Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Computational Engineering and Technical Physics
Technomatematics

Eero Siniluhta

TRUCK DOCK ASSIGNMENT WITH SEQUENTIAL STOPS IN CONTINUOUS DEMAND PROCESS

Master's Thesis

Examiners: Professor Heikki Haario
Associate Professor Ari Happonen

Supervisors: MSc (Econ.), Computer Science Janne Kuha
Associate Professor Ari Happonen
Professor Heikki Haario

ABSTRACT

Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Computational Engineering and Technical Physics
Technomatematics

Eero Siniluhta

TRUCK DOCK ASSIGNMENT WITH SEQUENTIAL STOPS IN CONTINUOUS DEMAND PROCESS

Master's Thesis

2019

44 pages, 17 figures, 6 tables, 1 appendices.

Examiners: Professor Heikki Haario
Associate Professor Ari Happonen

Keywords: optimization, transportation, logistics, branch and bound, manufacturing

The purpose of this study is to develop a practical solution to a truck dock assignment problem specific to manufacturing industry inbound articles reception. In this article a variant of Branch-and-Bound algorithm was used to find solutions to the truck dock assignment problem. The distinctive difference between a truck dock assignment scenario often found in logistics hubs and this kind of industrial setup is that here each truck may have to visit multiple docks. Based on artificially generated test inputs, test runs were carried out to evaluate the performance of the solution and to assess the suitability of the solution for similar real world uses cases.

TIIVISTELMÄ

Lappeenrannan-Lahden teknillinen yliopisto LUT
School of Engineering Science
Laskennallinen tekniikka ja teknillinen fysiikka
Technomatematics

Eero Siniluhta

KUORMANPURKUAIKATAULUTUS PERÄTTÄISILLÄ PYSÄHDYKSILLÄ JATKUVAN TARPEEN PROSESSISSA

Diplomityö

2019

44 sivua, 17 kuvaa, 6 taulukkoa, 1 liite.

Tarkastajat: Professori Heikki Haario
Tutkijaopettaja Ari Happonen

Hakusanat: optimointi, kuljetus, logistiikka, branch and bound, valmistus
Keywords: optimization, transportation, logistics, branch and bound, manufacturing

Tässä tutkimuksessa rakennettiin ratkaisu kuorma-autojen kuormanpurkuaikataulun rakentamiseksi teollisen tuotannon tavaranvastaanoton alalla. Työssä sovellettiin Branch-and-Bound algoritmin variaatiota aikatauluvaihtoehtojen rakentamiseksi. Tässä työssä käsitelty kuormanpurkuongelma eroaa esimerkiksi logistiikkakeskuksissa usein esiintyvästä ongelmasta siinä, että tässä tapauksessa kuorma-autot joutuvat usein tekemään useampia pysähdyksiä eri purkulaitureilla. Keinotekoisesti tuotettujen testisyötteiden pohjalta suoritettiin testiajoja, joiden tulosten pohjalta arvioitiin ratkaisun soveltuvuutta vastaaviin todellisiin käyttötapauksiin.

CONTENTS

LIST OF ABBREVIATIONS

Alg.	Algorithm
B_{min}	Minimum branching factor
B_{max}	Maximum branching factor
B&B	Branch and bound
BFS	Breadth-first search
$\|D\|$	Number of docks
$\|D\|_{min}$	Lower bound (inclusive) for number of docks to visit
$\|D\|_{max}$	Upper bound (exclusive) for number of docks to visit
DFS	Depth-first search
$earliestSlot$	Earliest time slot
Fig.	Figure
IT	Internal truck
JIT	Just in time
λ_i	Average number of invalid solutions for trials with at least one invalid solution
λ_s	Average number of valid solutions for trials with at least one valid solution
λ_{ta}	Average number of trucks too early for invalid solutions
μ_D	Mean deadline (latest allowed time slot)
μ_{vis}	Mean number of docks to visit per truck
MASPH	Multiple-appointment scheduling problems in hospitals
N_s	Number of trials with at least one valid solution
N_{slots}	Number of available time slots
N_{pre}	Number of extra time slots
$numSlots$	Number of time slots
$numNonAlloc$	Number of invalid time slots at the beginning of the day
OEM	Original equipment manufacturer
σ_D	Standard deviation of deadline
σ_{vis}	Standard deviation of docks to visit
$\tilde{ta}$	Median number of trucks too early for invalid solutions
t_r	Calculation time of single test run
$\|TR\|$	Number of trucks
TDC	Truck dock constraints matrix
TSP	Travelling salesman problem
R	Recursion step limit
$randType$	Type of random number generator

S	Timeslot occupancy matrix
$seed$	Seed number for the pseudo-random random number generator
ST	Set of truck specific occupancy matrics
U	Priority ordered list of trucks
V_{max}	Max number of visits per truck
VMI	Vendor managed inventory
VRP	Vehicle routing problem
WMS	Warehouse management system
XT	External truck

1 INTRODUCTION

1.1 Truck dock assignment problem

Truck dock assignment is a part of transport logistics processes where goods are loaded and unloaded to trucks or trailers from transportation hubs or warehouses. The problem of truck dock assignment arises when there is need to allocate a limited number of docks to be used by multiple delivery trucks or trailers. Now on in this paper the word *truck* is used to refer to both trucks and trailers and the distinction between trucks and trailers is explicitly made when discussing relevant operational differences between these modes of transportation.

Different variations of the truck dock assignment problem can be identified in various domains with domain and arrangement specific constraints. Cross dock terminal is an arrangement used in ground transportation hubs where there are docks for inbound and outbound material flows and the goal is to hand over and shuffle the goods from inbound trucks directly to outbound trucks minimising the need for storage in the hub [1]. One of the key activities is assignment of docks to trucks when there may be changes during the course of planning horizon [2]. A cross dock terminal is illustrated in Fig. 1.

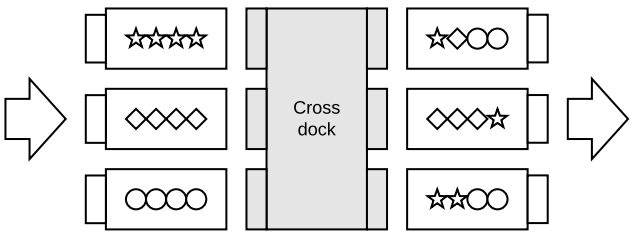

Figure 1. Cross dock terminal. Inbound trucks arrive from left and their goods are loaded to the outbound trucks on the right side with minimal delay. Illustration inspired by [3]

Air cargo terminal facilitates the transportation of goods from land based truck transports to airplanes. Air cargo terminal operators typically offer 48-hour free storage to their customers, but many shipments still arrive very close to the flight departure, which creates congestion to truck docks and the cargo handling facilities during the busiest departure hours [4]. An air cargo terminal arrangement is illustrated in Fig. 2

Maritime cargo handling in docks also involves truck arrival management problem, which

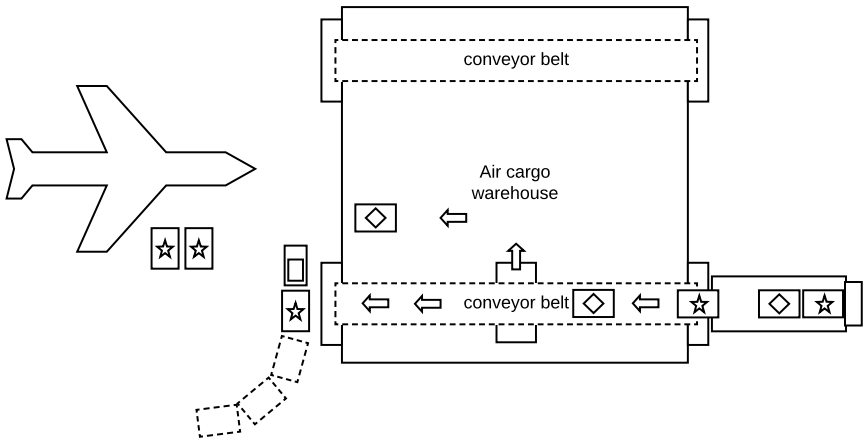

Figure 2. Air cargo terminal with warehouse. Some items are directly transferred from truck to plane, whereas some are stored in the warehouse. Cargo containers that will be loaded to the aircraft are transferred from the warehouse to the plane by a cargo dolly. Illustration was inspired by [5].

is a variant of the truck dock management problem. In a marine container terminal the process for truck operations is triggered by the arrival of a container ship, which generates need to handle both loading inbound cargo from the vessel to outbound trucks and delivery of inbound cargo from trucks to be loaded to the vessel. Arrival of a vessel thus creates large demand for ground transportation activities, and to reduce the congestion and pollution in the terminal area related to the truck traffic, terminal management tries to minimise truck service times and to manage the truck arrival times [6]. An illustration of a maritime container terminal is shown in Fig. 3.

Original equipment manufacturing (OEM) industries are also directly involved in transport logistics planning for example to shorten production time using just in time (JIT) delivery of inbound materials [8]. Industries such as automotive manufacturing may employ a milk run based approach where the customer controls the transportation arrangements of materials from variety of vendors as required by the manufacturing schedule [9]. In some industries it may be beneficial that transportation is instead managed by the vendor, for example in an arrangement where a vendor managed inventory (VMI) arrangement is used [10].

Important infrastructure services in cities, such as garbage collection and postal services, also exhibit a structure that may contain process phases with truck dock assignment problem. Garbage trucks collect material from large areas until the trucks are full and then proceed to unload to a waste processing plant. Scheduling of the truck arrivals to the

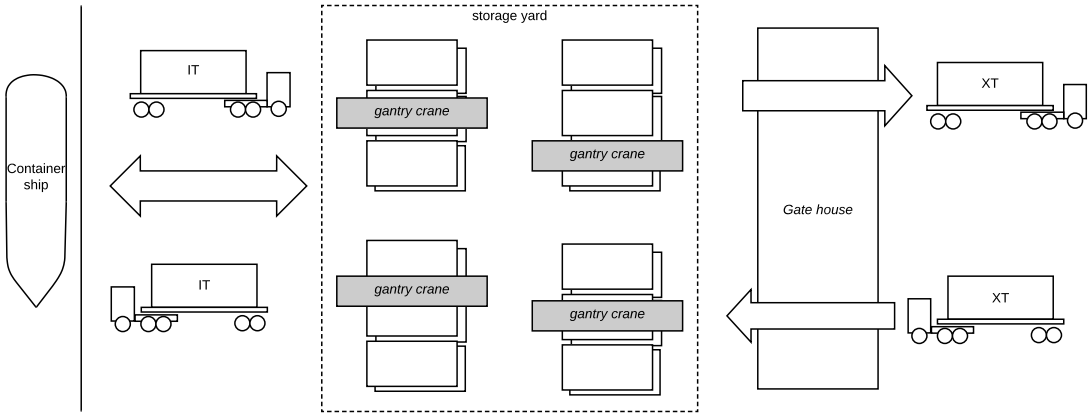

Figure 3. Maritime container terminal. External trucks (XT) transport containers to terminal while internal trucks (IT) transport containers from storage yard to container ship. Gantry cranes at the storage yard handle the loading of containers to both ITs and XTs. Illustration inspired from [7].

