Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Model Predictive Scheduling for Container Terminals

F.B. van Boetzelaer, T.J.J. van den Boom*, R.R. Negenborn*

Abstract— **In container terminals ships are loaded and unloaded. In this paper the problem of scheduling quay cranes, stack cranes, and vehicles used for the transport of containers between quay side and stack is discussed. In particular, we propose to model the container transport in a terminal using a switching max-plus linear model description to account for the synchronization between the cranes and the vehicles. The scheduling problem is formulated using this formalism and subsequently recast as a mixed integer linear programming problem. We apply the developed techniques in a number of test cases to illustrate the potential of the approach.**

I. Introduction

The competitiveness of seaport container terminals is mainly determined by the time in port for ships (transshipment time), and the rates for loading and discharging [12], [15], [16]. Container terminals generally utilize quay cranes to load and unload ships, specialized equipment to store containers in large stacks, and vehicles to transport containers between the quay side and the stacks. An unloading cycle consists of three distinct steps: the unloading by a quay crane, the transport from the crane to the stack, and finally the stacking. The dynamics of this transportation system as a whole can be influenced by choosing the vehicle that is assigned to each of the container jobs, and by choosing the order in which vehicles and stacking cranes handle their containers. This leads to a scheduling problem.

Since the preliminary work of Johnson [6] on scheduling problems, a lot of papers have discussed such problems, and several recent books present general surveys on the topic [11]. In the case of a container terminal, the set of operations may vary over a limited set of possible sequences of operations. One way to determine the optimal sequence of operations is by using an integrated generic hybrid flow shop and optimal control approach, as in [17]. Another approach is to consider flows instead of individual containers and use a receding flow scheduling approach, as in [1], followed by a translation from flows to individual equipment moves. In this paper, we investigate how, at the individual container level, specific characteristics of the sequence of operations can be taken into account when using a different approach: The semi-cyclic system dynamics considered considered in this paper are linear in the Max-Plus algebra. This means that the system can be described by switching max-plus linear (SMPL) models [14], which can switch between different modes representing alternative decisions or circumstances. In this paper we extend the work of [10], [13] for scheduling of semi-cyclic discrete event systems.

The use of SMPL systems in the scheduling procedure is motivated as follows: SMPL systems are appropriate for simulating the system by analyzing the evolution of its state. There are many system theoretical results for max-plus linear systems in literature. We can use these for finding bottlenecks in the scheduling process and for finding good initial scheduling values by using system properties, such as the max-plus eigenvalue and eigenvectors. Due to the close relation between a max-plus linear model and the graph representation of the system, graph based methods can be used in the scheduling procedure (see [9]) by transforming the implicit representation of an SMPL model into an explicit form from which the computation time may be reduced significantly [7].

The goal of this paper is to reduce transshipment times in container terminals by using Model Predictive Scheduling (MPS), which optimizes future control decisions by using predictions of the future behavior of the system. We aim for reactive operational scheduling which means that based on observations of the systems behavior we can reschedule the routes of the containers to optimize the performance of the system.

This paper is organized as follows. In Section II, we give a concise introduction to max-plus algebra. In Section III, we derive a max-plus linear model for container transport. In Section IV, the scheduling problem will be recast as a mixed integer linear programming problem, and in Section V we give some test cases. Conclusions and future research are given in Section VI.

II. Max-plus algebra

We start by summarizing the most important features of the max-plus algebra. The max-plus algebra consists of the set $\mathbb{R}_{\max}$, and two associated operations, the max-plus addition and the max-plus multiplication [13]. The set $\mathbb{R}_{\max}$ and the scalar max-plus-algebraic addition and multiplication are defined, respectively, as:

$$\mathbb{R}_{\max} = \mathbb{R} \cup \{-\infty\}$$
$$x \oplus y = \max(x, y)$$
$$x \otimes y = x + y.$$

F.B. van Boetzelaer and T.J.J. van den Boom are with the Delft Center for Systems and Control, R.R. Negenborn is with the Department of Maritime and Transport Technology, both in the Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology, Delft, The Netherlands.

* Corresponding authors: a.j.j.vandenboom@tudelft.nl, r.r.negenborn@tudelft.nl

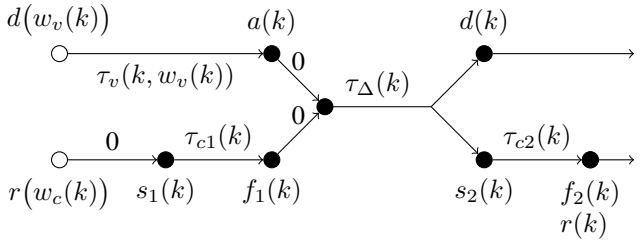

Fig. 1. Graph representing one container transfer cycle.

