilibrium of supply and demand.

Source code. available at: https://github.com/Monoese/fleet_container_routing_scheduling

References

1. David Guerrero, Jean-Paul Rodrigue, The waves of containerization: shifts in global maritime transportation, Journal of Transport Geography, **34**, 151-164 (2014), ISSN 0966-6923, https://doi.org/10.1016/j.jtrangeo.2013.12.003
2. Theo Notteboom, Container (Liner) Shipping, Editor(s): Roger Vickerman, International Encyclopedia of Transportation, Elsevier, 247-256 (2021), ISBN 9780081026724, https://doi.org/10.1016/B978-0-08-102671-7.10254-4
3. Shiqi Fan, Eduardo Blanco-Davis, Zaili Yang, Jinfen Zhang, Xinping Yan, Incorporation of human factors into maritime accident analysis using a data-driven Bayesian network, Reliability Engineering & System Safety, **203**, 107070 (2020), ISSN 0951-8320, https://doi.org/10.1016/j.ress.2020.107070
4. Shiqi Fan, Jinfen Zhang, Eduardo Blanco-Davis, Zaili Yang, Xinping Yan, Maritime accident prevention strategy formulation from a human factor perspective using Bayesian Networks and TOPSIS, Ocean Engineering, **210**, 107544 (2020), ISSN 0029-8018, https://doi.org/10.1016/j.oceaneng.2020.107544
5. Jiri de Vos, Robert G. Hekkenberg, Osiris A. Valdez Banda, The Impact of Autonomous Ships on Safety at Sea – A Statistical Analysis, Reliability Engineering & System Safety, **210**, 107558 (2021), ISSN 0951-8320, https://doi.org/10.1016/j.ress.2021.107558
6. Plomaritou, Evi, A proposed application of the marketing mix concept to tramp & liner shipping companies, **13**, 59-71 (2008)
7. Enna Hirata, Contestability of Container Liner Shipping Market in Alliance Era, The Asian Journal of Shipping and Logistics, **33**(1), 27-32 (2017), ISSN 2092-5212, https://doi.org/10.1016/j.ajsl.2017.03.004
8. Federico Quartieri, Are vessel sharing agreements pro-competitive?, Economics of Transportation, **11–12**, 33-48 (2017), ISSN 2212-0122, https://doi.org/10.1016/j.ecotra.2017.10.004
9. Sung-Min Lee, Myung-Il Roh, Ki-Su Kim, Hoeryong Jung, Jong Jin Park: Method for a simultaneous determination of the path and the speed for ship route planning problems, Ocean Engineering, **157**, 301-312 (2018), ISSN 0029-8018, https://doi.org/10.1016/j.oceaneng.2018.03.068
10. Gabriel Homsi, Rafael Martinelli, Thibaut Vidal, Kjetil Fagerholt: Industrial and tramp ship routing problems: Closing the gap for real-scale instances, European Journal of Operational Research, **283**(3), 972-990 (2020), ISSN 0377-2217, https://doi.org/10.1016/j.ejor.2019.11.068
11. Shanhua Wu, Yu Sun, Feng Lian & Zhongzhen Yang: Reposition of empty containers of different life stages integrated with liner shipping network design, Maritime Policy & Management, **47**(1), 43-56 (2019), DOI: 10.1080/03088839.2019.1657973
12. Ahmad Hemmati, Magnus Stålhane, Lars Magnus Hvattum, Henrik Andersson: an effective heuristic for solving a combined cargo and inventory routing problem in tramp shipping, Computers & Operations Research, **64**, 274-282 (2015), ISSN 0305-0548, https://doi.org/10.1016/j.cor.2015.06.011
13. Lu Zhen, Shuaian Wang, Gilbert Laporte, Yi Hu, Integrated planning of ship deployment, service schedule and container routing, Computers & Operations Research, **104**, 304-318 (2019), ISSN 0305-0548, https://doi.org/10.1016/j.cor.2018.12.022.
14. Daniel Wetzel, Kevin Tierney, Integrating fleet deployment into liner shipping vessel repositioning, Transportation Research Part E: Logistics and Transportation Review, **143**, 102101 (2020), ISSN 1366-5545, https://doi.org/10.1016/j.tre.2020.102101.
15. Botong Liu, Qi Zhang, Zhihong Yuan, Two-stage distributionally robust optimization for maritime inventory routing, Computers & Chemical Engineering, **149**, 107307 (2021), ISSN 0098-1354, https://doi.org/10.1016/j.compchemeng.2021.107307
16. Barras, B.: Ship design and performance for masters and mates. Oxford: Elsevier, (2004), ISBN 0-7506-6000-7