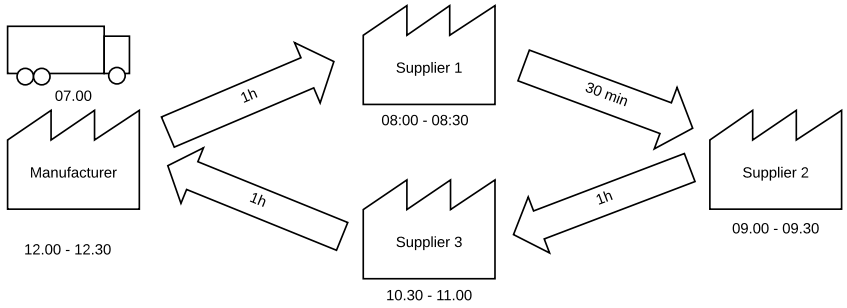

Figure 4. Milk run based transportation arrangement. A truck is sent out to collect supplies from one or more suppliers. Illustration inspired from [11].

processing plant is important to avoid congestion and waiting as there is limited number of docks for unloading the garbage and unloading itself takes time [12]. The distribution networks of the postal service industry are organized according to the hub-and-spoke paradigm, so that parcel distribution centers play a crucial role to consolidate the parcel flows to full truckloads. In these terminals, inbound trucks are unloaded at gates, shipments are identified, sorted by the central sortation conveyor system, and loaded into outbound trailers, in which they are moved toward their next destination. In this context, the scheduling of inbound trucks, which assigns a gate and a processing interval to each truck, is an essential operational decision problem [13].

1.2 Continuous demand process

Continuous demand processes are manufacturing processes that consume number of different articles necessary for building the end product with limited inventory capacity and thus a constant need to replenish the inventory. The inventory capacity could be limited due to budgetary reasons, e.g. it is desirable to keep the value of the inventory low, or it could be due to customer pressure to produce products with customer specific variations in acceptable time, thus making it impractical to keep all the articles stocked. In such process the orders for articles are placed with an outside supplier, and they arrive after a lead time, which may be constant or stochastic [14].

In this paper the process of interest is a continuous demand process where a large number of articles are received each day and consumed during the same day. However, not all articles are needed at the exactly same time, which gives some room for arranging the unloading order of the deliveries in such way that the receiving can possibly be done on the same day without needing extensive spot capacity of docks and personnel during any specific hour of the day.

1.3 Importance of truck dock assignment

Transportation is the single most expensive component of trade logistics and it plays an important role in defining whether a manufacturer can import and/or export goods in competitive way [15, 16]. Truck dock assignment is a part of many land based transportation scenarios and thus effectiveness of the truck dock assignment also affects the competitiveness of the transportation arrangement to some degree. A suboptimal solution to truck dock assignment can cause significant waiting times to other trucks, subsequently raising costs related to drivers and cargo handling and reducing the volume of material flow that can be achieved [17].

Truck dock assignment can be done manually when material flow is small, but manual handling may prove to be an obstacle for production expansion with higher material flow, thus prompting an algorithmic solution such as the one presented in this paper be taken into use.

2 BACKGROUND

2.1 Scheduling problems

Scheduling problems arise in many different domains. An example of a common scheduling problem is *appointment scheduling* for multiple people, each one having their personal calendars already filled with other appointments. Another example would be delivery time scheduling for a package delivery company for clients with different available time slots for reception.

In service systems the service provider would like to minimize costs in terms of the server's idle times, while the customers would like to be served with minimal waiting times. To accommodate these goals of the service provider and the customers, for example in case of a dentist and his patients, one may fix the arrival times of the customers beforehand in an appointment schedule [18].

One interesting scheduling problem that exhibits the same properties as the truck dock assignment regarding multiple sequential appointments are the problems of *multiple-appointment scheduling problems in hospitals* (MASPHs). In these arrangements a patient needs to visit multiple medical service providers in specific order and in a bounded time so the required services can be rendered to treat the patient effectively [19].

2.2 Branch-and-bound optimisation method

In any decision or design process, one attempts to make the best decision within a specified set of possible ones. The notion of "best" is in the eye of the beholder [20]. Optimality is a fundamental principle, establishing natural lows, ruling biologic behaviors, and conducting social activities. Therefore, optimization started from the earliest stages of human civilization [21].

Truck dock assignment problem is a special case of an NP-hard problem known as machine scheduling problem [22] . The branch-and-bound (B&B) framework is a fundamental and widely-used methodology for producing exact solutions to NP-hard optimization problems. The technique, which was first proposed by Land and Doig [23], is often referred to as an algorithm; however, it is perhaps more appropriate to say that B&B encapsulates a family of algorithms that all share a common core solution procedure. This

procedure implicitly enumerates all possible solutions to the problem under consideration, by storing partial solutions called subproblems in a tree structure. Unexplored nodes in the tree generate children by partitioning the solution space into smaller regions that can be solved recursively (i.e., branching), and rules are used to prune off regions of the search space that are provably suboptimal (i.e., bounding). Once the entire tree has been explored, the best solution found in the search is returned. [24]

3 FORMULATION OF THE PROBLEM

This paper presents an optimisation method for a truck dock assignment problem in industrial context. The subject is treated without any specific case or business domain, but inspiration for the problem definition has been drawn from earlier experiences of the author.

The industrial process is planned so that the deliveries of the articles needed for production are done as close as possible to the production moment. For efficient use of truck moving personnel and dock capacity the schedule optimisation needs to consider a few specific operational constraints: 1) Trucks may need to visit multiple docks as the carried articles may have different unloading points, 2) trucks have time deadlines specific to each dock imposed by the need times of the articles to be unloaded at each dock and 3) trucks proceed from their previous dock to next dock immediately without waiting. An example of this kind of industrial setting is illustrated in Fig. 5

Research questions:

1. Is it possible to create an unloading schedule for upcoming day automatically so that all articles would be unloaded before their respective need time during that day?
2. Can the created schedule be made so that there are empty time slots scattered over the day so that any unforeseen additions to the unloading operations during the day can likely be carried out without deviating from the schedule?
3. Can the automatic scheduling be made so that operations personnel can inspect the schedule for verification and thus trust the automatically generated schedule?

All trucks needed to be unloaded on a day are known at the morning of the day before and the question is what would be the most efficient arrangement for unloading on the date of arrival given the operational constraints. Unloading schedule for the date of arrival is planned in a time window of 4-8 hours on the day before and the schedule is not altered on day the plan is effective. There may be changes during the day, but they are handled by either extra capacity that is not part of the schedule or by flexibility afforded by the schedule itself.

This paper does not concern the separate problem of identifying which docks each truck should visit and what would be the deadline for each visit. The identification problem

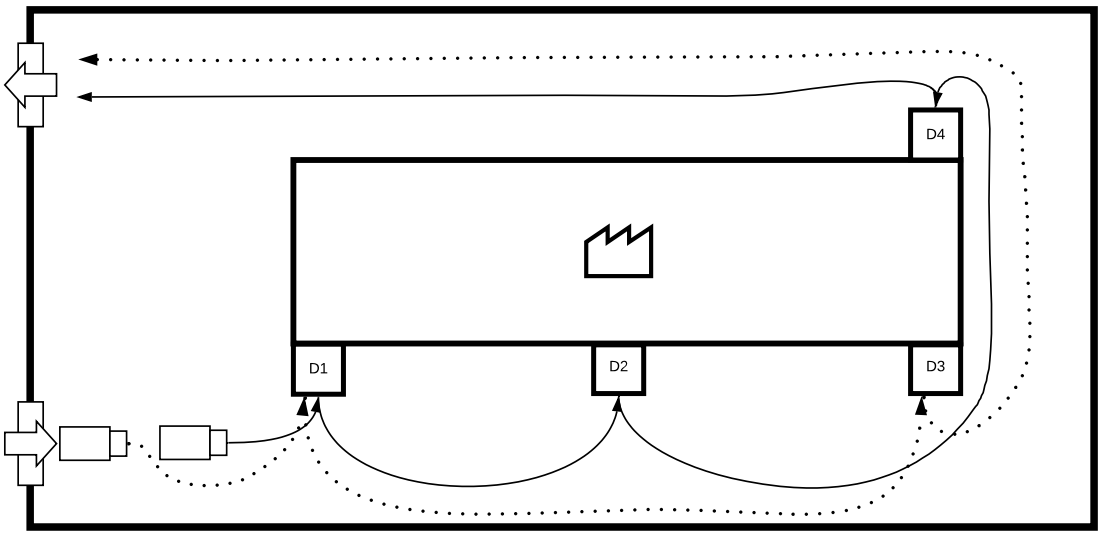

Figure 5. Industrial plant with multiple docks for receiving inbound materials. Two trucks are shown to arrive to the plant, one visiting three docks while the other visits two docks before proceeding to outbound gate.

involves uncertainties stemming from sources such as incomplete information about the trucks and the loads and, depending on the exactness of the identification of the deadline constraints per truck, deadlines may need to be constrained more than with a more precise identification method.

In addition to adhering to the constraints, an optimal schedule would have other desirable qualities: 1) The schedule should be as sparse as possible, e.g. the trucks to be unloaded should be as sparsely allocated as possible to accommodate possible changes during the effective day and 2) the representation of the schedule should be easily inspectable by the users so that it can be confidently trusted.

4 OPTIMIZATION MODEL

4.1 Constraints

Each truck has one or more deadlines for one or more docks. An example of deadlines is illustrated in Fig.6. From perspective of one truck any dock assignment pattern that satisfies the deadlines is equally suitable. Examples of three different equally valid assignments for the truck in Fig.6 are shown in Fig.7.

As it is required that there are no time gaps between dock visits, it is observed that all permutations of valid assignments fall into a bounding area of size $n \times n$ where n is the number of docks the truck must visit, when represented so that all the docks that the truck does not need to visit are removed as in Fig.7. The bounding box can be moved freely to left, giving upper limit of valid permutations $m * n!$, where m is the number of time steps the bounding box can be moved to left from the rightmost position that still has valid assignments.

Furthermore, few more assumptions are made: Number of trucks that can be unloaded is only limited by capacity of unloading dock. Each unloading dock can have one or more trucks being unloaded at the same time, depending on the dock. All trucks inbound for the day have to be unloaded. Each article has a designated unloading dock. One truck may need to visit multiple docks depending on the articles it is carrying. Each truck needs only one time slot for unloading in any dock.

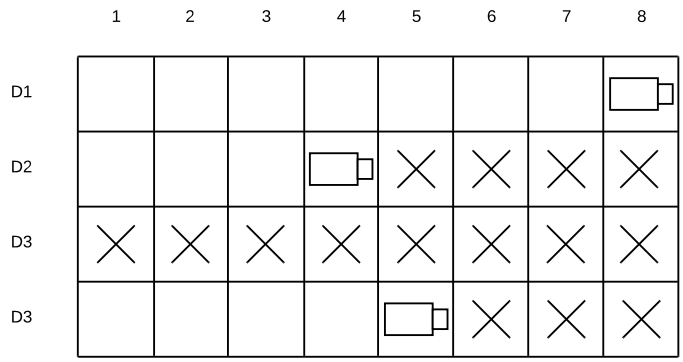

Figure 6. Last allowed unloading timeslot for each dock for a single truck.

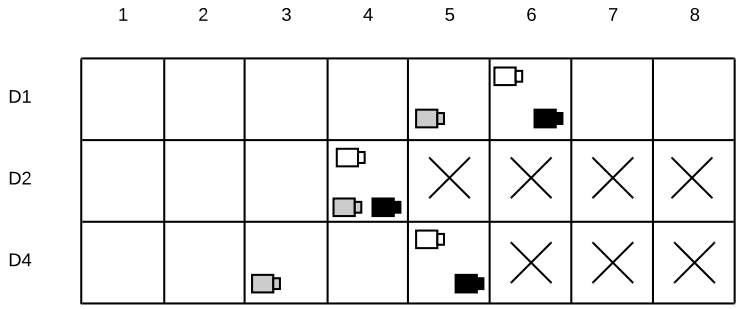

Figure 7. Three valid example assignment patterns for one truck, each pattern shown with different colored truck tokens. Note that dock 3 has been omitted from the figure as the truck does not visit the dock at all.

Let

$TR \subset \mathbb{N}_{>0}$ be the set of trucks to be unloaded on a day

$D \subset \mathbb{N}_{>0}$ be the set of docks

$T \subset \mathbb{N}_{>0}$ be the set of time slots

$t^D_{tr,d}$, $t \in T$ be the last allowed visiting time of truck $tr \in TR$ to dock $d \in D$

$t_{tr,d}$, $t \in \mathbb{N}_{>0}$ be visiting time of truck $tr \in TR$ to dock $d \in D$

Σ be the set of all valid solution sets

$s \in \Sigma$ set of dock visiting times $[t_{1,1}...t_{n,m}]$

For a solution $s \in \Sigma$ it is required that $\forall t_{tr,d} \ \in s$:

$$\begin{cases} t_{tr,d} \ \leq t^D_{tr,d}, \forall tr \in TR, d \in D \\ t_{tr,a} \neq t_{tr,b}, \forall tr \in TR, a \neq b \\ t_{k,d} \ \neq t_{j,d}, \forall k \neq j, d \in D \end{cases} \tag{1}$$

4.2 Optimization methodology

Given the constraints, an idea was conceived for algorithm that would operate a bit like a popular computer game where blocks are dropped from the top of the game area and

the player has to steer the block so that continuous horizontal lines are formed. Idea is that the trucks, modelled as blocks, are dropped down the day similarly to how the blocks are dropped in the game. Each block will have a virtual base level, which is the level for each dock the block cannot go below, and the virtual base level is established by the deadlines of the truck and the positions of previously placed trucks. For each block being dropped down the day only those docks are included in the playing field that the current truck has to visit. Until a block has reached the bottom level, it has multiple different configurations where it can be validly, thus the exact block shape is not determined until the block reaches the virtual base level. When one part of the block reaches the deadline level for that dock, then the block may still have freedoms regarding the other docks. The idea is to constrain the search space by attempting to place the trucks one by one, and branching on each block for all or some of the valid permutations. When the schedule starts to get full, it can possibly already be seen beforehand if the rest of the blocks can be filled in to some degree. This information could be used to short circuit the search, e.g. prune some search tree branches early on.

Based on the idea of the algorithm, a solution based on branch-and-bound algorithm was identified. Optimisation is done using a branch-and-bound algorithm by constructing a tree of solution candidates [24]. All such candidates are kept that have all trucks placed and a candidate is furthermore classified as *valid* if all trucks are placed in allowed positions and *invalid* if some if the trucks are placed in a too early position, e.g. before the first allowed time slot of the day.

Some other approaches were considered, such as genetic algorithms [25] and simulated annealing [26]. Due to the constraints of multiple dock visits and the required closeness of the visits, no clear way to define the problem was yet identified compatible with these approaches.

Constructing a tree has the benefit that the schedule is always free of multiple allocations to same time slot and dock. If the tree can be completed, then it has either a valid or invalid solution. Invalid solution might still be useful, as there may be ways to practically use the solution by means that are out of the scope of the optimisation, such as utilising extra capacity afforded by the operating domain but left out of the resources available for optimisation. Tree construction is also a deterministic approach in regard to the tree size and the expected run time of the algorithm for different sizes of problem is stable.

4.3 Algorithm

A tree of candidate solutions is generated by placing trucks one by one into the tree. The trucks are inserted into the search tree in priority order of least freedom of placement, e.g. the truck that has smallest amount of possible placements into an empty schedule will be inserted first. A solution is found for every branch that can be traversed up to the level where each truck has been placed.

An example tree structure used in this algorithm is depicted in Fig. 8. In this example there are three trucks to be placed. Initially in the root of the tree no trucks have been placed and the algorithm picks three different places for the first truck. Then in each branch two placements are selected for the second truck. Finally in in each branch one location is selected for the third truck. The number of generated branches monotonically decreases as more trucks are placed.

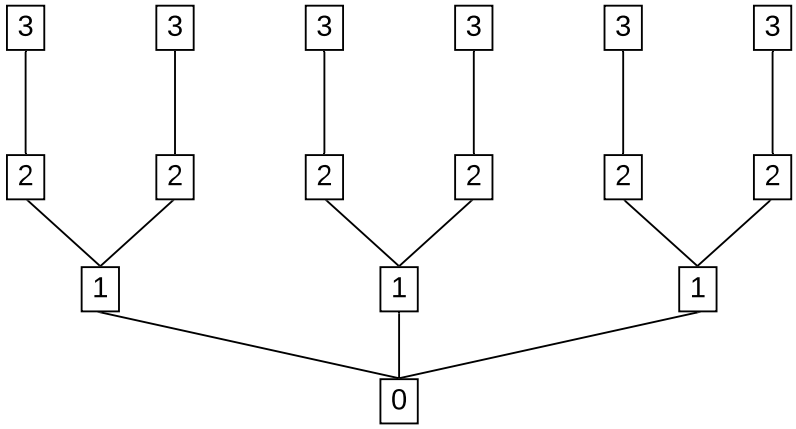

Figure 8. Search tree with dynamic branching factor. The dynamic branching factor cuts down the number of candidates when tree is traversed deeper. The number in each node shows how many trucks have been placed when the node has been traversed to.

A recursive depth-first search (DFS) is used to traverse through the search tree. The tree can have one or more branches per level. The algorithm operates on a data structure $STATE$ which consists of following data fields:

- N_{slots} Number of available timeslots, constant value over the optimization
- N_{pre} Number of extra timeslots prepended before the actual timeslots so that the

algorithm can try to place the trailers in earlier than allowed position. Constant value over the optimization.

- S Timeslot occupancy matrix of size $D \times N_{slots}$, where D is the number of docks. The matrix has value 1 in occupied cells and value 0 in free cells.
- ST Set of truck specific occupancy matrices with $s_{tr} \in ST$ being a matrix for a specific truck, having size $D_{tr} \times N_{slots}$ where D_{tr} is the number of docks the truck tr has to visit
- $|TR|$ Number of trucks to allocate
- U Priority ordered list of trucks that do not yet have a placement with $u_{tr} \in U$ being a set of $t^D_{tr,d}$ indicating last allowed time slot for the truck tr on dock d
- R Recursion step limit

The tail recursive optimisation algorithm described in Alg. 1 is initially called with a list of states having just one initial state in it, as well as with empty list final states. The initial state has empty time slot occupancy matrix S, empty set of truck specific occupancy matrices ST and priority ordered list of trucks U with all trucks in it.

The recursive depth-first search function described in Alg. 2 uses $placeNextTruck$ function which takes a state as parameter and produces zero or more new states. If any new states are produced, then those states have the truck that was next higher priority in the input state placed into all the produced new states, having a different placement in each of the produced states.

The algorithm run time is bounded by maximum number of recursion steps to make runs complete in limited time with the cost of possibly not finding any solutions. Maximum number of recursion steps is counted over all branches and since a depth-first recursive search is performed starting with the best fitting selections and thus the direction with highest potential, the recursion limit effectively limits the exploration of the branches that have less potential in regard to the fitness of individual truck placements.

4.4 Subproblems

The optimization algorithm embodies four subproblems. The problems are 3) the prioritisation of trucks to establish the search tree insertion order 2) fitness function for selection

Algorithm 1 Tail recursive depth-first search

```
1: function OPTIMIZERECURSIVE(recSteps : Int, currentStates : List[STATE], finalStates : List[STATE]): List[STATE]
2:     if callCount < R then return finalStates, recSteps
3:     else
4:         ▷ head is first element of the list, while rest is tail of the list
5:         head :: rest ← currentStates
6:         if head exists then
7:             newStates ← placeNextTruck(head)
8:             if newStates is non-empty then
9:                 ▷ new states are prepended to rest
10:                optimizeRecursive(recSteps + 1, newStates :: rest, finalStates)
11:            else
12:                if s.nonPlacedTrucks is empty then
13:                    ▷ This is a final state as no trucks are left unplaced
14:                    optimizeRecursive(recSteps + 1, rest, head :: finalStates)
15:                else
16:                    ▷ Terminated without legal placement for current truck to the state
17:                    optimizeRecursive(recSteps + 1, rest, finalStates)
18:                end if
19:            end if
20:        else
21:            return finalStates
22:        end if
23:    end if
24: end function
```

Algorithm 2

function PLACENEXTTRUCK($state : STATE$): List[STATE]
    $head :: rest \leftarrow state.U$ ▷ U list of non placed trucks
    if $head$ exists **then**
        Extract truck specific dock occupancy matrix s_{tr} from state
        occupancy mask S

        Combine truck last allowed times $t^D_{tr,d}$ for each dock with
        timeslot occupancy matrix S of current state to produce truck
        dock constraints matrix TDC

        Calculate all local truck permutation matrices of size $D_{tr} \times D_{tr}$

        Attempt to place all the permutations in all $D_{tr} \times D_{tr}$ submatrices of TDC
        and produce list of allowed candidates $allowedPlacements$

        Find up to N most approriate placements out all allowed,
        with N depending on the number trucks still not placed
        $suitablePlacements \leftarrow findNBestFittingPlacements(allowedPlacements)$

        for each $placement$ in $suitablePlacements$ **do**
            Create copy of current state $state$ with
            $\rightarrow$ occupancy matrix S combined with $placement$
            $\rightarrow$ $placement$ added into ST
            $\rightarrow$ $head$ removed from U
        end for
        Return the created new states or empty List
    end if
end function

of best fitting truck placements in truck placement step of Alg. 2, 3) choice of method to determine how many best fitting placements should be kept in each recursion step, and 4) fitness function for comparing the desirability of the found solutions.

The priority of truck is calculated by counting all valid placements of the truck if no other trucks have been placed. For example, if a truck has $N_{tr} = 3$ docks to visit and the visits have to be sequential without waiting times between the visits, then the truck in question has upper limit of $N_{tr}! = 6$ possible local placements in any 3×3 subset of the truck specific occupancy matrix s_{tr}. Number of local placements is then multiplied with number of 3×3 subsets, for example in s_{tr} of size 3×8 there would be 6 subsets of 3×3 matrices. Actual number of possible placements is lower than the upper limit if the truck has any last allowed timeslot $t^D_{tr,d} < N_{slots}$, and each permutation is verified against the last allowed time slots to establish the actual number of valid positions for the truck. Counting of valid placements is illustrated in 9.

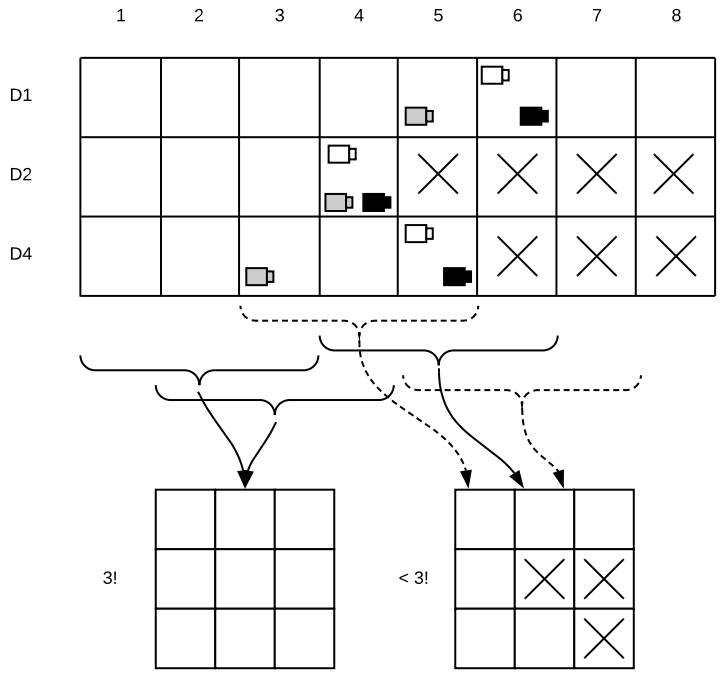

Figure 9. On the left side are shown 3×3 subsets that are all before a deadline position, thus yielding full number of 3! permutations. On the right side there are some subsets that do not yield full number of valid permutations due to deadline positions included in the regions.

The fitness function for selection of best fitting truck placement compares the largest time slot values of the placement candidates and the candidates with later time slots are pre-

ferred. The placements that occur later in the day are preferred because there is likely more congestion in early time slots, as there is no penalty for the unloading being scheduled too early, but scheduling too late is not allowed.

The method for determining how many placements to keep in each truck placement step determines how many branches the search tree will have. The branching factor can be static or dynamic. Static branching factor gives same amount of branches on every level of the search tree as long as there is at least same amount of possible placements for the truck that is allocated at the step. The dynamic branching method picks amount of branches to generate by equation

$$DBF(U) = max(B_{max} * \frac{|U|}{|TR|}, B_{min}) \tag{2}$$

Where B_{max} and B_{min} are maximum and minimum number of branches to generate, $|U|$ is the remaining number of trucks to be placed and and $|TR|$ is total number of trucks to be placed.

The dynamic branching method allows the algorithm to try wider range of search directions in respect to the high priority trucks with least amount of freedoms, but limits the tree expansion as the traversal proceeds to lower priority trucks deeper in the tree that have more options for placement.

The fitness function for selection of most suitable solution from the set of all found solutions measures how uniformly the time slots have been allocated over the time range by equation

$$FIT(S) = \frac{1}{|T'|} \sum_{t}^{T'} \sum_{d=1}^{D} g(t_{tr,d}) \tag{3}$$

where

$$g(t_t r, d) = \begin{cases} 1, if t_{tr,d} > 0 \\ 0, otherwise \end{cases} \tag{4}$$

and $T' = nets(D, t) \subset T$, where

$$nets(D, t) = \begin{cases} 1, if \sum_{d=1}^{D} t_{tr,d} > 0 \\ 0, otherwise \end{cases} \tag{5}$$

4.5 Implementation

The algorithm was implemented using Scala programming language [27], which is a multi-paradigm programming language supporting both object oriented and functional programming paradigms. The language was chosen as the author has long working experience with the language, but also because the language has extensive support for high level control flows and immutable data types, making it well suited to correctly express the required behaviours for recursion and structural sharing of immutable data structures. It is also important to note that the implementation can be taken into use in authors other solutions written in Scala with minimal changes.

All the occupancy matrix operations are expressed with the 64-bit integer type Long and most of the computations of suitable permutations and occupation matrices are performed using bitwise AND and OR operations. One Long value represents single dock, and each bit in the value represent one time slot. This implies that there is upper limit of 64 time slots that can be operated on by the current algorithm. This was deemed sufficient for most cases, for example if the time slots are 30 minutes each and the docks are open 24 hours a day, then up to 48 bits are needed to represent one dock.

Furthermore, the optimisation state is represented with linked lists. Each optimisation state corresponds to a node in the search tree, and due to use of linked lists the computations deeper in the tree can structurally share data of the parent nodes, which is both fast as no copying is needed as well as conserving memory.

Early on a breadth-first search (BFS) was implemented, but the problem with any branching factor $B_{max} > 2$ was that the memory consumption was huge and the tree would have to be explored to the leaf node level before any solutions would be found. In practice the BFS implementation was infeasible due to the memory consumption. DFS was then implemented with the goal of reducing memory usage and to allow potentially best branches to be explored early on, thus potentially providing at least some solutions with minimal tree traversal.

5 MODEL VALIDATION

5.1 Test inputs

All test inputs were generated with pseudo-random algorithms. All variations of algorithm parameters were tested with 100 problem instances and the seed value of the random number generator was fixed for each test run so that each test would use different seed number corresponding to the trial number, e.g. $1 \leq seed \leq 100$. Thus two test runs that differ only in a parameter that does not affect the pseudorandom values themselves, such as the recursion step limit, effectively operate on the same problem instance. This is important notion in that some insight can be received from the tests where the number of recursion steps is adjusted to see if it makes any difference when the algorithm is working on exactly the same problem instances.

Two different approaches were used for test data generation, one based on uniform distributions and another based on normal distributions. The test cases with uniform distributions were constructed for assessment of the performance against normal distributed test cases. Both distribution types were selected arbitrarily so that different kinds of test populations would be seen, as there is no data available for any specific domain and thus no claims are made regarding the suitability of these distributions to represent any real world arrangement. All test data generator parameters are shown in Table. 1. For each test data instance the optimisation algorithm was parametrised as shown in Table. 2.

The genetrated test inputs may contain trials that can have only *invalid* solutions, for example if there are two trucks that have the same deadline at the first time slot of the day for the same dock as shown in Table. 3.

Table 3. Example of situation that does not have any valid solutions since both truck 001 and truck 002 have a deadline on same dock D4 at first valid time slot 3.

		Time slot									
		1	**2**	3	4	5	6	7	8	9	10
Dock	D1	-	-	—-	—-	—-	—-	—-	—-	005	—-
	D2	-	-	—-	—-	—-	—-	007	—-	—-	005
	D3	-	-	—-	—-	—-	—-	002	—-	—-	—-
	D4	-	-	001+002	—-	—-	—-	003	007	—-	—-
	D5	-	-	—-	—-	—-	—-	—-	002	—-	005
	D6	-	-	—-	—-	—-	—-	—-	002	008	001

Table 1. Parameters for test data generator.

Parameter	Description
$\|TR\|$	Number of trucks
$\|D\|$	Number of docks
$\|D\|_{min}$	Lower bound (inclusive) for number of docks to visit for any truck
$\|D\|_{max}$	Upper bound (exclusive) for number of docks to visit for any truck
μ_D	Mean deadline (latest allowed time slot) to generate for any visit*
σ_D	Standard deviation of deadlines*
$numSlots$	Number of time slots
$numNonAlloc$	Number of invalid time slots at the beginning of the day
$earliestSlot$	Earliest time slot the generator is allowed to place a deadline
V_{max}	Max number of docks to visit per truck
μ_{vis}	Mean number of docks to visit per truck*
σ_{vis}	Standard deviation of docks to visit*
$randType$	Type of random number generator, e.g. "normal" or "uniform"
$seed$	Seed number for the pseudo-random random number generator

* Used only when $randType == normal$

Table 2. Parameters for optimization algorithm invocations.

Parameter	Description
B_{min}	Minimum branching factor
B_{max}	Maximum branching factor
R	Number of recursion steps to take

5.2 Analysis of results

Two sets of test runs were run for normal distributed deadlines and different number of visits per truck, one with earliest deadline being time slot 1 and another with time slot 3. Test parameters and results are shown in Table 4 and Table 5 respectively.

In both test sets in Table 4 and Table 5 more than 80 percent of trials could be solved at least one way. However, one trial stands out in both sets, the one with 15 docks, 100 trucks, maximum of 3 visits per truck, $\mu_D = 12$ and $\sigma_D = 4$. In this test most of the cases were not solvable with the current parametrization. Even for the cases that were solved, the ratio between λ_s and λ_i is clearly different from the other cases. However, when the mean deadline λ_s was changed to 18 and λ_i to 6, the algorithm could again work out the

solution for most of the cases. Due to the constraint that all visits for individual trucks should take place sequentially without time gaps, having the bulk of the deadlines earlier should make finding solution considerably harder and this could explain these results.

It was anticipated that having input sets with later earliest deadline would have significant difference in the number of found solutions, as the algorithm would have more often room to place trucks in earlier positions. This was also anticipated to hold for the tests cases in Table 6, as with uniform distributed deadlines and number of visits there would be very early deadlines in most of the cases, effectively forcing every truck to be placed in early time slots. Even with earliest deadline being in slot 3, it can be seen that the algorithm has clear difficulties in finding solutions in comparison the test cases in Table 4 and Table 5 with normal distributed deadlines and number of visits. However, having the earliest deadline being constrained to time slot 3 in Table 5 versus not being constrained at all in Table 4 does not seem to have so pronounced effect on normal distributed deadlines arrangement, which is likely due to the early hours not being so popular due to the used λ_s and λ_i setting most of the deadlines later in the day.

The average number of found solutions for those test cases that had any solutions are shown in Fig. 10 and Fig. 11. The cases where the number of trucks is small compared to the number of docks have high amount of solutions regardless of normal distribution parameters. When number of trucks increases, amount of found solutions diminishes but stays in the same level for both 20 and 40 trucks. When number of docks is $|D| = 15$ and number of trucks $|TR|$ is 100, maximum number of visits $V_{max} = 3$ and earliest deadline is 1 the solution rate for test cases with earliest mean deadline $\mu_D = 12$ and $\sigma_D = 4$ starts to drop in comparison to others as shown in Fig. 10. When number of docks is reduced to $|D| = 10$, the diminishing of amount of solutions is seen already at $|TR| = 60$ and there are no solutions found anymore when $|TR| = 100$. Corresponding figures Fig. 12 and Fig.13 show that number of test cases with any solutions found diminish in same order, but the loss is more pronounced when number of trucks $|TR| = 100$. Conclusion is drawn that number of trucks is limiting number of solutions the algorithm can find more when the trucks have earlier deadlines. However, as the goal is to find at least one valid solution, likelihood of finding a solution is better if the distribution of deadlines is such that most of the deadlines are later in the day.

One important factor that could affect the chance of finding solutions is the recursion step limit. When $|TR|$ grows, the number of full branches from root to leaf nodes the algorithm can explore shrinks in comparison to smaller amount of trucks, as the depth of each branch is higher. The effect of increasing the recursion step limit was tested with

the case where $|TR| = 100$ and recursion step limits of 500000 and 1000000 steps were tested in addition to the default 200000. The results shown in Fig. 14 and Fig. 15 indicate that the increase of the recursion size limit did not have any effect on how many problems could be solved, but average number of solutions found for those that had at least one solution was increased roughly in proportion to the amount of recursion steps.

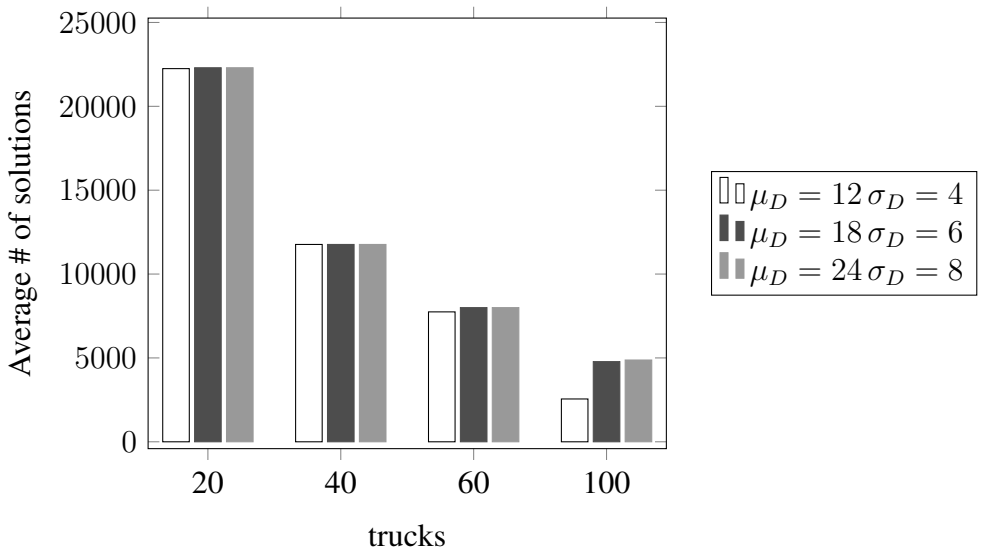

Figure 10. Average number of solutions for test cases that had at least one solution when $|D| = 15$, $V_{max} = 3$ and earliest deadline 1 over all result sets with different normal distribution parameters.

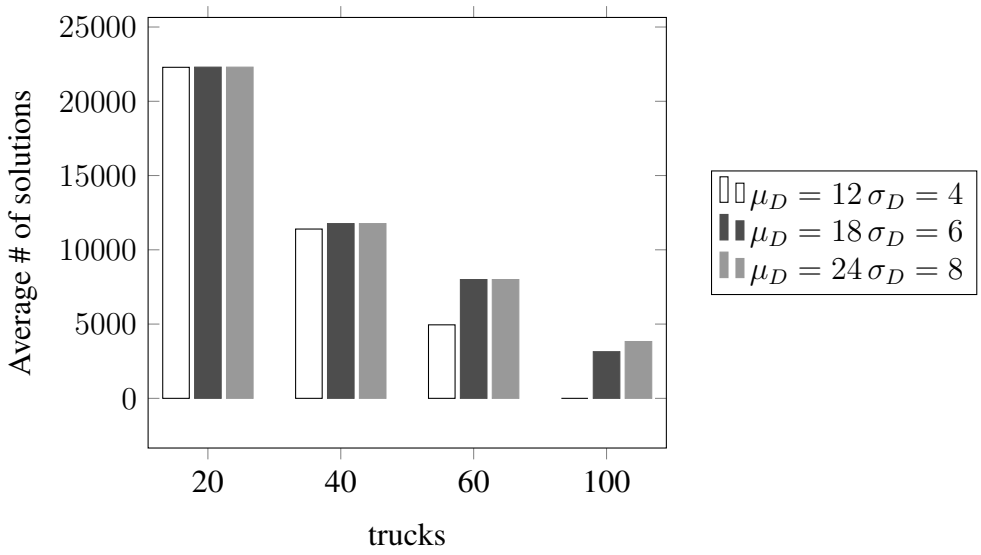

Figure 11. Average number of solutions for test cases that had at least one solution when $|D| = 10$, $V_{max} = 3$ and earliest deadline 1 over all result sets with different normal distribution parameters.

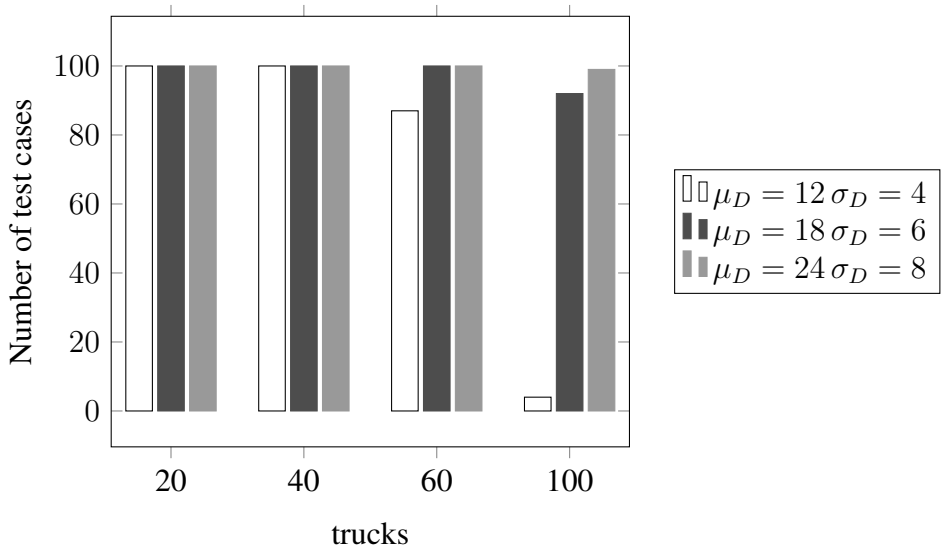

Figure 12. Number of test cases with at least one solution found when $|D| = 15$, $V_{max} = 3$ and earliest deadline 1 over all result sets with different normal distribution parameters.

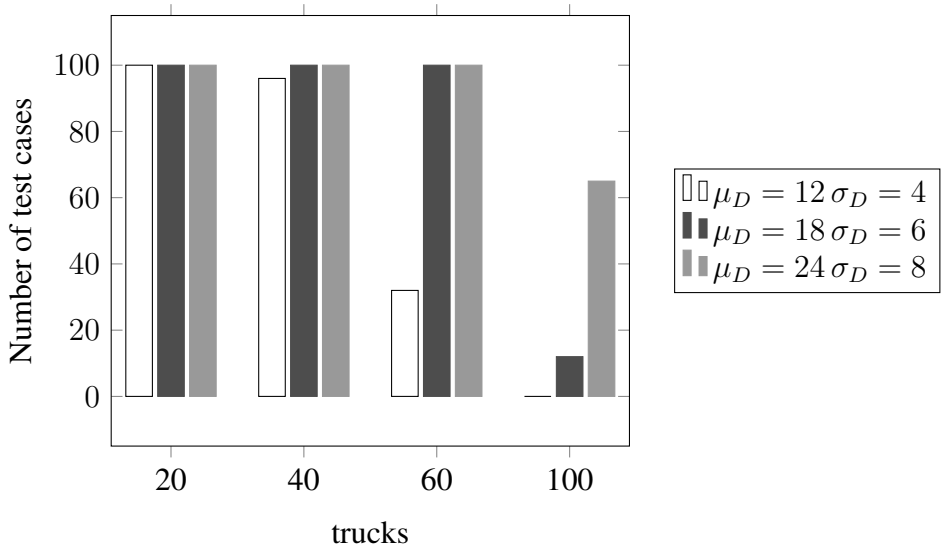

Figure 13. Number of test cases with at least one solution found when $|D| = 10$, $V_{max} = 3$ and earliest deadline 1 over all result sets with different normal distribution parameters.

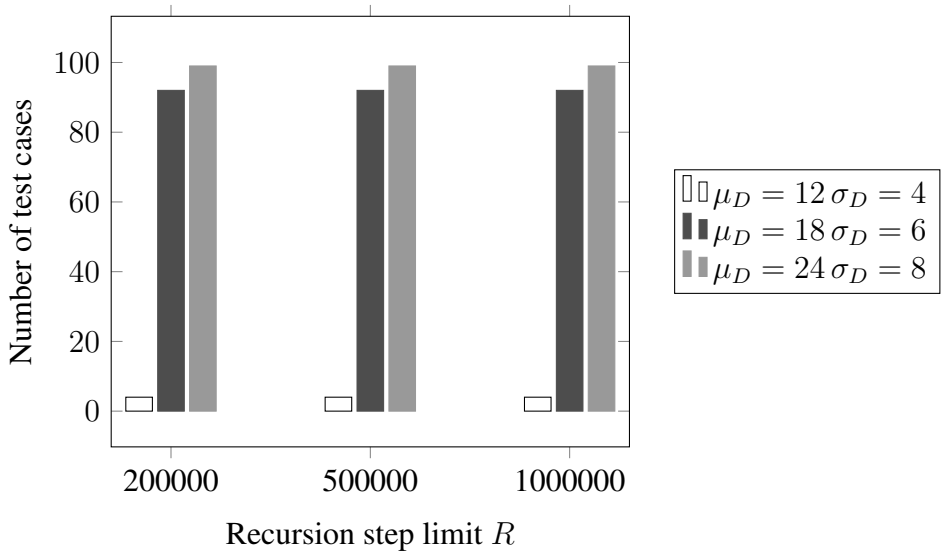

Figure 14. Number of test cases with at least one solution found when recursion step limit R is changed. $|D| = 15$, $V_{max} = 3$, $|TR| = 100$ and earliest deadline is 1.

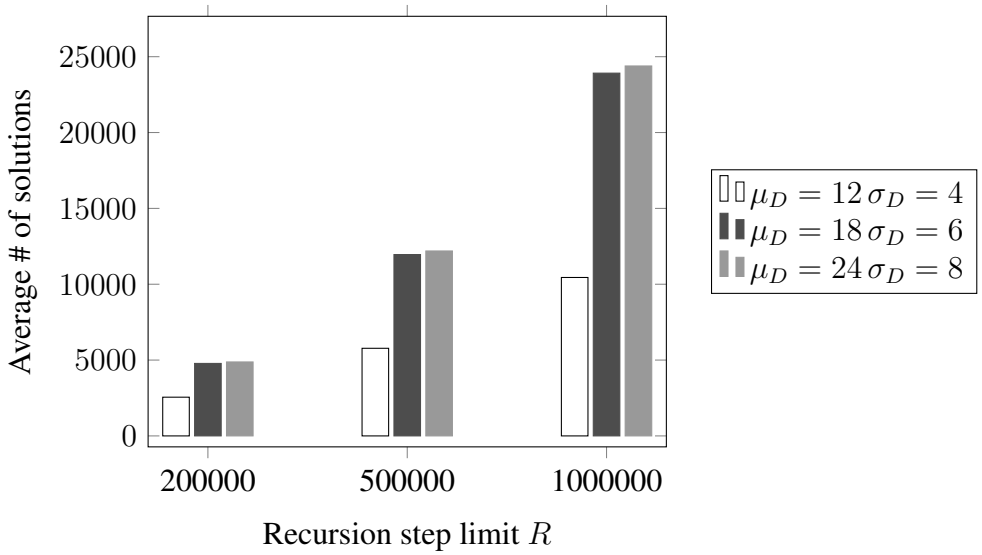

Figure 15. Average number of solutions for test cases with at least one solution found when recursion step limit R is changed. $|D| = 15$, $V_{max} = 3$, $|TR| = 100$ and earliest deadline is 1.

One set of test runs was carried out with uniform distributed deadlines and number of visits per truck. Parameters and results for these tests are shown in Table 6. The number of solutions was more varied than with the normal distributed test data. Especially the case with with 15 docks, 60 trucks, maximum of 3 visits per truck was more difficult for the algorithm. This was anticipated, as with normal distributed deadlines most of the trucks

will have at least one early deadline, which effectively constraints most of the trucks to be allocated on the early time slots. However, this result is not seen as significant, as the underlying assumption of uniformly distributed deadlines is likely unrealistic in real world scenarios, at least when there is at least some degree of coordination done in a centralised way.

Average algorithm execution time over all tests cases was 9.2 seconds. The average execution time per test instance is directly proportional to the number of docks and inversely proportional to the number of trucks as shown in Fig. 16. On the other hand, number of test instances that have at least one solution is decreased when the number of trucks is increased as shown in Fig. 17. A possible explanation for this is that when the ratio of trucks to docks $\frac{|TR|}{|D|}$ is high, there are more trucks with early deadlines for each dock, causing search tree branch traversals to terminate earlier and the search to more often complete without reaching the recursion step limit. However, it was also observed that test cases with high number of solutions run slower as more solutions are found and it seems the algorithm is pending more time managing the memory structures than in the computations themselves, which leads to conclusion that the current data structure does not scale well enough so that the execution performance would be mostly limited by CPU time.

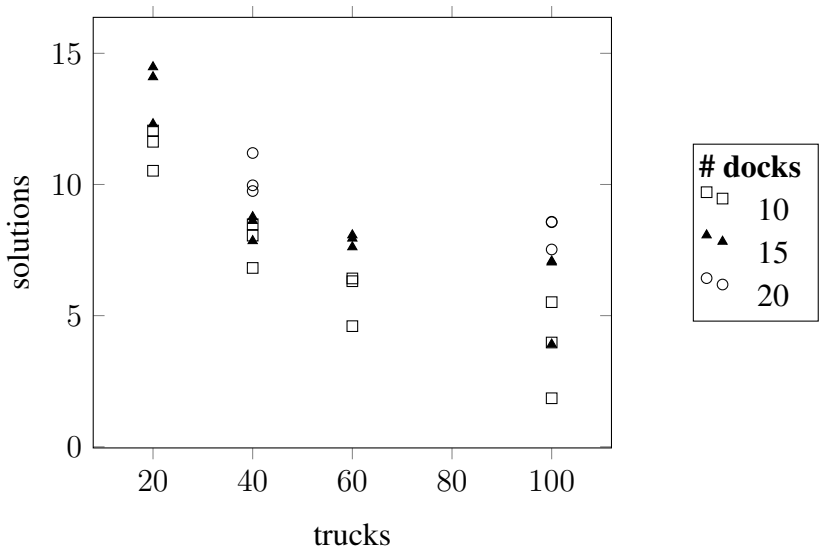

Figure 16. Average execution time per test for test instances in Table 4 with maximum number of visits $V_{max} = 3$ and earliest deadline 1.

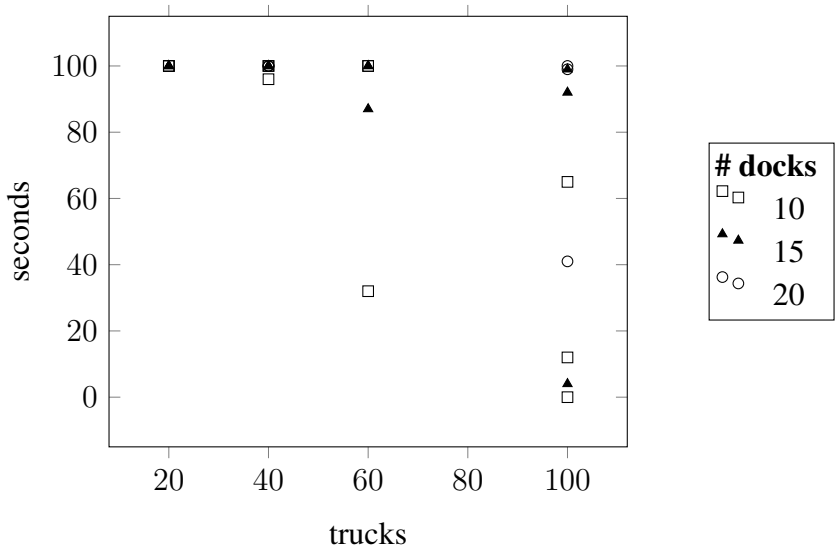

Figure 17. Average number of test instances with at least one solution from Table 4 with maximum number of visits $V_{max} = 3$ and earliest deadline 1.

Regarding the other desired qualities of a solution, there were very small visible differences in sparseness between the best found and worst found solutions in general. The values from the fitness function had minimal variation and it was not evident by visual inspection that there would be any conclusive difference between solutions with higher and lower fitness value.