Furthermore, we define:

$$[\mathbf{A} \oplus \mathbf{B}]_{i,j} = a_{i,j} \oplus b_{ij} = \max(a_{ij}, b_{ij}) \tag{1}$$

$$[\mathbf{A} \otimes \mathbf{C}]_{i,j} = \bigoplus_{k=1}^{n} a_{ik} \otimes c_{kj} = \max_{k=1,\ldots,n} (a_{ik} + c_{kj}) \tag{2}$$

$$[\mathbf{A} \odot \mathbf{B}]_{i,j} = a_{ij} \otimes b_{ij} = a_{ij} + b_{ij}, \tag{3}$$

for matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}_{\max}^{m \times n}$ and $\mathbf{C} \in \mathbb{R}_{\max}^{n \times p}$. The neutral element of max plus addition ϵ, and the neutral element of max-plus multiplication e are defined as:

$$\epsilon = -\infty \qquad e = 0.$$

III. Modeling of container transport

In this section we model the relevant operations that take place inside a container terminal and rewrite these in a max-plus linear model. We consider a container terminal that utilizes multiple quay cranes to unload ships, multiple stack cranes to store containers in stacks, and multiple vehicles to transport containers between quay cranes and stacks. The sequence of events that occurs when one container is transferred between a crane and a vehicle will be called a container transfer cycle. Such a cycle is characterized by three processes. Firstly the vehicle needs to travel from its previous location to the crane where the container transfer will take place. At the same time, the crane needs to prepare itself for the container transfer. Once the vehicle has arrived and the crane has finished preparing itself, the container can be transferred between the crane and the vehicle. Finally, the crane may need some time to finish handling this container before it can start its next cycle.

These processes are represented by the graph in Fig. 1. In this graph d and a represent a vehicle's departure and arrival times; s_1, f_1, s_2 and f_2 represent the times at which a crane starts and finishes its processes before and after the container has been transferred; r represents the time at which the crane is ready to start its next cycle; w_v and w_c are the event counters of the preceding cycles of the vehicle and the crane; τ_v is the vehicle travel time; τ_{c1} and τ_{c2} are the times required by the crane to handle the container, and τ_Δ is the time required to transfer the container between the crane and the vehicle. Because the vehicle travel time depends on the distance between the current and the preceding crane, τ_v is a function of both k and $w_v(k)$. The closed nodes in this graph represent all events which belong to cycle k, while the open nodes are events belonging to the preceding cycles.

Using the relation between directed graphs and max-plus algebra [5], the following equations for $r(k)$ and $d(k)$ can be determined:

$$r(k) = \tau_{c2}(k) \otimes \tau_\Delta(k) \otimes \Big(\tau_{c1}(k) \otimes r\big(w_c(k)\big) \oplus \tau_v(k, w_v(k)) \otimes d\big(w_v(k)\big)\Big) \tag{4}$$

$$d(k) = \big(-\tau_{c2}(k)\big) \otimes r(k). \tag{5}$$

By substituting this last equation into the first equation, we obtain the following equation describing the whole cycle:

$$r(k) = \tau_{c2}(k) \otimes \tau_\Delta(k) \otimes \Bigg(\tau_{c1}(k) \otimes r\big(w_c(k)\big) \oplus \tau_v(k, w_v(k)) \otimes \Big(\big(-\tau_{c2}\big(w_v(k)\big)\big) \otimes r\big(w_v(k)\big)\Big)\Bigg). \tag{6}$$

Using a receding horizon principle the schedule for the complete loading/unloading task is not calculated at once, but in several iterations. In every iteration the schedule is calculated for a limited number of containers instead of all containers in the complete loading/unloading scheduling task. There are two main reasons for using the model predictive scheduling method:

1) The scheduling task may contain many jobs. The computation time of the optimal solution increases as the number of scheduling variables increases. The negative impact of the computation time can be avoided by using the receding horizon principle.
2) We aim for reactive operational scheduling, which means that based on observations of the system's behavior we can reschedule the (un)loading of the containers to optimize the performance.

To make a prediction we stack the ready times of all cycles in the prediction interval into one state vector $\mathbf{x}$:

$$\mathbf{x}_0 = \begin{bmatrix} r(1) \\ \vdots \\ r(N_{\text{past}}) \end{bmatrix} \quad \mathbf{x}_{\text{operational}} = \begin{bmatrix} r(N_{\text{past}}+1) \\ \vdots \\ r(N) \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_{\text{operational}} \end{bmatrix},$$

where N_{past} is the last cycle which lays in the past, and N is the total number of containers in the prediction interval. The system can be controlled by choosing the preceding cycles. This will be done by assigning the values of the elements of the following matrices:

$$[\hat{\mathbf{W}}_v]_{k,j} = \begin{cases} e & \text{if the vehicle handled cycle } j \text{ directly before cycle } k \\ \epsilon & \text{otherwise} \end{cases}$$

$$[\hat{\mathbf{W}}_c]_{k,j} = \begin{cases} e & \text{if the crane handled cycle } j \text{ directly before cycle } k \\ \epsilon & \text{otherwise.} \end{cases}$$

Now the ready times of the preceding cycles for the crane and vehicle can be found by multiplying these matrices in the max plus sense with the state vector $\mathbf{x}$:

$$r\big(w_c(k)\big) = [\hat{\mathbf{W}}_c \otimes \mathbf{x}]_k$$

$$r\big(w_v(k)\big) = [\hat{\mathbf{W}}_v \otimes \mathbf{x}]_k.$$

As mentioned above, the vehicle travel time τ_v depends on the cranes between which the vehicle travels. Therefore the following matrices are defined:

$$[\hat{\mathbf{C}}]_{j,k} = \begin{cases} e & \text{if cycle } k \text{ takes place at crane } j \\ \epsilon & \text{otherwise} \end{cases}$$

$$[\mathbf{T}]_{i,j} = \text{the vehicle travel time from crane } i \text{ to crane } j.$$

Now $\tau_v(k)$ can be defined by using $\hat{\mathbf{C}}$ and $\hat{\mathbf{W}}_v$ to select the correct element from $\mathbf{T}$:

$$\tau_v(k, w_v(k)) = \hat{\mathbf{W}}_v \otimes \hat{\mathbf{C}}^{\mathrm{T}} \otimes \mathbf{T} \otimes \hat{\mathbf{C}}.$$

The system equation given above can then be written as one max-plus linear matrix equation:

$$\mathbf{x} = (\hat{\mathbf{W}}_c \odot \mathbf{A}_c \oplus \hat{\mathbf{W}}_v \odot \mathbf{A}_v) \otimes \mathbf{x} \oplus \mathbf{x}_0, \tag{7}$$

where

$$\mathbf{A}_c = \mathbf{T}_{c1} \odot \mathbf{T}_{c2} \odot \mathbf{T}_\Delta$$

$$\mathbf{A}_v = (\hat{\mathbf{C}}^{\mathrm{T}} \otimes \mathbf{T} \otimes \hat{\mathbf{C}}) \odot (-\mathbf{T}_{c2})^{\mathrm{T}} \odot \mathbf{T}_{c2} \odot \mathbf{T}_\Delta$$

$$\mathbf{T}_{c1} = \begin{bmatrix} \tau_{c1}(1) & \cdots & \tau_{c1}(1) \\ \vdots & \ddots & \vdots \\ \tau_{c1}(N) & \cdots & \tau_{c1}(N) \end{bmatrix}$$

$$\mathbf{T}_{c2} = \begin{bmatrix} \tau_{c2}(1) & \cdots & \tau_{c2}(1) \\ \vdots & \ddots & \vdots \\ \tau_{c2}(N) & \cdots & \tau_{c2}(N) \end{bmatrix}$$

$$\mathbf{T}_\Delta = \begin{bmatrix} \tau_\Delta(1) & \cdots & \tau_\Delta(1) \\ \vdots & \ddots & \vdots \\ \tau_\Delta(N) & \cdots & \tau_\Delta(N) \end{bmatrix}.$$

IV. Mixed integer linear programming formulation

The model derived above is formulated in the max-plus algebra. In this section we show how this problem can be recast as a mixed integer linear programming (MILP) problem.

A. Conversion to linear constraints

As the first step the system equations will be rewritten in terms of inequality constraints in the conventional plus-times algebra. The control matrices $\hat{\mathbf{W}}_v$ and $\hat{\mathbf{W}}_c$ are binary in the max-plus sense, i.e., $[\hat{\mathbf{w}}]_j \in \{\epsilon, e\}$. Since ϵ is not an element of $\mathbb{R}$, these matrices will be replaced by conventional binary matrices $\bar{\mathbf{W}}_v$ and $\bar{\mathbf{W}}_c$. This can be done by introducing a finite number $\beta \ll 0$. The max-plus control matrices in max-plus multiplications can then be replaced by their plus-times counterpart by writing:

$$\begin{aligned} \hat{\mathbf{w}}^{\mathrm{T}} \otimes \mathbf{x} &= \max_k([\hat{\mathbf{w}}]_{ik} + [\mathbf{x}]_{kj}) \\ &= \max_k([(\beta - \beta\bar{\mathbf{w}}]_{ik} + [\mathbf{x}]_{kj}) \\ &= (\beta - \beta\bar{\mathbf{w}}^{\mathrm{T}}) \otimes \mathbf{x}. \end{aligned}$$

Using this method one can rewrite (7) as an expression in the conventional algebra, by replacing all max-plus operations with their plus-times equivalent:

$$\begin{aligned} [\mathbf{x}]_k = \max_{j=1,\ldots,N}(&\beta - \beta[\bar{\mathbf{W}}_c]_{k,j} + [\mathbf{A}_c]_{k,j} + [\mathbf{x}]_j, \\ &\beta - \beta[\bar{\mathbf{W}}_v]_{k,j} + [\mathbf{A}_v]_{k,j} + [\mathbf{x}]_j, [\mathbf{x}_0]_k). \end{aligned}$$

This expression finally leads to the linear constraints:

$$\begin{aligned} [\mathbf{x}]_k &\geq \beta - \beta[\bar{\mathbf{W}}_c]_{k,j} + [\mathbf{A}_c]_{k,j} + [\mathbf{x}]_j \\ [\mathbf{x}]_k &\geq \beta - \beta[\bar{\mathbf{W}}_v]_{k,j} + [\mathbf{A}_v]_{k,j} + [\mathbf{x}]_j \\ [\mathbf{x}]_k &\geq [\mathbf{x}_0]_k. \end{aligned} \tag{8}$$

B. Additional constraints

Not all combinations of the decision variables represent a valid schedule. Some examples of invalid schedules are schedules in which cycles are preceded by multiple other cycles at either the crane or the vehicle, cycles where containers are unloaded from a different vehicle than they where loaded onto, and cycles where vehicles are loaded with more containers than they can carry. In this section some constraints are introduced that will limit the decision variables to their allowed space.

Each operational cycle has exactly one preceding cycle at the crane and at the vehicle. This is represented by:

$$\sum_j [\bar{\mathbf{W}}_c]_{k,j} = 1 \ \forall k > N_{\text{past}} \tag{9}$$

$$\sum_j [\bar{\mathbf{W}}_v]_{k,j} = 1 \ \forall k > N_{\text{past}}. \tag{10}$$

Each cycle can only precede one other cycle at the crane and the vehicle. The last cycles that are executed, however, precede none of the other cycles. This is represented by:

$$\sum_k [\bar{\mathbf{W}}_c]_{k,j} \leq 1 \tag{11}$$

$$\sum_k [\bar{\mathbf{W}}_v]_{k,j} \leq 1. \tag{12}$$

The preceding cycle for the crane cannot take place at a different crane:

$$[\bar{\mathbf{W}}_c]_{k,j} = 0 \ \forall k, j \text{ where } [\hat{\mathbf{C}}]_{*,k)} \neq [\hat{\mathbf{C}}]_{*,j}. \tag{13}$$

At the quay crane, the order in which containers are handled is fixed. Therefore the preceding cycle is known for all quay cranes. Let $k_{\text{preceding}}(k)$ be the preceding cycle at the crane for cycle k. Now one can add the following set of constraints:

$$[\bar{\mathbf{W}}_c]_{k,j} = 1 \qquad \begin{array}{l}\text{if } j = k_{\text{preceding}}(k) \text{ and cycle } j \\ \text{takes place at a quay crane.}\end{array} \tag{14}$$

If the vehicles can only carry one container at a time, each container must be unloaded from the vehicle directly after it was loaded onto it. Define the variable $k_{\text{load}}(k)$ as the cycle in which the container which is to be unloaded in cycle k is loaded onto the vehicle. Now one can introduce the following sets of constraints:

$$[\bar{\mathbf{W}}_v]_{k,k_{\text{load}}(k)} = 1 \qquad \forall \text{ vehicle unloading cycles } k. \quad (15)$$

For vehicles with a carrying capacity larger than one the problems described above need to be solved in a different way. This can be done by keeping track of the containers loaded onto each of the vehicles. Therefore the following variables are defined:

$$[\bar{\mathbf{L}}]_{k,j} = \begin{cases} 1 & \text{if the container which is loaded onto a vehicle in cycle } j \text{ is carried by the current vehicle after cycle } k \\ 0 & \text{otherwise.} \end{cases}$$

