JST: Smart Systems and Devices
Volume 32, Issue 2, May 2022, 064-073

Solving a Real-World Problem of Truck-Trailer Scheduling in Container Transportation by Local Search

Van Son Nguyen[1,2*], Quang Dung Pham[1]
[1]*Hanoi University of Science and Technology, Hanoi, Vietnam*
[2]*Academy of Cryptography Techniques, Hanoi, Vietnam*
[*]*Email: son.nvncs17020@sis.hust.edu.vn*

Abstract

Container transportation plays a very important role in global supply chains where companies of different countries exchange goods overseas. Operational costs of transporting containers are extremely high. Hence, optimizing itinerary schedules brings substantial benefits to logistics companies. In this paper, we investigate a realistic container transportation schedule problem in which trucks, trailers, and containers are separate objects located at different terminals, and trucks are planned to take trailers and carry containers between container depots, ports, and customer warehouses. In this context, containers can be 40ft or 20ft, and a trailer can carry one 40ft container or one 20ft container, or two 20ft containers. Moreover, a truck can detach the trailer and leave both the container and trailer at the warehouse if there is no forklift available. We first propose a mixed-integer linear programming model for describing the problem. Then, we propose a local search algorithm for solving the problem. Experiments on generated data instances show the benefits of the proposed algorithm compared to the previously proposed algorithm in which the improvement rate of the solution quality is 19.62% on average. Furthermore, the application of the model leads to rapidity in generating the solution; this task that used to take days is decreased to just one hour.

Keywords: vehicle routing problem; container transportation; drayage operation; local search.

1. Introduction

Nowadays, freight transportation by containers has conferred substantial flexibility to logistic operations. Container transportation operations by truck between ports, shippers, customer warehouses and inland container terminals are usually called *container drayage operations*. Due to globalization, the growth in container transportation had a significant impact on the reduction of transportation costs and time. Efficient scheduling of these operations is extremely important for transportation companies since the relevant costs consume a high percentage of the total operational costs.

In recent research, an inland container transportation problem (ICTP) was discussed by [1-6]. In [1], the authors studied the integrated drayage scheduling problem and formulated it as an asymmetric multiple Traveling Salesman Problem with Time Windows (TSPTW). In an extension of the problem [2], they limited the number of empty containers available at a depot. Besides, several extensions of this problem have also been studied, and several incomplete algorithms have been proposed for solving them, for example, [4, 5]. For exact solutions, a generalization of a capacitated truck-trailer routing problem with time windows was considered in [6]. A branch-and-price-and-cut algorithm to solve the ICTP problem and a hybrid acceleration strategy for performance improvement are also proposed in [7]. They often investigated the ICTP with a single depot and a single terminal and assumed that trucks have their own trailer. However, several affiliated third-party logistics companies lease trucks, trailers, and containers. Thus, objects (i.e., trailers and trucks) are placed in various locations and can start and terminate at one of the depots. In particular, the trailer can also be detached from the truck at each port or warehouse. Therefore, they must be treated as separate objects. On the other hand, containers are often assumed to be homogeneous. For example, in [1-3] the authors investigated the problem that a truck transports a fixed-size container at a time to simplify the mathematical model and reduce the complexity of computation. However, in most real-life situations, several customers want to receive/ship freight in 20ft containers or 40ft containers. Thus, the problem with multiple containers per truck at a time needs to be considered. Although several other papers considered the container transportation problems with heterogeneous containers [4-8], trucks and trailers are not separate objects in these works. The combination of these real-world factors makes the truck-trailer scheduling problem for transporting containers more challenging.

ISSN: 2734-9373
https://doi.org/10.51316/jst.158.ssad.2022.32.2.9
Received: December 21, 2021; accepted: March 28, 2022

In this paper, we consider the problem of truck-trailer scheduling in container transportation (TTCRP), in which trucks, trailers, and containers are separate objects located at different terminals. A truck is scheduled to take a trailer and then carry containers. The truck cannot carry a container itself. A trailer can carry only one 40ft container or at most two 20ft containers. In addition, when a container is transported to a customer or port, the trailer can be detached from the truck. The truck leaves both the trailer and the container (the container lies on the trailer) at that drop-off point for serving other transportation requests. In our model, there are four types of container requests. The time window constraint is also considered (this specification is described by the SmartLog company, one of the biggest technology companies providing services for logistics operations). The main contributions of our paper are listed as follows:

- We define a problem of truck-trailer scheduling in multi-size container transportation that combine the following features: trucks, trailers, and containers are separate objects located at different terminals; a truck is scheduled to take a trailer and then carry containers; a trailer can be detached from the truck at ports or warehouses.

- We formulate the considered problem with a mixed-integer linear programming (MILP) model. We solve small-scale instances by using GUROBI Optimizer to validate our model.

- We propose a local search algorithm for solving the considered problem with a strategy including generating initial solutions and subsequently applying neighborhood operators to improve the quality of solutions. The performance of algorithms is compared by conducting extensive numerical experiments to evaluate their applicability in real-world applications. The instances and experimental results are available on (https://github.com/sonnv188/TTCRP.git) for further research and comparison.

This paper is structured as follows. Section 2 presents the TTCRP problem formulation. Then, proposed algorithms for solving the problem are described in Section 3. Experimental settings and numerical results are displayed in Section 4. The last section concludes the proposed solution and draws some future works.

