Combined Strip and Discharge Delivery of Containers in Heterogeneous Fleets with Time Windows

Xinan Yang[*], Hajem A. Daham[†], Abdellah Salhi[‡]

Abstract

Inland transportation of containers contributes significantly to the total cost of container intermodal transportation. For this reason, it has received a lot of attention in the last few decades. While there are many reports of attempts to solve the problem out there, the variants considered are all simpler than the form addressed in this paper. Here, we consider the container transportation problem where pick-up and delivery orders, empty and loaded containers, Discharge and Strip of heterogeneous container types with time windows, are handled with heterogeneous truck fleets that carry one or two 20ft or one 40ft chassis. Moreover, to manage the movement and reuse of empty containers, two strategies for empties, i.e. Depot-turn and Street-turn, are simultaneously allowed in the problem setting. A novel MILP model for this rich transportation problem is developed, which applies to various scenarios, even when some functions/delivery modes/types of container, are disabled. Given the complexity of this problem, exact solution of large instances is not realistic. We, therefore, suggest a novel implementation of the Genetic Algorithm (GA) tailored to this particular problem in its rich form. It differs from existing ones for VRP problems due to the nature of the problem we are considering. Numerical experiments on examples with real geographical data show that combining Discharge and Strip containers in transportation saves on cost and increases fleet utilisation.

Keywords: Container Transportation, MILP, Genetic Algorithm, Strip, Discharge.

1 Introduction

The inland transportation refers to the delivery of loaded and empty containers amongst port terminals, container depots, rail hubs and customers. Since it contributes between 40% and 80% [29] to the total cost of the intermodal container business and involves large-scale,

[*]Dept. of Mathematical Sciences, University of Essex, Colchester CO4 3SQ, United Kingdom. email: xyangk@essex.ac.uk

[†]Dept. of Mathematics, Al-Muthana University, Samawah, Iraq. email: hajem.daham@mu.edu.iq

[‡]Dept. of Mathematical Sciences, University of Essex, Colchester CO4 3SQ, United Kingdom. email: as@essex.ac.uk

combinatorial decision makings, the inland transportation has received a lot of attention in the last decades [28]. The standard sizes of containers used for intermodal shipping around the world are the 20ft and 40ft. Consequently, most truck chassis are designed to carry these two types of containers; the 20ft chassis carries a single 20ft container; the 40ft combination chassis carries a single 40ft container, a single 20ft container or two 20ft containers [25]. Customers are categorized into two classes: importers who receive loaded containers from the port and exporters who send loaded containers to the port. Empty containers are moved between importers, exporters, the port and inland container depots based on the requests of customers. In the container industry, some of the shipping lines own fleets of trucks to perform the transportation, while others outsource it to local haulage companies. This study considers the former case, where the shipping line serves customer demands with its own fleet and decides where to collect/drop empty containers involved in the servicing process.

In practice, two standard container loading/unloading modes are used, i.e., "*Strip*" and "*Discharge*". In the *Strip* case, a container can be separated from its carrying truck/chassis and be left at its customer location to deal with (loading or unloading). The empty container that is demanded before loading or after unloading can be dropped-off/collected by the same truck or by a different one, depending on which option leads to a lower cost. This process will generate many requests for empties, and normally the customer does not care about where the empty box comes from/goes to. While under the *Discharge* mode, only the cargo will be loaded onto/unloaded from the delivery truck; containers and trucks are not separated after services. Traditional studies separate these two delivery modes when planning the route. However, there is a great potential in merging all possible types together to capture the practical situation, as both types exist in the demand based on customer choices and do not conflict to deal with on the same truck. Here, we suggest a methodology to plan the joint delivery of both types so as to reduce overall transportation cost. Note that the "*Strip*" mode may also be referred to as the "drop-and-pull" mode where a container with a chassis is left at a customer; mathematically they are not different when only one container is allowed per truck. However in this study we allow the simultaneous transportation of two 20ft containers on 40ft truck/chassis, which is obviously more complicated than the standard "drop-and-pull" works.

For this purpose, a novel MILP model for combining the inland transportation of heterogeneous (20ft and 40ft), loaded and empty containers under *Strip* and *Discharge* modes is developed in this paper. It is then solved using an original implementation of the Genetic Algorithm (GA) for large scale instances. We investigate the delivery process of 12 different order types, which covers all means of container transportation practices, using a heterogeneous fleet consisting of both 20ft and 40ft trucks. The two loading/unloading modes (*Strip* and *Discharge*) are examined both separately and jointly to evaluate the benefit of combining them. The two common strategies for managing the empty containers, say the *Street-turn* where empty containers are collected from import customers and directly delivered to exporter customers to fill with cargos, and the *Depot-turn* where empty containers can be stored at/collected from inland depots for later usages, are also simultaneously allowed by the model. Time window constraints at customer locations and

port/depots are incorporated in the model.

This article extends the modelling of the complicated real-world operations in container transportation industry by combining all container transportation modes (i.e., *Strip* and *Discharge*, loaded and empty, *Street-turn* and *Depot-turn*) in routing optimization with both 20ft and 40ft truck/chassis. Methodologically, the suggested MILP model extends all previous works on the modelling of loaded and empty container flows at customer nodes and inland depots, following the *Strip* and *Discharge* modes. It is also distinct from existing work on homogeneous chassis (e.g. [45]) because simple decisions on whether one has to visit a depot for empty drop-off/pickup between each pair of nodes become unachievable beforehand; instead, these decisions have to be made through the model according to the load of the truck/chassis and the demand/supply type on customer nodes. The implemented GA approach also, differs from existing approaches to solve VRPTW in its chromosome design and feasibility control after crossover and mutation. With this implementation, problems with 500 orders can be solved in about 1 hour of elapsed time.

Note further that in this study, despite having separate categories for *Strip* and *Discharge* containers in the model and solution approaches, we do allow a *Strip* order to be satisfied in the same way as a *Discharge* one, i.e. a truck that carries an empty/loaded container to a customer location, waits there for the container to be loaded/unloaded and then departs with the processed one. In this case both parts of the *Strip* order are performed by the same truck, one right after another, and the waiting time is the same as the processing time of a *Discharge* order. This means the model selects the suitable transportation modes (i.e. *Strip* or *Discharge*) for the so-called *Strip* orders. On top of this, we also include *Discharge* orders which are not split from the truck/chassis for practical considerations.

The structure of this paper is as follows. The relevant literature is surveyed in Section 2. The problem is described in Section 3, followed by the optimization model. The GA is explained in Section 4 and computational results are presented in Section 5. Section 6 is the conclusion and future research.

2 Literature Review

Historically, most works on container inland transportation consider one type of loading/unloading rules out of *Strip* and *Discharge*. We therefore organise the review accordingly. For the *Strip* mode (noted in Table 1 as the Strip column), [42] and [19] study the pickup and delivery problem of homogenous containers based on the M-TSPTW (Multiple Traveling Salesman Problem with Time Windows). The former concentrates on the replacement of time-window constraints in its heuristic, while the latter hybridises DP (Dynamic Programming) and GA to solve large scale problems. [7] extends the previous work by considering heterogeneous types of containers and proposes an insertion heuristic, while [27] addresses the problem as full truckload pickup and delivery problem with time windows (FTPDPTW), which allows the reuse of empty containers based on a 2-stage heuristic. [18] focuses explicitly on the reuse of empty containers in deterministic and dynamic settings. No loaded container movements are considered in this problem. [37] adapts

the initial M-TSPTW with a Tabu Search (TS) heuristic to improve the total operating time of trucks. On the other hand, [2, 3] investigate the same type of problems at multiple planning levels (strategic, tactical and operational). Recently, [36] extends [2, 3] by solving the problem as a asymmetric vehicle routing problem with time window (a-VRPTW) based on an arc-flow formulation.

For slightly different problem set-ups, [41] addresses the transportation of homogenous containers by trucks and trains. A hybrid Tabu Search (TS) method is implemented to solve the problem. [10] as well as [45, 44] examine the delivery and pickup of homogeneous containers under the separation of trucks and trailers ("drop-and-pull"). Both exact and heuristic methods are developed to solve this problem. [43] investigate the transportation of loaded and empty containers. An integer programming model is formulated and solved by GA. Two scenarios are considered by [20] when dealing with the delivery of homogenous containers, i.e. when empty containers are reused by the same owner and when sharing of empty containers is allowed. In [48, 49, 51, 50], the *Strip* of single- and multi-size containers are investigated. Based on the M-TSPTW model, they develop heuristics to obtain solutions for large size instances. Following the assignment models as proposed in [40, 39] and [31], [46] investigates the inland transportation of loaded heterogeneous containers.

Other papers can be found considering the *Discharge* mode (noted in Table 1 as the Discharge column). [13] and [17] study the transportation of full truckloads of homogenous containers. [13] solves it as a pickup and delivery problem with time windows (PDPTW) using four different heuristics. [17] solves the problem as a multi-travelling salesman problem with time windows (M-TSPTW) using a Lagrangian relaxation heuristic. Later, [4] and [33] adapt the model in [17] to study the pre- and end-haulage problem for rail container transportation. [26] and [34] study truck appointment systems at the port terminal to reduce waiting costs and emissions. In the work of [22, 21], the delivery and repositioning of loaded and empty containers with homogenous and heterogeneous fleet of trucks are investigated. [38], [35] consider the transportation of containers where trucks and trailers are separated and stored in different depots. [38] solves the problem as a truck and trailer vehicle routing problem (TTVRP), [35] investigates it as an updated multiple travelling salesman problem with time windows (M-TSPTW). [14] considers the delivery of homogenous containers and proposes an assignment model to solve it, which can be adapted to different practical scenarios. Similarly, [12] addresses a heterogeneous drayage problem with a set-covering model. No more than four orders can be satisfied per route since no depot-turn is allowed.

As shown in Table 1, only six articles have considered both cases of *Strip* and *Discharge*. [16] and [6] consider the situation where trucks (tractors) and chassis (trailers) are located in different depots. [16] ([1]) consider the delivery of homogenous containers, formulate it as a set partitioning model and solve it using column generation (branch-and-price). [6] solves a heterogeneous container case using a GA approach. Our work is different from [16] and [1] as we consider heterogeneous containers, and is different from [6] ([1]) as they only consider *Depot-turn* (*Street-turn*) for empty management while we allow both *Street-turn* and *Depot-turn*. *Street-turn* is a more efficient approach allowing the reuse of

Literature	Strip	Discharge	Type of Containers	Methodology
[42]	✓	x	homogenous	Exact & Heuristic
[13]	x	✓	homogenous	Exact & Heuristic
[19, 18]	✓	x	homogenous	Exact & Heuristic
[38]	x	✓	heterogeneous	Heuristic
[16]	✓	✓	homogenous	Exact
[7]	✓	x	heterogeneous	Exact
[17]	x	✓	homogenous	Exact & Heuristic
[10]	✓	x	homogenous	Exact & Heuristic
[43]	✓	x	heterogeneous	Exact & Heuristic
[26]	x	✓	homogenous	Heuristic
[48, 49]	✓	x	homogenous & heterogeneous	Exact & Heuristic
[51, 50]	✓	x	homogenous & heterogeneous	Exact & Heuristic
[4]	x	✓	homogenous	Heuristic
[6]	✓	✓	heterogeneous	Heuristic
[20]	✓	x	homogenous	Exact
[40, 39]	✓	x	heterogeneous	Exact & Heuristic
[2, 3]	✓	x	homogenous	Exact & Heuristic
[37]	✓	x	homogenous	Exact & Heuristic
[22, 21]	x	✓	heterogeneous	Exact & Heuristic
[41]	✓	x	homogenous	Exact & Heuristic
[27]	✓	x	homogenous	Exact & Heuristic
[31]	✓	x	heterogeneous	Exact & Heuristic
[47]	✓	✓	homogenous	Exact & Heuristic
[45, 44]	✓	x	homogenous	Exact & Heuristic
[33]	x	✓	homogenous	Exact
[11]	✓	✓	homogenous	Exact
[36]	✓	x	homogenous	Exact
[14]	x	✓	homogenous	Exact
[35]	x	✓	homogenous	Exact & Heuristic
[34]	x	✓	homogenous	Exact
[46]	✓	x	heterogeneous	Exact & Heuristic
[12]	x	✓	heterogeneous	Exact
[1]	✓	✓	homogenous	Exact & Heuristic
[24]	✓	✓	heterogeneous	Exact
This paper	✓	✓	heterogeneous	Exact & Heuristic

Table 1: Classification of papers by order mode, container type and solution methodology

empty containers without going through the depot, which has obvious practical significance. While *Depot-turn* is also allowed in this study so as to balance the gaps in demand and supply of empty containers. Another closely related article is [24], whose model allows both *Street-turn* and *Depot-turn* but to a limited extend; *Street-turn* is only considered when a perfectly matched pair of empty requests are raised by the same demander, which actually passed the job of matching *Street-turn* duties to the demander so as to reduce modelling/solution difficulty. Our model, on the other hand, allows an emptied container to be reused by any customer if needed and provides the best match in its solution. In [47], a unified definition of a drayage order is introduced and the drayage problem of homogenous containers is studied. A mixed integer nonlinear programming model based on a determined-activities-on-vertex (DAOV) graph is presented. Later, [11] adapts [47] by formulating an MIP model for the delivery of heterogeneous containers by homogenous fleets (40ft trucks). Only small instances (11 requests) can be solved by this model. In contrast, we consider both 20ft and 40ft chassis for delivery and the GA approach proposed in our work solves instances with up to 500 requests in one hour.

