

Determination of the number of AGVs required at a semi-automated container terminal

Iris F.A. Vis, René de Koster, Kees Jan Roodbergen, Leon W.P. Peeters
Rotterdam School of Management, Erasmus University Rotterdam,
P.O. Box 1738, 3000 DR Rotterdam, The Netherlands

October, 2000

Abstract

This paper describes the development of a minimum flow algorithm to determine the number of Automated Guided Vehicles (AGVs) required at a semi-automated container terminal. At such a terminal the containers are transported by AGVs from the Quay Cranes to the Automated Stacking Cranes and vice versa. A model and a strongly polynomial time algorithm are developed to solve the case in which containers are available for transport at known time instants.

Keywords: Logistics; Container Terminal; AGVs; Networks and graphs;

Introduction

Containerised transportation was introduced in the early 1960s. The first-generation ships had a capacity of about 400 TEU (Twenty-foot-Equivalent Unit, length of container is twenty feet). The steadily increasing proportion of cargo handled with containers has resulted in the use of much larger ships (capable of carrying 4000 TEU and more). In addition, there has been an increase in the importance of marine transport systems, including networks of ports and terminals. These ports and terminals are used for the transshipment of containers from a ship to other modes of transportation, such as barges, trains and trucks, and vice versa.

To satisfy the demands of customers, such as short berth times of the ship, short lead times and high quality products, all operations must now be carried out quickly and efficiently. Furthermore, the strong competition between terminals means that it is essential to reduce the costs. An efficient terminal must therefore ensure that container ships are unloaded and loaded quickly. The introduction of much larger ships and the requirements for shorter ship handling times have made it necessary to develop new techniques and methods which allow effective co-ordination of all activities at a terminal.

Most of the terminals use manned equipment. However, a few terminals, such as the port of Rotterdam, are semi-automated. The process of unloading and loading a ship at a semi-automated terminal is illustrated in Figure 1 and may be described as follows: when a ship arrives at the port, the containers have to be taken off the ship. This is done by manned Quay Cranes (QCs), which take the containers from the ship's hold and the deck. Next, the QCs put the containers on Automated Guided Vehicles (AGVs), robotic vehicles which travel along a predefined path. After receiving the container, the AGV moves to the stack. This stack consists of a number of lanes where containers can be stored for a certain period. These lanes are served by automatically controlled Automated Stacking Cranes (ASCs). When an AGV arrives at a lane, the ASC takes the container off the AGV and stores it in the stack. After a certain period the containers are retrieved from the stack by the ASCs and transported by the AGVs to transportation modes such as barges, deep-sea ships, trucks or trains. This process is also executed in reverse order, to load a ship. As can be seen in Figure 1, the stack separates deep-sea loading and unloading operations from land-side (including short-sea and barges) loading and

unloading operations. AGVs serve either at the deep-sea side or at the land-side. The most critical problems arise at the deep-sea side.

INSERT FIGURE 1

In designing a semi-automated container terminal, a number of problems have to be solved. These include the development of adequate planning and control concepts for the various systems in the terminal. The planning and control concepts have a strong impact on the performance that can be reached, e.g. throughput time per ship and response time of individual containers. As a result, research in this area is highly relevant for the design of container terminals and from a wider point of view for meeting customer and shipowners demand. Furthermore, it is possible to alter or generalise the planning and control concepts for container terminals, such that they can be used in other environments like warehouses and manufacturing systems.

As described above, the containers have to be transported from the stack to the ship and vice versa by AGVs. To ensure that this operation is carried out quickly and efficiently, one of the problems that has to be solved is the determination of the necessary number of vehicles to transport all containers. This paper describes a model and algorithm to solve this problem. The paper is structured as follows: first an overview of literature is given. Furthermore, a model and algorithm for solving the problem are presented. Subsequently, the model and algorithm are illustrated with an example. Finally, conclusions and suggestions for further research are given.

Literature

At a semi-automated container terminal AGVs are used for the internal transportation of containers from ship to stack and vice versa. The routing and control of vehicles in a container terminal is studied in several papers. In Evers and Koppers¹ the traffic control of large numbers of AGVs is studied. A formal tool to describe traffic infrastructure and its control is developed by using four types of entities: node, track, area and semaphore (i.e. a non-negative integer variable which can be interpreted as free capacity). The tool is evaluated with simulation. It can be concluded that the technique is a powerful tool for modelling transportation infrastructure and its control. Also, the performance and the capacity of the area increases. A complete review of routing of vehicles is given in Bodin et al.². Steenken³ and Steenken et al.⁴ describe the more specific problem of the routing of straddle carriers at a container terminal. The objective is to minimise empty-travel distances by combining unloading and loading jobs. Routing and scheduling systems are tested and integrated into a radio data transmission system of a real terminal. Steenken³ obtains savings of 13% in empty drives compared with the previously existing situation at the terminal, by solving the problem as a linear assignment problem. Steenken et al.⁴ solve the problem by formulating it as a network problem with minimum costs. Savings of 20-35% in empty-travel distances can be obtained within quite acceptable computation times.