2. Problem Description

2.1. Definitions and Assumptions

Every day, a transportation company receives container transportation requests. There are four types of containers corresponding to four types of requests:

- *Inbound full* (IF): A loaded container located at a port needs to be transported to a customer.

- *Inbound empty* (IE): An unloaded container located at a customer or port needs to be retained at an empty container depot.

- *Outbound empty* (OE): A container located at an empty container depot needs to be transported to a port or customer for loading export goods.

- *Outbound full* (OF): A full container located at a customer needs to be delivered to a port.

There is a set of homogeneous trucks K, homogeneous trailers P, and empty heterogeneous containers Q. We denote sets of truck depots and terminals by DK and TK, respectively. Each truck departs from a depot in DK and terminates at a designated terminal in TK. We denote a set of trailer depots by DP. TP is a set of trailer terminals. DQ is a set of container depots. Ports and customer warehouses have time windows for container activities, whereas truck, trailer, and container depots have no time window. It means that all IF and OF requests have two time windows, while IE and OE requests have only a time window. If there are not any available forklifts for taking the container out of the trailer, both the containers and the trailer can be detached from the truck. The truck can be scheduled to take another trailer for other itineraries.

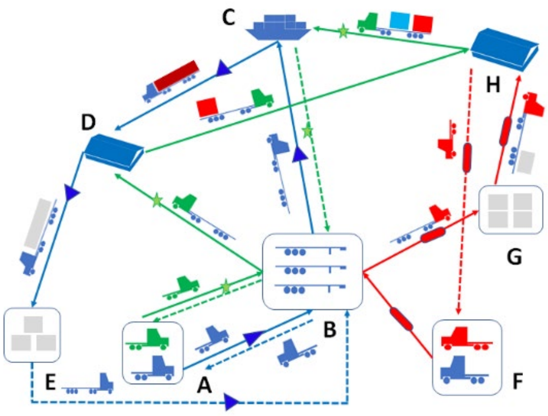

Fig. 1. An example of the TTCRP problem.

Fig. 1 presents an example of this problem. There are four trucks and three trailers available at depots in this instance. A truck leaves from depot A, arrives at the trailer depot B to take a trailer, then reaches port C to pick up the loaded container and transport it to customer D. And then, the empty container is retained at depot E by this truck, and the trailer is retained at depot B. Finally, the truck comes back to depot A. In the second route, a truck takes a trailer at depot B. It transports to customer D to take a 20ft loaded container that needs delivery to port C. The truck continues to go to customer H to collect another 20ft loaded container that also needs delivery to port C. Another truck starts at truck depot F. It then reaches depot B to take a trailer. Then, this truck arrives at container depot G to take an empty container and delivers this container to customer H. Due to no available forklifts at customer G, the truck leaves both the trailer and container at customer G and returns to depot F.

Some notions and parameters used throughout the rest of the paper will be presented in Table 1. Table 2 defines the modelling variables.

Table 1. Input

Notation	**Definition**
K	A list of trucks $K = \{0,1,...,\|K\|-1\}$. For each truck $k \in K$, *d(k)* and *t(k)* are the depot and termination of truck k respectively. $DK = \{d(k)\}_{k\in K}$, $TK = \{t(k)\}_{k\in K}$.
P	A list of trailers $P = \{0,1,...,\|P\|-1\}$. For each trailer $p \in P$, *d(p)* and *t(p)* are the depot and termination of trailer p respectively. $DP = \{d(p)\}_{p\in P}$, $PK = \{t(p)\}_{p\in P}$.
Q	A list of empty containers $Q = \{0,1,...,\|Q\|-1\}$. For each container $q \in Q$, *d(q)* is the depot of container q. $DQ = \{d(q)\}_{q\in Q}$.
T	A list of returned container depots $T = \{0,1,...,\|T\|-1\} \subseteq Q$.
IF	Set of IF requests, for each $r \in IF$, *p(r)* and *w(r)* are points of port (pickup point) and customer (delivery point). $IF_p = \{p(r)\}_{r\in IF}$, $IF_w = \{w(r)\}_{r\in IF}$, $IF^* = \{(p_r, d_r) \mid \forall r \in IF \wedge p_r = p(r) \wedge d_r = w(r)\}$.
IE	Set of IE requests, for each $r \in IE$, *w(r)* is a point of customer (pickup point). $IE_w = \{w(r)\}_{r\in IE}$, $IE^* = IE_w \times T = \{(p_r, d_r) \mid \forall r \in IE \wedge p_r = w(r) \wedge d_r \in T\}$.
OF	Set of OF requests, for each $r \in OF$, *w(r)* and *p(r)* are points of customer (pickup point) and port (delivery point). $OF_w = \{w(r)\}_{r\in OF}$, $OF_p = \{p(r)\}_{r\in OF}$, $OF^* = \{(p_r, d_r) \mid \forall r \in OF \wedge p_r = w(r) \wedge d_r = p(r)\}$.
OE	Set of OE requests, for each $r \in OE$, *w(r)* is a point of customer (delivery point). $OE_w = \{w(r)\}_{r\in OE}$, $OE^* = DQ \times OE_w = \{(p_r, d_r) \mid \forall r \in OE \wedge p_r \in DQ \wedge d_r = w(r)\}$.
R	Set of all customer requests $R = IE \cup IF \cup OE \cup OF$. We denote $\|R\| = N$.
R*	Set of point-pairs of pickup and delivery points $R = IE^* \cup IF^* \cup OE^* \cup OF^*$. We denote $\|R^*\| = N^*$
S	Set of intermediate points where each truck visits each point of S at most once. $S = IF_p \cup IF_w \cup IE_w \cup OF_w \cup OF_p \cup OE_w \cup DP \cup TP \cup DQ \cup T$.
V	Set of all vertices $V = DK \cup S \cup TK$.
A	Set of possible arcs $A = \{(i,j) \mid \forall i \in DK \cup S, \forall j \in S \cup TK\}$. $\delta^+(i) = \{j \mid (i,j) \in A\}, \forall i \in V$, $\delta^-(i) = \{j \mid (j,i) \in A\}, \forall i \in V$.
e_v, l_v, s_v	The earliest, latest arrival time and serving duration at point $v \in V$.
$t_{i,j}, d_{i,j}$	The travel time and distance from point i to point j $\forall (i,j) \in A$.
c_p, c_t	Penalty cost of one unserved request and initial cost of using one truck.
$r_{p,d}$	A parameter identifies request $r \in R$ that has the pickup point p and the delivery point d, $\forall (p,d) \in R^*$.
M	A very big constant.

