Hartmann, Sönke

A general framework for scheduling equipment and manpower on container terminals

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 566

Provided in Cooperation with:
Christian-Albrechts-University of Kiel, Institute of Business Administration

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 566

A General Framework for Scheduling Equipment and Manpower on Container Terminals

Sönke Hartmann

Abstract

In this paper, we propose a general model for various scheduling problems that occur in container terminal logistics. The scheduling model consists of the assignment of jobs to resources and the temporal arrangement of the jobs subject to precedence constraints and sequence-dependent setup times. We demonstrate how the model can be applied to solve several different real-world problems from container terminals in the port of Hamburg (Germany). We consider scheduling problems for straddle carriers, automated guided vehicles (AGVs), stacking cranes, and workers who handle reefer containers. Subsequently, we discuss a simple dispatching heuristic as well as a more advanced genetic algorithm for the general model. Based on a tailored generator for experimental data, we examine the performance of the heuristics in a computational study. We obtain promising results that suggest that the genetic algorithm is well suited for application in practice.

Keywords: Container logistics, container terminal, optimization, scheduling, heuristics, genetic algorithm.

1 Introduction

In the 1960s, the container was introduced as a universal carrier for various goods. It soon became a standard in worldwide transportation. The success of the container is associated with the increasing containerization (which means that the number of goods transported in containers has steadily grown) and with increasing world trade. At the same time, container terminals are continuously facing the challenge of strong competition between ports and of turning around more and larger ships in shorter times. This leads to the necessity to use the highly expensive terminal resources such as quai cranes, straddle carriers, automated guided vehicles, and stacking cranes as efficiently as possible. A key factor of success is the optimization of the logistic processes. Therefore, many researchers and practitioners have developed optimization approaches for container terminal logistics.

Important optimization problems include the assignment of berths to arriving vessels (see Imai et al. [12, 13], Lim [22]) and the assignment of quai cranes to vessels or ship-bays (see Daganzo [3], Peterkofsky and Daganzo [24]). Scheduling the container transport on the terminal has been studied for two different types of equipment, namely straddle carriers (see Böse et al. [2], Kim and Kim [19], Steenken et al. [27]) and automated guided vehicles (see Bae and Kim [1]). Also the problem of allocating and scheduling stacking cranes has been considered (see Zhang et al. [32]). Strategies for locating containers in the yard have been discussed for several problem settings (see de Castilho and Daganzo [4], Kim and Kim [16], Kim et al. [17], Taleb-Ibrahimi et al. [29], Zhang et al. [31]). In order to study the complex processes on container terminals with their dynamic nature, simulation models have been developed (see Gambardella et al. [7, 8], Kim et al. [18], Legato and Mazza [21], Yun and Choi [30]). Finally, a method to generate data for experiments with optimization and simulation approaches has been suggested (see Hartmann [10]).

In this paper, we introduce a general model for scheduling container terminal resources such as different types of equipment and manpower. This way, we take an approach that is different from the scheduling papers listed above because we provide a model that is not designed for a single application. Our model consists of a set of jobs (which could be transportation tasks or other activities) that must be scheduled and assigned to a resource. For the temporal arrangement of the jobs, sequence-dependent setup times have to be observed. They can be used to cover, e.g., the empty times of the equipment between two container transports. The objective is to minimize the average tardiness per job with respect to due dates. In order to demonstrate the generality of

our model, we present four applications from the port of Hamburg (Germany), namely straddle carriers, automated guided vehicles (AGVs), stacking cranes, and workers. Subsequently, we propose a genetic algorithm for the general model and analyze it in a computational study. The paper closes with a summary of the results and discusses opportunities for future research.

2 A General Model for Scheduling Problems

In this section, we propose a general model for scheduling problems in container terminal logistics. After the definition of the model, we outline how several real-world scheduling problems can be captured by the general model.

2.1 Model Formulation

We consider a set $J = \{1,\ldots,n\}$ of jobs that have to be carried out. Moreover, we consider a set $R = \{1,\ldots,m\}$ of identical resources. Each job $j \in J$ has to be executed on one of the resources. Since the resources are identical, each job can be performed on any of the resources. Each resource $r \in R$ can only be occupied with one job at a time. Some of the jobs are related by precedence constraints. The predecessors of job j are given by the set P_j. The processing of a job may not start before all of its predecessors are completed. The set of successors of job j is denoted as S_j.

Each job $j \in J$ has a processing time $p_j > 0$. Before job j can actually be processed, a setup time is required. Let $i \in J$ denote the job that is executed on resource $r \in R$ immediately before job j. A sequence-dependent setup time $s_{ij} \geq 0$ has to be taken into account before processing job j. Once started, a job may not be interrupted, that is, both the setup and the processing phase of a job must be carried out as a non-preemptable whole. For the setup times, the following triangle inequality must hold: We assume $s_{ik} \leq s_{ij} + s_{jk}$ for any three jobs i, j, and k.

Each job $j \in J$ is related to two specific time instants. First, time d_j is the due date of job j. That is, job j should be completed at or before time d_j. Completing job j later than time d_j is allowed, but it will lead to penalty costs in the objective function. Second, time e_j denotes the earliest time at which resource $r \in R$ that carries out job j is available again after the execution of job j. Hence, if job j is completed earlier than time e_j, then resource r becomes available at time e_j. Otherwise, resource r becomes available immediately after the completion of job j.

