Real-time hiring of vehicles for container transport

Joseph J.M. Evers
Logistics Technology
Delft University of Technology
The Netherlands
e-mail: j.j.m.evers@tbm.tudelft.nl

EJTIR, **6**, no. 2 (2006), pp. 173-198

Received: March 2005
Accepted: October 2005

This study concerns the deployment of large numbers of container transporting vehicles on a network consisting of (dedicated) roads and logistic sites such as container terminals. Each site is managed autonomously by a site manager. The fleet of vehicles is deployed by a fleet manager. On request of the site managers, the fleet manager allocates vehicles to be hired by the site managers. Next, within these allocations, each site manager issues transportation jobs and assigns these jobs to the allocated vehicles. The objective of the fleet manager is to avoid unpaid empty driving and to keep the size of its fleet just large enough the serve the demand for vehicles. The objective of the site managers is to minimize the operational costs. In this context a new coordination system is proposed to allocate vehicle capacity, operating on the basis of a win-win situation for all actors. The coordination takes place at two levels. One level concerns the drawing up of an adaptive fleet deployment plan. The other level resides under control of the individual site managers and concerns the assignment of transportation jobs to vehicles, given the allocations of the fleet deployment plan. Both use repetitive linear programming procedures. Small size simulation experiments show that the approach might be effective.

Keywords: Deployment of vehicles, Container logistics, Online transportations control.

1. Introduction

1.1 The challenge

The problem is relevant in the context of container transport in the harbour of Rotterdam. At the moment each terminal takes care of its own internal transport, using its own (automated) terminal vehicles. The inter terminal transport operates as separate service on the basis of bilateral appointments and is not integrated with the internal transport systems. Now suppose that all vehicles are suitable for all types of transport and that the deployment of these

vehicles is coordinated optimally, using real-time information. The benefits are clear: because of mutual peak shaving a smaller number of vehicles will suffice and, because of optimized coordination, empty driving will be reduced.
However, because of the autonomy of the terminal operators, such a setup is organizationally not acceptable. More precisely, the objectives of the fleet manager on one side and the objectives of the site manager will differ and, dependent of the operational situation, may be conflicting. The objective of the fleet manager is to avoid unpaid empty driving and to keep the size of its fleet just large enough the serve the demand for vehicles. The objective of the site managers is to minimize the operational costs. Then the challenge is to develop a real-time optimizing coordination system for the deployment of vehicles and the execution of transportation jobs that respects the autonomy of all actors.
To that we propose a transport system, where all vehicles are deployed (and owned) by a fleet manager and where, on ad hoc basis, these vehicles are hired by the terminal mangers for transports to other terminals and for internal transport on their own terminal. We need procedures that anticipate on the need for vehicles, where the actual instructions to vehicles to execute specific jobs are postponed up to last moments, but such that the processes proceed without waiting for instructions. In this way one may react to changing situations, maintaining reliable efficient service providing. The efficiency criteria cover low driving distances (by avoiding empty driving) and minimal size of the fleet (by peak shaving, i.e. by anticipating execution of jobs), but such that the service requirements are met.
In principle, this approach can be applied to the hinterland container transport related to the harbors of Rotterdam. Under the name *Rhine delta Transport System (RTS)* a network for automated container transport is proposed, aimed to connect the container terminals of the harbor with various inland terminals. On the average, internal transport jobs on a terminal will take ±5 minutes, whereas the inter terminal transports, depending on the driving distances, will take 10 to 120 minutes. Then, following the prognoses of the Central Planbureau on harbor related container transport it appears that in the year 2010 for such an *RTS* at least 800 vehicles are required: 200 vehicles for the transport on the large harbor area Maasvlakte and 600 for the hinterland transport; see (Gemeentelijk Havenbedrijf Rotterdam, 1998).
For the harbor area of Maasvlakte the inter terminal transport might be extremely important. For instance it can be shown that, provided the inter terminal transport is functioning efficiently, a special terminal for coastal and large scale inland feeder shipping is very comparative when compared with distributed ship service (see Evers and De Feijter, 2004).

1.2 Related literature

A recent survey of research in transportation systems operating with *Automated Guided Vehicles* (AGVs) is presented by Vis (2004). This concerns vehicles in warehousing, manufacturing, container terminals and external (underground) transportation systems. From our viewpoint especially the sections on vehicle requirements and AGV control are interesting. The literature on vehicle requirements (i.e. the minimal fleet size for internal transport) shows various approaches. There are network flow models where, with the help of (integer) linear programming, minimal fleet sizes are estimated before starting the actual operations; see Maxwell et al. (1982) and Vis et al. (2001). Estimates for appropriate fleet sizes are also studied with the help of simulation models; see Kasilingam et al. (1996) and Evers et al. (1998) and Duinkerken et al. (2001).

The survey on AGV control (Vis, 2004) covers dispatching of loads to AGVs, route selection, scheduling and positioning of idle AGVs; also see Qiu et al. (2002). Given the stochastic nature of the intended transportation process, the literature recommends real-time control. Scheduling is aimed to dispatch a (rather limited) number of AGVs to execute a sequence of transportation jobs under constraints such as earliest starting times, latest completion times, priorities, etc. Usually the objectives concern minimizing the number of AGVs, or minimizing the total travel times. Given a scheduling procedure, the mission of routing is to find a suitable route with respect to the driving distance or driving time, possibly taking into account the current traffic situation.

Dispatching can be triggered by a pending job which needs to be assigned to a vehicle, or, the other way round, by an idle vehicle looking to be assigned a new job. There are many articles on this topic, mostly about the performance of heuristic dispatching rules. An example is the *greedy dispatching rule* studied by Chen et al. (1998), where vehicles and jobs are matched such that empty driving is minimized, but without anticipation on succeeding jobs. Van der Meer (2000) showed that in general vehicle initiated rules are outperformed by load initiated rules. Van der Heijden et al. (2002) studied dispatching policies for an underground transportation system with a large number of AGVs. They show that rules using information on future orders are superior over simple rules. Studying real-time vehicle dispatching in a distribution centre, a container terminal and a production site, Koster et al. (2004) conclude that rules based on pre-information on jobs and arriving vehicles perform best in all cases.

In the *look ahead heuristics* of Kim and Bay (2004) information is used on the state of operations, to synchronise the transhipment moves of the quay cranes and the arrival of AGVs at these cranes. The tasks are assigned with the help of a mixed-integer programming model, but also with a heuristic algorithm (to avoid excessive calculation times). Both perform much better than conventional dispatching rules. Look ahead dispatching rules also are applied by Evers et al. (1998) and Duinkerken et al. (2001) in control of quay transport during ship loading. Each quay crane triggers the supply of a new specific container the moment his *work stock* falls below an *order point.* The *work stock* is defined as the number of vehicles waiting at the platform plus being on their way. The *order point* is defined as the 97% safety level with respect to the average *lead time* (i.e. the execution time of a job) divided by the average handling time at the crane. In order to maintain the sequencing order of arriving containers, the timing of the start of the job to deliver that container takes the expected driving time into account. Note: in addition this policy avoids congestion at the transition platforms (i.e. the platforms where the loading and unloading take place) avoids waiting of vehicles. Simulations show that in this way the only 3% of crane capacity is wasted by waiting for an AGVs to deliver a container and that 95% of the containers are delivered in the preferred sequencing order.

Another subject is the management of idle vehicles. A vehicle becomes idle when he delivered his load at the destination and is not immediately assigned a new job. Becoming idle it must be decided where to locate the vehicle, such that he can react efficiently to a new assignment. Several rules are proposed, such as *central positioning* and *circulatory loop positioning*; see Egbelu (1993). The first rule directs an idle vehicle to a parking area where they will wait for a new job. The second rule directs an empty vehicle to drive a specific path (or loop), aimed to reduce the response time. Evers et al. (1998) and Duinkerken et al. (2001) studied a variant of *circulatory positioning* in the context of an intensive loading operation of a jumbo container ship. The vehicles follow a loop along the quay to supply their loads to quay cranes and next along the transition platforms of the stack lanes of the container stack.

When a vehicle becomes idle, he is directed to the entrance of the stack and, arriving there, he is instructed to go to a specific stack lane. There he receives a container with the instruction to drive to a specific quay crane. In this the specific job assignments are postponed without sacrificing efficiency and, consequently, vehicle capacity is kept flexible. Indeed simulations show that this setup is effective: grid locks (i.e. the situation where vehicles block each other because of the rules of giving right of way) are completely avoided.

With each transportation job a route must be associated, specifying the path the vehicle should be taken from pick-up to delivery. Also there must be a route for the vehicle to drive from his current position to the pick-up point. In our case of single load transportation there are no additional stops between pick-ups and deliveries which, of course, simplifies the routing problem. Routes can be generated by *static* or by a *dynamic* procedure. With static procedures the routes are associated in advance and non-adaptive. In dynamic procedures real-time information (on traffic congestion) is used to optimize the driving.

Vehicle routing is extensively studied using (complicated) mathematical programming techniques, where a set of *n clients* are to be served by *m vehicles*, where *n* is much larger than *m.* However, since we are focussed on single-load vehicles and in addition the random perturbation component in the duration of the trips is relatively rather large, it is better to restrict the *planning horizon* such that $n < 3 * m$ and also to use real-time information to make scheduling adaptive. Consequently the approach as a *dynamic scheduling vehicle problem* seems to be more effective. From this there is analogy with the *dial-a-ride problem,* where the control reacts in real-time to a client's request, such that their waiting times are minimized.

Most of the studies on routing are concerned with constructing conflict-free shortest-time paths. In this context Evers and Koppers (1996) propose the *semaphore*, borrowed from computer science, as basic intelligent traffic controller. Simulation studies by Evers et al. (1998) show that traffic control with the help of semaphore can be very effective. For our intended domain of application we assume that the infrastructure is well-designed, that advanced traffic control takes keeps the traffic flowing and that appropriate job-dispatching avoids overloading of transhipment platforms, critical tracks and parking places. Consequently, static procedures (eventually with real-time route selection) will suffice.

Vis (2004) observes that most of the literature on scheduling hardly considers constraints imposed by limited capacities of transition equipment, parking places, critical tracks, etc.. For instance, the integrated mathematical model presented by Meermans (2002) on terminal logistics neglect limited capacities on transition platforms. An exception is the study Ebben et al. (2004b) presenting an approach where limited capacities are taken into account put into a multi-period, rolling finite horizon programming model. They use heuristic optimization rules.

An integrated approach on terminal control is presented by Murty et al. (2005). They propose a decision support system for operations in a container terminal, focussing on a system that reacts adequately to momentary changes in the workload level over time and to uncertainty in working conditions. One the functions to be supported concern the estimation of vehicle requirements each half-hour and to hire the minimal number of vehicles each day, such that the requirements are met. Following the practice of the terminals in Hong Kong, they propose a planning period of 4 hours (i.e. half of an 8-h shifts), because over such periods work loads can be estimated with reasonable accuracy. At the end planning period, the terminals use the latest information for the following period to formulate operation plans.

The study of Fink and Reiners (2005) concerns a car rental company, running a number of autonomous acting corporate stations. The company centrally decides about the allocation of cars. Customers hire a car of a specific type from specific station; most of them make a reservation and most of them return the car at the same station on time. On that basis (and on general statistics) it is possible to make estimates on the availability of cars on the stations and the demand for cars; next one deduces estimates on surpluses and deficiencies of cars at the stations on the forthcoming days. The company wishes to maintain a high service level (> 98%) and therefore cars may be transported (by special trucks) from one station to another. To support the decisions, they developed a multi-period rolling horizon model, minimizing the transportation cost, given a (minimal) number of cars the company owns. The duration of a period is a half-day; the horizon is 5 days; the updating is daily. Compared with the practice operating without such model, indeed substantial savings are possible maintain a service level of 99.9%.

1.3 The contribution of the study

In the literature we did not find studies on fleet deployment where the operations on the logistic sites are interwoven with fleet deployment, but where, in the same time, the site managers act autonomously. Nevertheless most of the recommendations found in the literature also apply to our problem. More precisely:

- Use job initiated dispatching rules acting on pre-information on process states, such as dispatching triggered by maintaining optimal work stocks for handling equipment;
- Follow the principle *postpone binding,* when dispatching jobs to vehicles and when positioning idle vehicles;
- Take into account limited capacities or limited availability of handling equipment and roads (then, combined with intelligent traffic control, static routing suffices).

Integrating models in the literature show a multi-period, rolling horizon set-up, using intelligent heuristics and/or (integer) linear programming to optimize vehicle deployment.
In our proposed model, the coordination takes place at two levels: on concerning the drawing up of an adaptive fleet deployment plan and the other concerning the assignment of transportation jobs to vehicles on the sites, given the allocations of the fleet deployment plan. Incorporating the recommendations from the literature, the optimizing heuristics are based on multi-period rolling horizon (integer) linear programming. The interface between the fleet manager and the site managers is based on cumulatively formulated categorical lower and upper bounds of the number vehicles the site managers asking for. With the help of differentiated tariffs it is possible to provide incentives to support overall efficiency.
In this context the objective of the paper is to propose a setup of a decision support system for vehicle deployment, fitting in the paradigm of distributed intelligence. More precisely:

- To present this new generic model for optimizing deployment of vehicles, such that there is a win-win situation for all autonomously operating actors;
- To elaborate this model with the help of well-known linear programming software;
- To evaluate the consistency and the potential effectiveness of the models with the help of small size simulation experiments.

It should be noted that this verification is rather limited. The intension is simply to provide a first feasibility scan. A positive outcome (which appears to be the case) may justify the investments needed to study our proposed approach in simulation studies, reflecting a real

application such as the terminal transport on the Maasvlakte. However, reporting on the underlying modelling study, this beyond the scope of this article.

1.4 Setup of the study

Section 2 concerns the logistical aspects. Paragraph 2.1 on terminal logistics sketches the most important basic processes, to show that most of the equipment is endowed with (de-coupling) stacks. These stacks offer a margin to execute jobs in advance and with this possibilities of peak shaving. This is elaborated further in the paragraphs 2.2 and 2.3. A proposal for the interaction in real-time between the fleet manager and a site manager is described in paragraph 2.4. The linear program for fleet deployment is outlined in 2.5.
Using the *Algebraic Mathematical Programming Language* (Fourer, 1997), Section 3 elaborates this linear programming model for cyclic adaptive fleet deployment planning. Using the LP-solver MOSEK, the performance is studied with the help of various small scale simulation experiments. Some attention is given to the computational complexity, but we refrain from a mathematical analysis. In addition we show that, on the sites, a similar model can be used for job initiated deployment. Section 4 shows how the model can be used for job assignment control on the terminals. Section 5 presents some of the conclusions.

2 Logistics of container transport

2.1 Terminal logistics

On the level of physical handling, varying flows of transportation jobs are dispatched, which next must be assigned to the available vehicles. This must be compatible with the fleet deployment plan on the first time-cycle. Each job also includes two transhipment moves: loading and unloading. For a 40-feet container a transhipment move takes 1 to 2 minutes. To this the waiting time must be added. The classic mathematical queuing model for randomly arriving 'clients' and one 'server' gives an indication on the average waiting time, being $W = ½ \times R \times B / (1-R)$, where R is the rate of occupation and B the service time.
From the viewpoint of logistics, one may distinguish a critical rate of occupation (higher than 95%) and a non-critical (less than 80%). The average waiting times for a transhipment move will vary from 15 to 20 minutes for a critical occupation and 1 to 3 minutes for the non-critical. Waiting times 15 to 20 minutes are far too long to be efficient. Moreover long queues of vehicles will require large parking platforms, which may block other terminal operations. Therefore, in the case of critical occupation, random arrival of vehicles is not appropriate. The alternative is to let the transhipment equipment trigger the issuing of jobs.
A typical example of such a setup shows the handling of large intercontinental container vessels. On the average 7 containers per minute must be transported between the stack and the quay, where the quay cranes take care of the transhipment. To achieve such a performance, special logistics on a special layout are required. The stack (usually called the *main stack*) is situated directly in front of the quay; see figure 1.

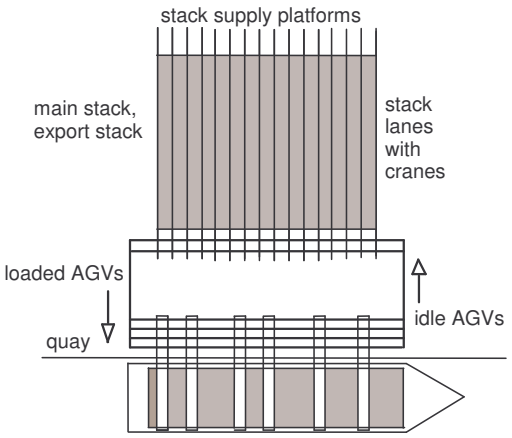

Figure 1. Circular shuttle between quay and stack

During the loading operation the vehicles drive circularly from the stack to quay and back: along the stack to receive a container and next along the quay to deliver the container. Then they drive empty to the entrance(s) of the stack, getting a new order to pick up a new container. Each crane triggers its supply from the stack; c.f. Evers et.al. (1998). It is clear that such a dedicated operation can fit in a fleet deployment plan under the category 'hired for internal use on the site'.