Table 2. Modelling variables

Notation	Definition
$X_{i,j}^k$	A binary variable equals 1 if truck k travels on arc *(i, j),* or equals zero otherwise, $\forall k \in K, (i, j) \in A$
b(v)	A binary variable equals 1 if the trailer is unhitched from a truck at point $v \in V$, equals 0 otherwise.
m(v)	A variable to identify the trailer weight of points. $m(d(p_i)) = 2\mid P\mid + i + 1,\ m(t(p_i)) = -(2\mid P\mid + i + 1), \forall p_i \in P,\ \forall i = \overline{0, \mid P\mid - 1}.$ $m(v) = -\lvert P\rvert\ b(v)$, otherwise.
c(v)	A variable to identify the container weight and type of the container at point $v \in V$: *c(v) = 2* if a 40ft container is picked up*, c(v) = -2* if a 40ft container is delivered, *c(v) = 1* if a 20ft container is picked up, *c(v) = -1* if a 20ft container is delivered at point *v, c(v) = 0*, otherwise.
$c_{oe}(v)$	A variable to identify the container weight at point *v* related a request in OE. $c_{oe}(v) = c(v) \forall v \in DQ \cup OE_w,\ c_{oe}(v) = 0$, otherwise.
$c_{ie}(v)$	A variable to identify the container weight at point *v* related a request in IE. $c_{ie}(v) = c(v) \forall v \in IE_w \cup T,\ c_{ie}(v) = 0$, otherwise.
TL(v)	The accumulated trailer weight after leaving point *v* of the itinerary visiting *v*, $v \in S$.
CL(v)	The accumulated container weight after leaving point *v* of the itinerary visiting *v*, $v \in S$.
$CL_{oe}(v)$	The accumulated container weight after leaving point *v* of the itinerary visiting *v*, $v \in DQ \cup OE_w$.
$CL_{ie}(v)$	The accumulated trailer weight after leaving point *v* of the itinerary visiting *v*, $v \in IE_w \cup T$.
AT(v)	The time point when a truck arrives at point $v \in V$.
DT(v)	The time point when a truck departs from point $v \in V$.
ST(v)	Start serving time at point $v \in V$.
y_0, y_1	Two binary variables.

2.2. Mathematical Formulation

In this section we formulate the considered problem with a MILP model. We denote the number of used trucks is denoted by $g_t = \sum_{k\in K}\sum_{j\in S} X_{d(k),j}^k$ the number of unserved requests by the notation:

$$g_r = N - \sum_{r\in OE}\sum_{i\in\delta^-(\mathrm{w}(r))}\sum_{k\in K} X_{i,\mathrm{w}(r)}^k - \sum_{r\in IE}\sum_{i\in\delta^-(\mathrm{w}(r))}\sum_{k\in K} X_{i,\mathrm{w}(r)}^k - \sum_{r\in OF}\sum_{i\in\delta^-(\mathrm{w}(r))}\sum_{k\in K} X_{i,\mathrm{w}(r)}^k - \sum_{r\in IF}\sum_{i\in\delta^-(p(r))}\sum_{k\in K} X_{i,p(r)}^k$$

and the total travel distance by $g_c = \sum_{k\in K}\sum_{(i,j)\in A} d_{i,j} X_{i,j}^k$.

The TTCRP problem formulation is as follows:

$$F = \min(c_p g_r + c_t g_t + g_c) \tag{1}$$

subject to:

$$\sum_{j\in\delta^+(i)}\sum_{k\in K} X_{i,j}^k \le 1, \forall i \in DK \cup S \tag{2}$$

$$\sum_{j\in\delta^+(i)} X_{i,j}^k = \sum_{j\in\delta^-(i)} X_{j,i}^k, \forall i \in S, k \in K \tag{3}$$

$$\sum_{k\in K}\sum_{\substack{(i,j)\in A\\ i,j\in\mathbb{S}}} X_{i,j}^k \le \mid\mathbb{S}\mid - 1, \forall \mathbb{S} \subset V, \mathbb{S} \ne \varnothing \tag{4}$$