From the methodological point of view, existing methods are not directly applicable to the highly hybrid problem considered in this article. For example, many of the aforementioned works (e.g. [42], [19], [7], [31], [46]) do not distinguish between loaded and empty

containers but consider them in the same fashion as a movement request from one location to another. The approach proposed under this assumption cannot recognise the empty container and therefore ignores the possibility of reusing them. Others who consider empty containers explicitly either forbid the joint movement of empties with loaded ones such as [18], or restrict it only to the *Depot-turn* such as [37], or assume that each truck delivers a single container at a time such as [27], [45], [2, 3], [36], [16], [20], [6], [47], or only consider *Discharge* which is much less flexible such as [22, 21]. All these cases reduce the problem complexity significantly due to the obviously less options allowed in forming a valid route. For example, when only one container is allowed per truck, the connection time (including travel and handling) between each pair of locations can be pre-determined according to the type of requests, because to connect certain types of request (e.g. an Import Empty and an Export Full) a depot has to be inserted to allow the pick-up/drop-off of empty containers. This type of pre-processing cannot be performed in our problem setting due to the larger capacity of 40ft truck/chassis, and therefore need to be captured and decided by the model itself. This, obviously, increases the modelling and solution complexity.

In addition to the above, some articles as listed in Table 1 consider the *Strip* mode where the container/trailer can be removed and separated from the chassis/tractor, while also allowing the truck to wait at the customer location to pickup the same container subject to a longer processing time, if this were beneficial. Such articles include [7] and [45, 44]. Although described differently, our article also covers this type of situation due to the fact that for all *Strip* orders, we allow them to be delivered separately by different trucks but we do not insist on this structure. In other words, they can also be transported in a similar fashion as the *Discharge* orders, subject to a longer waiting time (equivalent to that of the *Discharge* orders). While different from the aforementioned works, our model also includes the orders that cannot be split by definition (*Discharge*). This type of orders includes specialised containers which are not suitable/safe for removing from chassis, requests from customer locations without desired spaces/equipment to keep/remove containers, or share containers which have to be discharged right away and then continue with the second part of the journey. Also, when considering the delivery of 20ft containers on 40ft trailers (chassis), in many countries we cannot pull two separate trailers but only one 40ft trailer which can carry two 20ft containers. In this case, removing the trailer with the container is not doable, but not all clients' sites have the equipment for taking the full container off the trailer. Note that although we didn't state it explicitly, the shared container can also be handled using our model. It can be treated as a standard *Discharge* order, while in the distance matrix we have the "distance-to" element calculated with its first destination (the sequence of destinations is decided by the First-In-Last-Out (FILO) rule on the cargo) and "distance-from" element calculated with the last destination. The service time in this case should include the discharging time at all served nodes, and the travel time in between.

3 Mathematical Formulation

3.1 Problem Statement

We consider a shipping line managing the transportation of containers (loaded/empty) between the port, inland depots and customers (importer/exporter). Two common types of trucks, i.e. 20ft truck which carries a single 20ft container at a time and 40ft truck which carries a single 40ft container, a single or two 20ft containers at a time are selected to perform the delivery, at cost $\rho_1 = 0.35$ and $\rho_2 = 0.5$ pounds per mile, respectively ([9]). Note that in this cost structure we only meant to capture the vehicle-based cost such as fuel, maintenance and insurance etc., rather than the driver-based costs such as wages. All routes are supposed to start from and end at the port; the travelling cost from the home depot of the trucks to the port is ignored. Both *Strip* and *Discharge* containers exist in the transportation request, which can be delivered jointly on the same route if it is profitable.

An *order* in this study is defined as the request of transporting a 20ft or 40ft container, loaded or empty, from its origin to its destination. Following this definition, let's standardise the orders generated from each type of *Strip* and *Discharge* delivery requests. For a *Strip* request, each delivery (pickup) of loaded container is accompanied by a request of removing (drop-off) the empty container that is left-over (demanded) to fulfil the whole process. Precedence constraints are necessary to ensure that these requests are carried out in correct sequence. For a *Discharge* request, the container is not separable from the truck/chassis, which means the empty container that is demanded before (left-over after) service has to be carried by the same truck and the truck has to wait a the customer location for the container to be processed (loaded or unloaded). For all loaded containers both the origin and the destination are known, one of which has to be the port depending on whether it's an import or export order. Whereas for empty orders, only one end of the origin and destination is known depending on it being an empty pickup or empty delivery (detailed instances are given in Table 2). To meet our aim of dealing with all possible situations, orders are put into 12 groups:

- 20IF_Strip, 20IE_Strip, 20EF_Strip, 20EE_Strip;
- 20IF_Discharge, 20EF_Discharge;
- 40IF_Strip, 40IE_Strip, 40EF_Strip, 40EE_Strip;
- 40IF_Discharge, 40EF_Discharge.

where 20 and 40 indicate the container size, IF is for Import-Full, IE for Import-Empty, EF for Export-Full and EE for Export-Empty. Note that there are no xxIE_Discharge, xxEE_Discharge, because the discharge process has to deal with the empty containers right before/after the contents are loaded/removed from the container. So implicit empty movements for discharge orders can be covered by this categorisation. The developed model and methodology will allow the combination of all these types to construct the cost efficient route.

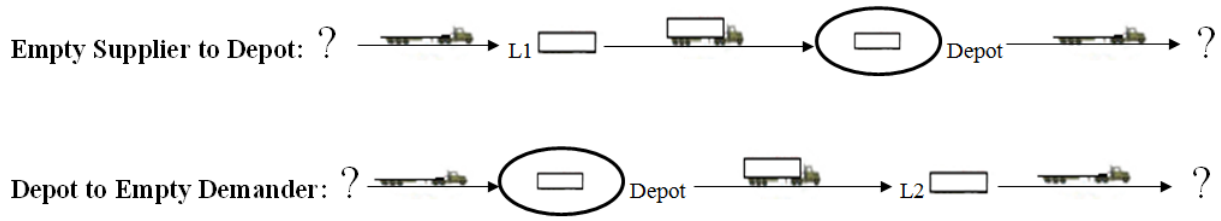


Figure 1: Empty movement with Depot-turn

Figure 2: Empty movement with Street-turn

Given the existence of empty containers, there are two major strategies for storing and relocating them: *Depot-turn* and *Street-turn*. In the *Depot-turn* case (Figure 1), empty containers are stored in the port and/or inland depots. All empty container movements have to originate from or terminate at a depot/port, rather than moved directly from an empty supplier (normally an importer) to an empty demander (normally an exporter). However, in the *Street-turn* case (Figure 2), the direct movement of empty containers between two customer locations is allowed. It is obvious that the *Depot-turn* is easier to manage, since the closest container depot for every customer location can be identified beforehand, so all of the empty movement requests come with a fixed origin and destination. On the other hand, the *Street-turn* case is more efficient, due to the obviously lower number of movements one has to perform. Here, we allow both types of empty movements (*Street-turn* or *Depot-turn*), as long as the delivery route is feasible and cost-saving. Note that in the *Depot-turn* case, balancing the number of empty containers at depots to meet future needs is out of the scope of this research. We simply select the least cost depot to drop/collect the empty container (which might be different from the closest depot from/to the customer who is requesting the empty, depending on the actual route the vehicle is travelling), and assume all empty container depots to have infinite storage capacity.

To illustrate how some orders are dealt with here, we present in Table 2 examples supported by figures. For instance, Order 1 and Order 2 are generated from a single 20ft Import-Full request of the *Strip* type. Order 1 is an 20IF_Strip order, which is to be picked up from the port and delivered to location L1, whereas Order 2 is the subsequent 20IE_Strip order set for the removal of the empty left at L1. As shown in Figure 3 (Order 1), a truck carries the loaded container, travels from the port to location L1 where the container is removed together with its cargo, and leaves to somewhere else to service another order. The cargo is then removed from the container by the customer leaves an empty container at L1 for later collection. To complete the service, Order 2, an 20IE_Strip, is generated to remove the empty container from L1, which has zero payload and no designated destination. It can be relocated to a depot for later usage, or directly to an empty demander to meet

Order index	Type	Origin	Destination	Payload Weight(kg)	Size(ft)	Precedence	Demonstration
1	IF_Strip	Port	L1	13000	20	n/a	Figure 3
2	IE_Strip	L1	NULL	0	20	Order 1	Figure 3
3	IF_Discharge	Port	L2	15000	20	n/a	Figure 4
4	IF_Strip	Port	L1	13500	20	n/a	Figure 5
5	IF_Discharge	Port	L2	15500	20	n/a	Figure 5
6	EE_Strip	NULL	L3	0	20	n/a	Figure 5
7	EF_Discharge	L4	Port	20000	40	n/a	Figure 5

Table 2: Sample list of orders

another request. Note that despite drawing two different routes in Figure 3 to demonstrate the process, Orders 1 and 2 can actually be processed by the same truck, either with other orders in between or directly connected. We don't force *Strip* orders to be satisfied separately if it is not beneficial. Note also that in practice some *Strip* orders may not be accompanied by their second part (e.g. Orders 4 and 6 in Table 2). This might be due to the fact that consecutive jobs are performed on other days, and/or the container is owned by the customer. We will cover examples for both cases in numerical study (Section 5).

In contrast to the *Strip* case, Order 3 is a 20IF_Discharge. A truck carries it from the port to location L2 where its cargo is unloaded, then the same truck moves away carrying the empty container (Figure 4).

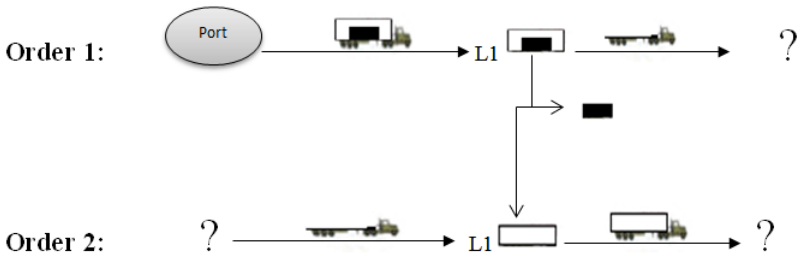


Figure 3: Orders out of a *Strip* request

Figure 4: An example of *Discharge* order

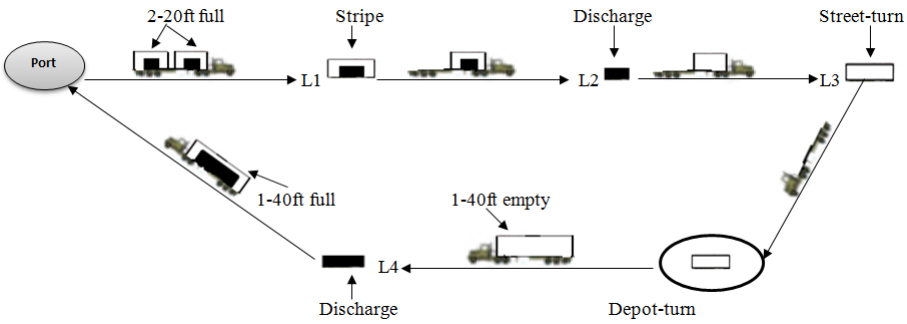


Figure 5: An example of the combined delivery route

As said earlier, here we consider the joint delivery of all types of orders following both the *Depot-turn* and the *Street-turn* empty strategies. An example route is shown in Figure 5, where the truck performs 20IF_Strip (Order 4) and 20IF_Discharge (Order 5) orders to L1 and L2, followed by a *Street-turn* empty (Order 6) movement between L2 and L3, then collects another empty container from an inland depot (*Depot-turn*) to meet an 40EF_Discharge demand (Order 7) at L4. We see that all delivery possibilities are allowed and combined in the same route and by the same truck.

3.2 Notations and Parameters

Based on the problem description, a decision should be made on how to construct the delivery routes to satisfy all demands with minimum cost. The costs considered are travelling costs of the truck which reflect the mileage/fuel cost and the potential penalty for driving hours exceeding the maximum working time regulation. Note that when empty containers exist and can be reused, the number of possible route configurations becomes very large which prevents the usage or extension of the assignment formula, as is done in [40] and [46]. Therefore, we present a new formulation based on network flow, which extends [11] to include heterogeneous fleets and network flow constraints at inland depots.

Let $\mathcal{N}$ be a set of containers which consists of:

- $\mathcal{S}_{20}^{-}$ for 20IF_Strip, $\mathcal{E}_{20}^{+}$ for 20IE_Strip, $\mathcal{S}_{20}^{+}$ for 20EF_Strip, $\mathcal{E}_{20}^{-}$ for 20EE_Strip;
- $\mathcal{D}_{20}^{-}$ for 20IF_Discharge, $\mathcal{D}_{20}^{+}$ for 20EF_Discharge;
- $\mathcal{S}_{40}^{-}$ for 40IF_Strip, $\mathcal{E}_{40}^{+}$ for 40IE_Strip, $\mathcal{S}_{40}^{+}$ for 40EF_Strip, $\mathcal{E}_{40}^{-}$ for 40EE_Strip;
- $\mathcal{D}_{40}^{-}$ for 40IF_Discharge, $\mathcal{D}_{40}^{+}$ for 40EF_Discharge.

