Transportation Research Part C 33 (2013) 1–21

Contents lists available at SciVerse ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Scheduling of Intelligent and Autonomous Vehicles under pairing/unpairing collaboration strategy in container terminals

Shahin Gelareh [a,b,c,d,*], Rochdi Merzouki [b,d], Kay McGinley [e], Roisin Murray [e]

[a] UArtois, LGI2A, F-62400 Béthune, France
[b] LAGIS UMR CNRS 8219, Ecole Polytechnique de Lille, Avenue Paul Langevin, F-59655 Villeneuve d'Ascq, France
[c] Univ Lille Nord de France, F-59000 Lille, France
[d] Ecole Polytechnique de Lille, Avenue Paul Langevin, 59655 Villeneuve d'Ascq, France
[e] Department of Transport Engineering, Dublin Institute of Technology, Bolton St., Dublin 1, Ireland

ARTICLE INFO

Article history:
Received 24 October 2011
Received in revised form 12 March 2013
Accepted 8 April 2013

Keywords:
Intelligent Autonomous Vehicle
Automated guided vehicle
Mixed integer programming
Scheduling
Lagrangian relaxation
Discrete-event simulation

ABSTRACT

A new class of Intelligent and Autonomous Vehicles (IAVs) has been designed in the framework of Intelligent Transportation for Dynamic Environment (InTraDE) project funded by European Union. This type of vehicles is technologically superior to the existing Automated Guided Vehicles (AGVs), in many respects. They offer more flexibility and intelligence in maneuvering within confined spaces where the logistic operations take place. This includes the ability of pairing/unpairing enabling a pair of 1-TEU (20-foot Equivalent Unit) IAVs dynamically to join, transport containers of any size between 1-TEU and 1-FFE (40-foot Equivalent) and disjoin again. Deploying IAVs helps port operators to remain *efficient* in coping with the ever increasing volume of container traffic at ports and eliminate the need for deploying more 40-ft transporters in the very confined area of ports. In order to accommodate this new feature of IAVs, we review and extend one of the existing mixed integer programming models of AGV scheduling in order to minimize the makespan of operations for transporting a set of containers of different sizes between quay cranes and yard cranes. In particular, we study the case of Dublin Ferryport Terminal. In order to deal with the complexity of the scheduling model, we develop a Lagrangian relaxation-based decomposition approach equipped with a variable fixing procedure and a primal heuristics to obtain high-quality solution of instances of the problem.

1. Introduction

The ever increasing volume of international trade of which up to 90% is fully containerized, demanded appropriate solutions for several other issues arising within the logistics operations across the entire supply chain network.

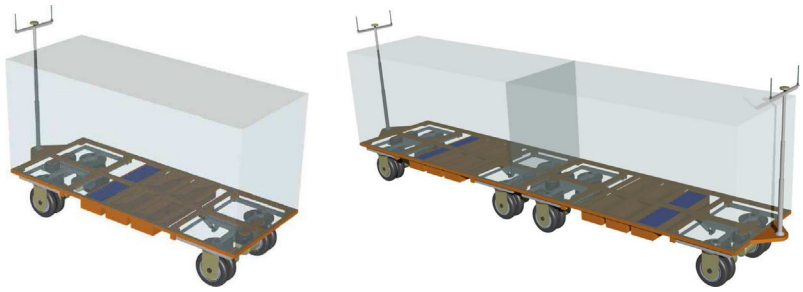
The circulation of this huge number of containers is currently taking place on three main routes: (1) Asia–Europe, (2) Trans-Pacific, and (3) Trans-Atlantic. In total more than 500 ports and tens of liner shipping companies are involved in the global maritime logistics.

Due to the intensive interaction and global-wide spatial distribution of components of the containerized transport system, inefficiencies in individual elements of the system (both from liner service providers and also from the port authority's point of views) propagate their negative impacts, locally and temporally, across the entire network of systems.

From the liner shipping industry point of view, larger vessels are needed to help the service providers benefit from the economies of scale in transporting the growing volumes. Deploying such vessels is very expensive (often more than several

* Corresponding author at: UArtois, LGI2A, F-62400 Béthune, France. Tel.: +33 3 21 63 23 00/7 86 64 73 34.
E-mail addresses: shahin.gelareh@univ-artois.fr, shahin.gelareh@gmail.com (S. Gelareh).

0968-090X/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.trc.2013.04.006



(a) An IAV for a 20-ft containers.

(b) Paired IAVs for larger than 20-ft (e.g. 40-ft.) containers.

Fig. 1. Intelligent and Autonomous Vehicles (IAVs).

million dollars per day) and in fact, this is only during the sailing time that vessels generate profit for the owners. The part of voyage time spent at different port calls on a route is referred to as *turnaround time* of a complete voyage and is actually the unprofitable part of the service. Therefore, economies of scale may not be exploited unless in long-haul transport. This means that the Liner Service Providers (LSPs) usually do not find it profitable to call too many ports along a service rotation (the so called '*string*'). Consequently, some major transshipment hubs, which are consolidation and distribution ports came into play, and in turn proxy service for other smaller ports in their region. The selection of such ports depends on several different factors. The most important ones are efficiency and infrastructure—of course assuming that a port has the potential to become a hub port, i.e., it has enough draught for accepting larger vessels, and it fulfils many more eco-political criteria.

From the port (equivalently terminal) operators' point of view, their competitiveness is essentially dependent on their ability to minimize the turnaround time of vessels while maximizing the port throughput. In this way, they can compete with other neighboring ports and survive in such a highly competitive market. Due to the dynamism and scale of competition, the goal is not reached unless some strategic, tactical and operational aspects of terminals are being constantly reviewed and addressed in better ways. This includes three main aspects: (i) the layout and equipment, (ii) the routing decisions and (iii) the scheduling of machines and resource allocation.

Since the beginning of containerization and containerized transport, several different types of port equipments have been developed. This includes various classes of cranes (for quay side and yard side) and carriers (e.g. single and multiple carriers, with and without the ability to lift) with very different physical and mechanical characteristics.

Along with the enhancements in Information and Communication Technologies (ICTs), the concept of automated and semi-automated infrastructures emerged and the port authorities started to gradually incorporate ICT tools to improve their efficiencies. Nowadays, the European Container Terminal (ECT), Ports of PSA in Singapore, Kaoshiung in Taiwan, Pusan in Korea, Kawasaki and Kajima in Japan, Thamesport in the UK, Bremerhafen and Hamburg in Germany, and Antwerp in Belgium are among the most automated container terminals in the world.

Similar trend is being observed all around the globe, and several projects are funded to develop new technologies for ports. Among them, vehicles with a certain degree of intelligence and autonomy—at the same time—which also exploit ICT and are equipped with several kinds of sensors as well as Geographical Information System (GIS) tools (see InTraDE[1] as an example) are of special interest.

Although there exist a fierce competition between the hub ports in attracting more customers by improving their efficiencies, but such a competition is not only limited to them. The smaller ports (e.g. Dublin Ferryport Terminal (DFT) in our case) are concerned with the efficiency of operation under the current trend of the import/export volume increase. This becomes even more serious when knowing that DFT (as many other ports in that region) has almost no possibility for any expansion of its terminal due to several reasons, including political ones and those related to the land-use and infrastructures in the neighborhood. This suggests investing on advanced technological development that would eventually lead to more efficient facilities (both transporters and stacking/unstacking machines).

Intelligent and Autonomous Vehicles (IAVs) (see Fig. 1) belong to a new class of transporters designed in the framework of *InTraDE* which generalize the concept of AGVs. Some distinguishing characteristics of IAVs are listed in the sequel:

– In contrast to AGVs, IAVs do not need to follow signed segments of the road to reach a destination and do not have to follow particular itineraries. Rather, they are MIMO (Multi-Input Multi-Output) systems equipped with several sensors enabling them to benefit from Geographical Positioning Systems (GPSs) based navigation systems and several other sensors to detect the distance to other vehicles, etc.

[1] Intelligent Transportation for Dynamic Environment (InTraDE) (http://www.intrade-nwe.eu/).

- Unit capacity of an IAV is one TEU and for transporting any container size between 1-TEU and 1-FFE, two IAVs can couple (pair) in a leader–follower manner resulting in a 40-ft capacity able to transport the containers of relevant size. There is no limit on how many times IAVs can pair and unpair again during a loading/unloading operation time horizon (Fig. 1a depicts a 20-ft single IAV and Fig. 1b a 40-ft paired IAVs).
- IAVs can form platoons based on a leader–follower manner, and every IAV is conceived in such a way as it can become leader or follower at any point in time during operations.
- All four wheels are equipped with actuators and a failure in any of the individual wheels, does not stop the vehicle from operation. Rather, it runs into a *degraded* mode of operation in which the operation continues with a lower performance.
- IAVs move laterally and longitudinally. Contrary to the normal transporters (such as AGVs and trucks), IAVs do not need larger spaces in order to maneuver. The wheels offer a 360-degree rotation flexibility.
- Finally, an IAV system should adapt itself to its surrounding environment. While in the case of AGV systems, this is the existing environment that must be adapted to them.

While in total, hundreds of ports all around the globe appear on the liner transport network, there are only few major LSPs serving major ports. This makes it quite natural that as far as the professional and academic literatures are concerned, there has been a larger corpus of studies on the optimization and simulation of operations at the port terminals.

1.1. Objective and contribution

This paper exploits the model presented in Ng et al. (2007) for AGVs (with further technical revision. See Appendix A for explanations.) and shows that it can be generalized to accommodate in both AGVs and IAVs cases. We then examine the impact of a class of valid inequalities on the computational performance of a general-purpose solver for solving instances of the problem. As an intrinsic property of most of the scheduling problems, only instances of very small size are *efficiently* solvable by general-purpose solvers. Therefore, we propose a Lagrangian relaxation-based decomposition approach equipped with a variable fixing and primal bound generation heuristic (exploiting information about the dual of the problem) to solve practical instances for high-quality solutions in reasonable time. Several classes of violated valid inequalities are identified and relaxed in Lagrangian fashion to accelerate convergence. As a case study, we apply the model on Dublin Ferryport Terminal in Ireland—one of the InTraDE's partner terminals for which IAVs are designed.

1.2. Literature review

As mentioned earlier, IAVs are technologically superior but very similar to the AGVs. Many differences such as independency from detecting signs on the ground surface in order to follow a path, the ability of in-place 90° rotation of wheels, leader–follower behavior, etc., are more relevant in the routing problems. However, the ability to join and cooperate in performing tasks is of interest in scheduling and makespan minimization. The literature of Operations Research is aware of several contributions dealing with scheduling of AGVs in container terminals and in manufacturing systems.

Meersmans and Wagelmans (2001b,a) proposed a heuristic algorithm for combined AGV and crane allocation problems.

Grunow et al. (2004) proposed an online logistics operation control using a priority-based approach which is compared against an offline approach. A simulation study of AGVs in an automated container terminal is proposed by Grunow et al. (2007) to examine the efficiency of different dispatching strategies.

Similarly, we can refer to Bish et al. (2001) for AGV dispatching and yard allocation; Bish (2003) extended this work to also take into account loading and unloading scheduling at quay cranes. Bish et al. (2005) extended the work in Bish (2003) and added some analytical performance studies on the proposed algorithm.

Cao et al. (2010) proposed an MIP formulation for an integrated yard truck and yard crane scheduling problems while only import containers were taken into account. They developed a combinatorial Benders decomposition to solve instances of their model.