$$\sum_{j\in S} X_{d(k),j}^k \le 1, \forall k \in K \tag{5}$$

$$\begin{aligned} &DT(i) + t_{i,j} + M(\sum_{k\in K} X_{i,j}^k - 1) \le AT(j),\\ &DT(i) + t_{i,j} + M(1 - \sum_{k\in K} X_{i,j}^k) \ge AT(j),\\ &\forall (i,j) \in A \end{aligned} \tag{6}$$

$$ST(i) + s_i = DT(i), \forall i \in V \tag{7}$$

$$ST(i) \ge AT(i), \forall i \in V \tag{8}$$

$$ST(i) \geq e_i, \forall i \in V \tag{9}$$

$$AT(i) \leq l_i, \forall i \in V \tag{10}$$

$$\begin{aligned} &TL(i) + m(j) + M(\sum_{k \in K} X^k_{i,j} - 1) \leq TL(j), \\ &TL(i) + m(j) + M(1 - \sum_{k \in K} X^k_{i,j}) \geq TL(j), \\ &\qquad \forall (i,j) \in A \end{aligned} \tag{11}$$

$$\begin{aligned} &CL(i) + c(j) + M(\sum_{k \in K} X^k_{i,j} - 1) \leq CL(j), \\ &CL(i) + c(j) + M(1 - \sum_{k \in K} X^k_{i,j}) \geq CL(j), \\ &\qquad \forall (i,j) \in A \end{aligned} \tag{12}$$

$$\begin{aligned} &CL_{oe}(i) + c_{oe}(j) + M(\sum_{k \in K} X^k_{i,j} - 1) \leq CL_{oe}(j), \\ &CL_{oe}(i) + c_{oe}(j) + M(1 - \sum_{k \in K} X^k_{i,j}) \geq CL_{oe}(j), \\ &\qquad \forall (i,j) \in A \end{aligned} \tag{13}$$

$$\begin{aligned} &CL_{ie}(i) + c_{ie}(j) + M(\sum_{k \in K} X^k_{i,j} - 1) \leq CL_{ie}(j), \\ &CL_{ie}(i) + c_{ie}(j) + M(1 - \sum_{k \in K} X^k_{i,j}) \geq CL_{ie}(j), \\ &\forall (i,j) \in A \end{aligned} \tag{14}$$

$$0 \leq TL(i) \wedge TL(i) \leq 3MY, \forall i \in V \tag{15}$$

$$0 \leq CL(i) \wedge CL(i) \leq 2, \forall i \in V \tag{16}$$

$$TL(d(k)) = 0, \forall k \in K \tag{17}$$

$$CL(d(k)) = CL(t(k)) = 0, \forall k \in K \tag{18}$$

$$CL_{oe}(d(k)) = CL_{oe}(t(k)) = 0, \forall k \in K \tag{19}$$

$$CL_{ie}(d(k)) = CL_{ie}(t(k)) = 0, \forall k \in K \tag{20}$$

$$\begin{aligned} &TL(t(k)) + M(1 - y_0) \geq 0, \forall k \in K \\ &TL(t(k)) - M(1 - y_0) \leq 0, \forall k \in K \\ &TL(t(k)) + M(1 - y_1) \geq MY + 1, \forall k \in K \\ &TL(t(k)) - M(1 - y_1) \leq 2MY, \forall k \in K \\ &\mathrm{y}_0 + \mathrm{y}_1 = 1 \end{aligned} \tag{21}$$

$$CL(i) \leq TL(i), \forall i \in V \tag{22}$$

$$\sum_{i \in \delta^-(w(r))} \sum_{k \in K} X^k_{i,w(r)} \leq 1, \forall r \in R \tag{23}$$

$$AT(w(r)) \leq AT(p(r)), \forall r \in OF \tag{24}$$

$$AT(p(r)) \leq AT(w(r)), \forall r \in IF \tag{25}$$

$$\begin{aligned} &\sum_{i \in \delta^-(w(r))} X^k_{i,w(r)} = \sum_{i \in \delta^-(p(r))} X^k_{i,p(r)}, \\ &\forall r \in OF \cup IF, \forall k \in K \end{aligned} \tag{26}$$

Constraint (2) ensures that each intermediate point is visited at most once. Constraint (3) presents flow conservation constraints. Subtours are eliminated by Constraint (4). Constraint (5) specifies that a truck starts from its depot at most once. Constraints (6)-(7) relate to the arrival, departure and service duration between two points. Time windows are set by Constraints (8)-(10). Constraints (11)-(12) relate to the accumulated trailer and container weights after leaving each point. The accumulated container weights related to OE and IE requests are computed by Constraints (13)-(14). Constraint (15) specifies that no more than one trailer visits a point. The boundary of container weight is set by Constraint (16). Constraints (17)-(20) compute the accumulated trailer and container weights at truck depots. Constraint (21) enforces that a trailer must be delivered to a customer or its termination. For more detail, the number of trailers at a truck depot equals zero. If a trailer is unhitched from a truck at the customer, the accumulated trailer weight at the termination of the truck is equal to *(m(d(p)) - m(v))*, or equal to zero otherwise. Constraint (22) states that a container must be carried by trailer. Constraint (23) guarantees that each request is served at most once. Constraints (24) - (25) state that the pickup point is visited before visiting the delivery point for each OF or IF request. Constraint (26) specifies that the pickup and delivery container points of OF and IF requests must be assigned to the same truck route.

The objective function F states that the number of unserved requests, the number of used trucks and total travel distance should be minimized. The number of unserved requests is firstly minimized with a large value of c_p. If c_t is sufficiently large, the model will primarily minimize the number of used trucks.

3. Solution Methods

We have experimented with the proposed MILP model by using the GUROBI solver. Experimental results show that the GUROBI solver cannot find optimal solutions for instances with more than six requests within one hour. Due to the huge computational complexity, the MILP model cannot be solved with large instances. Therefore, we aim to propose heuristic algorithms to handle large instances of the TTCRP in this section. We give some notations to facilitate the presentation of heuristic algorithms as follows. A solution to the TTCRP problem is denoted by $s = \langle s^0, s^1, ..., s^{|K|-1} \rangle$ where s^k is a route of truck k represented by a sequence $\langle s^k_0, s^k_1, ..., s^k_{len(k)} \rangle$ where $s^k_0 \in DK, s^k_{len(k)} \in TK$ are the starting and terminal points of the truck k, $s^k_i \in S$ $(i = \overline{1, len(k)-1})$ are intermediate points that are visited by truck k. An empty solution contains routes that are initialized without intermediate points (i.e., $len(k) = 2$, $\forall k = \overline{0, |K|-1})$. We denote:

- $ir(v)$: The index of the truck route containing point $v \in V$. $ir(v) = \perp$ if point *v* is not in any truck route.
- $pr(v)$, $sc(v)$: The previous point and successor point of point *v,* respectively. If $ir(v) = \perp$, $pr(v) = sc(v) = \perp, \ \forall v \in V$.
- $cl(v)$: The number of container units attached to the truck after leaving point $v \in V$. A 40ft container unit equals 2. A 20ft container unit equals 1.
- $tl(v)$: The number of trailer units attached to the truck after leaving point $v \in V$. Each trailer is assumed to be 2 container units (each time the truck arrives at a point for taking a trailer, the number of trailer units is increased by 2, each time the trailer is detached from the truck, the number of trailer units is decreased by 2).
- $I(s^k, v_1, v_2)$: The operator that inserts point v_1 right after point v_2 of route s^k, and returns a new route.

3.1. Quality Function

One of the cores of a heuristic algorithm is a function that models the quality of solutions. This function is used to control the solution search. Basically, the function is the objective function *F* described above combines three components: the number of unserved customers g_r, the number of used trucks g_t, and the total travel distance g_c in a linear way with coefficients. However, according to [10], these objectives are very often conflicting in practice. Indeed, to reduce the number of unserved customers, some trucks are scheduled to carry too little cargo in real-world situations. This schedule requires more trucks and provides low profitability and causes resource imbalances. In addition, choosing coefficients in actual operation is difficult. In actual operation, g_r is often treated with the highest priority, and g_c is treated with the lowest priority. Hence, in our proposed algorithms, we form the control function *F'(s)* over a solution *s* as a vector of three components treated in a lexicographic order: given two solutions s_1 and s_2. $F'(s_1) < F'(s_2)$ if:

- $g_r(s_1) < g_r(s_2)$
- $g_r(s_1) = g_r(s_2)$ and $g_t(s_1) < g_t(s_2)$
- $g_r(s_1) = g_r(s_2)$ and $g_t(s_1) = g_t(s_2)$ and $g_c(s_1) < g_c(s_2)$.

3.2. Trailer Point Insertion

Algorithms proposed and presented in this paper are based on operations of inserting points into routes under construction while maintaining constraint satisfaction. More precisely, when a container pickup point is inserted into the truck's route, a trailer pickup point must be inserted before the container pickup point (if necessary). It means that the truck must pick up the trailer before picking up the container because the container must lie on the trailer attached to the truck. If the truck has a trailer attached with sufficient capacity, it can go to pick up a container without having to pick up other trailers. We denote the function of inserting trailer points into a solution *s* satisfying constraints by *TrailerInsertion(s)* (see Algorithm 1). This function uses a greedy strategy that schedules a truck to visit the nearest trailer depot before picking up a container (lines 5-6). If a trailer is attached to the truck and the number of container units equals 1, the truck can carry an extra 20ft containers. Line 7 marks the delivery point of the attached trailer. Lines 11-13 require that each truck must deliver the attached trailer to the trailer depot if the truck does not leave the trailer at the last delivery point.

Algorithm 1: $TrailerInsertion(s = \langle s^0, ..., s^{|K|-1} \rangle)$

1 **for** $k = 0$ *to* $|K|$ - *1* **then**
2     $v \leftarrow s_1^k; v_l \leftarrow \perp;$
3   **while** $v \neq s_{len(k)}^k$ **do**
4   **if** $tl(v) < cl(v)$ **then**
5             $p^* \leftarrow$ An available trailer at the nearest trailer depot;
6             $s^k \leftarrow I(s^k, d(p^*), pr(v));$
7             $v_l \leftarrow t(p^*);$
8   **end**
9         $v \leftarrow sc(v);$
10   **end**
11   **if** $b(s_{len(k)-1}^k) = 0$ **then**
12         $s^k \leftarrow I(s^k, t(p^*), s_{len(k)-1}^k);$
13   **end**
14 **end**

3.3. Neighborhoods

The neighborhoods we consider in our algorithm are based on neighborhoods proposed by [11]. This paper uses popular neighborhoods, including one-request-move, two-request-move, two-opt-move, or-opt-move, three-opt-move, three-request-move cross-exchange neighborhoods. We note that for each local move, the pickup and delivery points of each request are always on the same route ($ir(p_r) = ir(\mathrm{d}_r), \forall (p_r, d_r) \in R^*$). Due to lack of space, we do not present in detail these neighborhoods. Interested readers can refer to [11] for more detail about these neighborhood structures.