Note that the $+$ $(-)$ on the top-right corner of the above definitions means that a container is loaded to (removed from) the carrying truck at the customer location in *Strip* case, or to be filled (emptied) at the customer location in *Discharge* case.

Every order is associated with a customer location denoted by $L_i, i \in \mathcal{N}$. This location indicates the destination for IF and EE orders, and the origin for EF and IE orders. Note

that in practice, some customer locations may have more than one container demand. But for the sake of simplicity, in the mathematical model we allocate every order an unique location indicator L_i despite the fact that some L_is may refer to the same location. We use $\phi(i)$ to denote the precedence order of an order i, which is non-null for xxIE_Strip and xxEF_Strip orders only. Let $\mathcal{N}_d$ denote the set of inland depots including the port (L_0), then the full set of locations is indicated by $\bar{\mathcal{N}} = \mathcal{N} \cup \mathcal{N}_d$. A directed map $\mathcal{A} = \{(L_i, L_j) | \forall i \neq j \in \bar{\mathcal{N}}\}$ is formed with all locations. Let $f(L_i, L_j)$ and $t(L_i, L_j)$ denote the travelling cost and time from location L_i to location L_j, respectively. Let $\mathcal{H}_1$ and $\mathcal{H}_2$ denote the set of 20ft and 40ft trucks, with per-mile cost ρ_1 and ρ_2 respectively.

The handling time at location L_i is denoted by O_i, with

$$O_i = \begin{cases} 0, & \text{if } i \in \mathcal{S}_{20}^- \cup \mathcal{E}_{20}^+ \cup \mathcal{S}_{20}^+ \cup \mathcal{E}_{20}^- \cup \mathcal{S}_{40}^- \cup \mathcal{E}_{40}^+ \cup \mathcal{S}_{40}^+ \cup \mathcal{E}_{40}^- \cup \mathcal{N}_d; \\ h, & \text{if } i \in \mathcal{D}_{20}^- \cup \mathcal{D}_{20}^+ \cup \mathcal{D}_{40}^- \cup \mathcal{D}_{40}^+. \end{cases}$$

Note also that we do allow a *Strip* order to be processed as a *Discharge* one (the same truck performs both the loaded and empty trips and waits at customer location while the container is processed); in this case the processing time is h rather than 0. Let $[TWs_i, TWe_i]$ denote the time window within which order i has to start processing at its customer location. Like many studies in container shipment (e.g. [7]), we assume the problem is planned on a daily basis. The total daily working time of drivers, which includes travelling time, waiting time and handling time, is restricted by the UK domestic working hour regulation (on-duty hours) for drivers with the longest working time per day denoted by T_{max}. If it is violated, a penalty C is incurred per extra working hour per driver.

3.3 Decision Variables

Discrete Variables

- $x_{ijk} = \begin{cases} 1, & \text{if link } (L_i, L_j) \text{ is travelled by truck } k, \\ 0, & \text{otherwise} \end{cases} \quad \forall i \neq j \in \bar{\mathcal{N}}, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2.$
- $y_{ijk}^1, \forall i \neq j \in \bar{\mathcal{N}}, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2$: the number of 20ft containers on truck k when travelling on link (L_i, L_j).
- $y_{ijk}^2, \forall i \neq j \in \bar{\mathcal{N}}, \forall k \in \mathcal{H}_2$: the number of 40ft containers on truck k when travelling on link (L_i, L_j).
- $z_{ijk}^1, \forall i \neq j \in \bar{\mathcal{N}}, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2$: the number of loaded 20ft containers on truck k when travelling on link (L_i, L_j).
- $z_{ijk}^2, \forall i \neq j \in \bar{\mathcal{N}}, \forall k \in \mathcal{H}_2$: the number of loaded 40ft containers on truck k when travelling on link (L_i, L_j).
- $\gamma_{ijk}^1, \forall i \neq j \in \bar{\mathcal{N}}, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2$: the number of 20ft EF (export full) containers on truck k when travelling on link (L_i, L_j).

- $\gamma^2_{ijk}, \forall i \neq j \in \bar{\mathcal{N}}, \forall k \in \mathcal{H}_2$: the number of 40ft EF (export full) containers on truck k when travelling on link (L_i, L_j).

Continuous Variables

- $w_k, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2$: extra working hours beyond T_{max} for driver on truck k.
- $\tau_i, \forall i \in \mathcal{N}$: visiting time (service start time) at location L_i.
- $\tau_{ik}, \forall i \in \mathcal{N}_d \setminus \{L_0\}, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2$: visiting time at depot i by truck k.
- $s_k, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2$: departure time of truck k from the port terminal.
- $e_k, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2$: finishing time of truck k at the port terminal.

3.4 Formulation

The following Mixed Integer Linear Programming (MILP) model is formulated then solved to find the best route to travel by all trucks.

$$\min \sum_{i\in\bar{\mathcal{N}}} \sum_{j\in\bar{\mathcal{N}},j\neq i} \sum_{k\in\mathcal{H}_1} \rho_1 f(L_i,L_j) x_{ijk} + \sum_{i\in\bar{\mathcal{N}}} \sum_{j\in\bar{\mathcal{N}},j\neq i} \sum_{k\in\mathcal{H}_2} \rho_2 f(L_i,L_j) x_{ijk} + \sum_{k\in\mathcal{H}_1\cup\mathcal{H}_2} C w_k \quad (3.1)$$

$$s.t. \sum_{j\in\bar{\mathcal{N}},j\neq i} x_{ijk} = \sum_{j\in\bar{\mathcal{N}},j\neq i} x_{jik}, \qquad \forall i \in \bar{\mathcal{N}} \setminus \{L_0\}, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2 \quad (3.2)$$

$$\sum_{j\in\mathcal{N}\cup\{L_0\}} x_{0jk} = 1, \qquad \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2 \quad (3.3)$$

$$\sum_{j\in\mathcal{N}\cup\{L_0\}} x_{j0k} = 1, \qquad \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2 \quad (3.4)$$

$$\sum_{j\in\bar{\mathcal{N}},j\neq i} \sum_{k\in\mathcal{H}_1\cup\mathcal{H}_2} x_{ijk} = 1, \qquad \forall i \in \mathcal{N} \quad (3.5)$$

$$\sum_{j\in\bar{\mathcal{N}},j\neq i} \sum_{k\in\mathcal{H}_1} x_{ijk} = 0, \qquad \forall i \in \mathcal{S}^-_{40} \cup \mathcal{E}^+_{40} \cup \mathcal{S}^+_{40} \cup \mathcal{E}^-_{40} \cup \mathcal{D}^-_{40} \cup \mathcal{D}^+_{40} \quad (3.6)$$

$$y^1_{ijk} \leq x_{ijk}, \qquad \forall i \neq j \in \bar{\mathcal{N}}, \forall k \in \mathcal{H}_1 \quad (3.7)$$

$$y^1_{ijk} \leq 2x_{ijk}, \qquad \forall i \neq j \in \bar{\mathcal{N}}, \forall k \in \mathcal{H}_2 \quad (3.8)$$

$$y^2_{ijk} \leq x_{ijk}, \qquad \forall i \neq j \in \bar{\mathcal{N}}, \forall k \in \mathcal{H}_2 \quad (3.9)$$

$$z^1_{ijk} \leq y^1_{ijk}, \qquad \forall i \neq j \in \bar{\mathcal{N}}, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2 \quad (3.10)$$

$$z^2_{ijk} \leq y^2_{ijk}, \qquad \forall i \neq j \in \bar{\mathcal{N}}, \forall k \in \mathcal{H}_2 \quad (3.11)$$

$$\gamma^1_{ijk} \leq z^1_{ijk}, \qquad \forall i \neq j \in \bar{\mathcal{N}}, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2 \quad (3.12)$$

$$\gamma^2_{ijk} \leq z^2_{ijk}, \qquad \forall i \neq j \in \bar{\mathcal{N}}, \forall k \in \mathcal{H}_2 \quad (3.13)$$

$$y^1_{ijk} + 2y^2_{ijk} \leq 2, \qquad \forall i \neq j \in \bar{\mathcal{N}}, \forall k \in \mathcal{H}_2 \quad (3.14)$$

$$\sum_{j\in\bar{\mathcal{N}},j\neq i}\sum_{k\in\mathcal{H}_1\cup\mathcal{H}_2} y^1_{ijk} - \sum_{j\in\bar{\mathcal{N}},j\neq i}\sum_{k\in\mathcal{H}_1\cup\mathcal{H}_2} y^1_{jik} = r_1, \qquad \forall i \in \mathcal{N} \tag{3.15}$$

$$\sum_{j\in\bar{\mathcal{N}},j\neq i}\sum_{k\in\mathcal{H}_1\cup\mathcal{H}_2} z^1_{ijk} - \sum_{j\in\bar{\mathcal{N}},j\neq i}\sum_{k\in\mathcal{H}_1\cup\mathcal{H}_2} z^1_{jik} = r_2, \qquad \forall i \in \mathcal{N} \tag{3.16}$$

$$\sum_{j\in\bar{\mathcal{N}},j\neq i}\sum_{k\in\mathcal{H}_1\cup\mathcal{H}_2} \gamma^1_{ijk} - \sum_{j\in\bar{\mathcal{N}},j\neq i}\sum_{k\in\mathcal{H}_1\cup\mathcal{H}_2} \gamma^1_{jik} = r_3, \qquad \forall i \in \mathcal{N} \tag{3.17}$$

$$\sum_{j\in\bar{\mathcal{N}},j\neq i}\sum_{k\in\mathcal{H}_2} y^2_{ijk} - \sum_{j\in\bar{\mathcal{N}},j\neq i}\sum_{k\in\mathcal{H}_2} y^2_{jik} = r_4, \qquad \forall i \in \mathcal{N} \tag{3.18}$$

$$\sum_{j\in\bar{\mathcal{N}},j\neq i}\sum_{k\in\mathcal{H}_2} z^2_{ijk} - \sum_{j\in\bar{\mathcal{N}},j\neq i}\sum_{k\in\mathcal{H}_2} z^2_{jik} = r_5, \qquad \forall i \in \mathcal{N} \tag{3.19}$$

$$\sum_{j\in\bar{\mathcal{N}},j\neq i}\sum_{k\in\mathcal{H}_2} \gamma^2_{ijk} - \sum_{j\in\bar{\mathcal{N}},j\neq i}\sum_{k\in\mathcal{H}_2} \gamma^2_{jik} = r_6, \qquad \forall i \in \mathcal{N} \tag{3.20}$$

$$\sum_{j\in\mathcal{N},j\neq i}\sum_{k\in\mathcal{H}_1\cup\mathcal{H}_2} z^1_{ijk} - \sum_{j\in\mathcal{N},j\neq i}\sum_{k\in\mathcal{H}_1\cup\mathcal{H}_2} z^1_{jik} = 0, \qquad \forall i \in \mathcal{N}_d \setminus \{L_0\} \tag{3.21}$$

$$\sum_{j\in\mathcal{N},j\neq i}\sum_{k\in\mathcal{H}_1\cup\mathcal{H}_2} \gamma^1_{ijk} - \sum_{j\in\mathcal{N},j\neq i}\sum_{k\in\mathcal{H}_1\cup\mathcal{H}_2} \gamma^1_{jik} = 0, \qquad \forall i \in \mathcal{N}_d \setminus \{L_0\} \tag{3.22}$$

$$\sum_{j\in\mathcal{N},j\neq i}\sum_{k\in\mathcal{H}_2} z^2_{ijk} - \sum_{j\in\mathcal{N},j\neq i}\sum_{k\in\mathcal{H}_2} z^2_{jik} = 0, \qquad \forall i \in \mathcal{N}_d \setminus \{L_0\} \tag{3.23}$$

$$\sum_{j\in\mathcal{N},j\neq i}\sum_{k\in\mathcal{H}_2} \gamma^2_{ijk} - \sum_{j\in\mathcal{N},j\neq i}\sum_{k\in\mathcal{H}_2} \gamma^2_{jik} = 0, \qquad \forall i \in \mathcal{N}_d \setminus \{L_0\} \tag{3.24}$$

$$\tau_j \geq \tau_i + t(L_i, L_j) + O_i - M(1 - \sum_{k\in\mathcal{H}_1\cup\mathcal{H}_2} x_{ijk}), \qquad \forall i \neq j \in \mathcal{N} \tag{3.25}$$

$$\tau_{jk} \geq \tau_i + t(L_i, L_j) + O_i - M(1 - x_{ijk}), \qquad \forall i \in \mathcal{N}, \forall j \in \mathcal{N}_d \setminus \{L_0\}, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2 \tag{3.26}$$

$$\tau_j \geq \tau_{ik} + t(L_i, L_j) + O_i - M(1 - x_{ijk}), \qquad \forall i \in \mathcal{N}_d \setminus \{L_0\}, \forall j \in \mathcal{N}, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2 \tag{3.27}$$

$$TWs_i \leq \tau_i \leq TWe_i, \qquad \forall i \in \mathcal{N} \tag{3.28}$$

$$s_k \leq \tau_i - t(L_0, L_i) + M(1 - x_{0ik}), \qquad \forall i \in \mathcal{N}, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2 \tag{3.29}$$

$$e_k \geq \tau_i + t(L_i, L_0) + O_i - M(1 - x_{i0k}), \qquad \forall i \in \mathcal{N}, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2 \tag{3.30}$$

