Optimizing the Steel Plate Storage Yard Crane Scheduling Problem Using a Two Stage Planning/Scheduling Approach

Anders Dohn
Informatics and Mathematical Modelling, Technical University of Denmark
Richard Petersens Plads, 2800 Lyngby, Denmark
adh@imm.dtu.dk

This paper presents the Steel Plate Storage Yard Crane Scheduling Problem. The task is to generate a schedule for two gantry cranes sharing tracks. The schedule must comply with a number of constraints and at the same time be cost efficient. We propose some ideas for a two stage planning/scheduling solution approach to the problem.

Problem Description

The Steel Plate Storage Yard Crane Scheduling Problem is a difficult optimization problem, combining planning and scheduling in an effort to generate feasible schedules for a number of interacting cranes. The schedules must be applicable in real world settings and the objective is to minimize the overall cost. The cost is an aggregation of various costs directly related to crane operation and of other indirect yard operation costs. The schedules should at the same time introduce certain desired long-term planning features to the storage yard. Such features are introduced to improve the quality of future schedules.

The problem instances origin from real world data. Costs and constraints have been defined in close cooperation with the industry. The industrial problem instances are of a large size and hence it is important to create a solution method that can make superior heuristic choices in little time.

The problem at hand is from a large steel shipyard. Steel ships are constructed by welding together large steel plates. The plates arrive to the yard, where they are stored for a couple of weeks on average, before they are used in the production. The plates are so large and heavy that the only possible way of storing them is by placing them in stacks in a large storage yard. The plates are moved by electromagnets mounted on two gantry cranes. The gantry cranes move on the same set of tracks. As a consequence, the two cranes can never pass each other. The plates arrive by ship and are intermediately moved to arrival stacks by a tower crane. The movement of this crane is independent of the other cranes and not a part of the planning problem considered here. We consider the plates as a part of the optimization problem, when the arrival stacks of the day have been built.

For all plates, we have a due date, i.e. the time where this plate is to be moved to the exit belt. For all plates with due date equal to the day of planning, we further have an ordering in which they must be put on the exit belt. There is some uncertainty on the due dates of the plates. Sometimes the due date of a plate is changed, and we have to respect a new due date instead.

Figure 1 illustrates the structure of the yard. The problem is described in more detail in (Hansen 2003).

Modeling the Problem

The problem is modeled in two stages: a planning stage and a scheduling stage. In the planning stage we generate a plan that takes us from the current state of the yard to a final state of the day. In the final state, all plates with deadline on the current day are brought to the exit belt. At the same time, the plan should leave the yard in the best possible condition for the next day. The condition of the yard is assessed by a *static evaluator*, which is described briefly later. To arrive at a feasible and superior plan within reasonable computational time, our idea is to relax a number of the real world constraints in the planning stage. Whatever is relaxed here is fixed in the scheduling stage so that the final solution is always fully descriptive. In the planning stage, we are going to consider a solution as defined by a number of successive actions. An action is defined by the following:

- The **plate** that is moved by the action.
- The **destination** of the action, i.e. where the plate is moved.
- The **time** of the action.
- The **crane** to which the action is assigned.

The following relaxations of the definitions are possible:

- We may set the destination of an action to *undefined*.
- The time of the action may be left *undefined*. We may still want to describe precedence between some actions without specifying an exact time.
- Crane may be left *undefined*.

With the above relaxations, the idea is that we are able to explore a large number of diverse plans and thereby we are able to arrive at superior plans in the end. However, the plans are not completely deterministic, as we have a number of decisions which are still undefined. In the planning stage, an estimate will be used for the costs of the undefined choices.

A call to the scheduling routine is going to determine whether the plan has a feasible schedule and whether the schedule is actually as good as we have estimated.

