Crane Scheduling on a Plate Storage

Center for Industrial Application of Mathematical Models

Jesper Hansen[1]

1. April 2002

Introduction

Center for Industrial Application of Mathematical Models (CIAMM) is a virtual center consisting of several industrial companies and universities. The purpose of the center is to "industrialize" the research done at the universities within mathematical modeling.

Our work in the project is to extend models and solutions for prototype problems to real world problems. From a few cases we want to generalize these approaches and methods in order to construct a "cookbook" for developing optimization software for the industry.

For the remainder of this paper, we will consider a case from Odense Steel Shipyard (OSS).

The Plate Storage Problem

OSS produces the worlds largest container ships. The first process of producing the steel ships is handling arrival and storage of steel plates until they are needed in production.

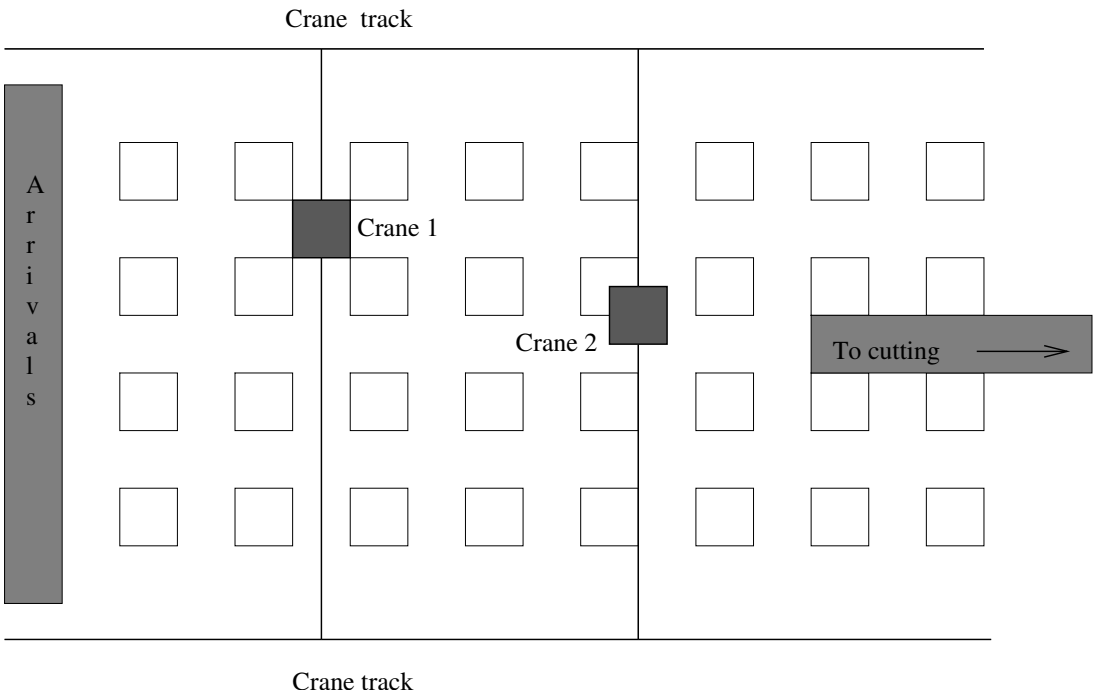
The plate storage is organized in 8 times 32 stacks. Approximately 20 plates are stored in each stack on average. One quarter of the stacks is used for special purpose plates for instance stacks with identical plates. The rest of the storage is used for plates, which for the main part have different sizes. Each plate is ordered for a specific purpose and it is known at which date it is needed in production. Changes in the overall plan for the yard, can however influence the due dates of the plates on the storage, which means that the due date of a plate can change several times before needed in production.

Two gantry cranes carry out the movements of plates. The cranes share tracks and hence can not pass each other. For each of the 32 rows the cranes can reach each of the 8 stacks. The cranes are equipped with an electromagnet which can lift only the top plate from a stack and move it to some other stack or remove it from the storage. Since the cranes are gantry cranes there is a limit on the height of the stacks.

Each day a set of plates must leave the storage to be cut up. The plates are put on a conveyer belt with a capacity of 8 plates, which is called the exit-belt. The exit-belt works as a queue and plates are drawn from the queue with a time interval depending on the order and dimensions of the plates on the belt.

If a specific plate is requested, which is not on the top of a stack, the plates on top must be moved to other stacks before the requested plate can be removed from the storage. This procedure introduces a number of *unproductive* movements, which we call *dig-up movements*.

[1] Informatics and Mathematical Modelling, Technical University of Denmark, 2800 Lyngby, email:jha@imm.dtu.dk



The planning task is now to create a schedule of movements for the 2 cranes without collision, that delivers the plates needed for the given day and minimizes costs.

Many different classes of problems within scheduling and routing have similarities with the problem considered in this paper. One problem is the Traveling Salesman Problem (TSP) where a salesman must visit a number of customers or rather nodes in a graph. For the Stacker-Crane Problem (SCP) a crane must move items from a node to another node in other words visit a subset of arcs in a graph. In our problem the items are steel plates.

If k plates are to be moved from a stack s to other stacks it means that k arcs going out of s must be visited. The order in which they are visited must be such that the arc corresponding to the top plate are visited before the arc corresponding to the next plate and so on. I.e. we have precedence relations between the arcs leaving s.

In our application a plate may be moved more than once if it is put on a stack from which plates are going to be moved later on. This means that extra arcs are dynamically inserted in the graph. Insertions of extra arcs are depending on the order in which the arcs are traversed and we have precedence relations between arcs corresponding to movements of the same plate. We could restrict ourselves to solutions where all plates to leave a stack a given day must be removed before plates can be put on the stack. This again introduces precedence constraints, but could result in problems with no feasible solutions.

In the SCP the destination is given for a movement, but for our application the destinations for the dig-up movements are decision variables. In the graph setting it means that the head of an arc out of node s corresponding to a dig-up movement from stack s is not given, but is part of the optimization problem.

In our application the two gantry cranes share tracks and in our schedule we must avoid collisions of the two cranes. Extra *positioning* and *wait movements* may have to be inserted in the sequence of movements in order to avoid this.

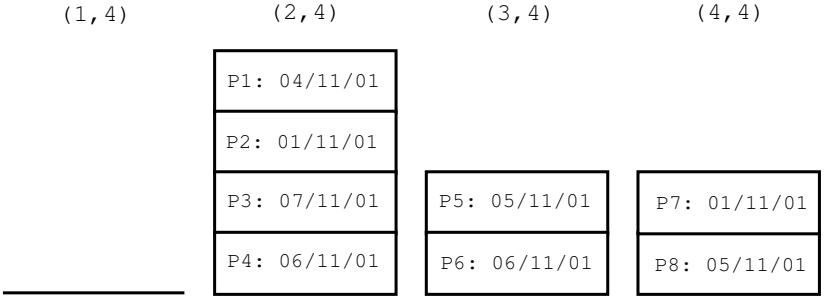
Objective Function

Costs include salary to the crane operators, power for the cranes to move and activation of the electromagnetics as well as maintenance on the cranes. In other words, make-span of the schedule, total traveled distance and number of movements. Minimization of the time when the last plate is put on the exit-belt is also important to be able to start the following processes as early as possible. One day of production is planned at a time, so in order to minimize the long-term costs; we measure the quality of the *state* of the storage after each day. The state is evaluated by 3 different criteria:

1. How well are the stacks sorted by plate due date? More specifically, how many dig-ups have to be done for each stack with the given plate due dates. For stack (2,4) in the figure below plate p_2 is the first to be removed from the storage on the November 1. This requires the dig-up of p_1. The next plate to be removed from the stack is p_4, which causes the dig-up of p_3. The result is 2 dig-ups. We have defined the cost of a dig-up to be the sum of the following terms:
   a. Estimated cost of power used for lifting and dropping a plate.
   b. The cost of power for moving the crane to the nearest stack and back.
   c. The cost in salary for operating the crane in the amount of time according to the dig-up movement.

