Order Sequencing in the Automobile Industry

A Rule Based Approach with Color Change Reduction

Thomas Epping
Ifb AG
Cologne, Germany

Peter Oertel
Ford Motor Company
Cologne, Germany

Robert Nickel, Winfried Hochstättler
Mathematical Foundations of Computer Science
Institute of Mathematics
Brandenburg University of Technology at Cottbus, Germany

Outline

- A Framework for Order Sequencing
- Rules for Order Sequencing
- A Greedy Approach for Sequence Construction
- Order Clustering to Reduce Color Changes

A Typical Production Plant

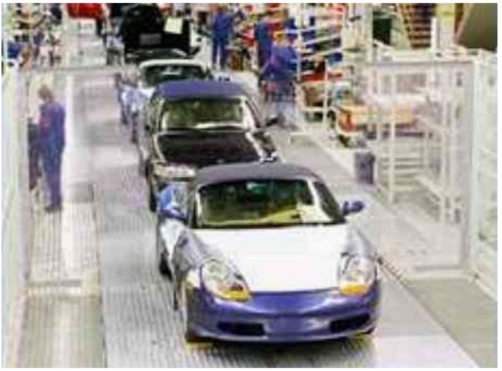
Body Shop

- press, weld, and mount the car body

Paint Shop

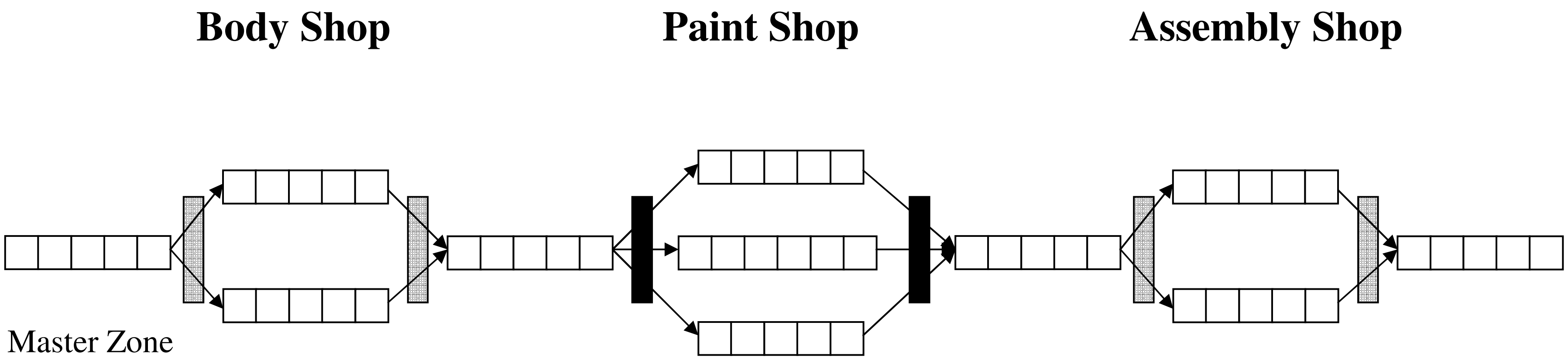
- enamel the body-in-white

Assembly Shop



- implement optional components

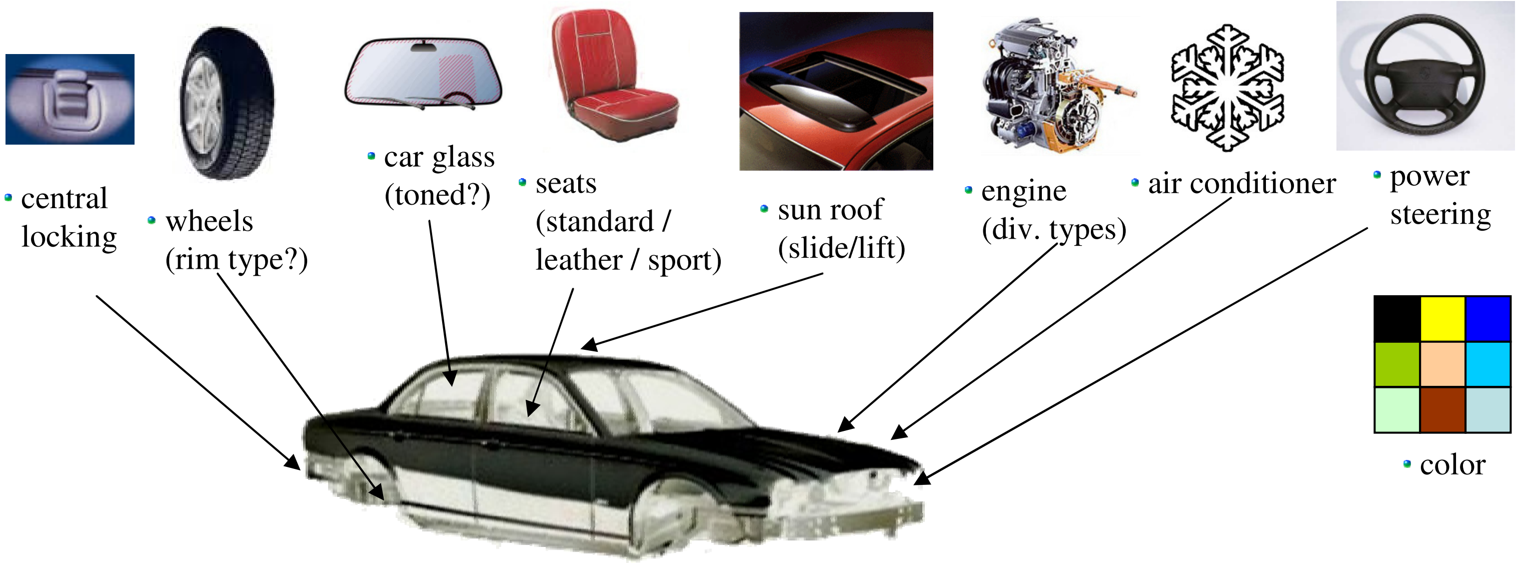
A Typical Production Plant



- For all zones a set of rules must be respected
- Storage systems allow color change reduction by short term interchanges prior to the paint shop

