Optimizing the Slab Yard Planning and Crane Scheduling Problem using a Two-Stage Heuristic

Anders Dohn, Jens Clausen

Optimizing the Slab Yard Planning and Crane Scheduling Problem using a Two-Stage Heuristic

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2008-IJPR-0935
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	17-Nov-2008
Complete List of Authors:	Dohn, Anders; Technical University of Denmark, Department of Management Engineering Clausen, Jens; Technical University of Denmark, Department of Management Engineering
Keywords:	SCHEDULING, HEURISTICS, SIMULATION
Keywords (user):	crane/hoist scheduling, generalized precedence constraints

International Journal of Production Research
Vol. 00, No. 00, 00 Month 200x, 1–23

RESEARCH ARTICLE

Optimizing the Slab Yard Planning and Crane Scheduling Problem using a Two-Stage Heuristic

Anders Dohn* and Jens Clausen

Department of Management Engineering, Technical University of Denmark, Richard Petersens Plads, 2800 Kongens Lyngby, Denmark.

(*Received 00 Month 200x; final version received 00 Month 200x*)

In this paper, we present the Slab Yard Planning and Crane Scheduling Problem. The problem has its origin in steel production facilities with a large throughput. A slab yard is used as a buffer for slabs that are needed in the upcoming production. Slabs are transported by cranes and the problem considered here is concerned with the generation of schedules for these cranes. The problem is decomposed and modeled in two parts, namely a planning problem and a scheduling problem. In the planning problem, a set of crane operations is created to take the yard from its current state to a desired goal state. In the scheduling problem, an exact schedule for the cranes is generated, where each operation is assigned to a crane and is given a specific time of initiation. For both models, a thorough description of the modeling details is given along with a specification of objective criteria. Preliminary tests are run on a generic setup with simulated data. The test results are very promising. The production delays are reduced significantly in the new solutions compared to the corresponding delays observed in a simulation of manual planning.

Keywords: large-scale scheduling; stacker crane problem; crane/hoist scheduling; generalized precedence constraints; schedule visualization; simulation

1. Introduction

The Slab Yard Planning and Crane Scheduling Problem is a complex optimization problem, combining planning and scheduling in an effort to generate feasible schedules for a number of interacting cranes. The problem instances originate from real-world data. Costs and constraints have been defined in cooperation with the industry. The industrial problem instances are of a large size and therefore it is important to create a solution method that can make superior heuristic choices in little time.

The problem here is from a steel hot rolling mill. A large number of slabs arrive by train at a slab yard, where they are stored until transported to the hot rolling mill by a roller table. The slabs need to be transported from the train to the yard and later from the yard to the roller table in the correct order and at specific points in time. Each slab has distinct properties, so we need to consider each slab as being unique.

Figure 1 shows an overview of the slab yard. The two gantry cranes are used to move slabs from one stack to another. As seen in the figure, both the train and the roller table can be modeled as special sets of temporary stacks. The generic slab yard under consideration in this paper consists of 16×16 stacks where each stack is a number of slabs on top of each other. Each crane can only carry one

*Corresponding author. Email: adh@imm.dtu.dk. Tel: +45 4525 3388. Fax: +45 4588 2673.

ISSN: 0020-7543 print/ISSN 1366-588X online

DOI: 10.1080/00207540xxxxxxxxx
http://www.informaworld.com

slab at a time and therefore only the top slab of a stack can be moved. The cranes operate in two directions. Horizontally, they run on a pair of shared tracks and can therefore never pass each other in this direction. Vertically, they operate by a trolley attached to the crane, which can move freely from top to bottom.

The problem is approached in a two-stage planning/scheduling conception. The planning problem of the yard is of an abstract nature. Desired locations for slabs are stated without specifying times of slab movement or crane allocations. For a pre-specified time horizon, a desired end state is formulated, i.e. the end positions of the slabs in the yard are determined. We also generate the operations that need to be carried out in order to arrive at this state. The aim of the crane scheduling problem is to concretize the decisions of the planning problem. Operations are allocated to cranes and all operations are sequenced and positioned in time. The final scheduling solution is directly applicable in practice.

The Slab Yard Planning and Crane Scheduling Problem is considered in a similar context by Hansen (2003). The problem is from a shipyard where ships are constructed by welding together steel plates. The plates are stored in stacks in a plate storage facility and are moved by two gantry cranes sharing tracks. A simulator and a control system are developed and implemented in a system to be used as decision support for the crane operators. Another similar problem is presented by Gambardella *et al.* (2001) (based on the work in Zaffalon *et al.* 1998) where containers are transported by cranes in a container terminal.

An immediate advantage of the two-stage approach is that the planning problem and the scheduling problem individually have received considerable attention in the literature.

The literature relating to The Slab Yard Planning Problem is mainly in other areas of slab yard planning and in container stacking. Tang *et al.* (2002) describe a steel rolling mill where slabs need to be transported from a slab yard according to a scheduled rolling sequence. The article builds on the initial work by the same authors (Tang *et al.* 2001). The layout of the slab yard is different from ours and the cranes are located so that they never collide. Another difference is that for each batch, several candidates exist among the slabs, and therefore the objective is to minimize the cost by choosing the right slabs among the candidates. Singh *et al.* (2004) address the same problem and solve it using an improved Parallel Genetic Algorithm. König *et al.* (2007) investigate a similar problem from storage planning of steel slabs in integrated steel production. The problem formulation in the article is kept at a general level to make the model versatile. The stacking problem is considered alone, thereby disregarding the crane schedules. The authors present a greedy construction heuristic and by a linear programming relaxation of a mixed integer formulation of the problem, they are able to quantify the quality of their solutions.

A problem in container stacking with many similarities to the slab stacking problem is described by Dekker *et al.* (2006). A significant difference is that the maximum height of container stacks is 3, whereas the corresponding number in slab stacks is usually considerably larger than this. A number of stacking policies are investigated by means of simulation and in this sense, the work of Dekker *et al.* resembles the work by Hansen (2003) in a container stacking context. Kim and Bae (1998) describe a container stacking problem where a current yard layout is to be reorganized to a new specific layout. The problem is to convert the current bay layout to the desired new layout by moving the fewest possible containers. The problem is decomposed into three sub-problems, namely a bay-matching, a move-planning, and a task-sequencing stage, where the latter two are similar to the two stages that we introduce for The Slab Yard Planning and Crane Scheduling

Problem. Kim *et al.* (2000) consider a similar container stacking problem. See Steenken *et al.* (2004) for a recent review of literature on container stacking.

The Crane Scheduling problem considered here is an example of a Stacker Crane Problem (Frederickson *et al.* 1978) with time windows and multiple cranes. Parallel crane/hoist scheduling has been thoroughly treated in production of electronics, especially in printed circuit board production. In circuit board production, the hoists are used to move products between tanks, where the products are given various chemical treatments. Leung and Zhang (2003) introduce the first mixed-integer programming formulation for finding optimal cyclic schedules for printed circuit board lines with multiple hoists on a shared track, where the processing sequence may be different from the location sequence of the tanks. The solution method itself is not transferable, but several of the elements in the modeling phase are very relevant to the Crane Scheduling problem of the slab yard. These include the formulation of collision avoidance constraints. Collision avoidance constraints are also described in a dynamic hoist scheduling problem by Lamothe *et al.* (1996) and in a fixed sequence production by Che and Chu (2004) and Leung *et al.* (2004).

As it becomes apparent in the following sections, in the present scheduling problem we are able to abstract from the practical context of the problem and consider the problem as a parallel scheduling problem with sequence-dependent setup times. Zhu and Wilhelm (2006) present a recent literature review for this type of scheduling problem.

This article is arranged as follows. In Section 2 the problem is described and the most important properties are extracted. The solution method is divided into two stages, first solving a planning problem and subsequently a scheduling problem. The two problems and their models are described in Section 3 and Section 4, respectively. The solution method itself is described in Section 5 and test results are presented in Section 6. Finally, conclusions and areas for future work are outlined in Section 7.

2. Problem Description

The slab yard is modeled as a large set of slab stacks. As was shown in Figure 1, train wagons and the roller table are also modeled as special stacks. The train is only at the yard for a certain amount of time and hence all slabs must be moved away from the wagons within a specific time window. In principle, we have access to the roller table in multiple positions as shown on Figure 1 (shown as 8 stacks wide). The order of the slabs on the roller table is essential and to ensure that the sequence of slabs leaving the roller table follows the order in which they were brought there, we only allow the cranes to bring slabs to the right-most of the roller table stacks. Further, as there is room for at most 8 slabs on the roller table, we have to wait whenever the roller table is full. As time goes, the slabs are removed from the roller table in a first-in, first-out manner. For each slab to be moved from the yard in a near future, we have the production time, *Aim Leave Time* (ALT). By looking forward 8 slabs in the sequence, we know when there will be free room for a new slab on the roller table, and this gives us the *Earliest Leave Time* (ELT). Hence, we have a time window for moving the specific slab to the roller table. Slabs which are not a part of the immediately following production instead have an *Estimated Leave Time* (EST).

Each move consists of lift time, transportation time, and drop time. We assume that the cranes move at constant velocity. Transportation time is equal to the maximum of the vertical and the horizontal transportation time, as the cranes are able to move horizontally at the same time as the crane trolley is moving in a

vertical direction.

2.1 *Objective*

The objective of the schedule is to minimize maximum tardiness (delay). The reason is as follows. Take all slabs leaving the slab yard within the scheduling horizon. Whenever a slab is not moved to the roller table before its Aim Leave Time, it causes a delay in the production. The production is not immediately able to catch up on this delay and therefore subsequent slabs are needed later in the production than we initially anticipated. Production is further delayed, only if subsequent slabs are delayed even more. Hence, the most delayed slab determines the quality of the solution.

A feasible schedule consists of a sequence of *operations* with crane allocation and time specification, i.e.: Crane X picks up slab Y (at its current location) at time T and moves it to position Z. Naturally, none of the operations are allowed to conflict with other operations, neither within the schedule of one crane nor between the two cranes.

2.2 *A Simple Example*

Before describing the details of the decomposition, we introduce an illustrative example to clarify the concepts and ideas that are introduced in the following sections.