During each cycle only one container is loaded or unloaded. Therefore only one element of $\bar{\mathbf{L}}$ changes during each cycle, while the other elements are equal to the corresponding elements in $\bar{\mathbf{W}}_v \cdot \bar{\mathbf{L}}$. This leads to the following equations for $\bar{\mathbf{L}}$:

$$[\bar{\mathbf{L}}]_{k,j} = \begin{cases} 1 & \text{if } k \text{ is a loading cycle and } j = k \\ 0 & \text{if } k \text{ is an unloading cycle and } j = k_{\text{load}}(k) \\ [\bar{\mathbf{W}}_v \cdot \bar{\mathbf{L}}]_{k,j} & \text{otherwise.} \end{cases}$$

The expression $\bar{\mathbf{W}}_v \cdot \bar{\mathbf{L}}$ results in nonlinear constraints. These can be converted to a set of linear constraints by introducing auxiliary variables $\delta_l(k)$:

$$[\bar{\mathbf{W}}_v \cdot \bar{\mathbf{L}}]_{k,j} = \sum_i [\delta_l(k)]_{i,j}. \quad (16)$$

The auxiliary variables δ_l can finally be given their values by introducing three linear inequality constraints for each auxiliary variable [2]:

$$\begin{aligned} -[\bar{\mathbf{L}}]_{i,j} + [\delta_l(k)]_{i,j} &\leq 0 \\ -[\bar{\mathbf{W}}_v]_{k,i} + [\delta_l(k)]_{i,j} &\leq 0 \\ [\bar{\mathbf{L}}]_{i,j} + [\bar{\mathbf{W}}_v]_{k,i} - [\delta_l(k)]_{i,j} &\leq 1. \end{aligned} \quad (17)$$

The expression for $\bar{\mathbf{L}}$ can now be written as:

$$[\bar{\mathbf{L}}]_{k,j} = \begin{cases} 1 & \text{if } k \text{ is a loading cycle and } j = k \\ 0 & \text{if } k \text{ is an unloading cycle and } j = k_{\text{load}}(k) \\ \sum_i [\delta_l(k)]_{i,j} & \text{otherwise.} \end{cases}$$

Now one can make sure that each container is only unloaded after it is loaded on the selected vehicle, by adding the following set of constraints:

$$\sum_i [\delta_l(k)]_{i,k_{\text{load}}(k)} = 1 \ \forall \text{ unloading cycles } k. \quad (18)$$

In order to make sure that the load capacity is not exceeded the following set of constraints can be added:

$$\sum_j [\bar{\mathbf{L}}]_{k,j} \leq l_{\max} \ \forall \text{ loading cycles } k, \quad (19)$$

where $l_{\max}$ is the load capacity.

Test case	Ships	QCs	SCs	Vehicles	Cycles
1	1	1	3	2	8
2	1	2	6	4	16
3	1	3	9	6	24
4	2	4	12	8	32
5	2	5	15	10	40
⋮	⋮	⋮	⋮	⋮	⋮

TABLE I
USED NUMBER OF EQUIPMENT IN THE VARIOUS TEST CASES (QCS = QUAY CRANES; SCS = STACKING CRANES).

C. Cost function

In order to complete the definition of the model predictive scheduling (MPS) problem, a cost function should be defined. The overall objective of the MPS problem is the minimization the weighted sum of the time required by the quay cranes to handle all containers for each ship. Because the time required by each quay cranes to handle all of its remaining containers might not be equal for all quay cranes, a penalty factor $[\mathbf{p}]_j$ will be defined for the time required to finish each cycle. Now the following cost function can be defined:

$$J(t) = \sum_{i=1}^{n_{\text{ships}}} \max_{j \in \mathcal{C}_i(t)} \Big(\big([\mathbf{x}(t)]_j - t \big) \cdot [\mathbf{p}(t)]_j \Big),$$

where $\mathcal{C}_i$ is the set of cycles in which containers for ship i are handled. This can be written as a linear cost function by introducing a set of linear constraints:

$$J(t) = \min \sum_i r_{\max,i} \quad (20)$$

subject to

$$\big([\mathbf{x}(t)]_j - t \big) \cdot [\mathbf{p}(t)]_j \leq r_{\max,i}(t) \forall j \in \mathcal{C}_i. \quad (21)$$

The final MILP problem can now be formulated a s follows:

$$\min_{r_{\max,i}, \mathbf{x}, \bar{\mathbf{W}}_c, \bar{\mathbf{W}}_v, \bar{\mathbf{L}}, \delta_l} \sum_i r_{\max,i},$$

subject to inequality constraints (8)–(19), (21), and some equality constraints that are related to past events, i.e., the past state x_0, and the part of the control decision variables $\bar{\mathbf{W}}_c, \bar{\mathbf{W}}_v, \bar{\mathbf{L}}, \delta_l$ that have already been implemented (and therefore cannot be changed any more).