Lee et al. (2010) studied a transshipment port where both loading and discharging containers were considered. While it has been often simplified by other authors, Lee et al. (2010) consider the delays at yard crane as well. The objective is to minimize the makespan of quay side operations as to reduce the turnaround time of vessels as a vessel can leave as soon as the last job of quay cranes is done. They proposed an MIP formulation but two heuristic approaches were used to solve the problem. Lee et al. (2010) work is based on Chen et al. (2007) but they consider loading and unloading simultaneously.

Ng et al. (2007) proposed a MIP model for scheduling a fleet of trucks at a container terminal while the fleet size is assumed to be given exogenously.

They compared several variants of genetic algorithms and showed that their variant outperforms all others.

While it must be emphasized that their proposed MIP model is not a basis for their solution method (except for solution quality evaluation), however, as shown in Appendix A using an intuitive example, the model in Ng et al. (2007) does not always produce a feasible solution to the problem of scheduling AGVs.

Other works on dispatching different equipments at container terminals can be found in Kim and Bae (1999, 2004) for AGV dispatching problem; Kim and Bae (2004) that employ a look-ahead strategy which considers local and temporal information of future tasks and assumes a dual cycle operation of AGVs; Nguyen and Kim (2009) as an extension of Kim and Bae

(2004) for Automated Lifting Vehicles (ALVs); Hartmann (2004) for a wider range of equipments and Narasimhan and Palekar (2002) where every truck is dedicated to a particular quay crane as opposed to that of Kim and Bae (2004).

2. Problem statement

As mentioned earlier, IAVs are intelligent vehicles which can work in groups. That means, given a set of individual IAVs, it is possible that anyone plays the role of a leader and the rest perform as followers to form a platoon. This resembles the behavior of locomotive and wagons of a train (however, in this article we restrict the size of such a train of IAVs to 2, i.e. the size of a 1-FFE container).

Given such a property, in order to perform a job of a 1-FFE container, two single IAVs must join together and perform the task. That is, should a 35-ft container be imported/exported, (1) two IAVs are chosen to move towards the corresponding crane, (2) do the pairing (join), (3) set the leader/follower designations, and (4) receive the container from the crane.

Except for the concept of cooperations in IAVs, the problem description of IAV scheduling is very similar to that of AGV scheduling as discussed in Ng et al. (2007):

At the earliest ready-time for vehicle $m \in \{1,\ldots,M\}$, i.e. t_m, the vehicle m is at location L_m. There are N tasks and to every task $n \in \{1,\ldots,N\}$ a pick-up P_n and a delivery D_n location is associated. There is an approximate travel time $t_{ll'}$ between every two locations $l \in L = \{L_1,\ldots,L_M, P_i,\ldots,P_N, D_i,\ldots,D_N\}$ and $l' \in L' = \{P_i,\ldots,P_N, D_i,\ldots,D_N\}$ and $t_{ll'} \neq t_{l'l}$, in general. The duration of job i (process time), T_i, is the time elapsed from the moment that IAV(s) arrive(s) at the pick-up location of task i, P_i, until the moment that IAV(s) leave(s) the drop-off location of task i, D_i (the average waiting times both at quay crane and yard crane are implicitly included). Then, T_i plus the maximum between (1) the time that the first IAV needs to travel empty from D_{i_1-1} (the preceding job on the first IAV) to P_i, and (2) the time that the second IAV needs to travel empty from D_{i_2-1} (the preceding job on the second IAV) to P_i (in the case that only one IAV is needed then it becomes: T_i plus the time that IAV needs to travel empty from D_{i-1} to P_i), accounts for the processing time of task i. There is a time at which a crane can generate task i (not earlier than that) and expect one or two IAVs to arrive at that location. This is referred to as "ready-time" a_i and $a_i \leqslant a_{i+1}$, $\forall i \in \{1,\ldots,N-1\}$. IAV scheduling problem seeks to minimize makespan such that the turnaround time of the corresponding vessel is minimized.

3. Mathematical model

The model is a mixed integer programming with objective function of minimizing the makespan. This means that we are minimizing completion time of the last task before the vessel is ready to depart.

We employ a similar notation as Ng et al. (2007).

Note 1. In the following, we use: (1) *container*, *task* and *job*, (2) *machine*, *vehicles* and *IAV*, interchangeably.

Note 2. Throughout this study, it is assumed that the time needed for two IAVs to physically join (once they both arrived at a certain pick-up location) is so small (few seconds) that can be safely ignored.

3.1. Parameters

The parameters are listed in Table 1:

3.2. Decision variables

The decision variables are listed Table 2:

Table 1
Model parameters.

r_m	The earliest time that the IAV_m will be available
L_m	The initial location of IAV_m at r_m
T_j	The process time of task j. The elapsed time between the arrival of an IAV to the location of pick up of task j and the time that its container being unloaded at the destination
t_{ij}	The travel time between locations i and j in terminal
a_i	The ready time of task i, i.e., the time when this task becomes available
N	The total number of tasks
M	The total number of available vehicles
I	The set of all tasks $I = \{i: 1 \leqslant i \leqslant N\}$
I'	The set of all tasks $I' = \{i: 0 \leqslant i \leqslant N+1\}$ where 0, $N+1$th tasks are *dummy source* and *dummy sink* tasks, respectively,
V	The set of all IAVs $V = \{m: 1 \leqslant m \leqslant M\}$
S_i	The size of container task i in terms of number of IAVs required
K	A sufficiently big constant (based on known upper bound on optimal solution)

The AGV scheduling problem in Ng et al. (2007) follows:
AGV-Scheduling (AGVS)

$$\min \quad W \tag{1}$$
$$\text{s.t.} \quad C_i \leqslant W \quad \forall i \in I \tag{2}$$
$$\sum_{1}^{M} y_{im} = 1 \quad \forall i \in I \tag{3}$$
$$\sum_{j \in \{I \cup \{N+1\}\}: j \neq i} x_{ijm} \leqslant y_{im} \quad \forall i \in I,\ m \in V \tag{4}$$
$$\sum_{i=1, j \neq i}^{N} x_{ijm} \leqslant y_{jm} \quad \forall j \in I,\ m \in V \tag{5}$$
$$1 \geqslant x_{ijm} + x_{jim} \geqslant y_{im} + y_{jm} - 1 \quad \forall i,j \in I : j \neq i,\ m \in V \tag{6}$$
$$C_i + t_{D_i,P_j} + T_j \leqslant K(1 - x_{ijm}) + C_j \quad \forall i,j \in I : j \neq i,\ m \in V \tag{7}$$
$$a_i + T_i \leqslant C_i \quad \forall i \in I,\ m \in V \tag{8}$$
$$r_m + t_{L_m,P_i} + T_i \leqslant C_i \quad \forall i \in I,\ m \in V \tag{9}$$
$$x_{ijm}, y_{im} \in \{0,1\}, \quad i \in I,\ j \in I',\ m \in V \tag{10}$$
$$W \geqslant 0, \quad C_j \geqslant 0, \quad j \in I. \tag{11}$$

It has been observed that the variable x_{ijm} needs to be more precisely redefined in order to avoid issues of infeasibility that are explained in Appendix A. Henceforward, we use the following definition for x_{ijm}:

x_{ijm}: 1, if the task j is performed *immediately* after task i on the vehicle m, 0 otherwise.

3.3. Intelligent and Autonomous Vehicle (IAV) Scheduling

We also needed to introduce one extra dummy task, *Source* (0th task), in addition to those introduced in the original AGVS model (N task and $N+1$th dummy task). The source task is the starting task on every IAV.

The dummy source node is required to model the cases where a larger than 1-TEU task is the first task of an IAV.

IAV Scheduling (IAVS)

$$\min \quad W \tag{12}$$
$$\text{s.t.} \quad (2), (7), (8), (9), (11),$$
$$\sum_{m \in V} y_{im} = S_i \quad \forall i \in I \tag{13}$$
$$x_{0im} + \sum_{j=1, j \neq i}^{N} x_{jim} = y_{im} \quad \forall i \in I,\ m \in V \tag{14}$$
$$x_{jN+1m} + \sum_{j=1, j \neq i}^{N} x_{ijm} = y_{im} \quad \forall j \in I,\ m \in V \tag{15}$$
$$x_{ijm} + x_{jim} \leqslant 1 \quad \forall i,j \in I : j \neq i,\ m \in V \tag{16}$$
$$\sum_{i} x_{0im} = 1 \quad \forall m \in V \tag{17}$$
$$\sum_{i} x_{iN+1m} = 1 \quad \forall m \in V \tag{18}$$
$$x_{ijm} \in \{0,1\}, (i,j,m) \in (I' \times I' \times V), \quad y_{im} \in \{0,1\}, \ \ (i,m) \in (I \times V) \tag{19}$$

The objective function (12) minimizes the makespan as the earliest possible time to complete the mission.

Constraints (2) are minimax constraints to determine the completion time of the last event. If a task j is performed after a task $i \neq j$ on machine m then the completion time of task j is no earlier than the completion time of task i, plus the travel time from the drop-off location of task i to the pick-up location of task j, plus the process time of task j (we say at least because one IAV *might need to wait* for another one before starting a task). Constraints (7) indicate this. Constraints (8) state that the completion time of task i cannot be earlier than its ready time, plus its process time. If task i is the first task assigned to machine m then it cannot be completed before the earliest time that the machine becomes ready, plus the travel time of machine from its initial location to the pick-up location of task i, plus the process time of task i as stated in constraints (9). Constraints (13) indicate that the number of IAVs allocated to the task i must be equal to the size of tasks. On the *same machine*, every task (including the source task and excluding the sink) is followed by a subsequent task. In the graph sense, from every node representing a task, one arc is encompassed for every machine assigned to complete that task. This is considered in (14). Similarly, on the *same IAV*, every task (represented by a node in the network) is carried out after another task (including sink and

Table 2
Decision variables.

x_{ijm}	1, if task j is performed after task $i \neq j$ on vehicle m, 0 otherwise
y_{im}	1, if task i is performed on vehicle m
C_i	The completion time of task i
W	The makespan, i.e. the completion time of the last task

excluding source). Constraints (15) stand for this. If both tasks i and $j \neq i$ are carried out by the same machine m then one precedes the other as stated by constraints (16). Constraints (17) and (18) ensure that the dummy source and sink tasks are performed as the first and the last tasks on every IAV.

Fig. 2 depicts an optimal solution of an instance with $|V| = 3$ and $|I| = 20$. Three IAVs are deployed (IAV1, the solid line; IAV2, the dashed line; and IAV3, the dotted line). The optimal schedule starts by a cooperation between IAV1 and IAV2 to perform tasks 1 and 3, while IAV3 alone performs task 2 as it is a 1-TEU container. Then IAV1 and IAV2 unpair (decouple) after completing task 3 and IAV2 performs task 4. Afterwards, IAV1 and IAV2 ally to perform task 5. The process continues following a similar manner until the final task which is the task 20 is completed at time 1400.67 via cooperation between IAV2 and IAV3.

3.3.1. Valid inequalities

If the machines were working alone (all the tasks were 1-TEU containers), then we could calculate the accumulated *time-in-service* of every machine independently, and take the maximum as the objective value. This can be done by simply following the arcs related to the machine on a graph similar to Fig. 2, starting from source and terminating at the sink. However, since in our model some IAVs may cooperate with each other at some points in time to accomplish some tasks, a number of dynamic pairings/unpairings takes place, which must be taken into account in order to calculate the makespan. In this cooperative environment, once the first IAV between the two assigned to a particular task arrives at the pick-up location, it must also wait for the second one to arrive (*wait-for-pairing*) before being able to pair and start a task of larger than 1-TEU. Therefore, unless a machine m is *exclusively* operating on the 1-TEU containers, all the way throughout the horizon, the value of following terms:

$$\sum_{i,j\neq i}\sum_{m}(t_{D_i,P_j}+T_j)x_{ijm}, \quad \forall m$$

would *not* exactly coincide with the actual time-in-service of that particular machine. Because the wait-for-pairing times corresponding to the tasks of larger than 1-TEU are not taken into account. However, this is still a valid lower bound on the makespan of machine m.

Let $\phi_i = \max\{a_i + T_i, r_m + t_{L_m P_i} + T_i : \ \forall m \in V\}, \ \forall i \in I$:

$$W \geqslant \sum_{i\in I}\phi_i x_{0im} + \sum_{i,j\neq i}(t_{D_i P_j}+T_j)x_{ijm}, \quad \forall m \in V. \tag{20}$$

Constraints (20) state that in a network representation of problem as in Fig. 2, the length of path from the source to the node preceding the sink node of every machine is a lower bound on W.

We show in the numerical section that these inequalities can have a significant impact on the performance of state-of-the-art general-purpose solvers such as CPLEX.

4. Solution method

IAVS is a challenging model for which even very small size instances with a few number of vehicles and 20 tasks could become relatively time-consuming and inefficient to solve by the general-purpose solvers. A major part of contributions in the literature has adopted (meta-) heuristic strategies, often without any indication of quality of their resulting solutions.

Ng et al. (2007) has proposed an efficient genetic algorithm for solving instances of the problem.

As the problem description of Ng et al. (2007) is based on AGVs whose operations are different from IAVs, their GA algorithm cannot be directly applied to the IAV scheduling problem. Unless we assume that all the tasks are 1-TEU containers which is neither realistic nor consistent with the ideas behind developing IAVs.

Here, we exploit some of the mathematical properties of our MIP model and propose a decomposition approach based on a Lagrangian relaxation (lower bound). It is equipped with an efficient local search approach and a variable fixing phase in order to produce upper bound on the optimal solution and obtain an indication of optimality.

4.1. Lagrangian decomposition for IAVS

Lagrangian relaxation for solving (mixed) integer programming problem was first proposed in Fisher (1981, 2004). The idea behind this method is to relax *complicating constraints* by penalizing the objective function upon violation of these constraints. The relaxed problem is expected to be easier to solve than the original problem and provides a dual bound on the

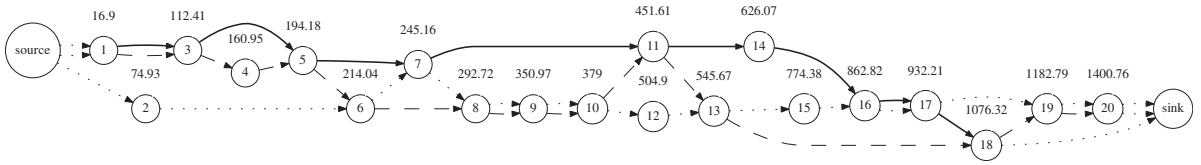


Fig. 2. Illustration of optimal solution of IAVS model instance.

optimal value of the problem as well as valuable information about the dual (see Guignard (2003) for a comprehensive survey of the method.).

Three well-known methods are commonly used in the literature for solving Lagrangian relaxation problems. The oldest and most well-known one is *subgradient* method as an iterative method for solving convex minimization problems. *Subgradient* was originally proposed in the 1960s in the former Soviet Union. Very similar method has also been proposed in Held and Karp (1971) for solving traveling salesman problem. Later, Lemarechal (1975) proposed the well-known *bundle* methods as an extension of subgradient. The volume algorithm was proposed in Barahona and Anbil (2000) as a method which simultaneously produces a primal feasible solution for the problem. Further analysis of volume algorithm and its relationship with the bundle method is studied in Bahiense et al. (2002).

Several variants of the subgradient method have been proposed in the literature. Here, based on some observations from the performance of bundle and volume algorithms in presence of big-M, we choose to employ the variant proposed by (Larsson et al., 1999) where an Ergodic sequence of subproblem solutions converges to the primal solution set.

A clever relaxation is the one applied in such a way as the resulting relaxed model can be further decomposed into some subproblems for which more efficient solution methods than only using general-purpose MIP solvers can be found. This is often referred to as *Lagrangian Decomposition*.

Michelon and Maculan (1991) proposed a Lagrangian decomposition for integer nonlinear programs with linear constraints. Reinoso and Maculan (1992) proposed a class of Lagrangian decomposition for integer linear programming. Gelareh et al. (2012b) has also developed an efficient Lagrangian decomposition for a network design problem in liner shipping. Several techniques in Lagrangian decomposition are surveyed in Guignard (2003).

We chose to relax constraints (14), (15) and henceforward we use a prefix `LRX` to indicate that a model is either a Lagrangian RelaXation problem or one of its decomposed parts.

(LRX-IAVS)'

$$\begin{aligned} \min \quad & W + \sum_{i,m} u^1_{im}\left(x_{0im} + \sum_{j=1, j\neq i}^{N} x_{jim} - y_{im}\right) + \sum_{i,m} u^2_{im}\left(x_{jN+1m} + \sum_{j=1, j\neq i}^{N} x_{ijm} - y_{im}\right) \\ \text{s.t.} \quad & (2), (13), (16)-(18), \\ & (7)-(9), (19), (11) \end{aligned} \tag{21}$$

In this relaxation, Lagrangian multipliers are chosen to be $u^1_{im} \in \mathbb{R},\ u^2_{im} \in \mathbb{R},\ \forall i, m$.

4.2. More constraints to relax

From Fig. 3, one observes that in a solution of a given iteration of subgradient for the above relaxation, there are no arcs arriving at nodes 5, 6, 7 and 8 and no arc leaves 1, 3 and 4 as it is not enforced by any constraint. Moreover, there is no constraint to enforce that there must be one task (including the dummy source and sink, wherever applies) before and after every (non-dummy) task.

In order to encourage this and improve convergence of subgradient, we additionally dualize the following constraints:

$$\sum_{m} x_{0im} + \sum_{j\neq i,m} x_{jim} = \sum_{m} x_{iN+1m} + \sum_{j\neq i,m} x_{ijm}, \quad \forall i \tag{22}$$

using $u^3_i \in \mathbb{R},\ \forall i$.

In this particular example shown in Fig. 4, three tasks namely 2, 6 and 8 are 2-TEU tasks, and the rest are 1-TEU. Nevertheless, no arc is arriving at any of these three. Therefore, we also consider dualizing the following constraints:

$$\sum_{j\neq i,m} x_{jim} \geqslant S_i, \quad \forall i \tag{23}$$

However, extensive computational experiment revealed that the following less tight constraints provide better results from the bound quality point of view:

$$\sum_{j\neq i,m} x_{jim} \geqslant 1, \quad \forall i \tag{24}$$

Similarly, the following constraints:

$$\sum_{j\neq i,m} x_{ijm} \geqslant 1, \quad \forall i \tag{25}$$

are dualized using $u_i^4,\ u_i^5 \in \mathbb{R}^-,\ \forall i$ multipliers:

Theorem 3. *The total number of arcs (between two non-dummy tasks), $|E|$, ($x_{ijm} = 1,\ i,\ j \notin \{i, 0, N+1\}$) in an optimal solution is bounded by:* $\max\{N-M, 0\} \leqslant |E| \leqslant (\sum_i S_i - 1)$.

Proof. If there is no job i where $S_i > 1$, then we have:

a. If the total number of tasks is equal to the total number of machines ($N = M$) *and* every machine does *only one* task, the total number of arcs is equal to 0. The lower bound is obtained because we also have $N - M = 0$ (see Fig. 5a).
b. If all the tasks are carried out by one single machine, the only tree contains $N - 1$ arcs. Since every node corresponds to one task, the upper bound is obtained (see Fig. 5b).

On the other hand if there is at least one task i with $S_i = 2$ then:

a. If the total size of tasks (total with respect to the size) is equal to the total available capacity, i.e. $|M| = \sum_i S_i$, such that every machine does *only* one task (perhaps two IAVs cooperate on a single task but no IAV contributes in more than one task) then $M > N$ and $\max\{N - M, 0\} = 0$ and the lower bound is valid (see Fig. 5d).
b. If for all the tasks i we have $S_i > 1$ and only two IAVs, say IAV1 and IAV2, cooperate in performing them, then the total number of arcs would be $\sum_i S_i - 2 = 2(N-1)$ ($N - 1$ arcs are needed to make the tree for every IAV). Therefore, the upper bound still remains valid ($\sum_i S_i - 2 < \sum_i S_i - 1$) (see Fig. 5d).

The aforementioned cases were extreme ones which hit (or approach) the bounds. Under any other condition, the total number of arcs remains between the two bounds. □

The Theorem 3 can be presented by the following constraint:

$$\sum_{i,j\neq i,m} x_{jim} \geqslant q = \max\{N-M, 0\} \tag{26}$$

This constraint is added to the model.

4.2.1. Decomposition

To facilitate resolution of larger instances, we decompose the ($LRX - IAVS$) into two sub-problems by taking into account that the sequencing part and the scheduling parts are linked only by the big-M constraints. Therefore, we add a set of copy constraints and dualize them using $u_{ijm}^6 \in \mathbb{R},\ \forall m, i \neq j$:

$$x_{ijm} = x'_{ijm} \quad \forall i, j, m \tag{27}$$

$$x'_{ijm} \in \{0, 1\} \tag{28}$$

By doing so, we separate the sequencing part (LRX-IAVS-Seq in the space of binary x, y) from the scheduling part (LRX-IAVS-Sch in the space of continuous W, C and binary x').

The resulting relaxation follows:

(LRX-IAVS)

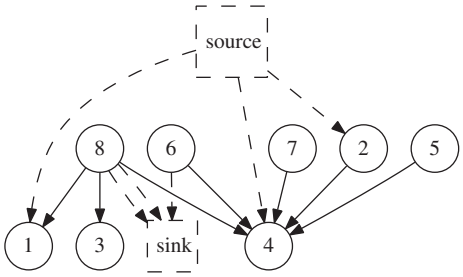


Fig. 3. There is no arc arriving at nodes 5, 6, 7 and 8 and no arc departing from 1, 3 and 4.

$$\min \quad W + \sum_{i,m} u^1_{im}\left(x_{0im} + \sum_{j=1,j\neq i}^{N} x_{jim} - y_{im}\right) + \sum_{i,m} u^2_{im}\left(x_{iN+1m} + \sum_{j=1,j\neq i}^{N} x_{ijm} - y_{im}\right)$$
$$+ \sum_i u^3_i\left(\sum_m x_{0im} + \sum_{j\neq i,m} x_{jim} - \sum_m x_{iN+1m} - \sum_{j\neq i,m} x_{ijm}\right) + \sum_i u^4_i\left(1 - \sum_{j\neq i,m} x_{jim}\right)$$
$$+ \sum_i u^5_i\left(1 - \sum_{j\neq i,m} x_{jim}\right) + \sum_{i,j,m} u^6_{ijm}(x_{jim} - x'_{jim}) \tag{29}$$

s.t. (2), (13), (16)–(18),

$$C_i + t_{D_i,P_j} + T_j \leqslant K(1 - x'_{ijm}) + C_j \quad \forall i,\ j \in I : j \neq i,\ \ m \in V \tag{30}$$

(8), (9), (26), (11),

$$x_{ijm}, x'_{ijm} \in \{0,1\}, \quad (i,j,m) \in (I' \times I' \times V),\ \ y_{im} \in \{0,1\},\ \ (i,m) \in (I \times V) \tag{31}$$

The problem decomposes into *four* independent problems:

(a) Scheduling subproblem of LRX-IAVS:
(LRX-IAVS-Sch)

$$\min \quad W - \sum_{i,j,m} u^6_{ijm} x'_{jim} \tag{32}$$

s.t. (2), (8), (9), (11),

$$C_i + t_{D_i,P_j} + T_j \leqslant K\left(1 - x'_{ijm}\right) + C_j \quad \forall i,j \in I : j \neq i,\ \ m \in V \tag{33}$$

$$x'_{ijm} \in \{0,1\}, \quad (i,j,m) \in (I \times I \times V) \tag{34}$$

(b) A semi-knapsack subproblem:
(LRX-IAVS-Seq)

$$\min \quad \sum_{i,m} u^1_{im} \sum_{j=1,j\neq i}^{N} x_{jim} + \sum_{i,m} u^2_{im} \sum_{j=1,j\neq i}^{N} x_{ijm} + \sum_i u^3_i \sum_{j\neq i,m} x_{ijm} - \sum_i u^3_i \sum_{j\neq i,m} x_{jim} - \sum_i u^4_i \sum_{j\neq i,m} x_{jim} - \sum_i u^5_i \sum_{j\neq i,m} x_{jim} + \sum_{i,j,m} u^6_{ijm} x_{jim}$$
$$+ \sum_i u^4_i(1) + \sum_i u^5_i(1) \tag{35}$$

s.t. (16), (26),

$$x_{ijm} \in \{0,1\}, \quad (i,j,m) \in (I \times I \times V) \tag{36}$$

(c) A semi-assignment subproblem only in the space of dummy source and sink variables which is again decomposable for the source and the sink variables:
(LRX-IAVS-SourceSink)

$$\min \quad \sum_{i,m} u^1_{im} x_{0im} + \sum_{i,m} u^2_{im} x_{iN+1m} + \sum_{i,m} u^3_i x_{0im} - \sum_{i,m} u^3_i x_{iN+1m} \tag{37}$$

s.t. (17), (18),

$$x_{iN+1m}, \quad x_{0im} \in \{0,1\},\ \ (i,j,m) \in (I \times I \times V),\ \ y_{im} \in \{0,1\},\ \ (i,m) \in (I \times V) \tag{38}$$

(d) A knapsack problem in the space of y variables:
(LRX-IAVS-y)

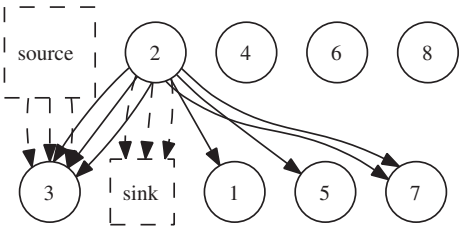
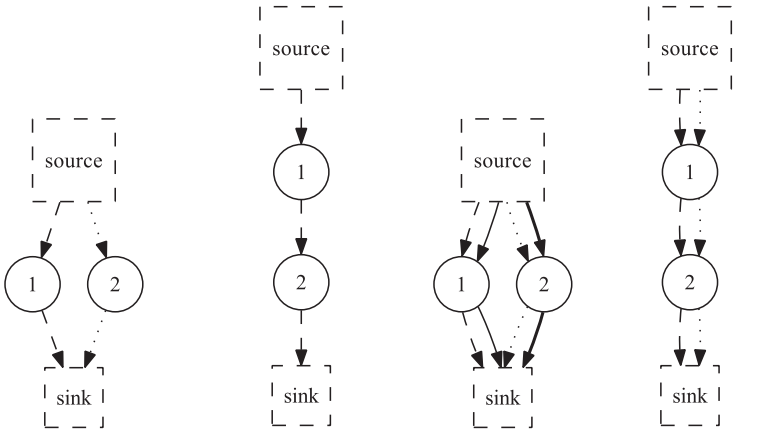


Fig. 4. The total number of arc arriving at a task node is at least equal to the size of the task.



(a) $S_i = 1,\ \forall i$ and $N = M$

(b) $S_i = 1,\ \forall i$ and all by one machine.

(c) $\exists i, S_i > 1$ and $\sum_i s_i = M$

(d) $\exists i, S_i > 1$ and all by two machines.

Fig. 5. Examples of extreme cases for Theorem 3.

$$\min \quad -\sum_{i,m} u^1_{im} y_{im} - \sum_{i,m} u^2_{im} y_{im} \tag{39}$$

$$\text{s.t.} \quad (13),$$

$$y_{im} \in \{0,1\}, \quad (i,m) \in (I \times V) \tag{40}$$

where $V^*(LRX - IAVS) = V^*(LRX - IAVS - Sch) + V^*(LRX - IAVS - Seq) + V^*(LRX - IAVS - SourceSink) + V^*(LRX - IAVS - y)$ and $V^*(\cdot)$ stands for the optimal value.

4.2.2. Algorithms for solving subproblems

The problems (LRX-IAVS-Seq), (LRX-IAVS-SourceSink) and (LRX-IAVS-y) are binary problems, which can be solved by inspection without any need to resort to a general-purpose LP/MIP solver. The only different problem is (LRX-IAVS-Sch), which does not show any trivial exploitable structure to help in solving it by inspection. Still, it can be relatively efficiently solved using a state-of-the-art solver.

In the following, we outline specialized algorithms for each subproblem.

Algorithm for (LRX-IAVS-Seq). Let α_{ijm} be the coefficient of x_{ijm} in (LRX-IAVS-Seq) after re-ordering the terms. The problem does not have too many constraints:

Algorithm 1. Inspection algorithm for $(LRX - IAVS - Seq)$.

Input: Lagrangian multipliers

Output: $x_{ijm}, \forall i, j, m$

$counter = 0$;

for $s = 1$ *to* q **do**

$(i, j, m) = argmin\{\alpha_{ijm} : i, j \neq i \in I, m \in V\}$;

$x_{ijm} := 1$;

$x_{jim} := 0$;

$\alpha_{ijm} = \infty$;

Except for the function *argmin*(), this algorithm terminates in linear time.

Algorithm for (LRX-IAVS-SourceSink). Let β_{0im} be the coefficient of x_{0im} and $\beta_{i\ N+1m}$ be the coefficient of $x_{i\ N+1m}$. Then let $(i',m') = argmin\{\beta_{0im}: i \in I, m \in V\}$, we set $x_{0i'm'} = 1$ and $x_{0i^*m^*} = 0,\ \forall i^* \neq i',\ m^* \neq m'$. Also let $(i'',m'') = argmin\{\beta_{i\ N+1m}: i \in I, m \in V\}$, we set $x_{0i''m''} = 1$ and $x_{0i^{**}m^{**}} = 0,\ \forall i^{**} \neq i'',\ m^{**} \neq m''$.

Algorithm 2. Inspection algorithm for ($LRX - IAVS - SourceSink$).

Input: Lagrangian multipliers

Output: $x_{0im}, x_{i\ N+1\ m}, \forall i, m$

$counter = 0$;

for $m \in V$ **do**

  $(i', m') = argmin\{\beta_{0im} : i \in I, m \in V\}$;

  $x_{0i'm'} := 1$;

for $m \in V$ **do**

  $(i'', m'') = argmin\{\beta_{i\ N+1\ m} : i \in I, m \in V\}$;

  $x_{i\ N+1\ m} := 1$;

This algorithm terminates in $\mathcal{O}(V)$.

Algorithm for (LRX-IAVS-y). Let γ^{y}_{im} be the cost of variable y_{im}. If $S_i = 1$ then $(i, m') = argmin\{\gamma^{y}_{im} : m \in V\},\ y_{im'} = 1$. If $S_i = 2$ then $(i, m'') = argmin\{\gamma^{y}_{im} : m \neq m'\}$ and $y_{im'} = y_{im''} = 1$. Set $y_{im^*} = 0,\ \forall i, m^* \notin \{m', m''\}$.

Algorithm 3. Inspection algorithm for ($LRX - IAVS - y$).

Input: Lagrangian multipliers

Output: $y_{im}, \forall i, m$

for $i \in I$ **do**

  $(i, m') = argmin\{\gamma^{y}_{im} : m \in V\}$;

  $y_{im'} = 1$;

  if $S_i = 2$ **then**

    $(i, m'') = argmin\{\gamma^{y}_{im} : m \neq m', m \in V\}$;

    $y_{im'} = y_{im''} := 1$;

This algorithm terminates in $\mathcal{O}(N)$.

4.3. Variable fixing

During iterations of subgradient optimization, after a few steps, when the multipliers are stabilized, we have lower bound obtained from the Lagrangian relaxation, LB^{LRX}, and also an upper bound, UB^{heur}, using a heuristic which we describe as follows.

Given the reduced cost (in terms of Lagrangian multipliers) of a binary variable *var*, (where currently *var* = 0), if the reduced cost $RC^{var} > UB - LB$ then this variable will not take 1 in any optimal solution. Because by setting it to 1 the lower bound would exceed the upper bound (best solution found by the heuristic) which is impossible. Therefore, we can exclude its column from further computations and obtain a reduced size problem.

We give priority to $y_{im}, \forall i \in I, m \in V$ variables for this elimination test. Because, eliminating one y_{im} implies a reduction of all variables x_{ijm} and x_{jim}, $\forall j \in I', m \in V$ which is quite significant and makes the problem size iteratively smaller and resolution becomes easier.

We perform this test whenever lower bound improves during the subgradient iterations.

4.4. Primal bound

A computationally inexpensive heuristic algorithm is needed in order to exploit the information obtained from the LR model and produce high-quality feasible solutions. For every given solution to the LR, the assignment of jobs to the machines

is determined in (LRX-IAVS-y) while the sequencing part (LRX-IAVS-Seq) may not produce a complete description or even a partial feasible solution.

The basic idea behind this heuristic is to do the following:

- accepting the assignment reported by (LRX-IAVS-y) and optimizing the sequence by taking into account the tasks which need to be done by cooperation between two machines,
- optimizing for the assignment, which inevitably leads to a sequence optimization as well.

We employ a local search approach which performs a re-assignment of jobs among IAVs in a systematic way aiming at minimizing the makespan for every machine.

4.4.1. Initial solution

We start by distributing the tasks ($\omega_i, \forall i \in I$) between the machines in such a way that a_i of the first tasks be close to each other. We follow the same pattern for all other tasks. Let $A = \{1,4,7,11,15,16,17,19,21\}$ and $M = 3$. We distribute the jobs as follows:

$$\omega_1(a_1 = 1) \rightarrow \omega_4(a_4 = 11) \rightarrow \omega_7(a_7 = 17)$$

$$\omega_2(a_2 = 4) \rightarrow \omega_5(a_5 = 15) \rightarrow \omega_8(a_8 = 19)$$

$$\omega_3(a_3 = 7) \rightarrow \omega_6(a_6 = 16) \rightarrow \omega_9(a_9 = 21)$$

Let assume that $S(\omega_4) = S(\omega_8) = 2$. The following pattern applies:

$$\omega_1(a_1 = 1) \rightarrow \omega_4(a_4 = 11) \rightarrow \omega_6(a_6 = 16) \rightarrow \omega_8(a_8 = 19)$$

$$\omega_2(a_2 = 4) \rightarrow \omega_4(a_4 = 11) \rightarrow \omega_7(a_7 = 17) \rightarrow \omega_9(a_9 = 21)$$

$$\omega_3(a_3 = 7) \rightarrow \omega_5(a_5 = 15) \rightarrow \omega_8(a_8 = 19)$$

where duplicate copies of every job with size >1 is present in the representation. By doing so, we aim at more evenly distributing the jobs among the machines such that the variances of completion times of the last tasks on the machines is minimized.

4.4.2. Neighborhood structure and move strategies

We employ two kinds of move: *Temporal* and *Spatial*. In temporal moves, the sequence of tasks performed on the same machine is modified while in the spatial moves, a job is assigned to different machine(s).

4.4.2.1. Temporal move. A temporal move is a move which transforms the current solution to another solution by putting forward or postponing a job by only one step on the same machine, if feasible. That is, if job ω_i is the jth job on the machine m then the temporal moves results in a solution having ω_i the $(j-1)$th or $(j+1)$th job on the same machine.

4.4.2.2. Spatial move. A spatial move is a move which transforms the current solution to another solution by changing the machine to which it is assigned. That is, if job ω_j is the jth job on machine m then applying temporal moves results in a solution having ω_i as the j'th job on another machine m' (the choice of j' is rather biased towards a greedy approach) minimizing the possible increase in the completion time of the last task on m'.

4.4.2.3. Search strategy. Our main emphasis is on distributing jobs on the machines such that the machines finish their final tasks as early as possible and also very close to each other. This helps to avoid having a few machines of heavily loaded with long makespan and the rest being less utilized with significantly shorter makespans. In order to achieve this goal, we must first ensure that—given our neighborhood structures and a greedy search—there is no other sequence better than the current one, which makes the makespan shorter for a given machine. This means that we employ a two-level search. In the first level, we only employ temporal moves for each machine in order to obtain high-quality sequences for a given assignment pattern. We try to greedily re-sequence the jobs on every machine by finding the best place of each job starting from the first to the last one in the current solution. It must be mentioned that for the jobs with $S(\omega_i) > 1$, any re-sequencing may result in change of makespans of its both collaborating machines. Therefore, except for the machines which are not cooperating at all with any other machine, the rest of machines are treated lexicographically to avoid any tie and confusion. In the second phase, spatial moves are performed in which we start from the machines with the longest makespans and try to re-assign some of their tasks to other machines with shorter makespans and subsequently perform some iterations of temporal moves. Of course, the issues related to the tasks of larger than 1-TEU are taken into account.

4.4.2.4. Tabu list. We employ a tabu list which keeps track of moves in the search space. A spatial or temporal move becomes forbidden for η^l, η^m iterations, respectively. That is, if a move has caused a job on machine m being postponed or put forward, then the reverse move will become forbidden for η^t iterations. This analogously applies to the spatial moves.

4.4.2.5. Escape strategies. The greedy approaches are often in danger of being trapped in local optima and encounter a premature convergence. In order to avoid that, we incorporate some degrees of randomness in our approach. In moving from one solution to a neighboring one, we accept even the degrading solutions if the value of $e^{\left(\frac{\Delta f}{\log iteration}\right)}$ is larger than θ^{tr} and reject it, otherwise.

4.4.2.6. Termination criteria. The termination criterion is set to reaching a total number of consecutive non-improving iterations.

5. Numerical results

We generated instances based on the perturbed data of Dublin Ferryport Terminal (DFT). Instances range from very few up to a few hundreds of jobs, and the number of deployed IAVs are chosen to be in {2,...,6}. Through an extensive experiments on the instances of the problem, we have observed that our heuristic is quite robust against possible changes in η^l and η^m. Therefore, we have fixed these two values to $\eta^l = \eta^m = 10$. The same applies to the choice of θ^{tr}. For θ^{tr}, the best value between $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$ turned out to be 0.8.

As for termination criteria, we stopped the search as soon as a consecutive $\frac{N}{M}$ unsuccessful iterations being observed.

As mentioned earlier there is a minimal level of perturbation in the real data in order to respect the confidentiality. A part of this perturbation is to linearly translate the time unit. That means, the time unit we present here is none of the second, minute or hour—rather, a linearly transformed one.

There are in general 16 drop-off/pick-up locations. The terminal yard is composed of 12 stacks—6 import and 6 export stacks. For the sake of simplicity and to avoid excessive computational efforts, we have considered that every stack has only one drop-off/pick-up point along its length.

All the numerical results have been carried out on an Intel(R) Core(TM)2 Duo CPU 2.93 Ghz with 4.00 GB RAM.

5.1. CPLEX and efficiency of the valid inequalities

In Table 3 we report the impact of valid inequalities on the performance of CPLEX. The table reports statistics of initial model, which includes the valid inequalities (20).

In Table 3, the first column represents the instance name. The rest of table is divided into two blocks—a block for the initial model and another one for the model with the additional inequalities. The first column of each block represents the number of branch-and-bound nodes. The second one represents CPU time taken by CPLEX before termination. The third column reports the termination status observed by CPLEX, and the last column of the block represents the MIP relative gap, if any, observed upon termination of CPLEX.

Wherever an improvement has been observed, we highlighted it by a **bold-faced** entry in the table. The *italic* entries are those cases for which the added inequalities deteriorated performance of CPLEX in the sense that the optimality has been lost or a similar solution quality has been obtained in a longer computational time compared to the initial model.

Instance names are written in *mX_nY* format, where *X* is the number of machines and *Y* is the number of tasks. We restrict the instance sizes to those which CPLEX could accommodate on our machine. Usually, on our machine $M = 6$ and $N = 100$ is the size limit that we could use CPLEX for resolution (still many instances within this range are not solvable on our machine, and we had to use CPLEX parameters to set some algorithmic limits).

We used CPLEX 12.1, a time limit (`tiLim`) of 1200 s and a limit on the maximum number of branch-and-bound node (`nodeLim`) 500,000.

In Table 3, we use the same status codes which are used by CPLEX to report the status of solver upon termination of MIP algorithm. `Optimal` is reported when CPLEX terminates with optimality, `NodeLimFeas` indicates that the solver terminated by hitting the user-defined node limit in the branch-and-bound tree *and* a feasible solution has been found. `NodeLimInfeas` states that the user-defined node limit has been reached but no feasible solution detected and `AbortTimeLim` indicates that CPLEX terminated when a user-defined time limit of 1200 s has been reached without any feasible solution being found.

After adding those valid inequalities, for some instances the number of branch-and-bound nodes were reduced by almost half a million nodes. In fact, whenever an improvement was observed the reduction (by some hundreds of thousand nodes) in the branch-and-bound tree size as well as the CPU time was very significant. There were of course cases where no improvement occurred or even the solver performance has deteriorated.

There were instances for which CPLEX reported `NodeLimFeas` (the node limit was reached) such as *m*2_*n*8 and *m*3_*n*12. Here, the improved formulation proved optimality and reduced the branch-and-bound tree by almost half a million nodes to 5613 and 63,367 nodes, respectively.

There were instances such as *m*2_*n*40, *m*2_*n*50, *m*3_*n*50, *m*4_*n*40, *m*4_*n*50, *m*5_*n*50, *m*6_*n*36, *m*6_*n*40, *m*6 *n*50 for which the initial model reported `AbortTimeLim`. For those instances the tightened model proved the optimality at the root node

Table 3
The impact of valid inequalities (20) on the performance of CPLEX.

Instance	Initial model				Model with (20)			
	nbNodes	CPUTime	CplexStatus	MIPRelativeGap	nbNodes	CPUTime	CplexStatus	MIPRelativeGap
m2_n4	541	0.34	Optimal	0.00	60	**0.09**	Optimal	0.00
m2_n8	500001	118.00	NodeLimFeas	18.11	5613	2.81	**Optimal**	0.00
m2_n12	500001	204.58	NodeLimFeas	1.95	500002	356.14	NodeLimFeas	10.91
m2_n16	0	0.17	Optimal	0.00	500001	576.85	*NodeLimFeas*	23.06
m2_n20	500002	424.68	NodeLimInfeas	–	500001	602.24	NodeLimInfeas	–
m2_n24	485382	1430.86	AbortTimeLim	–	500002	1190.63	**NodeLimFeas**	17.63
m2_n28	0	0.64	Optimal	0.00	500001	1241.77	NodeLimFeas	58.92
m2_n32	379179	1427.03	AbortTimeLim	–	388520	1411.86	AbortTimeLim	–
m2_n36	277723	1406.96	AbortTimeLim	–	373301	1435.74	AbortTimeLim	–
m2_n40	379774	1385.73	AbortTimeLim	–	0	1.45	**Optimal**	0.00
m2_n50	186465	1430.03	AbortTimeLim	–	0	2.93	**Optimal**	0.00
m2_n60	500001	1865.13	NodeLimInfeas	–	500001	1836.03	**NodeLimFeas**	33.00
m2_n80	500001	1867.12	NodeLimInfeas	–	500001	2010.20	**NodeLimFeas**	24.00
m2_n100	500001	2556.10	NodeLimInfeas	–	500001	2785.94	**NodeLimFeas**	29.00
m3_n4	671	0.50	Optimal	0.00	20	**0.25**	Optimal	0.00
m3_n8	0	0.22	Optimal	0.00	0	**0.08**	Optimal	0.00
m3_n12	500001	518.22	NodeLimFeas	20.69	63367	51.39	**Optimal**	0.00
m3_n16	500001	966.22	NodeLimFeas	0.30	500002	772.81	NodeLimFeas	45.51
m3_n20	0	0.81	Optimal	0.00	0	0.76	Optimal	0.00
m3_n24	500002	1360.42	NodeLimFeas	18.08	500002	1337.47	NodeLimFeas	19.60
m3_n28	509	3.96	Optimal	0.00	500001	1177.17	*NodeLimFeas*	0.20
m3_n32	0	1.28	Optimal	0.00	0	1.40	Optimal	0.00
m3_n36	9671	42.93	Optimal	0.00	482	**9.67**	Optimal	0.00
m3_n40	0	2.17	Optimal	0.00	0	2.78	Optimal	0.00
m3_n50	112378	1412.29	AbortTimeLim	–	0	3.85	**Optimal**	0.00
m3_n60	500001	1972.34	NodeLimInfeas	–	500001		**NodeLimFeas**	16.00
m3_n80	500001	2228.63	NodeLimInfeas	–	500001		**NodeLimFeas**	24.00
m3_n100	500001	2882.55	NodeLimInfeas	–	500001		**NodeLimFeas**	12.00
m4_n4	42	0.31	Optimal	0.00	97	1.11	Optimal	0.00
m4_n8	238	0.33	Optimal	0.00	75	0.41	Optimal	0.00
m4_n12	0	0.41	Optimal	0.00	500001	698.87	*NodeLimFeas*	9.93
m4_n16	0	0.48	Optimal	0.00	500001	885.35	*NodeLimFeas*	66.08
m4_n20	0	0.78	Optimal	0.00	0	**0.70**	Optimal	0.00
m4_n24	0	1.95	Optimal	0.00	0	**1.39**	Optimal	0.00
m4_n28	0	2.56	Optimal	0.00	86060	274.09	Optimal	0.00
m4_n32	0	2.23	Optimal	0.00	3	3.76	Optimal	0.00
m4_n36	545	22.59	Optimal	0.00	2847	37.92	Optimal	0.00
m4_n40	106152	1413.60	AbortTimeLim	3.52	0	4.77	**Optimal**	0.00
m4_n50	140954	1403.62	AbortTimeLim	–	0	9.20	**Optimal**	0.00
m4_n60	500001	1800.58	NodeLimInfeas	–	500001	1777.01	**NodeLimFeas**	10.00
m4_n80	500001	2206.94	NodeLimInfeas	–	500001	1866.89	**NodeLimFeas**	35.00
m4_n100	500001	2602.40	NodeLimInfeas	–	500001	2893.39	**NodeLimFeas**	38.00
m5_n4	31	0.36	Optimal	0.00	37	**0.25**	Optimal	0.00
m5_n8	538	1.51	Optimal	0.00	61	**0.51**	Optimal	0.00
m5_n12	0	0.39	Optimal	0.00	3061	9.67	Optimal	0.00
m5_n16	21162	52.18	Optimal	0.00	500001	756.45	*NodeLimFeas*	14.76
m5_n20	12626	52.01	Optimal	0.00	0	**1.26**	Optimal	0.00
m5_n24	0	2.89	Optimal	0.00	0	**2.18**	Optimal	0.00
m5_n28	488	6.43	Optimal	0.00	482	7.77	Optimal	0.00
m5_n32	–	–	–	0.00	497	19.84	**Optimal**	0.00
m5_n36	0	6.07	Optimal	0.00	0	**4.74**	Optimal	0.00
m5_n40	0	9.03	Optimal	0.00	0	**6.13**	Optimal	0.00
m5_n50	82710	1446.27	AbortTimeLim	0.00	0	35.12	**Optimal**	0.00
m5_n60	500001	1668.49	NodeLimInfeas	–	500001	1516.95	**NodeLimFeas**	36.00
m5_n80	500001	2119.25	NodeLimInfeas	–	500001	2190.70	**NodeLimFeas**	17.00
m5_n100	500001	2675.28	NodeLimInfeas	–	500001	2410.12	**NodeLimFeas**	10.00
m6_n8	156	0.59	Optimal	0.00	18	**0.36**	Optimal	0.00
m6_n4	39	0.19	Optimal	0.00	0	**0.11**	Optimal	0.00
m6_n12	359	1.65	Optimal	0.00	0	**0.45**	Optimal	0.00
m6_n16	0	0.80	Optimal	0.00	0	1.45	Optimal	0.00
m6_n20	0	1.09	Optimal	0.00	18	2.79	Optimal	0.00
m6_n24	9689	46.27	Optimal	0.00	339680	1324.03	Optimal	0.00
m6_n28	0	3.14	Optimal	0.00	0	4.85	Optimal	0.00
m6_n32	46305	421.70	Optimal	0.00	220828	1523.63	*AbortTimeLim*	0.07
m6_n36	123198	1439.56	AbortTimeLim	–	0	7.58	**Optimal**	0.00
m6_n40	141421	1410.61	AbortTimeLim	–	0	8.97	**Optimal**	0.00
m6_n50	10633	1515.78	AbortTimeLim	–	0	23.21	**Optimal**	0.00