3.4. Local Search Algorithm

The proposed local search is depicted in Algorithm 2. L is a list of considered neighborhoods, and *maxStables* is a given parameter. An initial solution is generated in a greedy constructive manner at line 1. Line 2 updates the best solution found so far. At each iteration of the local search, line 4 shuffles that order of the neighborhoods of L. Lines 7-11 iteratively explore these neighborhoods (see Algorithm 3). The neighborhood exploration will terminate whenever it discovers a first neighbor which is better than the current solution s (lines 9-10). Line 12 replaces the current solution by a randomly selected neighbor of E. If the selected neighbor is better than the best solution found so far s^* then s^* is updated (lines 13-14). Otherwise, the search augments the number of consecutive iterations *nic* in which no improvement is found by one. Due to the complexity of the problem, it is possible that no further improvements can be found after some iterations. The search will be restarted if *nic* exceeds *maxStables* to avoid getting stuck in local optima (see lines 15-20).

Algorithm 2: LS(*L, maxStables*)

1 $s \leftarrow$ Generate an initial solution;
2 $s^* \leftarrow s; nic \leftarrow 1;$
3 **while** *stop condition is not expired* **do**
4   Shuffle(L);
5    $E \leftarrow \{\varnothing\};$
6    $e \leftarrow \infty;$
7   **foreach** *neighborhood* $N_i \in L$ **do:**
8       $\langle E, e\rangle \leftarrow$ Explore(N_i, E, e);
9   **if** $e < F'(s)$ **then**
10   break;
11   **end**
12    $s \leftarrow$ Select(E);
13   **if** $F'(s) < F'(s^*)$ **then**
14       $s^* \leftarrow s; nic \leftarrow 1;$
15   **else**
16   *nic*++;
17   **if** *nic* > *maxStables* **then**
18          $s \leftarrow$ Generate an initial solution;
19          $nic \leftarrow 1;$
20   **end**
21   **end**
22 **end**

Algorithm 3 receives a neighborhood N and a set E of potential solutions which have been already found so far (i.e., by exploring previous neighborhoods). It scans all solutions of the considered neighborhood N (line 1), completes routes by inserting trailer points (line 2) and returns a new set of best solutions E and their evaluation e (lines 3-9).

Algorithm 3: Explore*(N, E, e)*

1 **for** $s \in N$ **do**
2    $s \leftarrow$ TrailerInsertion*(s)*;
3   **if** $F'(s) < e$ **then**
4       $E \leftarrow \{s\};$
5       $e \leftarrow F'(s);$
6   **else**
7   **if** $F'(s) = e$ **then**
8          $E \leftarrow E \cup \{s\};$
9   **end**
10 **end**
11 **return** $\langle E, e\rangle$

4. Experiments

In this section, we conduct experiments for evaluating the efficiency of two proposed algorithms and also compare them with the best algorithm *HAS(BPIUS)* in [9] (the algorithm in [9] is the strategy of manual scheduling procedures that lean on basic greedy approaches based on the rule of thumb). The implementation of these algorithms is based on the library CBLSVR [12] that is a constraint-based local search library for general vehicle routing problems. Test instances are solved on an Intel(R) Core(TM) i7-4790 CPU @ 3.660 GHz, CPU 16 GB RAM computer.

4.1. Instances and Setting

Our paper investigates a new realistic multi-size container transportation routing problem that has not been investigated in the literature to the best of our knowledge. Therefore, there is no available standard benchmark for the problem. We generate the instances for evaluating the efficiency of the proposed algorithms as follows. We collect the information of eight major seaports in the North of Vietnam and 40 warehouses of big import-export companies based on the Google Maps service. Truck, trailer and container depots are equipment yards of 7 hinterland transportation Vietnam companies. Instances follow a naming convention of *N-x-y* where x is the number of requests that need to be served and y is the index of the generated data set. The number of trucks is limited in 10% and 20% of x. In our instances, trucks are available from 2019-06-11 18:00:00. The starting time for pickup or delivery at each customer warehouse is randomly chosen between 2019-06-12 00:00:00 and 2019-06-14 23:59:59. The width of time windows is 7 hours and 15 minutes for serving time of every single point. The travel distance from one point to another is measured by using the Google Maps service, and the average speed is around 40 km/h. The total travel time of trucks is converted to distance by the average speed to compute the cost of each solution. We analyze the performance of our MILP model by experiments on the small generated instances *N-4-0* and *N-6-0*.

4.2. Mathematical Formulation Validation

The first experiment is to validate and analyze the performance of our MILP model for the TTCRP problem. By selecting instances where requests can be served, $c_p = 3N \max_{(i,j)\in A} d_{i,j}$, and $c_t = N \max_{(i,j)\in A} d_{i,j}$, it is possible to compare solutions of the MILP model with solutions of the proposed heuristic algorithms. Table 3 states that the results of HAS(BPIUS), LS algorithms and the MILP model are the same, and then, our formulation is validated. Moreover, these experiments show that our heuristic algorithms can find optimal solutions in small instances. GUROBI Optimizer cannot solve the MILP model for instances having more than six requests within one hour.