Orders and Commodities

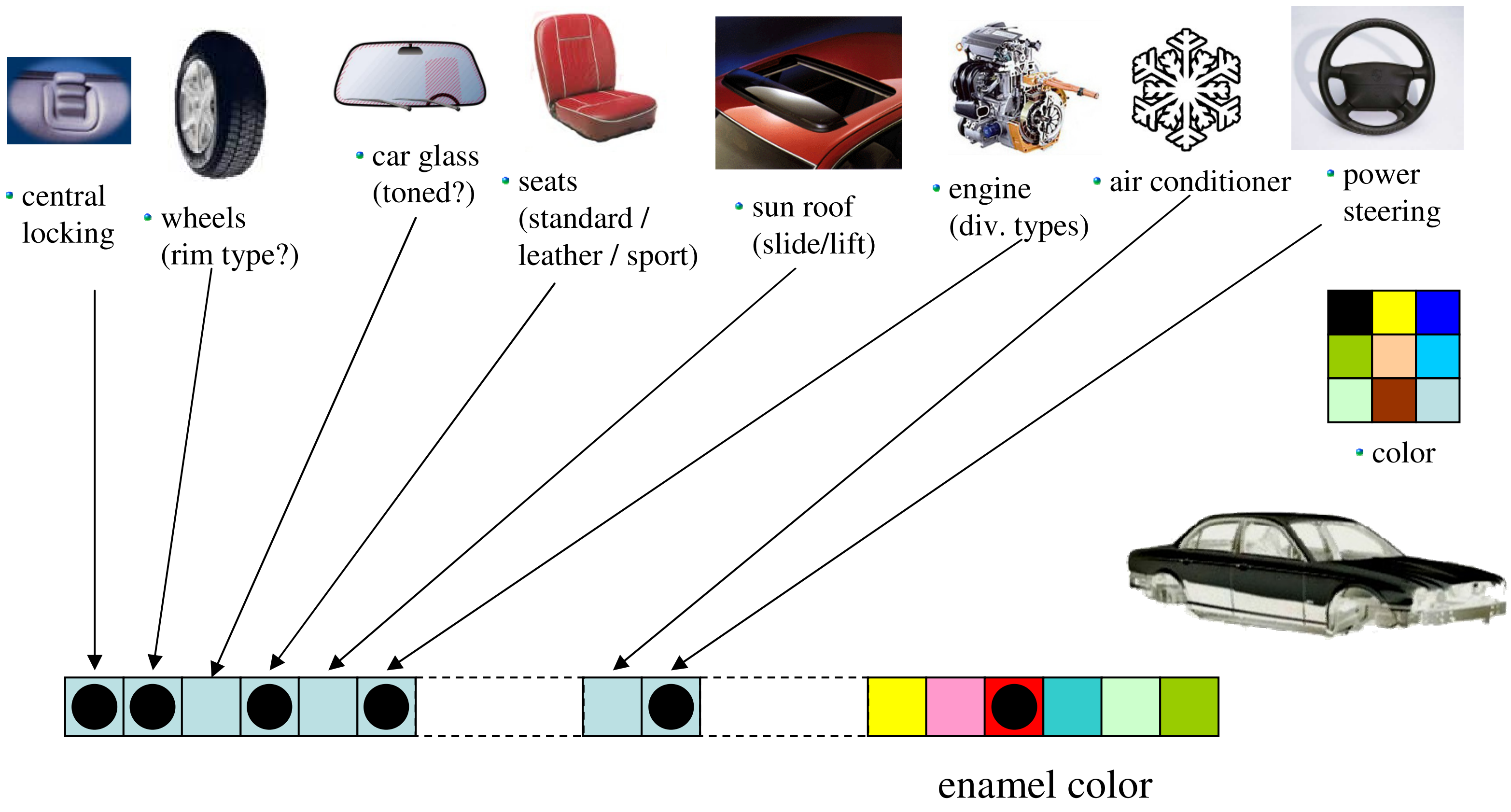
Commodities:



- central locking
- wheels (rim type?)
- car glass (toned?)
- seats (standard / leather / sport)
- sun roof (slide/lift)
- engine (div. types)
- air conditioner
- power steering
- color

Order:

Commodities:



- central locking
- wheels (rim type?)
- car glass (toned?)
- seats (standard / leather / sport)
- sun roof (slide/lift)
- engine (div. types)
- air conditioner
- power steering
- color

Order:

As Bit Vector:

enamel color

Manufacturing Errors

- Production process is essentially probabilistic
- Commodities are delayed due to manufacturing errors

➡ An exact optimization model is not sensible

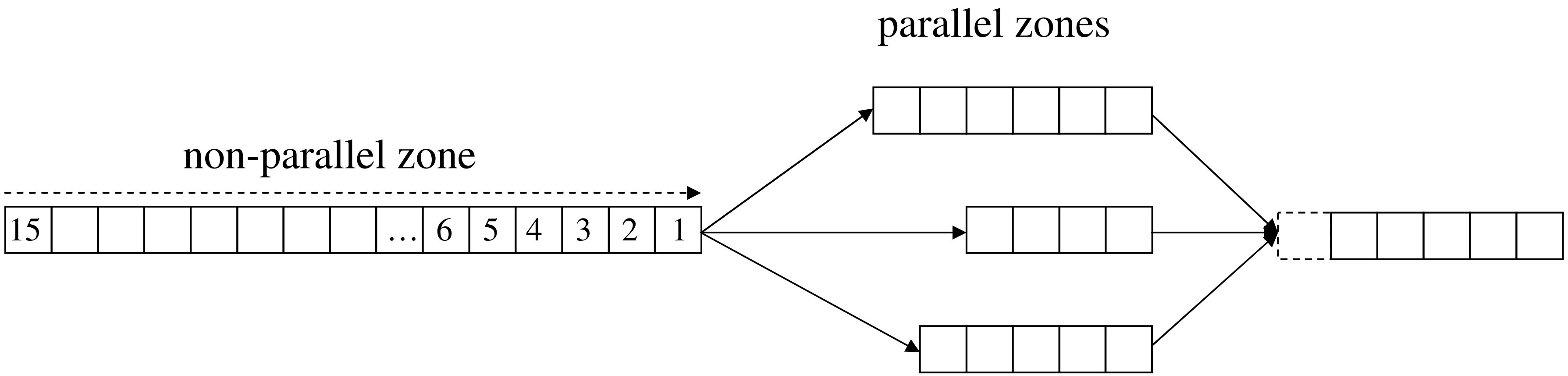
- Three major goals:
  - Robustness
  - Transparency
  - Performance

➡ Instead of objectives we use rules with priority to evaluate the result

➡ We model the production process in deterministic way

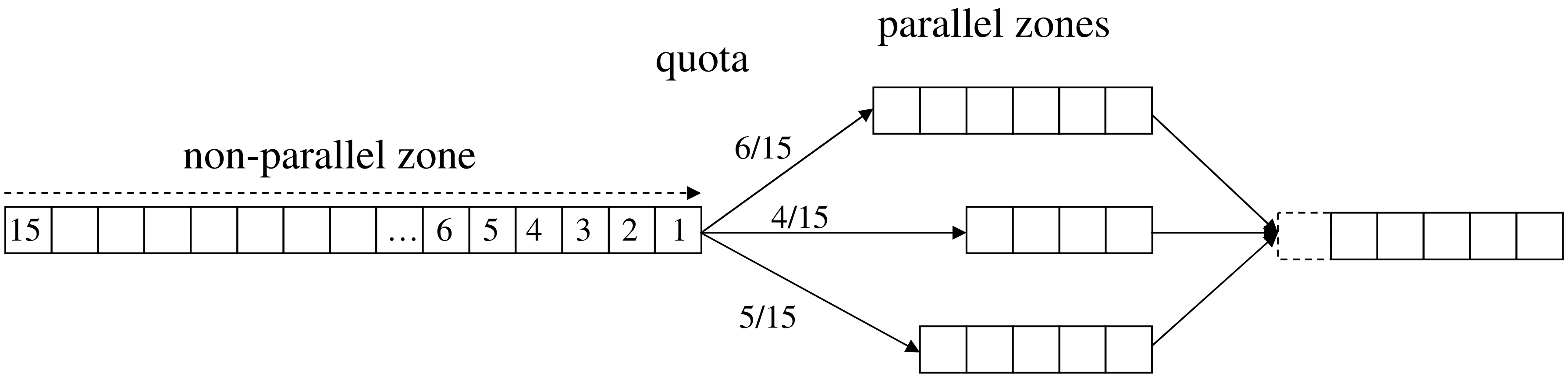
Deterministic Routing of Order Positions

- What happens at the split point of zones?
- Zones have different velocity / capacity
- The time an order needs to pass a zone is equal for parallel zones



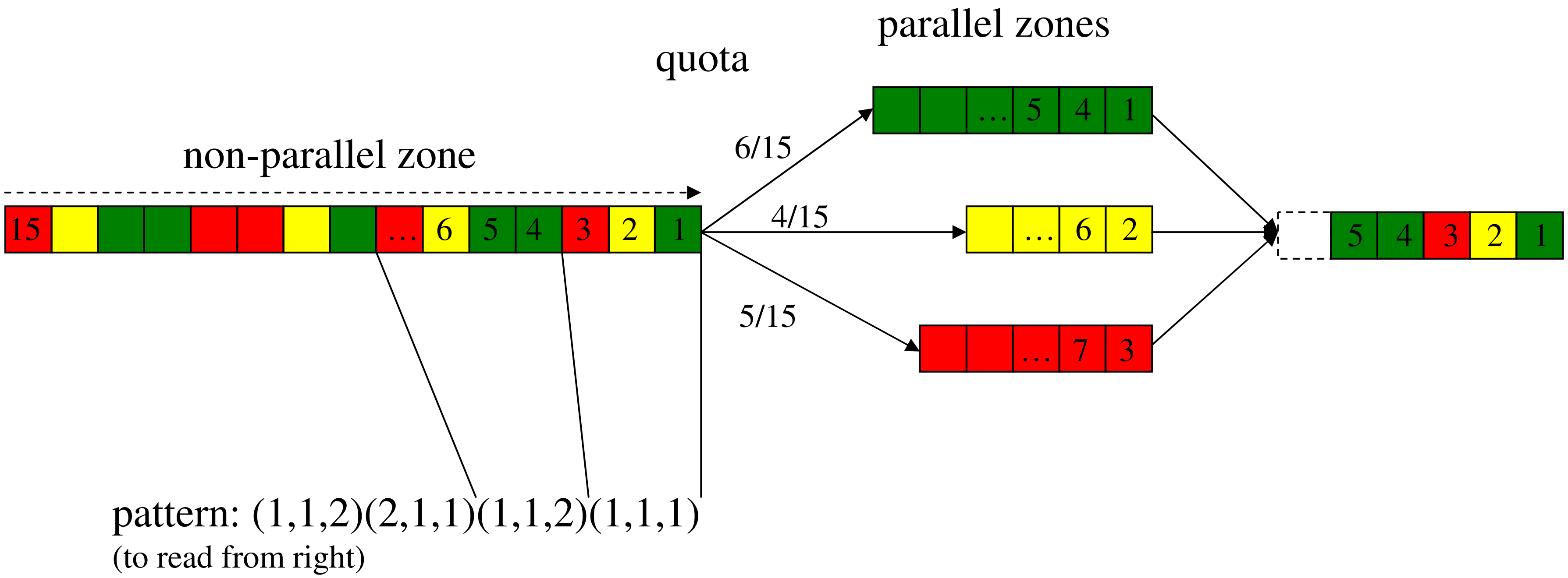
Deterministic Routing of Order Positions

- What happens at the split point of zones?
- Zones have different velocity / capacity
- The time an order needs to pass a zone is equal for parallel zones



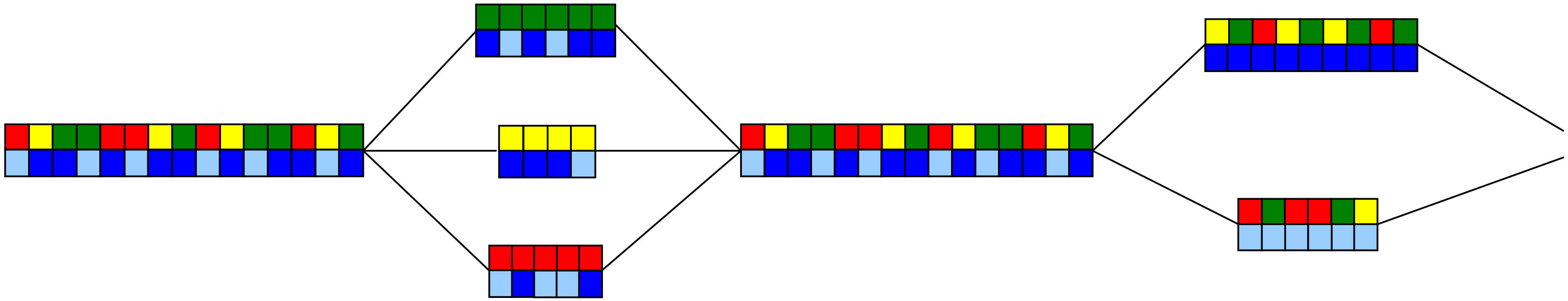
Deterministic Routing of Order Positions

- What happens at the split point of zones?
- Zones have different velocity / capacity
- The time an order needs to pass a zone is equal for parallel zones



Deterministic Routing of Order Positions

- There is more than one split point

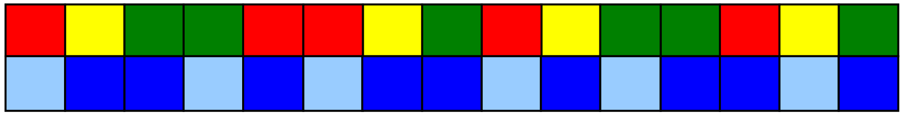


(1,1,2)(2,1,1)(1,1,2)(1,1,1)

(1,2)(1,1)(1,2)(1,1)(1,2)(1,1)

Deterministic Routing of Order Positions

It suffices to consider the master sequence and a set of patterns



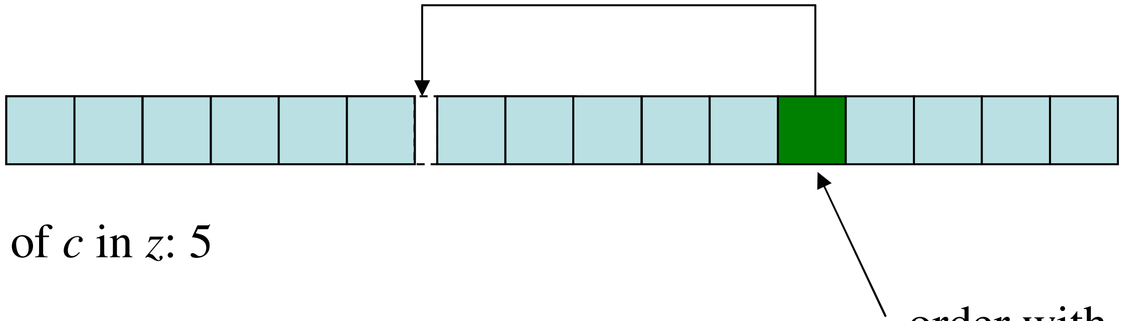
(1,1,2)(2,1,1)(1,1,2)(1,1,1)

(1,2)(1,1)(1,2)(1,1)(1,2)(1,1)

Manufacturing Errors

- Manufacturing errors could occur during the production process
- Use statistical data to determine the expected delay of a commodity on a zone

zone z:



expected delay of c in z: 5

order with commodity c

- c will appear later in zones succeeding z (respecting quotas)

➡ This enables to compute a more realistic order sequence

The Rule Set

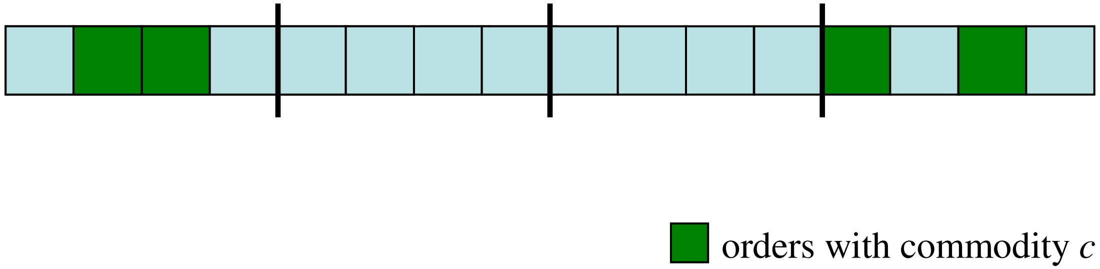
- Rules can be applied to each pair (c,z) of commodity and zone
- Any order containing c routed through zone z must respect this rule
- A priority p=1,…,10 is assigned to each rule
- Achieve a sequence with lexicographically minimal number of rule breaches