In contrast to the dedicated intensive quay transport appearing during the loading operation of a jumbo container vessel, the unloading operation is logistically less critical because there is much more freedom in delivering a containers at the stack than in picking-up. Therefore *import containers* (i.e. containers that arrive via a deep-sea vessel) may be transported directly to specific transhipment stations. Also the inland supply of containers to the main stack gives rise to transport from train, ship and truck transhipment stations. This typically generates distributed criss-cross transport, which is triggered by the arising jobs.

When the transport takes place between distributed transhipment platforms, the distances are longer and, consequently the random component in the driving times will increase. Then, to keep the waiting times acceptable, the occupation rate of the transhipment equipment has to be non-critical. In addition often, to de-couple the handling processes on the operational level, the transhipment stations are equipped with a (small) stack. Figure 2 shows an example of a service station for road trucks.

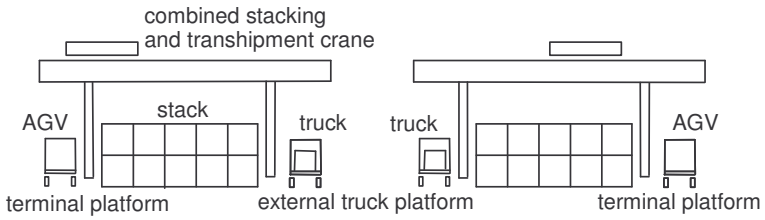

Figure 2. Profile of a duo truck service station

The example of figure 2 shows a transhipment station for large inland barges and (small) coastal ships, proposed by Evers and De Feiter (2004) . The station is equipped with quay crane and with an intelligent stacking crane, situated under the reach of the quay crane. The configuration is intelligently controlled, where the slots in the hold during unloading and loading are selected dynamically such that the ship stays in balanced and where conflicting positions of the cranes are avoided. In principle arriving vehicles are served directly by the quay crane, but when two vehicles are waiting, the arriving vehicle escapes to be served by the staking crane. Next, when the quay crane becomes free, he starts to tranship between the ship and the stack.

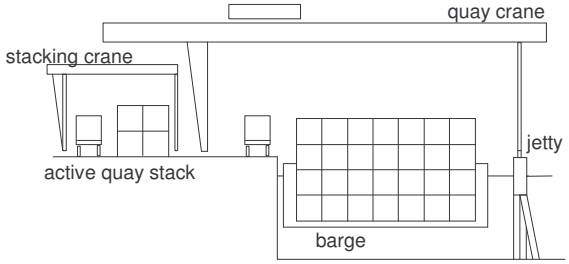

Figure 3. Profile of a Ship Service Station with Active Quay Stack

The idea is that the quay crane can deliver critical productivity (>90%), but such that the terminal transport can be handled as if the configuration is not critical. Simulations show that this is effective.
The relevant conclusions with respect to the set-up of our planning approach are the following:

- There can be dedicated operations, where a fixed number of vehicles are deployed in a collective task. Intensive ship loading is an example, where there will be no possibilities of anticipation.
- Normally there will be criss-cross transport between distributed transhipment platforms of a terminal. This type of transport is triggered by the issuing of jobs;
- Distributed transhipment stations may be equipped with stacks to de-couple the handling processes and, seen from the viewpoint of terminal transport, will not be critical with respect to waiting. These stacks offer a margin to execute jobs in advance, or afterwards.

2.2 Executing transport jobs in advance

Stacking facilities at the transition platforms and especially the main stacks provide the system with a margin to execute jobs in advance. This implies that, at any moment, the need for vehicles shows a lower bound, reflecting the current *latest departure times,* but also an upper bound, reflecting the current *earliest departure times* arising from the opportunity to do some of the transport in advance. Bounds like these must be modelled in such a way that, the number of jobs executed in advance, later on are subtracted from the lower bounds. Following Daganzo (1991), the easiest way to handle this is to formulate such bounds

cumulatively. Figure 4 shows an example: the line between the bounds represents a cumulative assignment.

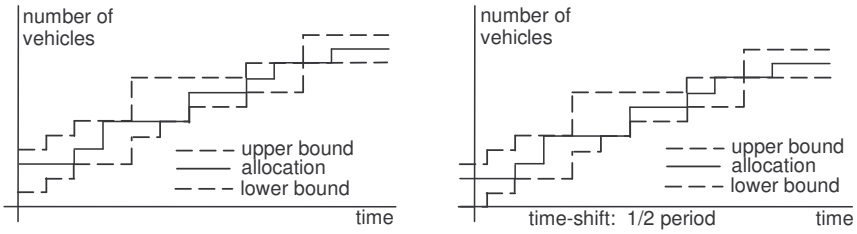

Figure 4. Cumulative requests for vehicle capacity, before and after updating

Given the diagram on the left, the diagram on the right is deduced as an updating of the left diagram after half a period (i.e. ½ * p), assuming that nothing has been changed and the execution has been proceed in conformity with the plan. Graphically this updating is done by shifting the diagram half a period to the left and down over half of the assignment on the first period. In conformity of this setup, each site manager is supposed to send his demand for capacity to the fleet manager in the form of cumulative upper and lower bounds.

2.3 Arranging a win-win situation by imposing special tariffs

The *fleet manager* may use the margin between the upper and lower bounds to avoid empty driving and to arrange peak shaving. Of course this is beneficial for him and therefore he will prefer large margins. On the other hand, for a site manager it is beneficial to get the capacities just-in-time, allowing him to optimise the use of the transhipment equipment. To stimulate the site managers to formulate their demands with a wide margin to anticipate on future transport, one may introduce incentives in the form of tariffs, such that anticipating transport is charged lower than dead-line transport. For such a system of tariffs, the following rules may be imposed:

- The assignments on surplus of the lower bounds go at lower tariffs than assignments up to the lower bounds;
- The tariffs are partly related to the actual delivery of capacity and partly to the reservation of capacity, where the latter will not be returned in case of cancellation;
- When the overall demand vehicle capacity shows a structural up-down pattern, one may introduce higher tariffs for the peak periods.
- It is possible to hire more capacity, after the delivery contract has been concluded, however then there will be a higher probability of refusal;

The innovative element is that, in stead of the usual centralized decision making, we now have decentralized decision making, where the overall efficiency can be stimulated with the help of differentiation of tariffs. These differences can be tuned in such a way that a win-win situation is obtained and has to be fixed in practice. However, this is beyond the scope of this study.

2.4 Interaction between fleet manager and site managers

Fleet manager and site manager have to control highly dynamic processes and therefore, they need anticipating adaptive management procedures. The data for fleet deployment planning typically is related to external service providing such as loading and unloading ships, trains or trucks. Each site manager's operation information system translates the demand for external service providing in the form of an update of timed numbers of vehicles, which are required. This concerns vehicles for the transport to other sites and own internal transport. These requirements for vehicles must be estimated, up to a day ahead.

In order to allow peak shaving and to reduce empty driving, fleet manager is supposed to use a rolling multi-period planning procedure with a horizon of (at least) 16 hours. Next, observing that the actual operations may show deviations with respect to the announced demand for transport, a rather short time-cycle is required. However a short cycle-time increases the number of planning periods and, with this, the size of optimization procedure; 15 minutes seems to be an appropriate compromise. It is clear that the interactions between the feet manager on one side and the site managers is intensive and requires a real-time communication; see table 1.

The result of a categorical contract between the fleet manager and the site manager of a site *S* is a numerical specification of the numbers of vehicles which are allocated to be used for the transport from *S* to specific other sites and for the internal transport on *S* it selves. Although the deployment plan may cover 16 hours or more, the binding part of a contract may cover only one hour ahead. Nevertheless also a contract might be updated every planning cycle, but within acceptable margins and possibly under contractual agreed penalties. Of course, t is the responsibility of the *site manager* to deliver the data.

Table 1. Interaction between the Fleet Manager and the generic Site Manger

Actions of Site Manager (SM)	Actions of Fleet Manager (FM)
SM updates figures on vehicle departures and on categorical needs for vehicles; SM sends the updated figures on vehicle departures and categorical demands to FM;	
	FM receives these figures from each SM, updates corresponding information and deduces new fleet deployment plan FM sends fleet deployment plan and data on vehicle's arrival times to each SM;
SM receives deployment plan and data on arrivals and decides on commitment for the next planning cycle; SM sends these decisions in the form of a categorical contract to FM;	
	FM receives SM's contacts and directs empty vehicles to planned positions;
SM decides on job-vehicle matching and instructs his handling equipment and vehicles;	Etc. repeat cycle

Clearly, in this setup the *fleet manager* plays the role of overall coordinator and consequently cyclically repeating flows of information are exchanged between the *fleet manager* on one side and the *site managers* on the other. It is clear that the interactions between the feet

manager on one side and the site managers are intensive and require a real-time communication.

2.5 Outline of the deployment model

In our model there are a fixed number of container terminals, called *sites*. As simplifying assumption we state that each pair of sites is connected by exactly two routes: one in each direction. A fleet of (automated) vehicles is available to carry out the transport. All vehicles are operationally equivalent and all carry single loads (i.e. a sea container); on the average a loading or unloading handling will take one to two minutes, exclusive waiting times. The fleet is managed by a fleet manager.

The deployment of the fleet is planned in the form of a multi-period procedure over a rolling time-window, where each time the information is updated and where, in conformity with this, the deployment plan is extended and possibly revised. Taking into account the daily fluctuations of the demand for transport capacity, the planning horizon (i.e. the duration of the planning window) has to be at least 16 hours. The cycle-time (i.e. the duration of a basic planning period) is introduced as a model parameter. To keep the model calculations manageable, a cycle-time of 10 to 20 minutes seems to be appropriate.

To be specific, let T_s, T_s+1, T_s+2, …,T_h be the periods of a current planning window. In practice the starting time of the planning window T_s will be 2 or more periods ahead with respect to the current period T_c. Each site manager i draws up the numbers vehicles he needs for the transports to other sites and for the transport on his own site. As we explain in section 2, this information is sent to the fleet manager in the form of cumulative lower bounds, denoted $lowReq[i,j,T_s]$, $lowReq[i,j,T_s+1]$, …, $lowReq[i,j,T_h]$, and cumulative upper bounds, denoted $upReq[i,j,T_s]$, $upReq[i,j,T_s+1]$, …, $upReq[i,j,T_h]$, where j refers to another site. We shall refer to this as a *categorical request.* Note, the case of $j = i$, refers to numbers of vehicles to be used for site-internal transports.

In each planning cycle, the fleet manager collects these *categorical requests* from all *site managers* and responds with a feasible *deployment plan* in the form of a categorical specification of numbers of vehicles $xAlloc[i,j,T_s]$, $xAlloc[i,j,T_s+1]$, …, $xAlloc[i,j,T_h]$. This means that (by contract) the manager of a site i is supposed to use $xAlloc[i,j,T]$ vehicles for transport jobs to site j, where execution starts during period T. Note: in practice only the allocations for the first planning period $xAlloc[i,j,T_s]$ are binding, whereas the future allocations $xAlloc[i,j,T_s+t]$ $(t>1)$ should be taken as indicative. Nevertheless, one may expect that the first allocations of the next planning cycle will deviate only slightly compared with allocations $xAlloc[i,j,T_s+1]$ being planned in the preceding cycle. In principle the planning procedure may be forced to restrict such deviation by imposing additional restrictions. Part of the *deployment plan* is the movement and parking of empty vehicles, i.e. vehicles which are idle and consequently reside under the control the fleet manager. In this study we propose and test an optimizing procedure to draw up such *fleet deployment plans.*

The related problem concerns site management. First, each planning cycle again, a *site manager* must produce a *categorical request*, which is based upon the planned (or expected) productivity of his plant and the state of execution of the current transport jobs. Of course he may use the preceding *categorical request* as a basis which can be modified and extended. Secondly, receiving a categorical allocation as part of a *fleet deployment plan*, a *site operating system* must continuously issue transportation jobs and assign these jobs to the vehicles. Of course this job-vehicle matching must be compatible with the categorical

allocations being in operation. In this study we show that this aspect of site operation management is compatible with the proposed approach of fleet deployment planning and we propose an optimizing procedure for job-vehicle matching.
As site managers are operating autonomously, an important subject is the efficiency of the system as a whole. In that respect the margins between lower and the upper categorical requests are interesting. More specifically, the fleet manager may use these margins to draw up categorical allocations, which avoids empty driving and which arranges peak shaving. Therefore the fleet manager prefers to have large margins. To stimulate the site managers to formulate their requests with a wide margin, the fleet manager may introduce incentives in such a form that the numbers of vehicles which are allocated in surplus on the lower bounds, i.e. *xAlloc[i,j,T] – lowReq[i,j,T],* are charged at lower tariffs than the numbers *lowReq[i,j,T].*

3. Fleet deployment planning with the help of Linear Programming

3.1 Outline of the Linear Programming Model

The decision variables in the process of deployment planning concern the numbers of vehicles which will be allocated to the *site managers,* to be used on specified destinations. In addition, the decision variables concern the numbers of empty vehicles, which must stay at the parking places or which must move to specified directions. The deployment plans must satisfy various constraints to be feasible; more precisely:

- Vehicle conservation conditions (i.e. vehicle may not enter or vanish "magically") applying to each site as a whole, to each parking place and to each driving track;
- Capacity constraints of transhipment equipment and parking places;
- The availability and the capacities of roads;
- The lower and upper bounds on the demand for vehicles.

Within these constraints the fleet manager must find a cost-minimal plan, which satisfies the requests for vehicles. In this context only the costs caused by empty driving and by reduced tariffs for anticipating allocation of vehicles are relevant. Both the constraints and the costs calculation can be expressed in linear forms and, thus, we can use the power of linear programming technology.
On the application of linear programming on transportation problems there is much literature is; see for instance Nemhauser (1989), Bal et al. (1995) and Fourer (1993). Although our model looks complicated, the formulation is rather straightforward.

3.2 Intermezzo: Linear Programming modelling with *AMPL*

We have elaborated this linear programming model with the help of the special modelling software *AMPL*; *AMPL* is the acronym of *A Mathematical Programming Language*, and has been developed by Fourer, Gay and Kernighan (Fourer, 1993; 1990; 1997). We have used the *AMPL* student edition (from Internet) with full functionality, but restricted to small size. As linear programming solver we use Mosek (www.mosek.com). An *AMPL-model* can be used directly for experiments without any computer programming. *AMPL* distinguishes five model entities:

- *Sets* (indicated by the syntax word **set**) representing the domains of algebraic LP-indices, for example the set of node identifier in a transport network;
- *Parameters* (indicated by **param**) to indicate constants and identifiers of model specifying data, for instance the driving times to pass the tracks our network;
- *Variables* (indicated by **var**) to indicate the optimization variables with are to be determined later on by the optimizer;
- Objective functions (indicated by **minimize** or **maximize**) to announce a linear form as linear mathematical function of optimization variables;
- Constraints (indicated by **subject to**) to announce a linear (in-) equality form with respect to a linear mathematical function of optimization variables.

Beside this *AMPL* is endowed with other syntactical words, which we shall discuss in the context of our *AMPL models.*

3.3 The basic deployment planning model: sets, parameters and variables

We start with introducing the sets, the parameters and the optimization variables in the form of the *AMPL* program of table 2. The syntax words are printed in bolds. As illustrated by *Figure 5,* the network consists of *nodes, sites* and *tracks*; the names of these entities must be specified in the data phase.

By definition, a site always lies on a node; this is forced by the expression ***set*** *Sites* ***within*** *Nodes*. A track is a connection between two nodes in one driving direction. With ***set*** *TrkNds {Tracks}* ***ordered within*** *Nodes*, it is possible to assign, in the data phase, to each track two nodes ordered to be the first and the last. Note: a parking place is modelled as a track with the same site as origin and as destination.

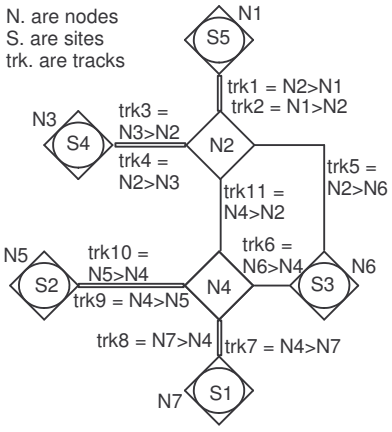

Figure 5. Example of a network

Next, expression ***set*** *Route { Sites, Sites }* ***ordered within*** *Tracks*, defines a route as a connection between two sites and is defined as sequence of tracks, to be specified in the data phase in the order of the driving direction. In the example, the route from *S4* to *S2* is to be specified by the sequence of tracks*: trk3, trk5, trk6, trk9*. Note: we have assumed that each pair of sites just two routes are selected, one for each driving direction.

Next, for each *n **in** Nodes*, the assignment (indicated by the symbol :=) of the expression *{ trk **in** Tracks: **last**(TrkNd[trk]) = n }* instructs the *AMPL compiler* to specify the corresponding *inTrks* as the set of all tracks which have *n* as endpoint. This is done in the data phase, when the required data is available. In a similar way the sets *outTrks* are specified. In the example, the ingoing tracks of node *N4* are *trk10*, *trk8* and *trk6*, whereas outgoing tracks are *trk11, trk9* and *trk7.*

Table 2. Declaration of the AMPL model (without capacity constraints)

AMPL model phase: declarations	remarks
set Nodes;	Set of network nodes.
set Sites **within** Nodes;	Set of sites = subset of nodes.
set Tracks;	Set of tracks.
set TrkNds { Tracks } **ordered within** Nodes;	Start and end nodes of tracks.
set Route { Sites, Sites } **ordered within** Tracks;	Route = sequence of tracks.
set inTrks { n **in** Nodes } := { trk **in** Tracks: **last**(TrkNd[trk]) = n };	Ingoing tracks of node.
set outTrks { n **in** Nodes } := { trk **in** Tracks: **first**(TrkNd[trk]) = n };	Outgoing tracks of node.
param T0 **integer** >= 0;	Start period of time axis.
param T1 **integer** > T0;	Start period of plan horizon.
param T2 **integer** > T1;	End period of plan horizon.
param T3 **integer** > T2;	End period of time axis.
param drvTm { Tracks } **integer** >= 1;	Driving times of the tracks.
param procTmIn { Sites } **integer** >= 0;	Proc.times incoming vehicles.
param procTmOut { Sites } **integer** >= 0;	Proc.times outgoing vehicles.
param jbTm { s1 **in** Sites, s2 **in** Sites } := (**if** s1 = s2 **then** 1 **else** procTmOut[s1] + procTmIn[s2] + **sum** { trk **in** Route[s1, s2] } drcTm[trk];	Integral execution times jobs. Note: job time intern transp = 1.
param lowReq { Sites, Sites, T1..T3 } >= 0;	Profile of minimal requests.
param upReq { Sites, Sites, T1..T3 } >= 0;	Profile of maximal requests.
param runAlloc { Sites, Sites, T0..T1 - 1 } >= 0;	Num. of freight carrying vehic.
param runEmt { Tracks, T0..T1 – 1 } >= 0;	Num. of driving empty vehicles.
param listBalance { Sites, Sites };	State of execution earlier plan.
var xAlloc { Sites, Sites, T1..T2 } >= 0;	Num. of vehicle to be allocated.
var yEmt { Tracks, T1..T2 } >= 0;	Num. of empty vehicles.
var zFail { Sites, Sites, T1..T2 } >= 0;	Num. of postponed allocations.

In the model the cycle-time is used as standard time-unit. The parameters *T0* to *T3* represent various time-points, taken as starting times of the corresponding cycles. The planning covers the time span over the cycles *T1* to *T2*. The past runs from *T0* to *T1 – 1*. Expression ***param** drvTm { Tracks } **integer** >= 1*, associates with each track a driving time. Note: the quasi driving time on a parking place is 1. In a similar way processing times for the arriving and for the outgoing transport are associated with the sites. With these data the integral durations of the transport jobs will be calculated, under the parameters *jbTm{s1 **in** Sites, s2 **in** Sites}*; in case *s1 = s2*, the job is just internal transportation on *s1* with a standard duration of 1; in case *s1' ≠ 's2*, the duration consists of the sum of the times for processing on the sites and the driving on the connecting tracks.

So far the sets and parameters will stay the same for each planning cycle, but the parameters in the following block must be updated every time. The lower and upper bounds of the cumulative demand for vehicles are modelled non-negatively as ***param** lowReq { Sites, Sites, T1..T3 }*, and ***param** upReq { Sites, Sites, T1..T3 }*, representing the numbers of from the first site to the second, during becoming available at the start of the cycles *T1, T1+1, …, T3*. Note:

when, in this, the first and the second sites are same, the quantity means hiring for use on that site. The numbers of vehicles which are allocated via earlier decisions minus the demand volumes from the past are taken into account with the help of ***param*** *listBalance { Sites, Sites }*. The numbers of transportation jobs in execution are represented by **param** *runAlloc { Sites, Sites, T0..T1 - 1 }*. The numbers of empty vehicles (driving or parking) will be specified under ***param*** *runEmt { Tracks, T0..T1 – 1 }*.
The decision variables on the numbers of vehicles, declared as ***var*** *xAlloc { Sites, Sites, T1..T2 } >= 0*, refer to a connection and a starting time-cycle. The decision variables on the numbers of empty vehicles, ***var*** *yEmt { Tracks, T1..T2 } >= 0*, refer to a track and the time-cycle when the empty trip starts. Finally, deficits in satisfying the lower bounds of the requests are introduces as ***var*** *zFail { Sites, Sites, T1..T2 } >= 0*; because positive values must be taken as failing, these variables must be fined in the objective function.

3.4 The basic deployment planning model: objective function and restrictions

The objective function and the constraints of the linear programming model are presented in table 3. In the objective function, *called relevant_costs*, only costs are taken into account that makes the difference. These include the costs of missed rewards because of the reduced tariffs for anticipating transport (these are charged at 1 cost unit per unit of driving time) and the cost of empty driving (charged at 5 cost units per unit of driving time).

Table 3. Objective functions and constraint of the AMPL model (without capacity constraints)

AMPL model phase: objective function and constraints	remarks

minimize relevant_costs
sum { s1 **in** Sites, s2 **in** Sites, T **in** T1..T2: s1 < s2 }
(99 * yFail[s1,s2,T] +
jbTm[s1,s2] (xAlloc[s1,s2,T] – lowReq[s1,s2,T]))
+ **sum** { trk **in** Tracks **diff** Parkings, T **in** T1..T2 }
5 * drvTm[trk] * yEmt[trk,T];
	Fictitious costs to be minimized:
	missed rewards reduced tariffs
	anticipating transport = 1;
	costs emty driving = 5 unit /km;
	failures are fined at 99.

subject to cum_bounds { s1 **in** Sites, s2 **in** Sites, T in T1..T2 }: **sum** { t **in** T1..T } lowReq[s1, s2, t] <= listBalance[s1, s2] + **sum** { t **in** T1..T } xAlloc[s1, s2, t] <= **sum** { t **in** T1..T } upReq[s1, s2, t];	Cumulative lower and upper bounds for allocation of vehicles.
subject to conserve_empties{ n **in** Nodes **diff** Sites, T **in** T1..T2}: **sum** { kOut **in** outTrks[n] } yEmt[kOut, T] = **sum** { kIn **in** Trks[n] } (**if** T – drvTm[kIn] < T1 **then** runEmt[kIn,T–drvTm[kIn]] **else** yEmt[kIn,T–drvTm[kIn]]);	Conservation condition on nodes without site: this is relevant for empty vehicles only.
subject to conserve_on_sites { s **in** Sites, T **in** T1..T2 }: (**sum** { kOut **in** outTrks[s] } yEmt[kOut, T]) + (**sum** { ss **in** Sites } xAlloc[s, ss, T] = (**sum** {kIn **in** inTrks[s] } (**if** T – drvTm[kIn] >= T1 **then** yEmt[kIn,T-drvTm[kIn]] **else** runEmt[kIn,T-drvTm[kIn]])) + (**sum** { sp **in** Sites } (**if** T – jbTm[sp,s] >= T1 **then** xAlloc[sp,s,T-jbTm[sp,s]] **else** runAlloc[sp,s,T-jbTm[sp,s]]));	Conservation condition on sites: this is relevant for both loaded vehicles, empty vehicles and parked vehicles.

The expression, ***sum*** *{trk* ***in*** *Tracks* ***diff*** *Parkings, T* ***in*** *T1..T2} 5 * drvTm[trk]*yEmt [trk, T]*, works as follows. The sum is executed over the values *5 * drvTm[trk] * yEmt[trk,T]*, with indices *trk* and *T* taken from the set *Tracks but not in Parkings,* and the set *T1, T1+1, …,T2*, respectively. Thus the expression between braces, following the syntactical word ***sum****,* specifies the index-values of the summation.
A programming constraint is announced by ***subject to***, followed by an index-expression in case the constraint is indexed. Concerning the cumulative lower and upper bounds for capacity allocation, called, *cum_bounds*, the constraint applies to each connection (site-to-site) and each time-point from *T1* to *T2*.
The conservation equalities require that, at any node, the sum of the outgoing vehicles plus the number of vehicles that will stay at that node must be equal to the number of incoming vehicles plus the number of vehicles which where already at that node. This applies separately for empty and for transporting vehicles. For the arriving vehicles a rather complex index calculation is needed because the travel must be taken into account.
This completes the *AMPL* definition phase of our basic model fleet deployment optimisation. In section 3.8 this model is extended with some additional constraints concerning the processing capacity.

3.5 Calculation experiments on optimized fleet deployment

To obtain an impression on the performance of the planning system we carry out a few experiments, where the sizes of the experiments are kept within the capacity of *AMPL's* student edition. Thus we confine ourselves to a model depictured by figure 6, with three sites (*A, B, C*) which are connected by seven tracks; three of these play the role as parking place. The driving times are: *A-to-B = 1, B-to-A = 1, B-to-C = 2 and C-to-B = 2*.

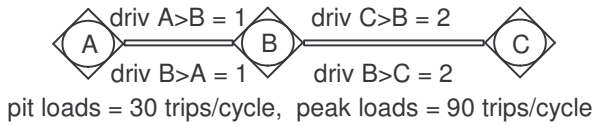

Figure 6. Calculation model with three sites

The pit loads are 30 jobs and the peak loads 90 jobs per time-cycle (the duration of a time-cycle is 1 hour). We have ruled out the possibility to use vehicles for internal transport on these sites. Concerning the demand two cases are studied: Case I shows a demand with two peaks and Case II shows one peak. Table 4 shows the time points of these peaks.

Table 4. Time point of the peak periods of the examples

	Case I: 2 peaks with duration 3 hours			**Case 2: 1 peak with duration 6 hours**		
	to A	**to B**	**to C**	**to A**	**to B**	**to C**
from A	-	06-08 18-20	06-08 18-20	-	13 - 18	10 - 15
from B	06-08 18-20	-	06-08 18-20	10 - 15	-	10-12 13-15
from C	06-08 18-20	06-08 18-20	-	04 - 09	13 - 18	-

For the margin to anticipate, four levels are studied: 0%, 20%, 40% and 60%. At the level of 20%, 20% of the demand may be served 1 and 2 periods earlier; at the 40% level, 40% may serve 1 to 4 periods earlier; at the 60% level, 60% of the demand may be served 1 to 6 periods earlier. At the 0% level, there is no anticipation. The plan horizon comprises 20 time-cycles. The optimization on each data set is 24 times repeated as a rolling horizon procedure, where each time the allocation on the first period is implemented.
Table 5 shows the results. It appears that anticipation reduces the number of vehicles to serve the demand drastically. For Case I these numbers are from 1080 to 590 vehicles and for Case II, from 1080 to 700. The column under *anticipation* shows the part of the demand for capacity which has been served in advance. The degree of empty driving is remarkable low: 1% to 2.7%; too low to observe any influence of anticipation.

Table 5. Number of required vehicles and average exploitation

	anticipation level (%)	**vehicles #**	**empty driv (%)**	**Parking (%)**	**job driving (%)**	**Anticipation (%)**
Case I	*00*	1080	1.0	49.1	49.9	-
	20	930	0.8	40.9	58.3	7
	40	720	0.6	23.1	76.3	28
	60	590	0.5	9.5	90.0	47
Case II	*00*	1080	2.5	41.7	55.6	-
	20	990	2.9	36.2	60.9	7
	40	840	3.2	24.8	71.9	31
	60	700	2.7	10.9	86.3	42

The simulation results suggest that the system might be very effective. However, the model-data of these examples are rather arbitrary. Simulations on realistic applications, such as the *RTS*, are necessary to evaluate the performance. However this is beyond the scope of this study.

3.6 Extension of the model with constraints on road capacity

The availability of roads is modelled by associating with each track a capacity profile over time; in table 6 this goes via parameter *trackCap*, declared by ***param*** *trackCap { Tracks, T1..T2 } >= 0*. A value 0 indicates that, on the time-period in question, the track is not available. To formulate these conditions properly, we need to know, for each track, the routes that pass through it. Of course this can be deduced from the data of table 2 and for that purpose the parameters *routViaTrack* and *drvTmToTrack* are introduced.

Table 6. Extension of the AMPL model with capacity bound on the roads

AMPL model	remarks
param trackCap { Tracks, T1..T2 } >= 0;	The capacities of the tracks.
set routeViaTrack { trk **in** Tracks } := { s1 **in** Sites, s2 **in** Sites: s1 <> s2 **and** trk **in** Route[s1, s2] };	Set of routes passing a given track trk.
param drvTmToTrack{s1 **in** Sites, s2 **in** Sites, trk **in** Route[s1,s2]} := procTmOut[s1] + **sum**{ k **in** Route[s1,s2]: **ord**(k) < **ord**(trk)} drvTm[k];	Driving times from a given site to arrive at a given track, driving to a given site.
subject to cap_restr_tracks { trk **in** Tracks, T **in** T1..T2 }: yEmt[trk, T] + **sum** { (s1, s2) **in** routeViaTrack[trk] } (**if** T – drvTmToTrack[s1, s2, trk] <T1 **then** (rumAlloc[s1,s2,T-drvTmToTrack[s1,s2,trk]]) **else** (xAolloc[s1,s2,T-drvTmToTrack[s1,s2,trk]]) <= trackCap[trk,T];	The capacity constraints on tracks.

Adding the model of table 6 to the model of table 2 and table 3, the number of variables remains the same, but the number of constraint increases. When some of the capacities are zero, modern *Linear Programming Solvers* will eliminate these zeros, putting the corresponding decision variables zero, before the actual optimisation starts.
In practical applications it might be necessary to add special linear conditions concerning the physical handling at some of the sites. For instance, this may apply to the transhipment capacities. In a similar way as table 6 shows, such extensions can be added easily.

3.7 Calculation complexity

Of course the size of this *linear programming problem* strongly depends on the number of sites and tracks and the number of time-periods, covered by the plan horizon. More specific from the model declaration one may deduce (see table 2):

- The number of decision variables = (2 * |Sites| ^{2} + |Tracks|) * (T2 + 1 – T1);
- The number of constraints = (|Sites| ^{2} + |Nodes|) * (T2 + 1 – T1).

For instance, a model without additional constraints, with 7 sites, 28 tracks, 12 nodes and 72 time-periods leads to 8064 decision variables and 3904 constraints. Dropping the integer-constraint on the solutions, such problems can be solved within a few minutes (using advanced solution techniques, the calculation complexity is polynomial related to the size of the problem).
Our experiments on the model without integer-constraints always produce integer solutions, which is what we need. Although there is no formal proof for, it conforms to the theory on linear minimal costs flow models with 0/1-constraints. Indeed, the model shows a similar structure. More precisely, consider the model (i.e. its basic form extended with capacity restrictions) in the following abstract form:

ILP_A:	minimize	$< p,x > + < q,y > + < r,z >$	(1)
	choosing	$x \in Int(R^k_+),\ y \in Int(R^m_+),\ z \in Int(R^n_+)$	(2)
	such that	$F(x,y) = 0$	(3)
		$C(x,y) \leq b$	(4)
		$d - z \leq Sx \leq a$	(5)

In this *Integer Linear Program*, denoted *ILP_A*, the decision variables *x, y* and *z* represent *xAlloc[.], yEmt[.])* and *zFail[.],* respectively. In this context we write $P^1(x,y,z)$ to refer to part of *(x,y,z)* corresponding to the first planning period of *(xAlloc[.], yEmt[.])* and *zFail[.]).*
In *IPL_A,* the constraints of (3) represent the conservation conditions, where *F* is a 0/1−matrix generated by the network structure. The constraints of (4) reflect all kind of capacity constraints (*C* is 0/1−matrix ≥ 0). The constraints of (5) represent the cumulative restrictions to satisfy the demand for vehicles (S is 0/1−matrix ≥ 0). The vectors *b, d, a* are non-negative integer; the vectors *p, q, r* are non-negative. We define *LP_A* as deduced from *ILP_A*, when dropping the integer restriction.
Observing that the objective function has a lower bound, the existence of a feasible solution for *ILP_A* implies the existence of optimal solutions for *ILP_A* and *LP_A.*
In general the calculation complexity of integer programs is excessive (i.e. non-deterministic polynomial related to the problem size), but in our case the problem seems to be manageable. To show this, let *(x,y,z)* be a solution of *LP_A* and let *(u,v,w)* be a corresponding dual solution related to (3), (4) and (5), respectively. Using *(x,y)* and *(v,w)* we deduce *LP_B*, as a *Lagrangian relaxation* of *LP_A*, with modified constraints:

LP_B:	*minimize*	$< p + v' C,(x,y)> + < q + w'S, y >$	(6)
	choosing	$x \in R^k_+,\ y \in R^m_+$	(7)
	such that	$F(x,y) = 0$	(8)
		$floor(x,y) \leq x \leq top(x,y)$	(9)

Note: *floor(x,y)* refers to the highest integer lower bound of *x* and *top(x,y)* to the lowest integer upper bound. From the Lagrangian duality theory we know that *(x,y)* also maximizes *LP_B*. Observing that *LP_B* is exactly a minimum costs flow problem with integer capacity constraints, the theory states that the *LP_B* has an optimal solution (x^o,y^o) which is integer valued. Moreover, algorithms exist which find such an (x^o,y^o) in a manageable time; see for instance Ahuja et al. in Nemhauser (1989) or Wolsey (1998). With these findings we propose the following procedure:

- Solve *LP_A*, let *(x,y,z)* be the optimal solution and let *(u,v,w)* be the dual;
- If $P^1(x,y,z)$ is integer, take *(x,y,z)* as deployment plan;
- Else use *(x,y) and (v,w)* to specify *LP_B;*
- Solve *ILP_B* and let (x^o,y^o) be an optimal solution, define $z^o = max\{0, d - Sx\}$;
- Take $P^1(x^o,y^o,z^o)$ as the first-period part of optimal solution of *ILP_A (?!)*;
- Add the resctriction $P^1(x,y,z) = P^1(x^o,y^o,z^o)$ to *LP_A,* solve this problem and take the optimal solution $(x^\#,y^\#,z^\#)$ as deployment plan, but if this *LP_A* is not solvable, take (x^o,y^o,z^o) as deployment plan.

We would like to take (x^o,y^o,z^o) directly as optimal in *ILP_A,* but unfortunately (x^o,y^o,z^o) not necessarily has to be feasible. Nevertheless there are reasons to take $P^1(x^o,y^o,z^o)$ as the first-period part of a fleet deployment plan and to neglect the integer-constraints of the succeeding periods. First of all, the optimization model does not claim to be an accurate model of planning practice; the setup is indented to be an adaptive optimizing heuristic. Both the cumulative delivery requirements and the capacity constraints are not immovable. However, the vehicle conservation constraints definitely are strict, but these are fully protected by the procedure. Secondly, the procedure is repetitive. Only the deployments of the first planning period, $P^1(x^\#,y^\#,z^\#)$, are transferred to the contract for the following period; the planned

deployments for the second and succeeding periods are indicative and will be revised in the following planning cycle.
Summarizing: in practice the procedure may produce acceptable deployment plans within manageable calculation times (also other algorithms might be considered, see Wolsey, 1998).

4. Job dispatching for criss-cross transport on terminals

The result of a categorical contract between the *fleet manager* and the *site manager* of a generic site *S* is a numerical specification of the numbers of vehicles which are allocated to be used for the transport from *S* to specific other sites and for the internal transport on *S* it selves. It is the responsibility of the *site manager* to allocate jobs to the vehicles in conformity of the contract. This section is meant to show that a similar linear programming approach as we propose for fleet deployment planning can be applied successfully for dynamic adaptive job-vehicle matching.

4.1 Principles of multi-period job dispatching

To optimize the use of vehicles in the context of criss-cross transport there are many heuristic methods. Most of them consider small numbers of potential job assignments, which often leads to inefficiencies. In the procedure we propose, the number of jobs and vehicles which are combined is increased drastically. The procedure works with repetitive time-cycle of 3 minutes, covering a rolling planning window of N time-cycles ($N = 5$, seems appropriate).
At the start of a new planning cycle two lists are updated: one on the open jobs for the coming N periods and one for the vehicles which are free or will become free within this horizon. The data for these lists are received real-time from the process control system and form the fleet deployment plan of the site.
As far as possible, the jobs and vehicles from these lists are matched but such, that the total time for the positioning of the vehicles (including empty driving) is minimal. Next, according to the *principle of postpone binding*, only those assignments are fixed and dispatched which are the necessary for the progress of the transportation process; the others are returned to the lists to be processed in the next planning cycle. Newly generated jobs and newly available vehicles also are added.
In the optimisation only costs are relevant that make the difference. Assuming that the vehicles are identical, the only difference is the empty trip between the endpoint of a preceding job and start point of the new job, giving rise to *positioning costs*. Of course these costs are zero in case the terminal position and the starting position are identical.

4.2 A linear programming model for anticipating adaptive job dispatchment

To support the job assignment we propose a simular approach as for the fleet deployment optimisation and in fact we use the same model in slightly modified form. In this case we have transhipment platforms (to be modelled as *sites*) and the internal roads and parking places (to be modelled as *tracks* and *nodes*). However, there are two differences. First, there are *external platforms* (possibly only one) functioning as gates, where vehicles externaly controlled enter, but internaly controlled depart. Secondly, the optimisation concerns the positioning (i.e. empty driving) and punctuality.

Thus we distinguish *internal and external platforms*. An external platform may represent an aggregated set of platforms and, in this way, an external platform may represent an entire external terminal of the network. The numbers of arriving and *departuring idle vehicles* on an external gate are controlled by the fleet manager. The externally arriving loaded vehicles are supposed to be bound to go to a specific internal platform.
In a simular way in fleet deployment planning, the numbers of jobs to be issued for execution must be not smaller than a given lower bound, but may not exceed a given upper bound. The lower bound arises from the latest execution times of the jobs, whereas the upper bound reflects the possibilities to execute jobs in advance.
In the numerical specification of the model, empty driving is subjected to a penalty of 1 cost-unit per time-cycle. In order to rule out undesirable solutions, wrongly postponed jobs, caused by a lack of vehicles, are penalised by 100 cost-units. Because the execution of jobs often shows random perturbations, it is reasonable to give priority to the optimisation on the first time-cycles over those of the last time-cycles. To that the objective function is equipped with an exponentially, over time, decreasing weight factor.
Before starting a new planning cycle, the state parameters refecting numbers of arriving loaded vehicles, arriving or departuring empty vehicles and the list of pending jobs must be updated. In the simulation experiments we have used an updating program for the case that the actual progress of execution runs in conformity of the plan, at least concerning the first time-cycle. However, in practice it will be necessary to adapt the state parameters to the actual progress.

4.3 Calculation experiments on optimal job assignment

The experiments are meant to obtain an idea about the performance of the optimisation procedure. They concern the example of the criss-cross transport of figure 7.

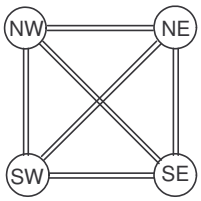

Figure 7. Layout for the example of criss-cross transport

To keep the size of the problem within the possibilities of the student edition of *AMPL*, the transport related to the external platform is left out. The capacities of the four platforms are identical and invariant over time. The static data of the layout are summarised in table 7.

Table 7. Example of criss-cross transport of the model of figure 2 (without external platform)

	driving times			
	NW	NE	SE	SW
NW	0	1	2	2
NE	1	0	2	2
SE	2	2	0	1
SW	2	2	1	0

issuing of jobs:			
each cycle, for	handling capacity	=	10 moves / cycle
each connection,	job anticipation (*shiftPre*)	=	4 cycles
with equal chance	job anticipation (maxPre)	=	4 jobs
selected	info. horizon (infoH)	=	3 or 4 cycles
0, 1 of 2	optimization horizon	=	6 cycles
	number of vehicles	=	40

During the simulation, for each time-cycle and on each connection, the numbers of jobs are 0, 1, or 2, randomly selected with equal probability. Thus, on the average, during each time-cycle 16 jobs are issued, where the standard-deviation is 3.3. These 16 jobs correspond with an average total driving time of 20 time-units and a vehicle occupation of 36 time-units per time-cycle (a time-unit, abbreviated *uct*, is equal to the duration of a time-cycle). Experimentally it appears that 40 vehicles are needed to process this workload without delays, and therefore, in the simulations, the number of vehicles is fixed at 40.
The typical parameters for the anticipation on job assignments are the limits on the number of jobs and the number of time-cycles (in the simulation 4 and 4, respectively). With respect to the *information horizon* we experiment with 3 and 4 time-cycles. The optimisation horizon is fixed at 6 periods. Table 8 summarises the simulation results on runs covering 50 time-cycles. The table shows the observed averages of the total occupation of the vehicles per time-cycle and the corresponding standard-deviations.