$$w_k \geq e_k - s_k - T_{max}, \qquad \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2 \tag{3.31}$$

$$\tau_i \geq \tau_{\phi(i)} + O_{\phi(i)}, \qquad \forall i \in \mathcal{E}^+_{20} \cup \mathcal{S}^+_{20} \cup \mathcal{E}^+_{40} \cup \mathcal{S}^+_{40} \tag{3.32}$$

$$x_{ijk} \in \{0, 1\}, \qquad \forall i \neq j \in \bar{\mathcal{N}}, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2 \tag{3.33}$$

$$y^1_{ijk}, z^1_{ijk}, \gamma^1_{ijk}, \in \{0, 1, 2\}, \qquad \forall i \neq j \in \bar{\mathcal{N}}, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2 \tag{3.34}$$

$$y^2_{ijk}, z^2_{ijk}, \gamma^2_{ijk}, \in \{0, 1\}, \qquad \forall i \neq j \in \bar{\mathcal{N}}, \forall k \in \mathcal{H}_2 \tag{3.35}$$

$$w_k \geq 0, \qquad \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2 \tag{3.36}$$

$$\tau_i \geq 0, \forall i \in \mathcal{N} \tag{3.37}$$

$$\tau_{ik} \geq 0, \forall i \in \mathcal{N}_d \setminus \{L_0\}, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2 \tag{3.38}$$

$$s_k, e_k \geq 0, \forall k \in \mathcal{H}_1 \cup \mathcal{H}_2 \tag{3.39}$$

Constraints (3.2-3.4) are the standard network flow constraints, which enforce that all trucks start/finish at the port (L_0) and travel through all customer nodes. Note that the port and the depot are all considered as empty container storages so the direct travel from the port to depot or vice versa is not allowed. While we assume the number of trucks to be more than what is needed, the option x_{00k} is available to allow the surplus trucks to stay at the port. Constraint (3.5) guarantees that all orders are serviced by exactly one truck. Constraint (3.6) enforces that no 40ft orders are serviced by 20ft trucks. Constraints (3.7-3.9) assigns the maximum number of containers on the routes travelled by 20ft trucks (3.7) and 40ft trucks (3.8 for 20ft containers and 3.9 for 40ft containers). Note that having 40ft containers on 20ft trucks is prohibited by the variable y, z and γ variables and these constraints. Constraint (3.7) also implicitly imposes the capacity limit of 20ft trucks, whereas for 40ft trucks the capacity limit is enforced by constraint (3.14). Constraints (3.10-3.13) ensure the number of EF (Export Full) containers transported on every route is bounded by the number of loaded containers, which is further bounded by the number of containers on the route. Constraints (3.15-3.20) share the same structures for all containers but the right hand sides depend on the container category. In detail, the vector $(r_1, r_2, r_3, r_4, r_5, r_6)$ indicates the difference between the outflow and the inflow of the containers of the corresponding variable type, which is $(-1, -1, 0, 0, 0, 0)$ for $\mathcal{S}^-_{20}$, $(1, 0, 0, 0, 0, 0)$ for $\mathcal{E}^+_{20}$, $(1, 1, 1, 0, 0, 0)$ for $\mathcal{S}^+_{20}$, $(-1, 0, 0, 0, 0, 0)$ for $\mathcal{E}^-_{20}$, $(0, 0, 0, -1, -1, 0)$ for $\mathcal{S}^-_{40}$, $(0, 0, 0, 1, 0, 0)$ for $\mathcal{E}^+_{40}$, $(0, 0, 0, 1, 1, 1)$ for $\mathcal{S}^+_{40}$, $(0, 0, 0, -1, 0, 0)$ for $\mathcal{E}^-_{40}$, $(0, -1, 0, 0, 0, 0)$ for $\mathcal{D}^-_{20}$, $(0, 1, 1, 0, 0, 0)$ for $\mathcal{D}^+_{20}$, $(0, 0, 0, 0, -1, 0)$ for $\mathcal{D}^-_{40}$ and $(0, 0, 0, 0, 1, 1)$ for $\mathcal{D}^+_{40}$. Constraints (3.21-3.24) are for inland depots, which reflect the fact that the latter can be used for the collection and drop-off of empty containers but the number of loaded containers remains the same after visiting these inland depots. Constraints (3.25-3.27) are enforced to calculate the start service time (arrival time) at customer nodes. Note that the definition of arrival time for different node types are different. For customer nodes, we only need to visit each by one truck so the visiting time is not indexed by trucks, while for inland depots they might be visited by different trucks at different times, therefore we define a visiting time for each truck in order to capture the full information. The customer time windows are imposed by constraint (3.28). The start and finish times of truck k at the port are calculated by constraints (3.29) and (3.30). The violation of the maximum working hours is then calculated in (3.31) and penalised in the objective. Parameter M in constraints (3.25) - (3.30) reflects the latest finishing time of a truck route, which can be defined as the end of the day since we make the delivery plan on a daily basis. (3.32) are precedence constraints.

Despite this MILP model being designed to consider all potential container types, transportation modes and empty movement strategies, it can also be applied to various problem settings. For example, one specific category of containers can be easily removed by setting the corresponding set to empty; the usage of inland depots can be prohibited by removing the set of depots together with constraints (3.21-3.24). This means the formulation can also be used to solve problems with *Strip*-only or *Discharge*-only orders, problems containing only 20ft or 40ft containers, problems with or without empty containers, problems with or without inland depots, etc. Note, however, that in each case the proposed model is still

very complicated. Indeed, the container transportation problem with only one category of containers is NP-hard [31]. As it is not likely to find exact solutions to the model in reasonable times, in the next section we design a solution methodology based on the Genetic Algorithm (GA) to solve large scale problem instances.

4 Genetic Algorithm Approach

The idea of GA is due to [15]. It emulates approximately the process of biological evolution. GA has been used widely to solve difficult combinatorial optimization problems ([8], [32], [23]). To implement GA, an initial population of solutions is randomly created reflecting the nature of the problem. These solutions are then evaluated for fitness usually using the objective function and the results of certain feasibility checks. Then the genetic operators (crossover, mutation and reproduction) are applied to create offspring of the selected solutions in the current generation to populate the next generation, based on their fitness values. The process continues until some stopping criteria are satisfied, of these is often the maximum number of generations.

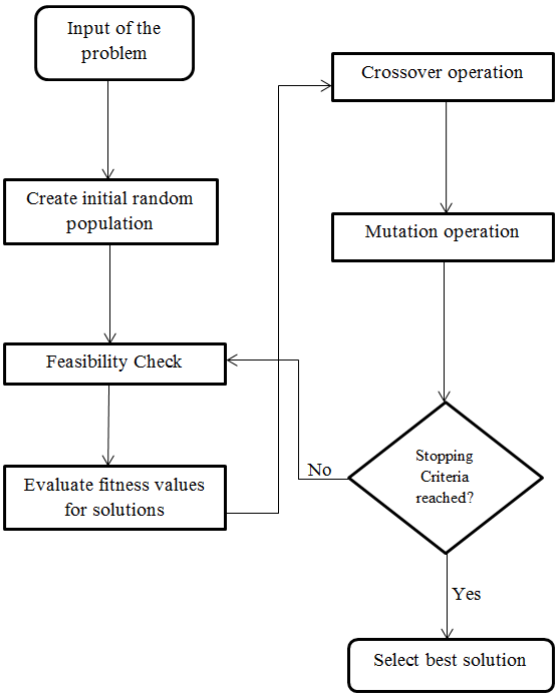


Figure 6: Generic flowchart of GA

A generic representation of the deployed GA approach is shown in Figure 6. The explicit components of our implementation of GA for solving the above model are described in more details below.

4.1 Chromosome representation of the problem

Traditionally, in transportation (vehicle routing) problems, a chromosome represents the final routes that each vehicle travels. However, this approach is not suitable here because of the very limited capacity (up to 2 containers per truck at a time) which can be easily violated when applying crossover and mutation operators. Also, since *Street-turn* and *Depot-turn* are both allowed for empty containers, it is hard to predict the length of a route. Therefore, in this study, we propose a different approach for chromosome definition. We use the chromosome to represent the sequence following which orders are to be considered in the "sequential insertion" method as described in Algorithm 1, below. In other words, the chromosome here can be represented by a permutation of all orders, without identifying which order is to be serviced by which truck. Therefore the chromosome has the same length as the number of orders. However, this chromosome representation omits many vital information such as the specific route travelled by each truck, the arrival time of each order, etc. We deal with these aspects in the "sequential insertion" method as proposed in Algorithm 1. This algorithm converts chromosomes into executable routes, which are feasible in terms of vehicle capacity, but may violate some others such as the fleet size, precedence and time window constraints (which will be penalized later in the fitness function).

In detail, the "sequential insertion" method constructs full delivery routes by inserting depots (D) and port (P) into the chromosome. Based on the current capacity usage of the considered truck, it checks if the next order can be satisfied with the remaining capacity (or availability of empty containers) on this truck. If yes then it moves one digit forward, updates the capacity usage and checks the next order. If not then it checks whether the infeasibility is caused by empty containers (on the truck or yet to be collected to meet an empty delivery request). If yes then it finds the nearest inland depot to drop/collect empty container(s), otherwise (which means there are too many loaded containers to be delivered to the port) it inserts the port to finish the route of this truck and activates the next truck to do the subsequent tasks. Note that when we use this insertion algorithm the truck size is always assumed to be 40ft. Every time a port is inserted, this means a full truck route is obtained, the actual truck size is then known based on the maximum capacity that is needed to execute the route.

We illustrate this insertion process on a 12 order scenario. The scenario involves two depots (D1, D2), a port (P) and 4 trucks ($2 \times$ 20ft, $2 \times$ 40ft). As shown in Figure 7(A), a chromosome is a random permutation of all orders. To make it an executable delivery plan, we insert the port (P) and depots (D1, D2) into the sequence so as to separate the chromosome into sub sequences, each representing a feasible route for a truck to travel (Figure 7(B)). More explicitly, the first truck (Truck1) starts from the port (P), taking a 20ft loaded *Strip* container to deliver to location $L1$, then travels to location $L3$ to collect a 20ft empty container. Since the next task is to deliver a 40ft empty container to location $L7$ and there is no 40ft empty containers on the truck but only a 20ft empty, the truck visits Depot (D1) to drop the 20ft empty on it and collect a 40ft empty for the following order. This "check and insert" process continues until accommodating order 6, when a

Algorithm 1 - Sequential insertion method

Step 0: Let n denote the total number of orders. Let s denote a random permutation of order indices (chromosome). Set digit counter $i = 1$.

Step 1: Indicate truck usage by two variables and initiate them to zero:

- Initial_load records the number of loaded and empty containers when truck depart from the depot;
- Current_load records the number of loaded and empty containers after servicing the order under consideration.

Step 2: ***while*** $i \leq n$

Step 2.1: Find the type of order i.

Step 2.2: According to the type update the Initial_load and Current_load of truck usage parameters.

if the updated Initial_load and Current_load is not violating the capacity of the truck (40ft by default) then set $i = i + 1$ and go to Step 2;

elseif the updated Current_load is negative – the order s_i is an empty delivery, there is empty space on the truck without empty containers (or the empty container size doesn't match):

- Find the nearest inland depot (D) to order s_i, insert it before order s_i;
- Collect the empty container that is needed by order s_i from the depot (and/or drop the empty container that is not needed by order s_i) – update Current_load;
- Service order s_i – update Current_load;
- Set $i = i + 1$ and go to Step 2.

elseif the updated Current_load violates the capacity of the truck but there are empty containers on the truck:

- Find the nearest inland depot (D) to order s_i, insert it before order s_i;
- Drop all empty containers at the depot – update Current_load;
- Service order s_i – update Current_load;
- Set $i = i + 1$ and go to Step 2.

else the updated Current_load violates the capacity of the truck but there is no empty containers on the truck: go to Step 3.

Step 3: Insert port (P) before s_i – terminate route for the current truck and go to Step 1.

40ft loaded container that occupies the full capacity of the truck is collected and to be delivered to the port (where a route finishes). Since the next order is a 40ft import full,

inserting of which will definitely violate the capacity of the truck, the port (P) is inserted to terminate the route of this truck. A new truck (route) is then considered to service the rest of orders in the chromosome. The process is repeated until all orders are examined, following the sequence stated by the chromosome. In the end, when we have inserted all the port and depots we can then examine each route to see which type of truck (20ft or 40ft) is needed to perform it so as to know the composition of the fleet required. In this specific example, we need three 40ft and one 20ft trucks to perform all tasks.

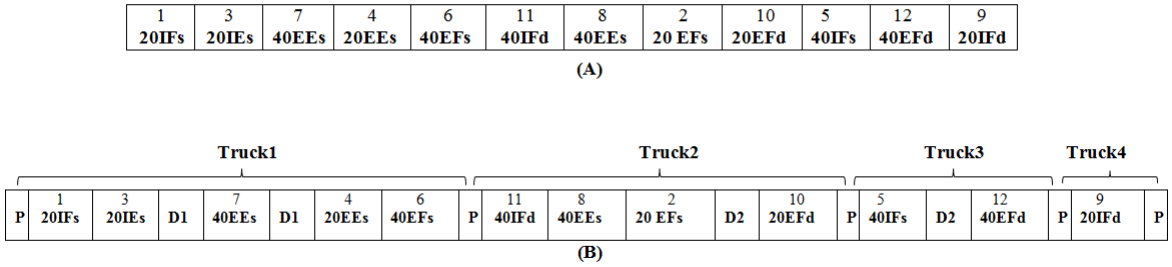


Figure 7: Sample chromosome representation and the executable delivery route after performing Algorithm 1

Note that the proposed "sequential insertion" method is just a heuristic which cannot guarantee optimality even when the chromosome sequence is derived from the optimal solution. Nevertheless, it is significant for two reasons:

1. It always tries to use the maximum capacity of the truck which is an important feature of the optimal solution.

2. It provides a route to turn chromosomes into feasible solutions which enables the GA approach to explore larger feasible areas within a limited number of iterations.

Traditional chromosome representations for VRP (VRPTW) concentrate largely on distance minimisation, while in container shipping capacity constraints become more vital since the maximum number that can be carried by a truck is just 2. This requires an approach which strives to maintain feasibility, and the "sequential insertion" heuristic is meant to do this.

It is obvious that we cannot foresee the number of trucks we are going to use to execute all routes following the sequence as given in each chromosome. Sometimes we may need less trucks than what are available but sometimes more. However, the infeasibility that can be introduced by a chromosome also boils down to the number of trucks used and the specified time window, rather than creating completely un-executable deliver plans as if we include a certain number of routes/trucks and depot visiting options in the chromosome itself. A large number of infeasible solutions will slow down the convergence process of the GA, that is why we try to avoid infeasibility by running this insertion process to convert a sequence into executable solutions. Figure 7 demonstrates one example of infeasible solution due

to the number/type of trucks needed; three 40ft and one 20ft trucks are needed by this schedule while two of each type are available according to the initial example setting. This infeasibility will be penalised in the fitness function together with the violation of time windows and precedence constraints.

4.2 Constructing the initial population

To construct the initial population for GA, a number of chromosomes are generated by random permutation of orders. As mentioned before, Algorithm 1 converts the random permutation into executable routes. Note that we still denote the order permutation (Figure 7(A)) as a chromosome, rather than the extended executable delivery plan (Figure 7(B)). So, although the number of trucks used and the number of nodes each truck visits are unpredictable, all chromosomes still have the same length.

4.3 Fitness function and evaluation

The fitness function evaluates the quality of chromosomes so as to rank them for future GA operations. The objective of this problem is to minimize the total travelling cost and working time violation penalty to satisfy all orders. Nevertheless, the quality of a chromosome also depends on its feasibility. As said before, infeasible ones can be expected after inserting port and depots, on the number of trucks used, the violation of time windows and precedence constraints. For this reason, in the fitness function we add terms representing the penalty on these terms. We define the fitness function for chromosome i as:

$$Fit(i) = OBJ(i) + PENALTY(i) \tag{4.1}$$

$$= [TTC(i) + WTP(i)] + [PT20(i) + PT40(i) + PTW(i) + PTP(i)] \tag{4.2}$$

where $OBJ(i)$ is the original objective function which comprises $TTC(i)$, the total travelling cost of solution i and $WTP(i)$, the total penalty cost for violating the maximum allowed daily working time. $PENALTY(i)$ includes the penalty for violating the fleet size constraints with $PT20(i)$ for 20ft trucks and $PT40(i)$ for 40ft trucks, that for violating the time window constraints represented by $PTW(i)$, and that for violating the precedence constraints represented by $PTP(i)$. Note that the per unit penalty for extra trucks are set to very large values in the numerical experiment so as to avoid the violation of these constraints. In practice, one can tune this parameter according to needs, such as setting the penalty to the actual cost of renting additional trucks and extra drivers to perform the additional route generated by the GA. Note further that in our numerical experiment there isn't any case in which a planned route takes longer than the maximum allowed working time. This rarely happens in practice if we suppose the cost and penalty parameters to be set to reflect the real situation. However to avoid extreme cases we can also penalize a chromosome whose working time exceeds an upper bound, e.g. 15 hours where $T_{max} = 11$ hours. By doing this the fitness value will drive the solution away from planning very long routes.

4.4 Selection process

The new population of chromosomes/solutions is created by performing certain recombination processes such as the crossover and mutation. An appropriate selection process should be applied to choose parents. In this study, we apply the Roulette Wheel Selection (RWS) approach [30], which select parents according to their proportional performance; individuals with a smaller fitness values have a higher probability to be selected to generate new offspring.

4.5 Genetic operators

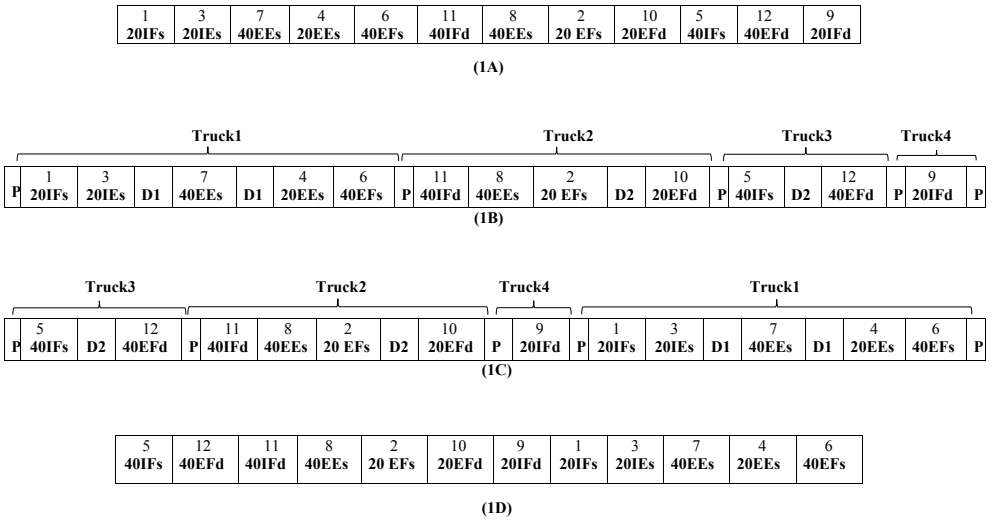
After selecting solutions as parents from the initial population, a new population of offspring (children) can be constructed. Genetic operators, i.e. crossover, mutation and reproduction, are used to create the new population as described later.

4.5.1 Crossover

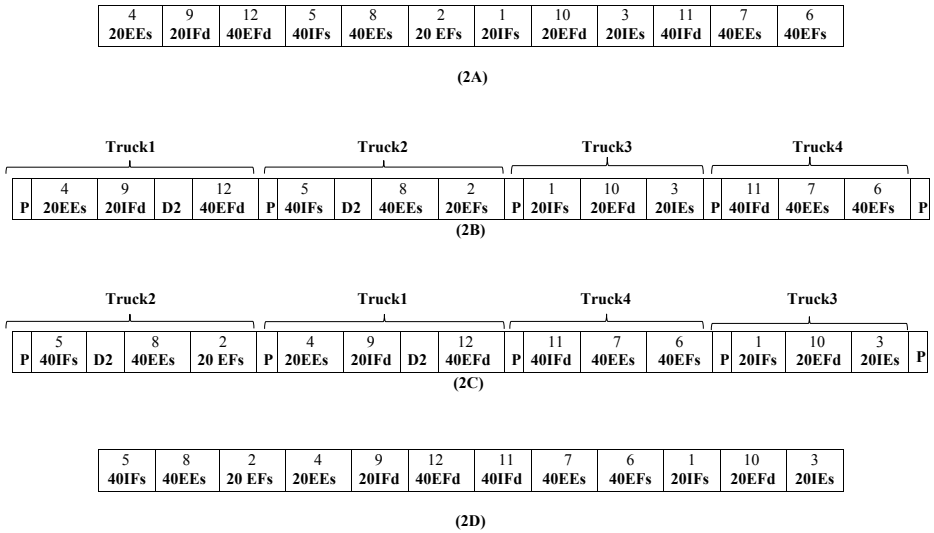
The "subschedule preservation crossover", which was designed by [5] as a permutation based operator, is applied to construct new offspring. In this crossover operator a random number of digits are chosen from the beginning of the first parent to start a new child. To complete the first child, the remaining digits are added following the sequence they are seen in the second parent, without repetition. To create the second child the same process is carried out, but starting from the second parent.

Figure 8 and 9 illustrates the crossover process with two sample chromosomes (parents). Firstly, we do a random permutation of the truck routes generated by the "sequential insertion" method before applying the crossover operator, so that any truck route has a chance to be in the front part of the chromosome. These steps are illustrated by Figure (1A-1D) for parent 1 and (2A-2D) for parent 2. Since our "sequential insertion" method checks orders from left to right, the crossover operator keeps the route structure for trucks that are servicing the orders before the crossover point. However, good genes may exist either to the left or to the right of the crossover point. This pre-processing will allow the information carried by the right piece of parent chromosomes to pass down to their children.

Once the permutation is done, we apply the crossover operator to the resulting chromosome (1D) and (2D). Like what is shown in Figure 9, we select a random digit where the chromosome is split into parts, e.g. 4. To generate Child 1, the first four digits (Orders 5, 12, 11 and 8) are copied from Parent 1. Removing these orders from Parent 2 we obtain the remaining part of Child 1, then we connect them together to form Child 1 as shown in Figure 9(3A). In the same way, Child 2 can be created with the same parents (Figure 9(3B)). The full truck routes generated after applying the "sequential insertion" method again are displayed in Figure 9(4A) and Figure 9(4B).



(a) Parent 1



(b) Parent 2

Figure 8: Random permutation of truck routes before applying the crossover operator

4.5.2 Mutation

In the mutation process, two orders are selected randomly from the same chromosome which is also chosen randomly from the current population. The two orders are then swapped to obtain a new chromosome. As illustrated in Figure 10(A), Order 7 and Order 1 are selected randomly from a chromosome of 9 orders. Then the two selected orders are exchanged to generate a new solution as given in Figure 10(B). When the two selected orders belong to the same category, this mutation operator will not change any other truck routes except the one/two servicing the swapped orders, since the "sequential insertion"

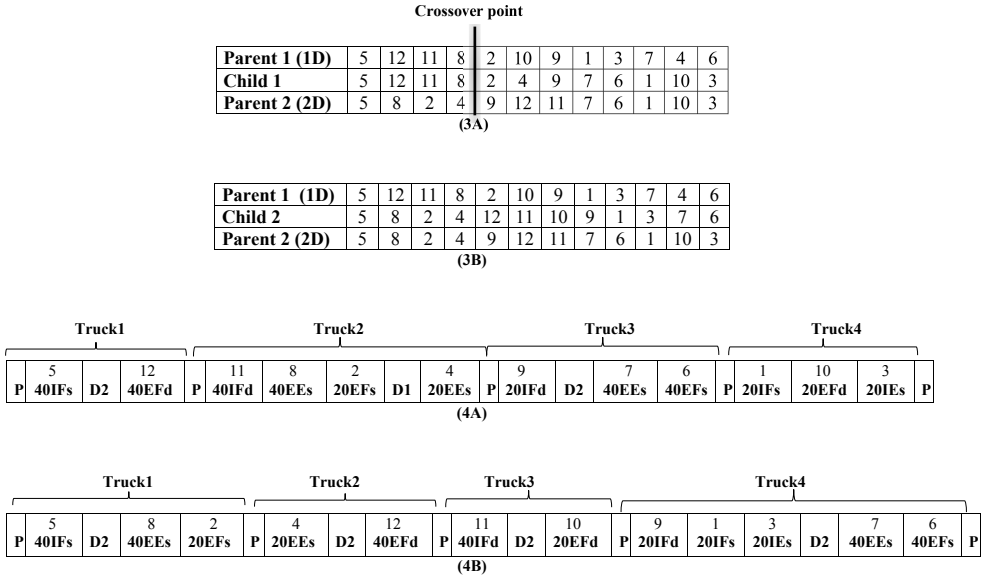


Figure 9: Crossover of orders

method only considers container types when splitting the chromosome array into executable routes. When the two selected orders to swap belong to different categories, however, the mutation may trigger partial updates of the executable route for all the remaining trucks because some orders might be moved from one truck to another by running the "sequential insertion" method. This provides a higher chance of getting rid of a local optimum.

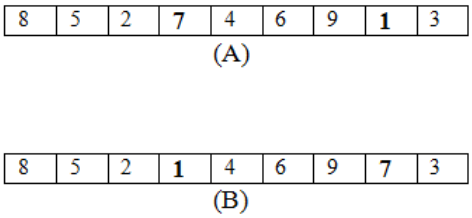


Figure 10: Mutation of orders

4.5.3 Reproduction

This is the operator that allows good solutions to be passed into the new population unchanged. In this study we rank all parents and children after crossover and mutation and select the best performing ones to remain.

4.6 Stopping criterion

The above procedures of our GA are repeated until the stopping criterion is satisfied. For this problem, the stopping criterion used is the maximum number of generations.

In numerical experiments we inspect the decrease in fitness values with the number of generations, to decide how many generations is needed by the algorithm to converge for every specific instance.