The Rule Set

- Rules can be applied to each pair (*c*,*z*) of commodity and zone
- Any order containing *c* routed through zone *z* must respect this rule
- A priority *p*=1,…,10 is assigned to each rule
- Achieve a sequence with lexicographically minimal number of rule breaches

- **Clustering Rule**: A commodity is allowed to occur in *s* out of *S* slots of the master sequence

cluster *c* in 2 out of 4 slots

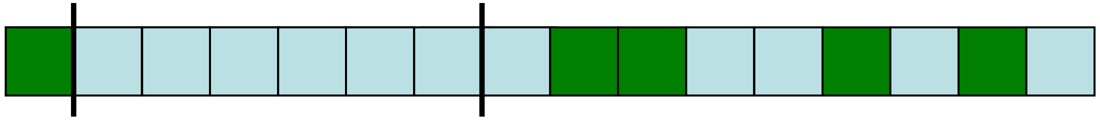


The Rule Set

- **Clustering Rule**: A commodity is allowed to occur in *s* out of *S* slots of the master sequence

- **Banning Rule**: A commodity is banned from an interval of a zone

ban *c* from 2,…,7



🟩 orders with commodity *c*

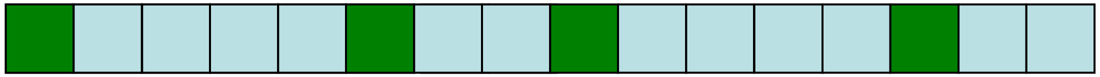
The Rule Set

- **Clustering Rule**: A commodity is allowed to occur in *s* out of *S* slots of the master sequence

- **Banning Rule**: A commodity is banned from an interval of a zone

- **Ratio Rule**: A commodity is allowed in at most *x* out of *y* consecutive sequence positions of a zone

allow *c* in 2 out of 8 positions



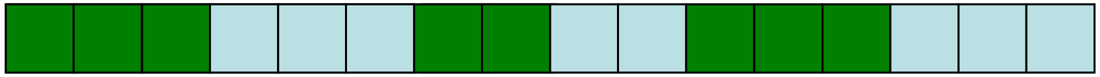
🟩 orders with commodity *c*

The Rule Set

- **Clustering Rule**: A commodity is allowed to occur in *s* out of *S* slots of the master sequence
- **Banning Rule**: A commodity is banned from an interval of a zone
- **Ratio Rule**: A commodity is allowed in at most *x* out of *y* consecutive sequence positions of a zone

- **Grouping Rule**: Orders that hold a commodity *c* should occur in groups with minimal and maximal size, where groups must keep a minimal distance

size of groups: 2,…,3
minimal distance: 2



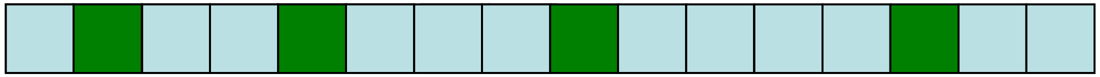
🟩 orders with commodity *c*

The Rule Set

- **Clustering Rule**: A commodity is allowed to occur in *s* out of *S* slots of the master sequence
- **Banning Rule**: A commodity is banned from an interval of a zone
- **Ratio Rule**: A commodity is allowed in at most *x* out of *y* consecutive sequence positions of a zone
- **Grouping Rule**: Orders that hold a commodity *c* should occur in groups with minimal and maximal size, where groups must keep a minimal distance

- **Spacing Rule**: Orders that hold a commodity *c* should occur with a minimal distance

minimal distance: 2



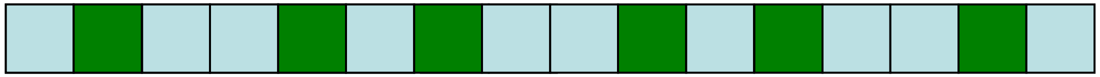
🟩 orders with commodity *c*

The Rule Set

- **Clustering Rule**: A commodity is allowed to occur in *s* out of *S* slots of the master sequence
- **Banning Rule**: A commodity is banned from an interval of a zone
- **Ratio Rule**: A commodity is allowed in at most *x* out of *y* consecutive sequence positions of a zone
- **Grouping Rule**: Orders that hold a commodity *c* should occur in groups with minimal and maximal size, where groups must keep a minimal distance
- **Spacing Rule**: Orders that hold a commodity *c* should occur with a minimal distance