Table 4. Results with normal distributed test data with earliest deadline 1

$\|D\|$	$\|TR\|$	V_{max}	μ_D	σ_D	N_s	λ_s	λ_i	λ_{ta}	$\tilde{ta}$	$\sum t_r$
10	20	3	12	4	100	22287.5	1152.0	1.0	1.0	17.5
10	20	3	18	6	100	22299.0				20.1
10	40	3	12	4	96	11398.3	5422.9	1.6	1.0	11.7
10	40	3	18	6	100	11765.0				13.4
10	60	3	12	4	32	4949.1	5845.2	3.1	1.0	7.7
10	60	3	18	6	100	7999.0				10.5
10	100	3	18	6	12	3141.3	3449.5	3.1	1.0	6.6
10	100	3	12	4	0					3.1
15	20	3	12	4	100	22247.2	5184.0	1.0	1.0	23.5
15	20	3	18	6	100	22299.0				24.1
15	40	3	12	4	100	11765.0				13.1
15	40	3	18	6	100	11765.0				14.6
15	60	1	12	4	99	7999.0	7999.0	1.0	1.0	7.1
15	60	1	18	6	100	7999.0				8.2
15	60	3	12	4	87	7762.0	5830.2	1.3	1.0	12.7
15	60	3	18	6	100	7999.0				13.5
15	60	5	12	4	80	7332.8	4798.1	1.9	1.0	22.4
15	60	5	18	6	100	7999.0				28.5
15	**100**	**3**	**12**	**4**	**4**	**2553.3**	**3681.1**	**4.1**	**1.0**	**6.5**
15	**100**	**3**	**18**	**6**	**92**	**4779.6**	**3990.0**	**1.6**	**1.0**	**11.7**
20	20	1	12	4	100	22299.0				15.9
20	20	1	18	6	100	22299.0				15.7
20	40	3	12	4	100	11765.0				16.6
20	40	3	18	6	100	11765.0				18.7
20	60	5	12	4	97	7897.0	5649.0	1.2	1.0	24.9
20	60	5	18	6	100	7999.0				28.1
20	100	3	12	4	41	4077.0	3895.1	2.3	1.0	12.5
20	100	3	18	6	99	4876.0	4876.0	3.0	3.0	14.3

Table 4. Results with normal distributed test data with earliest deadline 1 (continued)

$\|D\|$	$\|TR\|$	V_{max}	μ_D	σ_D	N_s	λ_s	λ_i	λ_{ta}	$\tilde{ta}$	$\sum t_r$
10	20	3	24	8	100	22299.0				19.4
10	40	3	24	8	100	11765.0				14.1
10	60	3	24	8	100	7999.0				10.7
10	100	3	24	8	65	3826.4	2846.0	2.0	1.0	9.2
15	20	3	24	8	100	22299.0				20.5
15	40	3	24	8	100	11765.0				14.4
15	60	1	24	8	100	7999.0				6.6
15	60	3	24	8	100	7999.0				13.2
15	60	5	24	8	100	7999.0				27.9
15	100	3	24	8	99	4876.0	4876.0	1.0	1.0	11.8
20	20	1	24	8	100	22299.0				12.6
20	40	3	24	8	100	11765.0				16.3
20	60	5	24	8	100	7999.0				29.2
20	100	3	24	8	100	4876.0				14.3

$|D|$ Number of docks, $|TR|$ Number of trucks, V_{max} Max number of visits per truck
μ_D Mean deadline (latest allowed time slot), σ_D Standard deviation of deadline
N_s Number of trials with at least one valid solution
λ_s Average number of valid solutions for trials with at least one valid solution
λ_i Average number of invalid solutions for trials with at least one invalid solution
λ_{ta} Average number of trucks too early for invalid solutions
$\tilde{ta}$ Median number of trucks too early for invalid solutions
$\sum t_r$ Calculation time in minutes

Common parameters for all tests: **Earliest deadline 1**, Number of trials 100, Maximum number of recursion steps 200000, Number of timeslots 27 (24 normal time slots + 3 spillover time slots prepended before the start of the day), minimum number of docks to visit per truck 1, minimum branching factor 1, maximum branching factor 5, Mean number of docks to visit per truck 1.5, Standard deviation of docks to visit per truck 1.5.

Table 5. Results with normal distributed test data with earliest deadline 3

$\|D\|$	$\|TR\|$	V_{max}	μ_D	σ_D	N_s	λ_s	λ_i	λ_{ta}	$\tilde{ta}$	$\sum t_r$
10	40	3	12	4	96	11433.5	4793.9	1.8	1.0	10.9
10	40	3	18	6	100	11765.0	-	-	-	12.5
15	20	3	12	4	100	22299.0	-	-	-	21.7
15	20	3	18	6	100	22299.0	-	-	-	23.0
15	60	1	12	4	100	7999.0	-	-	-	7.9
15	60	1	18	6	100	7999.0	-	-	-	7.2
15	60	3	12	4	89	7745.1	5399.5	1.3	1.0	12.6
15	60	3	18	6	100	7999.0	-	-	-	13.4
15	60	5	12	4	80	7323.5	4938.6	2.2	1.0	22.3
15	60	5	18	6	100	7999.0	-	-	-	26.3
15	**100**	**3**	**12**	**4**	**2**	**2985.5**	**4007.0**	**4.3**	**1.0**	**6.8**
15	**100**	**3**	**18**	**6**	**92**	**4789.9**	**4266.0**	**1.61**	**1.0**	**12.5**
20	40	3	12	4	100	11765.0	-	-	-	15.4
20	40	3	18	6	100	11765.0	-	-	-	16.7

$|D|$ Number of docks, $|TR|$ Number of trucks, V_{max} Max number of visits per truck
μ_D Mean deadline (latest allowed time slot), σ_D Standard deviation of deadline
N_s Number of trials with at least one valid solution
λ_s Average number of valid solutions for trials with at least one valid solution
λ_i Average number of invalid solutions for trials with at least one invalid solution
λ_{ta} Average number of trucks too early for invalid solutions
$\tilde{ta}$ Median number of trucks too early for invalid solutions
$\sum t_r$ Calculation time in minutes

Common parameters for all tests: **Earliest deadline 3**, Number of trials 100, Maximum number of recursion steps 200000, Number of timeslots 27 (24 normal time slots + 3 spillover time slots prepended before the start of the day), minimum number of docks to visit per truck 1, minimum branching factor 1, maximum branching factor 5, Mean number of docks to visit per truck 1.5, Standard deviation of docks to visit per truck 1.5 .

Table 6. Results with uniform distributed data

$\|D\|$	$\|TR\|$	V_{max}	N_s	λ_s	λ_i	λ_{ta}	$\tilde{ta}$	$\sum t_r$
15	100	3	1	4096.00	3144.08	6.71	1	5.11
15	60	3	67	6975.12	6083.86	1.87	1	11.01
15	20	3	100	22299.00	-	-	-	21.29
15	60	1	100	7999.00	-	-	-	5.94
15	60	5	0	-	3240.80	6.38	2	43.38
20	40	3	100	11706.12	2944.00	1.16	1	15.47
10	40	3	81	9947.49	7037.08	1.78	1	11.58

$|D|$ Number of docks, $|TR|$ Number of trucks, V_{max} Max number of visits per truck
N_s Number of trials with at least one valid solution
λ_s Average number of valid solutions for trials with at least one valid solution
λ_i Average number of invalid solutions for trials with at least one invalid solution
λ_{ta} Average number of trucks too early for invalid solutions
$\tilde{ta}$ Median number of trucks too early for invalid solutions
$\sum t_r$ Calculation time in minutes

Common parameters for all tests: **Earliest deadline 3**, Number of trials 100, Maximum number of recursion steps 200000, Number of timeslots 27 (24 normal time slots + 3 spillover time slots prepended before the start of the day), minimum number of docks to visit per truck 1, minimum branching factor 1, maximum branching factor 5

6 DISCUSSION

The truck dock assignment problem is an interesting subproblem in the field of transport logistics. From viewpoint of the study this relatively tightly bounded domain provides a suitably sized area to focus on. However, it is interesting to place the problem in the broader context of transport and warehouse logistics and to consider the implications of the performance characteristics of such a solution in this frame of reference.

Taking a step away from truck dock assignment problem, amongst first things to consider is that the trucks need be to routed to the destination possibly from multiple locations. The trucks may have to visit multiple places while en route to the destination and after leaving the destination. Different requirements imposed by equipment limitations and personnel related limitations, for example, need to be considered. In this level an instance of *vehicle routing problem* (VRP) inevitably arises. VRP is a generalization of *travelling salesman problem* (TSP) [28]. Depending on the competition and complexity of transportation needs in the particular area of transportation, very simple arrangements can possibly be manually handled, but in general computational means are needed to find efficient solutions.

VRP likely plays a major role in efficiency of a transportation arrangement, and thus also on the direct costs and indirect costs, such as carbon emissions, just because the trucks spend considerable amount of their usage time en route. However, truck loading and unloading also takes time, and even though the engines are not running, there are associated costs such as personnel expenses and cost of lost opportunity if loading and unloading cannot be done efficiently and in right time. Thus truck dock assignment likely has a considerable impact on the efficiency of the transportation arrangement.

Taking another step away, the transportation need not the be considered in isolation from warehouse arrangements. A good *warehouse management system* (WMS) boosts warehousing eco-friendliest by reducing space waste and unnecessary number of material movements, to ensure continuous fast materials flow and to keep the heating and cooling costs minimum [29]. When considering implementation of a WMS in any particular arrangement, existence of a solution to domain specific truck dock assignment problem should be taken into account. Regardless of transportation and WMS arrangements, there could also be strategic conflicts between supply chain participants that could have high impact on the overall utility of the supply chain. In such cases utility of the supply chain could possibly be improved more by changing the overall supply chain strategy than by optimising the existing operations [30].

7 CONCLUSION

The study was made out of interest of developing a solution to a problem similar to a real world use case, and as such to gain insight on implementation of solutions to such problems. The results show that the algorithm can often find multiple solutions to problems of size of interest in time that is acceptable for practical use. However, as the study does not contain any real world test data, no claims can be made regarding the suitability to any specific arrangement without further study. It is anticipated that the work could be used to solve practical truck dock assignment problems, even if globally optimal solution is not found.

Regarding the original research questions, it is shown that the scheduling can be done automatically assuming that the distribution of truck deadlines is similar to the deadlines used in test data. Amongst the found solutions such solution can be chosen that has free slots evenly spread during the day, thus allowing flexibility in case of unanticipated changes during the day. Also the presentation of a solution, as shown in appendix, is suitable for inspection by operating personnel and even more practical when shown in computer screen with color encoded truck numbers or even with a graphical layout.

The optimisation algorithm has predictable running time due to the used recursion step limit, and as such the time allowed for optimisation can be bounded. If optimisation is needed to be done in limited time, then in some cases even an *invalid* solution that was found but does not have every truck in valid place could be useful, for example as basis for manual truck dock assignment where the trucks in invalid positions could be handled with other flexibilities allowed by the domain of application.

As a learning opportunity many useful insights were gained during the preparation of this work. One specific insight found is that the constraints imposed by a specific problem case can lead to interesting properties, such as the permutation matrices for allowed placements from the truck specific dock submatrices utilised in this work. In this case that structural property narrowed down the number of possible placements for each truck, which allowed efficient implementation of scanning of possible truck placement positions using integer binary logic. These kinds of problem specific structures are very useful for implementation of optimisation heuristics in practical engineering contexts.

7.1 Future work

There are many aspects in the work that were left out of the scope of the work and many more interesting ones that were found during the research. A logical next step would be to generate test data based on real world distributions of truck deadlines and provide a case study on real world applicability of the solution. On theoretical side it would be interesting to assess the robustness of the algorithm by generating test cases where each test instance would be known to be solvable.

An important addition to current algorithm would be support for scheduling of multiple trucks to same docks. In practice some docks may have capacity to operate with multiple trucks concurrently and the algorithm in the current form does not support this arrangement. Adding support for such use case would likely be possible with nontrivial changes to the data structures the algorithm operates on.

Bounding step of Branch-and-Bound algorithm was not utilised in the algorithm, but it could be useful in limiting the exploration of the search space to potentially better subset of branches than the current recursion step limit does. With bounding step, before a node is branched, it is checked whether it can contain a solution better than the best one found so far by the algorithm using upper and/or lower bounds on its local solution. The node is discarded instead if it cannot [31]. It would be interesting to study what kind of heuristics could be used to gain insight of probability that a branch does not contain better solution than one of those already found by using the information of the subtree traversed already and the information about the trucks that have not yet been placed.

In addition to the Branch-and-Bound algorithm, it would be interesting to explore the solution space using other approaches such as genetic algorithms and simulated annealing. Also the current algorithm with BFS instead of DFS could possibly be combined with genetic algorithm to make the BFS variant practical.

REFERENCES

[1] Zhaowei Miao, Andrew Lim, and Hong Ma. Truck dock assignment problem with operational time constraint within crossdocks. *European Journal of Operational Research*, 192(1):105, Jan 01 2009.

[2] Yiyo Kuo. Optimizing truck sequencing and truck dock assignment in a cross docking system. *Expert Systems with Applications*, 40(14):5532 – 5541, 2013.

[3] Freightera entry for cross docking. `https://web.archive.org/web/20191213093222/https://www.freightera.com/blog/freight-terms-glossary/cross-docking/`. Accessed: 2019-12-13.

[4] Jinwen Ou, Vernon N. Hsu, and Chung-Lun Li. Scheduling truck arrivals at an air cargo terminal. *Production and Operations Management*, 19(1):83–97, Jan 2010.

[5] Chart Technica airfreight chart. `https://web.archive.org/web/20180413103254/http://freightcharts.net/index.php`. Accessed: 2018-04-13.

[6] Zhong-Zhen Yang, Gang Chen, and Dong-Ping Song. Integrating truck arrival management into tactical operation planning at container terminals. *Polish Maritime Research*, 20 (special issue):32–46, Jan 2013.

[7] Chuqian Zhang, Jiyin Liu, Yat wah Wan, Katta G Murty, and Richard J Linn. Storage space allocation in container terminals. *Transportation Research Part B: Methodological*, 37(10):883 – 903, 2003.

[8] Kiyoul Lee, Hyunbo Cho, and Mooyoung Jung. Simultaneous control of vehicle routing and inventory for dynamic inbound supply chain. *Computers in Industry*, 65(6):1001 – 1008, 2014.

[9] Depeng Fan. Research of inbound mode for automobile parts logistics based on the second factory of company h. *Journal of Investment and Management*, 5(4):34–38, Aug 2016.

[10] Katri Koljonen. Vendor managed inventory: The critical success factors from the manufacturer point of view. Master's thesis, Lappeenranta University of Technology, Finland, 2006.

[11] Patricia Guarnieri and Kazuo Hatakeyama. Formalização da logística de suprimentos: caso das montadoras e fornecedores da indústria automotiva brasileira. *Produção*, 20, 01 2010.

[12] Nikita Belyak. Simulation methods for transport logistics. Master's thesis, Lappeenranta University of Technology, Finland, 2018.

[13] Nils Boysen, Stefan Fedtke, and Felix Weidinger. Truck scheduling in the postal service industry. *Transportation Science*, 51(2):723–736, 2017.

[14] Sidney Browne and Paul Zipkin. Inventory models with continuous, stochastic demands. *The Annals of Applied Probability*, 1(3):419–435, 1991.

[15] Azmat Gani. The logistics performance effect in international trade. *The Asian Journal of Shipping and Logistics*, 33(4):279 – 288, 2017.

[16] Jane Korinek and Patricia Sourdin. To what extent are high-quality logistics services trade facilitating? Technical report, Organisation for Economic Cooperation and Development (OECD), Mar 01 2011.

[17] Shahin Gelareh, Rahimeh Neamatian Monemi, Frédéric Semet, and Gilles Goncalves. A branch-and-cut algorithm for the truck dock assignment problem with operational time constraints. *European Journal of Operational Research*, 249(3):1144 – 1152, 2016.

[18] Benjamin Kemper, Chris A.J. Klaassen, and Michel Mandjes. Optimized appointment scheduling. *European Journal of Operational Research*, 239(1):243 – 255, 2014.

[19] Joren Marynissen and Erik Demeulemeester. Literature review on multi-appointment scheduling problems in hospitals. *European Journal of Operational Research*, 272(2):407 – 419, 2019.

[20] W. Stadler. *Fundamentals of Multicriteria Optimization. In: Stadler W. (eds) Multicriteria Optimization in Engineering and in the Sciences. Mathematical Concepts and Methods in Science and Engineering, vol 37.* Springer, 1988.

[21] Christodoulos A. Floudas and P. M. Pardalos. *Encyclopedia of Optimization*. Springer, 2nd edition, 2009.

[22] Dinçer Konur and Mihalis M. Golias. Analysis of different approaches to cross-dock truck scheduling with truck arrival time uncertainty. *Computers & Industrial Engineering*, 65(4):663–672, 2013.

[23] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[24] David R. Morrison, Sheldon H. Jacobson, Jason J. Sauppe, and Edward C. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79 – 102, 2016.

[25] Maxim A Dulebenets. A diploid evolutionary algorithm for sustainable truck scheduling at a cross-docking facility. *Sustainability*, 10, 2018.

[26] Scott Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science (New York, N.Y.)*, 220:671–80, 06 1983.

[27] Martin Odersky, Lex Spoon, and Bill Venners. *Programming in Scala: Updated for Scala 2.12*. Artima Incorporation, USA, 3rd edition, 2016.

[28] G. B. Dantzig and J. H. Ramset. The truck dispatching problem. *Management Science, Vol. 6, No. 1 (Oct., 1959), pp. 80-91*, 6:80–91, 1959.

[29] Ari Happonen and Daria Minashkina. Decarbonizing warehousing activities through digitalization and automatization with WMS integration for sustainability supporting operations. In *7th International Conference on Environment Pollution and Prevention*, 12 2019.

[30] Erno Salmela and Ari Happonen. Synchronization of demand and supply in a supply chain manufacturing industrial products. In *Proceedings of 16th International Annual EurOMA Conference*, 6 2009.

[31] Juan F. R. Herrera, José M. G. Salmerón, Eligius M. T. Hendrix, Rafael Asenjo, and Leocadio G. Casado. On parallel branch and bound frameworks for global optimization. *Journal of Global Optimization*, 69(3):547–560, Nov 2017.

Appendix 1. Example results

A small example problem is presented here to show the actual inputs and results. Example input is given in Table A1.1 and two solutions found for this input are shown in A1.2 and A1.3. The values in the table cells represent the identification numbers of the trucks. These could correspond to license plate numbers, for example.

Time slots 1-2 are extra time slots the algorithm may use to produce *invalid* solutions if necessary. For *valid* solutions there must be no entries in these time slots. In Table A1.1 there are some cells with more than one deadline in one cell, separated with $+$.

Table A1.1. Example of input deadlines for case with 6 docks and 12 trucks. There are multiple deadlines for some trucks in same dock and time slot.

		Time slot									
		1	**2**	3	4	5	6	7	8	9	10
Dock	D1	-	-	—-	—-	—-	7	—-	—-	5	12
	D2	-	-	—-	—-	—-	—-	7+8	—-	—-	5
	D3	-	-	—-	—-	1	—-	2+3	12	—-	—-
	D4	-	-	—-	—-	—-	1+4	3	7+10+12	—-	—-
	D5	-	-	—-	—-	—-	—-	—-	2+8+11+12	3	5+6
	D6	-	-	—-	—-	3	—-	—-	2	9	1

Table A1.2. Best fitting solution found for example with 6 docks and 12 trucks. $fitness = 0.375$

		Time slot									
		1	**2**	3	4	5	6	7	8	9	10
Dock	D1	-	-	—-	—-	—-	7	12	—-	5	—-
	D2	-	-	—-	—-	—-	8	7	5	—-	—-
	D3	-	-	—-	—-	1	3	2	12	—-	—-
	D4	-	-	4	1	3	12	10	7	—-	—-
	D5	-	-	—-	3	12	2	8	11	6	5
	D6	-	-	3	—-	—-	1	9	2	—-	—-

Appendix 1. Example results

Table A1.3. Worst fitting valid solution found for example with 6 docks and 12 trucks. $fitness = 0.625$

		Time slot									
		1	**2**	3	4	5	6	7	8	9	10
Dock	D1	-	-	—-	—-	7	12	—-	5	—-	—-
	D2	-	-	—-	—-	—-	7	8	—-	—-	5
	D3	-	-	—-	12	1	3	2	—-	—-	—-
	D4	-	-	12	1	3	4	7	10	—-	—-
	D5	-	-	—-	3	12	2	11	8	5	6
	D6	-	-	3	—-	—-	1	—-	2	9	—-