A Tabu Search Heuristic for the Inland Container Transportation Problem

Sebastian Sterzik and Herbert Kopfer

Working Paper No. 3

January 2012
All rights reserved by the authors
A Tabu Search Heuristic for the Inland Container Transportation Problem

Sebastian Sterzik, Herbert Kopfer

Chair of Logistics, Department of Economics and Business Studies, University of Bremen, Wilhelm-Herbst-Str. 5, 28359 Bremen, Germany

Abstract

The Inland Container Transportation Problem describes the movement of full and empty containers among a number of terminals, depots and customers in a hinterland region. A trucking company with a homogeneous fleet of trucks has to serve customers which either receive goods by inbound containers or ship goods by outbound containers. While keeping given hard time constraints the total operating time of all trucks has to be minimized. A comprehensive mathematical formulation which considers vehicle routing and scheduling and empty container repositioning simultaneously is defined. The problem is solved by an efficient tabu search heuristic. Computational experiments carried out on instances taken from the literature indicate that the proposed algorithm performs well with respect to effectiveness and efficiency.

Keywords: hinterland container transportation, vehicle routing and scheduling, empty container repositioning, integrated routing, multi-depot, pickup and delivery

1. Introduction

Considering intermodal door-to-door services (see Figure 1) international container operators are faced with a complex and dynamic transport system that comprises ocean-going services, as well as transport-services on
land through trucks. Accompanied by a steady growth of container transportation worldwide and by ever-increasing customer expectations and speed requirements, the need to improve efficiency and to reduce costs rises ([1]; [2]). Although the transportation distances at land by trucks are very short compared to the maritime transportation by ships, the total costs per TEU (twenty-foot equivalent unit) are relatively high [3]. However, within the last decades the research for coordination in the hinterland of seaports has received limited attention while the coordination of containers has been studied extensively in port-to-port business [4]. Based on the fact that ports try to enhance more and more the quality of their transportation chains to stay competitive compared to other ports, optimizing hinterland transport services including container truck transportation and empty container repocusing have been becoming an important field in recent years [5].

This paper presents and solves the Inland Container Transportation Problem (ICT) which deals with the truck transportation in a local area. In detail, we face a container drayage operation, i.e. the land-based movements of containers by trucks between a number of customers, multiple depots and multiple terminals. While keeping time constraints, a trucking company has to serve its customers by providing and/or retrieving full and empty containers. The problem, thereby, not only entails resource planning of the fleet of vehicles, but also considers the allocation of empty containers between the locations.

Several authors have stressed the growing importance of hinterland truck transportation in the last few years. Imai et al. [6] addresses a full truck-
load problem and defines it as vehicle routing problem with full containers (VRPFC). By using a subgradient heuristic based on a Lagrangian relaxation they can identify near optimum solutions. An integer programming model for a real-world case of an Italian container trucking company is given by Coslovich et al. [7]. The authors simplify the solution process regarding time efficiency by decomposing the problem into three subproblems according to the different types of costs resulting in the defined problem. A full truck-load pickup and delivery-problem (FTPDPTW) was defined by Caris and Janssens [8] for the pre- and end-haulage of intermodal transport chains. As a solution approach the authors used a simple local search heuristic to improve an initial solution. Jula et al. [9] propose an asymmetric multi-traveling salesman problem with time windows (m-TSPTW) with social constraints to model the container movement by trucks in the hinterland of seaports. The authors have implemented a two-phase exact algorithm based on dynamic programming as well as a modified genetic algorithm to solve the problem. Similarly, Zhang et al. [3] model a container truck transportation problem with multiple depots, two types of customers and one terminal as a m-TSPTW in 2009. A cluster method and a reactive tabu search (RTS) algorithm have been developed to solve the problem.

The ICT as the underlying problem of this contribution has been defined by Zhang et al. [10] in 2010. The authors extend the setting of Zhang et al. [3] by considering more than one terminal. As an approach for solving the ICT the authors used a window-partition based method (WPB method) inspired by Wang and Regan [11]. In order to find a feasible solution the WPB method uses an over-constrained mathematical model. The quality of the obtained solutions is then tested by a second model which identifies a lower bound. Zhang et al. [10] refer to empty container repositioning as an important part of the optimization process but do not include containers explicitly as a self-contained resource during the modeling process.

During this paper an integrated holistic exact mixed-integer programming (MIP) model for the ICT is presented that does not only consider vehicle routing and scheduling but also empty container repositioning as an important component of the examined problem. In a second step we present a construction heuristic as well as a local search metaheuristic and test their effectiveness and efficiency by comparing the results of several data sets with the best known solutions published in Zhang et al. [10]. As a construction heuristic, a modified implementation of the well-known Clark & Wright Savings Algorithm [12] is developed for the ICT. The obtained solution is then
improved by a tabu search heuristic which is specifically adapted to the needs of the ICT. The computational experiments carried out on instances taken from the literature indicate that the proposed algorithm outperforms the existing heuristics in most instances.

The remaining content of this paper is organized as follows. The ICT is described in Section 2 and formulated as a comprehensive mixed-integer model in Section 3. In Section 4 the modified savings algorithm and the Tabu Search Heuristic are developed. The implemented heuristics are then tested and analyzed on known test instances from the literature. The paper is concluded in Section 6.