Likewise to the MILP, the proposed GA can also be applied to different scenarios, e.g. problems with *Strip*-only or *Discharge*-only orders, problems containing only 20ft or 40ft containers, problems with or without empty containers, problems with or without inland depots, etc. The population and genetic operators are both independent of the type of containers/empty strategies under consideration; the only difference is seen in the "sequential insertion" method (Algorithm 1) which converts chromosomes into executable routes. For example, when *Depot-turn* is disabled in a specific problem setting, the two "*elseif*" options in Step 2.2 of Algorithm 1 should be removed.

5 Computational experiments

All numerical experiments are carried out on real data obtained from the Port of Felixstowe (PoF), which is one of the largest container ports in the UK. A one-day request list consisting of 1067 orders from/to 346 customer locations are used as the source dataset from which our samples (in various sizes) are generated. The service area considered in the dataset spread over a pre-defined area that can be covered within 10 hours of driving from the port. The original dataset consists of full information about the order destinations, payload weights, assigned time windows and order types. The explicit location of every customer node is not available due to disclosure concerns, while road distances between each pair of locations are known. From this dataset, we randomly pick orders according to the sample size and split *Strip* orders into two parts as described in Section 3.1. A shipping line, who own a number of 20ft and 40ft trucks is assumed to fulfil the transportation of orders between the port terminal, customers (importers and exporters) and a number of inland depots.

A penalty cost (200 pounds/h) applies for any extra working hours (more than 11 hours per day), according to the UK/EU driving regulation. We assume that the service time at *Strip* locations is negligible [45] while at *Discharge* locations it is 2 hours. The average speed of trucks is estimated at 40 miles/h. Both solution approaches (MILP, GA) are coded in MATLAB R2016b and executed on a CPU with an Intel(R)Core(TM)i7-4790 processor. The MILP is solved by CPLEX 12.9. For the GA, each example is solved multiple times (depending on the size of the problem) to eliminate the influence of the randomness. Other parameters of GA, such as the population sizes, number of generations, crossover and mutation rates, etc. are selected specifically for every example.

5.1 MILP and GA on small and medium scale examples

In this section we test the performance of the developed GA approach by comparing its solution with the optimal/best possible solution obtained by solving the MILP using CPLEX. Due to the size limits that MATLAB/CPLEX can deal with, we firstly run experiments

with five orders and then increase the number of orders by five, until MATLAB/CPLEX run out of memory and/or the solution time becomes unbearably long. As said earlier, despite MILP and GA being both developed for the highly hybrid scenario where all kinds of transportation types and modes exist, the resulting model and methodology can also be used to cover simpler cases. To complement the tests, we run experiments in categories according to their problem settings. In Table 3, BLC stands for "balanced number of loaded and empty orders from *Strip* requests" whereas UBLC stands for "unbalanced number of loaded and empty orders from *Strip* requests". In practice, containers could be owned by customers which means a loaded request may not be accompanied by an empty one for every single *Strip* order. Therefore we proposed the UBLC category to cover the scenario where there are fewer empty requests than loaded ones. Both BLC and UBLC allow the usage of inland depots for empty turn around and/or temporary storage, while another category, STT, is proposed for "Street-turn only" where using inland depot is banned. Finally, category STP stands for "*Strip*-only" and DSC stands for "*Discharge*-only", where only the relevant type of containers are considered. The largest size instance for each category displayed in Table 3 are the largest problem, that can be solved by CPLEX to optimality. The sizes are different due to the different size and difficulty of the test scenarios.

Results are summarised in Table 3. For each instance, the problem index indicates the number of initial transportation requests while the second column shows the total number of orders after splitting *Strip* requests into loaded and empty ones. The number of orders for each container type are summarised in column number 3-6, in the sequence of (IF, EF, IE, EE) for *Strip* and (IF, EF) for *Discharge*. In all testing examples except the STT_x cases (Street Turn), there are four inland depots whose location reflect the real location of the inland depots used by the PoF. The GA results displayed are the average over 20 runs; the averaged cost, standard deviation, the number of runs when GA hit the optimal solution (last column) are reported.

Note that in this study, the number of trucks (routes) to perform the delivery is also selected by the optimization model/approach so the initial fleet number is not a decisive parameter. In all experiments, we set the fleet size by simply assuming that each 40ft container needs a 40ft truck and every two 20ft container require a 20ft truck, since otherwise the surplus constraints such as the time window can hardly be satisfied in general.

The solution time of the MILP using CPLEX depends largely on the specific example while on average it increases with problem size. It is obvious to see that the STT (Street-turn only) and DSC (Discharge-only) cases are relatively easier to solve by CPLEX than other scenarios, mainly due to the smaller number of variables. While for the GA approach, there is no significant differences; a sub-optimal solution can be achieved in a couple of seconds for all instances. The designed GA approach is capable of finding optimal solutions for all small instances across the test scenarios, and its sub-optimality gap is not monotonically increasing with the problem size. Note that there isn't any case throughout the tests that GA fails to find a feasible solution. There are certain instances where CPLEX finds optimal solution relatively fast but GA get stuck in a local optimum (e.g. STT_20), while instances also exist on which CPLEX takes longer to search through the Branch-and-Bound tree whereas GA approaches the optimal solution in 12/20 runs within

Index	# Orders	Number & Type of orders				MILP solution				GA solution (average)						
		# 20ft	# 40ft	# 20ft	# 40ft	# trucks		cost	time (s)	# trucks		cost	s.d.	time (s)	opt. gap (%)	# opt.
		Strip	Strip	Disch.	Disch.	20ft	40ft			20ft	40ft					
BLC_5	9	(1,0,1,0)	(1,2,1,2)	(0,0)	(0,1)	0	3	502.7	2.13	0.0	3.0	503.1	1.95	0.17	**0.08**	18
BLC_10	17	(1,3,1,3)	(1,2,1,2)	(1,0)	(1,1)	3	3	988.0	15.65	1.6	4.2	1022.5	36.21	0.83	**3.49**	5
BLC_15	23	(0,4,0,4)	(1,3,1,3)	(1,1)	(2,3)	5	6	1605.2	215.90	3.2	7.9	1624.9	25.13	1.53	**1.23**	7
BLC_20	33	(2,1,2,1)	(8,2,8,2)	(2,0)	(3,2)	4	11	2336.3	305.13	1.3	12.9	2460.2	48.94	3.19	**5.30**	0
UBLC_5	6	(2,0,1,0)	(0,0,0,0)	(0,0)	(1,2)	1	2	616.6	0.76	1.0	2.0	616.6	0.00	0.16	**0.00**	20
UBLC_10	12	(0,0,0,0)	(0,4,0,2)	(3,1)	(2,0)	1	4	1041.0	6.27	1.0	4.0	1041.0	0.00	0.21	**0.00**	20
STT_5	8	(1,1,1,1)	(0,1,0,1)	(1,0)	(1,0)	2	1	446.9	1.48	2.0	1.0	446.9	0.00	0.11	**0.00**	20
STT_10	13	(0,1,0,1)	(1,1,1,1)	(2,3)	(1,1)	4	2	849.4	1.95	4.0	2.0	850.4	4.38	0.17	**0.12**	19
STT_15	25	(3,1,3,1)	(1,5,1,5)	(0,1)	(4,0)	3	5	1185.4	51.84	0.9	6.5	1257.3	32.80	3.03	**6.07**	0
STT_20	31	(3,0,3,0)	(2,6,2,6)	(1,2)	(3,3)	4	9	1877.4	156.66	1.5	11.9	2091.0	76.27	4.74	**11.38**	0
STT_25	28	(2,1,2,0)	(3,6,0,1)	(2,3)	(5,3)	5	10	2407.4	3094.85	3.5	11.6	2416.7	15.64	0.98	**0.39**	12
STT_30	39	(4,3,1,2)	(5,4,3,3)	(4,4)	(2,4)	9	9	2560.6	1427.73	4.2	12.7	2745.9	74.96	2.37	**7.59**	0
STT_35	42	(4,2,1,2)	(6,3,3,1)	(8,3)	(5,4)	10	13	3546.8	1220.63	5.2	16.5	3669.8	72.80	2.52	**3.47**	1
STP_5	9	(0,3,0,2)	(2,0,2,0)	-	-	1	2	368.5	1.83	1.0	2.0	373.2	8.03	0.18	**1.28**	13
STP_10	16	(1,1,1,1)	(3,5,1,3)	-	-	1	5	965.4	1761.78	1.0	5.0	977.9	10.45	0.65	**1.29**	0
DSC_5	5	-	-	(1,2)	(0,2)	2	2	571.6	1.52	2.0	2.0	571.6	0.00	0.12	**0.00**	20
DSC_10	10	-	-	(1,2)	(5,2)	3	3	799.6	1.17	3.0	3.0	799.6	0.00	0.15	**0.00**	20
DSC_15	15	-	-	(4,5)	(5,1)	4	5	1330.8	5.78	3.2	6.0	1332.3	4.45	0.27	**0.11**	15
DSC_20	20	-	-	(5,4)	(7,4)	5	7	1868.6	27.27	4.2	9.0	1869.4	1.97	0.21	**0.04**	17
DSC_25	25	-	-	(2,7)	(6,10)	5	12	2360.5	304.88	4.6	12.4	2411.3	29.29	0.23	**2.15**	1
DSC_30	30	-	-	(7,11)	(7,5)	8	8	2242.8	293.05	7.4	10.9	2246.6	11.02	1.01	**0.17**	17
DSC_35	35	-	-	(9,14)	(5,7)	12	9	2900.6	506.00	8.8	11.9	3009.0	77.07	1.50	**3.74**	2
DSC_40	40	-	-	(4,9)	(13,14)	7	16	3245.7	396.72	5.6	17.4	3377.8	31.40	2.92	**4.07**	0
DSC_45	45	-	-	(10,12)	(13,10)	11	12	3801.5	1284.86	9.9	14.9	3855.7	52.38	5.33	**1.43**	4

Table 3: Comparison of MILP and GA solutions on small scale examples

only 1 second (e.g. STT_25). Note that the number of used trucks is fractional in the GA solution as reported in the table, since we run the GA for 20 times and take the average. On average, GA tends to use more 40ft trucks than the optimal solution, because the insertion heuristic assumes 40ft trucks are used by default and tries to place as many orders on it as possible. This could be one of the main reason for the sub-optimality gap. In addition, GA performs notably well on *Discharge*-only instances (DSC_x), no matter whether the numbers of import and export containers are balanced or not.

CPLEX is based on Branch-and-Bound which gives intermediate, feasible integer solutions although sometimes it cannot prove their optimality within reasonable time. So on slightly larger instances where MATLAB/CPLEX runs without memory issues, we run another set of tests to compare the GA solutions with the best feasible integer solutions or the tightest LP bound (lower bound for any feasible solutions) that CPLEX can find within limited time. Results are summarised in Table 4. To allow a fair comparison while avoiding CPLEX running indefinitely, for each instance we report the CPLEX solutions after 1,000 seconds and after one hour.

As shown in Table 4, when CPLEX finds feasible integer solutions (not necessarily optimal), GA finds better or close solutions in most instances such as BLC_25, UBLC_15, UBLC_20, STP_15 and STP_20. In other cases when GA gives slightly higher gaps (3% - 13%) such as for BLC_30, UBLC_30 and DSC_50, the sub-optimality gaps of the CPLEX solution are close to zero which means they might be the optimal solution but CPLEX needs longer than 1 hour to prove their optimality. However, the running time for GA is just a few seconds compared to the CPLEX running time of 1,000 or 3,600 seconds.

Due to the large size of instances, CPLEX can't even find a feasible integer solution in one hour of CPU time, for some of them. When CPLEX fails to find any feasible integer

Index	# Orders	Number & Type of orders				MILP solution					GA solution (average)						
		# 20ft Strip	# 40ft Strip	# 20ft Disch.	# 40ft Disch.	# trucks 20ft	# trucks 40ft	cost	time (s)	gap (%)	# trucks 20ft	# trucks 40ft	cost	s.d.	time (s)	gap from CPLEX (%)	# better than CPLEX
BLC_25	36	(3,4,3,4)	(1,3,1,3)	(3,4)	(3,4)	10	7	2118.7	1000	8.45	6.2	8.9	2131.2	50.27	6.64	**0.59**	6
						8	7	2080.0	3600	4.92						**2.46**	4
BLC_30	46	(3,5,3,5)	(4,4,4,4)	(6,2)	(2,4)	-	-	-	1000	-	3.6	12.1	2701.2	66.50	7.39	-	-
						7	8	2386.4	3600	0.05						**13.19**	0
BLC_35	49	(2,6,2,6)	(3,3,3,3)	(4,4)	(9,4)	Best LP bound after 3600s: 2726.8					5.5	15.8	3078.5	79.53	13.13	< **12.90**	-
UBLC_15	18	(1,2,1,0)	(2,4,0,2)	(2,2)	(0,2)	2	7	1409.1	1000	11.55	0.3	7.9	1399.7	13.84	1.37	**-0.67**	13
						2	7	1409.1	3600	11.55						**-0.67**	13
UBLC_20	23	(2,0,2,0)	(1,5,0,3)	(2,1)	(3,4)	1	10	2070.7	1000	8.51	1.0	10.1	2049.9	26.53	1.34	**-1.00**	16
						1	10	2049.8	3600	7.56						**0.00**	8
UBLC_25	29	(2,5,1,2)	(3,4,0,1)	(3,1)	(4,3)	5	8	2043.4	1000	10.09	2.6	10.5	2040.4	29.07	4.13	**-0.15**	11
						5	8	1929.8	3600	3.35						**5.73**	0
UBLC_30	35	(5,4,2,3)	(3,1,0,0)	(2,8)	(5,2)	8	8	2485.2	1000	1.55	4.3	10.9	2599.6	41.89	4.47	**4.60**	0
						8	8	2477.0	3600	0.16						**4.95**	0
UBLC_35	45	(3,7,1,1)	(7,5,5,3)	(2,2)	(4,5)	Best LP bound after 3600s: 2369.6					4.1	15.9	2778.6	61.98	9.50	< **17.26**	-
STT_40	52	(4,8,4,3)	(5,4,4,1)	(8,3)	(4,4)	Best LP bound after 3600s: 2840.1					5.1	16.4	3128.3	61.58	10.21	< **10.15**	-
STP_15	20	(2,4,0,1)	(4,5,3,1)	-	-	1	8	1302.2	1000	23.64	2.8	6.7	1203.0	20.53	2.60	**-7.62**	20
						4	5	1265.5	3600	19.58						**-4.94**	20
STP_20	29	(1,2,1,0)	(8,9,5,3)	-	-	0	11	2124.8	1000	10.85	0.7	9.8	2114.3	33.92	6.50	**-0.49**	12
						0	10	2061.4	3600	7.85						**2.57**	0
STP_25	39	(8,6,4,3)	(4,7,3,4)	-	-	Best LP bound after 3600s: 2189.1					3.9	10.4	2410.6	34.61	12.12	< **10.12**	-
STP_30	47	(8,9,6,3)	(6,7,3,5)	-	-	Best LP bound after 3600s: 2116.1					4.0	13.3	2850.9	40.63	12.31	< **34.72**	-
DSC_50	50	-	-	(7,11)	(20,12)	-	-	-	1000	-	5.5	23.8	4160.2	59.04	4.41	-	-
						9	20	4031.6	3600	0.17						**3.19**	0

Table 4: Comparison of MILP and GA solutions on medium scale examples

solutions, the best LP bound obtained after one hour is reported. Note that this LP bound is a theoretical lower bound which cannot be achieved by any feasible integer solution, and its tightness varies from instance to instance - in some cases the optimal objective could be much higher than the LP bound. While considering the difficulty of the problem it is unlikely to develop tighter bounds for these instances, so we compare the GA solution with the LP bound to show the maximum possible sub-optimality gap the GA solution generates, while bearing in mind that the actual gap might be much smaller than the values shown in these cases in Table 4. Despite this, the sub-optimality gaps of GA solution in all but one case are smaller than 18%, which justifies its performance on larger instances. The outlier of 34.72% for STP_30 could be due to the poor LP bound, as its value (2116.1) is even lower than that of the STP_25 (2189.1) with obviously less number of orders.

5.2 GA on large scale instances

Due to the NP-hard nature of the problem, large instances of the MILP model cannot be solved to optimality. In this section we report on the performance of GA on various instances with 50, 100, 200, 300, 400 and 500 orders. Again all orders are randomly generated from actual customer locations serviced by the Port of Felixstowe (PoF), the largest container port in the UK. Both *Depot-turn* and *Street-turn* are allowed in these tests. GA results are the average of 10 runs. To find how many iterations are actually needed for GA to converge, we inspect the decrease in the lowest fitness values as shown

in Figure 11 for the BLC_50 and UBLC_50. The final results are summarised in Table 5.

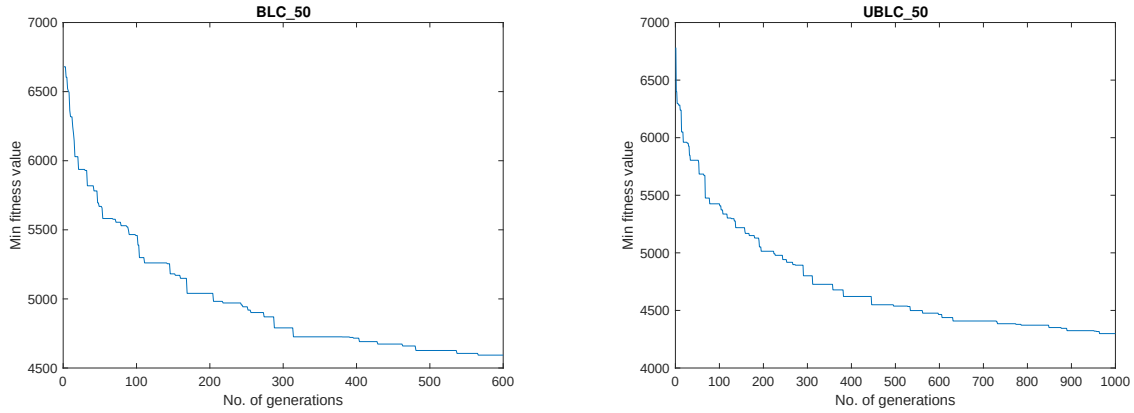


Figure 11: Convergence and number of iterations for the GA approach

Index	# Orders	Number & Type of orders				GA solution (average)				
		# 20ft Strip	# 40ft Strip	# 20ft Disch.	# 40ft Disch.	# fleet used 20ft	# fleet used 40ft	cost	s.d.	time (s)
BLC_50	76	(6,7,6,7)	(7,6,7,6)	(6,8)	(3,7)	2.1	21.5	4535.7	92.93	28.19
UBLC_50	67	(8,6,3,6)	(4,8,2,6)	(6,6)	(5,7)	2.1	20.6	4354.9	80.27	23.84
BLC_100	149	(10,12,10,12)	(12,15,12,15)	(8,16)	(10,17)	6.0	45.9	9380.6	107.31	94.30
UBLC_100	123	(11,9,3,5)	(11,17,6,9)	(8,15)	(12,17)	5.1	46.6	8382.9	48.20	90.28
BLC_200	298	(20,26,20,26)	(19,33,19,33)	(26,22)	(30,24)	10.8	92.9	21112.0	205.30	279.07
UBLC_200	248	(20,27,10,9)	(29,22,17,12)	(27,18)	(31,26)	12.4	85.2	17575.0	138.95	260.14
BLC_300	458	(32,41,32,41)	(53,32,53,32)	(27,41)	(42,32)	18.4	137.9	31255.0	230.64	845.52
UBLC_300	373	(24,37,14,19)	(45,44,23,17)	(27,36)	(41,46)	17.4	133.2	27216.0	179.48	679.83
BLC_400	593	(45,48,45,48)	(46,54,46,54)	(43,52)	(46,66)	22.8	188.8	40994.0	230.18	1481.80
UBLC_400	519	(46,62,26,27)	(51,57,28,38)	(32,37)	(66,49)	25.1	176.7	37769.0	210.32	1331.88
BLC_500	741	(42,54,42,54)	(64,81,64,81)	(64,67)	(55,73)	28.2	224.0	48178.0	257.58	2015.31
UBLC_500	628	(57,47,30,27)	(69,66,38,33)	(42,55)	(87,77)	31.3	223.3	47278.0	239.15	2009.33

Table 5: Results of solving larger examples by GA alone

Looking at the results we can see that the unbalanced problems are smaller in size so they end up with shorter solution times and smaller costs. While the number of iterations that are needed for the GA to converge in each category are not significantly different. With the GA method, examples with 500 orders can be solved within one hour.

5.3 The value of combining *Strip* and *Discharge* orders

Here, we investigate how much can be saved by combining *Strip* and *Discharge* orders. To this end, we use GA to solve three problem settings: *Strip* only, *Discharge* only, *Strip* and *Discharge* jointly with the same set of orders as used in the separate case. Both *Depot-turn* and *Street-turn* are allowed under all these three settings. In Table 6 we summarize the results for instances with 100, 200, 300, 400 and 500 orders, solved separately and jointly. Some of them have balanced order numbers in each category while some others have randomly allocated numbers. Results are the average of 10 runs with GA. Note that the full result for every example consists of four rows in Table 6: the first two rows show

the solution of solving *Strip* only and *Discharge* only problems, respectively; the third row is a summation of the first two rows which show the total cost and resource consumption for delivering these two types separately; the fourth row shows the solution of joint delivery of both *Strip* and *Discharge* orders. So the fleet usage and cost in the third row and the fourth row are comparable.

Index	# Orders	Number & Type of orders				GA solution (average)					% saving from separate
		# 20ft	# 40ft	# 20ft	# 40ft	# fleet used		cost	s.d.	time (s)	
		Strip	Strip	Disch.	Disch.	20ft	40ft				
STP_46	92	(10,12,10,12)	(10,14,10,14)	-	-	2.1	21.5	4563.3	23.62	27.01	-
DSC_54	54	-	-	(9,8)	(16,21)	1.5	24.3	5191.1	98.03	11.23	-
SPR_100	146	(10,12,10,12)	(10,14,10,14)	(9,8)	(16,21)	3.6	45.8	9754.4	121.65	38.24	-
JIT_100	146	(10,12,10,12)	(10,14,10,14)	(9,8)	(16,21)	4.3	43.6	9090.6	78.45	92.13	**6.81**
STP_98	144	(18,22,7,11)	(30,28,12,16)	-	-	5.2	43.7	8534.3	46.94	89.22	-
DSC_102	102	-	-	(15,30)	(19,38)	4.6	50.7	9261.1	85.02	37.82	-
SPR_200	246	(18,22,7,11)	(30,28,12,16)	(15,30)	(19,38)	9.8	94.4	17795.4	115.41	127.04	-
JIT_200	246	(18,22,7,11)	(30,28,12,16)	(15,30)	(19,38)	14.5	86.4	16432.0	94.82	297.38	**7.66**
STP_142	216	(40,34,25,15)	(30,38,14,20)	-	-	6.7	60.2	13259.0	134.32	124.74	-
DSC_158	158	-	-	(29,46)	(40,43)	10.3	64.3	13887.0	173.74	56.33	-
SPR_300	374	(40,34,25,15)	(30,38,14,20)	(29,46)	(40,43)	17.0	124.5	27146.0	308.06	181.07	-
JIN_300	374	(40,34,25,15)	(30,38,14,20)	(29,46)	(40,43)	21.8	119.8	24855.0	166.16	755.03	**8.44**
STP_201	402	(36,54,36,54)	(65,46,65,46)	-	-	15.4	83.5	20695.0	122.91	320.04	-
DSC_199	199	-	-	(36,48)	(64,51)	12.2	108.1	22103.0	308.56	74.90	-
SPR_400	601	(36,54,36,54)	(65,46,65,46)	(36,48)	(64,51)	27.6	191.6	42798.0	431.47	394.94	-
JIN_400	601	(36,54,36,54)	(65,46,65,46)	(36,48)	(64,51)	28.3	181.0	38422.0	264.44	1515.39	**10.22**
STP_252	504	(56,63,56,63)	(68,65,68,65)	-	-	17.4	97.6	23939.0	182.19	458.07	-
DSC_248	248	-	-	(65,58)	(64,61)	19.9	139.1	29886.0	316.88	195.82	-
SPR_500	752	(56,63,56,63)	(68,65,68,65)	(65,58)	(64,61)	37.3	236.7	54825.0	499.07	653.89	-
JIN_500	752	(56,63,56,63)	(68,65,68,65)	(65,58)	(64,61)	31.8	223.5	48872.0	219.89	2672.62	**10.86**

Table 6: Comparison of transporting *Strip* and *Discharge* orders separately or jointly

As illustrated in Table 6, the delivery cost is reduced by $6.81 - 10.86\%$ when combining *Strip* and *Discharge* orders in transportation is allowed. This cost reduction becomes more significant as the problem size grows. Compared to separate delivery, joint delivery tends to use more 20ft trucks whereas less 40ft ones, which is actually a feature of the optimal solution we have observed from solving small scale problems. On average, the *Strip*-only problem is harder to solve than the *Discharge*-only one with the same (or similar) number of initial delivery requests as it leads to larger problem size, whereas the cost of the *Strip*-only problem is in general cheaper due to the flexibility it offers in transportation. The computational time to perform the joint case is larger (but still acceptable) compared to the two individual cases, since more iterations are needed for the GA to converge.

5.4 Value of the inland depots

In this section we evaluate the value of using the inland depots in container transportation. To this end, we solve the same problem with or without inland depots, for the scenarios with roughly balanced (BLC_XX) or significantly unbalanced (UBLCE_XX) number of empty requests. Note that the "without inland depot" case is indicated by an extension "_STT" in Table 7, standing for "street-turn only". While the usage of port terminal as an empty storage is still allowed in the "_STT" case because all routes start and finish at the port terminal, by default.