We assume that the model is applied repeatedly in a rolling planning horizon. Therefore, the scheduling problem must take the "initial state" of each resource into account, that is, the availability time and the setup state of that resource. The initial state of a resource is characterized by the last job carried out by that resource. For each resource $r \in R$, we denote the last job as j_r^+. This last job is a dummy job in the sense that it is assumed to be fixed, that is, it cannot be (re-)scheduled. The last jobs are comprised in the set $J^+ = \{j_r^+ | r \in R\}$, for which we assume $J \cap J^+ = \emptyset$. Each last job j_r^+ is associated with two types of information. First, it is related to a finish time $f_{j_r^+}$ that reflects the time at which resource r is available. Note that $f_{j_r^+} \geq t_{\text{now}}$ must hold, where t_{now} denotes the current time (without loss of generality, we assume $t_{\text{now}} = 0$). Second, job $j_r^+ \in J^+$ contains the initial setup state of resource $r \in R$ (hence the setup time s_{ij} must be given for all $i \in J \cup J^+$ and $j \in J$). The definition of J^+ eases the notation of the model with respect to the initial state. Let us emphasize again that only the jobs in J (and not those in J^+) are considered for scheduling.

The problem now is to find a schedule for the jobs $j \in J$, that is, a resource r_j and a finish time f_j for each job $j \in J$ such that all constraints given above are observed. The objective is to minimize the average tardiness per job. Tardiness minimization and hence observing the due dates

is one of the most important goals in practice. Other objectives of high practical relevance will be discussed in Section 6.

The problem setting can be summarized by the following conceptual model. We have two types decision variables: f_j is the finish time of job j. Moreover, x_{jr} is a decision variable that determines wether job j is executed on resource r ($x_{jr} = 1$) or not ($x_{jr} = 0$). Finally, let $i_j \in J \cup J^+$ be the job that is carried out immediately before job j on the same resource. Now the conceptual model can be formulated as follows:

$$\text{Minimize} \quad \frac{1}{n} \cdot \sum_{\substack{j \in J \\ f_j > d_j}} (f_j - d_j) \tag{1}$$

$$\text{subject to} \quad \sum_{r \in R} x_{jr} = 1 \qquad j \in J \tag{2}$$

$$f_{i_j} + s_{i_j,j} + p_j \leq f_j \qquad j \in J \tag{3}$$

$$e_{i_j} + s_{i_j,j} + p_j \leq f_j \qquad j \in J \tag{4}$$

$$f_i + p_j \leq f_j \qquad j \in J,\ i \in P_j \tag{5}$$

$$x_{jr} \in \{0,1\} \qquad j \in J,\ r \in R \tag{6}$$

$$f_j \geq 0 \qquad j \in J \tag{7}$$

The objective function (1) minimizes the average tardiness. Constraints (2) ensure that each job is assigned to exactly one resource. Next, (3) take care of the setup times and the processing times between two consecutive jobs on the same resource. The earliest availability time of a resource after the completion of a job is considered by (4). The precedence relations are taken into account by (5). Finally, (6) and (7) define the decision variables.

The model contains several features that are well known from other scheduling problems in the literature. The concept of jobs with precedence constraints and resource requests can also be found in the classical resource-constrained project scheduling problem (see Pritsker et al. [25]). Moreover, there is a relationship between the resource assignment part of the new model on the one hand and the multi-mode extension of the resource-constrained project scheduling problem on the other hand (see Elmaghraby [6], Talbot [28]). The objective of minimizing the tardiness with respect to due dates has also been considered in the context of project scheduling (see, e.g., Kapuscinska et al. [15]). The concept of sequence-dependent setup times occurs in many problems in the field of lotsizing and batching (see Jordan [14]). We will make use of the similarity between the new model and resource-constrained project scheduling when designing a heuristic for the new model in Section 3.

Note that the model outlined above is a rather abstract formulation. The generality of the approach enables us to apply it to different problem settings that do not seem to have much in common at first glance. The jobs reflect tasks that have to be carried out in container handling. The resources can reflect the technical equipment for container handling or manpower. The setup and processing times can correspond to moving a resource to some location, moving a container, or other tasks. Several practical applications will be given in the following subsections.

2.2 Application to Straddle Carriers

Many container terminals employ straddle carriers (sometimes also called van carriers) for transportation of containers on the terminal. Straddle carriers are used for transportation between the

stack on the one hand and other locations such as a quai crane, the area for external trucks, or the train area on the other hand. Since straddle carriers are able to unload themselves and high enough to stack containers on top of each other, they can also be employed to transport containers to their positions within a yard block. Thus, they can also carry out shuffle moves within a block, that is, they can move a container to another position in case it stands on top of another container that must be moved out. If stacking cranes serve the blocks, however, straddle carriers transport containers from and to hand-over positions at a block, and they do not carry out shuffle moves.

Of course, the set of the transportation tasks of the straddle carriers defines the set of jobs for scheduling, and the number of resources is given by the number of straddle carriers. Precedence relations exist between jobs that correspond to containers that stand on top of each other. The job related to an upper container is a predecessor of the job related to the lower one.

The processing time p_j of a job j is the time that a straddle carrier needs for the transportation of a container between two locations on the terminal. It is defined as the time between picking up the container and putting it down. As the locations are fixed, the transportation times are fixed as well. The setup time s_{ij} between two jobs i and j is given by the time that a straddle carrier requires to get from the position where the container of job i was put down to the pick-up position of job j. Hence, the setup time models the empty travel time of a straddle carrier. For any two positions, an estimate of the processing and setup time is usually available for scheduling. Of course, scheduling can affect only the empty (or setup) times but not the transportation (or processing) times.