V. EXPERIMENTS

In order to test the performance of the derived algorithms, various test cases are defined. The time constants are loosely based on the test cases mentioned in [8], [4] and [3]. In particular, τ_{c1}, the time required by the cranes to handle each container, will be 75 seconds for all cycles taking place at a quay crane, and 100 seconds for all cycles taking place at a stacking crane (SC). $\tau_{\triangle}$, the time required to transfer a container between a crane and a vehicle, will be 30 seconds for all cycles.

Fig. 2 shows the layout of the terminal used in the test cases. It can be seen in this figure that the vehicles follow a cross-lane type guidepath. The vehicles are assumed to travel

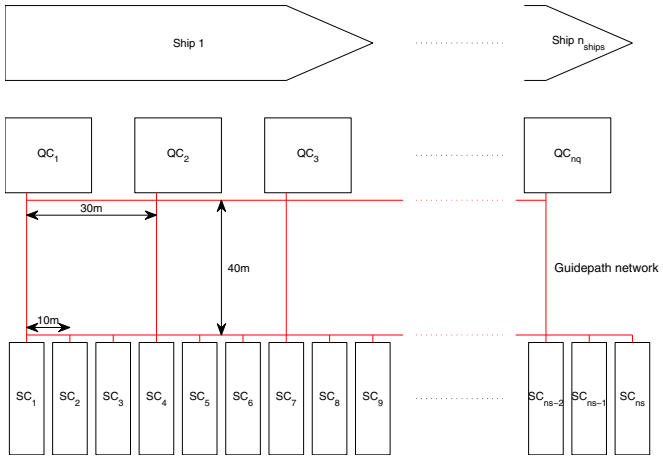

Fig. 2. Layout of the terminal used in the various test cases.

through this terminal with a speed of 3 m/s. In practice most vehicles will have a higher maximal speed than 3 m/s. The constant speed assumed here will compensate for the time required for acceleration, deceleration and waiting time when another vehicle travels through the same parts of the terminal. The total number of cranes and vehicles is chosen different for each of these test cases, as in Table I.

The system will be initialized by adding a set of dummy cycles. The ready times of these cycles determine the times at which the cranes and vehicles are available for the first time, and will be chosen from a uniform distribution of $[0, 30]$ seconds. The values of the vectors $\bar{\mathbf{W}}_c$, $\bar{\mathbf{W}}_v$ and $\bar{\mathbf{C}}$ for these dummy cycles will be chosen in such a way, that the initial locations of the vehicles are evenly distributed along the cranes.

A. Greedy algorithm

As a comparison for the system's performance, a simple heuristic vehicle dispatching algorithm will be introduced. This algorithm, called a greedy algorithm, can be found in literature in, e.g., [3], [8]. The greedy algorithm is a recursive algorithm that schedules one container per iteration. This algorithm can be described by the following steps:

1) Find the quay crane with the lowest ready time from the set of quay cranes which still have unscheduled cycles.
2) If the cycle for this quay crane is an unloading cycle, find the crane at which the container is to be loaded. If the quay crane cycle is a loading cycle, find the unloading crane.
3) Find the set of vehicles which can arrive in time at the loading crane.

4a) If the set is not empty, select from this set the vehicle which will arrive latest at the loading crane.

4b) Else, select the vehicle which can arrive earliest at the loading crane

4) If there are still unscheduled cycles, return to step 1.

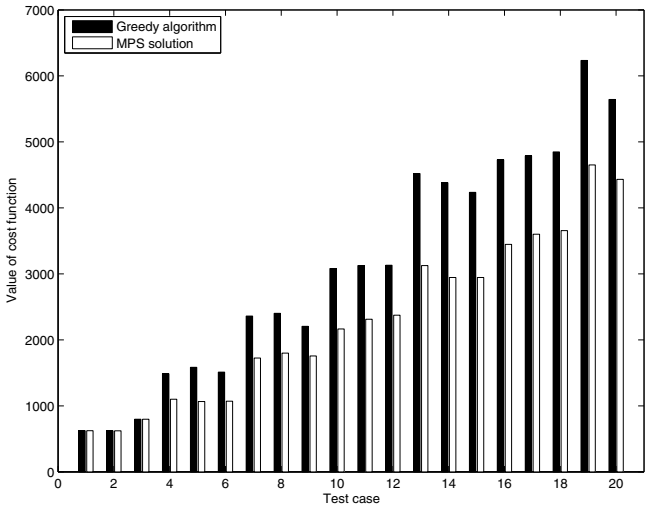

Fig. 3. Cost function value for greedy algorithm and MILP solution for various test cases given in Table I.