Index	# Orders	Number and Type of orders				GA solution (average)					% Gap
		# 20ft Strip	# 40ft Strip	# 20ft Disch.	# 40ft Disch.	# fleet used 20ft	40ft	cost	s.d.	time (s)	
BLC_50_STT	76	(6,7,6,7)	(7,6,7,6)	(6,8)	(3,7)	3.8	21.8	4546.4	95.94	15.03	**2.06**
BLC_50						1.7	21.3	4452.8	66.49	15.44	
UBLCE_50_STT	74	(7,2,7,2)	(13,2,13,2)	(5,4)	(14,3)	2.8	31.7	5445.4	67.23	22.25	**2.78**
UBLCE_50						2.3	31.9	5294.1	39.86	23.51	
BLC_100_STT	161	(16,11,16,11)	(17,17,17,17)	(8,8)	(11,12)	8.6	51.5	10935.0	147.33	104.82	**7.87**
BLC_100						4.6	46.0	10074.0	125.00	100.09	
UBLCE_100_STT	145	(9,4,9,4)	(25,7,25,7)	(16,5)	(25,9)	8.1	61.4	11950.0	146.98	103.81	**8.28**
UBLCE_100						6.0	56.7	10961.0	100.22	105.98	
BLC_200_STT	290	(16,20,16,20)	(24,30,24,30)	(21,26)	(27,36)	20.4	101.3	21291.0	134.14	351.87	**7.58**
BLC_200						11.2	93.0	19678.0	130.65	295.63	
UBLCE_200_STT	299	(30,13,30,13)	(45,11,45,11)	(32,13)	(49,7)	16.8	118.6	24667.0	162.49	265.15	**9.47**
UBLCE_200						13.6	119.5	22331.0	159.83	263.64	
BLC_300_STT	463	(30,36,30,36)	(53,44,53,44)	(20,29)	(48,40)	22.0	157.6	34453.0	252.00	655.03	**4.08**
BLC_300						17.1	146.8	33049.0	141.00	663.35	
UBLCE_300_STT	463	(48,14,48,14)	(68,33,68,33)	(35,15)	(66,21)	18.0	171.8	37302.0	250.75	965.58	**8.10**
UBLCE_300						18.5	169.8	34282.0	209.03	928.37	
BLC_400_STT	595	(45,46,45,46)	(45,59,45,59)	(35,58)	(56,56)	26.0	190.2	42663.0	314.84	1587.08	**2.45**
BLC_400						23.6	186.6	41619.0	220.62	1591.10	
UBLCE_400_STT	588	(66,18,66,18)	(67,37,67,37)	(67,25)	(93,27)	25.5	220.1	45653.0	357.19	1651.05	**5.02**
UBLCE_400						24.5	218.2	43363.0	191.44	1677.83	
BLC_500_STT	750	(63,64,63,64)	(61,62,61,62)	(41,62)	(63,84)	36.3	244.3	51561.0	445.86	2709.45	**1.98**
BLC_500						32.8	238.6	50540.0	441.43	2683.35	
UBLCE_500_STT	762	(87,24,87,24)	(122,29,122,29)	(66,35)	(90,47)	43.9	278.5	60975.0	244.00	2791.29	**4.60**
UBLCE_500						34.1	276.8	58169.0	319.36	2933.26	

Table 7: Comparison between the Street-and-Depot-turn and Street-turn-only cases

As shown in Table 7, the cost decreases by 1.98 - 9.47% for the inclusion of inland depots, which becomes more significant when the numbers of import and export orders, and therefore the demand and supply of the empty containers, are obviously unbalanced. The reduced cost is obtained from the better routes constructed when inland depots are used to temporarily store the empties and to balance the demands and supply of empties, and a higher truck utilization.

6 Conclusions

In this paper the most general form of the inland container transportation problem is solved. Explicitly, we consider the transportation of 12 types of orders involving, 20ft and 40ft, loaded and empty, *Strip* and *Discharge* containers using a heterogeneous fleet of 20ft and 40ft trucks. The two major strategies for storing and relocating empty containers, namely *Street-turn* and *Depot-turn* are both allowed. An MILP model is designed to represent this problem. Small instances of this problem can be solved to optimality. For industrial sized instances we suggest a novel implementation of GA which uses a sequential insertion method to convert chromosome/solutions to executable delivery routes, so as to maintain the viability of the routes with both crossover and mutation operators. This methodology works well and introduces an average sub-optimality gap of 2.21% on all tested small scale instances. We also proposed a route for the combined delivery of *Strip* and *Discharge* containers. These two types are investigated separately and jointly in route planning, to show how much can be saved by servicing them simultaneously. The result shows that by combining them in delivery one can save up to 11% in the total delivery costs.

Note that we assume orders are fixed and known in advance, while in practice orders are received dynamically covering multiple days. It is worthwhile to investigate this situation.

References

[1] Federica Bomboi. *New Routing Problems with possibly correlated travel times*. PhD thesis, Universitá degli Studi di Cagliari, January 2020.

[2] K. Braekers, A. Caris, and G.K. Janssens. Integrated planning of loaded and empty container movements. *OR Spectrum*, 35(2):457–478, 2013.

[3] K. Braekers, A. Caris, and G.K. Janssens. Bi-objective optimization of drayage operations in the service area of intermodal terminals. *Transportation Research Part E: Logistics and Transportation Review*, 65:50–69, 2014.

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

[5] R. Cheng and Gen M. Parallel machine scheduling problems using memetic algorithms. *Computers & Industrial Engineering*, 33(3-4):761–764, 1997.

[6] H. R. Choi, B. K. Park, J. W. Lee, and C. H. Park. Dispatching of container trucks using genetic algorithm. *Interaction Sciences (ICIS), 2011 4th International Conference*, pages 146–151, 2011.

[7] K. H. Chung, C. S. Ko, J. Y. Shin, H. Hwang, and K. H. Kim. Development of mathematical models for the container road transportation in korean trucking industries. *Computers & Industrial Engineering*, 53(2):252–262, 2007.

[8] L. Davis. *Handbook of genetic algorithms*. VNR computer library. Van Nostrand Reinhold, 1991.

[9] RTS Financial. How to calculate cost per mile for your trucking company. *https://www.rtsinc.com/guides/trucking-calculations-formulas*, Tuesday, Apr 16 2019.

[10] P. Francis, G. Zhang, and K. Smilowitz. Improved modeling and solution methods for the multi-resource routing problem. *European Journal of Operational Research*, 180(3):1045–1059, 2007.

[11] J. Funke and H. Kopfer. A model for a multi-size inland container transportation problem. *Transportation Research Part E: Logistics and Transportation Review*, 89:70–85, 2016.

[12] A. Ghezelsoflu, M. Di Francesco, A. Frangioni, and P. Zuddas. A set-covering formulation for a drayage problem with single and double container loads. *Journal of Industrial Engineering International*, 14(4):665–676, 2018.

[13] M. Gronalt, RF. Hartl, and M. Reimann. New savings based algorithms for time constrained pickup and delivery of full truckloads. *European Journal of Operational Research*, 151(3):520–535, 2003.

[14] A. D. Hajem, X. Yang, and M. K. Warnes. An efficient mixed integer programming model for pairing containers in inland transportation based on the assignment of orders. *Journal of the Operational Research Society*, 68(6):678–694, 2017.

[15] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[16] Y. Ileri, M. Bazaraa, T. Gifford, G. Nemhauser, J. Sokol, and E. Wikum. An optimization approach for planning daily drayage operations. *Central European Journal of Operations Research*, 14(2):141–156, 2006.

[17] A. Imai, E. Nishimura, and J. Current. A lagrangian relaxation based heuristic for the vehicle routing with full container load. *European Journal of Operational Research*, 176(1):87–105, 2007.

[18] H. Jula, A. Chassiakos, and P. Ioannou. Port dynamic empty container reuse. *Transportation Research Part E: Logistics and Transportation Review*, 42(1):43–60, 2006.

[19] H. Jula, M. Dessouky, P. Ioannou, and A. Chassiakos. Container movement by trucks in metropolitan networks: modeling and optimization. *Transportation Research Part E: Logistics and Transportation Review*, 41(3):235–259, 2005.

[20] H. Kopfer, J. Schönberger, S. Sterzik, and R. Y. Zhang. Optimization of inland container transportation with and without container sharing. *Proceedings of the 1st International Conference on Logistics and Maritime Systems*, 2010.

[21] M. Lai, M. Battarra, M. Di Francesco, and P. Zuddas. An adaptive guidance metaheuristic for the vehicle routing problem with splits and clustered backhauls. *Journal of the Operational Research society*, 66(7):1222–1235, 2015.

[22] M. Lai, TG. Crainic, M. Di Francesco, and P. Zuddas. An heuristic search for the routing of heterogeneous trucks with single and double container loads. *Transportation Research Part E: Logistics and Transportation Review*, 56:108–118, 2013.

[23] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998.

[24] M. Moghaddam, R.H. Pearce, H. Mokhtar, and C.G. Prato. A generalised model for container drayage operations with heterogeneous fleet, multicontainer sizes and two modes of operation. *Transportation Research Part E: Logistics and Transportation Review*, 139, 2020.

[25] P. Nagl. Longer combination vehicles (lcv) for asia and the pacific region: some economic implications. *United Nations Publications*, 2007.

[26] R. Namboothiri and AL. Erera. Planning local container drayage operations given a port access appointment system. *Transportation Research Part E: Logistics and Transportation Review*, 44:185–202, 2008.

[27] J. Nossack and E. Pesch. A truck scheduling problem arising in intermodal container transportation. *European Journal of Operational Research*, 230(3):666–680, 2013.

[28] T. Notteboom and W. Winkelmans. Factual report on the european port sector 2004-2005. *Report commissioned by European Sea Ports Organisation (ESPO)*, 2004.

[29] T. E. Notteboom and J-P. Rodrigue. Port regionalization: towards a new phase in port development. *Maritime Policy & Management*, 32(3):297–313, 2005.

[30] H. M. Pandey. Performance evaluation of selection methods of genetic algorithm and network security concerns. *Procedia Computer Science*, 78:13–18, 2016.

[31] D. Popović, M. Vidović, and M. Nikolić. The variable neighborhood search heuristic for the containers drayage problem with time windows. *Soft Computing in Industrial Applications*, pages 351–364, 2014.

[32] C.R. Reeves. Genetic algorithms for the operations researcher. *INFORMS journal on computing*, 9(3):231–250, 1997.

[33] L. B. Reinhardt, D. Pisinger, S. Spoorendonk, and Sigurd M. M. Optimization of the drayage problem using exact methods. *INFOR: Information Systems and Operational Research*, 54(1):33–51, 2016.

[34] F. Schulte, E. Lalla-Ruiz, R. G. González-Ramírez, and S. Voß. Reducing port-related empty truck emissions: a mathematical approach for truck appointments with collaboration. *Transportation Research Part E: Logistics and Transportation Review*, 105:195–212, 2017.

[35] S. Shiri and N. Huynh. Assessment of us chassis supply models on drayage productivity and air emissions. *Transportation Research Part D: Transport and Environment*, 61:174–203, 2017.

[36] Y. Song, J. Zhang, Z. Liang, and C. Ye. An exact algorithm for the container drayage problem under a separation mode. *Transportation Research Part E: Logistics and Transportation Review*, 106:231–254, 2017.

[37] S. Sterzik and H. Kopfer. A tabu search heuristic for the inland container transportation problem. *Computers & Operations Research*, 40(4):953–962, 2013.

[38] K. C. Tan, Y. H. Chew, and L. H. Lee. A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems. *European Journal of Operational Research*, 172(3):855–885, 2006.

[39] M. Vidović, D. Popović, B. Ratković, and G. Radivojević. Generalized mixed integer and vns heuristic approach to solving the multisize containers drayage problem. *International Transactions in Operational Research*, 24(3):583–614, 2017.

[40] M. Vidović, G. Radivojević, and B. Raković. Vehicle routing in containers pickup up and delivery processes. *Procedia-Social and Behavioral Sciences*, 20:335–343, 2011.

[41] W. F. Wang and W. Y. Yun. Scheduling for inland container truck and train transportation. *International journal of production economics*, 143(2):349–356, 2013.

[42] X. Wang and C. A. Regan. Local truckload pickup and delivery with hard time window constraints. *Transportation Research Part B: Methodological*, 36:97–112, 2002.

[43] S. Wen and P. Zhou. A container vehicle routing model with variable traveling time. *Automation and Logistics, 2007 IEEE International Conference*, pages 2243–2247, 2007.

[44] Z. Xue, W. H. Lin, L. Miao, and C. Zhang. Local container drayage problem with tractor and trailer operating in separable mode. *Flexible Services and Manufacturing Journal*, 27(2-3):431–450, 2015.

[45] Z. Xue, C. Zhang, W. H. Lin, L. Miao, and P. Yang. A tabu search heuristic for the local container drayage problem under a new operation mode. *Transportation Research Part E: Logistics and Transportation Review*, 62:136–150, 2014.

[46] X. Yang and A. D. Hajem. A column generation based decomposition and aggregation approach for combining orders in inland transportation of containers. *OR Spectrum*, 42:261–296, 2020.

[47] R. Zhang, J. C. Lu, and D. Wang. Container drayage problem with flexible orders and its near real-time solution strategies. *Transportation Research Part E: Logistics and Transportation Review*, 61:235–251, 2014.

[48] R. Zhang, W. Y. Yun, and Moon I. K. A reactive tabu search algorithm for the multi-depot container truck transportation problem. *Transportation Research Part E: Logistics and Transportation Review*, 45(6):904–914, 2009.

[49] R. Zhang, W. Y. Yun, and H. Kopfer. Heuristic-based truck scheduling for inland container transportation. *OR spectrum*, 32(3):787–808, 2010.

[50] R. Zhang, W. Y. Yun, and H. Kopfer. Multi-size container transportation by truck: modeling and optimization. *Flexible Services and Manufacturing Journal*, 27(2-3):403–430, 2015.

[51] R. Zhang, W. Y. Yun, and I. K. Moon. Modeling and optimization of a container drayage problem with resource constraints. *International Journal of Production Economics*, 133(1):351–359, 2011.