Example 2.1 An overview of a small yard is shown in Figure 2. The example is similar to the one shown in Figure 1, only significantly smaller.

In this example, we have a scheduling horizon of $[0, 22]$. A side view of the initial yard state is shown in Figure 3. Note that the vertical dimension of Figure 2 is not visible in the figure. However, in the figure, we are able to illustrate the exact composition of each stack. The yard consists of a single arrival stack, T_{ar1}, four stacks in the main yard, $T_1, ..., T_4$, and one exit stack, T_{exit}. In the yard are 14 slabs, $S_1, ..., S_{14}$.

3. The Slab Yard Planning Problem

In the planning stage of the algorithm, we generate a plan that takes us from the current state of the yard to a final state for the horizon. In the final state, all slabs with a deadline within the horizon are brought to the roller table. At the same time, the plan should leave the yard in the best possible condition for subsequent planning periods.

To arrive at a feasible and superior plan within reasonable computational time, the idea is to relax a number of the original constraints in the planning stage. Whatever is relaxed here is fixed in the scheduling stage so that the final solution is always fully descriptive.

3.1 *Operations*

In the planning stage, we are going to consider a solution as defined by a number of successive operations. An operation contains the following information:

Slab The slab to be transported.
Destination The stack where the slab is put on top.

Priority How important is it to include this operation in the final schedule.

A solution to the planning problem consists of a sequence of operations. Many operations are related directly to slabs which are moved to the roller table. Such operations are compulsory and hence have a priority of ∞. As is described in Section 4.3, some operations are optional, however, and the priorities give an ordering of their importance.

For a planning solution to be feasible, we require the following:

- All slabs with a deadline within the scheduling horizon are transported to the exit stack in the correct order.
- All incoming slabs (i.e. slabs in the train wagon stacks) must be moved to permanent stacks.
- All operations must be valid in the sequence. Only slabs on top of a stack may be moved and only to stacks where the maximum stack height has not been reached.

The two first criteria are easy to verify, when we know the set of incoming slabs and the set of outgoing slabs. The third criterion can be verified by updating a yard state as the sequence of operations is processed.

3.2 *Assessment Criteria*

To assess the quality of a plan, we introduce a number of objectives. The following properties characterize a good solution. The two first are directly concerned with the plan, where the two last evaluate the end state of the yard.

- The number of operations is low.
- Operations do not span too far vertically. Even though the operations are not allocated to cranes yet, we would like a solution to accommodate such an allocation. Operations are faster and have less risk of conflicting when they span over as little vertical space as possible.
- Slabs that are to leave the yard soon (but after the current horizon) are close to the exit stack.
- The number of *false positions* is low. Slabs in false positions are slabs that are in the way of other slabs below them. A false position in our context is a stochastic term, as many slabs only have an estimated leave time. We introduce probabilities to approximate the number of false positions in the non-deterministic context. The leave time is represented by a Gaussian distribution. The probability of one slab leaving before another is found by inspecting the cumulative distribution of the difference between the two distributions. The probability of a false position is calculated as the sum over all slabs in the stack, where correlation between the distributions is also taken into account. Details are found in (Dohn and Clausen 2008).

The criteria stated above are quantified suitably for each individual real-world application.

Example (2.1 continued). A planning solution to Example 2.1 is seen in Figure 4. The solution consists of a sequence of operations. The end state of the storage is fully determined by the planning solution and is shown in Figure 5.

4. The Crane Scheduling Problem

From a solution to the planning problem, the goal is now to generate a complete and feasible schedule. First, the ordering of operations is relaxed to allow for parallel execution of operations. Most operations are locally independent of each other. These independencies are detected and only meaningful precedence constraints are kept for the scheduler. The crane scheduling problem is similar to a traditional parallel scheduling problem. We have a number of operations that we need to allocate to two cranes (machines). Between operations there are several temporal constraints. The anti-collision constraint is an important temporal constraint added by the fact that we have two cranes in operation. As the crane operation times are of a stochastic nature, we also need to introduce buffers. The buffers ensure that no crane collision occurs, even with disturbances in operation time. For major disturbances, the scheduling problem and possibly the whole planning may have to be resolved.

4.1 *Precedence Relations*

To ensure that the end state of the schedule is identical with the end state of the planning solution, we establish a number of precedence relations. Using the planning sequence as a starting point, we ensure that, whenever relevant, the order of the operations in the schedule stays the same as in the plan. There are four cases where reordering operations may change the state of the storage and may therefore cause direct or indirect infeasibility of the solution. In these four cases we do not allow reordering of the operations. See Figure 6 for a graphical illustration of the four cases.

Case 1 Moving slab S_2 to a stack from which slab S_1 was moved away from earlier. If the order of these two operations is changed, S_2 is going to block S_1 and the solution becomes infeasible.

Case 2 Moving slab S_1 and then slab S_2, where S_1 is on top of S_2. Again, changing the order of the two operations leads to infeasibility.

Case 3 Moving the same slab twice. If the order of such two moves is changed, the final destination of the slab may change. If the slab is moved again at a later time the final destination, however, remains unchanged.

Case 4 Two slabs S_1 and S_2 are moved to the same stack. If the order is changed it may lead to infeasibility later. If the two slabs are not moved later, the end state is altered, but the solution remains feasible.

Example (2.1 continued). Going back to Example 2.1, we are now able to determine the precedence relations of the plan. Using the four cases depicted in Figure 6 we arrive at the precedences of Figure 7.

4.2 *Temporal Constraints*

The precedence constraints described in the previous section ensure that the end state of a parallel schedule is the same as the corresponding sequential plan. We still need to introduce temporal constraints to create a schedule that is feasible with respect to the individual movement of a crane and to create a schedule which is collision free.

For two operations i and j, we have four positions that have to be considered and where temporal constraints may have to be added correspondingly. The four positions are:

T_i^{orig}	Origin stack of operation i
T_i^{dest}	Destination stack of operation i
T_j^{orig}	Origin stack of operation j
T_j^{dest}	Destination stack of operation j

In the following, we say that i is before j, if i enters and leaves the conflict zone between the two operations, before j. When two operations are allocated to the same crane, the crane needs to complete the first operation before initiating the next. In this case, if i is before j, this means that operation i is completed before operation j is initiated. However, if the operations are allocated to different cranes, they may have a small conflict zone. Hence, even if operation i is before operation j, this does not necessarily mean that it is neither initiated first nor completed first. It only means that it will be the first of the two moves in any of their conflict positions. If two operations have no conflict zone, it is irrelevant whether i is considered to be before j or vice versa.

In the following, we calculate the required gap between two operations i and j, when i is before j. The gap is defined as the amount of time required from initiation of operation i to initiation of operation j. There are three different types of gaps depending on the crane allocation of operations i and j.

g_{ij}^s	Required gap when i and j are allocated to the same crane (s).
g_{ij}^l	Required gap when i is allocated to the left crane (l) and j to the right crane.
g_{ij}^r	Required gap when i is allocated to the right crane (r) and j to the left crane.

The following generalized precedence constraint is imposed: $t_i + g_{ij} \leq t_j$, where g_{ij} represents g_{ij}^s, g_{ij}^l or g_{ij}^r according to the situation. To calculate the gaps between operations, we need to introduce a number of parameters:

p_i	time required to pick up slab of operation i.
q_i	time required to drop off slab of operation i.
$m_{T_xT_y}$	time required to move a laden crane from stack T_x to stack T_y.
$e_{T_xT_y}$	time required to move an empty crane from stack T_x to stack T_y.
b	buffer time required between two cranes.

We assume that $m_{T_xT_y}$ and $e_{T_xT_y}$ are linear in the distance traveled. Both measures are independent of the crane involved. In the following we will use the assumption that the two cranes move at the same speed. We will also assume that a crane cannot move faster when laden than when it is empty.

Precedence relations are included in the generalized precedence constraints, so the values of g_{ij}^s, g_{ij}^l and g_{ij}^r hold all the information we need with respect to precedence constraints. If precedence relations disallow the execution of operation i before operation j, we set: $g_{ij}^s = g_{ij}^l = g_{ij}^r = \infty$.

When two operations are allocated to the same crane, we need to make sure that there is sufficient time to finish the first operation and to move to the start position of the second operation. Consequently, we get:

$$g_{ij}^s = p_i + m_{T_i^{orig}T_i^{dest}} + q_i + e_{T_i^{dest}T_j^{orig}}$$

See Figure 8 for a visualization of this. We use Time-Way diagrams that are frequently used when depicting solutions of crane/hoist scheduling problems, especially in printed circuit board production (see e.g. Liu and Jiang 2005). The

horizontal and vertical axes in the diagram represent the time and crane positions, respectively. Solid lines indicate that the crane is processing an operation, whereas dashed lines indicate that the crane is either waiting or moving to the start position of the next operation.

When two operations are allocated to two separate cranes, we need to make sure that the cranes never collide. Further, as we are dealing with a highly stochastic system, we introduce the concept of a buffer. The buffer denotes the amount of time we require between two cranes traversing the same position. By introducing a buffer we establish a certain degree of stability in the schedule. If one of the cranes is delayed by an amount of time less than the buffer size the schedule is still guaranteed to be feasible. The buffer size is set so that infeasibility only occurs in rare cases. In the following, a violation of the prespecified buffer size is considered to be a collision. In the Time-Way diagrams, the buffer is illustrated by a shaded area.

In Table 1 we describe how to calculate g^l_{ij}. For g^l_{ij}, the left crane is allocated to operation i and the right crane to operation j. In the event of conflict between the two operations, operation i enters and leaves the conflict zone before operation j. There are five different cases to be considered. These are shown in Table 1 and in Figure 9 - Figure 13. $(l2)$ and $(l3)$ may both apply at the same time and in that case g^l_{ij} is equal to the larger of the two values. The comparison of two stacks is done with respect to their horizontal position, e.g. $T^{orig}_i < T^{orig}_j$ means the origin stack of operation i is to the left of origin stack of operation j.

It is clear from each of the five figures (Figure 9 - Figure 13) why a violation of the constraint introduces a collision. These five cases are sufficient for avoiding all possible collisions. The proof is found in the technical report (Dohn and Clausen 2008).

In Table 2, we show how to calculate g^r_{ij}. The calculations are analogous to the ones of g^l_{ij}. Operation i is now allocated to the right crane and j to the left crane. Again, in case of any conflict between the two operations, operation i is before operation j. All coordinates are mirrored, which does not affect any of the movement times and hence the calculations are very similar.

Figure 14 illustrates how $(r1)$ is closely related to $(l1)$. The only difference is the precondition, which is mirrored.

Example (2.1 continued). With these definitions, we can illustrate how to calculate the coefficients for the generalized precedence constraints of Example 1. We have the three sets of coefficients: g^s_{ij}, g^l_{ij}, and g^r_{ij} represented by the three matrices of Table 3. First, we use the precedence constraints of Figure 7 to fill in the ∞-values. This includes the entailed precedence constraints (e.g. $a_1 \rightarrow a_2 \wedge a_2 \rightarrow a_3 \Rightarrow a_1 \rightarrow a_3$). In this example, we have for all operations: $p_i = 1, q_i = 1$, and $b = 1$. $m_{T_xT_y}$ and $e_{T_xT_y}$ are equal and are set to the horizontal distance between the two stacks, cf. Figure 3 (e.g. $m_{T_1T_{exit}} = 4$). Three examples of the calculations for the matrices are shown below ($g^r_{a_1a_2}$ is calculated from $(r2)$+$(r3)$ and $g^r_{a_1a_4}$ is calculated from $(r4)$).

$$g^s_{a_2a_3} = p_{a_2} + m_{T^b_{a_2}T^e_{a_2}} + q_{a_2} + e_{T^e_{a_2}T^b_{a_3}} = 1 + 3 + 1 + 1 = 6$$
$$g^r_{a_1a_2} = \max\{p_{a_1} + m_{T^b_{a_1}T^e_{a_1}} + q_{a_1} + b - (p_{a_2} + m_{T^b_{a_2}T^e_{a_1}}), p_{a_1} + m_{T^b_{a_1}T^b_{a_2}} + b\}$$
$$= \max\{1 + 1 + 1 + 1 - (1 + 1), 1 + 0 + 1\} = 2$$
$$g^r_{a_1a_4} = p_{a_1} + m_{T^b_{a_1}T^e_{a_4}} + b - (p_{a_4} + m_{T^b_{a_4}T^e_{a_4}}) = 1 + 0 + 1 - (1 + 2) = -1$$

4.3 *Operation Priority*

So far we have assumed that all operations had to be included in the final schedule. However we may deviate slightly from this strategy. We introduce the concept of *optional operations*. We may add a number of optional operations to the end of the plan. The purpose of adding these operations is to enhance the final state of the slab yard, so that the risk of delays in future scheduling is reduced.

Example (2.1 continued). In Example 2.1, we could e.g. add to the solution of Figure 4 the operation $(S_7 \to T_4)$ as an optional operation. This would give a final yard state with fewer false positions.

To quantify the importance of the individual optional operations we use the *operation priority*. A high priority means that we prefer the inclusion of this operation to other optional operations with lower priority. In this work, the priority is calculated as follows. We consider the two possible end states that will be entailed by respectively including or excluding the operation from the plan. For both states it is possible to calculate the number of false positions in the yard. As described in Section 3.2, the number of false positions, in our case, is a stochastic measure. The difference between the two sums, i.e. the possible gain in number of false positions, is used as the operation priority. We would never include an operation which increases the number of false positions.

4.4 *Objective Function*

As was described in Section 2.1, the objective function is to minimize the maximum tardiness of the schedule. At the same time, a good schedule includes many optional operations. The sum of the priorities of the optional operations included is used to evaluate this criterion. The individual priorities of operations are determined by the planning module, as described in the previous section. The objective function is two-layered so that minimization of maximum tardiness is always prioritized over the second objective. However, we still require all operations with priority ∞ (compulsory operations) to be in the schedule.

4.5 *Generic Formulation of the Crane Scheduling Problem*

We are now able to abstract fully from the real-world context and introduce an explicit formulation of the Crane Scheduling Problem as a parallel scheduling problem with generalized precedence constraints, non-zero release times, and sequence-dependent setup time. Using the three-field notation of Graham *et al.* (1979) ex-

tended by Brucker *et al.* (1999) and Allahverdi *et al.* (2008) we denote the problem $R2|temp, r_j, s_{ijm}|T_{max}$.

Sets:

$\mathcal{O}$	The set of operations.
$\mathcal{C} = \{C^l, C^r\}$	The two cranes, left crane and right crane respectively.

Decision variables:

$x_i \in \mathbb{B}$	$i \in \mathcal{O}$	1 if operation i is included in the schedule, 0 otherwise.
$t_i \in \mathbb{Z}$	$i \in \mathcal{O}$	Start time of operation i.
$c_i \in \mathcal{C}$	$i \in \mathcal{O}$	The crane allocation of operation i.
$y_{ij} \in \mathbb{B}$	$i \in \mathcal{O}, j \in \mathcal{O}$	1 if the operation i is considered to be before operation j, 0 otherwise.
$\tau_i \in \mathbb{Z}$	$i \in \mathcal{O}$	Tardiness of operation i.

Parameters:

$g^s_{ij} \in \mathbb{Z}$	$i \in \mathcal{O}, j \in \mathcal{O}$	The required gap between operations i and j when allocated to the same crane and i is before j.
$g^l_{ij} \in \mathbb{Z}$	$i \in \mathcal{O}, j \in \mathcal{O}$	The required gap between operations i and j when allocated respectively to the left crane and the right crane and i is first in a conflict.
$g^r_{ij} \in \mathbb{Z}$	$i \in \mathcal{O}, j \in \mathcal{O}$	The required gap between operations i and j when allocated respectively to the right crane and the left crane and i is first in a conflict.
r_i	$i \in \mathcal{O}$	Release time of operation i.
d_i	$i \in \mathcal{O}$	Due date of operation i.
p_i	$i \in \mathcal{O}$	Priority (weight) of operation i.
t^{max}_i	$i \in \mathcal{O}$	Deadline of operation i.

The Constraint Programming Model:

$$\min \max \tau_a \text{ and secondly } \max \sum_{i \in A} p_i x_i \tag{1}$$

$$\tau_i = \max\{0, t_i - d_i\} \qquad \forall i \in \mathcal{O} \tag{2}$$

$$x_i = 1 \wedge x_j = 1 \Rightarrow y_{ij} = 1 \vee y_{ji} = 1 \qquad \forall i \in \mathcal{O}, \forall j \in \mathcal{O}, i \neq j \tag{3}$$

$$t_i \geq r_i \qquad \forall i \in \mathcal{O} \tag{4}$$

$$t_i \leq t^{max}_i \qquad \forall i \in \mathcal{O} \tag{5}$$

$$y_{ij} = 1 \wedge c_i = C^l \wedge c_j = C^l \Rightarrow t_i + g^s_{ij} \leq t_j \qquad \forall i \in \mathcal{O}, \forall j \in \mathcal{O} \tag{6}$$

$$y_{ij} = 1 \wedge c_i = C^r \wedge c_j = C^r \Rightarrow t_i + g^s_{ij} \leq t_j \qquad \forall i \in \mathcal{O}, \forall j \in \mathcal{O} \tag{7}$$

$$y_{ij} = 1 \wedge c_i = C^l \wedge c_j = C^r \Rightarrow t_i + g^l_{ij} \leq t_j \qquad \forall i \in \mathcal{O}, \forall j \in \mathcal{O} \tag{8}$$

$$y_{ij} = 1 \wedge c_i = C^r \wedge c_j = C^l \Rightarrow t_i + g^r_{ij} \leq t_j \qquad \forall i \in \mathcal{O}, \forall j \in \mathcal{O} \tag{9}$$

$$p_i = \infty \Rightarrow x_i = 1 \qquad \forall i \in \mathcal{O} \tag{10}$$

$$x_i \in \mathbb{B}, t_i \in \mathbb{Z}, c_i \in \mathcal{C}, y_{ij} \in \mathbb{B}, \tau_i \in \mathbb{Z} \qquad \forall i \in \mathcal{O}, \forall j \in \mathcal{O} \tag{11}$$

The model (1)-(11) captures all the problem properties that have been described in this section. (1) is the objective function, which has two criteria. (2) sets the

tardiness of each operation. (3) ensures that if both operations are included in the schedule, then their internal precedence constraints must be respected either in one direction or the other. Operations cannot be started before their release date (4) and must be scheduled within the horizon (5). (6)-(9) connect the decision on crane allocation with the correct precedence constraints. Finally, (10) makes sure that all compulsory operations are included in the schedule, and (11) gives the domains of the decision variables.

The parameter p_i is calculated by the planning module, as explained previously. r_i and d_i are calculated as $r_i = ELT_i - dur_i$ and $d_i = ALT_i - dur_i$, where dur_i is the duration of an operation, i.e. $dur_i = p_i + m_{T_i^{orig} T_i^{dest}} + q_i$. g_{ij}^s, g_{ij}^l, and g_{ij}^r are calculated as described in Section 4.2.

Example (2.1 continued). We consider Example 2.1 again. We have the 5 operations of Figure 4 which make up the set of operations $\mathcal{O} = \{o_1, o_2, o_3, o_4, o_5\}$. We have the precedence coefficients from Table 3. The duration of each operation and subsequently the release date and due date of each operation is calculated in the table. In this example, the scheduling horizon is $0 \leq t_i \leq 22$:

Operation (i)	dur_i	r_i	d_i	t_i^{max}	p_i
o_1	3	0	19	19	∞
o_2	5	0	5	17	∞
o_3	3	8	9	19	∞
o_4	4	0	18	18	∞
o_5	4	0	18	18	∞

An optimal solution is (Solution 1):

Operation (i)	x_i	t_i	c_i	y_{io_1}	y_{io_2}	y_{io_3}	y_{io_4}	y_{io_5}	τ_i
o_1	1	0	C^r	–	1	1	1	1	0
o_2	1	2	C^l	0	–	1	1	1	0
o_3	1	9	C^r	0	0	–	1	1	0
o_4	1	12	C^l	0	0	0	–	1	0
o_5	1	18	C^l	0	0	0	0	–	0

Another solution is (Solution 2):

Operation (i)	x_i	t_i	c_i	y_{io_1}	y_{io_2}	y_{io_3}	y_{io_4}	y_{io_5}	τ_i
o_1	1	0	C^r	–	1	1	1	1	0
o_2	1	4	C^r	0	–	1	1	1	0
o_3	1	10	C^r	0	0	–	1	1	1
o_4	1	3	C^l	0	0	0	–	1	0
o_5	1	9	C^l	0	0	0	0	–	0

Solution 1 of Example 2.1 is visualized in a Gantt chart in Figure 15 (top). The Gantt chart does not depict the value of either y_{ij}-variables or τ_i-variables.

Solution 2 is illustrated in Figure 15 (bottom). The solution is more compact and may actually look more attractive. However, the due date of o_3 is violated and therefore this solution is worse than Solution 1.

The nature of the problem makes the transitive closure valid for all choices of priority on conflicting operations, i.e. if operation i is before j (with respect to conflicts) and j is before k then we may assume that i is before k ($y_{ij} = 1 \wedge y_{jk} = 1 \Rightarrow y_{ik} = 1$). We also find this property in lists and therefore we can use a list

to represent all sequencing decisions. If, further, we state the crane allocation of each operation, c_i, and if we assume that all operations are scheduled at the earliest possible time according to the given sequence and the crane allocations, then the list representation is sufficient to explicitly represent the solution. The earliest possible times are found in polynomial time by running through the list. For every operation i the generalized precedence constraints to all preceding operations are checked and the most limiting of those determine the starting time of operation i. We adapt the graphical representation from the planning solutions but add information on the crane allocation. We still lack information on t_i and τ_i and therefore the objective function of the solution is not immediately available, but it can be calculated by running through the list. The two solutions from before are represented as seen on Figure 16.

The advantage of this representation is clear from Figure 16. The only difference between the two solutions is the change in crane allocation of operation o_2. All other variable changes (that were observed on Figure 15) can be interpreted as consequences of this variable change. Another nice feature of the list representation is that any permutation that respects all precedence relations is also feasible with respect to (2)-(4) + (6)-(11). Only the scheduling horizon is possibly violated.

Another way of visualizing a scheduling solution graphically is by using Time-Way diagrams as introduced in Section 4.2. Solution 1 and Solution 2 are depicted in Figure 17.

5. Solution Method

A solution method has been implemented based on the presented model. In the following, we present two greedy methods, one for the planning problem and one for the scheduling problem. The two methods are straight-forward in their implementation and more sophisticated methods will probably enhance performance. However, the simple methods are still able to generate good results, and so we will use them to assess the value of the model.

5.1 *Planning*

The planning method will provide a plan as described in Section 3. When the final schedule is created in the second stage of the method, all precedence constraints are respected, so the sequence of operations that we specify in the planning solution fully determines the state of the yard. As a consequence, we are able to update the yard state as the operations are added to the plan. For any partial plan, we have a current yard state. When we refer to the location of slabs, it is with respect to the current state of the yard.

The implemented method is divided into three steps:

- Generation of operations for slabs that must leave the yard during the scheduling horizon (*exit slabs*).
- Generation of operations for incoming slabs (*arrival slabs*).
- Generation of optional operations.

In this solution method, we treat the three steps separately, one by one, in a state space exploration.

First, we generate a list of operations for the slabs that must leave the yard during the scheduling horizon. We already have their Aim Leave Time and hence we have a predetermined ordering of these operations. The slabs may not be on top of their stacks, so we may also need to generate *reshuffle* operations and move the

slabs on top to other stacks. For reshuffle operations we must specify a destination stack. The destination stack is chosen from a number of criteria. First, we disallow movement to stacks still containing exit-slabs. Moving a slab to such a stack will trigger another reshuffle operation later, where the same slab has to be reshuffled again. This should be avoided if possible. Further, when choosing a destination stack, we look for stacks within a short horizontal range. This limits the duration of the operation and at the same time decreases the risk of crane collision involving this operation. We also look for stacks, where the slab has a small chance of being in a new false position (and hence in need of another reshuffle in a future plan).

When all exit operations have been generated, we proceed with the arrival operations. For each slab on the railway wagons, we generate an operation that will bring the slab to the yard. When choosing a destination for these slabs it is particularly important to keep the sum of false position probabilities low. All arrival operations are sequenced after the exit operations. This does not necessarily mean that they are also scheduled later than all arrival operations. The reason for sequencing the operations in this way in the planning solution is that all stacks involved in both exit and arrival operations will have the exit operation executed first, which is obviously a desirable feature. To introduce flexibility in the scheduler, we try to select destination stacks that do not have any outgoing exit operations. The order of arrival operations is partially predetermined. We have to move the slabs from top to bottom from the stacks on the railway wagons. However, we have a choice between the arrival stacks.

Finally, we generate a number of operations that are not mandatory for the feasibility of schedules, but that will increase the quality of the solution by reducing the total false position probability and may also move slabs with an upcoming due date closer to the exit stack. The optional operations are always added at the end of the plan to ensure that the remaining plan is feasible, even if some of the optional operations are not included in the schedule.

5.2 *Scheduling*

Given a planning solution, we need to schedule the operations on the two cranes. The generic formulation of the problem is given in Section 4.5. In the following we describe a greedy heuristic on which the current implementation is based.

The heuristic is very simple. We process the operations in the order given in the planning solution. For each operation, the earliest possible time of initiation is calculated for both cranes. The operation is allocated to the crane that is able to initiate first. As we have release times for operations, there may still be some waiting time from the preceding operation to the current one. Therefore, we check if we are able to squeeze in any of the unscheduled operations. The operations with high priority are preferred to the others. When squeezing in operations like this, we need to make sure that all precedence constraints are respected.

The heuristic is greedy and may therefore make decisions, which are not advantageous in the end. This issue could be addressed by the implementation of a local search procedure to enhance the results of a greedy construction heuristic. As a starting point, a steepest descent algorithm would probably increase quality significantly. Adding metaheuristic features to such a search will enhance the solution even further. Preliminary test results from a metaheuristic show promising results, and are reported in the following section.

6. Test results

To evaluate the quality of the solutions, we generate a reference solution that represents the solution obtainable by manual planning for each instance. We try to imitate the behavior of the cranes when they are under the control of the individual crane operator. The operators work on an ad-hoc basis. We expect them to deal with exit slabs as we approach their deadlines and reshuffles are carried out when needed. More specifically, we equip the crane operators with a two hour foresight. Slabs that are to leave within this period will not be blocked by new slabs. If a crane has free time in between moves, it will use the time to move slabs from the train wagons to the yard. In the schedules described earlier, we needed a buffer between cranes to make the schedules more robust. To the advantage of this simulation, the buffer is disregarded in this part, as we are not really creating a schedule. Rather, we are simulating manual planning/scheduling, and hence the operations are to be interpreted as happening in real time and not as a pre-made schedule to be followed. Again, a more detailed description is found in the technical report (Dohn and Clausen 2008).

By comparing the solutions of the method presented in this paper to the solutions of such a simulation, we are able to assess the value of the proposed method. In the following we run a number of simulations. The average yard throughput is fixed in each of the test instances. The throughput is increased in the hard test instances to check the effect on the quality of the schedules. The simulations are kept as close as possible to the real world conditions. The details of the data simulation and the settings for simulation of manual planning are in the technical report (Dohn and Clausen 2008).

For the simulations, we assume that the requested throughput of the yard for each day is randomly drawn from a Gaussian distribution. In the same way, we assume that the production time and through time (i.e. storage time in the yard) for each slab are also drawn from Gaussian distributions.

From Table 4 it is clear that the proposed method provides significantly better results than the simulation of manual planning. In the table we have shown four performance measures. For each method, the first column gives the average number of times a slab is moved before it leaves the yard. As slabs in our setup are never transferred directly from train wagons to the exit belt, the minimum number of moves of each slab is 2. This measure illustrates how well the moves are planned, i.e. a low number indicates that the slabs are seldom in the way of others. From the tests, we see a significant difference between the two methods, especially for the harder problems, where the Two-Stage algorithm on average uses approximately one move less per slab. Also the duration of each move is of interest and is shown in the second column. The difference between the two methods is not remarkable, even though the Two-Stage algorithm is a few seconds faster in all cases. The duration seems stable over the set of test instances.

The two last columns report on deadline violations. The rightmost of the columns gives the maximum deadline violation, which is, as stated earlier, the main objective considered in this work. The first of the two columns reports on the average violations. This is interesting if we assume that the following production is able to catch up on the delays we may have caused. A low average deadline violation is equivalent to a low sum of violations, which is another objective often used for scheduling problems in the literature. For both objectives, we see that the Two-Stage algorithm clearly outperforms the other. The manual planning has severe problems in the hard instances, where the results of the Two-Stage method are still satisfactory. The figures for manual simulation may seem very large, but it

is noted that the numbers should only be used for comparison with other similar tests. As soon as a method is unable to keep up with the rate at which slabs enter the yard, it will lead to larger and larger violations as we let the simulation run. In each run of these tests, the two methods naturally span over the same production plan.

Both methods are able to produce results in less than a second. Such computation times are insignificant in these settings and are therefore not compared here.

Figure 18 illustrates a schedule created by the Two-Stage algorithm.

7. Conclusions

The Slab Yard Planning and Crane Scheduling Problem has been modeled in a novel way that facilitates a beneficial, and at the same time transparent optimization. The model is generic enough to capture several variations of the problem. The solution methods adapt to variations of the problem, correspondingly.

From the test results it is clear that the model facilitates an algorithm that is capable of providing solutions superior to those achievable by manual planning. The tests are, however, preliminary and based on simulations which rely on a number of assumptions.

We have introduced a model that, by splitting The Slab Yard Planning and Crane Scheduling Problem into two stages, facilitates a solution procedure that is clear in the formulation of objectives and is able to generate superior schedules by addressing the problem at two different abstraction levels.

Future work should be aimed at real-world applications. So far, the experimental conclusions are based on simulations and artificially generated data. In a practical application it is possible to tailor the algorithms to fit the exact properties of that particular problem. In this paper, we have made sure not to take advantage of structures in the problem data, as such structures may not transfer to variations of the problem. Therefore, in a practical application, it may be possible to utilize problem-specific knowledge in the creation of the planning and the scheduling method, and get even better results. On the other hand, practical problems may also introduce new challenges. Either way, the model presented in this paper will be a valuable starting point for exhaustive purpose-built practical models.

References

Allahverdi, A., Ng, C., Cheng, T. and Kovalyov, M., 2008. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187 (3), 985–1032.

Brucker, P., Drexl, A., Mohring, R., Neumann, K. and Pesch, E., 1999. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112 (1), 3–41.

Che, A. and Chu, C., 2004. Single-track multi-hoist scheduling problem: a collision-free resolution based on a branch-and-bound approach. *International Journal of Production Research*, 42 (12), 2435–2456.

Dekker, R., Voogd, P. and Asperen, E., 2006. Advanced methods for container stacking. *OR Spectrum - Quantitative Approaches in Management*, 28 (4), 563.

Dohn, A. and Clausen, J., Optimizing the Slab Yard Planning and Crane Scheduling Problem Using a Two-Stage Approach (Technical Report). , 2008. , Technical report, Technical University of Denmark.

Frederickson, G., Hecht, M. and Kim, C., 1978. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7 (2), 178–93.

Gambardella, L., Mastrolilli, M., Rizzoli, A. and Zaffalon, M., 2001. An optimization methodology for intermodal terminal management. *Journal of Intelligent Manufacturing*, 12 (5-6), 521.

Graham, R., Lawler, E., Lenstra, J. and Rinnooy Kan, A., 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Discrete Optimisation*, 5, 287–326.

Hansen, J., 2003. Industrialised application of combinatorial optimization. Thesis (PhD). Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby.

Kim, K. and Bae, J., 1998. Re-marshaling export containers in port container terminals. *Computers and Industrial Engineering*, 35 (3-4), 655–658.

Kim, K., Park, Y. and Ryu, K.R., 2000. Deriving decision rules to locate export containers in container yards. *European Journal of Operational Research*, 124 (1), 89–101.

König, F., Lübbecke, M., Möhring, R., Schäfer, G. and Spenke, I., 2007. Solutions to real-world instances of PSPACE-complete stacking. *Algorithms - ESA 2007. Proceedings 15th European Symposium. (Lecture Notes in Computer Science vol. 4698)*, 729–40.

Lamothe, J., Thierry, C. and Delmas, J., 1996. A multihoist model for the real time hoist scheduling problem. *Symposium on Discrete Events and Manufacturing Systems. CESA'96 IMACS Multiconference. Computational Engineering in Systems Applications*, 461–6.

Leung, J. and Zhang, G., 2003. Optimal cyclic scheduling for printed circuit board production lines with multiple hoists and general processing sequence. *IEEE Transactions on Robotics and Automation*, 19 (3), 480–484.

Leung, J., Zhang, G., Yang, X., Mak, R. and Lam, K., 2004. Optimal Cyclic Multi-Hoist Scheduling: A Mixed Integer Programming Approach. *Operations Research*, 52 (6), 965–976.

Liu, J. and Jiang, Y., 2005. An efficient optimal solution to the two-hoist no-wait cyclic scheduling problem. *Operations Research*, 53 (2), 313–27.

Singh, K., Srinivas and Tiwari, M., 2004. Modelling the slab stack shuffling problem in developing steel rolling schedules and its solution using improved Parallel Genetic Algorithms. *International Journal of Production Economics*, 91 (2), 135–147.

Steenken, D., Voß, S. and Stahlbock, R., 2004. Container terminal operation and operations research - a classification and literature review. *OR Spectrum*, 26 (1), 3–49.

Tang, L., Liu, J., Rong, A. and Yang, Z., 2001. An Effective Heuristic Algorithm to Minimise Stack Shuffles in Selecting Steel Slabs from the Slab Yard for Heating and Rolling. *Journal of the Operational Research Society*, 52 (10), 1091–1097.

Tang, L., Liu, J., Rong, A. and Yang, Z., 2002. Modelling and a genetic algorithm solution for the slab stack shuffling problem when implementing steel rolling schedules. *International Journal of Production Research*, 40 (7), 1583–95.

Zaffalon, M., Rizzoli, A., Gambardella, L. and Mastroiilli, M., 1998. Resource allocation and scheduling of operations in an intermodal terminal. *Simulation Technology: Science and Art. 10th European Simulation Symposium 1998. ESS'98*, 520–7.

Zhu, X. and Wilhelm, W., 2006. Scheduling and lot sizing with sequence-dependent setup: a literature review. *IIE Transactions*, 38 (11), 987–1007.

	Precondition	Gap
$(l1)$	$T_j^{orig} \leq T_i^{dest}$	$g_{ij}^l = p_i + m_{T_i^{orig}T_i^{dest}} + q_i + e_{T_i^{dest}T_j^{orig}} + b$
$(l2)$	$T_j^{dest} \leq T_i^{dest} < T_j^{orig}$	$g_{ij}^l \geq p_i + m_{T_i^{orig}T_i^{dest}} + q_i + b - (p_j + m_{T_j^{orig}T_i^{dest}})$
$(l3)$	$T_i^{dest} < T_j^{orig} \leq T_i^{orig}$	$g_{ij}^l \geq p_i + m_{T_i^{orig}T_j^{orig}} + b$
$(l4)$	$T_i^{dest} < T_j^{dest} \leq T_i^{orig} < T_j^{orig}$	$g_{ij}^l = p_i + m_{T_i^{orig}T_j^{dest}} + b - (p_j + m_{T_j^{orig}T_j^{dest}})$
$(l5)$	Otherwise	$g_{ij}^l = -\infty$

Table 1. Calculation of the required gap between two operations. Operation i is allocated to the left crane and operation j to the right crane. In conflict, i is moved before j.

	Precondition	Gap
$(r1)$	$T_j^{orig} \geq T_i^{dest}$	$g_{ij}^r = p_i + m_{T_i^{orig}T_i^{dest}} + q_i + e_{T_i^{dest}T_j^{orig}} + b$
$(r2)$	$T_j^{dest} \geq T_i^{dest} > T_j^{orig}$	$g_{ij}^r \geq p_i + m_{T_i^{orig}T_i^{dest}} + q_i + b - (p_j + m_{T_j^{orig}T_i^{dest}})$
$(r3)$	$T_i^{dest} > T_j^{orig} \geq T_i^{orig}$	$g_{ij}^r \geq p_i + m_{T_i^{orig}T_j^{orig}} + b$
$(r4)$	$T_i^{dest} > T_j^{dest} \geq T_i^{orig} > T_j^{orig}$	$g_{ij}^r = p_i + m_{T_i^{orig}T_j^{dest}} + b - (p_j + m_{T_j^{orig}T_j^{dest}})$
$(r5)$	Otherwise	$g_{ij}^r = -\infty$

Table 2. Calculation of the required gap between two operations. Operation i is allocated to the right crane and operation j to the left crane. In conflict, i is moved before j.

g^s_{ij}	a_1	a_2	a_3	a_4	a_5
a_1	$-$	4	4	6	6
a_2	∞	$-$	6	10	10
a_3	∞	∞	$-$	8	8
a_4	∞	∞	6	$-$	6
a_5	∞	∞	6	∞	$-$

g^l_{ij}	a_1	a_2	a_3	a_4	a_5
a_1	$-$	5	$-\infty$	7	7
a_2	∞	$-$	7	11	11
a_3	∞	∞	$-$	9	9
a_4	∞	∞	$-\infty$	$-$	7
a_5	∞	∞	$-\infty$	∞	$-$

g^r_{ij}	a_1	a_2	a_3	a_4	a_5
a_1	$-$	2	5	-1	-1
a_2	∞	$-$	4	-1	-1
a_3	∞	∞	$-$	$-\infty$	$-\infty$
a_4	∞	∞	7	$-$	2
a_5	∞	∞	7	∞	$-$

Table 3. Coefficients of generalized precedence constraints for Example 2.1.

Slabs per day	Two-Stage Avg moves per slab	Two-Stage Avg move duration	Two-Stage Avg deadline violation	Two-Stage Max deadline violation	Manual Avg moves per slab	Manual Avg move duration	Manual Avg deadline violation	Manual Max deadline violation
400	2.37	43.91	0.01	1.66	2.61	46.55	0.24	11.78
450	2.39	43.76	0.02	1.35	2.60	46.28	0.09	8.52
500	2.37	43.84	0.02	1.86	2.61	46.68	5.53	61.13
550	2.38	43.80	0.02	1.86	2.59	46.50	2.94	74.64
600	2.41	43.84	1.08	10.54	2.61	46.62	26.69	234.82
650	2.39	43.91	0.40	7.28	2.58	46.63	27.30	253.55
700	2.38	43.91	0.38	7.51	2.59	46.60	139.64	543.17
750	2.38	43.90	0.72	16.50	2.68	46.61	878.07	1883.29
800	2.39	44.00	2.59	44.61	2.90	46.80	3092.32	4840.96
850	2.40	43.92	1.68	43.33	3.14	46.75	5202.69	7874.27
900	2.41	43.88	13.72	135.50	3.30	46.45	7930.29	13108.21
950	2.39	43.95	47.02	270.68	3.34	46.04	9204.57	15641.77
1000	2.44	43.93	118.06	490.39	3.43	45.72	10704.42	18626.53

Table 4. Comparison of test results from the Two-Stage method and simulation of manual planning. Each value is an average over 10 identical runs. Deadline violations and durations are measured in seconds.

Figure 1. Overview of the slab yard.

Figure 2. Overview of a very simple slab yard used in Example 2.1.

Figure 3. Slab Yard Crane Scheduling Problem: Side-view of the slab yard of Example 2.1. Gray slabs are slabs that must leave the yard during the scheduling horizon. Leaving slabs (gray): [ELT, ALT]. Non-leaving slabs (white): (EST)

Figure 4. A solution to the planning problem of Example 2.1.

Figure 5. End state for the solution of Figure 4.

Figure 6. Graphical description of the state preserving precedences.

Figure 7. Precedence relations for the plan of Figure 4.

Figure 8. Time-Way diagram visualizing the calculation of gap size between two operations executed sequentially by the same crane.

Figure 9. Time-Way diagram visualizing the calculation of gap size between two operations in case ($l1$), where $T_j^{orig} \leq T_i^{dest}$. The no-collision requirement in this case becomes: $t_i + p_i + m_{T_i^{orig} T_i^{dest}} + q_i + e_{T_i^{dest} T_j^{orig}} + b \leq t_j$.

Figure 10. Time-Way diagram visualizing the calculation of gap size between two operations in case ($l2$), where $T_j^{dest} \leq T_i^{dest} < T_j^{orig}$. The no-collision requirement in this case becomes: $t_i + p_i + m_{T_i^{orig} T_i^{dest}} + q_i + b \leq t_j + p_j + m_{T_j^{orig} T_i^{dest}}$.

Figure 11. Time-Way diagram visualizing the calculation of gap size between two operations in case ($l3$), where $T_i^{dest} < T_j^{orig} \leq T_i^{orig}$. The no-collision requirement in this case becomes: $t_i + p_i + m_{T_i^{orig} T_j^{orig}} + b \leq t_j$.

Figure 12. Time-Way diagram visualizing the calculation of gap size between two operations in case ($l4$), where $T_i^{dest} < T_j^{dest} \leq T_i^{orig} < T_j^{orig}$. The no-collision requirement in this case becomes: $t_i + p_i + m_{T_i^{orig} T_j^{orig}} + b \leq t_j + p_j + m_{T_j^{orig} T_j^{dest}}$.

Figure 13. Time-Way diagram visualizing the calculation of gap size between two operations in case ($l5$), where there are no direct temporal relations between operation i and operation j.

Figure 14. The situation of Figure 9 mirrored vertically. Operation i is now allocated to the right crane.

Figure 15. Gantt charts of Solution 1 (top) and Solution 2 (bottom).

Figure 16. List representations of Solution 1 (top) and Solution 2 (bottom).

Figure 17. Time-Way diagrams of Solution 1 (a) and Solution 2 (b).

Figure 18. Schedule created by the Two-Stage algorithm. The two curves describe the horizontal positions of the two cranes over time. Each curve contains a number of pick-ups ($\times$) and drop-offs ($\circ$).

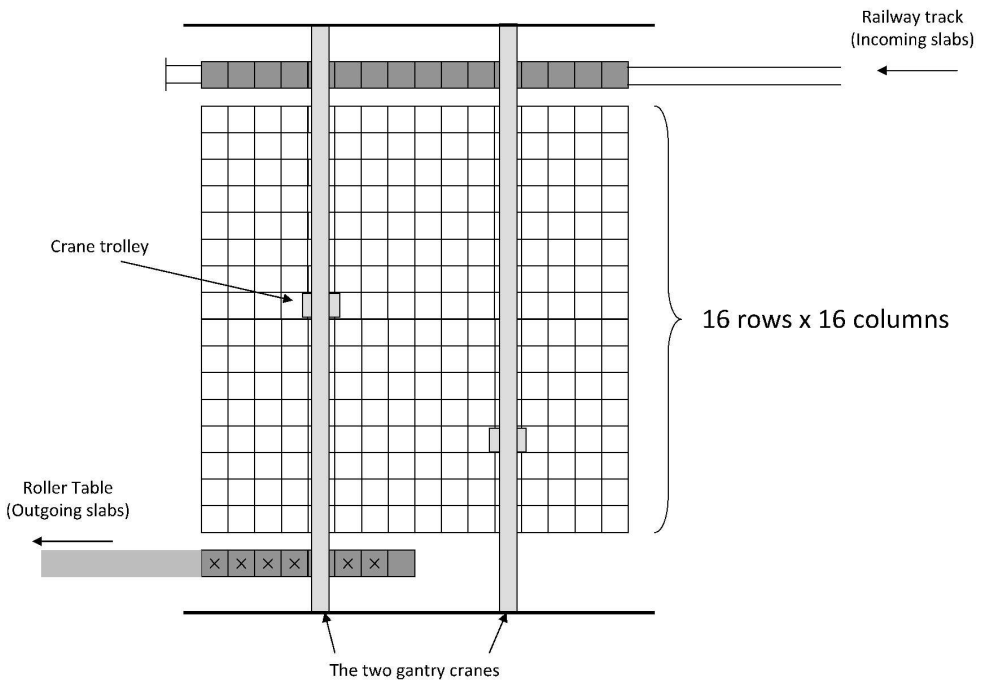

Figure 1
225x153mm (600 x 600 DPI)

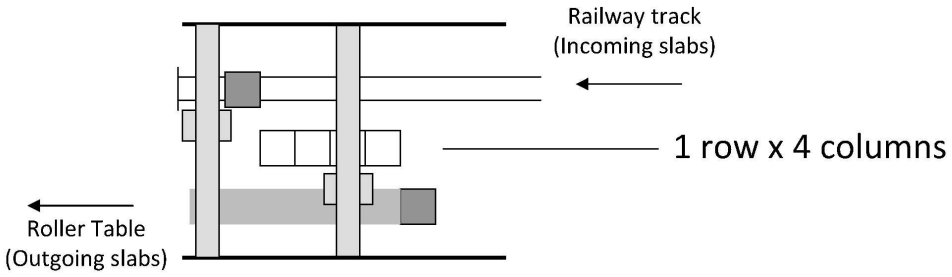

Figure 2
171x48mm (600 x 600 DPI)

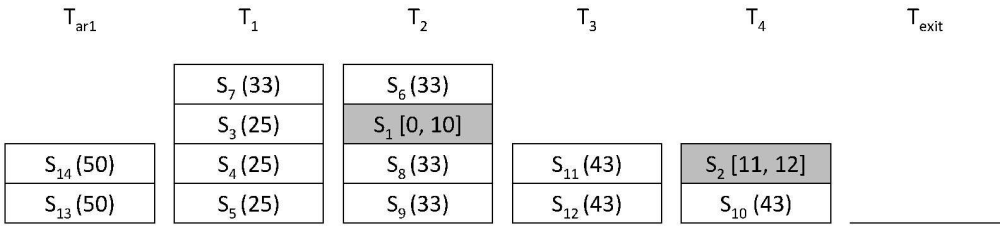

Figure 3
201x46mm (600 x 600 DPI)

o_1: ($S_6 \rightarrow T_3$)	o_2: ($S_1 \rightarrow T_{exit}$)	o_3: ($S_2 \rightarrow T_{exit}$)	o_4: ($S_{14} \rightarrow T_2$)	o_5: ($S_{13} \rightarrow T_2$)

Figure 4
152x10mm (600 x 600 DPI)

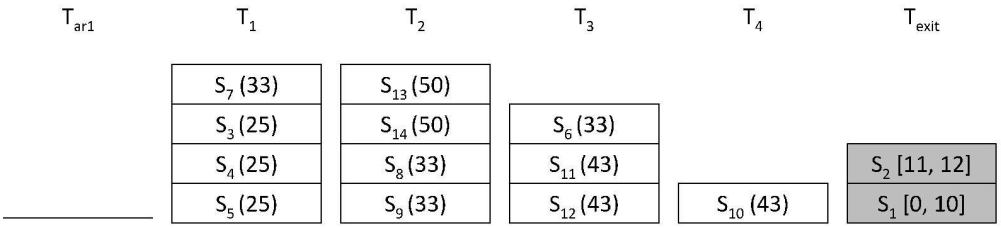

Figure 5
201x46mm (600 x 600 DPI)

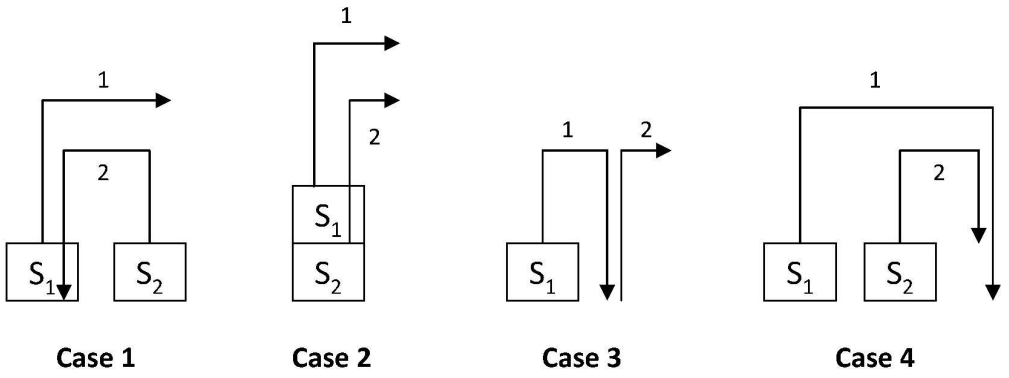

Figure 6
140x53mm (600 x 600 DPI)

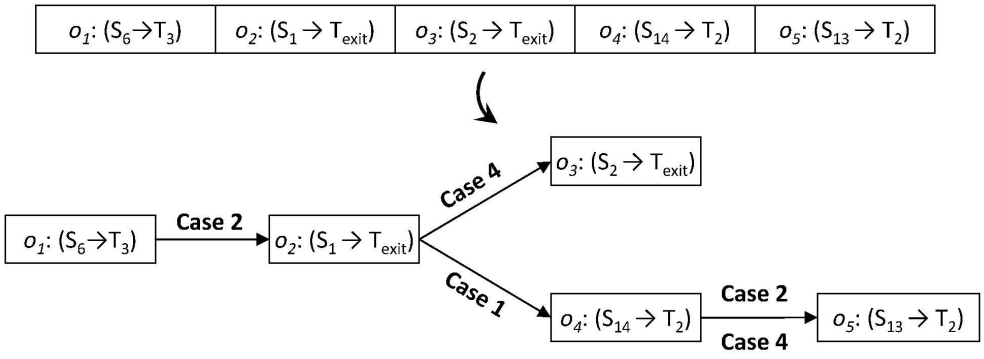

Figure 7
166x60mm (600 x 600 DPI)

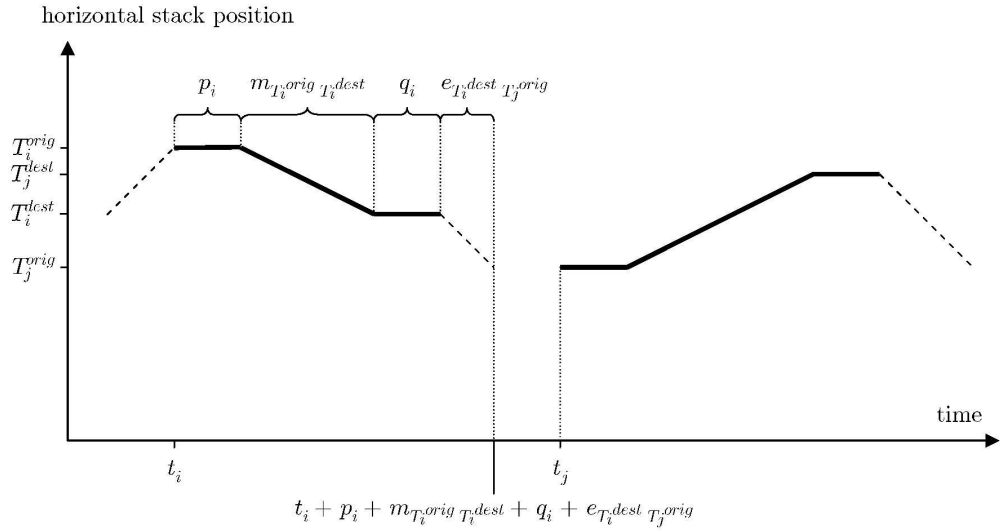

Figure 8
150x80mm (600 x 600 DPI)

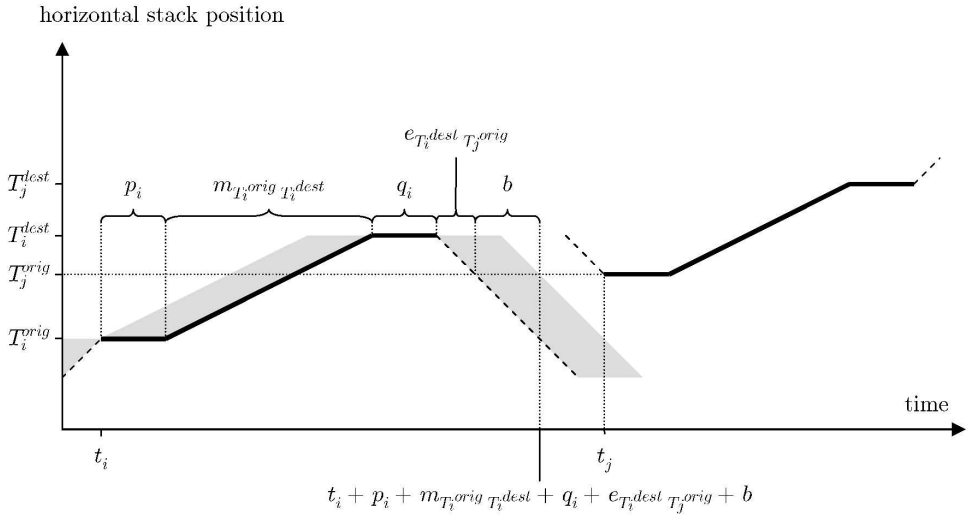

Figure 9
155x80mm (600 x 600 DPI)

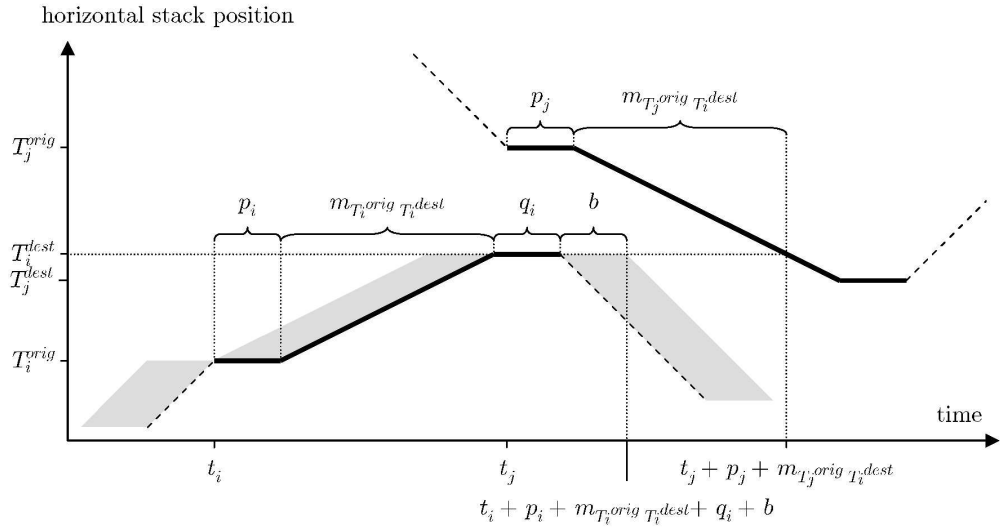

Figure 10
150x80mm (600 x 600 DPI)

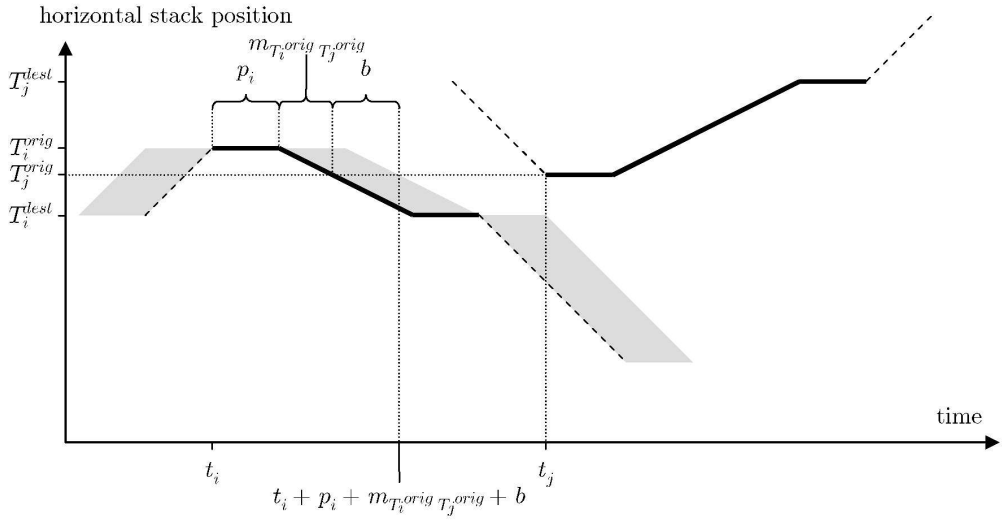

Figure 11
149x78mm (600 x 600 DPI)

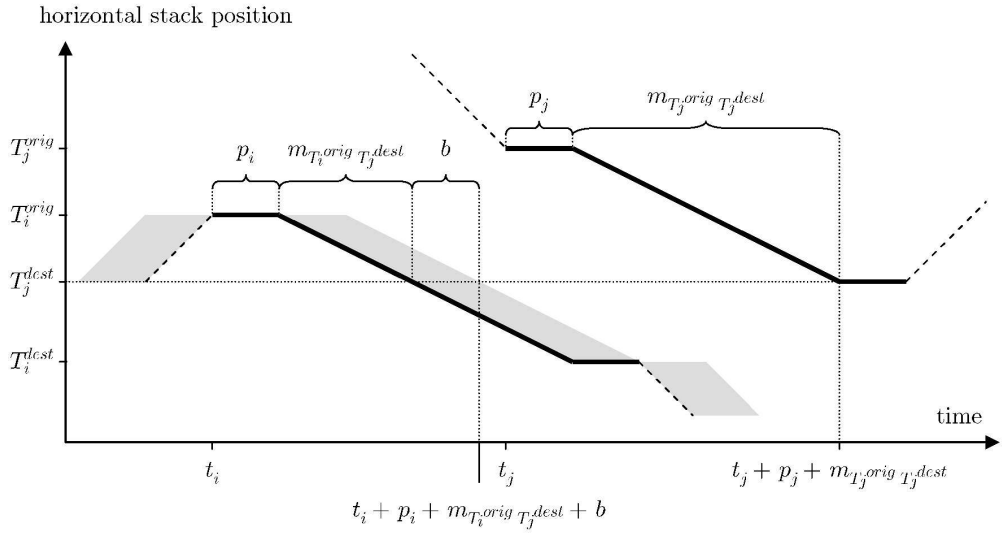

Figure 12
149x80mm (600 x 600 DPI)

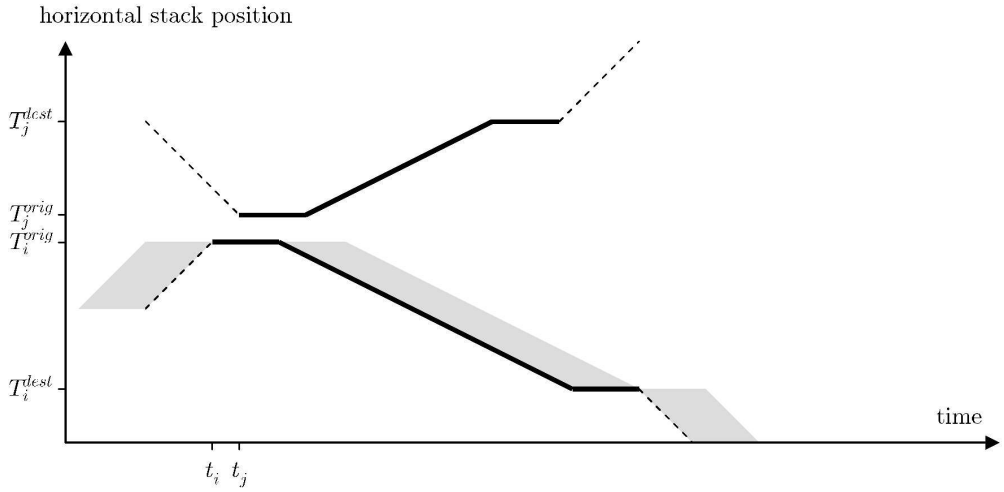

Figure 13
149x74mm (600 x 600 DPI)

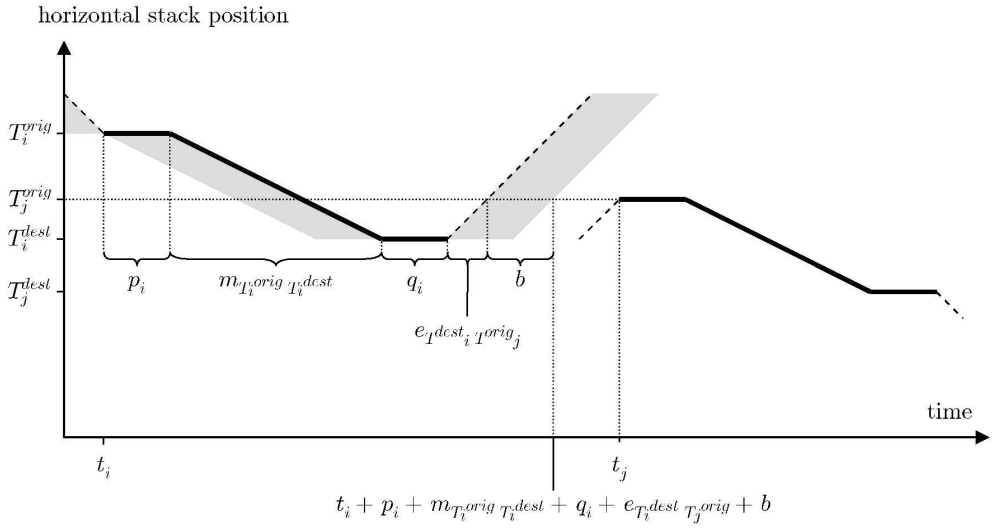

Figure 14
151x80mm (600 x 600 DPI)

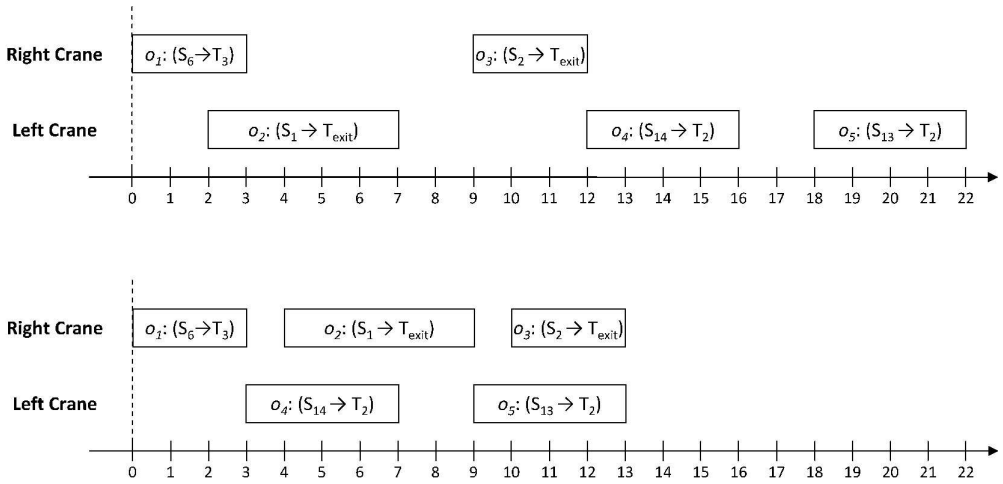

Figure 15
211x101mm (600 x 600 DPI)

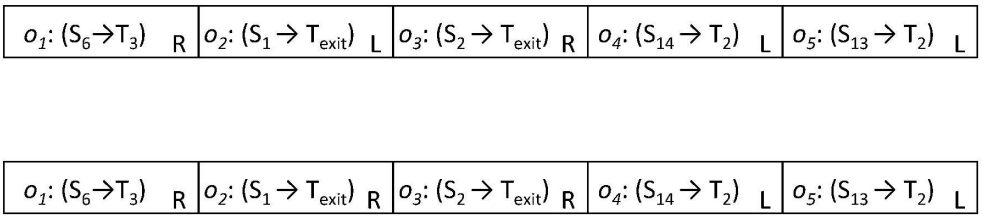

Figure 16
154x34mm (600 x 600 DPI)

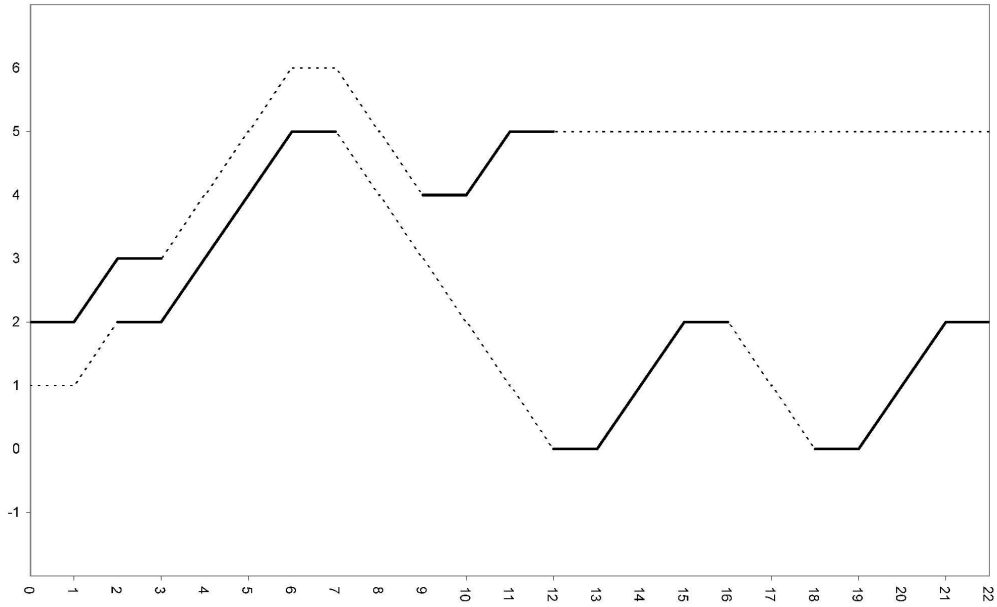

Figure 17a
251x160mm (600 x 600 DPI)

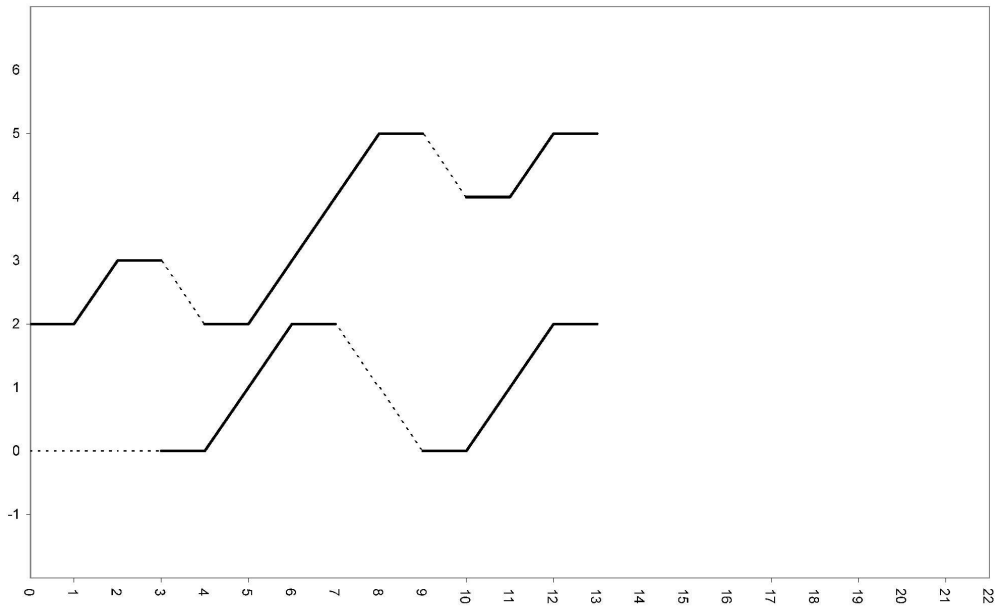

Figure 17b
251x160mm (600 x 600 DPI)

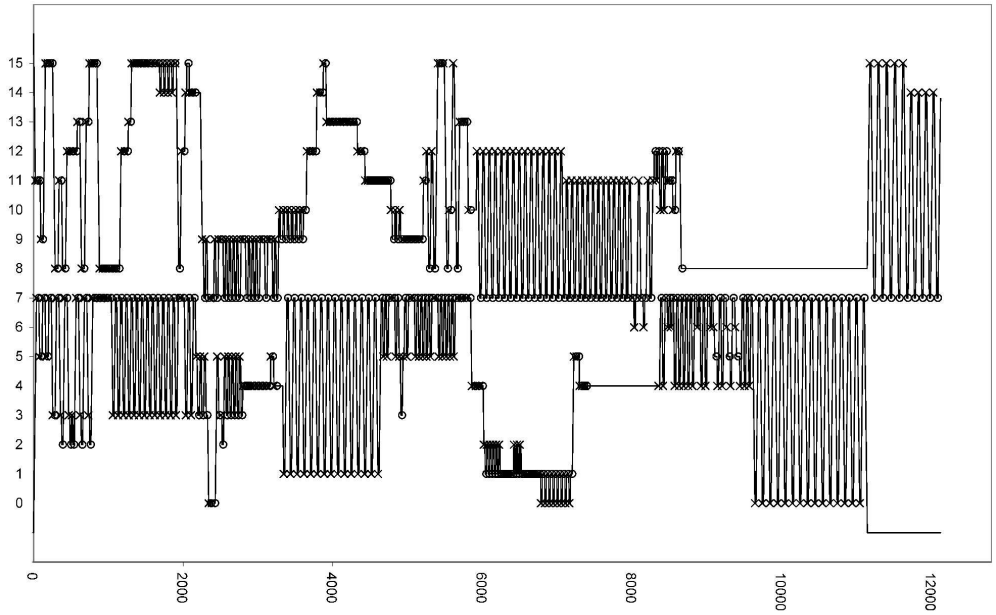

Figure 18
251x162mm (600 x 600 DPI)