Table 3 (*continued*)

Instance	Initial model				Model with (20)			
	nbNodes	CPUTime	CplexStatus	MIPRelativeGap	nbNodes	CPUTime	CplexStatus	MIPRelativeGap
m6_n60	500001	1914.72	NodeLimInfeas	–	500001	1535.65	**NodeLimFeas**	22.00
m6_n80	500001	2136.14	NodeLimInfeas	–	500001	1842.02	**NodeLimFeas**	32.00
m6_n100	500001	2720.65	NodeLimInfeas	–	500001	2882.80	**NodeLimFeas**	40.00

with *no* branching (but perhaps with the primal/dual reduction, preprocessing and the cuts added by CPLEX at the root node).

Several instances such as *m*2_*n*60, *m*2_*n*80, *m*2_*n*100, *m*3_*n*60, *m*3_*n*80, *m*3_*n*100, *m*4_*n*60, *m*4_*n*80, *m*4_*n*100, *m*5_*n*60, *m*5_*n*80, *m*5_*n*100, *m*6_*n*60, *m*6_*n*80, *m*6_*n*100 initially reported `NodeLimInfeas` while after adding the new inequalities, they turned to `NodeLimFeas` and the solution qualities were relatively good, in general.

For *m*5_*n*32, CPLEX was unable to report any status using the initial model, while after the tightening, it was solved to optimality in 19.84 s and used only 497 branch-and-bound nodes.

In one instance of the initial model, *m*2_*n*24, we observed `AbortTimeLim` status without any feasible solution while after adding those valid inequalities CPLEX terminated to `NodeLimFeas` with a relatively good quality feasible solution being found.

On the other hand, there were also cases where adding inequalities (20) deteriorated CPLEX performance (mainly due to introducing degeneracy) such as *m*2_*n*16, *m*3_*n*28, *m*4_*n*12, *m*4_*n*16, *m*5_*n*16, *m*6_*n*32. For these instances CPLEX has initially reported `Optimal` while introducing the new inequalities has caused CPLEX to terminate by `NodeLimFeas`.

In general, in 28 cases improvements and in 6 cases deterioration in CPLEX performance has been observed.

For every fleet size and on larger instances, namely instance with 60, 70, 80, 100 tasks, while CPLEX was unable to find any feasible solution, employing valid inequalities (20) helped CPLEX to find feasible solutions with relatively good qualities.

5.2. Lagrangian relaxation vs. local search

DFT is a rather small terminal and Dublin port is mostly a non-transshipment port. It is not expected that the terminal authority deploys more than 6 IAVs in the initial pilot phase.

In our generated instances, 2-TEU (i.e. 1-FFE) containers comprising approximately 60% of the total discharged/uploaded containers for every vessel call. As a result, we know *a priori* that it is almost impossible that enlarging the fleet of IAVs by a factor of, say, two would reduce the makespan to $\frac{1}{2}$ of the current value. These are not proportional because of collaborations on larger than 1-TEU containers.

We use the subgradient algorithm to solve our Lagrangian relaxation-based decomposition (LRX) problem and every 10 iterations we invoke the heuristic algorithm to produce a feasible solution as a primal bound. The algorithm terminates whenever either the average subgradient size drops below a certain threshold depending on the size of instance or there was not a sufficient improvement in the last 20 iterations—even after adjusting the step size of the sub-gradient algorithm. The big-M constraints are modeled using the notion of *indicators* in CPLEX to avoid numerical instability in cases that the big-M constraints are not removed at the preprocessing phase of CPLEX.

The variable fixing procedure is invoked every five iterations, if there has been any improvement in either bounds.

In Table 4, the values are chosen as $n \in \{20,30,40,50,100,150,200,250,300,350,400\}$ and $m \in \{2,3,4,5,6\}$ and for each m there is a corresponding block in the table. For each block, we have the following structure:

The first column represents the number of machines, and the second one reports the number of tasks. The computational time spent in LR resolution is reported in the subsequent column which is followed by the column representing the number of subgradient iterations. We then report the computational time spent in the heuristic algorithm in the next column. The following column reports the gap between the best solution of heuristic, and the LR bound. The subsequent column reports the best-found makespan in its modified time unit. Finally, in the last column, we report a performance comparison against CPLEX when it is directly applied to the IAVS model.

Due to the numerical difficulties in solving (LRX-IAVS-Sch) MIP model in our Lagrangian relaxation model, the number of subgradient iterations is fairly small, except in cases where Lagrangian problem to an instance becomes rather easy. The computational times of Lagrangian relaxation ranges from roughly 35 s to about 14,902. This time also includes the time spent in the heuristic search and variable fixing procedures during the resolution process.

Yet, the computational times of heuristic are very small when compared to the CPU time spent in the overall algorithm.

The gap between the heuristic and the LR bound are practically acceptable for this application, and the quality of solutions are confirmed. The maximum average gap is 16.72%, which is practically good and acceptable.

A fair comparison of approach with a direct application of CPLEX to IAVS model (with additional constraints) would be to allow CPLEX run until it finds a solution with a similar quality (regarding GAPs) as the solution found by the LR. We also set a time limit to twice the maximum time needed for LR approach for the same $|V|$ (same block) and among all its $|I|$s. Moreover, we set a branch-and-bound node limit to 1,000,000 nodes.

Table 4
Lagrangian decomposition numerical results.

$\|V\| = m$	$\|I\| = n$	Lagrangian Relaxation		Local Search		Makespan	CPLEX	$\|V\| = m$	$\|I\| = n$	Lagrangian Relaxation		Local Search		Makespan	CPLEX
		Time (s)	# Iteration	Time (s)	GAP (%)		Time (s)			Time (s)	# Iteration	Time (s)	GAP (%)		Time (s)
2	20	208.53	81	21.19	28.72	134.40	0.00	5	20	751.29	71	13.81	2.93	99.48	0.00
	30	441.13	36	18.58	13.86	393.71	0.00		30	234.07	47	17.22	5.96	272.42	0.00
	40	132.90	49	13.14	16.50	815.27	36.20		40	185.25	33	13.82	2.50	564.10	0.92
	50	758.25	35	48.20	8.34	2409.73	35.91		50	420.68	84	48.30	28.18	1587.21	28.18
	100	820.60	46	78.97	25.54	6384.74	nodeLim		100	3812.05	37	17.00	8.24	4448.04	nodeLim
	150	35.95	59	208.29	6.59	11680.89	nodeLim		150	6303.94	14	450.62	26.58	8646.29	nodeLim
	200	95.29	77	28.47	29.00	26687.28	nodeLim		200	9316.84	46	586.58	18.31	17261.11	tiLim
	250	1053.25	42	321.45	27.05	64034.56	tiLim		250	11517.96	21	29.96	23.09	42228.12	tiLim
	300	3460.36	52	111.83	10.17	127765.75	tiLim		300	11587.74	83	515.69	0.87	89414.73	tiLim
	350	4676.58	82	525.20	0.75	227634.73	tiLim		350	901.47	81	74.06	2.34	147232.25	tiLim
	400	4988.63	53	60.46	3.01	439510.88	tiLim		400	8413.12	27	126.56	11.35	325677.56	tiLim
Avg.					15.41			Avg.					11.85		
3	20	515.30	16	39.70	15.36	123.20	0.00	6	20	145.11	14	22.99	10.53	87.88	0.00
	30	879.88	19	88.01	28.97	301.85	0.00		30	716.34	20	71.37	19.81	248.81	0.00
	40	816.33	24	18.68	9.83	625.04	45.46		40	234.54	35	32.51	5.72	515.21	159.22
	50	593.26	64	79.13	6.34	2088.44	33.61		50	1015.63	67	18.36	7.60	1449.65	112.55
	100	2596.86	41	235.69	17.26	5852.68	nodeLim		100	1262.78	89	95.09	28.76	3929.10	nodeLim
	150	3837.04	99	84.63	15.58	10707.48	nodeLim		150	456.43	48	109.62	12.46	7896.95	tiLim
	200	2087.99	97	198.90	12.27	19125.88	nodeLim		200	2706.71	64	124.87	10.64	15247.31	tiLim
	250	2135.31	46	506.60	23.82	52294.89	tiLim		250	953.77	74	72.62	15.74	38568.35	tiLim
	300	1911.92	89	528.74	13.15	110730.31	tiLim		300	7798.86	95	407.25	29.48	81665.45	tiLim
	350	5167.25	46	475.29	27.63	163138.22	tiLim		350	3987.62	36	658.41	12.00	130055.15	tiLim
	400	10404.03	41	1124.90	13.74	380909.43	tiLim		400	7273.53	41	881.28	14.02	287681.85	tiLim
Avg.					16.72		Avg.						15.16		
4	20	98.01	53	32.01	0.09	104.72	0.00								
	30	170.59	24	34.22	8.79	286.76	0.00								
	40	1263.83	35	138.95	7.94	593.79	81.27								
	50	249.93	40	150.46	25.30	1670.75	152.33								
	100	2215.72	41	238.49	11.34	4682.14	nodeLim								
	150	3515.87	42	165.68	9.55	9101.36	nodeLim								
	200	2766.83	60	186.66	9.84	18169.59	tiLim								
	250	984.18	69	54.35	15.73	49680.15	tiLim								
	300	9476.48	67	192.11	1.39	94120.77	tiLim								
	350	7519.82	36	836.61	25.16	154981.31	tiLim								
	400	14902.79	46	1184.40	6.35	361863.96	tiLim								
Avg.					11.04										

The numerical values of this column represent the time needed by CPLEX to find a similar solution quality. Although the results can be instance-dependent, however, starting from $|I| = 100$ for all $|V|$, CPLEX always runs either into node limit or time limit without being able to obtain a solution with a similar quality as the one found by our LR approach. This is due to the fact that the LP is still easy for $|I| = 100$ and $|I| = 200$ and allows many branch-and-bound nodes to be processed. But, even that becomes challenging in larger $|I|$.

