Steel stacking

A problem in inventory management in the steel industry

João Pedro Pedroso

International Symposium on Mathematics of Logistics
TUMSAT, Japan, November 2011

Part of the presentation concerns
joint work with **Rui Rei** and **Mikio Kubo**.

Contents

Contents

Informal problem description

Context

A steel producer has a warehouse where the final product is stocked

- large steel bars enter the warehouse when production finishes
- bars leave the warehouse on trucks or ships for transporting them to the final customer
- there is a crane in the warehouse, which moves the bars one at a time
- the warehouse has p different places
- each place can be empty, or keep a stack of steel bars

Informal problem description

Assumptions

- capacity of the stacks is infinite
- no delays on crane movements
- crane can move only one item at a time
- only the item on the top of the stack can be moved
- item on top of each stack may have to be relocated (*reshuffling*)

Objective

- minimize the number of movements made by the crane

Steel stacking

Steel stacking

Data

$$
\begin{cases}
p \in \mathbb{N} & \text{number of stacks on the warehouse} \\
n \in \mathbb{N} & \text{number of items} \\
R_i \in \mathbb{N}, i = 1, \dots, n & \text{release dates} \\
D_i \in \mathbb{N}, i = 1, \dots, n & \text{delivery dates}
\end{cases}
$$

Steel stacking

Constraints

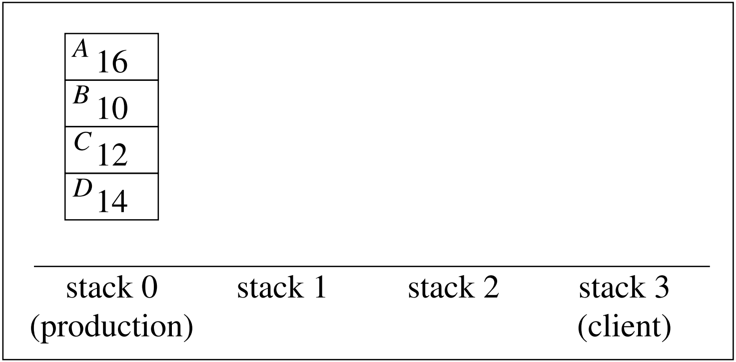
- crane can move only the item on top of the stack
- release and delivery dates must be satisfied
- the valid movements depend on R, D, and on the choices made up to the moment.

Solution representation

List of movements from a stack (o) to another (d)
$M = [(o_1, d_1), \ldots, (o_k, d_k)]$

- $0 \leq o_i \leq p$ and $1 \leq d_i \leq p + 1\}$
- stack 0 represents the production facility
- stack $p + 1$ represents the customer track/ship.
- we want to minimize the number of movements (size of M)

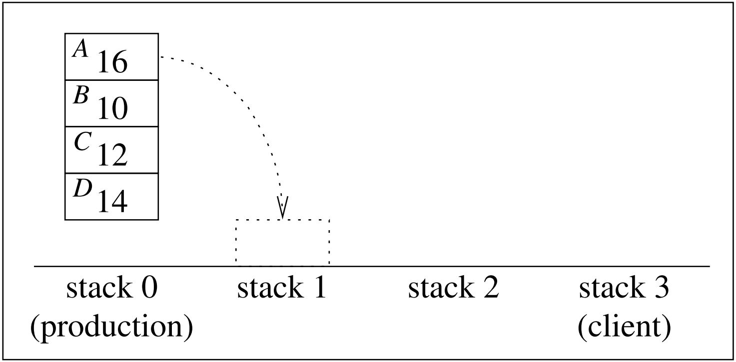
Example



Movements:

[]

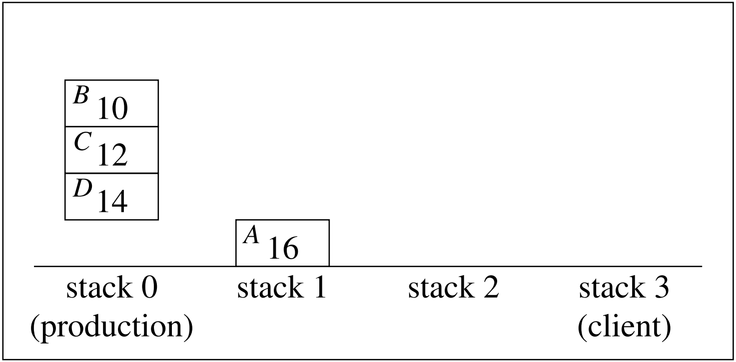
Example – step 1



Movements:

[]

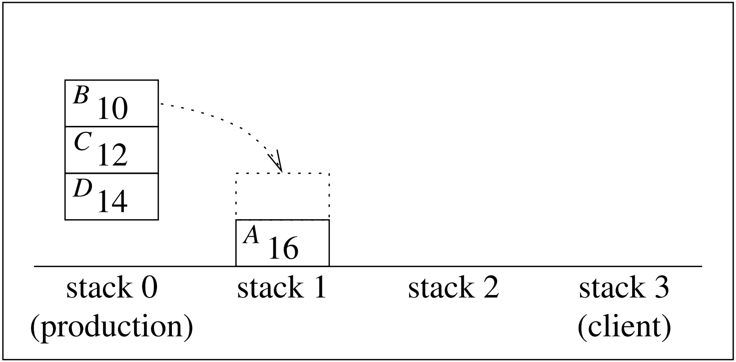
Example – step 1



Movements:

$[(0, 1)]$

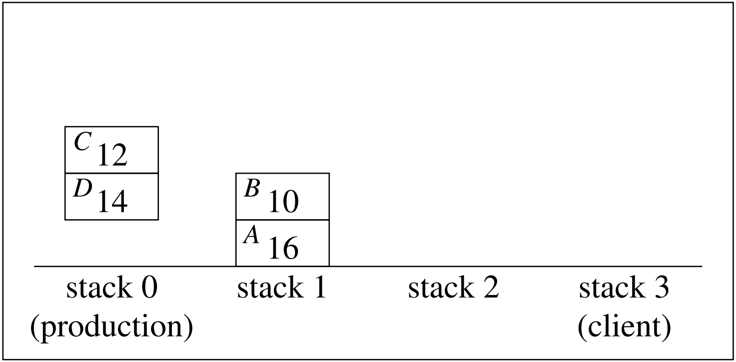
Example – step 2



Movements:

$[(0, 1)]$

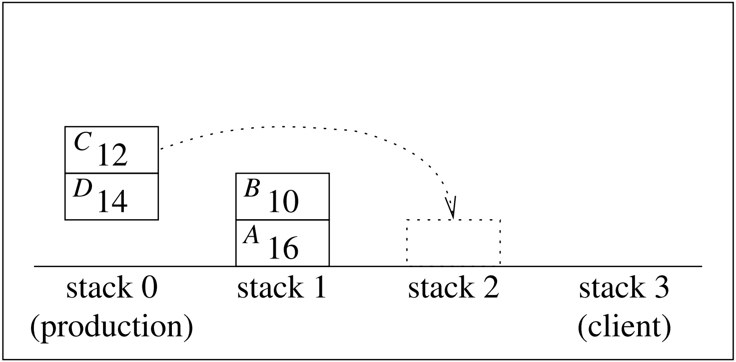
Example – step 2



Movements:

$[(0,1),(0,1)]$

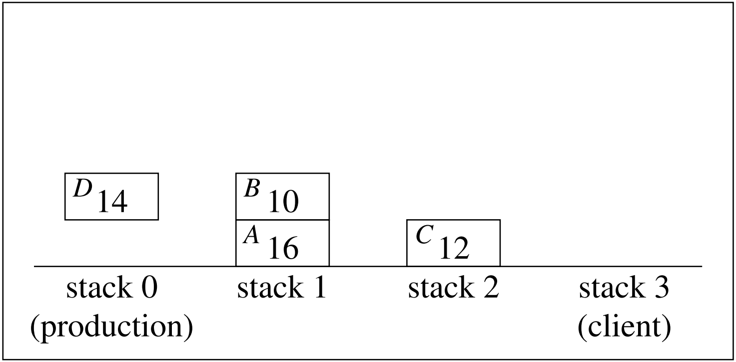
Example – step 3



Movements:

$[(0,1),(0,1)]$

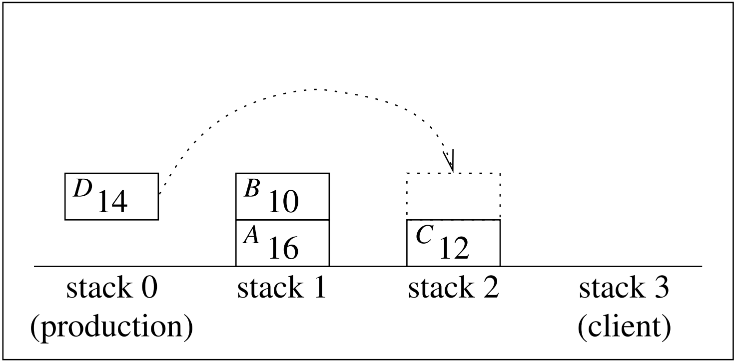
Example – step 3



Movements:

$[(0,1),(0,1),(0,2)]$

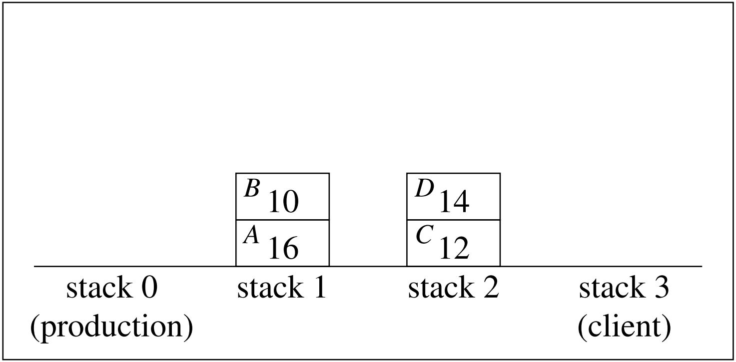
Example – step 4



Movements:

$[(0,1),(0,1),(0,2)]$

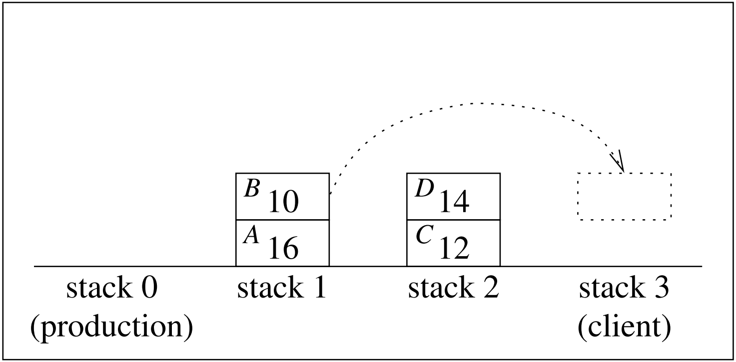
Example – step 4



Movements:

$[(0,1),(0,1),(0,2),(0,2)]$

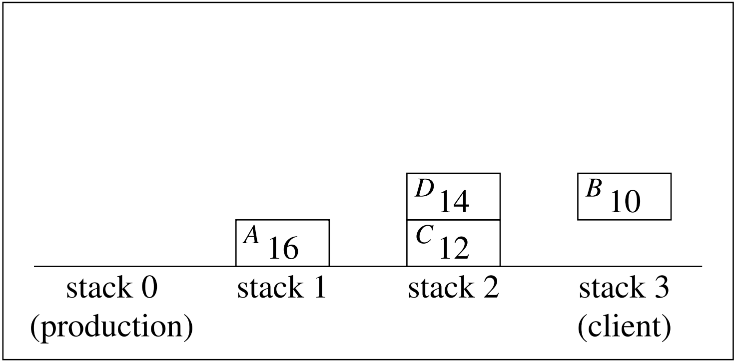
Example – step 5



Movements:

$[(0,1),(0,1),(0,2),(0,2)]$

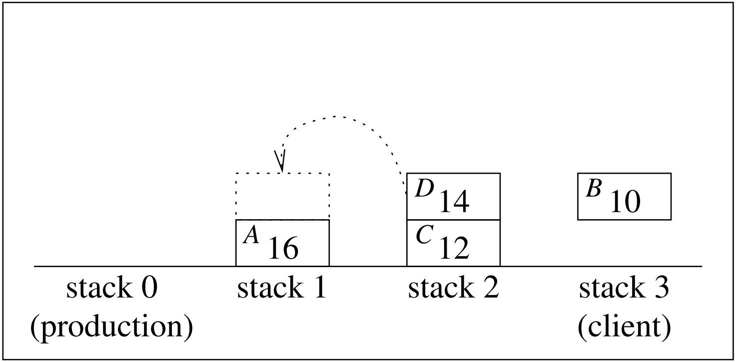
Example – step 5



Movements:

$[(0,1),(0,1),(0,2),(0,2),(1,3)]$

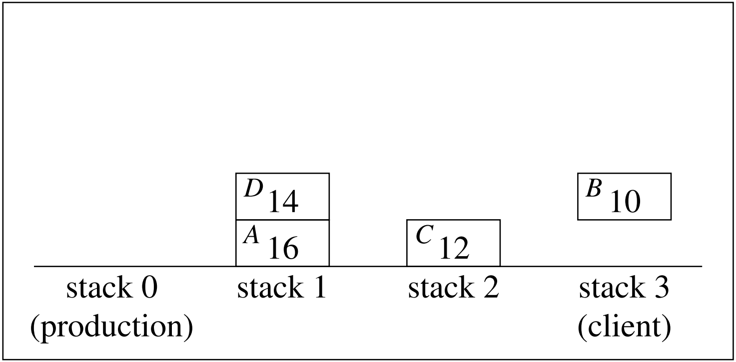
Example – step 6



Movements:

$[(0,1),(0,1),(0,2),(0,2),(1,3)]$

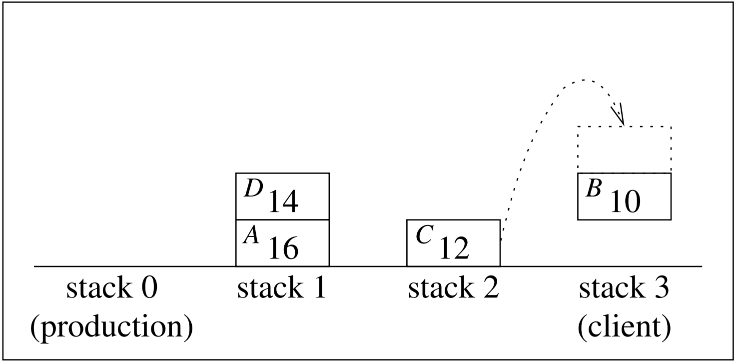
Example – step 6



Movements:

$[(0,1),(0,1),(0,2),(0,2),(1,3),(2,1)]$

Example – step 7



Movements:

$[(0,1),(0,1),(0,2),(0,2),(1,3),(2,1),\rightarrow(2,3),(1,3),(1,3)]$

This information is complemented with release and due dates.

Solution representation: MIP

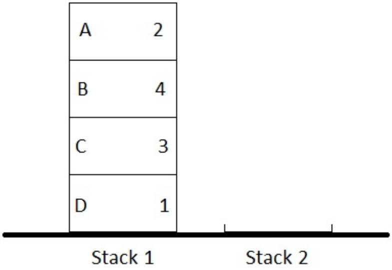
If we want to solve the problem with standard optimization tools: **MIP formulation**:

Sets

- $T \in \mathbb{N}$ – time horizon (the number of periods in the model)
- $N \in \mathbb{N}$ – number of items
- $W \in \mathbb{N}$ – the number of stacks in the warehouse (warehouse width).
- $H \in \mathbb{N}$ – the maximum number of items that can be in a stack at any given instant (warehouse height).
- $R \in \mathbb{R}^N$ – item release dates (R_i denotes the release date of item i).
- $D \in \mathbb{R}^N$ – item due dates (D_i denotes the due date of item i).

Solution representation: MIP

Problem: number of **periods** that have to be considered



Worst case: $T = 2N + \sum_{n=1}^{N-1} n$

Solution representation: MIP

MIP formulation

Variables

x_{ijnt} – 1 if item n is released into position (i, j) at period t

y_{ijklnt} – 1 if item n is relocated from position (i, j) into (k, l) at period t

z_{ijnt} – 1 if item n is delivered from position (i, j) at period t

a_{nt} – 1 if item n has not entered the warehouse yet at period t

b_{ijnt} – 1 if item n is in row j of stack i at period t

c_{nt} – 1 if item n has already left the warehouse at period t

Solution representation: branch-and-bound

Branch-and-bound

- When an item is released/relocated:
  - Check all stacks where it can be placed
  - Create a branch for each of them
- When an item is delivered from top: move without branching.

Contents

MIP

- formulate the problem, create model
- read an instance
- send it to a solver

Branch-and-bound method

```
create empty queue Q
push root node into Q
while Q is not empty
    μ ← pop a node from Q
    i ← item to be placed next on node μ
    foreach stack s where i can be placed
        μ′ ← a copy of node μ
        place item i in stack s on node μ′
        execute deliveries from top of stacks in μ′
        if μ′ is a leaf node
            check if μ′ contains a better solution
        push node μ′ into Q
```

Discrete event simulation

- type of simulation used on systems where the state variations are discrete
- computationally "inexpensive"

In our case:

- each simulation run involves some randomness: stack for each item is selected randomly from list of candidates
- different runs lead to different solutions
- at the end, choose the best run (a large number of runs may be required for obtaining good results)

- several stacking strategies (heuristics) can be used and tested

Simulation strategy:

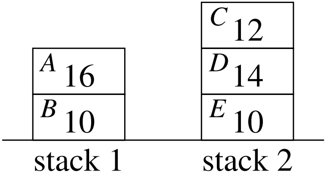
Heuristics for choosing a stack for an item are applied

1. on arrival of an item
2. when moving an item from the top, to reach another below it (*reshuffling*).

Processing order

- delivery has precedence over stacking released items
- "simultaneous" releases are processed by inverse delivery date
- "simultaneous" deliveries are processed from top to bottom of the stack

"Simultaneous" deliveries:



No specified order on simultaneous deliveries:

1. $\{C, D\} : s_2 \to s_1$
2. $E : s_2 \to$ client
3. $\{D, C, A\} : s_1 \to s_2$
4. $B : s_1 \to$ client

$\hookrightarrow$ *7 movements*

Deliveries processed by item depth:

1. $A : s_1 \to s_2$
2. $B : s_1 \to$ client
3. $\{A, C, D\} : s_2 \to s_1$
4. $E : s_2 \to$ client

$\hookrightarrow$ *6 movements*

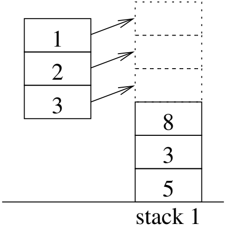
Positioning heuristics

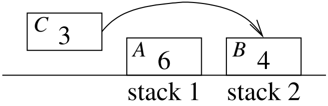
When placing an item

- each position (i.e., each stack) is assigned a value, according to some rules
- construct a **candidate list** of positions with best classification (depends on the heuristics used)
- from this list, randomly choose one for placing the item

Heuristics: Optimize flexibility

- we define **flexibility** of a position as the *maximum number of items* with different delivery dates that can be stacked, *without causing an inversion*
- candidate stacks for an item are those which:
  - maximize flexibility and cause no inversion
  - minimize flexibility, if an inversion is unavoidable





Simulation-based optimization

- For this stacking heuristics:
  - do N independent simulation runs
  - choose the run that lead to less crane movements
- For each of the N runs:
  - for each item, choose a stack according to the selected heuristics
  - continue processing item releases and deliveries, until having all items delivered
  - return the number of movements required, and the corresponding movement list

Contents

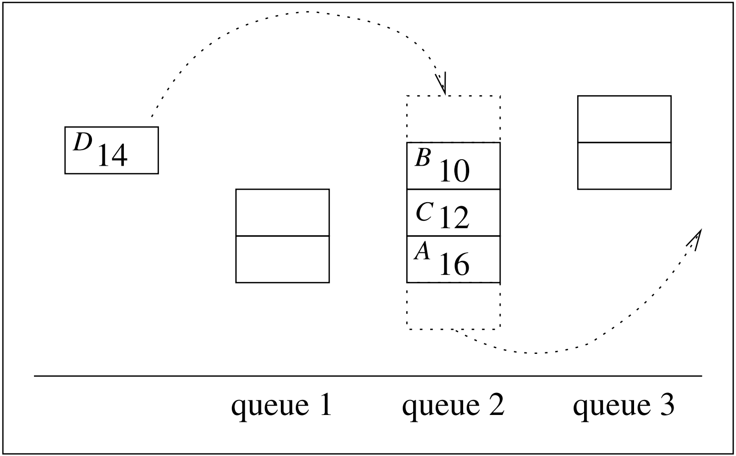
Problem variants

There are many variants of the problem with practical interest:

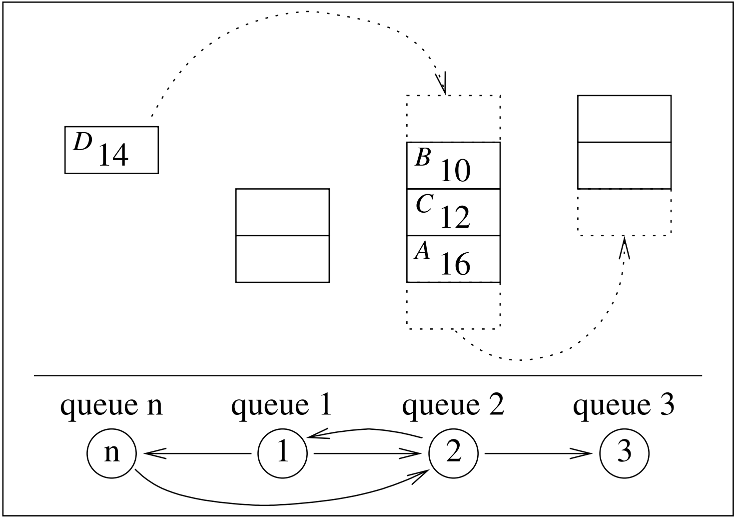
1. stacks are replaced by more general structures, such as double-ended queues; this allows modeling *e.g.* cases where items are placed in a horizontal configuration, with access from two ends;
2. crane movements are limited, and thus the number of stacks that can be reached from a given position is limited;
3. placement in certain stacks limits crane movements, *i.e.*, the graph of connections is dynamically changed when items are placed;
4. crane movements induce significant delays, making the assumption of immediate delivery unviable;
5. other objectives.

More general structures

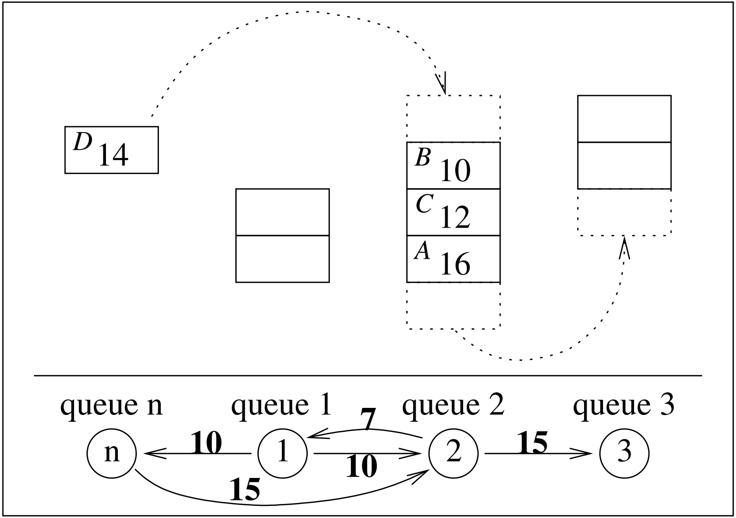
E.g., double ended queues



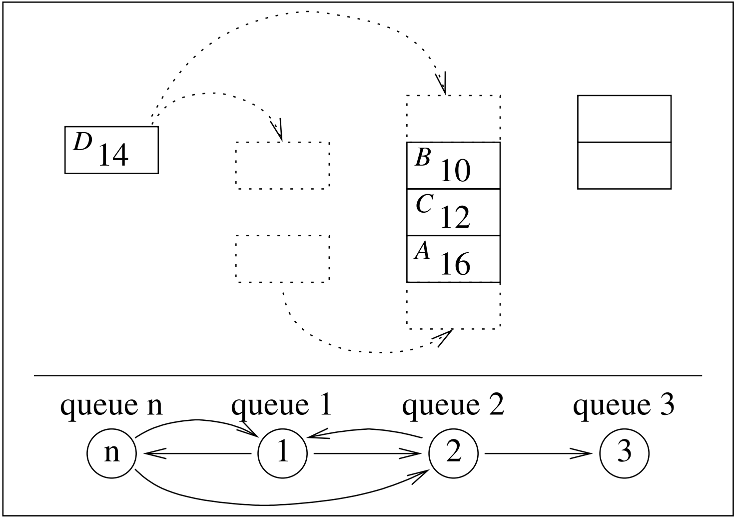
Limited movements



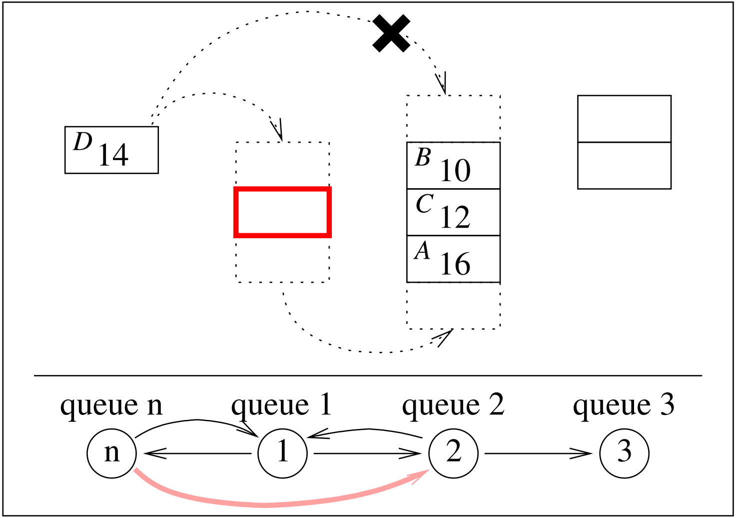
Dynamics



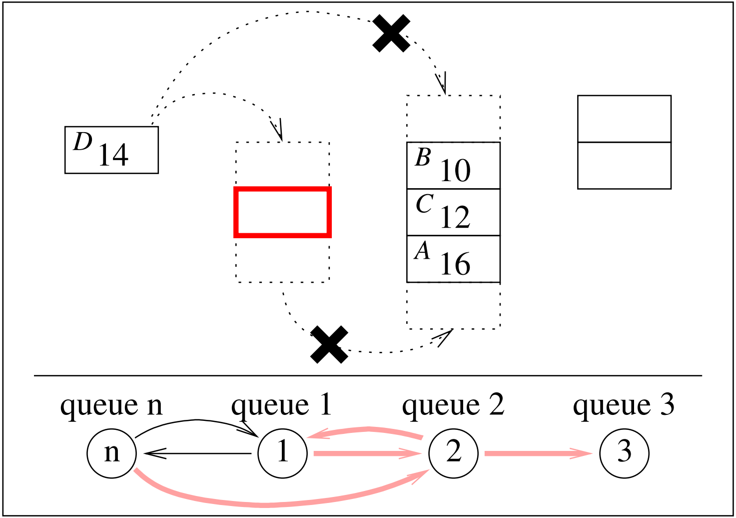
Dynamics: allowed movements change 1



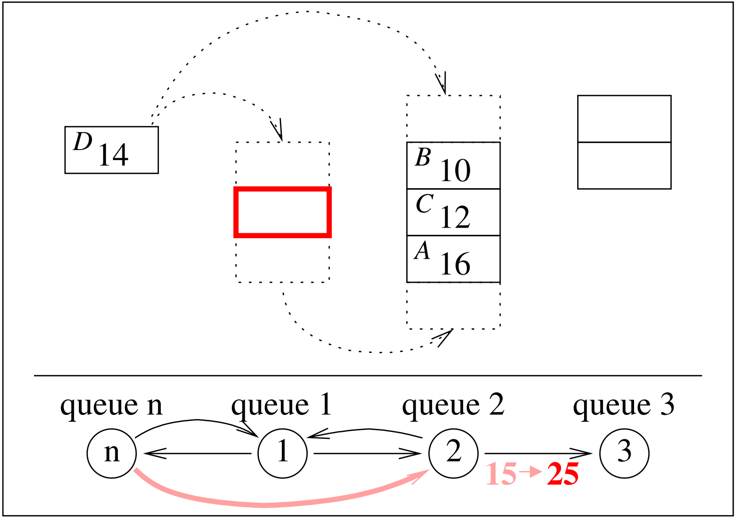
Dynamics: allowed movements change 2



Dynamics: allowed movements change 3



Dynamics: times change



Other objectives

Another interesting variant:

1. minimize time of service of track/ship;
2. items are rearranged before track of ship arrive;
3. aim: having the minimum number of movements for loading the track/ship;
4. in some variants: more than one item may be moved a the same time.

Contents

Conclusions

- Problem tackled: difficult (decision in one step may have consequences much later).
- For tackling it:
    - MIP model;
    - Specialized branch-and-bound;
    - Simulation-based optimization.
- Problem has many interesting variants.
- Dynamics: configuration may change upon decisions taken.
- Modeling issues: language for formalizing problem description.
- New solution approaches: finite state automata.