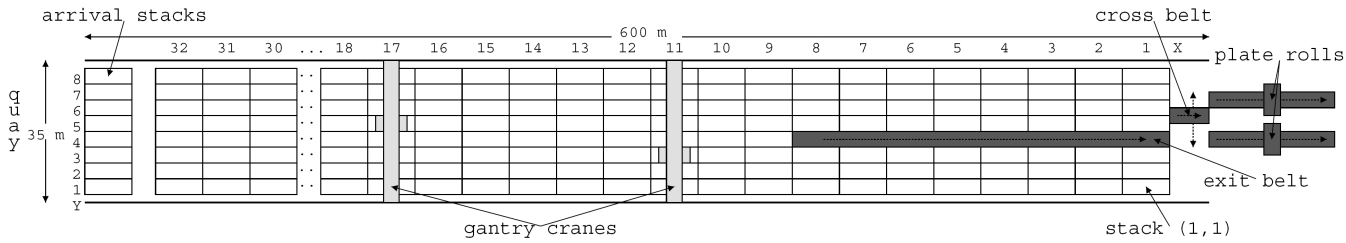


Figure 1: The storage yard as seen from above.

Solution Approach

The problem has been decomposed into two subproblems: a planning problem and a scheduling problem. In the following we assume that the destinations of all actions are specified in the planner. There may also be precedence constraints between actions. Some are directly implied by the problem instance, e.g. moving a plate that is not on the top of its stack, must naturally succeed the move of the top plate. Other precedence constraints may not be compulsory from a physical perspective, but may still provide the final state of the yard with desired features, and are therefore applied. Assignment of a crane to the actions and specification of the exact time is left for the scheduler to do.

As emphasized earlier, this is just one way of defining the split between planning and scheduling. However, this is a promising split, as it allows for a very flexible search in the planning algorithm. At the same time, estimating whether a feasible schedule exists, and what the cost of such a schedule will be, is easier than in a case, where more decisions are left undefined in the planning.

Figure 2 is a side-view of a toy example with only five stacks. Plates S_6 and S_{11} are due on the current day with S_6 leaving the yard before S_{11}. The exit belt is represented by stack 5. We know that any feasible plan now holds the actions: $(S_6 \rightarrow 5)$ and $(S_{11} \rightarrow 5)$. Furthermore, there will be some actions moving S_5 prior to the action $(S_6 \rightarrow 5)$.

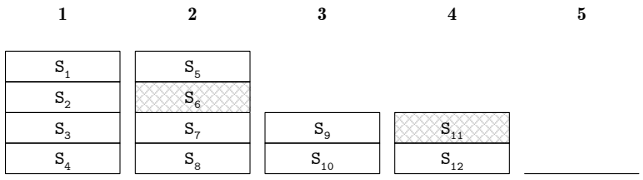


Figure 2: Side-view of a toy example.

The Planning Subproblem

The idea is to use well-known optimization techniques to solve the planning problem. As the size of the problem is huge, exact solution of a mathematical model is unlikely to yield good results.

Our initial idea is to apply a local search metaheuristic (e.g. Simulated Annealing) to the problem. This requires the definition of the solution space, and from that the definition of a neighborhood. Further, we also need to define the objective function.

Represent the solution of a planning problem as an ordered list of actions. With the earlier definition of an action, the final state of the yard is fully determined by the planning solution. From this representation of a solution, a neighborhood can be defined as follows:

- **Remove an action** from the plan.
- **Add a new action**, i.e. choose a new plate to be moved and specify where to move it.
- **Change destination** of an existing action.
- **Reorder actions**, i.e. give an existing action a new position in the ordering.

When moving to neighbor solutions, we need to check that the new solution is feasible. All actions must be feasible with respect to their position in the ordering given by the solution.

To illustrate what a solution may look like, assume that we have the following solution to the planning problem of Figure 2:

(S_5→1)	(S_6→5)	(S_5→3)	(S_{11}→5)	(S_1→2)

The solution yields the final state of the storage yard illustrated on Figure 3.

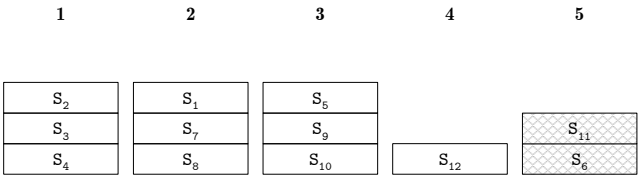


Figure 3: The final state of the storage yard for the toy example of Figure 2.

Another solution leading to the same final state can obviously be reached by two neighborhood transitions, e.g.:

1. **Remove** action $(S_5 \rightarrow 3)$.
2. **Change destination** of action $(S_5 \rightarrow 1)$ to stack 3.