During operation, containers have to be assigned to vehicles. This problem is studied in several papers. Kim and Bae⁵ suggest mixed integer linear programming formulations and give a heuristic method for the dispatching of containers to AGVs such that the delay of the ship is minimised. As a secondary objective, the minimisation of the total travel time of AGVs is considered. The problem of dispatching AGVs to containers is also studied in Chen et al.⁶ and Demir et al.⁷. To obtain an efficient rule for the dispatching of AGVs to containers one of the most important requirements is to know the exact number of AGVs in the container terminal. Consequently, to obtain an AGV control concept that yields satisfactory performance, one of the problems that must be solved is the determination of the number of vehicles required to transport all containers within time. This problem will be studied in this paper.

In other environments like warehouses and manufacturing systems, problems with a similar structure have been studied. The deterministic case of the determination of the minimum number of vehicles is studied in Maxwell and Muckstadt⁸. They use an integer programming

formulation for seeking the optimum solution. According to Ilić⁹, the total number of vehicles in simple systems can be estimated by the number of roundtrips that each vehicle can make per hour and the total number of roundtrips. In stochastic cases the problem can be solved by using queueing networks. For example, Mantel and Landeweerd¹⁰ use a hierarchical queueing network approach to determine the number of AGVs. The problem can also be studied with the use of simulation. Models of real systems are designed and experiments are performed with these models to gain an understanding of the behaviour of Automated Guided Vehicle systems. Gobal and Kasilingam¹¹ present a simulation model that gives an estimation of the number of AGVs needed to meet material handling requirements. This estimate is based on the sum of the idle times of vehicles and machines and waiting times of the transported parts.

Problems, that have an analogy to the determination of the number of AGVs required at a terminal, are e.g. the tanker scheduling problem and the scheduling of workers to different tasks. One of the first articles to describe a deterministic model to solve the tanker scheduling problem is Dantzig and Fulkerson¹². The problem of determining the minimum number of oil tankers required to meet a fixed transportation schedule is formulated as a linear programming problem and solved with the simplex algorithm. The same tanker scheduling problem is discussed by Ahuja, Magnanti and Orlin¹³, but they suggest a different approach, namely that the problem can be solved by constructing a network and using any maximum flow algorithm. They also give an overview of different maximum flow algorithms. In general, every maximum flow problem can be formulated as a minimum cost flow problem. Another way of solving the problem of the determination of the number of vehicles, is therefore to formulate it as a minimum cost flow problem and to use any minimum cost flow algorithm. An overview of minimum cost flow algorithms is given in Ahuja, Magnanti and Orlin¹³.

A different approach, with the same time complexity as the method suggested in Ahuja, Magnanti and Orlin¹³ is to use bipartite networks. This approach is used in Phillips and Garsia-Diaz¹⁴ to determine the minimum number of workers necessary to accomplish a fixed schedule of tasks. The maximum flow in this bipartite network indicates pairs of tasks that should be performed by the same individual. To obtain the complete list of jobs for each individual, the arcs in the maximum flow have to be searched. Another method to solve the problem of the determination of the minimum number of individuals to meet a fixed schedule of tasks is given in Ford and Fulkerson¹⁵. The tasks can be partially ordered as follows: job i precedes job j if the start time of i is earlier than the start time of job j and if the jobs can be executed by the same individual. An ordered chain therefore represents a possible assignment of jobs to one individual. A minimum chain decomposition consequently represents the minimum number of individuals required.

While these authors look at the finite horizon version of the problem, Orlin¹⁶ considers the case where a finite number of tasks are executed periodically over an infinite horizon. The problem is solved efficiently as a finite network problem.

Model and Algorithm

In this section, a model and algorithm are developed to solve one of the problems that arises at the tactical level, namely the determination of the number of AGVs required to transport all containers within time. The model and algorithm are capable of handling large data sets. The results from the tactical level serve as input for the operational level. During operation, it has to be decided which AGV transports which container along which route at which time instant. The possibility of adjusting the number of vehicles to accommodate unexpected events remains open. The required number of AGVs can be estimated by means of a simplified model in which containers are available for transport at known time instants. By varying these time instants, robust estimates can be made of the necessary number of vehicles.

The terminal which we consider in this paper is comparable to the semi-automated terminal described in the introduction. Containers are available at the ASC or QC for transport by AGVs at known time instants (release instants). We will from now on refer to containers as jobs. An unloading job is transported from the QC to the ASC and a loading job from the ASC to the QC. A loading job at the ASC is a job that is taken from the stack by the ASC and placed on an AGV. An unloading job at the ASC is taken from an AGV and placed in the stack by the ASC.

As mentioned in the previous section there are several possibilities for solving the problem of the determination of the minimum number of vehicles required. In this paper, we choose to follow the approach of Ahuja, Magnanti and Orlin¹³. They discussed a similar problem and suggested that it can be solved by constructing a network and using any maximum flow algorithm. However, they only modelled the network and did not explicitly solve the problem. In this section, we explicitly formulate the network model for this specific situation and also construct a solution algorithm. A transformation of the graph is executed on the basis of a feasible flow in the original network. A maximum flow is thereafter determined in the transformed network. The values originating from the feasible and the maximum flow are used to determine a minimum flow through the original graph. The value of this minimum flow corresponds to the minimum number of AGVs needed at the terminal.