2. Problem Description

The ICT refers to a local region in which full and empty containers have to be moved between different locations. In detail, we consider a hinterland of at least two terminals, a number of customers and several depots belonging to the trucking company. Regarding the terminals not only seaports can be enclosed within the hinterland region. Further terminal types can be given by rail yards or river ports. Figure 2 shows a simplified ICT with only one depot and one terminal. The company’s transportation requests can be separated into those requiring the transportation of inbound containers and those referring to outbound containers. Containers located at a terminal that need to be moved to their destinations in the hinterland are called inbound containers. Reversely, containers located in the hinterland that need to be delivered to a terminal are called outbound containers. The defined container terms derive from the well-known research field of inbound and outbound logistics. Furthermore, we distinguish two types of customers. On the one hand, shippers offer freight which is to be transported to a foreign region via the terminals. A full container which has to be transported from a shipper location to a terminal is defined as outbound full (OF) container. As stated, this transportation request is defined as outbound full since a full container needs to be moved from the hinterland to a terminal. On the other hand, receivers require the transport of their goods from an outside region via a terminal. The full container which has to be transported from a terminal to a receiver is called inbound full (IF) container. Obviously, both transportation tasks lead to an empty container repositioning problem. Firstly, before an OF task can be handled, a shipper requires an empty container to fill its freight into it. The origin of this empty container must be determined during
Due to the imbalance between import- and export-dominated areas, we need to take care of outbound empty (OE) or inbound empty (IE) containers which either have to be moved to a terminal or derive from it. The origin of an OE container within the hinterland (i.e. which container to take for the OE process) and vice versa the destination of an IE container is not given in advance and, thus, has to be determined during the solution process. Considering an import-dominated area, a surplus of empty containers is available in the defined hinterland related to this area. Therefore, these supplemental empty transportation resources must be moved to export-dominated regions as OE containers via the terminals. The possible origins of these containers are the locations at which empty containers accrue. Within the ICT these places are the depot and the receiver locations after an IF container is emptied. Besides, in an export-dominated area, a lack of empty transportation resources arises and leads to necessary transportations of empty containers from different regions via the terminals to the hinterland. I.e. the trucking company needs to move empty containers from the terminals to locations at which empty containers are required. If there is no shipper node which needs an empty container, there is the possibility to store the containers temporarily at a depot. Usually, we observe regions in which solely IE or OE transportation tasks have to be considered. However, because of the intransparency of the entire set of processes, one can obtain regions that consider terminals with OE transportation tasks and other terminals with IE trans-
Table 1: Origins and Destinations of the transportation types

<table>
<thead>
<tr>
<th>Origin</th>
<th>Inbound Full (IF)</th>
<th>Outbound Full (OF)</th>
<th>Inbound Empty (IE)</th>
<th>Outbound Empty (OE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination</td>
<td>Receiver</td>
<td>Terminal</td>
<td>Shipper/Depot</td>
<td>Terminal</td>
</tr>
<tr>
<td>/Other terminal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

According to Zhang et al. [10]

portation tasks. Hence, empty containers can then be interchanged between these terminals. The possible origin and destination locations of an IE and OE transportation request, as well as the defined pickup and delivery nodes of an IF and OF transportation request can be seen in Table 1.

To complete the problem description, it has to be marked that the company serves its customer requests using a homogeneous fleet of vehicles. At every depot, a specified number of vehicles can be parked and, moreover, the depots are defined as repositories for an infinite number of empty containers. Each vehicle starts its route at its corresponding initial depot and ends the route at the depot that is chosen by minimizing a vehicle’s total operating time. Within a route the vehicles must keep the time windows and consider the service times at the customer nodes and at the terminal vertices, respectively. It has to be marked that a vehicle can start and end its route at a depot at any time; i.e. time windows at these nodes do not have to be considered. Since we assume 40-feet-containers, a vehicle can only move one container at a time. The underlying problem leads to a deterministic full truckload problem for a given period. By knowing all transportation tasks in advance the objective is to minimize the total operating time.

3. Model Formulation

3.1. Introduction to the Model

The ICT is characterized by pairs of pickup and delivery locations whereat locations of a single pair have to be served by the same vehicle during given time windows at the terminal and customer locations. Thus, it has many similarities to the pickup and delivery problem with time windows (PDPTW) (see e.g. Parragh et al. [13]). The main differences between the ICT and the
PDPTW can be seen in two factors. Firstly, with respect to the ICT, a vehicle’s end depot depends on the obtained route; i.e. a vehicle must not return to a defined depot but can minimize the operating time by driving to the nearest repository. Secondly, not all pickup and delivery nodes are given in advance. Due to the containers as transportation resources we have to find the optimal destinations for empty containers which arise either at the terminals as IE containers or at a receiver location after IF containers are emptied. Moreover, the origin of an OE container has to be determined during the search process.

The resulting problem leads to an integrated model which does not only consider vehicle routing and scheduling but also simultaneously the allocation of empty containers. By considering the containers as scarce transportation resources which have to be routed and scheduled in order to fulfill the given freight requests, it is possible to determine within the mathematical model:

- where the IE containers and the empty containers obtained after IF transportation requests should be delivered,
- at which location empty containers should be picked up for OF and OE transportation requests, and
- in which order and by which truck the containers’ transportation tasks should be carried out.

The integrated model is based on the directed graph $G = \{V, A\}$ whereas $V$ describes the node sets and $A = \{(i, j) \mid i, j \in V\}$ denotes the arc set. A detailed description of the different node types, the arcs and the mathematical model is given in the following.

3.2. Node Sets, Variables and Parameters

The set of nodes $V$ consists of customer node set $V_C$, terminal node set $V_T$ and depot node set $V_D$. $V_C$ is defined by the shippers $V_S = \{1, \ldots, s\}$ and the receivers $V_R = \{s+1, \ldots, s+r\}$ whereas $n = s+r$ determines the total number of customers. Since for all IF and OF transportation requests, the pickup and delivery node are explicitly given by the input data, every customer node has its corresponding terminal node. In case of an OF transportation request this means that, after a shipper $i \in V_S$ has been served by a vehicle, the full container has to be moved to terminal node $(i + n) \in V_{TOF}$. In case of an IF transportation request, a full container has to be moved from terminal
Table 2: Definition of service time

<table>
<thead>
<tr>
<th>Container Type</th>
<th>Pickup Node</th>
<th>Delivery Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outbound Full (OF)</td>
<td>(l + p_i + l)</td>
<td>(l)</td>
</tr>
<tr>
<td>Outbound Empty (OE)</td>
<td>(l)</td>
<td>(l)</td>
</tr>
<tr>
<td>Inbound Full (IF)</td>
<td>(l)</td>
<td>(l + p_i + l)</td>
</tr>
<tr>
<td>Inbound Empty (IE)</td>
<td>(l)</td>
<td>(l)</td>
</tr>
</tbody>
</table>

\(l\)=time for the picking up/dropping off operation of a container  
\(p_i\)=time for the packing/emptying process of a container at node \(i\)

node \(i \in V_{TI}\) to its corresponding receiver location \((i - n) \in V_R\). Therefore, \(V_T\) is defined through the nodes \(\{n + 1, \ldots, n + s, \ldots, 2n, \ldots, 2n + e_i, \ldots, v\}\) whereas node sets \(V_{TIE} = \{2n + 1, \ldots, 2n + e_i\}\) and \(V_{TOE} = \{2n + 1, \ldots, v\}\) illustrate the set of IE and OE containers which have to be transported. Additionally, depot set \(V_D\) is subdivided into the start and end depot node sets \(V_{D_s} = \{v + 1, \ldots, v + d\}\) and \(V_{D_e} = \{v + d + 1, \ldots, v + 2d\}\). Since each depot can constitute a start or end location for a vehicle, we doubled the nodes so that e.g. nodes \(v + 1\) and \(v + d + 1\) describe the same depot.

A vehicle \(k \in K = \{1, \ldots, m\}\) has to start from its corresponding depot \(d_{k}^{ru}\) where \(m_{d_{k}^{ru}}\) vehicles are initially located. As stated, we do not only regard the vehicles but also the containers as being transportation resources to be considered. The container set \(c \in C = \{1, \ldots, r + e_i + e_a\}\) is defined through the containers’ possible starting locations. I.e. a container arises either from the terminal as an IF or IE container or it starts its path from the depot as an additional empty container. The service of node \(i \in V_C \cup V_T\) can only be assured if the operating vehicle keeps the specified time window \([a_i/b_i]\). Thus, a vehicle has to arrive at location \(i\) before time \(b_i\). However, arrival before \(a_i\) is allowed and leads to waiting time for the truck.

The service time \(s_i\) depends on the container type and on the pickup/delivery location. As shown in Table 2 the service consists of several activities. These activities are the picking up/dropping off of a container and the packing/emptying process performed by a shipper or receiver. E.g. a truck has to pick up an IE container at the terminal and drop the container off at the delivery location. Thereby, both activities take \(l\) minutes. Regarding an OF or IF transportation request, we also have to consider \(p_i\) minutes for the packing or emptying process at the shipper or receiver location \(i\). The time
for this process depends on the goods transported in a container. The goods to be transported require a defined time in which the customer or terminal personnel, respectively, can pack or empty the container.

A trivial problem occurs if nodes $i$ and $j$ of a traversed arc $(i, j)$ mark the same customer location and if, additionally, node $i$ determines the delivery location of an IF transportation request and node $j$ declares the pickup location of an OF request. In this case we drop off the IF container at its delivery location and wait until the emptying process is finished. Since the obtained empty container can immediately be used for the filling process of the OF transportation request at this location, the picking up and dropping off of an empty container is redundant and is thus omitted.

To sum it up, the following parameters, variables and node sets have to be defined:

$d$ : Number of depots

$n = s + r$ : Number of all customers
- $s$: Number of shippers
- $r$: Number of receivers

$e_i$ : Number of IE containers

$e_o$ : Number of OE containers

$e_a$ : Number of additional empty containers originating from the depots

$v = 2n + e_i + e_o$ : Number of all customers and terminal nodes

$m$ : Number of vehicles

$m_i$ : Initial number of vehicles at depot $i$

$d_{kru}^u$ : Start depot of vehicle $k$

$t_{ij}$ : Travel time from node $i$ to $j$

$s_i$ : Service time at node $i$

$[a_i/b_i ]$: Time window of node $i$

$M$ : Sufficiently big constant
\[ V = V_C \cup V_T \cup V_D : \text{All node sets} \]

\[ V_C = V_S \cup V_R : \text{Customer node sets} \]
- \( V_S = \{1, \ldots, s\} \): Shipper node set
- \( V_R = \{s + 1, \ldots, n\} \): Receiver node set

\[ V_T = V_{TOF} \cup V_{TIF} \cup V_{TIE} \cup V_{TOE} : \text{Terminal nodes (corresponding to the number of customers and IE/OE containers)} \]
- \( V_{TOF} = \{n + 1, \ldots, n + s\} \): OF terminal nodes
- \( V_{TIF} = \{n + s + 1, \ldots, 2n\} \): IF terminal nodes
- \( V_{TIE} = \{2n + 1, \ldots, 2n + e_i\} \): IE terminal nodes
- \( V_{TOE} = \{2n + e_i + 1, \ldots, v\} \): OE terminal nodes

\[ V_D = V_{Ds} \cup V_{De} : \text{Start and end depot nodes:} \]
- \( V_{Ds} = \{v + 1, \ldots, v + d\} \): Start depot nodes
- \( V_{De} = \{v + d + 1, \ldots, v + 2d\} \): End depot nodes

\[ K = \{1, \ldots, m\} : \text{Set of vehicles} \]

\[ C = \{1, \ldots, r + e_i + e_a\} : \text{Container set (corresponding to the number of IF/IE transportation requests and additional empty containers)} \]