4.3. Comparison between the Efficiency of Algorithms

The results of 25 instances for 20 to 200 requests are presented in Table 4. The best solution is found after 10000 iterations (*maxStables* = 1000) or one hour (*stop condition*) of execution and shown in boldface. The average values are determined after running 10 times. The results reveal that for most instances, both the average number of unserved requests (*avg* g_r) and the average number of trucks needed (*avg* g_t) of solutions obtained by the LS algorithm are lower than the corresponding values of the solution obtained by HAS(BPIUS) algorithm. We note that a solution having fewer unserved requests is always considered better than a solution with more unserved requests, regardless of the total travel distance. The results also show that the proposed LS algorithm can find good quality solutions for large instances of this problem within a reasonable computational time (less than 1 hour). The small standard deviation (*std* g_c) of the results found by the LS algorithm implies that the LS algorithm is more stable than the HAS algorithm. Besides, at instances *N-70-0, N-70-2, N-70-4, N-100-0, N-100-1, N-100-4, N-150-1 and N-150-4*, the average cost found by the LS algorithm is worse than that of HAS(BPIUS) due to the number of unscheduled requests of the LS algorithm or the number of trucks needed is lower than that of HAS(BPIUS). To illustrate the improvement rate, we compute the $GAP = (F_1^{'} - F_2^{'})*100 / F_1^{'}$ between values of the objective function $F^{'}$, where $c_p = 3N \max_{(i,j)\in A} d_{i,j}$, and $c_t = N \max_{(i,j)\in A} d_{i,j}$. In general, using the proposed LS algorithm, we found that the average improvement rate is 19.62 (the rate is from 0 to 68.17). Although the improvement rate is equal to zero in some instances, the LS algorithm still obtains solutions better than HAS(BPIUS). Furthermore, the application of the model led to the rapidity in generating the solution; this task that used to take days is decreased to just one hour.

4.4. The impact of different objective functions

In this experiment, we implement the LS algorithm with the different quality functions

- $F_{g_r}^{'}(s_1) < F_{g_r}^{'}(s_2)$ if $g_r(s_1) < g_r(s_2)$
- $F_{g_t}^{'}(s_1) < F_{g_t}^{'}(s_2)$ if $g_t(s_1) < g_t(s_2)$
- $F_{g_c}^{'}(s_1) < F_{g_c}^{'}(s_2)$ if $g_c(s_1) < g_c(s_2)$

and compute the GAP between their values with the value of $F_2^{'}$ at instance *N-100-0* in Table 4. The results are illustrated by Fig. 2. In Fig. 2, we found that these objectives are actually conflicting. The objective function $F_{g_r}^{'}$ that reduces the number of unserved requests leads to the increasing of used trucks. Besides, to minimize the using truck cost $F_{g_t}^{'}$, some low profit requests are unserved.

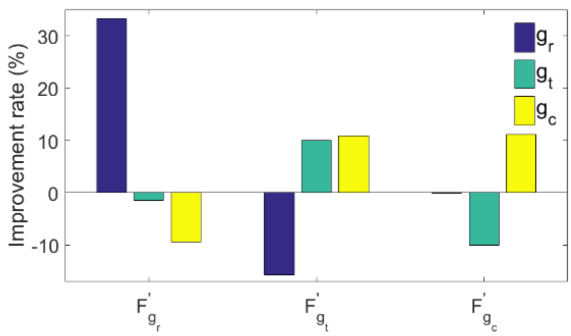

Fig. 2. The impact of the different quality functions.

Table 3. Comparison between results of HAS(BPIUS), LS, and MILP model

Ins	MILP model				HAS(BPIUS)				LS			
	g_r	g_t	g_c	t(s)	g_r	g_t	g_c	t(s)	g_r	g_t	g_c	t(s)
N-4-0	0	1	280	205	0	1	280	1.03	0	1	280	13
N-6-0	0	1	216	2311	0	1	216	1.15	0	1	216	37