Model

A model is developed for a semi-automated terminal. We make the following assumptions:

1. There are N jobs to be executed. The jobs are divided into groups: loading jobs and unloading jobs. For each of the jobs the availability instant is known.
2. Jobs at QCs and ASCs are known in advance and are regarded as input. Because there is no interference at either the QCs or the ASCs, we aggregate all QCs to one QC and all ASCs to one ASC. As a result, all unloading jobs are released at one QC and all loading jobs are released at one ASC.
3. The waiting time of jobs at the QC and the ASC should be zero. Jobs that are ready for transportation therefore have to be transported immediately. This assumption is quite realistic, since the waiting times of QCs and ASCs are much more expensive than the waiting times of AGVs.
4. The release instant of a job is the time instant at which the job arrives at the 'pickup and delivery point' of the ASC or QC (p&d point) and is ready for transportation. That is, we assume that there is no interference at ASCs or QCs between loading and unloading jobs. It is therefore not necessary to define the handling time of the job at the ASC or QC (i.e. the time required to take the container from the stack or ship and bring it to the p&d point). This time is incorporated in the release instant of the job.
5. The arrival instant of a job is the time at which the job arrives at the p&d point of the ASC or QC. The ASC or QC then has to store the container in the stack or ship and return to the p&d point. The corresponding handling time of the (un-)loading job is known in advance.
6. The capacity of an AGV is one container.

For every job the following is known:

s_i release instant, the point in time at which job i ($1 \leq i \leq N$) is available at the ASC or QC for transport by an AGV.

w_i travel time of a full AGV from the origin of job i ($1 \leq i \leq N$) to the destination of job i .

v_i arrival instant, the point in time at which job i ($1 \leq i \leq N$) arrives at the destination p&d point at the ASC or QC. This instant depends on the travel time of the full AGV and the release instant of the job. Consequently

$$v_i = s_i + w_i \quad (1)$$

r_{ij} travel time of an empty AGV from the destination of job i to the origin (pick-up location) of job j , with $s_j > s_i$ ($r_{ij} = 0$ if the destination of job i equals the origin of job j).

The travel times are assumed to be deterministic. This is because delays at the QC or ASC are not allowed. If we want to be sure that the AGV is on time at the ASC or QC (to transport the next job), we should take into account the worst case, namely the longest travel time. To obtain this deterministic value we can take the upper bound of a stochastic distribution representing the travel time.

As mentioned above, it takes the ASC or QC a certain amount of time to place the container at the right place in the stack or ship and return to the p&d point. This time is defined as follows:

b_i handling time of job i ($1 \leq i \leq N$) at the destination ASC or QC.

As described in the introduction the container is taken off the AGV by a free ASC or QC. An AGV can obviously only start a new job if it is empty, so any previous container must first be removed. If the ASC or QC is handling another job, a loaded AGV has to wait for the ASC or QC before it can leave for a new job. Consequently, the point in time (t_i , delivery instant) at which job i can be taken off the AGV by the ASC or QC depends on the handling time and delivery instant of the previous job at the ASC or QC. The delivery instant t_i is therefore determined as follows :

$$t_i = \max(v_i, t_j + b_j), \text{ where job } j \text{ is the previous job at the ASC or QC.} \quad (2)$$

Equation (2) can be understood as follows. Job j is the previous job at the crane and has arrived earlier than job i ($v_j \leq v_i$). The order in which jobs are handled by the crane is the same as the order in which jobs arrive at the crane. Job j will therefore always be handled before job i ($t_i \geq t_j$). If the point in time at which job j was delivered at the ASC(QC) plus the handling time of job j at the ASC(QC) is later than the point in time at which an AGV arrives with container i at the ASC(QC), then the AGV has to wait at the ASC(QC) until the ASC(QC) is free. In this case, the point in time at which container i is delivered therefore equals the finish instant of job j at the ASC(QC).

Summarizing, the release instants s_i of the containers as well as the travel times w_i and r_{ij} and the handling times b_i are known. Consequently, all values of v_i and t_i become known by applying equations (1) and (2). Note the difference between v_i and t_i . v_i represents the arrival of job i at the crane. t_i represents the time at which job i is taken off the AGV.

The network for our problem is constructed as follows: every job is represented as a node, and there is a directed arc from node i to node j with capacity one if one AGV can execute

both jobs. In this case we call a set of jobs (i, j) compatible. Consequently, there is no directed arc (j, i) , because job j can be executed after job i but by definition job i cannot be executed after job j . If job j can be transported after job i , the release instant of job j is later than the release instant of job i because jobs are not allowed to wait for transport. Consequently, job j is released after job i and cannot be transported first. Since jobs are not allowed to wait for pick-up, jobs (i, j) are compatible if:

$$t_i + r_{ij} \leq s_j \quad (3)$$

Using the above data, we construct a directed network as follows:

Directed Graph G :

$$\begin{aligned} V &= \{1, \dots, N\}, \text{ where every node represents a job.} \\ A &= \{(i, j) : i, j \in V, \text{ jobs } i \text{ and } j \text{ are compatible}\} \\ &\quad \text{the capacity on all arcs equals one} \end{aligned}$$

We further introduce a source node s and sink node t and add all arcs (s, i) and (i, t) with capacity 1. This is because the maximum flow algorithm, that we will use in the next section, seeks a feasible solution that sends the maximum amount of flow from s to t . Such a solution (flow y_{ij}) would have to satisfy the following conditions:

•

$$y_{ij} = \begin{cases} 1 & \text{if job } i \text{ and job } j \text{ are compatible and arc } (i, j) \text{ is used in the maximum flow} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

• For source node s and sink node t :

$$\begin{cases} \sum_j y_{sj} = \sum_j y_{jt} \\ \sum_j y_{js} = 0 \\ \sum_j y_{tj} = 0 \end{cases} \quad (5)$$

• For the other nodes:

$$\sum_j y_{ij} = \sum_j y_{ji} = 1: \text{ incoming flow equals outgoing flow} \quad (6)$$

Note that in this network a flow from the source to the sink represents a sequence of jobs that can be executed by a single vehicle.

An example of constructing such a directed graph is given in the section with the illustrative example (Figure 2).

The constructed network makes it possible to solve the problem of determining the minimum number of AGVs required at a semi-automated container terminal. In this network a directed path corresponds with a feasible sequence of jobs which can be executed by one AGV. The aim is to determine the minimum number of directed paths, which corresponds with the minimum number of AGVs, such that each node in the network is included in exactly one path.

To solve the problem, the network (graph G) is transformed into G_1 as follows:

- split every node i (except source s and sink t) into two nodes i' (origin) and i'' (destination) and add the arc (i', i'') . As a result, we obtain $V_1 = \{s, 1', 1'', \dots, N', N'', t\}$ and $A_1 = \{(i'', j') : i'', j' \in V_1, \text{ jobs } i \text{ and } j \text{ are compatible}\} \cup \{(i', i'') : i', i'' \in V_1\} \cup \{(s, i') : s, i' \in V_1\} \cup \{(j'', t) : j'', t \in V_1\}$.
- the lower bound for the flow on each arc $(i', i'') \forall i \in V$ is set at 1

The purpose of these two steps is to replace the capacity restrictions on the nodes with a capacity restriction on the connecting arc. One unit of flow obviously has to pass through this arc, since every job has to be executed. Consequently every node will be visited.

After this transformation each directed path from s to t corresponds to a sequence of jobs that can be executed by one AGV. Consequently, a flow of value v (v directed paths each with a flow of one) corresponds with schedules for v AGVs. Hence, the problem is reduced to finding a feasible flow with minimum value.

A minimum flow algorithm to solve the model

To solve the problem of finding a feasible flow with minimum value in graph G_1 , we introduce the following algorithm:

Step 1

- Determine a feasible flow x through G_1 . Without computation we can obtain a feasible flow in G_1 as follows:
- Each node is included in a different path from s to t . This corresponds with the following flow x_{ij} :

$$\begin{aligned}
 x_{si'} &= 1 & \forall (s, i') \in A_1 \\
 x_{i''t} &= 1 & \forall (i'', t) \in A_1 \\
 x_{i''j'} &= 0 & \forall (i'', j') \in A_1 \\
 x_{i'i''} &= 1 & \forall (i', i'') \in A_1
 \end{aligned} \tag{7}$$

Note that this flow is actually a maximum flow in G_1 .

Step 2

Assign the triple (l_{ij}, x_{ij}, u_{ij}) to every arc.

l_{ij} equals the lower bound of the capacity ($l_{ij} = 0 \forall (i, j) \in A_1 \setminus \{(i', i'')\}$ and $l_{ij} = 1$ otherwise)

u_{ij} equals the upper bound of the capacity ($u_{ij} = 1 \forall (i, j) \in A_1$)

x_{ij} corresponds with the value of the flow as given by equation (7).

Step 3

Construct the graph G_2 as follows:

$$A_2 = A_1 + \{(j, i) : (i, j) \in A_1\}$$

$$V_2 = V_1$$

A_2 consists of all arcs in A_1 plus a backward directed arc for each arc $(i, j) \in A_1$.

- The upper bound of the capacity of the forward arcs (i, j) in G_2 is:

$$\Delta_{ij} = x_{ij} - l_{ij} \tag{8}$$

- The upper bound of the capacity of the backward arcs (j, i) in G_2 is:

$$\Delta_{ji} = u_{ij} - x_{ij} \quad (9)$$

Step 4

Determine a maximum flow x' through network G_2 by using any maximum flow algorithm (for an overview see Ahuja, Magnanti and Orlin¹³).

Step 5

Define the flow x^* as follows:

$$x_{ij}^* = x_{ij} - x'_{ij} + x'_{ji} \quad \forall (i, j) \in A_1 \quad (10)$$

Step 6

The flow x^* from step 5 is the flow with minimum value in graph G_1 .

The idea behind the algorithm is that the feasible flow from step 1 can be reduced to a minimum flow. By using the backward arcs, paths can be constructed which consist of several nodes. The maximum flow in graph G_2 is used to determine how the feasible flow in graph G_1 can be reduced to a minimum flow. Every backward arc between two jobs contained in the maximum flow in G_2 represents two jobs being transported by the same vehicle. Clearly, every combination of two jobs in the maximum flow in G_2 results in the saving of one vehicle. The value of the maximum flow consequently indicates the total saving of vehicles compared to the value of the feasible flow from step 1. In other words, this maximum flow in G_2 indicates which forward arcs have to be removed from the feasible flow in G_1 and which backward arcs have to be added to the feasible flow in G_1 . The value of the maximum flow in graph G_2 indicates how much the value of the feasible flow in graph G_1 must be reduced to obtain a minimum flow in graph G_1 .

Theorem 1: The result of the algorithm *Minimum Flow* is a feasible flow of minimum value in graph G .

Proof: see Appendix

Theorem 2: The Minimum Flow algorithm solves the problem of determining the minimum number of AGVs in $O(n^{5/2})$ time, where n is the number of jobs.

Proof: From Ahuja, Magnanti and Orlin³ it follows that the unit capacity maximum flow algorithm solves a maximum flow problem on unit capacity simple networks in $O(n^{1/2}m)$ time, where n is the number of nodes and m the number of arcs. In this kind of networks, every node (except source and sink) has at most one incoming arc or at most one outgoing arc. The number of arcs in graph G_2 is $O(n^2)$. Hence, the complexity is $O(n^{1/2} * n^2) = O(n^{5/2})$.

Example

After the arrival of the ship, a number of containers have to be unloaded and loaded. For this purpose, one Quay Crane (QC) and one Automated Stacking Crane (ASC) are used. The transport from the QC to the ASC and vice versa is carried out by Automated Guided Vehicles (AGVs). Table 1 gives the travel times of empty and full AGVs from the QC to the ASC (and vice versa).

INSERT TABLE 1

We assume that the waiting times of containers at QCs or ASCs are zero. In this simplified example we consider a set of three containers. The characteristics of these containers are

given in Table 2. The arrival instants are determined by using the data from Table 1 and the aforementioned assumptions.

INSERT TABLE 2

From the data in Table 2 it can be seen that the AGV with container 2 has to wait at the ASC p&d point until the ASC has finished storing container 1. From equation (2) it follows that the values of t_i ($1 \leq i \leq 3$) are:

INSERT TABLE 3

As described, we solve this problem by constructing a network. Containers i and j can be transported by the same AGV if the start time of container j is later than or equal to the delivery instant of container i plus the travel time from the destination of i to the origin of j . From Tables 2 and 3 it is obvious that the jobs $(1, 2), (1, 3), (2, 3)$ are compatible. By adding a source s and sink t and the arcs (s, i) and $(i, t) \forall i \in V$ we can construct graph G with $V = \{s, 1, 2, 3, t\}$ and $A = \{(s, 1); (s, 2); (s, 3); (1, 2); (1, 3); (2, 3); (1, t); (2, t); (3, t)\}$. This graph is presented in Figure 2.

INSERT FIGURE 2

First, we perform the transformation as described before. To solve the problem of finding the minimum number of AGVs we use the minimum flow algorithm from the previous section.

From step 1 it follows that the maximum number of AGVs equals the number of jobs. Hence, the value of the maximum flow equals 3. This flow is shown in Figure 3.

INSERT FIGURE 3

Next, graph G_2 is constructed (see Figure 4). The upper bound of the capacity of each arc follows from equations (8) and (9) and is shown in Figure 4.

INSERT FIGURE 4

Any maximum flow algorithm is used to determine the maximum flow through G_2 . The maximum flow equals 2 and the following arcs are included in this maximum flow:

$(s, 2'); (s, 3'); (1'', t); (2', 1''); (3', 2''); (2'', t)$.

The arcs that belong to this maximum flow are shown in Figure 4.

Using this maximum flow, the values of x_{ij}^* (see step 5 of the algorithm) are determined:

$$\begin{aligned} x_{s2'}^* &= 0 & x_{1''2'}^* &= 1 \\ x_{s3'}^* &= 0 & x_{2''3'}^* &= 1 \\ x_{1''t}^* &= 0 & x_{s1'}^* &= 1 \\ x_{2''t}^* &= 0 & x_{3''t}^* &= 1 \end{aligned}$$

Consequently, the minimum value of a flow in graph G_1 is 1. All containers can therefore be transported by one AGV. The flow with minimum value is shown in Figure 5 .

INSERT FIGURE 5

Conclusions and further research

Container terminals are faced, more and more, by customers and shipowners that demand delivery of their containers and fast unloading and loading of their ships. To handle the strong competition between terminals it is also essential to reduce the costs. To meet these demands terminals are looking for new techniques, such as automated transportation systems and ways of control. Adequate planning and control concepts for the various systems in the terminal therefore need to be developed.

One of the problems that has to be solved is the determination of the number of AGVs required to transport all containers within time. This paper presents an algorithm to determine the minimum number of AGVs required at a semi-automated container terminal. These AGVs are used to transport containers from QCs to ASCs and vice versa at known time instants. From a theoretical point of view the algorithm is 'good' because it is a strongly polynomial time algorithm. The various assumptions will also be checked in practice with data from a real terminal. Furthermore, the model and algorithm can be generalised and can be used in warehouses and manufacturing systems.

Further research will be carried out on an extension of the described problem, namely the determination of the number of vehicles required if every job has a time window. In this case, a release time and a due time are given for every job.

Appendix

Theorem 1

The result of the algorithm *Minimum Flow* is a feasible flow of minimum value in graph G .

Proof

Original problem

The original problem consists of finding a minimum flow in a graph $G = (V, A)$ with lower bounds l_{ij} and upper bounds u_{ij} on the arcs $(i, j) \in A$. This problem is formally formulated as follows

$$\begin{aligned} \mathcal{P} = \quad & \text{Minimise } v \\ \text{subject to } & \sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = \begin{cases} v & i = s \\ 0 & i \neq s, t \\ -v & i = t \end{cases} \quad \forall i \in V \\ & l_{ij} \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A \end{aligned}$$

Initialisation (step 1)

Given is a feasible flow x in the graph $G_1 = (V_1, A_1)$ with value K , i.e. given $x_{ij} \forall (i, j) \in A_1$ satisfying

$$\begin{aligned} V_1 &= \{s, 1', 1'', \dots, N', N'', t\} \\ A_1 &= \{(i'', j') : i'', j' \in V_1, \text{ jobs } i \text{ and } j \text{ are compatible}\} \\ &\cup \{(i', i'') : i', i'' \in V_1\} \cup \{(s, i') : s, i' \in V_1\} \\ &\cup \{(j'', t) : j'', t \in V_1\} \\ \sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} &= \begin{cases} K & i = s \\ 0 & i \neq s, t \\ -K & i = t \end{cases} \quad \forall i \in V_1 \\ l_{ij} \leq x_{ij} \leq u_{ij} & \quad \forall (i, j) \in A_1 \end{aligned}$$

Note that such a flow can easily be constructed by computing a maximum flow in G_1 .

Maximum flow in transformed graph $G_2 = (V_2, A_2)$

Define (according to step 3) a graph $G_2 = (V_2, A_2)$ where the node set V_2 equals V_1 and the arc set A_2 consists of all arcs in A_1 , and of a backward directed arc (j, i) for each arc $(i, j) \in A_1$. The capacities of the arcs in A_2 are defined by

$$\begin{aligned} \Delta_{ij} &= x_{ij} - l_{ij} \quad \forall (i, j) \in A_2 : (i, j) \in A_1 \\ \Delta_{ij} &= u_{ji} - x_{ji} \quad \forall (i, j) \in A_2 : (j, i) \in A_1 \end{aligned}$$

Solve the maximum flow problem stated below (step 4).

$$\begin{aligned} \mathcal{P}' = \text{Maximise } & v' \\ \text{subject to } & \sum_{j:(i,j) \in A_2} x'_{ij} - \sum_{j:(j,i) \in A_2} x'_{ji} = \begin{cases} v' & i = s \\ 0 & i \neq s, t \\ -v' & i = t \end{cases} \quad \forall i \in V_2 \\ & 0 \leq x'_{ij} \leq \Delta_{ij} \quad \forall (i, j) \in A_2 \end{aligned}$$

A solution to the minimum flow problem (step 5)

An optimal solution x^* to \mathcal{P} can be constructed by combining the feasible flow x with the maximum flow x' as follows

$$x^*_{ij} := x_{ij} - x'_{ij} + x'_{ji} \quad \forall (i, j) \in A_1.$$

To prove that the result of the algorithm Minimum Flow is a feasible flow with minimum value, the following has to be checked.

Proof of feasibility and optimality of x^*

Consider an arc $(i, j) \in A_1$, and the corresponding variables x'_{ij} and x'_{ji} from \mathcal{P}' . For these variables the following holds

$$\begin{aligned} 0 &\leq x'_{ij} \leq x_{ij} - l_{ij} \\ 0 &\leq x'_{ji} \leq u_{ij} - x_{ij}. \end{aligned}$$

Adding these two inequalities gives

$$l_{ij} - x_{ij} \leq -x'_{ij} + x'_{ji} \leq u_{ij} - x_{ij}.$$

Adding x_{ij} to this inequality gives $x_{ij} - x'_{ij} + x'_{ji} = x^*_{ij}$ for the middle term, and it follows that x^* is feasible with respect to the flow bounds:

$$l_{ij} \leq x^*_{ij} \leq u_{ij} \quad \forall (i, j) \in A$$

Next, consider the flow conservation constraint in \mathcal{P}'

$$\begin{aligned} & \sum_{j:(i,j) \in A_2} x'_{ij} - \sum_{j:(j,i) \in A_2} x'_{ji} = \\ & \sum_{j:(i,j) \in A_1} x'_{ij} + \sum_{j:(j,i) \in A_1} x'_{ij} - \sum_{j:(j,i) \in A_1} x'_{ji} - \sum_{j:(i,j) \in A_1} x'_{ji} = \\ & \sum_{j:(i,j) \in A_1} (x'_{ij} - x'_{ji}) - \sum_{j:(j,i) \in A_1} (x'_{ji} - x'_{ij}). \end{aligned}$$

The step from the first to the second equation follows by considering A_1 rather than A_2 as the index set over which to sum, and by correspondingly indexing the variables x'_{ij} for which $(i, j) \notin A_1$ on the backward directed arc (j, i) that does exist in A_1 .

Using the definition of x^*_{ij} we obtain

$$\begin{aligned} & \sum_{j:(i,j) \in A_1} (x'_{ij} - x'_{ji}) - \sum_{j:(j,i) \in A_1} (x'_{ji} - x'_{ij}) = \\ & \sum_{j:(i,j) \in A_1} (x_{ij} - x^*_{ij}) - \sum_{j:(j,i) \in A_1} (x_{ji} - x^*_{ji}) = \\ & \left(\sum_{j:(i,j) \in A_1} x_{ij} - \sum_{j:(j,i) \in A_1} x_{ji} \right) - \left(\sum_{j:(i,j) \in A_1} x^*_{ij} - \sum_{j:(j,i) \in A_1} x^*_{ji} \right) \end{aligned}$$

From the initialisation we know that

$$\sum_{j:(i,j) \in A_1} x_{ij} - \sum_{j:(j,i) \in A_1} x_{ji} = \begin{cases} K & i = s \\ 0 & i \neq s, t \\ -K & i = t \end{cases}$$

With all the above, problem \mathcal{P}' can be rewritten in terms of x_{ij}^* as

$$\begin{aligned} \mathcal{P}' = & \text{Maximise } v' \\ \text{subject to } & \sum_{j:(i,j) \in A_1} x_{ij}^* - \sum_{j:(j,i) \in A_1} x_{ji}^* = \begin{cases} K - v' & i = s \\ 0 & i \neq s, t \\ v' - K & i = t \end{cases} & \forall i \in V_1 \\ & l_{ij} \leq x_{ij}^* \leq u_{ij} & \forall (i, j) \in A_1 \end{aligned}$$

Introducing $v^* = K - v'$, the objective becomes to maximise $v' = K - v^*$, or, equivalently, to minimise $v^* - K$, where K is a constant and can thus be left out of the objective. We finally write

$$\begin{aligned} \mathcal{P}' = & \text{Minimise } v^* \\ \text{subject to } & \sum_{j:(i,j) \in A_1} x_{ij}^* - \sum_{j:(j,i) \in A_1} x_{ji}^* = \begin{cases} v^* & i = s \\ 0 & i \neq s, t \\ -v^* & i = t \end{cases} & \forall i \in V_1 \\ & l_{ij} \leq x_{ij}^* \leq u_{ij} & \forall (i, j) \in A_1 \end{aligned}$$

It follows that x^* is both feasible and optimal for \mathcal{P} . ■

References

1. Evers JJM and Koppers SAJ (1996). Automated guided vehicle traffic control at a container terminal. *Transportation Research A* **30**: 21-34.
2. Bodin LD, Golden BL, Assad AA and Ball MO (1983). Routing and scheduling of vehicles and crews. *Computers and Operations Research* **10**:63-211.
3. Steenken D (1992). Route Optimization at a Seaport Terminal under real-time conditions. *OR Spektrum* **14**:161-168 (in German).
4. Steenken D, Henning A, Freigang S and Voss S. (1993) Routing of straddle carriers at a container terminal with the special aspect of internal moves. *OR Spektrum* **15**: 167-172.
5. Kim KH and Bae JW (1999). A dispatching method for automated guided vehicles to minimize delays of containership operations. *International Journal of Management Science* **5**: 1-25.
6. Chen Y, Leong YT, Cheong Ng JW, Demir EK, Nelson BL and Simchi-Levi D (1998). Dispatching Automated Guided Vehicles in a Mega Container Terminal. Presented at the *INFORMS conference*. May 1998, Montreal, Canada.
7. Demir EK, Leong T, Li C, Cheong Ng JW and Simchi-Levi D (1998). Locating Containers in a Mega Terminal. Presented at the *INFORMS conference*. May 1998, Montreal, Canada.
8. Maxwell WL and Muckstadt JA (1982). Design of Automatic Guided Vehicle systems. *IIE Transactions* **14**:114-124.

9. Ilić OR (1994). Analysis of the number of automated guided vehicles required in flexible manufacturing systems. *The International Journal of Advanced Manufacturing Technology* **9**:382-389.
10. Mantel RJ and Landeweerd HRA (1995). Design and operational control of an AGV system. *International Journal of Production Economics* **41**: 257-266.
11. Gobal SL and Kasilingam RG (1991). A simulation model for estimating vehicle requirements in automated guided vehicle systems. *Computers and Industrial Engineering* **21**: 623-627.
12. Dantzig GB and Fulkerson DR (1954). Minimizing the number of tankers to meet a fixed schedule. *Naval Research Logistics Quarterly* **1**: 217-222.
13. Ahuja RK, Magnanti TL and Orlin JB (1993). *Network Flows, theory, algorithms, and applications*. Prentice Hall: New Jersey.
14. Phillips DT and Garsia-Diaz A (1981). *Fundamentals of network analysis*. Prentice-Hall, Inc.:Englewood Cliffs, New York.
15. Ford LR and Fulkerson DR (1962). *Flows in Networks*. Princeton University Press: Princeton, New Jersey.
16. Orlin JB (1982). Minimizing the number of vehicles to meet a fixed periodic schedule: an application of periodic posets. *Operations Research* **30**: 760-776.

	QC to ASC (and vice versa)
empty AGV	1
full AGV	2

Table 1 travel times for empty and full AGVs

container i	origin	destination	release s_i	arrival v_i	handling b_i
1	QC	ASC	1	3	4
2	QC	ASC	4	6	1
3	ASC	QC	9	11	2

Table 2 container characteristics

container i	t_i
1	3
2	7
3	11

Table 3 delivery instant

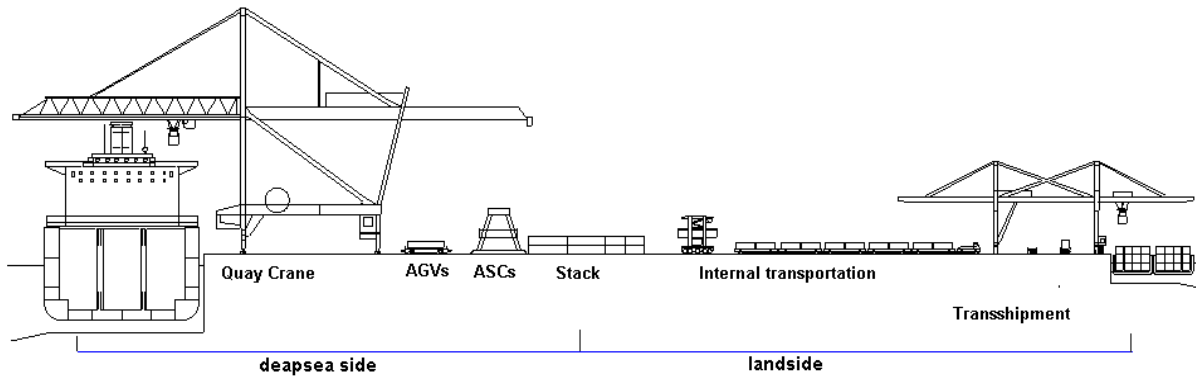


Figure 1: Process of (un)-loading a ship in an automated terminal

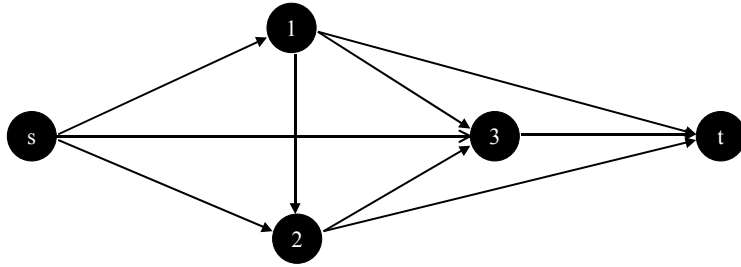


Figure 2: Graph G

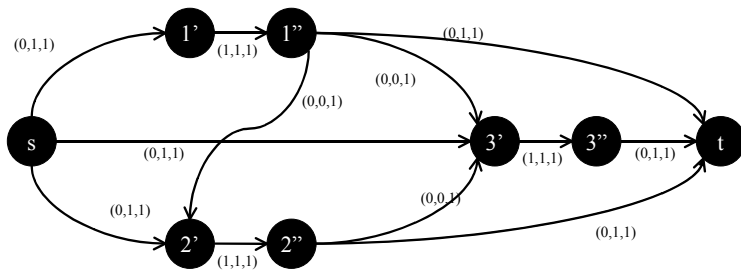


Figure 3: Graph G_1 with triple (l_{ij}, x_{ij}, u_{ij})

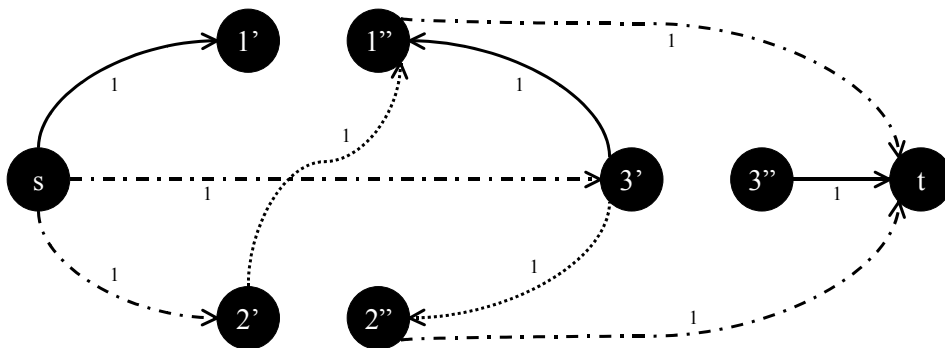


Figure 4: Graph G_2 with forward arcs (-.-) and backward arcs (...) that belong to the maximum flow

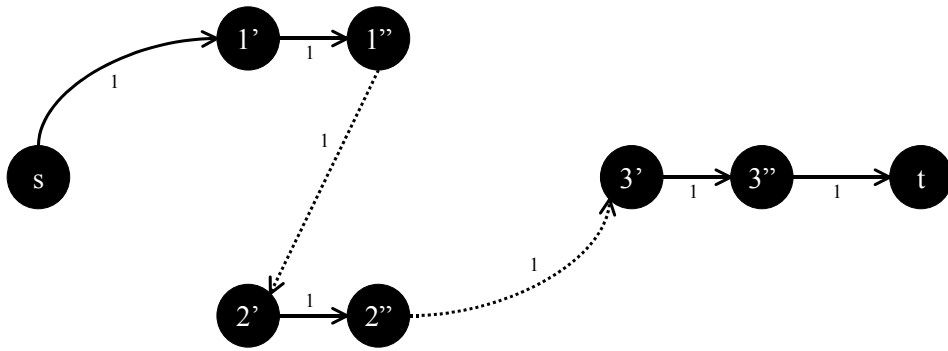


Figure 5: Minimum Flow in graph G_1