The results also confirm that: while the increase in the makespan is not linearly proportional to the number of tasks, decrease in the length of makespan is not proportional to the size of the fleet of IAVs, either. This is justified by the fact that most of the discharged containers actually lead to cooperation between IAVs, which in turn results in a significant waiting time spent by IAVs awaiting each other to start a task.

5.3. Comparison with the current practice of port

The present practice in the port is comprised of a fleet of man-driven vehicles (called *shunters*) each of which transports a 1-TEU or 1-FFE container. A precise performance comparison between the on-going practice and the IAV-based logistics is not very straightforward as a fair comparison is rather difficult. That is, comparing output of a deterministic model against the real practice, including several sources of uncertainties, failure, etc. might not be fair.

In this work we do not compare IAVs against AGVs due to two reasons: (1) AGVs are not currently in service at DFT, and therefore, we do not have any relevant statistics. (2) The main issue of DFT (and those port of North-West Europe involved in InTraDE) is the shortage of space for the manoeuvre of 40-ft transporters (including 40-ft AGVs) and their resulting bottleneck in traffic flow, which demands an alternative solution such as IAVs with pairing/unpairing capability.

A generic simulation platform has been used to develop a model and simulate the output of our optimization algorithm in terms of the sequence of jobs on every IAV as well as the pairing/unpairing plan reported in the output.

FlexSim[2] has been chosen as an extendible platform to develop our simulation model.[3]

DFT has two berths (see Fig. 6), Eastern berth (180 m long) and Southern berth (300 m long). Three rail-mounted quay cranes are shared between the two berths. The six stacks closest to the quay side are the export stacks, and those six farthest ones are the import stacks (shown in Fig. 6).

At the Northwest corner (next to the Eastern berth) of DFT layout, there is a rectangular area reserved for empty container stacking or a place for temporary stacking of containers for special operations. This is beyond the scope of this work.

The service rate of the quay cranes is approximately 70–90 unit of time for a cycle of (1) collection of a container, (2) transfer between vessel and IAVs, and (3) release of the container.

Every call discharges roughly between 300–500 containers and loads almost 80% of the discharged quantity. Moreover, almost 60% of the total number of handled containers in this terminal are 40-ft (1-FFE) containers and very few 45-ft ones (can be safely ignored).

As the current practice makes use of man-driven shunters as transport means at DFT and is not bound to follow any special guiding signs on the ground (except that the drivers have to obey some generic traffic rules), the real distance table can be roughly approximated by the network distance. The network distances are calculated based on a discretization we have made. The same distances are used in the developed routing engine of IAVs in a software package called `InTraDE-IAV-Router` (see Gelareh et al. (2012a) for further details). Fig. 7 illustrates the network structure of DFT based on the discretization made by `InTraDE-IAV-Router`. The nodes of this network are chosen to be the center of gravity of every square in the figure. The dark cells represent the blocked nodes and edges representing non-driving areas.

Further details of the simulation study can be found in Gelareh et al. (2012a).

The following scenarios were considered based on an analysis of data, which we have collected, and the observations from the current practice. For every scenario, an output of the optimization algorithm (the profile of loading/discharging, job sequences of IAVs, etc.) is imported into the simulation model. Doing so helps to observe the system behavior which is expected to be closer to the reality than the deterministic optimization output:

Scen. (i)	Only Eastern berth is engaged with the loading/unloading, and it uses only one quay crane (low season).
Scen. (ii)	Only Eastern berth is engaged with the loading/unloading, and it uses two quay cranes.
Scen. (iii)	Only Southern berth is engaged with the loading/unloading, and it uses two quay cranes.
Scen. (iv)	Only Southern berth is engaged with the loading/unloading, and it uses three quay cranes.
Scen. (v)	Both Easter and Southern berths are engaged with the loading/unloading while the southern one uses two quay cranes and the Eastern one uses one (high/pick season).

[2] FlexSim is a discrete-event manufacturing simulation software developed by FlexSim Software Products, Inc.

[3] Although, *FlexSim CT*, as a separate platform for simulation of container terminals is already available, however, due to the complexity of operations (such as management and synchronization needed to control the whole system) the software producer has decided to trade flexibility for simplicity and offers fewer possibilities to the users in order to extend FlexSim CT, compared to what has been offered in FlexSim (of course, at the time this article is being written). Therefore, we were not able to model IAVs collaborative manner in FlexSim CT and we used FlexSim instead.

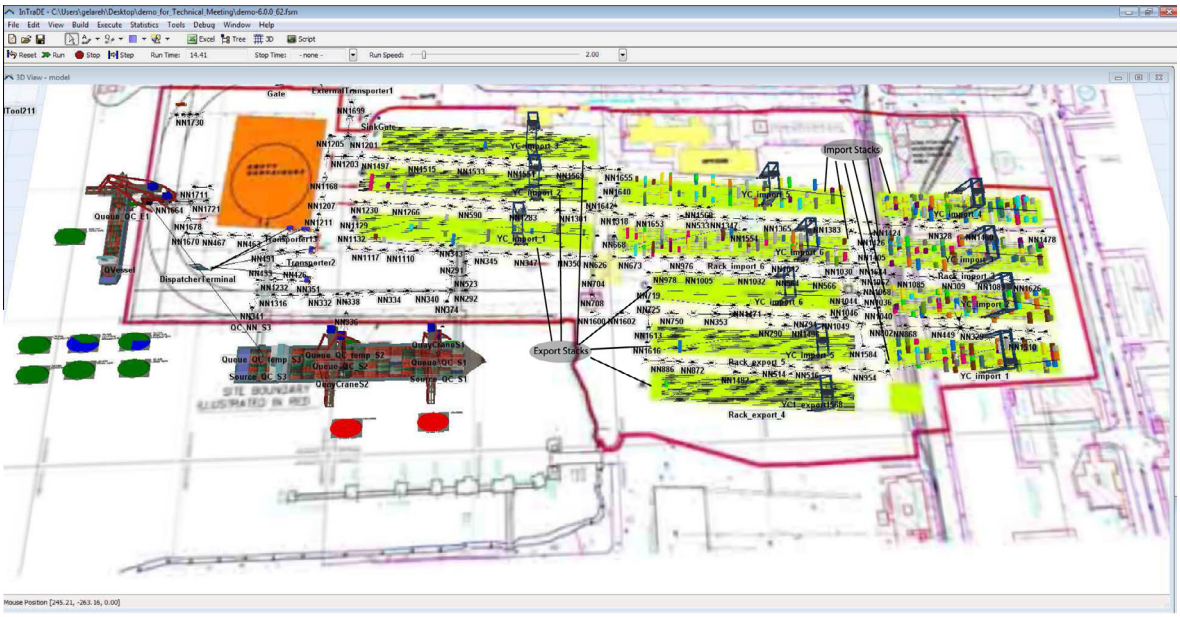


Fig. 6. Simulation model of Dublin Ferryport Terminal (DFT).

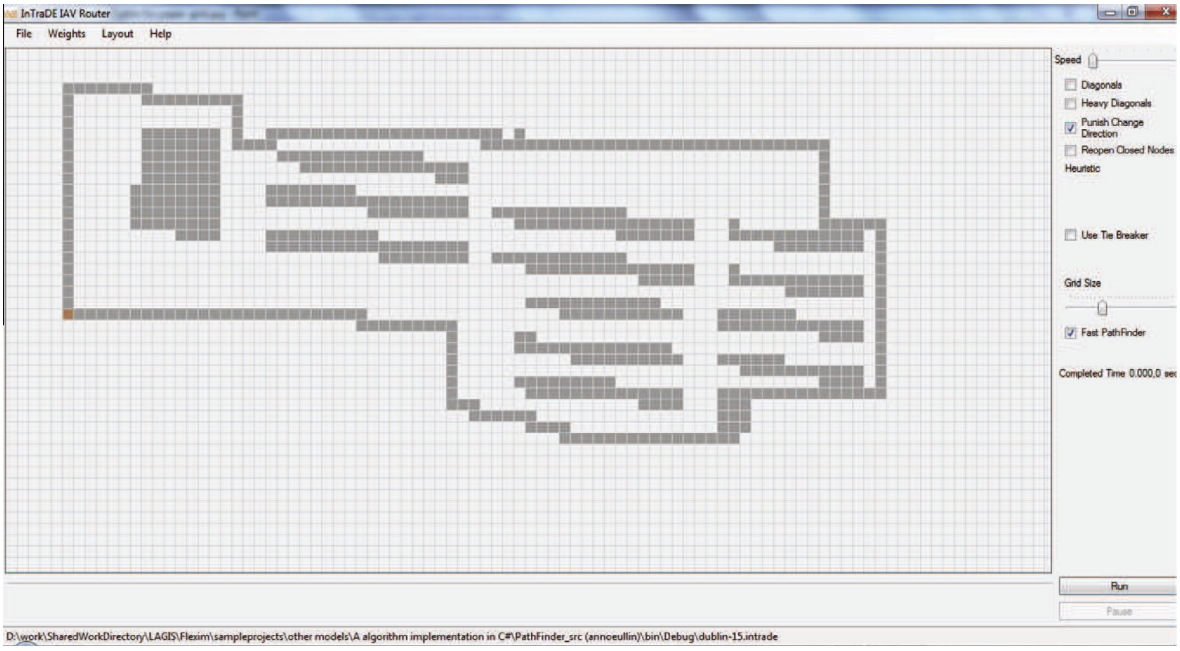


Fig. 7. Discretization—the network structure of DFT layout.

It must be emphasized that special attention has been paid to capture and incorporate, as much as possible, the real-life features and uncertainty involved in the system. This includes those related to the service time of cranes (yard and quay), traffic congestions, etc.

In order to make a fair comparison, we recorded from the real practice the times for completion of $n_c \in \{20,30,50,100,150,200,300,400\}$ containers. Such a measured time includes also the marginal time which may be caused by all the uncertainties, disturbances, etc. that may occur in reality. We then set the parameters of the simulation model and run the simulation model of IAVs for the same profile of containers (same origin/destinations, almost same service rate of equipment, etc.) as the output of our optimization algorithm. The fleet of IAVs, which we have used had the same capacity as that of existing shunters at the port. We then run the simulation and recorded the time for completion of all

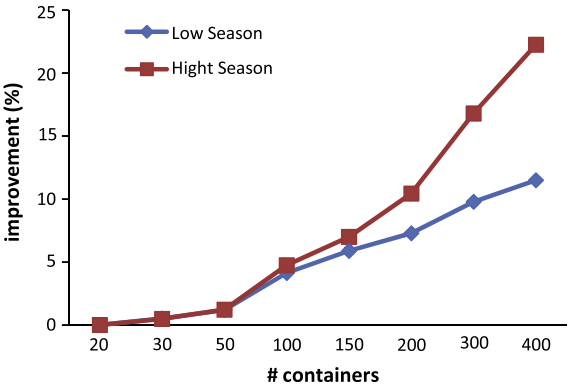
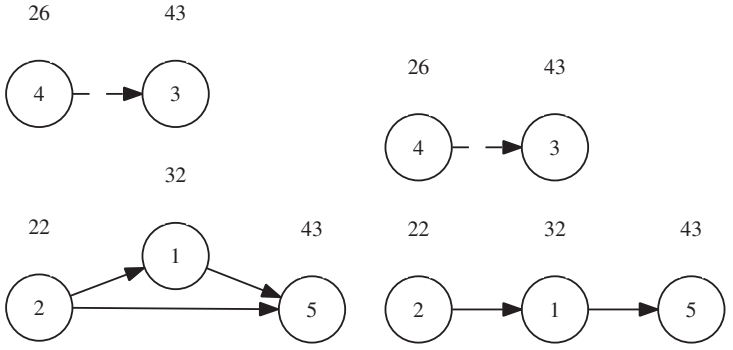


Fig. 8. Improvement over current practice in *low* and *high* season.



(a) x_{ijm}: task j is performed after i on vehicle m.

(b) x_{ijm}: task j is performed *immediately* after i on vehicle m.

Fig. 9. The graph representation of different variable definitions.

possible values of n_c containers from the whole load. The resulting time also includes the time needed for pairing/unpairing of IAVs. This has been carried out for all the aforementioned scenarios.

The results indicate that starting from $n_c = 50$ the time needed to perform n_c tasks using IAVs becomes smaller than that of the current practice. In Fig. 8, we report the improvement over the current practice of DFT for the low and high seasons. For the low season, when only the Eastern berth is busy, the improvement over the time needed to complete operations starts from 0.01% for the first 20 containers and increased to 11.5% after 400 containers. In the high season, when all three quay cranes and two berths are serving vessels, the current practice of DFT suffers from a space problem encountered when deploying too many 40-ft (45-ft) trucks. However, when using IAVs, the simulation results show an improvement starting from 0.01% for the first 20 containers and increases to 22.26% after completing all 400 containers.

6. Summary, conclusion and outlook to future works

The autonomy and intelligence of IAVs promotes their application in different environments—container terminals as the most important ones.

IAVs, developed in the framework of InTraDE, are expected to be deployed in some of the small container terminals in North-West Europe. These container terminals are bound to very confined spaces and facing an ever increasing volume of containerized trade while do not have the possibility for further expansions (even not enough space for installing space-demanding recent technological port facilities). The space issues are addressed through deploying IAVs of 1-TEU capacity, which can dynamically expand their capacity up to 40-ft through collaborations among them. This eliminates the need for deploying too many space-demanding large 40-ft transporters.

We reviewed and extended the model in Ng et al. (2007) to accommodate cases where a sequence of containers (including 1-TEU and also larger than 1-TEU) needs to be transported using a limited-size fleet of IAVs. Such a collaboration is realized by dynamic pairing/unpairing of IAVs throughout the planning horizon in order to prepare sufficient capacity for transporting containers of larger than 1-TEU. Once an IAV arrives at a crane to pick up a container of larger than 1-TEU, it has to wait for another partner IAV to arrive, pair and do the pick-up and transport jointly. A given IAV may collaborate with

another IAV on a container during a planning period. However, such a collaboration does not guarantee that the same IAVs which have collaborated earlier on a container will pair again to serve transporting another container in the same planning period.

We then developed a Lagrangian relaxation-based decomposition to solve instances of the case of Dublin Ferryport Terminal and simulated the output of Lagrangian algorithm within a discrete-event simulation model of DFT. The numerical results confirm the efficiency of our solution method and quality of solutions. From a practical point of view, a considerable improvement has been observed (from the simulation model) on using IAVs instead of the current means of transport at DFT.

IAVs work in three modes: (1) fully-functional, (2) degraded and (3) faulty modes. While in the fully-functional mode, an IAV works efficiently, in the degraded mode, perhaps part of the system (the system is comprised of 4 independent wheels and several independent sensors) has encountered a failure, but the system is still able to complete the jobs with less performance. In the third case, the IAV has to declare a break-down and stop operating. In the future, we will take into account these features and extend the model to accommodate this case in a stochastic modeling framework. Of course, studies on improving the mathematical model is of high importance and also developing effective solution approaches such as exact decomposition methods deserve particular attention.

Acknowledgements

This work has been supported by the European Union fund in the framework of Intelligent Transportation for Dynamic Environment (InTraDE) project within Interreg IVB NWE funding program. The authors thank DFT management for their kind cooperation and support by sharing relevant data. The authors would also like to thank anonymous reviewers for their constructive comments.

Appendix A. A short note on mathematical model of Ng et al. (2007)

The definition of x_{ijm} is confusing. In Fig. 9a and b we depict an example with 5 tasks and 2 two vehicles (distinguished by connection style)—excluding the dummy source and sink nodes in this representation. Depending on a precise definition of x_{ijm}, only one of Fig. 9a and b is expected. The issue arises in the definition of x variables. One of the following definitions should be the correct one for x_{ijm}:

- *The* task j is performed after the task i on the vehicle m: in which case $x_{ij'm} = 1$ for all tasks $j' \neq i$ which are completed after completion of i on the same vehicle m. Therefore the total number of links arriving at any non-first task is more than one and this contradicts with (5). This is depicted in Fig. 9a (machine 2, with solid connection between tasks) where $x_{212} = x_{152} = x_{252} = 1$. In such a case we have $y_{12} = y_{22} = y_{52} = 1$ and consequently the corresponding constraint (5) is violated by $2 = \sum_{i=1}^{N} x_{ijm} \leqslant y_{jm} = 1$ for $j = 5$, $m = 2$.
- *The* task j is performed immediately after the task i on the vehicle m: in which case $x_{ij'm} = 0$ for all tasks $j' \neq j$ which are completed after completion of i on the same vehicle m. There is no link between two non-consecutive tasks on the same machine which contradicts with (6). This is depicted in Fig. 9b (machine 2, with solid connection between tasks) where only $x_{212} = x_{152} = 1$. In such a case we have $y_{12} = y_{22} = y_{52} = 1$ and consequently the corresponding constraint (6) is violated by $0 = x_{252} + x_{522} \geqslant y_{22} + y_{52} - 1 = 1$.

Under of the aforementioned definitions of x_{ijm}, the model seems being infeasible. However, it is still possible to correct it for use in planning AGVs and also extend it for accommodating joint operations, of handling e.g. an FFE, by pairing IAVs when $\exists i: S(i) > 1$.

References

Bahiense, L., Maculan, N., Sagastizábal, C.A., 2002. The volume algorithm revisited: relation with bundle methods. Mathematical Programming 94 (1), 41–69.
Barahona, F., Anbil, R., 2000. The volume algorithm: producing primal solutions with a subgradient method. Mathematical Programming 87 (3), 385–399.
Bish, E., 2003. A multiple-crane-constrained scheduling problem in a container terminal. European Journal of Operational Research 144 (1), 83–107.
Bish, E., Chen, F., Leong, Y., Nelson, B., Ng, J., Simchi-Levi, D., 2005. Dispatching vehicles in a mega container terminal. OR Spectrum 27 (4), 491–506.
Bish, E., Leong, T., Li, C., Ng, J., Simchi-Levi, D., 2001. Analysis of a new vehicle scheduling and location problem. Naval Research Logistics (NRL) 48 (5), 363–385.
Cao, J., Lee, D., Chen, J., Shi, Q., 2010. The integrated yard truck and yard crane scheduling problem: Benders' decomposition-based methods. Transportation Research Part E: Logistics and Transportation Review 46 (3), 344–353.
Chen, L., Bostel, N., Dejax, P., Cai, J., Xi, L., 2007. A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal. European Journal of Operational Research 181 (1), 40–58.
Fisher, M.L., 1981. The lagrangian relaxation method for solving integer programming problems. Management Science 27 (1), 1–18.
Fisher, M.L., 2004. The lagrangian relaxation method for solving integer programming problems. Management Science 50 (12), 1861–1871.
Gelareh, S., Lavrov, A., Merzouki, R., 2012a. Simulation Platform for Control Strategy Tuning of Intelligent and Autonomous Vehicles. Tech. Rep., LAGIS, Polytech'Lille.
Gelareh, S., Maculan, N., Mahey, P., Monemi, R.N., 2012b. Hub-and-spoke network design and fleet deployment for string planning of liner shipping. Applied Mathematical Modelling 37 (5), 3307–3321.
Grunow, M., Günther, H., Lehmann, M., 2004. Dispatching multi-load AGVs in highly automated seaport container terminals. OR Spectrum 26 (2), 211–235.

Grunow, M., Günther, H., Lehmann, M., 2007. Strategies for dispatching AGVs at automated seaport container terminals. Container Terminals and Cargo Systems, 155–178.
Guignard, M., 2003. Lagrangian relaxation. TOP 11 (2), 151–228.
Hartmann, S., 2004. A general framework for scheduling equipment and manpower at container terminals. OR Spectrum 26 (1), 51–74.
Held, M., Karp, R., 1971. The traveling-salesman problem and minimum spanning trees: Part II. Mathematical Programming 1 (1), 6–25.
Kim, K., Bae, J., 1999. A dispatching method for automated guided vehicles to minimize delays of containership operations. International Journal of Management Science 5 (1), 1–25.
Kim, K., Bae, J., 2004. A look-ahead dispatching method for automated guided vehicles in automated port container terminals. Transportation Science 38 (2), 224.
Larsson, T., Patriksson, M., Strömberg, A., 1999. Ergodic, primal convergence in dual subgradient schemes for convex programming. Mathematical Programming 86 (2), 283–312.
Lee, L., Chew, E., Tan, K., Wang, Y., 2010. Vehicle dispatching algorithms for container transshipment hubs. OR Spectrum 32 (3), 663–685.
Lemarechal, C., 1975. An extension of Davidon methods to non differentiable problems. Nondifferentiable Optimization, 95–109.
Meersmans, P., Wagelmans, A., 2001a. Dynamic Scheduling of Handling Equipment at Automated Container Terminals. ERIM Report Series Reference No. ERS-2001-69-LIS.
Meersmans, P., Wagelmans, A., 2001b. Effective Algorithms for Integrated Scheduling of Handling Equipment at Automated Container Terminals. Erasmus Research Institute of Management, Erasmus Universiteit.
Michelon, P., Maculan, N., 1991. Lagrangian decomposition for integer nonlinear programming with linear constraints. Mathematical Programming 52 (1), 303–313.
Narasimhan, A., Palekar, U.S., 2002. Analysis and algorithms for the transtainer routing problem in container port operations. Transportation Science 36 (1), 63–78.
Ng, W., Mak, K., Zhang, Y., 2007. Scheduling trucks in container terminals using a genetic algorithm. Engineering Optimization 39 (1), 33–47.
Nguyen, V., Kim, K., 2009. A dispatching method for automated lifting vehicles in automated port container terminals. Computers & Industrial Engineering 56 (3), 1002–1020.
Reinoso, H., Maculan, N., 1992. Lagrangian decomposition in integer linear programming: a new scheme. In: INFOR, vol. 30.