### 3.3. Arcs

Usually, for each two distinct locations \( t_{ij} \) represents the driving time from location \( i \) to location \( j \). However, in some cases an arc \( (i, j) \in A \) also includes a detour to the depot and the time for picking up or dropping off of an empty container. These cases can be obtained if a truck has to serve two inbound or outbound transportation requests in succession as it can be seen in Table 3. I.e. a vehicle which is disposed for an IF/IE request after it just served an IF/IE request needs to drive to the nearest depot to drop off the transported empty container at first. Afterwards it can serve the assigned first location of the second transportation request. Besides, a vehicle which shall handle two outbound (OE/OF) requests in sequence needs to pick up an empty container at the depot with the minimum total driving distance before it can attend the second transportation request.
Table 3: Special cases for transfer time $t_{ij}$

<table>
<thead>
<tr>
<th>$i \in V_R \cup V_{TIE}$</th>
<th>$j \in V_{TIF} \cup V_{TIE}$</th>
<th>$\min_{d \in V_D} (t(i,d) + t(d,j)) + l$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i \in V_{TOF} \cup V_{TOE}$</td>
<td>$j \in V_S \cup V_{TOE}$</td>
<td>$\min_{d \in V_D} (t(i,d) + t(d,j)) + l$</td>
</tr>
</tbody>
</table>

3.4. Mathematical Model

For the mathematical model describing the ICT the following decision variables are used:

$x_{ijk} = \begin{cases} 
1 & \text{if vehicle } k \text{ drives from node } i \text{ to } j \\
0 & \text{else}
\end{cases}$

$y_{ijc} = \begin{cases} 
1 & \text{if container } c \text{ is carried from node } i \text{ to } j \\
0 & \text{else}
\end{cases}$

$T_{ik}$: Arrival time of vehicle $k$ at node $i$

$L_{ic}$: Arrival time of container $c$ at node $i$

$$\min z = \sum_{i \in V_D} \sum_{k \in K} T_{ik} - \sum_{k \in K} T_{(v+d^{fu})k}$$

$$\sum_{j \in V} \sum_{c \in C} y_{ijc} = 1 \ \forall i \in V_C \cup V_{TIF} \cup V_{TIE}$$  \hspace{1cm} (2)

$$\sum_{i \in V_D} \sum_{j \in V} \sum_{c \in C} y_{ijc} = e_a$$  \hspace{1cm} (3)

$$\sum_{i \in V} \sum_{j \in V_{TOF} \cup V_{TOE} \cup V_D} y_{ijc} = 1 \ \forall c \in C$$  \hspace{1cm} (4)

$$\sum_{j \in V_S \cup V_D} \sum_{c \in C} y_{ijc} = 1 \ \forall i \in V_{TIE}$$  \hspace{1cm} (5)

$$\sum_{i \in V_R \cup V_D} \sum_{c \in C} y_{ijc} = 1 \ \forall j \in V_{TOE}$$  \hspace{1cm} (6)
\[
\sum_{c \in C} y_{i(-n)c} = 1 \quad \forall i \in V_{TF}
\]

\[
\sum_{c \in C} y_{i(n)c} = 1 \quad \forall i \in V_S
\]

\[
\sum_{j \in V} y_{jic} - \sum_{j \in V} y_{ijc} = 0 \quad \forall i \in V_C, c \in C
\]

\[
L_{jc} \geq L_{ic} + t_{ij} + s_i - M(1 - y_{jic}) \quad \forall i, j \in V, c \in C
\]

\[
\sum_{j \in V} \sum_{k \in K} x_{ijk} = 1 \quad \forall i \in V_C \cup V_T
\]

\[
\sum_{j \in V} x_{(v+d_{ik})jk} = 1 \quad \forall k \in K
\]

\[
\sum_{i \in V} \sum_{j \in V_D} x_{ijk} = 1 \quad \forall k \in K
\]

\[
\sum_{j \in V} \sum_{k \in K} x_{ijk} \leq m_i \quad \forall i \in V_D
\]

\[
\sum_{j \in V} x_{ijk} - \sum_{j \in V} x_{ijk} = 0 \quad \forall i \in V_C \cup V_T, k \in K
\]

\[
T_{jk} \geq T_{ik} + t_{ij} + s_i - M(1 - x_{ijk}) \quad \forall i, j \in V, k \in K
\]

\[
a_i \leq T_{ik} \leq b_i \quad \forall i \in V_C \cup V_T, k \in K
\]

\[
\sum_{k \in K} x_{ijk} \geq y_{jic} \quad \forall i \in V_S \cup V_R \cup V_T, j \in V, c \in C
\]

\[
T_{ik} = L_{ic} \quad \forall i \in V_C \cup V_T, k \in K, c \in C
\]

\[
x_{ijk}, y_{ijc} \in \{0, 1\} \quad \forall i, j \in V, k \in K, c \in C
\]

\[
T_{ik}, L_{ic} : \text{real variables} \quad \forall i \in V, k \in K, c \in C
\]

The objective function deals with the minimization of the total operating and waiting time of the trucks. Thereby, the presented model is separated into three components. While the first component marks the definition of the containers’ flows ((2)-(10)), the second component characterizes the vehicles’ routes ((11)-(16)). The main component of the integrated model is
given through equations (17) and (18) which assure the interlinking of the transportation resource and the mean of transport.

In detail, restriction (2) ensures that every customer node is visited exactly once by a container and, moreover, that the origin of an IE container and of an IF container are the corresponding IE and IF terminals, respectively. Restriction (3) assures that $e_a$ containers are carried from their repository to any other node. In case a container is carried directly to an end depot, the container’s route will be void and, thus, not existent. A container’s end point can either be an outbound terminal or an end depot node marked by (4). While (5) and (6) define the possible pickup and delivery locations of OE and IE transportation requests, respectively, equations (7) and (8) declare the exact corresponding location of an IF or OF transportation request. The route and time continuity of a container are given by (9) and (10).