Table 8. Simulation results of the procedure of job assignment

information horizon	empty driving	parking	# alloc jobs	loaded driving
infoH = 3 cycles ahead	2.0 ±1.4 uct	3.4 ± 3.8 uct	15.6 ± 2.6 uct	19.0 ± 4.5 uct
infoH = 4 cycles ahead	1.8 ±1.0 uct	3.2 ± 3.1 uct	15.7 ± 2.1 uct	19.4 ± 4.3 uct

Taking into account the randomness in the generation of jobs, it appears that the fraction of unproductive vehicles (i.e. empty driving and parking) with respect of productive vehicles (i.e. job executing) is rather low (i.e. ± 14.5%). This also can be said for empty driving related to load driving vehicles (i.e. ± 9.5%). It also appears that the results for an information horizon of 4 cycles are slightly better than when the system operates with a horizon of 3 cycles.
Again we may conclude that these calculation experiment show positive results, but simulations on realistic applications are necessary to evaluate the performance.

5. Conclusion

This study concerns the deployment of large numbers of freight vehicles on a network consisting of roads and logistic sites. The transport on each centre is autonomously managed by a site manager. The vehicles are allocated by a fleet manager. In this context a coordination system is proposed to allocate vehicle capacity, such that the demand is served punctually and efficiently, but also that there is a win-win situation for each manager. The proposed coordination takes place at two levels: one to make an adaptive fleet deployment plan and one, compatible with this, to issue transportation jobs. Both use repetitive linear

programming model. The basic structure is a minimum cost flow model with additional constraints.
The fleet deployment planning is setup as a multi-period model, covering a planning horizon of 16 hours, using a time-cycle of 15 minutes. Typically the model explores the possibilities to anticipate on the request for vehicle capacity, within a limited margin. At each planning round only the outcomes of the first period are implemented, offering the basis for optimising job assignment. Also this model is multi-periodical. It may cover 15 minutes and uses a time-cycle of 3 minutes. The models are specified with the help of the well known modelling software *AMPL*. For realistic applications these models give rise to 5,000 to 10,000 decision variables. With modern programming techniques such models are manageable.
Simulations experiments are performed on small size models, involving 300 decision variables show that these models are promising. The nature of modelling with AMPL implies that the same models can be used for full sized problems.
Further R&D should cover the following topics: (i) simulations on models for potential applications; (ii) the development of generic software for the interaction between the fleet management and the site management; (iii) The development of business models, including methods to determine the tariffs.

Acknowledgements

The study was supported by the department of Transport, Logistics and Organisation of Delft University of Technology. The author acknowledges Bert van Wee, Caspar Chorus and the referees for their critical suggestions.

References

Ball, M.O., Magnanti, T.L., Monma, C.L. and Nemhauser, G.L. (1995). *Network models*. Handbooks in Operations Research and Management Science, Volume 7. North Holland.

Chen, Y., Leong, Y.T., Hg, J.W.C., Demir, E.K., Nelson, B.L. and Simchi-Levi, D. (1998). *Dispatching Automated Guided Vehicles in a Mega Container Terminal*. INFORMS Montreal, May 1998

Daganzo, C.F. (1991). *Logistics System Analysis*. Lecture notes in Economics and Mathematical Systems, Springer-Verlag, N 361, Berlin.

Duinkerken, M.B., Evers, J.J.M. and Ottjes, J.A. (2001). A simulation modelfor integrating quay transport and stacking policies on automated container terminals. *Proceedings of the 15th European Simulation Multiconference*. June 2001. Prague (SCS).

Ebben, M., Van der Zee, D.J. and Van der Heiden, M.C. (2004a). Dynamic one-way traffic control in automated transportation systems. *Transportation Res. Part B*, vol. 38, pp. 441-458.

Ebben, M., Van der Heijden, M.C., Hurrink, J. and Schutten, M. (2004b). Modelling of capacitated transportation systems for integral scheduling. *OR spectrum*, 26, pp.263-282.

Egbelu, P.J. (1993). Positioning of automated guided vehicles in a loop layout to improve response time. *European Journal of Operational Research*, vol. 71, pp. 32-44.

Evers, J.J.M. and Koppers, S.A.J. (1996). Automated guided vehicle traffic control at a container terminal. *Transportation Research A*, vol. 30, no. 1, pp. 21-34.

Evers, J.J.M., Van der Wielen, G.J.M., Duinkerken, M.B. and Ottjes, J.A. (1998). The jumbo container terminal: quay transport during ship loading. *TRAIL Research School*, Delft University of Technology, TRAIL/98/308.

Evers, J.J.M. and De Feijter, R. (2004). Centralized versus Distributed Feeder Ship Service: The case of the Maasvlakte harbour area of Rotterdam. *Transportation Planning and Technologies*, vol. 27, no. 5, pp. 367-384.

Fink, A. and Reiners, T. (2005, forthcoming) Modeling and solving the short-term car rental logistics problem. *Transportation Research Part E.*

Fourer, R., Gay, D.M. and Kernighan, B.W. (1993). *AMPL, A Modeling Language for Mathematical Programming*. Boyd & Fraser Publ. Comp.

Fourer, R., Gay, D.M. and Kernighan, B.W. (1990). A Modelling Language for Mathematical Programming. *Management Science*, vol. 36, no. 5.

Fourer, R., Gay, D.M. and Kernighan, B.W. (1997). *AMPL, with AMPL Plus Student Edition*. Duxbury Press, http://www.ampl.com.

Gemeentelijk Havenbedrijf Rotterdam. (1998). *Integrale verkenningen voor haven en industrie 2020*. Rotterdam.

Van der Heijden, M.C., Ebben, M., Gademan, N. and Van Harten, A. (2002) Scheduling vehicles in automated transportation systems. *OR spectrum,* 24.

Kim, K.H. and Bea., J.W. (2004). A look-ahead dispatching method for automated guided vehicles in automated port container terminals. *Transportation Science*, vol. 38, no. 2, pp. 224-234.

De Koster, R.B.M, Le-Anh, T. and Van der Meer. J.R. (2004). Testing and classifying dispatching rules in three real-world settings. *Journal of Operations Management*, vol. 22, pp. 369-386.

Kurstjens, S.T.G.L., Dekker, R., Dellaert, N.P., Duinkerken, M.B., Ottjes, J.A. and Evers. J.J.M. (1996). Planning of Inter Terminal Transport at the Maasvlakte. *TRAIL Proceedings, 2nd TRAIL Congress*.

Meermans, P.J.M. (2002). *Optimization of container handling systems*. Ph.D.Thesis, Tinbergen Institute 271, Erasmus University Rotterdam.

Murty, K.G., Li, J., Wan, Y. and Linn., R. (2005). A decision support system for operations in a container terminal. Decision Support Systems, vol. 39, pp. 309-332.

Nemhauser, G.L., Rinnooy Kan, A.H.G. and Todd, M.J. (1989). Optimization. *Handbooks in Operations Research and Management Science*, vol. 1, North Holland.

Qiu, L., Hsu, W.J., Huang, S.Y. and Wang, H. (2002). Scheduling and routing algorithms for AGVs: a survey. *Int. J. Prod. Res.*, vol. 40, no. 3, pp. 745-760.

Vis, I.F.A., De Koster, R., Roodbergen, K.J. and Peeters, L.W.P. (2001). Determination of the number of automated guided vehicles at a semi-automated container terminal. *Journal of the Operational Research Society,* vol. 52, pp 409-417.

Vis, I.F.A. and De Koster, R. (2003). Transhipment of containers at a container terminal: an overview. *European Journal of Operations Research*, vol. 147, pp. 1-16.

Vis, I.F.A. (2004). Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operations Research*. Article in press, available online.

Van der Meer, R. (2000). Operational control of internal transport. *ERIM Ph.D series Research in Management 1*.

Wolsey. L.A. (1998). Integer Programming. *Wiley Interscience Series in Discrete Mathematics and optimization*, 1998.