The due date d_j of a job j is determined as follows. Let us first consider the case that a straddle carrier is supposed to bring a container from the stack to another location such as a quai crane or an external truck. In this case, the due date reflects the latest time at which the container should have arrived there. For a quai crane waiting for a container, the due date corresponds to the time at which the quai crane requires the container in order to keep its loading sequence on time. Considering an external truck, the due date reflects the acceptable waiting time of the truck driver. Moreover, keeping the waiting times for trucks short helps to avoid congestions in the truck area. In the second case, a straddle carrier is supposed to bring a container to the stack. Here, the due date is used to model the latest time at which a straddle carrier has to pick up a container at some location. For example, an arriving container must be picked up from a quai crane or an external truck at or before the due date. In this case, we have a due date for pick-up, say $\bar{d}_j$. The latter can be transformed into a due date related to the completion of the job by setting $d_j = \bar{d}_j + p_j$. This is possible because the time between pick-up and completion is a constant, namely the transportation or processing time p_j. Recall that the due date is always related to the finish time of a job, see the objective function (1).

The earliest availability time e_j of a resource after carrying out a job j is used to model the blocking of straddle carriers by other resources. Consider a job j that requires a straddle carrier to transport a container from a quai crane to a yard block. Let $\bar{e}_j$ denote the time at which the container is available for the straddle carrier (i.e., the time at which it has been put on the ground by the quai crane). The earliest release time of the straddle carrier is $e_j = \bar{e}_j + p_j$ which reflects the earliest possible time to complete this job. Now let us consider a job j to transport a container from the stack to a truck which is already waiting for the container. In this case, we have $e_j = 0$ because the straddle carrier is available immediately after completing the job. These two cases should be sufficient to illustrate the use of the earliest release times. Generally speaking, we have $e_j > 0$ if the straddle carrier will have to wait for another resource, whereas we have $e_j = 0$ if it is not blocked by another resource.

The use of the initial state and of the objective function is obvious. Summing up, we have employed all features of the general scheduling model of Subsection 2.1 to capture the straddle

carrier scheduling problem.

2.3 Application to Automated Guided Vehicles

On modern container terminals with a high degree of automation, often automated guided vehicles (AGVs) are employed to carry containers between the quai and the yard blocks. Unlike straddle carriers, AGVs are unable to unload themselves. Therefore, stacking cranes are needed to serve the yard blocks. Hence, the jobs to schedule are the transportation of containers from a quai crane to a stacking crane and from a stacking crane to a quai crane. While the former case corresponds to the discharging of a vessel, the latter case occurs when loading a vessel.

The application of the general model to the AGV case is similar to the straddle carrier case with only a few differences. Again, the processing time p_j reflects the transportation of a container while the setup time s_{ij} models the empty time between two successive jobs. The due dates d_j reflect the latest hand-over times at the quai cranes. If an AGV exceeds its due date, this will lead to a waiting time of the quai crane. Thus, keeping the due dates is crucial for a high quai crane productivity and hence for a short time in port for the vessels. The earliest release times e_j consider the fact that AGVs can be blocked by the other terminal resources. Since AGVs cannot unload themselves, an AGV arriving before its due date will have to wait for the crane related to this job. Therefore, the earliest release time is equal to the due date, that is, $e_j = d_j$. Finally, precedence relations are needed to control the order of AGVs arriving at a quai crane with a container. This AGV order is important because the sequence of containers to be loaded onto a vessel is often fixed.

2.4 Application to Stacking Cranes

Usually, the stack is organized in several yard blocks. Many container terminals employ stacking cranes to serve these blocks. Normally, there is one crane for each block (occasionally, two cranes per block are used). So-called rail-mounted gantry cranes cannot be moved to another block. On the other hand, so-called rubber-tyred gantry cranes can be moved, but this takes a long time and is not done very often (decisions to transfer a crane are based on workload estimations of the blocks, see Zhang et al. [32]). Thus, for both crane types, crane assignments to blocks can be assumed to be fixed for the detailed scheduling problem considered here. Moreover, the assignment of containers to blocks is given (it is not part of this scheduling problem). Therefore, we have a separate crane scheduling problem for each block. That is, we consider the set of jobs associated with a single block, and we have a single crane resource (or, in the rare case of double cranes, two resources). The jobs include transportation moves between positions within the block on the one hand and AGVs, straddle carriers, or external trucks on the other hand. Furthermore, jobs representing shuffle moves have to be taken into account.

As for the straddle carrier case, we have precedence relations between jobs that correspond to containers that stand on top of each other. The processing time p_j of a job j is the transportation time that a crane needs between picking up the container and putting it down. Again, the setup time s_{ij} between two jobs i and j is given by the time that the crane requires to get from the position where the container of job i was put down to the pick-up position of job j. The due date d_j of a job j determines the latest acceptable completion time and hence also the waiting time of the other resource (AGV, straddle carrier) or the external truck. The earliest release times again model the blocking of the crane by other resources. For example, if a crane serves an AGV which arrives in a just-in-time fashion, the crane will not be available before the due date, that is, we have $e_j = d_j$.

On the other hand, if a crane is to serve an external truck which has already arrived at the block, it will be available after completing the job, that is, we have $e_j = 0$.

2.5 Application to Reefer Workers

Reefer containers are used to carry goods that require a controlled temperature (such as refrigerated or frozen goods). They have a device for temperature regulation which requires electricity. Hence, they are stored in specific yard blocks or areas of blocks that provide electricity connections. Reefer containers require special handling by a manpower resource, the reefer workers.

The jobs carried out by the reefer workers are connecting arriving containers, disconnecting departing ones, doing small repair tasks, and controlling the temperature of the reefer containers in the stack. For each of these job types, a processing time p_j is given. Note that, unlike the previously described applications, here the processing time does not correspond to moving to another location. The use of the setup times, however, is similar to equipment scheduling. The setup time s_{ij} between a job i and a job j is the time that a reefer worker needs to move from the container associated with job i to the container related to job j. An estimate of this time can be assumed to be available. This estimate is based on the container positions in the stack.