2. How close are the plates to the exit-belt compared to the due date of the plates? When minimizing traveling distance it is better always to move plates in direction of the exit-belt and when the due date of a plate is close in time, we want it to be close to the exit-belt. More formally the cost for a given plate is: (*distance to exit-belt*)/(diff(*plate due date*, *today*) + 1) where diff(date1, date2) returns the difference in days between two dates. Let us assume that the exit-belt is located at the coordinates (1,4), then stack (2,4) is 1 closer than (3,4). If for instance today is 01/11/2001 then the cost for plate p_1 is 1/4, for p_4: 1/6 and p_6 it is 1/3. The above exit distance can be converted to time and the cost is then achieved by multiplying with the salary of the operators.

3. How many plates are there in each stack? We want the plates to be as evenly spread out on the stacks as possible, which will result in less dig-ups when changes in due dates occur. Generally the cost for a stack is: $(\textit{no. of plates in stack})^2/(\textit{max. no. of plates in stack})$, where the maximum no. Of plates in a stack is a user supplied upper limit on the number of plates common for all stacks. We observe that the cost per stack will be in the interval from 0 to the maximum number of plates in the stacks. If we have an upper limit of 10 plates, then the stacks in the figure below will have a cost of (16+4+4)/10=2.4 while if p_1 was moved to stack (3,4), the cost would be (9+9+4)/10=2.2, and hence better. What we are minimizing is in some sense the worst case number of dig-ups if a plate in the bottom of a stack is requested. The cost of a dig-up in 1. is therefore used here as well.



We have managed to convert all criteria functions into a common unit, which is then minimized. All the user has to supply are the operating costs of the crane and the salary of the operators.

Solution Procedure

Each day we know which plates must leave the storage. These movements of plates out of the storage are called *exit movements*. Given these movements we can identify plates that have to be dug up during that day. Note that these plates can be identified before scheduling any movements.

When plates arrive by ship they are registered in a database and the system must handle the scheduling of picking up the plates from the quay and where to put it in the storage online.

In order to improve the state of the storage, we may perform some sorting of the stacks. We call these movements for *sort movements*. The reason for doing sorting is that in the future less dig-ups will be necessary, and less time will be spent removing plates from the storage. If the due dates of the plates change frequently there is a risk of doing a lot of sorting that later turn out to be wasted. Experiments show that sorting such that plates to leave the storage the next day are on top of the stacks is an appropriate amount of sorting, since no dig-ups will be necessary the day after if the due dates are not changed.

Until now we have implemented a construction heuristic and local search heuristics for improving the initial plan. The times for lifting and dropping plates, moving the crane and speed of the exit-belt are not constant, but for the purpose of planning they are considered constant. During execution of the plan changes in times may cause the plan to be infeasible. In that case the plan must be revised. One approach, which is under development, is to adjust the plan to the changes and the other approach is to use the construction heuristic as an on-line algorithm.

It has shown to be difficult to construct neighbourhood structures where local changes can easily be propagated to global changes in the objective-value. Therefore the first attempt was to evaluate the change in objective by ”simply’’ simulating the crane movements from start to end for every new neighbour solution. This is off course computationally expensive and we are now trying to estimate the change in costs instead.

The solution achieved by the construction heuristic mentioned earlier can be seen as a path from the root to a leave in a search tree, where the branches correspond to choices on which move to perform. Until now we have only found one solution in the search tree, but we consider introducing a search mechanism to further explore the tree. This could turn out to be a better approach than local search, since it does not depend on estimation of costs. Because of the combinatorial explosion it will still need to be an incomplete search of the solution space.