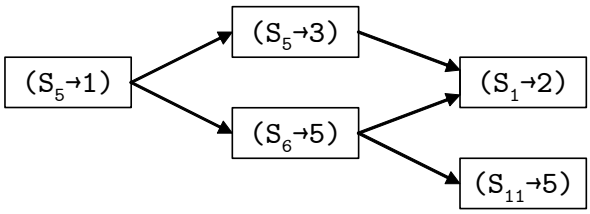
The new solution is:

(S_5→3)	(S_6→5)	(S_{11}→5)	(S_1→2)

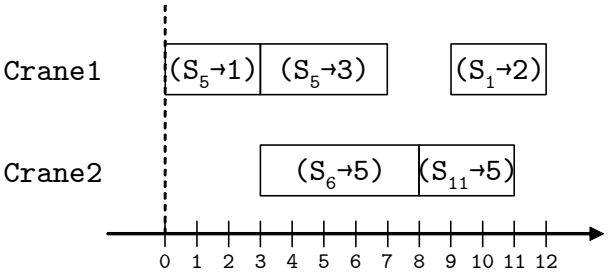
The Scheduling Subproblem

From a solution to the planning problem it is now the task to generate a complete and feasible schedule. First, the ordering of tasks must be relaxed to allow for parallel execution of actions. Most actions are locally independent from each other. These independencies are detected and only meaningful precedence constraints are kept for the scheduler. The problem is now similar to a traditional scheduling problem. We have a number of actions (operations) that we need to carry out on 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. Since the crane operation times are of a stochastic nature, we need to introduce buffers, enforced by temporal constraints. 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.

For the example of Figure 2, let us assume that the first of the two solutions are found by the planning solver. Now, we have to schedule the actions. For the actions we have the following precedence constraints:



We may hence, if no other constraints are violated, be able to schedule actions ($S_5 \rightarrow 3$) and ($S_1 \rightarrow 2$) on one crane and in parallel actions ($S_6 \rightarrow 5$) and ($S_{11} \rightarrow 5$) on the other crane. In a very simple world, where moving from one stack to its neighbor takes 1 time unit and lifting or dropping a plate takes 1 time unit and when these times are not stochastic, the following schedule is feasible:



Including all precedence constraints from the planning solution may restrict the scheduler too much. If initial testing shows this to be the case, we could allow the scheduler to slightly rearrange actions in order to arrive at better schedules.

We plan to approach the scheduling problem with standard solution methods. A promising possibility is to use Constraint Programming to solve the scheduling problem. We may also try Integer Programming models or perhaps tailored heuristics.

Evaluation of Solutions

It still remains to define the direct and indirect costs of the problem. Moreover, these have to be transformed into estimates for partly undefined actions and for the static evaluation. The yard has a fixed daily throughput and hence it becomes our task to comply with throughput constraints and at the same time minimize costs.

The costs that apply include:

- Salary to the crane operators.
- Electricity costs of crane movement and magnet activation.
- Crane maintenance.

These costs can be mapped to the make-span of the schedule, the total distance traveled by the cranes, and the total number of lifts/drops. The salary is adjustable only in large steps (i.e. if we can go from 2 working shifts to 1.5 shifts, some of the crane operators may be employed part time only, or they may get reassigned to other tasks for half of their shift). According to the management of the yard, the main part of the cost comes from maintenance of the cranes.

The make-span of a schedule is not directly inferable from the planning solution. Therefore, it has to be estimated at this point. We do know that it is closely related to the total execution time of all actions. Total travel distance and number of lift/drops are apparent in the planning solution.

The overall objective is to minimize the cost of the resulting schedules. However, the cost minimized is the sum of the cost of the day of planning plus all future costs (to some extend). Hence, the optimal solution is not necessarily a solution that minimizes the cost of the current day; it also has to leave the storage in an attractive state. The static evaluator is used to measure attractiveness of a state by the following measures:

- How well are stacks sorted?
- How close are almost due plates to the exit-belt?
- How close are the stack heights to a uniform distribution?

What is Next?

The next step in the development is to implement the described methods and to conduct some initial experiments on problems of a reasonable size. It may still be too hard to do experiments on real size data, but we need to assess the approach described. These tests are going to reveal whether major changes have to be imposed. The advantages and possible disadvantages of the decided split between planning and scheduling module will become more transparent and we will modify the model accordingly.

References

Hansen, J. 2003. *Industrialised application of combinatorial optimization*. Ph.D. Dissertation, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby.