Hybrid adaptive large neighborhood search for multi-load AGV scheduling

van de Sande, D.T.J.

Award date:
2020

Eindhoven, May 12, 2020

Hybrid adaptive large neighborhood search for multi-load AGV scheduling

Student:
D.T.J. (Dirk) van de Sande
0862030

Committee members:
prof.dr.ir. I.J.B.F. Adan
dr. T.G. Martagan
dr. E. Torta
dr. Q.V.Dang

External Supervisors:
Jan Willem Sietsma
Roel Nieuwenhuizen

Manufacturing Systems Engineering
Mechanical Engineering
Dynamics and Control

Contents

Abstract

This thesis addresses the scheduling of multi-load automated guided vehicles (AGVs) while considering requests with soft time windows and different priorities, a heterogeneous fleet of AGVs with different capabilities, and battery management with partial recharging and a critical battery threshold. In contrast to single-load AGVs, multi-load AGVs are able to transport more than one load simultaneously. Each request consists of a pickup and a delivery task associated with an origin, destination, time window and required capability. The scheduling problem consists of assigning tasks to AGVs, sequencing these task and determining the arrival times and charging duration. A hybrid adaptive large neighborhood search (ALNS) with an integrated local search procedure is proposed to find a feasible sequence of tasks with the aim to reduce the tardiness costs of requests and travel costs of AGVs. A schedule generation scheme is applied to determine the arrival times and charging duration for a given sequence of tasks. The performance of the hybrid ALNS is compared to both a proposed mixed-integer linear programming (MILP) model and a dispatching model. Computational results indicate that the proposed hybrid ALNS significantly outperforms the MILP model and the dispatching approach in terms of CPU time and solution quality, respectively.

1 Introduction

Automated guided vehicles (AGVs) are commonly used for material handling operations in manufacturing plants, warehouses, distribution centers and transshipment terminals (Le-Anh and De Koster, 2006). AGVs transport materials from one production center to another or to another location in the facility. Jobs can be, for example, transporting raw materials, tools, semi-finished products or finished products (Angra et al., 2018). The Brainport Industries Campus (BIC) is an example of a facility that is trying to implement an automated material handling system using AGVs. The BIC is a new, high-tech campus located in Eindhoven, Netherlands, in which companies work together to innovate and produce. They share resources including flexible production areas, clean rooms, warehouses and material handling systems such as AGVs (Brainport Industries Campus, 2020).

In order to implement an AGV system, several components must be considered, including guide-path design, AGV fleet size, AGV scheduling, idle-vehicle positioning, battery management and vehicle routing (Le-Anh and De Koster, 2006). This research focuses on the scheduling of multi-load AGVs. The problem of scheduling AGVs can be decomposed into three sub-problems: a dispatching problem (i.e., selecting and assigning tasks to AGVs and sequencing these tasks), a routing problem (i.e., selecting specific paths to reach the assigned destination) and a scheduling problem (i.e., determining the arrival times and charging duration of AGVs) (G. Li et al., 2019). AGVs can be classified as single-load or multi-load. Single-load AGVs are able to transfer one job at a time, while multi-load AGVs can carry and transport more than one job simultaneously, which can significantly improve the produc-

tivity of material transfer operations (Bilge and Tanchoco, 1997). Using multi-load AGVs for material handling operations may reduce the number of AGVs needed or increase the throughput of the system (Le-Anh and De Koster, 2006).

Applying multi-load AGVs provides additional flexibility since they can deviate from their routes to pick up additional loads. The downside of this increased flexibility is that scheduling multi-load AGVs becomes more complex. Commonly, dispatching rules are used to control multi-load AGVs. This thesis proposes a more advanced control algorithm to increase the system's performance, which is characterized by the tardiness costs of requests and the travel costs of AGVs. The main novelty of this research is the combination of characteristics including capacity of multi-load AGVs, requests with soft time windows and different priorities, heterogeneous fleet of AGVs with different capabilities and battery management with partial recharging and a critical threshold.

The main contributions of this research consists of (i) a mixed-integer linear programming (MILP) model formulation for the problem of scheduling multi-load AGVs; (ii) a hybrid adaptive large neighborhood search (ALNS) (Ropke and Pisinger, 2006) with an integrated local search procedure to solve industry-sized instances, which outperforms the MILP and the commonly used dispatching rules; (iii) managerial insights showing the effect of several key components of multi-load AGVs, including fleet size, amount of requests, tightness of the time window, capacity and computational time, on the system's performance.

This research is part of the Advanced Manufacturing Logistics (AML) project, a joint collaboration between several companies and universities with the goal of improving production logistics and planning (Advanced Manufacturing Logistics, 2020). The AML project has a field lab located at the BIC, and the case studies generated for this research are based on the layout of this field lab. In addition, this research is performed in collaboration with IJssel Technology. They are one of the companies involved in the AML project and are responsible for the control of the AGVs within the field lab. The core business of IJssel Technology is developing, improving and maintaining production processes based on thorough knowledge of practice. This research can be used by company managers to support the decision making process and the proposed solution approach can be applied at the operational level.

This thesis is organized as follows. A literature review about scheduling multi-load AGVs and recent developments concerning ALNS is provided in Section 2. The problem is formally defined in Section 3, and a mathematical model is formulated in Section 4. Section 5 presents the proposed hybrid ALNS, and the current AGV dispatching rules are explained in Section 6. The results of computational experiments are provided in Section 7, which include a parameter tuning, an evaluation of the impact of different operators, comparisons against the MILP model and dispatching rules, and a sensitivity analysis. Finally, Section 8 provides a conclusion with remarks for future research.

2 Literature review

The common approach to control a fleet of multi-load AGVs is to use dispatching rules. Le-Anh and De Koster (2006) present a literature review about the control of single-load AGVs and briefly address multi-load AGVs. They provide some guidelines for vehicle dispatching, idle-vehicle positioning, battery management, guide-path design, vehicle requirements and vehicle routing. Egbelu and Tanchoco (1984) classify the dispatching rules for AGVs into two categories: load-initiated rules, in which jobs at the workstation have the priority to claim vehicles, and vehicle-initiated rules, in which vehicles have the priority to claim jobs. The authors conclude that vehicle-initiated rules perform better, especially in cases of high transport demand. Y.-C. Ho and Chien (2006) divide the dispatching problem for multi-load AGVs into three sub-problems: task-determination (i.e., pickup or delivery), pickup-dispatching problem (i.e., which task to pick up) and delivery-dispatching problem (i.e., which task to deliver first). Several simulation studies have been performed in an attempt to find the best task-determination, pickup-dispatching and delivery-dispatching rules (Y.-C. Ho and Liu, 2006; Lee and Srisawat, 2006; Y.-C. Ho and Liu, 2009; Azimi et al., 2010; Angra et al., 2018). It can be concluded that the performance of individual dispatching rules depends on the problem setting (e.g., layout, guide path, demand), so there is no rule that always outperforms all other rules. Y.-C. Ho et al. (2012) develop a multi-attribute pickup-dispatching rule that takes into account the slack time, waiting time and distance. More recently, M. P. Li and Kuhl (2017) proposes the pickup-or-delivery-en-route task-determination rule, which allows AGVs to pick up and deliver loads if they pass the corresponding pickup/delivery locations while en route to their next assigned task.

Dispatching rules are transparent, robust and generate real-time solutions. However, these solutions are typically not optimal and leave significant room for improvement. Hence, there is a need for a more advanced control strategy to ensure high system performance. Most research about AGVs is focused on single-load and there are only a few papers devoted to scheduling multi-load AGVs that do not apply dispatching rules. Chen et al. (2015) and G. Li et al. (2019) apply multi-load AGVs to supply buffers at machines from a single distribution point. Chen et al. (2015) implement a reinforcement learning based approach, while G. Li et al. (2019) present a harmony search algorithm. Huo et al. (2016) and Ma et al. (2020) study the scheduling problem of multi-load AGVs in an automated container terminal. Huo et al. (2016) develop a genetic algorithm and Ma et al. (2020) use an improved shuffled frog leaping algorithm. Chawla et al. (2018) design a modified memetic particle swarm optimization algorithm, which combines particle swarm optimization for global search and a memetic algorithm for local search, to solve the problem of scheduling multi-load AGVs in a flexible manufacturing system. Rahimikelarijani et al. (2020) propose a variable neighborhood search (VNS) to solve the problem of scheduling multi-load AGVs in a tandem layout for small-sized scenarios. However, none of the proposed methods in literature consider soft time windows of requests, heterogeneous fleet with different capabilities and battery management of AGVs.

The problem of scheduling multi-load AGVs can be modelled as a Dial-a-Ride Problem (DARP) because the DARP already takes several of the constraints into account. The DARP formulation can cope with the following sub-problems of scheduling multi-load AGVs: sequencing pickup and delivery actions, variable start/end locations, time windows of requests, heterogeneous fleets and vehicle capacities. Both Molenbruch et al. (2017) and S. C. Ho et al. (2018) conduct a survey about the solution method for the DARP and devote attention to recent developments. They divide the solution methods into exact and approximate methods. The DARP is an NP-hard problem (Baugh Jr et al., 1998), hence optimal solutions are not expected to be found in polynomial time. Therefore, approximate methods are applied to solve realistic-sized problems. The approximate methods consist of classical heuristics, metaheuristics based on local search, metaheuristics based on population search, matheuristics and simulation studies. The main methods used for solving this type of problems are metaheuristics based on local search, which include tabu search (e.g. Paquette et al. (2013); Detti et al. (2017)), variable neighborhood search (e.g. Parragh et al. (2010); Muelas et al. (2015)), large neighborhood search (e.g. Masmoudi et al. (2016); Belhaiza (2019)), deterministic annealing (e.g. Braekers et al. (2014)) and simulated annealing (e.g. Mauri and Lorena (2006)). The survey of Molenbruch et al. (2017) states in particular that metaheuristics based on local search have obtained good results within realistic computational times and that their performance can still be improved by using operators that exploit the specific problem structure. The DARP captures all the components of scheduling multi-load AGVs except battery management.

However, battery management for electric vehicles has been extensively studied in several vehicle routing problems (VRP), see Schneider et al. (2014); Felipe et al. (2014); Goeke and Schneider (2015); Desaulniers et al. (2016); Keskin and Çatay (2016); Schiffer and Walther (2017); Zhao and Lu (2019). Typical battery management policies are: (i) full recharge (e.g. Schneider et al. (2014)), (ii) battery swapping (e.g. Yang and Sun (2015)) and (iii) partial recharging (e.g. Keskin and Çatay (2016)). There are only a few papers that combine electric vehicles and the DARP problem. Masmoudi et al. (2018) propose an evolutionary variable neighborhood search to solve the DARP with electric vehicles and battery swapping stations and Bongiovanni et al. (2019) investigate the DARP with a homogeneous fleet of electric vehicles while allowing partial recharging and solves it for small-sized instances with a Branch-and-Cut algorithm. The literature review shows that there is a research gap for scheduling electric vehicles in combination with the DARP for industry-sized instances. Furthermore, this problem is extended by also considering requests with soft time windows and different priorities, and a heterogeneous fleet of AGVs with different capabilities. A clear summary of the comparison of this work to existing literature is shown in Table 2.1.

Table 2.1: Comparison of current work to related literature.

	Mauri and Lorena (2006)	Parragh et al. (2010)	Demir et al. (2012)	Paquette et al. (2013)	Braekers et al. (2014)	Felipe et al. (2014)	Schneider et al. (2014)	Chen et al. (2015)	Goeke and Schneider (2015)	Muelas et al. (2015)	Yang and Sun (2015)	Desaulniers et al. (2016)	Huo et al. (2016)	Keskin and Çatay (2016)	Y. Li et al. (2016)	Masmoudi et al. (2016)	Detti et al. (2017)	Schiffer and Walther (2017)	Chawla et al. (2018)	Masmoudi et al. (2018)	Belhaiza (2019)	Bongiovanni et al. (2019)	Gschwind and Drexl (2019)	G. Li et al. (2019)	Singh (2019)	Zhao and Lu (2019)	Ma et al. (2020)	Rahimikelarijani et al. (2020)	**Current work**
Vehicle routing problem																													
Electric						X	X		X		X	X		X				X		X		X			X	X			**X**
Pickup and delivery	X	X		X	X			X		X			X		X	X	X		X	X	X	X			X	X	X	X	**X**
Capacitated	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	**X**
Vehicle fleet																													
Homogeneous		X	X			X	X	X		X	X	X	X	X	X			X	X			X	X	X			X	X	
Heterogeneous	X			X	X				X							X	X			X	X				X	X			**X**
Time windows																													
None						X		X			X								X					X			X	X	
Soft										X												X			X				**X**
Hard	X	X	X	X	X		X		X			X	X	X	X	X	X	X		X	X		X			X				
Requests																													
Equal priority	X	X	X	X	X	X	X		X	X	X	X	X	X		X	X	X	X	X	X	X	X	X		X	X	X	
Priority jobs								X							X										X				**X**
Battery management																													
Battery swapping											X									X									
Full charge							X		X			X														X			
Partial Charge						X						X		X				X				X			X				**X**
Search method																													
ALNS			X						X		X			X	X	X					X		X		X	X			**X**
Exact method					X			X			X	X	X	X	X		X	X				X			X			X	**X**
Reinforcement learning approach								X																					
Harmony search																								X					
Genetic algorithm													X																
Shuffled frog leaping algorithm																											X		
Variable neighborhood search		X					X			X										X								X	
Tabu search				X													X												
Memetic particle swarm optimization																			X										
Deterministic annealing					X																								
Simulated annealing	X					X																							

This thesis proposes a metaheuristic based on local search to solve the problem of scheduling multi-load AGVs. The metaheuristic is a combination of adaptive large neighborhood search (ALNS) with a simulated annealing (SA) acceptance criterion. The ALNS framework uses destroy and repair operators to remove a considerable percentage of requests from a solution. After, it attempts to reinsert these requests into a better position. The destroy operators typically include random removal, worst removal and related removal (also referred to as Shaw removal (Shaw, 1997)). Common repair operators are random insertion, greedy insertion and k-regret insertion (Schrimpf et al., 2000; Ropke and Pisinger, 2006). Additional operators are usually added, which are problem specific or even tailored to the problem characteristics (e.g. Häll and Peterson, 2013; Santos and de Carvalho, 2019). Ropke and Pisinger (2006) extend the LNS framework to an adaptive LNS (ALNS) in which the probability of selecting destroy/repair operators is based on its past performance. The ALNS framework has been extensively applied to a variety of vehicle routing problems with promising results, see Demir et al. (2012); Y. Li et al. (2016); Masmoudi et al. (2016); Belhaiza (2019); Gschwind and Drexl (2019).

Singh (2019) proposes an ALNS algorithm for scheduling single-load AGVs and his results are used as a reference point. Additionally, the performance of the proposed hybrid ALNS algorithm is evaluated and compared to a proposed MILP model and a dispatching model.

3 Problem description

This section addresses the problem of scheduling requests that are serviced by a heterogeneous fleet of multi-load AGVs, denoted by V. Each request has a pickup and a delivery node. Since the AGVs are able to carry multiple loads simultaneously, each transport request can be separated into a pickup and delivery task. Let n_R denote the number of transport requests to be served. The set of tasks are defined by $R = P \cup D$, with $P = \{1, ..., n_R\}$ and $D = \{n_R + 1, ..., 2n_R\}$ representing the pickup and delivery tasks, respectively. Therefore, each pickup task r is associated with its corresponding delivery task $n_R + r$, where $r = 1, ..., n_R$. The destination of task r is denoted by d_r^R. Each request has an earliest pickup time e_r^R and a latest delivery time l_r^R. Here, the pickup task $r \in P$ of each request has a hard time window $[e_r^R, \infty]$ since the requests cannot be picked up before their earliest pickup time. On the other hand, the delivery task $r \in D$ of each request has a soft time window $[0, l_r^R]$ and if the transport request arrives after the latest delivery time, a penalty charge incurs proportional to the penalty cost per time unit c_r^R. Requests with higher c_r^R lead to a larger penalty when missing the deadline and hence have higher priority. Each task $r \in R$ requires a set of capability requirements A_r^R. Each AGV $k \in V$ has a set of capabilities A_k^V, thus the AGV is able to perform task $r \in R$ if $A_r^R \subseteq A_k^V$. For example, a request capability can be that the load is placed on a pallet. Therefore, only AGVs that are able to lift a pallet can service this request. Each AGV also has a travel cost per time unit c_k^V. The capacity of AGV $k \in V$ is defined by C^k, which represents the maximum number

of products the AGV can carry. Pickup tasks are associated with a positive load q_r and delivery tasks with a negative load such that $q_r = -q_{n_R+r}$, where $r = 1, ..., n_R$.

The AGVs are operated by battery energy, and their battery levels decrease proportional to the travelled distance with the discharge rate defined by dr_k^V. At the start of each request, the battery level needs to be above the critical threshold bc_k^V. If the battery level drops below the critical threshold while travelling, the AGV must finish its current request and visit a charging station before continuing operations. The critical threshold bc_k^V has to be set to a sufficiently high value such that the battery never gets empty. The charge rate at the charging station is defined by cr_k^V. Partial charging is allowed, so the AGV is permitted to leave the charging station when the battery level surpasses the critical threshold bc_k^V. Notably, the AGV may visit a charging station while it is carrying loads. These loads remain on the AGV until the battery level is high enough to continue.

The problem can be defined on a complete directed graph $G{=}(N, A)$, where $N{=}\{1, 2, ..., n\}$ is the set of nodes and $A = \{(i, j) \mid i, j \in N, i \neq j\}$ is the set of arcs. A node can represent either a pickup/delivery location or a charging station. Each pickup/delivery node has a handling time h_i^N, which represents the time needed to load or unload the requests, while the charging stations have zero handling time. It is assumed that there is sufficient space at the stations for multiple AGVs to load/unload/charge at the same time if needed, thus no waiting time at the stations, and issues such as traffic control and congestion are ignored and left as issues to be considered during real-time control (Ulusoy et al., 1997; Jerald et al., 2006; Gnanavel Babu et al., 2010; Baruwa and Piera, 2016). The AGVs can detect objects on the shop floor and move around them while traveling. Consequently, traffic congestion and collisions between AGVs are already avoided by hardware. Therefore, the distance d_{ij} and travel time t_{ijk} for each arc $(i, j) \in A$ are assumed to be deterministic. Note that the distances and travel times are also symmetrical ($d_{ij} = d_{ji}$ and $t_{ijk} = t_{jik}$).

Let $E \subset N$ denote the set of charging stations and $B = \{1, 2, ..., n_B\}$ denote a set of charging tasks (requests), such that n_B is a sufficiently large number that allows AGVs to charge as many times as needed. Hence, only a subset of B needs to be included in the schedule. All charging requests $r \in B$ have an earliest pickup time and latest delivery time of zero and infinity, respectively ($e_r^R = 0, l_r^R = \infty$), and the load of a charging request is zero (i.e., $q_r = 0$). Let T denote the set of all transport and charging requests as $T = R \cup B$. Additionally, the route of each AGV k starts at an origin $s(k)$ and ends at a destination $e(k)$ which are both at a charging station ($s(k), e(k) \in E$).

The considered problem requires to make decisions on: (a) assigning requests to AGVs, (b) sequencing requests for AGVs, (c) determining the arrival and departure times and (d) determining the duration of recharging at the charging stations. The quality of solutions is characterized by the total tardiness cost of requests and the total travel cost of the AGVs. The goal is to find a feasible solution that satisfies the problem constraints, while optimizing the performance measures.

4 Mathematical formulation

In this section, the mathematical model is formulated based on the problem description in Section 3. This model is inspired by the work of Singh (2019), Keskin and Çatay (2016), Cordeau (2006) and Jorgensen et al. (2007). The model formulation is presented below.

4.1 Decision variables

In addition to the notation used in Section 3, the following decision variables are introduced.

$x_{rr'k}$	binary variable, equal to 1 if AGV k performs request r prior to request r', 0 otherwise
y_{rk}	arrival time of AGV k at the destination of request r
w_{rk}	load on AGV k directly after performing request r
z_{rk}	amount of discharge for AGV k when it reaches the destination of request r
τ_{rk}	tardiness of request r serviced by AGV k
δ_{rk}	travel time of AGV k when transporting request r

4.2 Mixed-integer linear programming model

The mathematical formulation of the described problem is as follows:

Objective:

$$\min \quad \alpha \sum_{k \in V} \sum_{r \in D} c_r^R \cdot \tau_{rk} + (1-\alpha) \sum_{k \in V} \sum_{r \in T} c_k^V \cdot \delta_{rk} \tag{1}$$

Subject to:

$$\sum_{k \in V} \sum_{r' \in T} x_{rr'k} = 1 \qquad \forall r \in P \tag{2}$$

$$\sum_{r' \in T} x_{rr'k} - \sum_{r' \in T} x_{r',n_R+r,k} = 0 \qquad \forall r \in P, k \in V \tag{3}$$

$$\sum_{r' \in T} x_{r'rk} - \sum_{r' \in T} x_{rr'k} = 0 \qquad \forall r \in P \cup D \cup B, k \in V \tag{4}$$

$$\sum_{r' \in T} x_{rr'k} = 0 \qquad \forall r \in P \cup D, k \in V, A_r^R \notin A_k^V \tag{5}$$

$$y_{rk} \geq e_r^R \qquad \forall r \in T, k \in V \tag{6}$$

$$y_{r'k} \geq y_{rk} + h^N_{d_r^R} + t_{d_r^R d_{r'}^R k} - M\left(1 - x_{rr'k}\right) \qquad \forall r, r' \in T, k \in V \tag{7}$$

$$y_{n_R+r,k} \geq y_{rk} + h^N_{d_r^R} + t_{d_r^R d^R_{n_R+r} k} \qquad \forall r \in P, k \in V \tag{8}$$

$$w_{rk} \leq C^k \qquad \forall r \in T, k \in V \tag{9}$$

$$w_{r'k} \geq w_{rk} + q_{r'} - M\left(1 - x_{rr'k}\right) \qquad \forall r, r' \in T, k \in V \tag{10}$$

$$\tau_{rk} \geq y_{rk} + h^N_{d_r^R} - l_r^R \qquad \forall r \in T, k \in V \tag{11}$$

$$\delta_{r'k} \geq t_{d_r^R d_{r'}^R k} \cdot x_{rr'k} \qquad \forall r, r' \in T, k \in V \tag{12}$$

$$z_{r'k} \geq z_{rk} + dr_k^V \cdot t_{d_r^R d_{r'}^R k} - M\,(1 - x_{rr'k}) \qquad \forall r \in P \cup D, r' \in T, k \in V \tag{13}$$

$$z_{r'k} \geq z_{rk} + dr_k^V \cdot t_{d_r^R d_{r'}^R k} - cr_k^V (y_{r'k} - (y_{rk} + h_{d_r^R}^N + t_{d_r^R d_{r'}^R k})) - M\,(1 - x_{rr'k}) \qquad \forall r \in B, r' \in T, k \in V \tag{14}$$

$$z_{rk} \leq b_u - bc_k^V + M\,(1 - x_{rr'k}) \qquad \forall r \in T, r' \in P \cup D, k \in V \tag{15}$$

$$b_l \leq z_{rk} \leq b_u \qquad \forall r \in T, k \in V \tag{16}$$

$$w_{rk}, \tau_{rk}, \delta_{rk} \geq 0 \qquad r \in T, k \in V \tag{17}$$

$$x_{rr'k} \in \{0, 1\} \qquad \forall r, r' \in T, k \in V \tag{18}$$

Objective (1) minimizes the sum of tardiness costs of requests and travel costs of AGVs. The objective criteria are weighted by coefficient α to prioritize one over the other. Constraint (2) ensures that each transport request is served exactly once, and Constraint (3) acts as a coupling constraint, i.e., the pickup and delivery task are performed by the same AGV. Constraint (4) conserves the incoming and outgoing arcs for each request performed by AGV k. Constraint (5) ensures that a request can only be served if the capabilities of the request and the AGV match. Constraint (6) makes sure that the AGV can only start loading the request after the earliest pickup time. Constraint (7) ensures that the arrival time y_{rk} is set correctly along the routes, in which M is a very large number. Constraint (8) acts as a precedence constraint, i.e., the delivery is performed after the corresponding pickup. Constraints (9) and (10) guarantee the consistency in the load variables and prevent the AGV from exceeding the maximum capacity, respectively. Constraint (11) and non-negativity Constraint (17) define the tardiness of each request, while Constraint (12) determines the travel time of each request.

The amount of battery discharge due to travelling is given by Constraint (13). The battery discharge after visiting a charging station is determined by Constraint (14). Constraint (15) sets an upper bound for the amount of battery discharge if the next request is a pickup or delivery, while Constraint (16) sets a lower and upper limit on the battery discharge ($b_l = 0, b_u = 100$) for all (other) requests. Finally, Constraints (17) and (18) guarantee valid domains for the other decision variables.

5 Proposed hybrid ALNS

In this section, a hybrid ALNS algorithm is proposed to solve the problem defined in Section 3. Hybrid ALNS is an integration of ALNS with a local search procedure to exploit the new best or better than current solution's neighborhood for intensification purpose. The ALNS is a framework in which several destroy and repair operators compete to improve the solution. The destroy operators remove requests from the sequence, and the repair operators reinsert the destroyed requests back into the sequence in an attempt to modify the current

solution. The destroy and repair operators are chosen according to a probability based on their past performance.

Let Q_0, S_0, $Q_{current}$, $S_{current}$, Q_{best} and $S_{current}$ define the initial sequence/schedule, current sequence/schedule and best sequence/schedule, respectively. The different request destroy operators are grouped in the set Ω_c^-. Similarly, Ω_c^+, Ω_s^- and Ω_s^+ denote the set of request repair, station destroy and station repair operators, respectively. The individual request destroy and repair operators are denoted by $d_c \in \Omega_c^-$ and $r_c \in \Omega_c^+$, while $d_s \in \Omega_s^-$ and $r_s \in \Omega_s^+$ represent the station destroy and repair operators, respectively. The probabilities of choosing request destroy/repair and station destroy/repair operators are stored in vectors P_c^-, P_c^+, P_s^- and P_s^+, respectively. Algorithm 1 shows the pseudocode of the proposed hybrid ALNS.

The hybrid ALNS algorithm starts with initializing a feasible solution Q_0 and the probabilities of choosing operators in vectors P_c^-, P_c^+, P_s^- and P_s^+ (line 1). At the start, all operators have an equal probability of being chosen. The initial sequence Q_0 is decoded into an initial schedule S_0 by means of a schedule generation scheme which is reffered to as 'SGS' (line 2). Then, the current sequence $Q_{current}$, best sequence Q_{best}, current schedule $S_{current}$ and best schedule S_{best} are updated (line 3). The SA temperature θ is set to its initial temperature θ_{init} (line 4). Next, the algorithm executes a loop that will iterate until the stopping criterion is met (lines 6-59). The loop starts by selecting a request destroy operator $d_c \in \Omega_c^-$ based on probability P_c^- and a request repair operator $r_c \in \Omega_c^-$ based on probability P_c^+ (line 7). The chosen destroy and repair operators are applied to the current sequence $Q_{current}$, which results in a new sequence Q_{new} (line 8). The new schedule S_{new} is computed with input Q_{new} (line 9). If a new best solution is found, the best sequence and schedule are updated, and the variable ζ, controlling the number of iterations without improving the best solution, is reset to zero (lines 12-14). If the new solution is better than the best or current one, the solution space is exploited by a local search procedure and station destroy/repair operators until n_{LS} and n_S consecutive iterations do not improve the best or current solution, respectively (lines 18-45). During both the local search procedure and the station destroy/repair iterations, worse solutions are not accepted.

If the new solution is not better than the current or best solution, it is accepted if it satisfies the SA acceptance criterion, in which $r^{U(0,1)}$ is a random value between 0 and 1, generated according to the uniform distribution $U(0, 1)$ (lines 48-50). The probabilities of choosing a destroy and repair operator for both requests and station operators are updated based on their performance (lines 46 and 51). Next, the SA temperature is updated, in which ϵ is the cooling rate (line 52). If the best solution is not improved after n_D consecutive iterations, a method is applied to diversify the search (lines 53-58). Finally, if the stopping criterion (line 59) is met, the algorithm returns the best solution (line 60). Particular components of the algorithm are explained in more detail in the sections corresponding to the comments in the pseudocode.

Algorithm 1 Hybrid adaptive large neighborhood search

Require: set of AGV V, set of transport requests R, set of charging tasks B

Initialize Q_0 and $P_c^-, P_c^+, P_s^-, P_s^+$ ▷ Sections 5.2 and 5.3
$S_0 \leftarrow SGS(Q_0)$ ▷ Section 5.1
$Q_{current}, Q_{best} \leftarrow Q_0; S_{current}, S_{best} \leftarrow S_0$
$\theta \leftarrow \theta_{init}$ ▷ Section 5.6
$\zeta \leftarrow 0$
repeat
  Select $d_c \in \Omega_c^-$, $r_c \in \Omega_c^+$ using P_c^-, P_c^+ ▷ Section 5.4.1
  $Q_{new} \leftarrow r_c\left(d_c\left(Q_{current}\right)\right)$
  $S_{new} \leftarrow SGS(Q_{new})$
  $\zeta \leftarrow \zeta + 1$
  if $f(S_{new}) < f(S_{best})$ **or** $f(S_{new}) < f(S_{current})$ **then**
    if $f(S_{new}) < f(S_{best})$ **then**
      $Q_{best} \leftarrow Q_{new}$, $S_{best} \leftarrow S_{new}$
      $\zeta \leftarrow 0$
    end if
    $Q_{current} \leftarrow Q_{new}$, $S_{current} \leftarrow S_{new}$
    $\upsilon, \xi \leftarrow 0$
    while $\upsilon \leq n_{LS}$ **do**
      $Q_{new} \leftarrow LS(Q_{current})$ ▷ Section 5.5
      $S_{new} \leftarrow SGS(Q_{new})$
      $\zeta \leftarrow \zeta + 1$
      if $f(S_{new}) < f(S_{best})$ **then**
        $Q_{best}, Q_{current} \leftarrow Q_{new}$; $S_{best}, S_{current} \leftarrow S_{new}$
        $\upsilon, \zeta \leftarrow 0$
      else if $f(S_{new}) < f(S_{current})$ **then**
        $Q_{current} \leftarrow Q_{new}$, $S_{current} \leftarrow S_{new}$
        $\upsilon \leftarrow 0$
      else
        $\upsilon \leftarrow \upsilon + 1$
      end if
    end while
    while $\xi \leq n_S$ **do**
      Select $d_s \in \Omega_s^-$, $r_s \in \Omega_s^+$ using P_s^-, P_s^+ ▷ Section 5.4.3
      $Q_{new} \leftarrow r_s\left(d_s\left(Qcurrent\right)\right)$
      $S_{new} \leftarrow SGS(Q_{new})$
      $\zeta \leftarrow \zeta + 1$
      if $f(S_{new}) < f(S_{best})$ **then**
        $Q_{best}, Q_{current} \leftarrow Q_{new}$; $S_{best}, S_{current} \leftarrow S_{new}$
        $\xi, \zeta \leftarrow 0$
      else if $f(S_{new}) < f(S_{current})$ **then**
        $Q_{current} \leftarrow Q_{new}$, $S_{current} \leftarrow S_{new}$
        $\xi \leftarrow 0$
      else
        $\xi \leftarrow \xi + 1$
      end if
      Update P_s^-, P_s^+ ▷ Section 5.3
    end while
  else if $r^{U(0,1)} < e^{-(f(S_{new}) - f(S_{current}))/\theta}$ **then**
    $Q_{current} \leftarrow Q_{new}$, $S_{current} \leftarrow S_{new}$
  end if
  Update P_c^-, P_c^+ ▷ Section 5.3
  $\theta \leftarrow \theta \cdot \epsilon$ ▷ Section 5.6
  if $\zeta > n_D$ **then**
    $Q_{new} \leftarrow DIV\left(Q_{current}\right)$ ▷ Section 5.5
    $S_{new} \leftarrow SGS(Q_{new})$
    $Q_{current} \leftarrow Q_{new}$, $S_{current} \leftarrow S_{new}$
    $\zeta \leftarrow 0$
  end if
until stopping criterion is met
return S_{best}

5.1 Solution representation

The solution of the problem is a schedule in which tasks are carried out at specific time moments. Let g_r^R denote the type of task r which can be either a pickup (P), a delivery (D) or a charging (B) task. Then, the start and end time of a task r serviced by AGV k are defined by y_{rk}^S and y_{rk}^D, respectively. Note that the start time of a task is equal to the end time of the previous task (i.e., $y_{rk}^S = y_{r-1,k}^D$). Finally, let β_{rk} represent the battery level at the start of task r serviced by AGV k.

The output of the destroy and repair operators in the ALNS framework is a sequence of tasks which can be decoded into a feasible solution by means of a schedule generation scheme. In particular, a serial schedule generation scheme is applied which makes use of a task-incrementation principle to schedule the tasks in the order dictated by the task sequence (Lambrechts et al., 2008). Each task is scheduled at its earliest possible starting time such that no constraints are violated. The schedule generation scheme introduces calculation variables Y_k and $\mathcal{B}_k$ that represents the current time and current battery level, respectively. The initial time is set to zero and the initial battery level is defined by β_{0k}. The pseudocode of the schedule generation scheme is shown in Algorithm 2.

The algorithm loops over each AGV $k \in V$ (line 1) and initializes the current battery level $\mathcal{B}_k$ and time Y_k (lines 2-3). For every task r in task sequence Q^k, the algorithm checks if the task is either a charging, pickup or delivery task (lines 5 and 25). If it is a charging task, the battery level at the start of the task β_{rk} and the start time of the task y_{rk}^S are updated (lines 6-7). Next, the battery level upon arrival at the charging station is calculated by decreasing the current battery level $\mathcal{B}_k$ according to the discharge rate dr_k^V multiplied by the travel time of the task $t_{d_{r-1}^R d_r^R}$ (line 8). In order to determine the charging duration, the minimal required charge $\mathcal{B}_R$ is introduced which represents the minimal amount of charge needed to service all tasks up to the next charging request in the task sequence (lines 10-16). In that case, the battery level needs to be increased up to a certain value so that the last task before the next charging request starts with a battery level equal to the critical threshold. Hence, the required charge variable $\mathcal{B}_R$ is increased for the tasks up to the last task i before the next charging request, where $i+1 \subseteq B$ holds (lines 11-12). Additionally, the required charge is increased by the difference between the critical threshold and the current charge level upon arrival at the charging station (line 17).

If the next task $r+1$ is a pickup, which cannot be serviced before its earliest pickup time e_{r+1}^R, it might be possible to charge more than only the minimal required charge. Hence, the possible charge $\mathcal{B}_P$ is calculated (lines 18-20). Then, the maximum of the required charge $\mathcal{B}_R$ and the possible charge $\mathcal{B}_P$ is taken and added to the current charge (line 21). Next, the current time Y_k and the end time of the task are updated (lines 22-23). Finally, if the task $r \in Q^k$ is a pickup or delivery task, the start time, end time and battery level at the start of the task are determined (lines 25-33).

Algorithm 2 Schedule generation scheme

Require: Task sequence Q, initial battery level β_{0k}

for each $k \in V$ **do**
  $\mathcal{B}_k \leftarrow \beta_{0k}$
  $Y_k \leftarrow 0$
  for each $r \in Q^k$ **do**
    if $r \subseteq B$ **then**
      $\beta_{rk} \leftarrow \mathcal{B}_k$
      $y^S_{rk} \leftarrow Y_k$
      $\mathcal{B}_k \leftarrow \mathcal{B}_k - dr^V_k \cdot t_{d^R_{r-1}d^R_r}$
      $\mathcal{B}_R, \mathcal{B}_P \leftarrow 0$
      for $i \in Q^k(r+1, ..., |Q^k|-1)$ **do**
        if $i+1 \not\subseteq B$ **then**
          $\mathcal{B}_R \leftarrow \mathcal{B}_R + dr^V_k \cdot t_{d^R_{i-1}d^R_i}$
        else
          break
        end if
      end for
      $\mathcal{B}_R \leftarrow max(0, \mathcal{B}_R + bc^V_k - \mathcal{B}_k)$
      if $r+1 \in P$ **then**
        $\mathcal{B}_P \leftarrow max(0, cr^V_k \cdot (e^R_{r+1} - Y_k - t_{d^R_{r-1}d^R_r} - t_{d^R_r d^R_{r+1}}))$
      end if
      $\mathcal{B}_k \leftarrow \mathcal{B}_k + max(\mathcal{B}_R, \mathcal{B}_P)$
      $Y_k \leftarrow Y_k + t_{d^R_{r-1}d^R_r} + \mathcal{B}_R / cr^V_k$
      $y^D_{rk} \leftarrow Y_k$
    end if
    if $r \subseteq P \cup D$ **then**
      $\beta_{rk} \leftarrow \mathcal{B}_k$
      $Y_k \leftarrow max\left(Y_k, e^R_r\right)$
      $y^S_{rk} \leftarrow Y_k$
      $\mathcal{B}_k \leftarrow \mathcal{B}_k - dr^V_k \cdot t_{d^R_{r-1}d^R_r}$
      $Y_k \leftarrow Y_k + t_{d^R_{r-1}d^R_r} + h^N_{d^R_r}$
      $y^D_{rk} \leftarrow Y_k$
    end if
  end for
end for

After generating the schedule, its quality is measured by the weighted sum of the tardiness costs of requests and travel costs of the AGVs according to Equation 19:

$$f(S) = \alpha \sum_{k \in V} \sum_{\substack{r \in S_k | \\ g^R_r = D}} c^R_r \cdot \left(y^D_r - l^R_r\right)^+ + (1-\alpha) \sum_{k \in V} \sum_{r \in S^k} c^V_k \cdot t_{d^R_{r-1}d^R_r k} \tag{19}$$

where S^k denotes the schedule of AGV k, $g^R_r = D$ states that the task is a delivery and $\left(y^D_r - l^R_r\right)^+$ defines the tardiness of the requests.

5.2 Initial solution construction

According to the experimental results, the quality of the initial solution has a crucial impact on the final outcome, especially since the computational time is limited. The initial sequence

of tasks is generated by a constructive heuristic, in combination with the critical charge insertion method (CCI) presented in Section 5.4.4. The constructive heuristic is inspired by the work of Hosny and Mumford (2012). The authors insert a new request in a feasible position in the best route while considering hard time windows. In this thesis, the proposed constructive heuristic looks for the best position in the best route to insert a request. The pseudocode of the constructive heuristic is shown in Algorithm 3.

The algorithm starts with ordering all requests r according to their latest delivery time l_r^R (line 1). Next, the algorithm loops over the requests in the sorted set $\hat{R}$ (lines 2-13). Then, a subset of capable AGVs V_r is determined for servicing request r (line 3). Let $Q_{r\to i}^k$ and $S_{r\to i}^k$ denote the task sequence and schedule when request r is inserted in position i, respectively. The objective value $f(S_{r\to i}^k)$ is calculated for each feasible insertion position i in AGV k (lines 4-9). The request (a pickup and delivery pair) is inserted into the position where it leads to the minimum added cost to the objective value $f(S_{r\to i}^k)$ (lines 10 and 11). Notice that inserting the pickup and delivery task of each request together overcomes the precedence and coupling constraints and speeds up the insertion process. Finally, the algorithm applies the CCI method described in Section 5.4.4 to ensure that the solution is always feasible with respect to the battery constraints.

Algorithm 3 Constructive heuristic

Require: Set of AGVs V, set of transport requests R

1: $\hat{R} \leftarrow \text{sort}(R \mid r \prec r' \Rightarrow l_r^R < l_{r'}^R, \forall r, r' \in R)$
2: **for each** $r \in \hat{R}$ **do**
3: $\quad V_r \leftarrow \{k \in V \mid A_r^R \subseteq A_k^V\}$
4: $\quad$ **for each** $k \in V_r$ **and** $i \in \{1, ..., |Q^k|\}$ **do**
5: $\quad\quad$ **if** $w_{ik} < C^k$ **then**
6: $\quad\quad\quad S_{r\to i}^k \leftarrow SGS(Q_{r\to i}^k)$
7: $\quad\quad\quad f(S_{r\to i}^k) \leftarrow \alpha \sum_{k\in V} \sum_{\substack{\bar{r}\in S_{r\to i}^k| \\ g_{\bar{r}}^R = D}} c_{\bar{r}}^R \cdot \left(y_{\bar{r}}^D - l_{\bar{r}}^R\right)^+ + (1-\alpha) \sum_{k\in V} \sum_{\bar{r}\in S_{r\to i}^k} c_k^V \cdot t_{d_{\bar{r}-1}^R d_{\bar{r}}^R k}$
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: $\quad (k^*, i^*) \leftarrow \text{argmin}\{f(S_{r\to i}^k) \mid \forall k \in V_r, \forall i \in \{1, ..., |Q^k|\}\}$
11: $\quad$ Insert r at position i^* in sequence Q^{k^*}
12: $\quad$ Apply $CCI(Q^{k^*})$ ▷ Section 5.4.4
13: **end for**

5.3 Adaptive selection of destroy/repair operators

At each iteration, a destroy and a repair operator are selected using a roulette wheel mechanism (Ropke and Pisinger, 2006). The repair operator is chosen independently of the destroy operator and vice versa. The probability of choosing an operator is updated based on the performance of past iterations of that operator. Successful operators will therefore obtain a higher weight and will be selected more often. The weight of operator $d_c \in \Omega_c^-$ is defined by $\rho_c^- \in P_c^-$ and is selected with probability $\rho_c^- / \sum_{d_c \in \Omega_c^-} \rho_c^-$, and the weights for $r_c \in \Omega_c^+$, $d_s \in \Omega_s^-$ and $r_s \in \Omega_s^+$ are defined similarly. Initially, all weights are equal to 1. The performance of an operator is measured by the score value ψ. If an operator finds a new best

solution, its score is rewarded by ψ_1. If a better solution than the current one is found, its score is rewarded by ψ_2, and if a new deteriorating solution is found, but accepted according to the SA acceptance criterion, its score is rewarded by ψ_3. After each iteration, the weights of the used operators are updated by the following equations:

$$\rho_x^- = \lambda\rho_x^- + (1-\lambda)\psi \tag{20}$$

$$\rho_x^+ = \lambda\rho_x^+ + (1-\lambda)\psi \tag{21}$$

where reaction factor $\lambda \in [0,1]$ controls the speed of the weight adjustment and takes the success of the previous iterations into account. The index x represents either c (for request methods) or s (for station methods). The main idea of using this adaptive mechanism instead of only using the best performing operators is that it is not known which operators will be successful in a particular instance. In addition, worse performing operators could lead to more diversification and eventually result in a better solution.

5.4 Description of destroy and repair operators

In this section, the individual destroy and repair operators are discussed. There are two different types of destroy/repair operators: request destroy/repair operators, which remove and reinsert transport requests, and station destroy/repair operators, which remove and reinsert charging requests. Note that if transport requests are removed from a task sequence, they must be reinserted. For charging requests, this does not have to be the case. The destroy and repair operators of the proposed hybrid ALNS are either adapted or inspired by Ropke and Pisinger (2006), Keskin and Çatay (2016) and Shaw (1997).

5.4.1 Request destroy operators

In all request destroy operators, only complete requests are considered, that is, both the pickup and delivery tasks. Due to the coupling and precedence constraints (i.e., the pickup must be performed before the delivery and by the same AGV), only removing the pickup or delivery task strongly bounds the reinsertion possibilities (Häll and Peterson, 2013). Hence, both the pickup and delivery tasks are removed simultaneously. This also allows the repair operator to reinsert the request into the task sequence of another AGV.

This section introduces five destroy operators, where each one removes q customers from the task sequence and returns a partial destroyed sequence. The request destroy operators are described as follows.

Shaw destroy (SHD): Shaw (1997) first introduced the Shaw destroy operator. The idea is to destroy similar requests, since they can be easily reshuffled in the task sequence. The similarity of requests is based on the distance of both the pickup and delivery nodes, the time window (i.e., earliest pickup time and latest delivery time) and the capability requirements. The similarity of requests r and r' is determined using a relatedness measure $\mathcal{R}(r,r')$, according to Equation (22):

$$\mathcal{R}(r,r') = \phi_1(d_{d_r^P d_{r'}^P} + d_{d_r^D d_{r'}^D}) + \phi_2(|e_r^R - e_{r'}^R| + |l_r^R - l_{r'}^R|) + \phi_3\left(1 - \frac{|V_r \cap V_{r'}|}{min\{|V_r|, |V_{r'}|\}}\right) \quad (22)$$

where ϕ_1, ϕ_2 and ϕ_3 represent the Shaw weight parameters corresponding to the distance, time window and capability requirement terms, respectively, and where V_r denotes the set of AGVs capable of servicing request r. A lower value of $\mathcal{R}(r,r')$ indicates a greater similarity of the requests. The values of d_{xy}, e_x^R and l_x^R are normalized by scaling them such that they only take on values from $[0,1]$. Hence, the relatedness measure is bounded by $0 \leq \mathcal{R}(r,r') \leq 2(\phi_1+\phi_2)+\phi_3$. The Shaw destroy operator starts by randomly selecting a request to destroy. The other $n_R - 1$ transport requests are sorted according to their similarity to the selected request. Then, the request in position $[y^\eta(n_R-1)]$ is selected to be destroyed, where $y \in [0,1]$ is a random number and η is a deterministic parameter to introduce some randomness in selecting the requests. This procedure reiterates until q requests are removed from the task sequence (Keskin and Çatay, 2016; Ropke and Pisinger, 2006).

Random request destroy (RRD): This operator randomly selects q requests to destroy. Randomly destroying requests can help in the diversification of the search mechanism.

Worst travel cost destroy (WTRD): This operator orders the requests according to their travel cost (i.e., travel time multiplied by the travel cost of the AGV) and destroys the q requests with the highest travel cost.

Worst tardiness cost destroy (WTAD): This operator sorts the requests according to their tardiness cost (i.e., tardiness multiplied by the penalty cost) and destroys the q requests with the highest tardiness cost.

Worst task destroy (WTD): The worst task destroy operator is a combination of both the WTRD and WTAD operators. It sorts the requests according to their added score to the objective value (i.e., sum of tardiness cost and travel cost of the request) and destroys the q requests that add the highest score to the objective value.

5.4.2 Request repair operators

The selected destroy operator returns a partially destroyed sequence and a list of destroyed requests, which are used as inputs for a chosen repair operator. The output of the repair operator is a feasible sequence of requests. The list of destroyed requests contains both pickup and delivery tasks. The repair operator inserts the related pickup and delivery tasks directly after one another into the partially destroyed sequence to bypass the coupling and precedence constraints. If the pickup and delivery tasks were inserted separately, the repair operator would need to keep track of the position of the related pickup or delivery task. It could also lead to a deadlock if there is no feasible position to insert a task due to capacity constraints. Hence, the repair operators insert the pickup and delivery task directly after each other. After the insertion of a request, the local search procedure is applied to the delivery task of the request. The local search procedure consists of an intra-route relocate method, which checks for a better insertion position for the delivery task in the next ten positions

at maximum from the related pickup. Further details about the local search procedure are explained in Section 5.5. The request repair operators are described as follows.

Greedy insertion (GI): The greedy insertion heuristic is similar to the proposed constructive heuristic (Algorithm 3), which is used to create the initial solution. It iterates through all destroyed requests and inserts them into their best possible positions. The differences are that the GI heuristic only inserts the destroyed requests instead of all requests, and after the insertion of the request in line 11 of Algorithm 3, the local search procedure is performed to relocate the delivery task of the request in the sequence. The GI heuristic is applied in the hybrid ALNS framework in two versions. In the first version, it iterates through the destroyed requests in the order in which they are destroyed (GID), and in the second, the operator iterates randomly through the destroyed requests (GIR).

Randomized greedy insertion (RGI): This operator is an adaption of the GI heuristic. Instead of inserting a request into its best position, the operator selects a random integer in the interval [1,3] and uses it as index to insert the pickup and delivery pair. For example, if the random integer is 2, the operator inserts the request into its second best position. Similar to the GI heuristic, the local search procedure is performed and two versions are applied, either iterating though the destroyed requests in the destroy order (RGID) or at random (RGIR).

Travel cost insertion (TC): This operator inserts the requests into the position in which they add the least travel cost to the objective value. This operator is similar to the GI heuristic but does not consider tardiness of requests in the calculation for the cost of the possible insertion position. Hence, the only difference is that the objective function (line 7 of Algorithm 3) is replaced by Equation (23).

$$f(S^k_{r\to i}) = \sum_{k\in V} \sum_{r\in Q^k_{r\to i}} c^V_k \cdot t_{d^R_{r-1} d^R_r k} \tag{23}$$

This operator is also applied in two versions, by inserting in the destroy order (TCD) or at random (TCR).

Earliest delivery time insertion (EDT): All the repair operators described above are computationally expensive because they calculate the cost of insertion for each feasible position in the sequence. The EDT operator is faster since it only reinserts a destroyed request into the sequence of capable AGV $k \in V_r$ at the first position i, where $l^R_r < l^R_{Q^k_{i+1}}$. The idea behind this operator is that requests with earlier delivery times should be serviced before requests with later delivery times.

5.4.3 Station destroy operators

Positioning the charging requests in the task sequence is crucial to the AGV scheduling problem. Hence, removing or repositioning the charging requests could improve the solution. The station destroy operators are described as follows.

Worst station destroy (WSD): This operator randomly selects an AGV $k \in V$ and calculates the increase in the charge level for each charging request. The one that leads to the smallest increase is removed from the sequence.

Random station destroy (RSD): This operator chooses a random charging request to remove from the sequence of a randomly selected AGV $k \in V$.

5.4.4 Station repair operators

After removing charging requests, the partially destroyed solution may become infeasible with respect to the battery constraints. In order to repair the solution, one or more charging requests may have to be inserted in the infeasible sequence. Unlike the request repair operators, the station repair operators do not necessarily have to reinsert all destroyed charging requests. If a charging request is inserted, the charging takes place at the charging station closest to the location of the AGV's previously serviced request. The station repair operators are described as follows.

Critical charge insertion (CCI): This operator determines the first request at which the AGV reaches the destination with a battery level below the critical threshold bc_k^V and inserts a charging request afterwards since the battery level only needs to be above the critical threshold at the start of a task.

Critical charge insertion with comparison (CCIC): Similar to the CCI method, this operator determines the battery charge level after servicing each request in the sequence and finds the position in which the battery level drops below the critical threshold bc_k^V. The CCIC method compares the insertion of a charging request in the critical position and the position before that and inserts the charging request in the best position of the two.

Non-critical charge insertion (NCCI): This operator starts by inserting a critical charge request when necessary according to the CCI method. Additionally, it inserts an extra non-critical charge request. The NCCI operator calculates the objective value for all possible insertion positions and inserts the charge request in the position that results in the best objective value.

5.5 Local search and diversification methods

A local search procedure is proposed to intensify the search in a promising region of the search space. The local search procedure is inspired by B. Li et al. (2016) and uses intra-route relocation to reposition the pickup and delivery tasks within the sequence. The method randomly removes either a pickup or a delivery task from the sequence and reinserts the task into its best feasible position. Note that during the insertion the coupling and precedence constraints need to be considered to guarantee a feasible solution. The local search procedure only accepts improvements and continues until the current or best solution is not improved over n_{LS} consecutive iterations.

In the ALNS framework a diversification method is proposed, which is inspired by the work of Y. Li et al. (2016). The diversification method is applied after a maximum number

n_D of successive iterations without improving the best solution. In that case the algorithm may be stuck in a local optimum. The diversification method consists of a destroy/repair iteration in which the RRD operator removes 40% of the requests from the sequence and the GI operator repairs the solution. The value of 40% is chosen since it is commonly used in literature as an upper bound for the number of tasks to remove (Ropke and Pisinger, 2006; Keskin and Çatay, 2016; Zhao and Lu, 2019). Additionally, the CCI operator is applied to guarantee a feasible solution with respect to the battery constraints. The new solution is always accepted, and the search continues from this point.

5.6 Acceptance and stopping criterion

The SA acceptance criterion is used to allow the hybrid ALNS to explore more promising regions at the beginning and focus more on exploitation at the end of the search. A better solution than the current or best solution is always accepted, while a deteriorating solution is accepted with probability $e^{-(f(S_{new})-f(S_{current}))/\theta}$, where θ is the current temperature. The temperature θ is initialized as θ_{init} and decreases gradually via a cooling rate ϵ, where $0 < \epsilon < 1$. The initial temperature θ_{init} is set to $-\omega \cdot f(S_0)/ln(0.5)$, where ω is the initial control parameter (Ropke and Pisinger, 2006; Zhao and Lu, 2019). The stopping criterion of the hybrid ALNS is denoted by a user defined time limit on the computational time.

6 Established practice

The common practice for controlling a fleet of multi-load AGVs is to use a set of dispatching rules. Multi-load AGVs normally have a high utilization and are therefore not expected to remain idle for long periods. Hence, the dispatching rules are vehicle-initiated rather than load-initiated. The decision-making problems associated with dispatching multi-load AGVs are task-determination, pickup-dispatching and delivery-dispatching problems (Y.-C. Ho and Chien, 2006). The task-determination problem decides if the AGV's next task should be a pickup task, a delivery task, or a charging task. The pickup- (delivery-)dispatching problem determines which pickup (delivery) task to perform first if the task-determination problem decides that the next task is a pickup (delivery).

The solution approach for the task-determination problem is inspired by the pickup-or-delivery-en-route rule presented in M. P. Li and Kuhl (2017). With the layout of the case study introduced in Section 7 (see Figure A.1), the AGV does not pass other pickup/delivery stations en route. The adapted rule to solve the task-determination problem states that the next task is a pickup task if there is another task to be picked up within the same department and selects a delivery task otherwise. The different departments are shown in the layout of the problem in Figure A.1. If the AGV's battery level is below the critical threshold, the next task is always a charging task. Additionally, a non-critical threshold bn_k^V is introduced. If an AGV is idle, it is allowed to move to a charging station only if the battery level is below the non-critical threshold. For the pickup-dispatching problem, the shortest-distance

rule is applied. So, the AGV is sent to the pickup task closest to its current location. The delivery-dispatching problem occurs if there are multiple loads on the AGV and the next task is a delivery. This problem is solved by the earliest-due-date rule, so the AGV delivers the item with the earliest delivery time first.

It is observed in the experiments that the dispatching rules work well for this problem setting in general. The solution of the problem using dispatching rules is used as a benchmark for the hybrid ALNS in the computational results in Section 7. The pseudocode of the established practice for dispatching multi-load AGVs is shown in Algorithm 4.

Algorithm 4 Dispatching model

Require: Set of AGVs, set of transport requests, layout, guide-path, task-determination rule, delivery-dispatching rule, pickup-dispatching rule, critical threshold, non-critical threshold

```
1: while requests do
2:     for each AGV do
3:         if critical threshold not met or
4:         (AGV empty and no waiting task and non-critical threshold not met) then
5:             Start charging at nearest charging station
6:             while battery threshold not met or no waiting task do
7:                 Continue charging
8:             end while
9:         else
10:            Determine next task type
11:            if delivery then
12:                Apply delivery-dispatching rule
13:            else if pickup then
14:                Apply pickup-dispatching rule
15:            end if
16:        end if
17:    end for
18: end while
```

7 Computational results

To evaluate the performance of the hybrid ALNS, computational experiments were conducted on scenarios with varying numbers of requests, AGV fleet sizes, tightness of time windows, AGV capacities and computational times. For small scenarios, the results are compared to the MILP model presented in Section 4 and to the dispatching model described in Section 6. For larger scenarios with more requests and AGVs, the performance of the hybrid ALNS is only compared to the dispatching model, since the MILP model cannot obtain a feasible solution within the time limit of the experiments. In both the comparison against the MILP model and the sensitivity analysis, the hybrid ALNS model is run with a time limit of 15 seconds, except in Section 7.5.2, where the computation time is varied. For all experiments, the weight factor of the objective function, α, is set to 0.5.

All experiments were conducted on a computer with Intel(R) Core(TM) i7-3630QM 2.40 GHz processor and 8 GB RAM with 64-bit Windows 7 operating system. The hybrid ALNS algorithm was coded in Python v3.6, the MILP model was solved by the Gurobi Python

package and the dispatching model was implemented in the simulation software Anylogic University v8.3.1.

The following sections describe the generation of instances, parameter settings used in the hybrid ALNS, comparisons against the MILP model and dispatching model and finally a sensitivity analysis in which the performance is also compared to a proposed matheuristic for single-load AGVs by Singh (2019).

7.1 Generation of instances

The instances in this study are generated for the layout shown in Figure A.1, and there are 10 instances generated for each scenario used in the experiments. Each scenario has the name format TW-N-A-C. The time window of a request is randomly generated and TW represents the probability that determines if the request has a relatively tight or loose time window. A higher TW value results in more requests with a tight time window. N, A, and C define the number of requests, the number of AGVs, and capacity of AGVs, respectively. For all instances, the capacity of all AGVs is equal, the time horizon in which the requests randomly arise is set to 1800 seconds and the tardiness cost of requests can be either 1 or 10, to make a distinction between high- and low-priority requests. The heterogeneous fleet contains AGVs with different travel costs. The user-defined parameters of AGVs including speed, initial charge, discharging and charging rates, travel cost and critical battery threshold are reported in Table A.3. The pseudocode to generate an instance for given TW, N, A and C values is shown in Algorithm 5 in Appendix A.

7.2 Parameter tuning

The parameter tuning methodology for the hybrid ALNS is in line with those adopted in the literature (Ropke and Pisinger, 2006; Demir et al., 2012; Keskin and Çatay, 2016). The parameters in Table 7.1 are tuned based on five scenarios and two generated instances per scenario, thus 10 instances in total. The names of the scenarios used for the parameter tuning are reported in Table A.1. Ten values are considered for each parameter and each instance is run ten times for each parameter value. For each value, the tuning procedure calculates the average percent of deviation from the average of the best achieved solutions, determines the one that has the least average percent of deviation, and fixes this parameter value. This procedure is repeated until all parameter values have been tuned and their best values are summarized in Table 7.1.

The complete results of the parameter tuning for each parameter value are given in Table A.2. Additionally, a lower and upper bound for the number of requests to remove is set to $\underline{q}$=min$\{0.05|N|,4\}$ and $\overline{q}$=min$\{0.25|N|,10\}$, respectively. No claim is made that these parameter values are optimal, but the experiments have shown that they perform well.

Table 7.1: Obtained values of parameter tuning.

Parameter	Description	Value
n_{LS}	# of iterations between which the local search is performed	25
n_S	# of iterations during which station destroy/repair operators are performed	7
n_D	# of iterations that invokes the diversification method	60
ψ_1	Score of the best solution	0.8
ψ_2	Score of the better solution	0.65
ψ_3	Score of the worse solution	0.5
λ	reaction factor for the weight adjustment	0.69
η	SHD randomness parameter	9
ϕ_1	SHD weight distance	2.5
ϕ_2	SHD weight time window	1.2
ϕ_3	SHD weight capability	2
ω	SA initial temperature control parameter	0.005
ϵ	SA cooling rate	0.995

7.3 Performance of search methods

The hybrid ALNS framework consists of several components. Table 7.2 reports the time it takes to construct the initial solution and apply a single iteration of request destroy/repair, station destroy/repair, local search procedure and diversification method.

Table 7.2: CPU time of the ALNS components for different number of requests (in seconds).

	Number of requests				
Search method	20	40	60	80	100
Initial solution construction	0.220	1.220	4.070	8.114	14.856
Request destroy/repair	0.057	0.120	0.304	0.499	0.773
Station destroy repair	0.006	0.011	0.025	0.056	0.059
Local search procedure	0.015	0.037	0.094	0.190	0.256
Diversification method	0.245	1.159	4.252	9.251	15.576

The time to create the initial solution increases exponentially for larger scenarios since the constructive heuristic looks for the best insertion position for a request within all feasible insertion positions and this increases with the number of requests. Notice that creating the initial solution for a scenario with 100 requests already takes close to 15 seconds. Therefore, scenarios of 100 requests are the limit for the hybrid ALNS algorithm for a computational time of 15 seconds. The computational time required to perform an iteration of request destroy and repair operators increases nearly linear due to the upper bound on the number of tasks to destroy. The station destroy/repair operators and the local search procedure are significantly faster, while the diversification method is particularly slow, since it must remove and reinsert 40% of the requests. For different computational times, the results in Table 7.2 can be used to make an educated estimate on the number of iterations that can be performed within a given computational time.

To show some insight into the different search methods, their performance is analyzed in Table 7.3. The table shows, for each search method, the frequency of use as a percentage

of the total number of iterations in column 'Used'. The percentage of total computation time spent on each method is reported between brackets. The performance of each method is reported as the percentage of times it finds a new best solution (Best) and leads to an improvement of the current solution (Imp.). The results are generated by performing experiments on the same scenarios as the ones used for the parameter tuning.

Table 7.3: Impact of search methods.

Search method	Used [%]	Best [%]	Imp. [%]
Request destroy/repair	13.6 (34.5)	20.9	23.2
Station destroy/repair	4.1 (0.9)	10.3	10.5
Local search procedure	82.2 (63.8)	10.0	10.2
Diversification method	0.03 (0.8)	0	0

The number of times each method is applied is influenced by the user-defined parameters, n_{LS}, n_S and n_D given in Table 7.1. By using the best parameter values from the parameter tuning, the search is dominated by the local search procedure (82.2% of the total number of iterations and 63.8% of the computational time). The request destroy/repair operators have a higher chance of improving the best solution (20.9%) than the local search procedure (10.0%) does since they reposition more requests within a single iteration. However, the request destroy/repair operators take more time. Notice that the performance of the request destroy/repair operators also affects the number of times the other methods are used since if it finds a new best or improving solution, the local search procedure and station destroy/repair operators are applied. The diversification method is only used for 0.03% of the iterations, which shows that the search rarely gets stuck in a local optimum within a computation time of 15 seconds. The station destroy/repair operators are applied in 4.1% of the iterations, and they improve the best solution in 10.3% of the time. This shows the importance of repositioning charging requests in the task sequence and that it can lead to an improvement of the solution.

Furthermore, the effectiveness of the individual destroy and repair operators is recorded and shown in Table 7.4. The results show that the frequencies of use for different operators do not significantly vary from one another. The request destroy operators all perform similarly, except the Shaw destroy operator, which leads to slightly less improvement. For the request repair operators, most improvement is achieved by GID, GIR and RGIR. For the station destroy and repair operators, WSD and CCI perform best. However, it is important to observe that all the operators play non-negligible roles in the improvements to the best and current solution.

Table 7.4: Performance of individual operators.

Group	Operator	Chosen [%]	Best [%]	Imp. [%]
Request destroy	RRD	17.1	26.0	27.9
	SHD	22.7	8.9	12.1
	WTAD	22.3	24.9	25.1
	WTRD	18.5	23.1	23.6
	WTD	19.4	21.7	29.7
Request repair	GID	15.2	23.4	24.1
	GIR	16.2	25.1	31.4
	RGID	13.4	16.2	19.3
	RGIR	15.2	24.3	24.6
	EDT	12.8	13.5	13.6
	TCD	14.1	15.5	18.2
	TCR	13.2	14.3	17.7
Station destroy	RSD	50.9	7.8	8.0
	WSD	49.1	11.9	12.1
Station repair	CCI	24.9	15.5	15.9
	CCIC	37.8	13.0	13.2
	NCCI	37.3	10.2	10.6

7.4 Numerical experiments for small-sized problems

In this section, the performance of the proposed hybrid ALNS, exact MILP and dispatching model (DP) is studied for small-sized problems. Test instances are generated with varying the numbers of requests in the range N=[3,6,9,12], while the number of AGVs is set to 3, the tightness of the time windows is 0.5 and the capacity of the AGVs is 2. A computational time limit of 15 seconds is used for all approaches. Additionally, the MILP model is also solved for 3600 seconds to further validate the solutions of the hybrid ALNS. Table 7.5 shows the average result of ten runs for each instance and solution approach. A percentage gap in the table is calculated from the solution costs of two respective approaches. Negative numbers indicate that the solution costs of the hybrid ALNS are smaller (better) than those of the MILP or DP and vice versa.

The results show that the MILP and hybrid ALNS are able to solve the problems up to 3 requests to optimality. However, when increasing to 6 requests, the MILP can only find the optimal solution for 3 out of 10 instances, and even at the 9 and 12-request scenarios, the MILP cannot obtain a feasible solution within 15s for most of the instances. This indicates that the computational limit of the MILP has been reached. The hybrid ALNS, on the other hand, is able to find feasible solutions for all instances which are similar to those of the MILP in 3600s with a small deviation (up to 3.8%) or even better with up to 25.1% of improvement. Hence, the hybrid ALNS, in general, outperforms the MILP in both quality of solution and computational time. Additionally, the hybrid ALNS produces considerably better solutions than the DP with an improvement of 29.3% on average. A more extensive comparison against the DP for large-sized problems is presented in the following section.

Table 7.5: Comparison of the MILP model, dispatching model and the hybrid ALNS model.

Scenario		MILP 15 sec		MILP 3600 sec		DP	Hybrid ALNS		Gap		
TW-N-A-C	Inst-ance	Avg solution cost	Avg CPU time [s]	Avg solution cost	Avg CPU time [s]	Avg solution cost	Avg solution cost	Avg CPU time [s]	ALNS-MILP 15 [%]	ALNS-MILP 3600 [%]	ALNS-DP [%]
0.5-3-3-2	1	707.9*	0.2	707.9*	0.2	800.1	707.9	0.0	0.0	0.0	-11.5
	2	638.7*	0.1	638.7*	0.1	665.1	638.7	0.0	0.0	0.0	-4.0
	3	576.9*	0.8	576.9*	0.8	758.5	576.9	0.5	0.0	0.0	-23.9
	4	596.8*	0.7	596.8*	0.7	691.0	596.8	0.0	0.0	0.0	-13.6
	5	660.3*	0.4	660.3*	0.4	829.2	660.3	0.0	0.0	0.0	-20.4
	6	793.1*	0.9	793.1*	0.9	828.4	793.1	0.0	0.0	0.0	-4.3
	7	523.9*	3.0	523.9*	3.0	776.8	523.9	0.0	0.0	0.0	-32.6
	8	634.2*	0.3	634.2*	0.3	668.6	634.2	0.1	0.0	0.0	-5.1
	9	570.7*	0.7	570.7*	0.7	788.8	570.7	0.0	0.0	0.0	-27.7
	10	511.5*	0.8	511.5*	0.8	789.0	511.5	0.0	0.0	0.0	-35.2
0.5-6-3-2	1	1123.4°	15.0	1081.2*	176.5	1357.3	1081.2	0.1	-3.8	0.0	-20.3
	2	1389.1*	4.6	1389.1*	4.6	1557.0	1389.1	0.1	0.0	0.0	-10.8
	3	1106.4°	15.0	840.8*	58.1	1278.6	840.8	0.3	-24.0	0.0	-34.2
	4	1167.2°	15.0	1136.7	2176.6	1404.4	1147.2	2.1	-1.7	0.9	-18.3
	5	1328.4*	1.5	1328.4*	1.5	1712.3	1328.4	0.1	0.0	0.0	-22.4
	6	1115.4*	1.3	1115.4*	1.3	1440.5	1115.4	2.2	0.0	0.0	-22.6
	7	1202.4	11.2	1202.4*	11.6	1825.7	1210.9	0.2	0.7	0.7	-33.7
	8	1027.2	14.7	1020.8*	32.5	1666.9	1020.8	0.2	-0.6	0.0	-38.8
	9	942.0°	13.3	942.0*	25.2	1265.6	942.0	0.0	0.0	0.0	-25.6
	10	1068.5	11.6	1068.5*	13.2	1453.8	1068.5	0.0	0.0	0.0	-26.5
0.5-9-3-2	1	-	15.0	1364.4	3600.0	2213.7	1367.6	9.0	-	0.2	-38.2
	2	3403.7°	15.0	1414.7	3600.0	2060.6	1414.1	2.1	-58.5	0.0	-31.4
	3	-	15.0	1465.1°	3600.0	1908.9	1148.3	5.0	-	-21.6	-39.8
	4	1328.5	15.0	1297.0*	35.4	1863.8	1297.0	1.9	-2.4	0.0	-30.4
	5	-	15.0	1363.7	3600.0	2359.4	1373.4	4.1	-	0.7	-41.8
	6	5397.2°	15.0	1617.7	3600.0	1903.6	1634.5	0.0	-69.7	1.0	-14.1
	7	1762.7°	15.0	1170.1	3600.0	1681.2	1154.3	1.7	-34.5	-1.3	-31.3
	8	1714.1°	13.9	1633.8	3600.0	2345.4	1633.8	0.1	-4.7	0.0	-30.3
	9	-	15.0	1299.9	3600.0	1651.6	1349.0	1.1	-	3.8	-18.3
	10	5117.0°	15.0	1137.9	3600.0	2185.2	1137.9	0.1	-77.8	0.0	-47.9
0.5-12-3-2	1	-	15.0	-	3600.0	3354.4	1592.1	7.7	-	-	-52.5
	2	-	15.0	1882.8°	3600.0	5089.7	1882.8	0.1	-	0.0	-63.0
	3	-	15.0	-	3600.0	4078.6	2036.6	7.9	-	-	-50.1
	4	-	15.0	1649.0°	3600.0	2157.3	1569.9	3.7	-	-4.8	-27.2
	5	-	15.0	-	3600.0	2407.6	1506.3	4.8	-	-	-37.4
	6	-	15.0	-	3600.0	2750.1	1830.3	6.1	-	-	-33.4
	7	-	15.0	-	3600.0	2677.0	1708.5	1.5	-	-	-36.2
	8	-	15.0	2730.2°	3600.0	3638.6	2045.2	5.2	-	-25.1	-43.8
	9	-	15.0	-	3600.0	2278.3	1555.9	8.4	-	-	-31.7
	10	-	15.0	1577.1°	3600.0	2672.8	1577.0	3.9	-	0.0	-41.0

(*) all iterations found the optimal solution, (°) not all iterations found a solution for the given time limit

7.5 Numerical experiments for large-sized problems

Section 7.4 showed that the MILP model did not scale up to larger-sized problems. Hence, this section further studies the performance of the hybrid ALNS and the DP only by performing a sensitivity analysis on large-sized scenarios. Section 7.5.1 analyses the impact of capacity, time tightness, fleet size and the number of requests, and Section 7.5.2 investigates the effect of the computational time limit on the performance of the hybrid ALNS model.

7.5.1 Impact of capacity, time window tightness, fleet size and number of requests

Instances are generated in which N is varied in the range [20,40,60,80,100], A in [3,6,9], TW in [0.2,0.5,0.8] and C in [1,2,4,8]. For each scenario, ten instances are generated and each instance is run ten times. The detailed results of both methods for the different scenarios are shown in Table 7.6.

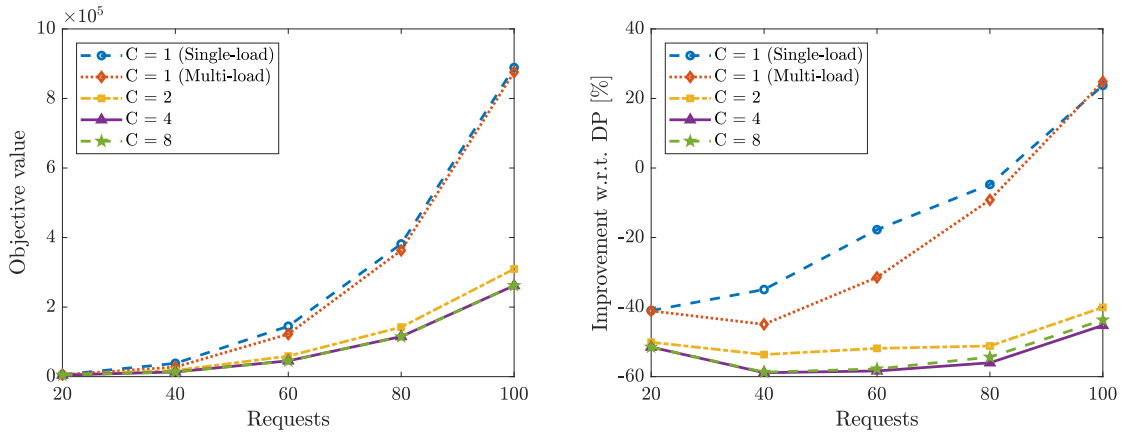
Table 7.6: Results of the MH, the proposed hybrid ALNS and the DP.

Scenario	**MH**	**Hybrid ALNS**				**DP**				**Improvement hybrid ALNS w.r.t. DP**					**Improvement hybrid ALNS w.r.t. MH**			
TW-N-A	C = 1 (Single)	C = 1 (Multi)	C = 2	C = 4	C = 8	C = 1	C = 2	C = 4	C = 8	C = 1 (Single)	C = 1 (Multi)	C = 2	C = 4	C = 8	C = 1 (Multi)	C = 2	C = 4	C = 8
0.2-20-3	6530.1	6357.7	4952.8	4430.7	4448.4	18233.5	15514.6	14241.0	13864.6	-64.2	-65.1	-68.1	-68.9	-67.9	-2.6	-24.2	-32.1	-31.9
0.2-20-6	5309.5	5374.0	4594.8	4335.3	4307.0	7379.8	7411.2	7384.4	7384.4	-28.1	-27.2	-38.0	-41.3	-41.7	1.2	-13.5	-18.3	-18.9
0.2-20-9	5111.6	5219.9	4407.8	4201.4	4195.2	7864.0	7866.3	7866.3	7866.3	-35.0	-33.6	-44.0	-46.6	-46.7	2.1	-13.8	-17.8	-17.9
0.2-40-3	78579.3	58185.0	33614.9	19168.7	19024.2	129572.5	101294.7	98784.3	96425.1	-39.4	-55.1	-66.8	-80.6	-80.3	-26.0	-57.2	-75.6	-75.8
0.2-40-6	12441.2	11881.2	8852.5	7889.9	7908.9	24323.4	19218.1	18593.8	18593.8	-48.9	-51.2	-53.9	-57.6	-57.5	-4.5	-28.8	-36.6	-36.4
0.2-40-9	10379.1	10570.4	8467.1	7771.7	7769.6	14742.4	13911.9	13861.8	13834.9	-29.6	-28.3	-39.1	-43.9	-43.8	1.8	-18.4	-25.1	-25.1
0.2-60-3	326838.5	267719.0	126846.3	95447.7	95211.6	353411.9	262394.7	244168.5	237429.0	-7.5	-24.2	-51.7	-60.9	-59.9	-18.1	-61.2	-70.8	-70.9
0.2-60-6	55700.8	45092.5	22029.2	15179.6	14819.5	67036.6	49237.8	42382.4	41132.1	-16.9	-32.7	-55.3	-64.2	-64.0	-19.0	-60.5	-72.7	-73.4
0.2-60-9	18048.2	17298.6	12780.1	11172.1	11054.8	23844.2	20551.3	19859.2	19505.8	-24.3	-27.5	-37.8	-43.7	-43.3	-4.2	-29.2	-38.1	-38.7
0.2-80-3	800640.7	766453.3	304424.7	229332.4	227024.3	723732.1	589093.6	500243.9	492912.3	10.6	5.9	-48.3	-54.2	-53.9	-4.3	-62.0	-71.4	-71.6
0.2-80-6	200862.4	196207.9	69330.9	54438.2	54029.1	214515.2	148336.1	129201.7	122914.9	-6.4	-8.5	-53.3	-57.9	-56.0	-2.3	-65.5	-72.9	-73.1
0.2-80-9	47121.0	48918.6	20942.4	18329.8	18410.1	67790.9	46030.8	43143.8	41454.8	-30.5	-27.8	-54.5	-57.5	-55.6	3.8	-55.6	-61.1	-60.9
0.2-100-3	1752723.3	1698406.4	592335.6	481973.3	482268.2	1354970.8	1114218.5	1042369.3	952666.0	29.4	25.3	-46.8	-53.8	-49.4	-3.1	-66.2	-72.5	-72.5
0.2-100-6	554933.7	536205.4	201818.2	160790.3	160990.8	467570.7	329287.9	266442.9	265947.8	18.7	14.7	-38.7	-39.7	-39.5	-3.4	-63.6	-71.0	-71.0
0.2-100-9	190662.8	198351.9	64413.3	50074.6	50074.6	174679.8	106861.3	88565.1	90574.3	9.1	13.6	-39.7	-43.5	-44.7	4.0	-66.2	-73.7	-73.7
0.5-20-3	6149.9	6231.3	4806.1	4566.0	4582.3	13301.5	13851.7	13512.1	13512.1	-53.8	-53.2	-65.3	-66.2	-66.1	1.3	-21.9	-25.8	-25.5
0.5-20-6	5011.2	5127.5	4362.8	4243.8	4226.5	7431.9	7303.6	7303.6	7303.6	-32.6	-31.0	-40.3	-41.9	-42.1	2.3	-12.9	-15.3	-15.7
0.5-20-9	4840.7	4893.9	4232.5	4102.2	4104.7	8049.2	7876.2	7876.2	7876.2	-39.9	-39.2	-46.3	-47.9	-47.9	1.1	-12.6	-15.3	-15.2
0.5-40-3	86447.0	54182.6	34163.4	23114.3	23431.8	128832.3	98517.8	90908.6	89087.0	-32.9	-57.9	-65.3	-74.6	-73.7	-37.3	-60.5	-73.3	-72.9
0.5-40-6	12516.7	11168.0	8573.2	8171.6	8187.4	22763.6	18816.3	18222.2	18222.2	-45.0	-50.9	-54.4	-55.2	-55.1	-10.8	-31.5	-34.7	-34.6
0.5-40-9	9512.3	9634.5	8136.7	7759.0	7715.0	14237.6	13202.5	13209.5	13209.5	-33.2	-32.3	-38.4	-41.3	-41.6	1.3	-14.5	-18.4	-18.9
0.5-60-3	335804.3	289997.5	137407.2	99053.5	98171.3	378947.6	294383.3	274959.2	257417.2	-11.4	-23.5	-53.3	-64.0	-61.9	-13.6	-59.1	-70.5	-70.8
0.5-60-6	69627.9	47915.9	23935.1	17981.6	18011.8	95597.2	63634.2	59199.9	57823.9	-27.2	-49.9	-62.4	-69.6	-68.9	-31.2	-65.6	-74.2	-74.1
0.5-60-9	19235.8	17378.8	12858.3	11798.9	11879.6	25692.6	23731.1	22627.8	22717.0	-25.1	-32.4	-45.8	-47.9	-47.7	-9.7	-33.2	-38.7	-38.2
0.5-80-3	872119.5	830643.2	334457.6	275082.3	274980.5	815115.1	645700.3	591409.3	566331.2	7.0	1.9	-48.2	-53.5	-51.4	-4.8	-61.7	-68.5	-68.5
0.5-80-6	242697.5	238831.2	92752.7	73954.3	73312.9	254848.0	179816.0	153724.2	144402.8	-4.8	-6.3	-48.4	-51.9	-49.2	-1.6	-61.8	-69.5	-69.8
0.5-80-9	69248.1	60130.6	27338.2	22080.0	22115.2	86193.1	58071.8	55226.2	52769.7	-19.7	-30.2	-52.9	-60.0	-58.1	-13.2	-60.5	-68.1	-68.1
0.5-100-3	1856568.9	1826422.7	644759.1	556308.0	557334.1	1448811.8	1145716.7	1082710.0	1059732.6	28.1	26.1	-43.7	-48.6	-47.4	-1.6	-65.3	-70.0	-70.0
0.5-100-6	614282.4	636542.1	214342.2	194153.1	194339.2	508445.2	351779.0	342399.2	325006.1	20.8	25.2	-39.1	-43.3	-40.2	3.6	-65.1	-68.4	-68.4
0.5-100-9	226979.5	243894.2	88521.5	61796.9	62116.0	190184.8	130797.6	118557.5	111408.7	19.3	28.2	-32.3	-47.9	-44.2	7.5	-61.0	-72.8	-72.6
0.8-20-3	9129.3	8139.4	5453.4	5235.3	5186.8	17985.1	17263.9	16889.8	16889.8	-49.2	-54.7	-68.4	-69.0	-69.3	-10.8	-40.3	-42.7	-43.2
0.8-20-6	5323.8	5337.2	4634.6	4611.9	4591.3	7926.8	7597.8	7597.8	7597.8	-32.8	-32.7	-39.0	-39.3	-39.6	0.3	-12.9	-13.4	-13.8
0.8-20-9	5123.5	5178.9	4468.2	4422.6	4434.7	7754.2	7675.7	7675.7	7675.7	-33.9	-33.2	-41.8	-42.4	-42.2	1.1	-12.8	-13.7	-13.4
0.8-40-3	103797.9	67152.4	40586.8	30439.1	29874.1	154665.6	114274.9	112391.2	108504.5	-32.9	-56.6	-64.5	-72.9	-72.5	-35.3	-60.9	-70.7	-71.2
0.8-40-6	19998.2	15005.0	9943.3	9232.3	9105.1	27858.6	26214.3	25185.6	24863.6	-28.2	-46.1	-62.1	-63.3	-63.4	-25.0	-50.3	-53.8	-54.5
0.8-40-9	10697.0	10468.8	8539.5	8297.8	8274.7	14171.8	13833.8	13940.6	13940.6	-24.5	-26.1	-38.3	-40.5	-40.6	-2.1	-20.2	-22.4	-22.6
0.8-60-3	365424.9	328453.7	153022.6	121255.9	120790.0	419437.3	323965.2	286589.7	282920.7	-12.9	-21.7	-52.8	-57.7	-57.3	-10.1	-58.1	-66.8	-66.9
0.8-60-6	88030.3	66167.9	28318.4	23950.2	23756.1	97689.6	71073.0	67922.8	65896.3	-9.9	-32.3	-60.2	-64.7	-63.9	-24.8	-67.8	-72.8	-73.0
0.8-60-9	25899.2	20907.4	14449.6	13560.4	13421.9	34276.2	27606.3	28552.8	28617.8	-24.4	-39.0	-47.7	-52.5	-53.1	-19.3	-44.2	-47.6	-48.2
0.8-80-3	877053.8	816155.3	310130.4	260825.2	259987.9	766518.7	638597.9	582843.2	575428.9	14.4	6.5	-51.4	-55.2	-54.8	-6.9	-64.6	-70.3	-70.4
0.8-80-6	250641.6	241175.3	93181.7	78045.4	78372.0	241737.4	172557.6	163911.5	153971.7	3.7	-0.2	-46.0	-52.4	-49.1	-3.8	-62.8	-68.9	-68.7
0.8-80-9	74741.5	68834.2	29219.4	25672.0	25570.5	90116.7	68848.8	66805.9	65764.8	-17.1	-23.6	-57.6	-61.6	-61.1	-7.9	-60.9	-65.7	-65.8
0.8-100-3	1884950.3	1812617.6	646262.3	558959.8	559077.5	1467610.8	1152487.8	1091591.9	1044836.2	28.4	23.5	-43.9	-48.8	-46.5	-3.8	-65.7	-70.3	-70.3
0.8-100-6	663895.3	659712.6	245113.6	216850.5	216956.3	501979.3	378025.9	342441.7	337728.7	32.3	31.4	-35.2	-36.7	-35.8	-0.6	-63.1	-67.3	-67.3
0.8-100-9	254089.4	269586.2	89191.1	75607.3	75594.9	199147.1	151810.4	137857.0	139101.1	27.6	35.4	-41.2	-45.2	-45.7	6.1	-64.9	-70.2	-70.2
Average										**-14.9**	**-20.4**	**-49.4**	**-54.0**	**-53.2**	**-7.2**	**-47.2**	**-53.6**	**-53.7**

The scenarios' names for varying TW, N and A are given in the rows of the table, while the results are shown in the columns for different AGV capacities. The results in the column 'C = 1 (Single)' are obtained from the matheuristic (MH) for single-load AGV scheduling proposed by Singh (2019), while the column 'C = 1 (Multi)' represents the results of the proposed hybrid ALNS with capacity C equal to one. The table also shows the gap of the hybrid ALNS with respect to the DP and the MH for each scenario. Based on the results of Table 7.6, the following paragraphs analyze the impact of capacity, time window tightness and fleet size on the performance of the solution approaches.

Impact of capacity

The effect of varying capacity is illustrated in Figure 7.1. Figure 7.1a presents the objective values for different capacities, while Figure 7.1b summarizes the improvement over the DP for different numbers of requests and capacities (by taking the average of all scenarios corresponding to each specific number of requests and capacity).



(a) Objective value for different capacities.

(b) Improvement w.r.t. DP.

Figure 7.1: Performance of multi-load AGVs with varying capacity.

It is reported by Singh (2019) that the average improvement of the MH over a company's current practice was about 28.1% for the scenarios with 20, 40 and 60 requests as well as equal unit penalty costs. In this experiment, the results show a similar performance of the MH with an average improvement of 31.2 % over the DP for the scenarios of 20 to 60 requests with different request costs. With capacity equal to 1, the proposed hybrid ALNS improves upon the DP by 39.2 % on average for 20 to 60 requests. However, both the hybrid ALNS and the MH start performing less effective with more requests, and the DP even outperforms both methods at the scenarios of 100 requests (and C = 1). For higher capacities, the hybrid ALNS performs better than the DP, i.e., 49.4%, 54.0% and 53.2% improvement on average for C = 2, 4, 8, respectively. Moreover, Figure 7.2 provides the improvements of the hybrid ALNS against the MH for different capacities.

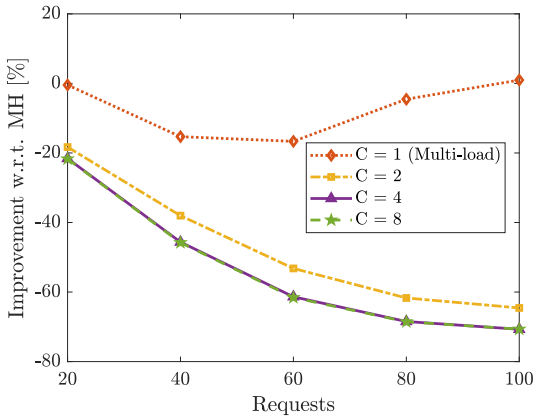
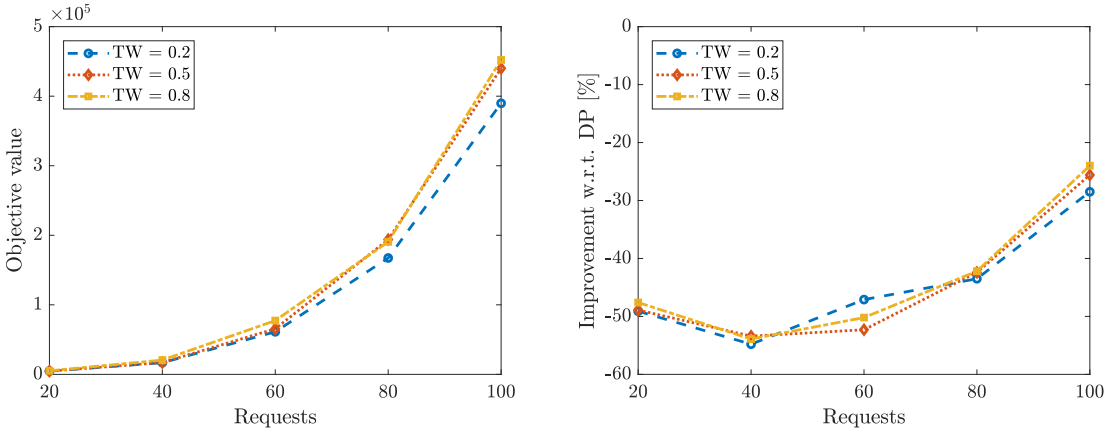


Figure 7.2: Improvement of multi-load hybrid ALNS w.r.t. single-load MH.

The hybrid ALNS with C = 1 outperforms the MH for the scenarios with 40 and 60 requests and performs similarly in the other scenarios. For higher capacities, the hybrid ALNS method perform significantly better than the single-load model. With C = 2, 4, 8, the average improvements are 47.2%, 53.6% and 53.7%, respectively. There is only a slight improvement by increasing the capacity from 4 to 8. Hence, it can be concluded that increasing the capacity will not bring further benefits at some point.

Impact of time window tightness

The effect of time window tightness with varying the number of requests is shown in Figure 7.3. Figure 7.3a presents the objective values of the hybrid ALNS and Figure 7.3b shows the performance of the method in comparison to the DP, for different degrees of time window tightness. It can be seen that requests with tighter time windows (e.g., TW = 0.8) result in an increase in the objective value since tardiness is more likely to occur in these instances. The results also show that there are no distinct difference in the improvement of the proposed hybrid over the DP when varying time window tightness.



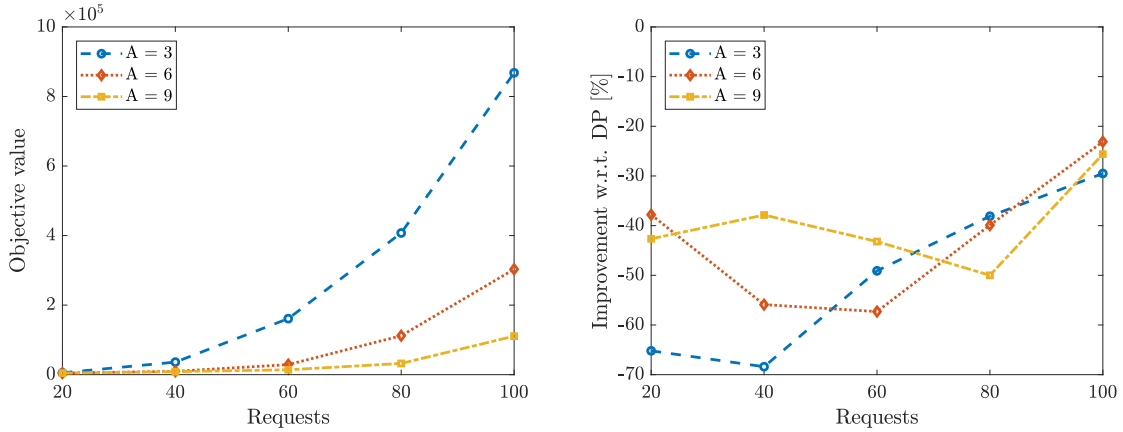
(a) Effect of TW on objective value.

(b) Improvement w.r.t. DP

Figure 7.3: Performance of the hybrid ALNS for different degrees of time window tightness.

Impact of fleet size

The performance of the hybrid ALNS for different fleet sizes is revealed in Figure 7.4. One can see the decrease of the objective value by increasing the fleet size from Figure 7.4a and the improvement with respect to the DP when varying this parameter value from Figure 7.4b.



(a) Effect of fleet size on objective value.

(b) Improvement w.r.t. DP.

Figure 7.4: Performance of the hybrid ALNS model for different fleet sizes.

This improvement depends much on the fleet size and therefore, on whether or not the system is overloaded. However, it seems that the best improvement against the DP can be achieved when using less AGVs for a small number of requests and using more AGVs, but up to a certain point, for larger number of requests.

The increased performance of multi-load AGVs over single-load AGVs introduces the possibility of reducing the fleet size. Figure 7.5 compares the case of 9 single-load AGVs to the cases of 6 and 3 multi-load AGVs for the scenarios with TW of 0.5 (from Table 7.6).

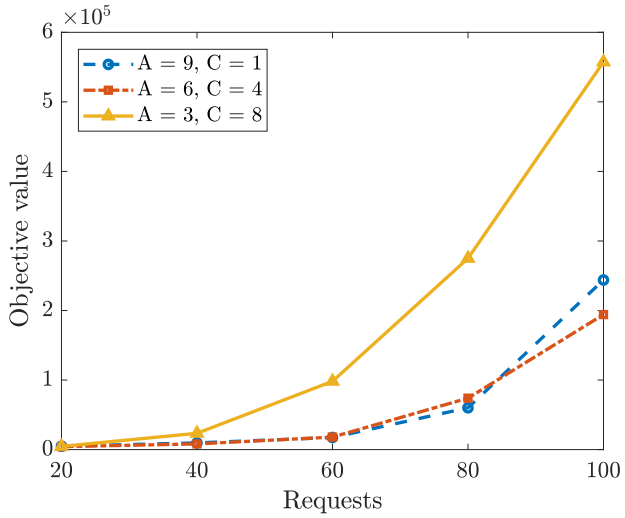


Figure 7.5: Comparison of nine single-load AGVs against six and three multi-load AGVs.

The results show that using 6 multi-load AGVs with capacity of 4 produces results similar to or better than those of 9 single-load AGVs. However, reducing the fleet size further (to 3 multi-load AGVs) even with more capacity is no longer beneficial. Hence, it can be concluded that the fleet size can be reduced to a certain extent by employing multi-load AGVs instead of single-load AGVs.

7.5.2 Impact of computational time

The effect of the computational time limit on the performance of the hybrid ALNS is analyzed by increasing the computational time to 30, 45, 60, 75 and 90 seconds. The experiment is conducted with a subset of the scenarios, where N is varied in the range [20,40,60,80], A in [3,9], C in [2,8], while TW is set to 0.5. For each scenario, ten instances are also generated and each instance is run ten times. The average objective values of each scenario obtained in 30 s, 45 s, 60 s and 75 s are compared to that found in 90 s. The results are shown in Table 7.7 and Figure 7.6.

Table 7.7: Average deviation with respect to the average solution found in 90 seconds.

Scenario	ALNS (90 s)	ALNS (15 s)	ALNS (30 s)	ALNS (45 s)	ALNS (60 s)	ALNS (75 s)
TW-N-A-C	**Avg**	**Avg**[%]	**Avg**[%]	**Avg**[%]	**Avg**[%]	**Avg**[%]
0.5-20-3-2	4835.51	4.08	2.70	2.02	1.08	0.70
0.5-20-9-2	4274.84	1.89	0.86	0.92	0.58	0.32
0.5-20-3-8	4504.25	5.32	3.90	1.62	1.34	0.28
0.5-20-9-8	4148.67	1.86	1.22	0.36	0.32	0.12
Average	**4440.82**	**3.29**	**2.17**	**1.23**	**0.83**	**0.36**
0.5-40-3-2	24413.34	36.57	12.88	10.04	7.03	2.26
0.5-40-9-2	8015.19	4.06	1.35	1.12	0.63	0.51
0.5-40-3-8	18665.93	40.60	21.89	10.80	2.97	1.77
0.5-40-9-8	7574.02	6.55	4.48	2.47	1.08	0.30
Average	**14667.12**	**21.95**	**10.15**	**6.11**	**2.93**	**1.21**
0.5-60-3-2	96247.83	44.25	25.53	16.47	10.17	2.12
0.5-60-9-2	11632.40	10.65	4.13	3.41	1.31	0.98
0.5-60-3-8	71805.82	60.47	33.15	19.97	11.16	3.79
0.5-60-9-8	10713.80	10.57	5.52	2.95	1.61	0.77
Average	**47599.96**	**31.48**	**17.08**	**10.70**	**6.06**	**1.92**
0.5-80-3-2	267242.28	23.75	12.03	9.62	7.46	4.29
0.5-80-9-2	20397.71	38.40	24.12	15.77	7.01	1.44
0.5-80-3-8	216721.79	38.31	24.82	13.81	7.43	4.11
0.5-80-9-8	16430.19	44.37	22.84	14.24	9.52	4.40
Average	**130197.99**	**36.21**	**20.95**	**13.36**	**7.85**	**3.56**

For scenarios with 20 requests, the average objective value after 15 s is only 3.29% higher than that after 90 s. When the number of requests increases, so does the deviation between the averages found after 15 s and 90 s. The average objective value after 15 s for scenarios with 80 requests is 36.21% higher than that found after 90 s. Nevertheless, after 75 s there is only a deviation of 3.56%. The gap deviation progressively decreases as the time limit increases. For scenarios with 80 requests, the gap is 15.26% (36.21-20.95%) when increasing from 15 s to 30 s, 7.59% (20.95-13.36%) from 30 s to 45 s seconds, 5.51% (13.36-7.85%) from 45 s to 60 s and 4.29% (7.85-3.56%) from 60 s to 75 s. The decrease in the average gap with respect to the average objective value after 90 s is summarized in Figure 7.6.

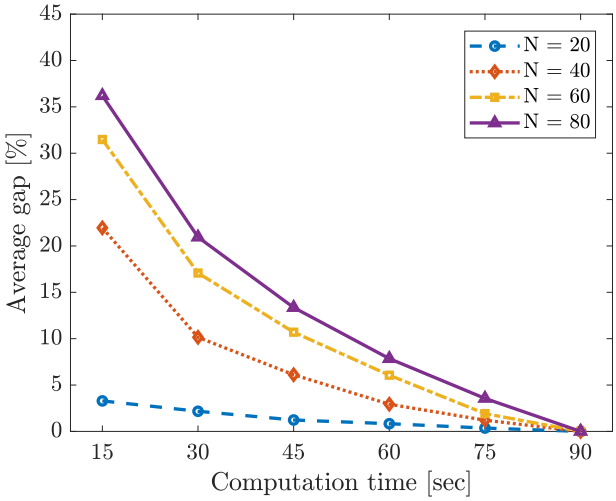


Figure 7.6: Average gap with respect to the average solution found in 90 s.

Figure 7.6 shows that when increasing the computational time, the objective value converges. For small scenarios the objective value converges more quickly than it does for larger ones. It can be concluded that for small scenarios the hybrid ALNS is able to achieve good solutions within 15 s and for large scenarios, the method can considerably improve the solution if more computational time is available. In practice, the maximum time limit to compute the schedule of AGVs in the BIC shop floor is about 60s. The results here also show that the proposed hybrid ALNS is, in general, able to obtain good solutions after a minute for different scenarios with a deviation of up to only 8%.

8 Conclusion and further research

In this thesis, a hybrid ALNS algorithm is proposed to solve the scheduling problem for multi-load AGVs. The work is inspired from the multi-tenant setting environment of the BIC. The main novelty of the problem lies in the concurrent consideration of characteristics including capacity, soft time windows, transport requests with different priorities, battery management considering a critical threshold and partial charging, and heterogeneous fleets with different AGVs' capabilities to carry out tasks. The objective is to minimize the weighted sum of the tardiness costs of requests and travel costs of AGVs.

An MILP model was formulated for the problem. However, the MILP is only applicable to small-sized problems (i.e., up to 12 requests). Therefore, an ALNS algorithm, which is intensified by a local search technique and includes several new destroy and repair operators, is proposed to find good quality solutions in the offline scheduling mode. The computational experiments on industry case studies reveal that the proposed hybrid method outperforms the MILP and a dispatching model (current practice).

To fully evaluate the effectiveness of the hybrid ALNS, sensitivity analyses were performed on different parameters such as capacity, time window tightness, fleet size and computational time. On average, the analyses indicate a 20-50% improvement over the dispatching model by using the proposed method. The improved performance results in a higher utilization of AGVs and introduces the possibility of reducing the fleet size. The results also show that the hybrid ALNS can achieve good solutions within the maximum computational time in practice, i.e., 60 s, with a small deviation of 8% compared to longer computing.

The hybrid ALNS is also compared to the single-load model proposed by Singh (2019) by using a capacity equal to one. The hybrid ALNS outperforms the single-load model by 7.2% on average for scenarios with a range of 20 to 100 requests. This improvement is achieved as a result of a more sophisticated initial solution constructive heuristic, faster destroy and repair methods and a local search procedure, which are proposed in the ALNS framework. Additionally, the proposed hybrid ALNS with AGVs of higher capacities can reduce 47-54% of the solution cost on average with respect to the single-load model.

An interesting direction for future research is the extension of the proposed solution method to parallel computing that could improve the speed of the model by considering multiple solutions simultaneously. Research about running an ALNS algorithm in parallel on the GPU of a computer has been performed by Bach et al. (2019) with promising results.

In order to apply the proposed hybrid ALNS model in practice, it must be converted into a dynamic model with a rolling horizon. Currently, the hybrid ALNS solves the static problem of scheduling multi-load AGVs in which all requests are known in advance, no requests are added and transportation times are considered to be deterministic. Scheduling AGVs in real-time requires timely responses to changes due to dynamic events including new requests, old requests being modified or canceled and vehicle breakdowns.

List of symbols

Requests and charging	
R	set of transport requests
P	set of pickup tasks
D	set of delivery tasks
B	set of charging tasks
T	set of all transport requests R and charging tasks B
n_R	number of transport requests
A_r^R	set of capability requirements for request r
d_r^R	destination node of request r
c_r^R	penalty cost incurred per time unit of tardiness for request r
e_r^R	earliest pickup time of request r
l_r^R	latest delivery time of request r
g_r^R	type of task r (pickup, delivery or charging)
q_r^R	load of request r (positive for pickup, negative for delivery)
Automated guided vehicles	
V	set of automated guided vehicles
C^k	Capacity of AGV k
A_k^V	set of capabilities of AGV k
c_k^V	travel cost per time unit of AGV k
cr_k^V	charging rate of AGV k
dr_k^V	discharging rate of AGV k
bc_k^V	critical battery threshold of AGV k
bn_k^V	non-critical battery threshold of AGV k
b_l	minimum battery charge level
b_u	maximum battery charge level
Shop floor layout	
N	set of all nodes in the layout
h_i^N	service time at node i
d_{ij}	distance between node i and node j
t_{ijk}	travel time between node i and node j for AGV k
Mixed-Integer Linear Programming (MILP) parameters	
$x_{rr'k}$	binary variable, equal to 1 if AGV k performs request r prior to r', 0 otherwise
y_{rk}	arrival time of AGV k at the destination of request r
w_{rk}	number of loads on AGV k after performing request r
z_{rk}	amount of discharge of AGV k when it reaches the destination of request r
τ_{rk}	tardiness of request r transported by AGV k
δ_{rk}	travel time of AGV k, due to transporting request r
M	arbitrary large number

Hybrid Adaptive Large Neighborhood Search (ALNS)	
Q^k	sequence of tasks of AGV k
Q	set of task sequences of AGVs $k \in V$, where $Q = \{Q^1, Q^2, .., Q^k\}$
S^k	schedule of tasks of AGV k
S	set of task schedules of AGVs $k \in V$, where $S = \{S^1, S^2, .., S^k\}$
V_r	set of AGVs that have the required capability to service request r
α	weight factor
θ	Temperature of SA acceptance criterion
ζ	number of iteration without improving best solution
υ	number of iteration local search without improving current solution
ξ	number of iterations station destroy/repair without improving current solution
q	number of tasks to be removed by a request destroy operator
η	degree of randomness in choosing related task in the Shaw destroy operator
λ	reaction factor for the adaptions of the weights of the operators
ω	initial temperature control parameter of simulated annealing
ϵ	cooling rate of simulated annealing
ϕ_1	weight for relatedness in terms of distance for the Shaw destroy operators
ϕ_2	weight for relatedness in terms of time window for the Shaw destroy operators
ϕ_3	weight for relatedness in terms of capability for the Shaw destroy operators
ψ_1	reward for the score of an operator when finding a new best solution
ψ_2	reward for the score of an operator when finding a better solution
ψ_3	reward for the score of an operator when finding a worse solution, but accepted by SA
n_{LS}	Number of consecutive unfruitful iterations before stopping local search
n_S	Number of consecutive unfruitful iterations before stopping station destroy/repair
n_D	Number of consecutive unfruitful iterations before applying diversification method
Schedule generation scheme	
y^S_{rk}	start time of request r
y^D_{rk}	end time of requests r
Y_k	calculation variable for the start and end time of requests
β_{rk}	battery level at the start of request r
$\mathcal{B}_k$	calculation variable for the current battery level
$\mathcal{B}_R$	required charge to service the next tasks until the following charging request
$\mathcal{B}_P$	possible amount to charge until the earliest pickup time of the next request

References

Advanced Manufacturing Logistics. (2020). *Hotspot for smart logistic in the manufacturing industry.* https://advancedmanufacturinglogistics.com. (Accessed 26 March 2020.)

Angra, S., Chanda, A., and Chawla, V. (2018). Comparison and evaluation of job selection dispatching rules for integrated scheduling of multi-load automatic guided vehicles serving in variable sized flexible manufacturing system layouts: A simulation study. *Management Science Letters*, *8*(4), 187–200.

Azimi, P., Haleh, H., and Alidoost, M. (2010). The selection of the best control rule for a multiple-load agv system using simulation and fuzzy madm in a flexible manufacturing system. *Modelling and Simulation in Engineering*, *2010*.

Bach, L., Hasle, G., and Schulz, C. (2019). Adaptive large neighborhood search on the graphics processing unit. *European Journal of Operational Research*, *275*(1), 53–66.

Baruwa, O. T., and Piera, M. A. (2016). A coloured petri net-based hybrid heuristic search approach to simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Research*, *54*(16), 4773–4792.

Baugh Jr, J. W., Kakivaya, G. K. R., and Stone, J. R. (1998). Intractability of the dial-a-ride problem and a multiobjective solution using simulated annealing. *Engineering Optimization*, *30*(2), 91–123.

Belhaiza, S. (2019). A hybrid adaptive large neighborhood heuristic for a real-life dial-a-ride problem. *Algorithms*, *12*(2), 39.

Bilge, U., and Tanchoco, J. M. (1997). Agv systems with multi-load carriers: basic issues and potential benefits. *Journal of Manufacturing Systems*, *16*(3), 159.

Bongiovanni, C., Kaspi, M., and Geroliminis, N. (2019). The electric autonomous dial-a-ride problem. *Transportation Research Part B: Methodological*, *122*, 436–456.

Braekers, K., Caris, A., and Janssens, G. K. (2014). Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transportation Research Part B: Methodological*, *67*, 166–186.

Brainport Industries Campus. (2020). *The evolution of the high-tech manufacturing industry.* https://www.brainportindustriescampus.com/en/innovating/factory-of-the-future. (Accessed 26 March 2020)

Chawla, V., Chanda, A., and Angra, S. (2018). Multi-load agvs scheduling by application of modified memetic particle swarm optimization algorithm. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, *40*(9), 436.

Chen, C., Xia, B., Zhou, B.-h., and Xi, L. (2015). A reinforcement learning based approach for a multiple-load carrier scheduling problem. *Journal of Intelligent Manufacturing*, *26*(6), 1233–1245.

Cordeau, J.-F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, *54*(3), 573–586.

Demir, E., Bektaş, T., and Laporte, G. (2012). An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, *223*(2), 346–359.

Desaulniers, G., Errico, F., Irnich, S., and Schneider, M. (2016). Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, *64*(6), 1388–1405.

Detti, P., Papalini, F., and de Lara, G. Z. M. (2017). A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare. *Omega*, *70*, 1–14.

Egbelu, P. J., and Tanchoco, J. M. (1984). Characterization of automatic guided vehicle dispatching rules. *The International Journal of Production Research*, *22*(3), 359–374.

Felipe, Á., Ortuño, M. T., Righini, G., and Tirado, G. (2014). A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. *Transportation Research Part E: Logistics and Transportation Review*, *71*, 111–128.

Gnanavel Babu, A., Jerald, J., Noorul Haq, A., Muthu Luxmi, V., and Vigneswaralu, T. (2010). Scheduling of machines and automated guided vehicles in fms using differential evolution. *International Journal of Production Research*, *48*(16), 4683–4699.

Goeke, D., and Schneider, M. (2015). Routing a mixed fleet of electric and conventional vehicles. *European Journal of Operational Research*, *245*(1), 81–99.

Gschwind, T., and Drexl, M. (2019). Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. *Transportation Science*, *53*(2), 480–491.

Häll, C. H., and Peterson, A. (2013). Improving paratransit scheduling using ruin and recreate methods. *Transportation planning and technology*, *36*(4), 377–393.

Ho, S. C., Szeto, W., Kuo, Y.-H., Leung, J. M., Petering, M., and Tou, T. W. (2018). A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological*, *111*, 395–421.

Ho, Y.-C., and Chien, S.-H. (2006). A simulation study on the performance of task-determination rules and delivery-dispatching rules for multiple-load agvs. *International journal of production research*, *44*(20), 4193–4222.

Ho, Y.-C., and Liu, H.-C. (2006). A simulation study on the performance of pickup-dispatching rules for multiple-load agvs. *Computers and Industrial Engineering*, *51*(3), 445–463.

Ho, Y.-C., and Liu, H.-C. (2009). The performance of load-selection rules and pickup-dispatching rules for multiple-load agvs. *Journal of Manufacturing Systems*, *28*(1), 1–10.

Ho, Y.-C., Liu, H.-C., and Yih, Y. (2012). A multiple-attribute method for concurrently solving the pickup-dispatching problem and the load-selection problem of multiple-load agvs. *Journal of manufacturing systems*, *31*(3), 288–300.

Hosny, M. I., and Mumford, C. L. (2012). Constructing initial solutions for the multiple vehicle pickup and delivery problem with time windows. *Journal of King Saud University-Computer and Information Sciences*, *24*(1), 59–69.

Huo, K., Zhang, Y., Hu, Z., et al. (2016). Research on scheduling problem of multi-load agv at automated container terminal. *Journal of Dalian University of Technology*, *56*(3), 244–251.

Jerald, J., Asokan, P., Saravanan, R., and Rani, A. D. C. (2006). Simultaneous scheduling of parts and automated guided vehicles in an fms environment using adaptive genetic algorithm. *The International Journal of Advanced Manufacturing Technology*, *29*(5-6), 584–589.

Jorgensen, R. M., Larsen, J., and Bergvinsdottir, K. B. (2007). Solving the dial-a-ride problem using genetic algorithms. *Journal of the operational research society*, *58*(10), 1321–1331.

Keskin, M., and Çatay, B. (2016). Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, *65*, 111–127.

Lambrechts, O., Demeulemeester, E., and Herroelen, W. (2008). A tabu search procedure for developing robust predictive project schedules. *International Journal of Production Economics*, *111*(2), 493–508.

Le-Anh, T., and De Koster, M. (2006). A review of design and control of automated guided vehicle systems. *European Journal of Operational Research*, *171*(1), 1–23.

Lee, J., and Srisawat, T. (2006). Effect of manufacturing system constructs on pick-up and drop-off strategies of multiple-load agvs. *International journal of production research*, *44*(4), 653–673.

Li, B., Krushinsky, D., Van Woensel, T., and Reijers, H. A. (2016). An adaptive large neighborhood search heuristic for the share-a-ride problem. *Computers and Operations Research*, *66*, 170–180.

Li, G., Li, X., Gao, L., and Zeng, B. (2019). Tasks assigning and sequencing of multiple agvs based on an improved harmony search algorithm. *Journal of Ambient Intelligence and Humanized Computing*, *10*(11), 4533–4546.

Li, M. P., and Kuhl, M. E. (2017). Design and simulation analysis of pder: A multiple-load automated guided vehicle dispatching algorithm. In *2017 winter simulation conference (wsc)* (pp. 3311–3322).

Li, Y., Chen, H., and Prins, C. (2016). Adaptive large neighborhood search for the pickup and delivery problem with time windows, profits, and reserved requests. *European Journal of Operational Research*, *252*(1), 27–38.

Ma, X., Bian, Y., and Gao, F. (2020). An improved shuffled frog leaping algorithm for multiload agv dispatching in automated container terminals. *Mathematical Problems in Engineering*, *2020*.

Masmoudi, M. A., Hosny, M., Braekers, K., and Dammak, A. (2016). Three effective meta-heuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. *Transportation Research Part E: Logistics and Transportation Review*, *96*, 60–80.

Masmoudi, M. A., Hosny, M., Demir, E., Genikomsakis, K. N., and Cheikhrouhou, N. (2018). The dial-a-ride problem with electric vehicles and battery swapping stations. *Transportation research part E: logistics and transportation review*, *118*, 392–420.

Mauri, G., and Lorena, L. N. (2006). A multiobjective model and simulated annealing approach for a dial-a-ride problem. In *Workshop dos cursos de computação.*

Molenbruch, Y., Braekers, K., and Caris, A. (2017). Typology and literature review for dial-a-ride problems. *Annals of Operations Research*, *259*(1-2), 295–325.

Muelas, S., LaTorre, A., and Pena, J.-M. (2015). A distributed vns algorithm for optimizing dial-a-ride problems in large-scale scenarios. *Transportation Research Part C: Emerging Technologies*, *54*, 110–130.

Paquette, J., Cordeau, J.-F., Laporte, G., and Pascoal, M. M. (2013). Combining multicriteria analysis and tabu search for dial-a-ride problems. *Transportation Research Part B: Methodological*, *52*, 1–16.

Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2010). Variable neighborhood search for the dial-a-ride problem. *Computers and Operations Research*, *37*(6), 1129–1138.

Rahimikelarijani, B., Saidi-Mehrabad, M., and Barzinpour, F. (2020). A mathematical model for multiple-load agvs in tandem layout. *Journal of Optimization in Industrial Engineering*, *13*(1), 67–80.

Ropke, S., and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, *40*(4), 455–472.

Santos, V. G. M., and de Carvalho, M. A. M. (2019). Tailored heuristics in adaptive large neighborhood search applied to the cutwidth minimization problem. *European Journal of Operational Research*.

Schiffer, M., and Walther, G. (2017). The electric location routing problem with time windows and partial recharging. *European Journal of Operational Research*, *260*(3), 995–1013.

Schneider, M., Stenger, A., and Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, *48*(4), 500–520.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., and Dueck, G. (2000). Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, *159*(2), 139–171.

Shaw, P. (1997). A new local search algorithm providing high quality solutions to vehicle routing problems. *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*.

Singh, N. (2019). A matheuristic for scheduling automated guided vehicles in a multi-tenant setting. *Master thesis. University of Eindhoven, Eindhoven, Netherlands*. https://research.tue.nl/nl/studentTheses/a-matheuristic-for-scheduling-automated-guided-vehicles-in-a-mult. (Accessed 27 November 2019)

Ulusoy, G., Sivrikaya-Şerifoğlu, F., and Bilge, Ü. (1997). A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles. *Computers and Operations Research*, *24*(4), 335–351.

Yang, J., and Sun, H. (2015). Battery swap station location-routing problem with capacitated electric vehicles. *Computers and Operations Research*, *55*, 217–232.

Zhao, M., and Lu, Y. (2019). A heuristic approach for a real-world electric vehicle routing problem. *Algorithms*, *12*(2), 45.

A Appendix

A.1 Layout of case study

The layout of the fieldlab at the BIC is shown in Figure A.1.

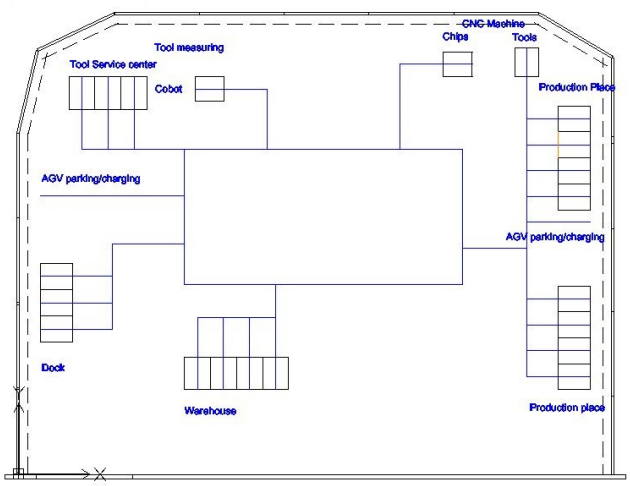


Figure A.1: Layout of the case study.

A.2 Generation of instances

A random number in the interval (a, b) is defined by $y^{(a,b)}$ and a random integer in the same interval by $y_{int}^{(a,b)}$. A random element from a list is denotes by $y^{listname}$. Let s_r^R denote the location of the pickup and d_r^R define the delivery location of a request. The pseudocode is shown in Algorithm 5.

Algorithm 5 Instance creation process

Require: tightness of time window TW; number of requests N; number of AGVs A; capacity of AGVs C; set of possible request Costs c^R; possible request capabilities A^R; maximum distance in layout d_{max}; minimum speed AGV v_{min}; set of pickup/delivery nodes N_{nodes}; Scheduling time horizon h.

repeat
  $R \leftarrow \emptyset$
  if $y^{(0,1)} < TW$ **then**
    $e_r^R \leftarrow y_{int}^{(0,h)}$
    $l_r^R \leftarrow e_r^R + (d_{max}/v_{min}) \cdot \left(0.5 + 0.5 \cdot y^{(0,1)}\right)$
  else
    $e_r^R \leftarrow y_{int}^{(0,h)}$
    $l_r^R \leftarrow e_r^R + (d_{max}/v_{min}) \cdot \left(1 + 0.5 \cdot y^{(0,1)}\right)$
  end if
  $A_r^R \leftarrow y^{c^R}$
  $s_r^R \leftarrow y_{int}^{(0,N_{nodes})}$
  $d_r^R \leftarrow y_{int}^{(0,N_{nodes} \setminus s_r^R)}$
  Append r to R
until N requests created

A.3 Parameter tuning

The results of the parameter tuning are given in Table A.2. The parameters are tuned in the sequence from top to bottom. Five different scenarios are used for the parameter tuning with two instances per scenario. The scenarios are given in Table A.1. Ten different parameter values are considered and for each value the model is run ten times. For each parameter value the average percentage deviation $'Dev\%'$ from the average of the best achieved solution is calculated. The parameter value with the least average percent deviation is selected and the initial value is updated. This procedure is repeated until all parameter have been tuned. The best parameter values are indicated in bold.

Table A.1: Scenarios for parameter tuning.

Parameter tuning instances
0.2-20-3-4
0.5-40-3-4
0.5-60-6-4
0.8-40-6-4
0.8-60-3-4

Table A.2: Parameter tuning.

Parameter		Initial value	Values tested									
n_{LS}	Value	30	5	10	15	20	**25**	30	35	40	45	50
	Dev%		0,57	0,64	0,53	0,74	**0,52**	0,58	0,55	0,54	0,58	0,56
n_S	Value	9	1	3	5	**7**	9	11	13	15	17	19
	Dev%		0,74	0,6	0,47	**0,44**	0,54	0,52	0,58	0,66	0,84	0,69
n_D	Value	45	20	25	30	35	40	45	50	55	**60**	65
	Dev%		0,75	0,55	0,62	0,59	0,66	0,5	0,52	0,48	**0,46**	0,62
ψ_1	Value	0,85	0,5	0,55	0,6	0,65	0,7	0,75	**0,8**	0,85	0,9	0,95
	Dev%		0,73	0,52	0,68	0,55	0,54	0,55	**0,45**	0,66	0,5	0,66
ψ_2	Value	0,6	0,5	0,55	0,6	**0,65**	0,7	0,75	0,8	0,85	0,9	0,95
	Dev%		0,51	0,58	0,64	**0,49**	0,7	0,53	0,6	0,61	0,56	0,55
ψ_3	Value	0,65	**0,5**	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95
	Dev%		**0,5**	0,55	0,52	0,53	0,6	0,59	0,54	0,57	0,62	0,62
λ	Value	0,6	0,09	0,19	0,29	0,39	0,49	0,59	**0,69**	0,79	0,89	0,99
	Dev%		0,51	0,71	0,52	0,54	0,54	0,69	**0,48**	0,59	0,52	0,51
η	Value	6	2	3	4	5	6	7	8	**9**	10	11
	Dev%		0,52	0,7	0,67	0,56	0,57	0,54	0,61	**0,51**	0,53	0,55
ϕ_1	Value	2,7	1,5	1,7	1,9	2,1	2,3	**2,5**	2,7	2,9	3,1	3,3
	Dev%		0,62	0,62	0,65	0,56	0,45	**0,44**	0,54	0,66	0,61	0,63
ϕ_2	Value	2,4	1	**1,2**	1,4	1,6	1,8	2	2,2	2,4	2,6	2,8
	Dev%		0,61	**0,47**	0,61	0,49	0,65	0,56	0,65	0,55	0,59	0,58
ϕ_3	Value	1,6	1	1,2	1,4	1,6	1,8	**2**	2,2	2,4	2,6	2,8
	Dev%		0,54	0,65	0,6	0,66	0,55	**0,45**	0,56	0,56	0,77	0,61
ω	Value	0,01	0,001	0,003	**0,005**	0,008	0,009	0,01	0,02	0,03	0,04	0,05
	Dev%		0,52	0,59	**0,46**	0,59	0,68	0,56	0,55	0,54	0,74	0,66
ϵ	Value	0,997	0,9	0,95	0,97	0,98	0,99	**0,995**	0,996	0,997	0,998	0,999
	Dev%		0,58	0,72	0,47	0,51	0,53	**0,44**	0,59	0,64	0,6	0,83

A.4 AGV data

The user defined parameters, including capability, speed, initial battery level, discharge rate, charge rate, travel cost and critical battery threshold, are given in Table A.3.

Table A.3: AGV user defined parameters.

AGV id.	Capability	Speed	Initial charge	Discharge rate	Charging rate	Travel cost	Critical battery threshold
0	A,B,C	1.0	40	0.01	0.02	2	30
1	A,C,D	1.5	40	0.01	0.02	1	30
2	E	1.0	40	0.01	0.02	3	30
3	A,B,C	1.0	40	0.01	0.02	2	30
4	A,C,D	1.5	40	0.01	0.02	1	30
5	E	1.0	40	0.01	0.02	3	30
6	A,B,C	1.0	40	0.01	0.02	2	30
7	A,C,D	1.5	40	0.01	0.02	1	30
8	E	1.0	40	0.01	0.02	3	30

TU/e Technische Universiteit Eindhoven University of Technology

Declaration concerning the TU/e Code of Scientific Conduct for the Master's thesis

I have read the TU/e Code of Scientific Conduct[i].

I hereby declare that my Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date

11-5-2020

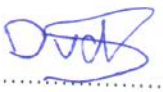
Name

D.T.J. van de Sande

ID-number

0862030

Signature



Submit the signed declaration to the student administration of your department.

[i] See: http://www.tue.nl/en/university/about-the-university/integrity/scientific-integrity/
The Netherlands Code of Conduct for Academic Practice of the VSNU can be found here also.
More information about scientific integrity is published on the websites of TU/e and VSNU