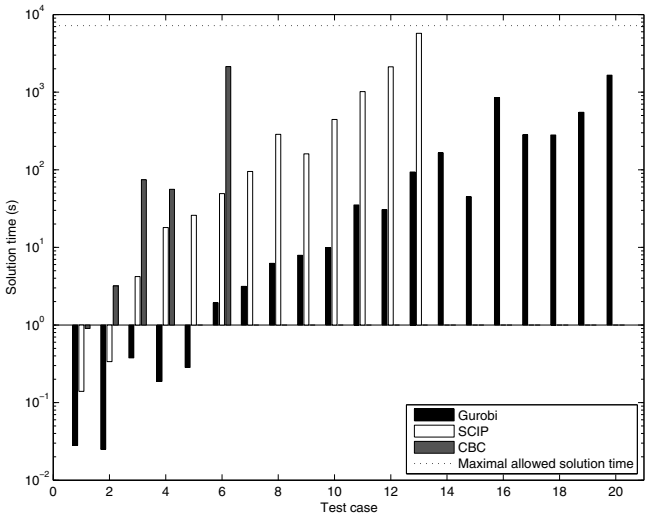

Fig. 4. Solution time of various MILP solvers for various test cases given in Table I, with a maximal allowed solution time of 7200 seconds.

B. Cost function value

Because the objective function is a measure of the total transshipment time, the cost function will be used to assess the performance of both the MILP solution and the greedy algorithm. Because the MPS solution is based on this cost function, it will by definition have better or equal performance compared to the greedy algorithm. Fig. 3 shows the cost function value for both solutions for various test cases. It can be seen that there is no difference in cost function value between the greedy algorithm and the MPS solution for test cases one to three. However, for all larger test cases the MPS solution outperforms the greedy solution by 25-50%.

C. Calculation time

In order to put the solution to the MILP problem to use, one should be able to find it within a reasonable time. The solution time for MILP problems is highly dependent on the used hardware and MILP solver. Several different MILP

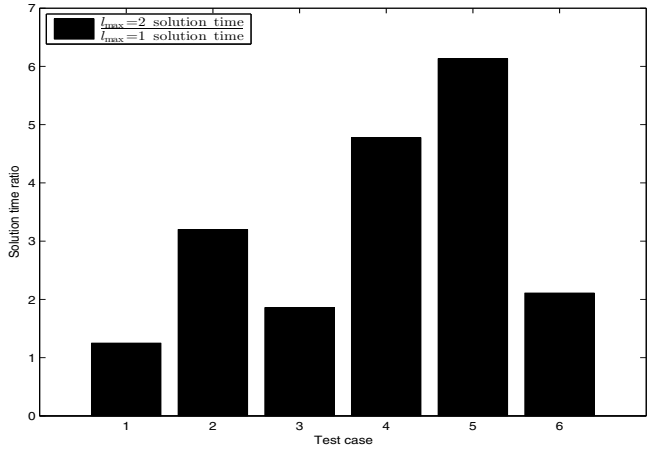

Fig. 5. Ratio of solution times required by the Gurobi solver for both MILP problems, for various test cases given in Table I.

solvers have been used to find the optimal solution to the first 20 test cases. This was done on a computer with a AMD Athlon 7750 dual core processor running at 2.7 GHz, with 3.25 Gb of RAM installed, running a 32 bits Windows 7. The solvers where given 7200 seconds to find the solution to each of of the test cases. The results of this experiment can be found in Fig. 4. In this figure the missing data points represent occurrences where the solver did not find a solution within 7200 seconds. In this figure it can be seen that the required solution time grows more than linearly with the number of predicted cycles for all solvers. In this figure it can furthermore be seen that there is a significant difference in solution time between the used solvers. This difference is illustrated by the solution times for test case 5: The CBC solver fails to find a solution in time, while the Gurobi solver only requires 0.28 seconds to find an optimal solution.

D. Multi-load case

The MILP problem for vehicles with a carrying capacity larger than one uses significantly more variables than the problem for vehicles with a unit load capacity. The additional variables $\bar{\mathbf{L}}$ and δ_l are by definition $\in \{0, 1\}$. However, these variables are already constrained to these integer values by the inequality constraints derived above. It is therefore not necessary to use integrality constraints for these variables in the MILP problem. This reduces the complexity of the optimization problem dramatically. There is one other complicating factor for the implementation of the MPS problem for vehicles with a larger carrying capacity. The δ_l matrices contain a total of $N_{\text{operational}} \cdot N^2$ variables, and 3 constraints are added for each of these variables. This high number of variables and constraints has a large impact on the amount of computer memory required by the MILP solver. In our case the MILP problem has been solved on a 32-bit operating system, and the system is unable to allocate enough memory to solve the problem for the test cases 7 and higher. Therefore the MILP could only be solved for the first 6 test cases, as can be seen in Fig. 5.

VI. Conclusions and future research

In this paper we have derived a scheduling procedure for container terminals. We have used switching max-plus linear models to describe the system. The scheduling problem can now be recast as a mixed integer linear programming problem. In future research we will look at the reduction of the computational effort. One way in which this problem will be addressed is by a priori eliminating certain combinations of binary decision values that will lead to infeasible schedules. Moreover, we will investigate the integration of the scheduling approach proposed here into a framework that also considers the operational control of the container terminal equipment itself.

Acknowledgment

This research is supported by the VENI project "Intelligent multi-agent control for flexible coordination of transport hubs" (project 11210) of the Dutch Technology Foundation STW.

References

[1] A. Alessandri, S. Sacone, and S. Siri. Modelling and optimal receding horizon control of maritime container terminals. *Journal of Mathematical Modelling and Algorithms*, 6:109–133, 2007.

[2] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35:407–427, 1999.

[3] Y. L. Cheng, H. C. Sen, K. Natarajan, C. P. Teo, and K. C. Tan. Dispatching automated guided vehicles in a container terminal. In *Supply Chain Optimization*. Springer US, 2005.

[4] M. B. Duinkerken and J. A. Ottjes. A simulation model for automated container terminals. In *Proceedings of the Business and Industry Simulation Symposium (ASTC 1999)*, 2000.

[5] B. Heidergott, G. J. Olsder, and J. W. van der Woude. *Max Plus at Work: Modeling and Analysis of Synchronized Systems*. Princeton Series in Applied Mathematics. Princeton University Press, 2006.

[6] S.M. Johnson. Optimal two- and three-stage production schedule with setup times included. *Naval Research Logistics Quarterly*, 1(1):61–68, 1954.

[7] B. Kersbergen, T. van den Boom, and B. De Schutter. Reducing the time needed to solve the global rescheduling problem for railway networks. In *16th International IEEE Annual Conference on Intelligent Transportation Systems*, The Hague, The Netherlands, October 2013.

[8] D. H. Lee, J. X. Cao, and Q. X. Shi. Synchronization of yard truck scheduling and storage allocation in container terminals. *Engineering Optimization*, 41(7):659–672, 2009.

[9] A. Mascis and D. Pacciarelli. Job shop scheduling with blocking and no-wait constraints. *European Journal of Operations Research*, 143(3):498–517, 2002.

[10] A. Nambiar and R. Judd. Max-plus-based mathematical formulation for cyclic permutation flow-shops. *International Journal of Mathematical Modelling and Numerical Optimisation*, 2:85–97, 2011.

[11] M. Pinedo. *Scheduling: Theory, Algorithms and Systems, second ed.* Prentice-Hall, Englewood Cliffs, NJ, 2001.

[12] R. Stahlbock and S. Voß. Operations research at container terminals: a literature update. *OR Spectrum*, 30(1):1–52, 2007.

[13] T. J. J. van den Boom, B. Kersbergen, and B. de Schutter. Structured modeling, analysis, and control of complex railway operations. In *Conference on Decision and Control*, pages 7366–7371, Hawaii, 2012.

[14] T.J.J. van den Boom and B. De Schutter. Modelling and control of discrete event systems using switching max-plus-linear systems. *Control Engineering Practice*, 14(10):1199–1211, October 2006.

[15] R. J. H. A. van Zijverden and R. R. Negenborn. Survey of approaches for integrated control of intermodal container terminals. In *Proceedings of the 2012 IEEE International Conference on Networking, Sensing and Control*, pages 67–72, Beijing, China, April 2012.

[16] I. F. A. Vis and R. De Koster. Transshipment of containers at a container terminal: an overview. *European Journal of Operational Research*, 147:1–16, 2003.

[17] J. Xin, R. R. Negenborn, and G. Lodewijks. Energy-aware control for automated container terminals using integrated flow shop scheduling and optimal control. *Transportation Research Part C*, 2014.