For connection jobs, the due date d_j reflects the time that a reefer container is allowed to be without electricity. Of course, a container can keep its temperature in the allowed range only for a certain time if it is without electricity. For disconnection jobs, the due date reflects the latest time the container must be disconnected such that the stacking crane or straddle carrier can pick it up on time (the terminal control system might include a buffer time when computing the due date). For repair jobs and temperature control jobs, a due date is given as well. After a worker has completed a job, he can immediately move on to the next job. Therefore, we can set the earliest availability times to $e_j = 0$. A definition of precedence relations is not necessary.

In practice, the reefer workers are often equipped with portable radio sets. When they have completed a job, they transmit this information to the terminal control system via the radio set. In return, they get the next job.

3 Optimization Heuristics

3.1 Priority Rule Based Dispatching Method

In this subsection, we present a simple dispatching heuristic for the scheduling problem. This method can be be viewed as a straightforward way to build job sequences for the resources. The following steps are repeated until all jobs have been scheduled:

- **Compute eligible jobs.** Compute the set E of eligible jobs as the set of the currently unscheduled jobs of which all predecessors have already been scheduled.
- **Select job.** Select the job j^* to be scheduled next as the eligible job with the smallest due date.
- **Select resource.** Select the resource r^* to be assigned to the selected job as the resource with the earliest completion time so far.
- **Update schedule.** Schedule job j^* at the end of the current job sequence of resource r^*.

From a more general point of view, the criteria for selecting a job and a resource can be seen as priority rules. According to the classification of Kolisch and Hartmann [20], this approach is a deterministic single-pass priority rule method. Due to its simplicity, this dispatching method does not contain much intelligence for optimization. In our computational study, we use it merely for comparison purposes.

3.2 Genetic Algorithm

Introduced by Holland [11], genetic algorithms (GAs) adopt the principles of biological evolution to solve hard optimization problems. For our GA, we exploit the similarities of the general container terminal scheduling problem to the resource-constrained project scheduling problem. These similarities allow us to use the project scheduling GA of Hartmann [9] as a starting point. This GA will be adapted to solve the problem introduced in this paper. In particular, the decoding procedure and the construction of the initial population require problem-specific knowledge.

3.2.1 Basic Scheme

We apply the generational management framework of Eiben et al. [5]. The GA starts with the computation of an initial population, i.e., the first generation. The number of individuals in the population is referred to as P. The GA then determines the fitness values of the individuals of the initial population. After that, we apply the crossover operator to produce new individuals ("children") from the existing ones ("parents"). Subsequently, we apply the mutation operator to the newly produced children. After computing the fitness of each child, we add the children to the current population. Then we apply the selection operator to reduce the population to its former size P. Doing so, we obtain the next generation to which we again apply the crossover operator and so on. This process is repeated for a prespecified number G of generations is reached. Note that exactly $P \cdot G$ individuals (and thus schedules) are computed. Of course, other stopping criteria could be employed as well, e.g., a time limit or a number of generations without improvement of the best objective function value found so far.

More formally, the GA scheme can be summarized as follows. Here, $\mathcal{P}$ denotes the current population (i.e., a set of individuals), C is the set of children, and g is the number of the current generation.

$g := 1$;
generate P individuals for the initial population $\mathcal{P}$;
apply decoding procedure to compute fitness for individuals $I \in \mathcal{P}$;
for $g := 2$ **to** G **do**
begin
    produce a set C of children from $\mathcal{P}$ by crossover;
    apply mutation to children $I \in C$;
    apply decoding procedure to compute fitness for children $I \in C$;
    $\mathcal{P} := \mathcal{P} \cup C$;
    reduce population $\mathcal{P}$ to size P by means of selection;
end.

3.2.2 Problem Representation

Genetic algorithms for scheduling problems often do not operate directly on schedules but on representations of schedules. The latter are then transformed into schedules by means of a problem-specific decoding procedure. The advantage of such an indirect approach is that it allows to employ standard representations together with standard genetic operators (crossover, mutation).

In our genetic algorithm, a schedule is represented by a precedence feasible job list $(j_1,\ldots,j_n)$. In a job list, each job appears exactly once, that is, we have $J = \{j_1,\ldots,j_n\}$. A job list is precedence feasible if all predecessors of a job j appear in the list before job j, that is, $P_{j_i} \subseteq \{j_1,\ldots,j_{i-1}\}$ for $i = 1,\ldots,n$. This representation is a generalization of the classical permutation based representation (see Reeves [26]). It has been shown to be superior to other representations for resource-constrained scheduling problems (see Hartmann [9]). Note that the representation does not contain information on resource selection. In fact, the task of assigning resources to jobs will be left to the decoding procedure.

3.2.3 Decoding Procedure

The decoding procedure employs problem-specific features to transform the standard representation into a solution for our scheduling problem. The job list determines the order in which the jobs are scheduled by the decoding procedure. The latter scans the job list from left to right and successively schedules the jobs using the following steps:

- **Select job.** Select the next job j_i from the job list.
- **Select resource.** Select resource r^* such that executing job j_i on r^* leads to the smallest possible tardiness among all resources. If there is more than one resource with zero tardiness, select the one with the maximum availability time.
- **Update schedule.** Schedule job j_i at the end of the current job sequence of resource r^*.

While the scheduling order of jobs is prescribed by the job list representation, the decoding procedure takes care of the resource assignment part of the problem. In accordance with the objective function, the resource selection mechanism minimizes the tardiness of the current job. As a tie-breaking criterion to choose among resources with tardiness $= 0$, we select the resource with the largest availability time in the current partial schedule. The idea behind this is that we do not want to waste resource capacities by selecting a resource that would lead to a large setup time or considerable earliness. Note that in the case of only one resource, there are no choices to be made by the decoding procedure. The fitness of an individual is defined as the objective function value of the schedule related to the individual.