A vehicle’s route is characterized by its given initial depot as the start point ((12)) and an end depot which minimizes the total operational time of the vehicle’s specified route ((13)). Within a route every customer and terminal node has to be visited exactly once by a vehicle, stated by (11). (14) guarantees that the truck limit of each depot is not exceeded. In analogy to the containers’ restrictions (9) and (10), equations (15) and (16) assure the time and route continuity of a vehicle. Thereby, a vehicle has to visit a node within the defined time window, given by (17).

Since a container cannot drive on its own, it has to be ensured that a vehicle covers the containers’ flow. The interlinking of the containers’ flows and the vehicles’ routes can be assured by restriction (18). I.e. the flows of the containers are covered but the vehicles have the possibility to interrupt these flows and e.g. use different "untaveled" arcs. Obviously, if a vehicle moves a container, both have to leave a node at the same time ((19)).

4. Solution Methodology

4.1. Overview of the Proposed Methods

The PDPTW can be considered as a subproblem of the ICT which, moreover, can be classified as an extension of the famous vehicle routing problem with time windows (VRPTW) (see e.g. Cordeau et al. [14]). Since both problem types are known to be NP-hard, the ICT can also be characterized as NP-hard. Only relatively small instances of the underlying problem can be solved to optimality with e.g. the help of our proposed mathematical model. To be capable to solve bigger test instances in an efficient way we
propose an algorithm which is based on tabu search [15]. Tabu search is a local-search metaheuristic that modifies a given solution $s$ to find the best solution in its neighbourhood $N(s)$. By using a memory structure, cycling, i.e. revisiting a solution again and again in a loop of the search trajectory, can be banned from the solution space for a defined number of iterations.

In what follows, we first give some basic configurations for applying the proposed heuristics, followed by a modification of the Clark & Wright-savings algorithm which is used for constructing an initial solution. We then give a detailed description of the Tabu Search Heuristic for the ICT.

### 4.2. Basic Configurations

Considering the requests of the ICT one can distinguish two basic types (see Figure 3): While the full container transportation requests always comprise two locations (origin and destination location) the empty container transportation requests are only defined through one location (origin or destination location). This is an important difference since the local-search operators as well as operators of the construction heuristic handle with requests being part of routes. Due to a better comprehension of the heuristics we will mainly talk about requests instead of the comprised locations of the requests in the following. However, considering the distance matrix, we do not merge the nodes of a full container transportation request into one node as it can be seen in Zhang et al. [10].

According to the setting of the problem type it can be assumed that in most data sets there exist many transportation requests which can not succeed one after another. Firstly, the ICT deals with time restrictions which certainly lead to the fact that many routes turn out to be infeasible. A second reason to exclude solutions from the set of attractive solutions is due to different types of transportation requests. If two inbound or outbound requests, respectively, have to be served in succession the operating truck loses a lot of time since he has to interrupt his route by dropping off/picking

<table>
<thead>
<tr>
<th>Type 1</th>
<th>Type 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image.png" alt="Diagram" /> Outbound Full</td>
<td><img src="image.png" alt="Diagram" /> Outbound Empty</td>
</tr>
<tr>
<td><img src="image.png" alt="Diagram" /> Inbound Full</td>
<td><img src="image.png" alt="Diagram" /> Inbound Empty</td>
</tr>
</tbody>
</table>

Figure 3: Request Types
up an empty container. These two factors lead to relatively small routes that comprise only a few requests. A confirmation for this assumption can be found in Zhang et al. [10] where some typical routes of a test instance for the ICT are shown. Hence, we exclude the requests that can not succeed one after another and define a distance matrix for each transportation request according to the travel distances between requests and the corresponding time windows. Thereby, the solution space can be enormously reduced and, in consequence, the computational time can be decreased as well.

4.3. Modified Savings Algorithm for the ICT

The construction of an initial solution is based on the famous Clarke & Wright savings algorithm [12] which was introduced for the vehicle routing problem (VRP). We modified the algorithm to consider, additionally, multiple depots and terminals, different customer types as well as time constraints. One of the biggest differences to adapt the algorithm of Clarke & Wright marks the request assignment to a depot which was introduced by Tillman [16] for the multi depot vehicle routing problem (MDVRP). Thereby, a request $r \in R = \{1...n + e_i + e_o\}$ is disposed to its nearest depot $d^r_{req}$. The outline of the algorithm is as follows:

**Algorithm 1 Savings Algorithm**

1: Request $r \in R$ is assigned to the nearest depot $d^r_{req}$;
2: Each request is served by exactly one truck;
3: Savings for all route pairs of the same depot are computed as follows:

$$saving_{ij} = t_{id^r_{req}} + t_{d^r_{req}j} - t_{ij}, \forall i \in V^T_{OF} \cup V^T_{IE} \cup V^T_{OE} \cup V_R, j \in V_S \cup V^T_{IE} \cup V^T_{OE} \cup V^T_{IF}, r \in R;$$
4: Route pairs for each depot are sorted in descending order of the savings;
5: From the top of the sorted list the given routes are merged into one if the established route is feasible and if this can be done without deleting a previously defined connection between two requests;

The adapted savings algorithm is capable to establish a viable initial solution. Nevertheless, its shortcoming is the generation of as many routes as necessary without keeping a depot’s truck limit.

4.4. Tabu Search Heuristic

Starting from the solution $s^*$ generated by the construction heuristic, the Tabu Search Heuristic moves at each iteration to the best non-tabu solution
in the neighbourhood of $s^\ast$. The decision which solution $s \in \mathcal{N}(s^\ast)$ is chosen next depends on the following criteria: objective function, route reduction, request selection, local-search parameters, tabu tenure and aspiration criteria, and finally, the intensification and diversification strategies.

**Objective Function.** Calculating the objective function to minimize the trucks’ total operating time is a problem of its own. The solution space for determining the trucks’ optimal arrival times for a given route is relatively large since it depends on the travel times, the number of requests in a route and mainly on the time windows’ amplitude of the corresponding customers and terminals. We propose a heuristic approach to reduce waiting times and, thus, to approximate the best start and end times of route $k \in K$.

First of all, we check if a route is feasible. Thereby, we allow a surplus of waiting time since $T_{ik}$ should always be defined as the minimal arrival time. I.e. a truck traversing arc $(i, j)$ arrives at node $j$ at time $a_j$ where applicable. Otherwise $T_{jk}$ is defined by $T_{ik} + s_i + t_{ij}$ if $a_j < T_{ik} + s_i + t_{ij} \leq e_j$. Assuming that the route of vehicle $k$ is feasible, we, thereafter, try to reduce unnecessary waiting times. The determined arrival time of the last customer in the route of vehicle $k$ is used to improve recursively the arrival times of the prior customers. Since the ICT does not allow multiple deployments of trucks, vehicle $k \in K$ is used as a synonym for the route of $k$ in the following. If $k_l$ determines the last position of $k$, $k_{l-1}$ is then defined as $k_l - s_{k_l-1} - t_{k_l-1}k_l$ or as $e_{k_{l-1}}$ if $k_l - s_{k_l-1} - t_{k_l-1}k_l \geq e_{k_{l-1}}$. The heuristic then uses the determined arrival time successively to calculate the remaining arrival times until $T_{d_{\overline{k}}}k_1$ is defined.

**Route Reduction.** Since the savings algorithm can lead to an initial solution that comprises more than the available trucks situated at a depot, we penalize every route which exceeds the truck limit $m_i$ of depot $i$ with the additional costs $cost_{pen}$. Let $p(s)$ determine the summation of all penalty costs which have to be added to the objective value $f(s)$ in order to reach a feasible solution. The algorithm, thereafter, forbids the excess of the defined truck limit and allows only feasible solutions for the remaining iterations.
Algorithm 2 Request Selection

1: \( \text{saving}_r = f(s) - f_{-r}(s) ; \)
2: \( \text{Requests are sorted in list } L \text{ in descending order of the savings; } \)
3: \( \text{saving}_{\text{sum}} \leftarrow 0 ; \)
4: \( \text{saving}_{\text{total}} \leftarrow \sum_{r \in R} \text{saving}_r ; \)
5: \( \Xi \leftarrow \text{random number in the interval } [0, 1] ; \)
6: \( \text{for } x \in L \text{ do } \)
7: \( \text{if } \Xi < \text{saving}_{\text{sum}} + \text{saving}_x / \text{saving}_{\text{total}} \text{ then } \)
8: \( r^* \leftarrow x ; \)
9: \( \text{Request } x \text{ is removed from } s ; \)
10: \( \text{Algorithm terminates; } \)
11: \( \text{end if } \)
12: \( \text{saving}_{\text{sum}} \leftarrow \text{saving}_{\text{sum}} + \text{saving}_x / \text{saving}_{\text{total}} \)
13: \( \text{end for } \)

Request Selection. We aim to identify the requests which seem to be assigned to an inappropriate position within a route or to an unsuitable route of the solution and should be replaced through other requests or inserted into other routes, respectively. Therefore, we define a remove saving \( \text{saving}_r = f(s) - f_{-r}(s) \) for each request \( r \in R \) located in the current solution \( s \in S \). Beside the usual cost function \( f(s) \), the term \( f_{-r}(s) \) defines the costs of \( s \) without request \( r \). To obtain better objective values the request with the highest savings should be selected for the local search operators. The emerging risk of cycle situations where always the same requests are chosen e.g. due to the fact that requests are located at the border of the observed hinterland should be avoided by including a certain degree of randomization. Thereby, the requests with higher savings are not always chosen but get a higher probability value for the selection process. The request selection’s execution sequence can be seen in the pseudocode of Algorithm 2.

Local-Search-Operators. The neighbourhood of a current solution is composed of all solutions that can be reached by applying one of the local-search operators. Three types of moves are used in the underlying tabu search approach: \textit{cross-operator}, \textit{insertion-operator} and \textit{combine-operator}. While the last operator does not need an input operation and, therefore, acts autonomously the first two move types require the proposed \textit{request selection} process beforehand.
• The *insertion-operator* removes the selected request \( r^* \) of the request selection process from its route and inserts it in another route or at another place in its current route.

• The *cross-operator* swaps the selected request \( r^* \) from its route and exchanges it with request \( r \in R \setminus \{ r^* \} \).

• The *combine-operator* tries to reduce the number of routes by reinserting the elements of each short route into another route. Thereby, a short route is defined as a route which comprises less than \( y \) requests.

As it can be seen in the pseudocode of Algorithm 3, the insertion- and cross-operator are applied in each iteration while the utilization of a combine-operator depends on the probability value \( \alpha \). Due to the fact that applying this operator takes a lot of computational time one has to find an adequate value for \( \alpha \) which does not impair the algorithm’s efficiency. After applying an operator the best non-tabu solution in \( N(s^*) \) becomes the new current solution \( s \). Besides, the tabu list \( T \) has to be updated.

### Algorithm 3 Operator Selection

1: if \( \Xi < \alpha \) then
2:   Combine-Operator is applied;
3:   \( s \leftarrow s^* \);
4:   Tabu list \( T \) is updated;
5: end if
6: Request Selection is applied;
7: \( r^* \) is used for Cross-Operator;
8: \( s \leftarrow s^* \);
9: \( T \) is updated;
10: Request Selection is applied;
11: \( r^* \) is used for Insertion-Operator;
12: \( s \leftarrow s^* \);
13: \( T \) is updated;

*Tabu Tenure and Aspiration Criteria.* The tabu list \( T \) is constituted as a deterministic list which records requests that are removed from their routes. After the removal \( r \) is not allowed to be served by truck \( k \) for \( \Theta \) iterations.

18
An exception of applying the tabu status can be marked for the combine-operator. Hereby, we want to grab any chance to always get the best neighbourhood solution even if \( r \) is tabu for \( k \). The risk to get caught in a cycle is not given since the combine-operator is not applied in every iteration and, moreover, the other two operators would be applied before returning to this move type. However, in general a tabu status is overruled if the algorithm finds a solution which is better than any solution known so far.

**Algorithm 4 Intensification Strategy**

1. if (\( \Xi < \beta \) \& \( f(s) < (1 + \gamma) * f(s_{\text{best}}) \)) then
2. \( T = \{\} \);
3. \( \Theta \) is updated;
4. for \( r \in R \) do
5. \( r \) is used for cross-operator and leads to solution \( s_{\text{CO}}^* \);
6. \( r \) is used for insertion-operator and leads to solution \( s_{\text{IO}}^* \);
7. if \( f(s_{\text{CO}}^*) < f(s_{\text{IO}}^*) \) then
8. \( s \leftarrow s_{\text{CO}}^* \);
9. else
10. \( s \leftarrow s_{\text{IO}}^* \);
11. end if
12. while \( \text{iter}_2 < \text{iter}_{2\text{max}} \) do
13. \( \text{Operator Selection} \) is applied;
14. \( \text{iter}_2 = \text{iter}_2 + 1 \);
15. end while
16. end for
17. \( \Theta \) is updated;
18. end if

**Intensification and Diversification Strategy.** According to the probability value \( \beta \) and the quality of \( s \) which is related to \( (1 + \gamma) * f(s_{\text{best}}) \) whereas \( s_{\text{best}} \) determines the best known solution so far and \( \gamma \) is a constant parameter in the interval \([0, 1]\), the usual search process can be interrupted for an intensification strategy. Thereby, \( s \) is modified \(|R|\) times by using each \( r \) in \( s \) once for the cross- and insertion-operator, respectively. Both solutions are compared and the best solution according to the objective value is chosen. Based on this modified solution the operator selection algorithm is applied for \( \text{iter}_{2\text{max}} \) iterations. For an efficient tabu search algorithm this iteration
limit value should be defined well since the execution of the operator selection algorithm is applied for $|R| \times \text{iter}^\text{max}$ iterations. The intensification strategy is an important component of the Tabu Search Heuristic and is, therefrom, defined as an autonomous algorithm within this framework. Hence, the tabu list is restarted each time the intensification strategy is applied. As it can be seen in the pseudocode of Algorithm 4, $\Theta$ can be adapted according to the modified framework within this component.

To diversify the search, we implement a mechanism which penalizes any neighborhood solution $s^N \in N(s)$ by a factor that is proportional to the addition frequency of its attributes and a scaling factor. In detail, $q_{rk}$ describes the number of times request $r$ has been added to route $k$ during the search process. The intensity of the diversification process can be adjusted by the parameter $\lambda$. Thus, unless $f(s^N) < f(s^\text{best})$ penalty term $\lambda \times q_{rk}$ is added to the total solution costs $f(s^N)$. The illustrated diversification strategy is a modification of the mechanism used in Taillard [17].

Algorithm 5 Framework of the Tabu Search Heuristic

1: Solution of Savings-Algorithm is used as $s^\text{best}$;
2: $\Theta \leftarrow$ number of tabu iterations;
3: $\alpha \leftarrow$ probability value;
4: $\beta \leftarrow$ probability value;
5: $\gamma \leftarrow$ number in the interval $[0, 1]$;
6: $\lambda \leftarrow$ probability value;
7: while $p(s) > 0$ do
8: Operator Selection is applied;
9: end while
10: $\alpha$ is updated;
11: while $\text{iter}_1 < \text{iter}_1^\text{max}$ do
12: Operator Selection is applied;
13: $\Xi$ is updated;
14: Intensification Strategy is applied;
15: $\text{iter}_1 = \text{iter}_1 + 1$;
16: end while

Global Description. The Tabu Search Heuristic comprises an initial phase and a main phase. The initial phase is only executed if the savings-algorithm does not succeed to find a feasible solution according to the limits of available
trucks at the depots. In consequence, the initial phase turns to the main phase if a feasible solution is found, i.e. \( p(s) = 0 \). To take any chance to get more rapidly to a feasible solution it is advisable to define a higher probability value for \( \alpha \) compared to the main phase. The numerical experiments have shown that, especially, in the initial phase of the algorithm a frequent use of the **combine-operator** is more efficient to reduce the number of routes. Attention should be paid to the **intensification strategy** in the main phase. It has proven to be very successful for the ICT but also very time-consuming if it is applied too often. Thus, a good balance of the parameters \( \beta \) and \( \gamma \) have to be determined. The main phase terminates if iteration \( \text{iter}^{\text{max}}_1 \) is reached. The outline of the Tabu Search Heuristic can be seen in Algorithm 5.

5. **Computational Results**

To test the performance of the heuristic, the data sets of Zhang et al. [10] are used. The authors generated 20 realistic-sized instances for the ICT which include five depots, three terminals and 75 transportation requests. Usually, 10 trucks per depot are considered which have to serve the customers within a time horizon of one day. The proposed algorithms have been implemented in Java 6 on an Intel Core i7, 3.2 GHz PC with 12GB system memory. After several experiments with the algorithm to characterize the tradeoff between computation time and solution quality, the following parameters for the main component of the Tabu Search Heuristic were used: \((\text{iter}^{\text{max}}_1, \text{iter}^{\text{max}}_2, \alpha, \beta, \gamma, \lambda, \Theta) = (1000, 10, 0.1, 0.12, 0.008, 1.5, 6)\). Since some random factors influence the heuristic’s search procedure, we tested the fluctuation of the generated solutions by applying the algorithm at least 10 times per test instance. The deviation of the obtained solutions lay in a range of < 3% compared to the best found solution of the underlying test instance and, thus, revealed the algorithm’s stability.

Taking a closer look at the resulting routes of test instance 13 Figure 4 shows four typical truck paths for the illustrated eight requests. Focusing on Route 1, one can see that a truck mostly serves an inbound request after an outbound request. In detail, the truck starts its path from the depot in the south to the shipper in the northwest to handle the OF container. In order to save time it is likely to serve a disposable IE transportation request from the same terminal if the succeeding time windows are consistent. Finally, the truck handles an OF request and drives back to the nearest depot which is - in this case - also the truck’s starting depot. Besides, Routes 2-4 show another
typical characteristic of the ICT concerning the size. While Route 3 solely includes Request 6, Route 2 and 4 comprise at the most two transportation requests. However, one has always to bear in mind the fact that an IF or OF transportation request is defined through two locations.

The generated results are compared with the best known solutions generated by the WPB method of Zhang et al. [10]. Table 4 lists the best known solutions due to these works. As can be seen the Tabu Search Heuristic works very well for the ICT. Except for instance 17, it outperforms the WPB method at all times. And even for this exceptional case, it finds a solution value which is quite near to the best-known solution (∼ 0.1%). Although
Table 4: Comparison of solution approaches on 20 test instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>Tabu Search Heuristic</th>
<th>Zhang et al. [10]</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Trucks</td>
<td>Obj. Value</td>
<td>Trucks</td>
</tr>
<tr>
<td>4</td>
<td>46</td>
<td>14995</td>
<td>46</td>
</tr>
<tr>
<td>5</td>
<td>46</td>
<td>15790</td>
<td>47</td>
</tr>
<tr>
<td>6</td>
<td>53</td>
<td>14788</td>
<td>53</td>
</tr>
<tr>
<td>7</td>
<td>50</td>
<td>15772</td>
<td>51</td>
</tr>
<tr>
<td>8</td>
<td>40</td>
<td>12791</td>
<td>41</td>
</tr>
<tr>
<td>9</td>
<td>47</td>
<td>17752</td>
<td>49</td>
</tr>
<tr>
<td>10</td>
<td>45</td>
<td>14686</td>
<td>45</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
<td>15898</td>
<td>50</td>
</tr>
<tr>
<td>12</td>
<td>42</td>
<td>12766</td>
<td>43</td>
</tr>
<tr>
<td>13</td>
<td>52</td>
<td>17671</td>
<td>54</td>
</tr>
<tr>
<td>14</td>
<td>40</td>
<td>14313</td>
<td>43</td>
</tr>
<tr>
<td>15</td>
<td>42</td>
<td>15458</td>
<td>45</td>
</tr>
<tr>
<td>16</td>
<td>42</td>
<td>14584</td>
<td>44</td>
</tr>
<tr>
<td>17</td>
<td>44</td>
<td>15976</td>
<td>44</td>
</tr>
<tr>
<td>18</td>
<td>40</td>
<td>13836</td>
<td>40</td>
</tr>
<tr>
<td>19</td>
<td>37</td>
<td>15260</td>
<td>39</td>
</tr>
<tr>
<td>20</td>
<td>34</td>
<td>13464</td>
<td>34</td>
</tr>
<tr>
<td>21</td>
<td>37</td>
<td>14337</td>
<td>39</td>
</tr>
<tr>
<td>22</td>
<td>40</td>
<td>16003</td>
<td>41</td>
</tr>
<tr>
<td>23</td>
<td>35</td>
<td>12024</td>
<td>35</td>
</tr>
<tr>
<td>Total</td>
<td>861</td>
<td>298164</td>
<td>883</td>
</tr>
</tbody>
</table>

the objective concerns the minimization of the total operating time, we also considered the number of used trucks and observed for some instances a high potential for reducing fixed costs (see e.g. instance number 14 and 15).

A comparison of the computational time needed by the addressed solution methods is difficult to realize objectively since different computers and programming languages have been used. Nevertheless, Figure 5 indicates that the proposed Tabu Search Heuristic also outperforms the WPB method of Zhang et al. [10] in terms of efficiency. The relatively long computational times of instances 6 and 13 derive from a solution space which is characterized by many "good" solutions situated close to the best found solution. In these cases, the intensification strategy is applied very often and leads to increasing computation times. However, regarding the other instances it has been demonstrated that the Tabu Search Heuristic performs very well.
6. Conclusions

The ICT describes the move of full and empty container flows among multiple depots, multiple terminals and several customers in a hinterland region. By keeping time constraints inbound as well as outbound transportation requests have to be handled by a homogeneous fleet of trucks. The objective is to minimize the total operating time of all trucks. Beside the routing of vehicles, a main challenge to solve the problem lays in the allocation of empty containers between the locations. In this contribution we presented a first holistic mathematical formulation for the ICT which handles vehicle routing and scheduling as well as empty container repositioning at the same time. On a second step, we proposed a tabu search algorithm which has been tested on 20 test instances from the literature. The results of our computational results show that the algorithm performs well with respect to effectiveness and efficiency.

The computational results indicate for some data sets a high potential for reducing fixed costs. Thus, for further studies, we will apply a modification of the Tabu Search Heuristic’s objective function to analyze the ICT according to the number of operating trucks. Since the WPB method of Zhang et al. [10] as well as the proposed Tabu Search Heuristic focus mainly on the routing of vehicles, a very interesting and also challenging approach can be the heuristic implementation of our proposed exact MIP model which considers vehicle routing and scheduling as well as empty container repositioning at the same time.
References