Table 4. Results of heuristic algorithms

Ins	HAS(BPIUS)[9]						LS						GAP
	avg g_c	*std* g_c	*avg* g_t	*avg* g_r	*avg* t	F_1'	*avg* g_c	*std* g_c	*avg* g_t	*avg* g_r	*avg* t	F_2'	
N-20-0	201	33	3	0.6	56	1209801	151	4	3	**0**	624	756151	37.5
N-20-1	196	46	3	1	609	1512196	187	3	3	**0**	974	756187	49.99
N-20-2	144	23	3	0.5	651	1035144	123	3	3	**0**	551	690123	33.33
N-20-3	154	4	2	0	1587	502154	**151**	8	2	0	1403	527251	0
N-20-4	171	5	2.2	0	1519	464371	154	**5**	**2**	0	1528	422154	9.09
N-70-0	468	52	7	7.8	1276	7904468	628	32	7.4	**3.9**	1354	4966628	37.17
N-70-1	655	57	7	2.2	448	3536655	585	21	7	**0**	610	1820585	48.52
N-70-2	598	31	7	5	430	5720598	782	22	7	**0**	366	1820782	68.17
N-70-3	580	38	7	0.8	233	2444580	494	35	7	**0**	265	1820494	25.53
N-70-4	441	28	7	11.5	443	10582941	596	43	7.1	**8**	553	7931096	25.06
N-100-0	769	20	10	7.3	271	8294769	873	31	10	**3.6**	484	5408873	34.79
N-100-1	750	42	10	5.8	517	7124750	792	36	10	**4**	561	5720792	19.71
N-100-2	884	48	10	1.5	405	3770884	872	29	10	**0**	420	2600872	31.03
N-100-3	870	47	10	0.6	654	3009870	**866**	43	10	0.6	871	3009866	0
N-100-4	850	35	10	4.1	296	5798850	892	25	10	**0**	346	2600892	55.15
N-150-0	1260	23	18.8	0	1774	4889260	**1194**	21	18.8	0	1691	4889194	0
N-150-1	1327	11	19	0	1523	4941327	1408	14	**18.2**	0	1437	4837408	4.21
N-150-2	1319	46	20	0.2	1151	5357319	1273	39	19.7	**0**	1163	5201273	4.37
N-150-3	1236	43	16.6	0	1553	4317236	1213	36	**16.1**	0	1470	4187213	3.01
N-150-4	1374	29	19.9	0.4	1405	5487374	1532	32	20	**0**	1311	5201532	5.21
N-200-0	1547	20	23	0	1389	5981547	**1487**	16	23	0	1502	5981487	0
N-200-1	1573	29	24.2	0	1511	6293573	1529	13	**23**	0	1521	6241529	4.96
N-200-2	1656	19	24	0	1612	6241656	**1562**	21	24	0	1567	6241562	0
N-200-3	1701	13	28.5	0	1773	7269201	1664	13	**27.5**	0	1878	7167164	3.51
N-200-4	1600	32	23.3	0	1637	6059600	**1541**	27	23.3	0	1730	6059541	0

5. Conclusion

In this paper, we considered a realistic container transportation problem in which trucks, trailers, and containers are separate objects located at different positions. A truck is scheduled to take a trailer and then carry containers. The truck cannot carry a container itself. A trailer can carry only one 40ft container or at most two 20ft containers. In addition, when a container is transported to a warehouse, the trailer can be detached from the truck depending on whether or not a forklift is available for taking the container out of the truck. We proposed a MILP model for describing and a local search algorithm for solving the problem. Experimental results showed the efficiency of the proposed algorithm on randomly generated instances.

Our future works will investigate the online scenario of this problem in which requests are not known beforehand and revealed online during the execution of the schedule. Moreover, we believe that extending the problem more flexibly and realistically with a stochastic environment and restricted access roads will be a valuable area.

References

[1] R. Zhang, Y. Yun, I. Moon, Modeling and optimization of a container drayage problem with resource constraints. International Journal of Production Economics, 2011, vol. 133, pp. 351-359. https://doi.org/10.1016/j.ijpe.2010.02.005

[2] R. Zhang, L. Jye-Chyi, D. Wang, Container drayage problem with flexible orders and its near real-time solution strategies. Transportation Research Part E: Logistics and Transportation Review, 2014, vol. 61, pp. 235–251. https://doi.org/10.1016/j.tre.2013.11.009

[3] Fazi, S., Fransoo, J. C., Van Woensel, T., Dong, J. X., A variant of the split vehicle routing problem with simultaneous deliveries and pickups for inland container shipping in dry-port based systems. Transportation Research Part E: Logistics and Transportation Review, 2020, vol. 142. https://doi.org/10.1016/j.tre.2020.102057

[4] J. You, L. Miao, C. Zhang, Z. Xue, A generic model for the local container drayage problem using the emerging truck platooning operation mode. Transportation Research Part B: Methodological, 2020, vol. 133, pp. 181-209. https://doi.org/10.1016/j.trb.2019.12.009

[5] Xue, Z., Lin, H., You, J., Local container drayage problem with truck platooning mode. Transportation Research Part E: Logistics and Transportation Review, 2021, 147:102211. https://doi.org/10.1016/j.tre.2020.102211

[6] M. Drexl, Branch-and-price and heuristic column generation for the generalized truck-and-trailer routing problem. Revista de Metodos Cuantitativos para la Economia y la Empresa, 2011, vol. 12, pp. 5-38.

[7] Song, Y., Zhang, J., Liang, Z., Ye, C., An exact algorithm for the container drayage problem under a separation mode. Transportation Research Part E: Logistics and Transportation Review, 2017, vol. 106. https://doi.org/10.1016/j.tre.2017.07.010

[8] J. Funke, H. Kopfer, A model for a multi-size inland container transportation problem. Transportation Research Part E: Logistics and Transportation Review, 2016, vol. 89, pp. 70-85. https://doi.org/10.1016/j.tre.2016.02.010

[9] V. S. Nguyen, Q. D. Pham, A new variant of truck scheduling for transporting container problem, in Proc. SoICT, Vietnam, 2019, pp. 139-146.

[10] O. Braysy, M. Gendreau, Vehicle routing problem with time windows, Part I: Route construction and local search algorithms, Transportation Science, 2015, vol. 39(1), pp. 104-118. https://doi.org/10.1287/trsc.1030.0056

[11] C. Groër, B. Golden, E. Wasil, A library of local search heuristics for the vehicle routing problem. Mathematical Programming Computation, 2010, vol. 2, pp. 79-101. https://doi.org/10.1007/s12532-010-0013-5

[12] Q. D. Pham, L. K. T. Nguyen, V. D. Pham, and Q. T. Bui, A constraint-based local search for offline and online general vehicle routing. International Journal on Artificial Intelligence Tools, 2016, vol. 26. https://doi.org/10.1142/S021821301750004X