3.2.4 Crossover

For crossover, the current population is randomly partitioned into pairs of individuals. From each pair of individuals (parents), two new individuals (children) will be produced. Let us assume that two individuals of the current population have been selected for crossover. We have a mother individual $M = (j_1^M,\ldots,j_n^M)$ and a father individual $F = (j_1^F,\ldots,j_n^F)$. Now two child individuals have to be constructed, a daughter D and a son S.

Let us start with a definition of the daughter $D = (j_1^D,\ldots,j_n^D)$. Combining the parent's activity lists, we have to make sure that each activity appears exactly once in the daughter's activity list. We make use of a general crossover technique presented by Reeves [26] for permutation based

genotypes. We perform a two-point crossover for which we draw two random integers q_1 and q_2 with $1 \leq q_1 < q_2 \leq n$. Now the daughter's job list is determined by taking the activity list of the positions $i = 1,\ldots,q_1$ from the mother, that is, $j_i^D := j_i^M$. The positions $i = q_1+1,\ldots,q_2$ are derived from the father. However, the activities already selected may not be considered again. We set $j_i^D := j_k^F$ where k is the lowest index such that $j_k^F \notin \{j_1^D,\ldots,j_{i-1}^D\}$. The remaining positions $i = q_2+1,\ldots,n$ are again taken from the mother, that is, we have $j_i^D := j_k^M$ where k is the lowest index with $j_k^M \notin \{j_1^D,\ldots,j_{i-1}^D\}$. The son individual is computed analogously. For the son's job list, the first and the third part are taken from the father and the second one is taken from the mother.

This crossover strategy constructs job lists in which each job appears exactly once. Moreover, it has been proven by Hartmann [9] that the resulting job lists are precedence feasible, given that the parents' job lists were precedence feasible as well. Since this crossover operator produces feasible offspring, there is no need for a repair operator. This property leads to a good inheritance behavior of building blocks of solutions.

3.2.5 Mutation

The following mutation operator is applied to each newly produced child individual. We move through the job list from left to right. Thereby, we apply a right shift to each job with a probability of p_{mutation}. Consider a current position $i \in \{1,\ldots,n-1\}$ in the job list

$$(j_1,\ldots,j_i,\ldots,j_h,\ldots,j_z,\ldots,j_n).$$

Let z be the smallest index of the successors of job j_i, that is, $z = \min\{k \mid j_k \in S_{j_i}\}$. Now job j_i can be shifted behind some randomly drawn position $h \in \{i+1,\ldots,z-1\}$, which leads to job list

$$(j_1,\ldots,j_{i-1},j_{i+1},\ldots,j_h,j_i,j_{h+1},\ldots,j_z,\ldots,j_n).$$

That is, activity j_i is right shifted within the activity list and inserted immediately after some activity j_h. Clearly, the resulting job list is still precedence feasible. On the basis of preliminary computational experiments, we selected a mutation probability of $p_{\text{mutation}} = 0.05$.

3.2.6 Selection

After the newly produced individuals have been added to the current population, the next step is to select the individuals that survive and make up the next generation. We have tested several variants of the selection operator which all follow a survival-of-the-fittest strategy (cf., e.g., Michalewicz [23]). Following the study of Hartmann [9]), we decided to employ the ranking method. The ranking method sorts the individuals with respect to their fitness values and selects the P best ones while the remaining ones are deleted from the population (ties are broken arbitrarily).

3.2.7 Initial Population

Finally, we define how to determine the first generation containing P individuals. That is, we have to specify the construction of job lists. A job list is constructed by a priority rule based heuristic which can be classified as a multi-pass biased random sampling method (see Kolisch and Hartmann [20]). For building up one job list, the following steps are repeated until the list contains all n jobs:

- **Compute eligible jobs.** Compute the set E of eligible jobs. A job is eligible if it has not yet been added to the current job list and if all its predecessors have already been added.

- **Select job.** Draw the eligible job $j^* \in E$ to be scheduled next using the probability

$$p(j) = \frac{\delta - d_j + 1}{\sum_{i \in E} \delta - d_i + 1}$$

for each eligible job $j \in E$, where $\delta = \max\{d_j \mid j = 1, \ldots, n\}$ is the maximal due date.

- **Update job list.** Add job j^* at the end of the current job list.

Generally speaking, the initial population should have two properties: It should be diversified because a large gene pool is advantageous for the artificial evolution. Moreover, it should be of good average quality (in terms of fitness or objective function value) because this provides a good starting point for the evolution. The random mechanism of our method should lead to a reasonable genetic diversity while the definition of the probabilities $p(j)$ should lead to job lists of good quality. This is due to the fact that the probabilities $p(j)$ are based on a priority rule which assigns higher priorities (and thus higher selection probabilities) to jobs with tighter due dates. That is, these jobs are likely to appear earlier in a job list, and they will be scheduled earlier by the decoding procedure. This way, the risk of exceeding their due dates is reduced, which leads to better objective function values.

Note that the method outlined above is different from the straightforward approach which selects jobs on a pure (unbiased) random basis, that is, with equal probabilities $P_j = \frac{1}{|E|}$. Our method puts more emphasis on constructing job lists of good quality. This point will be examined further in the computational tests of Subsection 5.2.

4 Generating Experimental Data

4.1 Generator

In order to demonstrate the applicability of the general scheduling model and the genetic algorithm, we carried out several computational experiments. These experiments required test instances as input data for the heuristic. In order to obtain a large number of test instances, we developed a data generator for our problem setting. It is controlled by parameters that allow to produce instance sets with specific characteristics. In particular, the parameters enabled us to produce quite realistic test sets for the container terminal applications mentioned in Section 2.

The generator works as follows. The number n of jobs and the number m of resources are specified by the user as parameters. In addition to the n regular jobs, the m dummy jobs that represent the initial setup states for the resources are generated. Each (non-dummy) job j is randomly assigned a processing time p_j from $\{p_{\min}, \ldots, p_{\max}\}$, where $p_{\min}$ and $p_{\max}$ are parameters that denote the minimal and maximal processing time, respectively. On the basis of a parameter T for the scheduling horizon, a due date d_j is drawn from $\{p_j, \ldots, T\}$ for each job j. The earliest release times are determined using a parameter $\alpha \in [0, 1]$. This parameter allows to control whether the earliest release time e_j of a job j is equal to its due date or zero (recall that these were the two typical cases in the applications of our model). With probability α, we set $e_j = 0$ and with probability $1 - \alpha$, we set $e_j = d_j$. Finally, we have to generate a setup time s_{ij} between two consecutive jobs i and j on the same resource. The following approach is designed to produce setup times which do not violate the triangle inequality. First, each job j is randomly assigned a value $y_j \in \{0, \ldots, S\}$ between zero and a parameter S representing the maximal setup time. Then the setup time between jobs i and j is defined as $s_{ij} = |y_j - y_i|$.

Test set	number of jobs	number of resources	scheduling horizon
straddle carrier	380	80	30 min
AGV	105	50	15 min
reefer worker	125	6	60 min
stacking crane	8	1	30 min

Table 1: Characteristics of the four generated test sets

4.2 Test Sets

Using the generator described in the previous subsection, we generated a set of test instances for each of the four container terminal applications discussed in Section 2. That is, we have a straddle carrier set, an AGV set, a reefer worker set, and a stacking crane set. These four sets differ in the settings of the generator parameters and hence in their characteristics. We attempted to generate realistic instances which may similarly occur in practice at peak time on a medium-sized container terminal. The characteristics of the test sets can be summarized as follows (see also the main parameter settings displayed in Table 1):

- The straddle carrier set contains the largest instances. This is because we assume the straddle carriers to carry out transportation jobs both on the seaside and on the landside as well as shuffle moves. This set includes the largest number of resources and jobs and a medium scheduling horizon.
- The AGV case considers only transportation jobs between quai and stack. Therefore, we have selected smaller numbers of jobs and resources than for the straddle carrier set. Moreover, we have a shorter scheduling horizon because shuffle moves are not considered.
- In the reefer worker set, we have fewer resources than in the previous two cases but a longer scheduling horizon. The latter results from substantially longer time windows for reefer jobs.
- The stacking crane case is the smallest problem setting with only one resource (corresponding to one yard block served by a single crane). Due to shuffle moves, the horizon is the same as for the straddle carrier set.

For each of the test sets, the parameter settings were done in a way that leads to scarce resources. Scarce resources make it difficult to find schedules with small tardiness—thus, we obtain challenging test problems. This is important for our computational analysis because we need an average tardiness > 0 to be able to compare the performance of the heuristics with respect to the objective funtion value (in fact, the parameters were adjusted such that we have an average tardiness > 0 even for the best heuristic). Moreover, the case of scarce resources is particularly important in practice. In this case, minimization of tardiness is crucial for successfully carrying out the logistic processes.

For each of the four cases, 250 instances were generated. Hence, our test set consists of 1,000 instances altogether which should be a reasonable basis for a thorough computational analysis.

Heuristic	$P \times G$	straddle carrier	AGV	reefer worker	stacking crane
Dispatching	—	202.4	128.4	292.4	29.0
GA	50×25	10.4	10.0	11.9	12.0
GA	100×50	5.7	8.1	3.8	11.9
GA	200×100	2.8	6.7	1.5	11.9

Table 2: Results of the heuristics (average tardiness per job in seconds)

Heuristic	$P \times G$	straddle carrier	AGV	reefer worker	stacking crane
Dispatching	—	0.05	0.01	0.01	< 0.01
GA	50×25	1.72	0.17	0.11	< 0.01
GA	100×50	6.22	0.62	0.38	0.01
GA	200×100	22.80	2.52	1.47	0.07

Table 3: Average computation times on a Pentium 4 with 1.6 GHz (in seconds)

5 Computational Results

5.1 Evaluation of the Heuristics

In order to evaluate the heuristics, they were coded in ANSI C and compiled with the lcc compiler. The experiments were carried out on a Pentium 4-M based computer with 1.6 GHz running under Windows XP.

Table 2 gives the results for the GA with three different settings for population size P and number of generations G. For comparison purposes, also the results of the simple dispatching method are displayed. As expected, increasing the number of schedules computed by the GA improves the results (although this does not hold for the stacking crane case, but that is due to the extremely small problem size which does not leave much potential for longer optimization). The average tardiness achieved by the dispatching method ranges between half a minute and almost five minutes. Consequently, the dispatching method is of no use in practice because it produces unacceptable delays. The GA, however, is able to reduce the average tardiness to only a very few seconds (recall that the GA still leads to an average tardiness > 0 due to the design of the test instances). Assuming that the test instances are quite realistic, we can state that these results suggest to apply the GA in practice. This is particularly true since the instances used here are very hard due to scarce resource capacities.

Let us now have a brief look at the computation times that were needed to produce the results of Table 2. An overview of the computation times is given in Table 3. We can see that the computation time increases with the number of jobs and the number of resources which is due to the structure of the decoding procedure. Of course, it also increases with the number of schedules computed. Most important, however, is that the GA requires only a moderate computational effort. A computation time of a few seconds (say, one or two) makes it applicable for online optimization and frequent rescheduling in practice. Consequently, the GA can be used in an online terminal control system even for the largest instances (in our case, straddle carrier scheduling), given that the number of schedules computed is restricted appropriately. Faster computers will further increase its applicability.

Heuristic	initial population	straddle carrier	AGV	reefer worker	stacking crane
GA	unbiased random	28.8	10.6	19.6	11.9
GA	biased sampling	5.7	8.1	3.8	11.9

Table 4: Impact of method for initial population (average tardiness per job in seconds)

Heuristic	criterion	straddle carrier	AGV	reefer worker	stacking crane
GA	tardiness only	28.2	11.4	5.3	11.9
GA	tardiness & availability	5.7	8.1	3.8	11.9

Table 5: Impact of decoding procedure (average tardiness per job in seconds)

5.2 Effectiveness of Genetic Algorithm Components

We now turn to the question whether the components of the GA really contribute to its performance. Three components will be examined in more detail, namely the initial population, the decoding procedure, and the crossover operator. In each case, we compare the original GA (with $P \times G = 100 \times 50$) with a variant in which the component under consideration is modified.

Let us first consider the method for computing an initial population. As mentioned in Subsection 3.2.7, we have employed due date based probabilities for job selection instead of equal probabilities. Table 4 displays the results for biased sampling and those for the straightforward unbiased random approach. The biased sampling method which makes use of the due dates leads to better solutions. It produces a better initial population which is still diverse enough, so that it provides a good starting point for the evolution.

Next, we discuss the decoding procedure. Recall that the procedure selects a resource with respect to minimum tardiness and, as a tie-breaker for resources with zero tardiness, with respect to maximum availability time. While the first criterion corresponds to the objective function, one might wonder if the second criterion has a positive effect. Table 5 gives the results for the GA with both criteria and for a variant in which only the first criterion is used. The results show that the availability time criterion is indeed useful.

Finally, we analyze the inheritance mechanism determined by the representation and the crossover operator. We want to know if the GA really benefits from inheriting genes from previous generations. The GA computes P schedules for the initial population and applies crossover, mutation, and selection over G generations. This way, it evaluates $P \cdot G$ schedules on the whole. The GA can now be compared with the method that computes the initial population which is stopped after it has computed $P \cdot G$ schedules. Thus, the latter requires the same computational effort as the full GA but does not apply the genetic operators. Of course, the genetic operators only make sense if the GA leads to better results than the method for the initial population alone, given that in both cases the same number of solutions is computed. In fact, according to the study of Hartmann [9], there are GAs in the literature which fail this test, that is, they do not benefit from genetic inheritance. The results provided in Table 6, however, show that our GA leads to much better results than the sampling method for constructing the initial population if the same number of solutions is computed. Hence, it clearly profits from the genetic principles.

Heuristic	#schedules	straddle carrier	AGV	reefer worker	stacking crane
biased sampling only	5000	24.0	17.3	44.0	12.3
full GA	100×50	5.7	8.1	3.8	11.9

Table 6: Impact of inheritance (average tardiness per job in seconds)

6 Conclusions and Research Perspectives

In this paper, we proposed a general optimization model for scheduling jobs on container terminals. We showed that our model is applicable to straddle carriers, automated guided vehicles, stacking cranes, and reefer workers. The generality of our model is advantageous in practice because it allows to use the same model and optimization algorithms for several different scheduling problems. Furthermore, we developed a genetic algorithm to solve the proposed problem. With a tailored instance generator, we generated several large sets of test instances for a computational analysis. Our experiments showed that the genetic algorithm is well suited to solve hard test instances of realistic size. The genetic algorithm appeared to be applicable for online scheduling within a terminal control system where good schedules are needed within very short computation times. The study also demonstrated that the genetric algorithm is clearly superior to a simple dispatching strategy for this problem.

An interesting topic for future research is the development of further algorithms to solve the general scheduling problem. Further heuristics might lead to even better results. Developing exact algorithms to compute optimal solutions may be promising as well. Optimal solutions are useful because they provide a natural basis for the evaluation of heuristics. Moreover, exact algorithms might even be applicable in practice to solve small instances such as stacking crane scheduling with one or two resources.

Another point for further research is the adaptation of the objective function to various practical requirements. We have selected the tardiness minimization with respect to due dates because this appears to be the most important objective according to our practical experience. An extension of this objective would be the minimization of weighted tardiness. That is, each job would be associated with a weight that reflects the penalty for exceeding its due date by one time unit. This way, jobs with higher priority could be preferred in the scheduling process. Moreover, employing weighted tardiness can be useful in an online environment which is typical for container terminal scheduling. Assigning a higher weight to jobs with earlier due dates puts an emphasis on the jobs in the first part of the schedule which will be carried out before rescheduling. Another popular objective is the minimization of the makespan. This might be relevant if the jobs are not directly connected to other processes (e.g., quai crane sequence of specific containers) such that they are not associated with individual due dates. (It is interesting to note that makespan minimization is a special case of the tardiness minimization objective. Simply define a dummy job with zero setup and processing time which is a successor of all other jobs. If the due date of this dummy job is set to zero and the due dates of the other jobs are set to very large numbers, the tardiness objective is equal to the makespan objective.) Finally, the minimum tardiness objective can be extended by a component to minimize the setup time per job. The tardiness part takes care of job completion on time in case of scarce resources. Thereby, the setup times are minimized implicitly because this helps to minimize the tardiness. A setup time component in the objective function would lead to short setup times also if the resources are not scarce and meeting the due dates is not difficult. In practical applications, minimizing setup times saves fuel for the equipment and

increases the acceptance of automatically generated schedules by the human resources (e.g., reefer workers are assigned shortest possible ways). Optimization approaches with alternative objectives could be tested in simulation models, particularly if the objectives explicitly consider aspects of online scheduling.

References

[1] J. W. Bae and K. H. Kim. A pooled dispatching strategy for automated guided vehicles in port container terminals. *International Journal of Management Science*, 6:47–70, 2000.

[2] J. Böse, T. Reiners, D. Steenken, and S. Voß. Vehicle dispatching at seaport container terminals using evolutionary algorithms. In R. H. Sprague, editor, *Proceedings of the 33rd Annual Hawaii Internatinal Conference on System Sciences*, pages 377–388. IEEE, Piscataway, 2000.

[3] C. F. Daganzo. The crane scheduling problem. *Transportation Research B*, 23:159–175, 1989.

[4] B. de Castilho and C. F. Daganzo. Handling strategies for import containers at marine terminals. *Transportation Research B*, 27:151–166, 1993.

[5] A. E. Eiben, E. H. L. Aarts, and K. M. van Hee. Global convergence of genetic algorithms: A markov chain analysis. *Lecture Notes in Computer Science*, 496:4–12, 1990.

[6] S. E. Elmaghraby. *Activity networks: Project planning and control by network models*. Wiley, New York, 1977.

[7] L. M. Gambardella, G. Bontempi, E. Taillard, D. Romanengo, G. Raso, and P. Piermari. Simulation and forecasting of an intermodal container terminal. In A. G. Bruzzone and E. J. H. Kerckhoffs, editors, *Simulation in Industry – Proceedings of the 8th European Simulation Symposium*, pages 626–630. SCS, Ghent, Belgium, 1996.

[8] L. M. Gambardella, A. E. Rizzoli, and M. Zaffalon. Simulation and planning of an intermodal container terminal. *Simulation*, 71:107–116, 1998.

[9] S. Hartmann. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics*, 45:733–750, 1998.

[10] S. Hartmann. Generating scenarios for simulation and optimization of container terminal logistics. Technical Report 564, Universität Kiel, Germany, 2002.

[11] H. J. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, 1975.

[12] A. Imai, K. Nagaiwa, and C. W. Tat. Efficient planning of berth allocation for container terminals in Asia. *Journal of Advanced Transportation*, 31:75–94, 1997.

[13] A. Imai, E. Nishimura, and S. Papadimitriou. The dynamic berth allocation problem for a container port. *Transportation Research B*, 35:401–417, 2001.

[14] C. Jordan. *Batching and scheduling – Models and methods for several problem classes*. Number 437 in Lecture Notes in Economics and Mathematical Systems. Springer, Berlin, Germany, 1996.

[15] T. T. Kapuscinska, T. E. Morton, and P. Ramnath. High-intensity heuristics to minimize weighted tardiness in resource-constrained multiple dependent project scheduling. Technical report, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1998.

[16] K. H. Kim and H. B. Kim. Segregating space allocation models for container inventories in port container terminals. *International Journal of Production Economics*, 59:415–423, 1999.

[17] K. H. Kim, Y. M. Park, and K.-R. Ryu. Deriving decision rules to locate export containers in container yards. *European Journal of Operational Research*, 124:89–101, 2000.

[18] K. H. Kim, S. H. Won, J. K. Lim, and T. Takahashi. A simulation-based test-bed for a control software in automated container terminals. In *Proceedings of the International Conference on Computers and Industrial Engineering*, pages 239–243. Montreal, Canada, 2001.

[19] K. Y. Kim and K. H. Kim. A routing algorithm for a single straddle carrier to load export containers onto a container ship. *International Journal of Production Economics*, 59:425–433, 1999.

[20] R. Kolisch and S. Hartmann. Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis. In J. Weglarz, editor, *Project scheduling: Recent models, algorithms and applications*, pages 147–178. Kluwer, Amsterdam, the Netherlands, 1999.

[21] P. Legato and R. M. Mazza. Berth planning and resources optimisation at a container terminal via discrete event simulation. *European Journal of Operational Research*, 133:537–547, 2001.

[22] A. Lim. The berth planning problem. *Operations Research Letters*, 22:105–110, 1998.

[23] Z. Michalewicz. Heuristic methods for evolutionary computation techniques. *Journal of Heuristics*, 1:177–206, 1995.

[24] R. I. Peterkofsky and C. F. Daganzo. A branch and bound solution method for the crane scheduling problem. *Transportation Research B*, 24:159–172, 1990.

[25] A. A. B. Pritsker, L. J. Watters, and P. M. Wolfe. Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science*, 16:93–107, 1969.

[26] C. R. Reeves. Genetic algorithms and combinatorial optimization. In V. J. Rayward-Smith, editor, *Applications of modern heuristic methods*, pages 111–125. Alfred Waller Ltd., Henley-on-Thames, 1995.

[27] D. Steenken, A. Henning, S. Freigang, and S. Voß. Routing of straddle carriers at a container terminal with the special aspect of internal moves. *OR Spectrum*, 15:167–172, 1993.

[28] F. B. Talbot. Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management Science*, 28:1197–1210, 1982.

[29] M. Taleb-Ibrahimi, B. de Castilho, and C. F. Daganzo. Storage space vs. handling work in container terminals. *Transportation Research B*, 27:13–32, 1993.

[30] W. Y. Yun and Y. S. Choi. A simulation model for container-terminal operation analysis using an object-oriented approach. *International Journal of Production Economics*, 59:221–230, 1999.

[31] C. Zhang, J. Liu, Y.-W. Wan, K. G. Murty, and R. J. Linn. Storage space allocation in container terminals. Technical report, Hong Kong University of Science and Technology, 2001.

[32] C. Zhang, Y.-W. Wan, J. Liu, and R. J. Linn. Dynamic crane deployment in container storage yards. *Transportation Research B*, 36:537–555, 2002.