- **Spreading Rule**: Orders with a commodity should be evenly spread

average distance: 1.5



🟩 orders with commodity *c*

The Rule Set

- **Clustering Rule**: A commodity is allowed to occur in *s* out of *S* slots of the master sequence
- **Banning Rule**: A commodity is banned from an interval of a zone
- **Ratio Rule**: A commodity is allowed in at most *x* out of *y* consecutive sequence positions of a zone
- **Grouping Rule**: Orders that hold a commodity *c* should occur in groups with minimal and maximal size, where groups must keep a minimal distance
- **Spacing Rule**: Orders that hold a commodity *c* should occur with a minimal distance
- **Spreading Rule**: Orders with a commodity should be evenly spread

- Two concurrent strategies for rule evaluation
  - Count the number of rule breaches for each priority (quantity of rule breaches)
  - Increase a penalty value for a broken rule (quality of rule breaches)

- Clustering rule is considered separately

Greedy Master Sequence Construction

Simplification:

- Clustering rules are zone independent
- There is at most one clustering commodity per order (suitable for color clustering)

Greedy Master Sequence Construction

Simplification:

- Clustering rules are zone independent
- There is at most one clustering commodity per order (suitable for color clustering)

- Orders are assigned to slots of the master sequence
- Compute each slot of the master sequence separately (Divide & Conquer)

Greedy Master Sequence Construction

Simplification:

- Clustering rules are zone independent
- There is at most one clustering commodity per order (suitable for color clustering)

➡ Orders are assigned to slots of the master sequence

➡ Compute each slot of the master sequence separately (Divide & Conquer)

- Known approach: Goal Chasing (Monden 1983)
  - Choose step by step an order which assures an even resource consumption
  - Only suitable for spreading

$$\left|\text{desired resource consumption} - \text{current resource consumption}\right|^2 \rightarrow \min$$

Greedy Master Sequence Construction

Simplification:

- Clustering rules are zone independent
- There is at most one clustering commodity per order (suitable for color clustering)

- Orders are assigned to slots of the master sequence
- Compute each slot of the master sequence separately (Divide & Conquer)

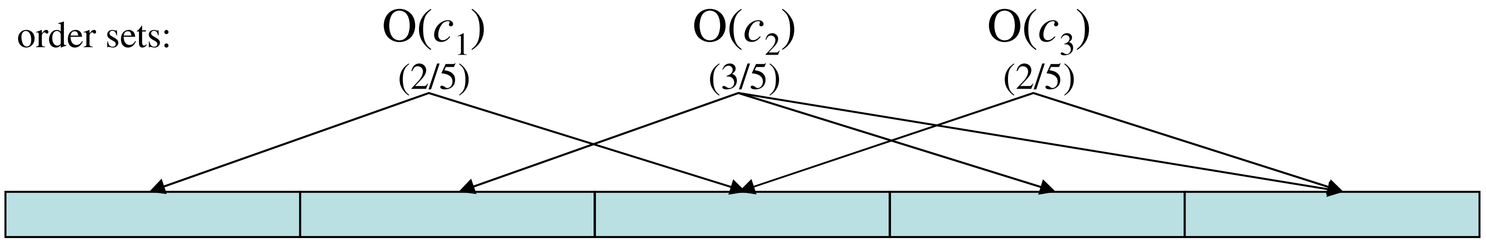
- Modified Goal Chasing approach:
  - Choose in every step a “best” order
    - with lexicographically minimized rule breach vector b=(b_1,…, b_{10})
    - and minimized penalty (priority-weighted penalties)

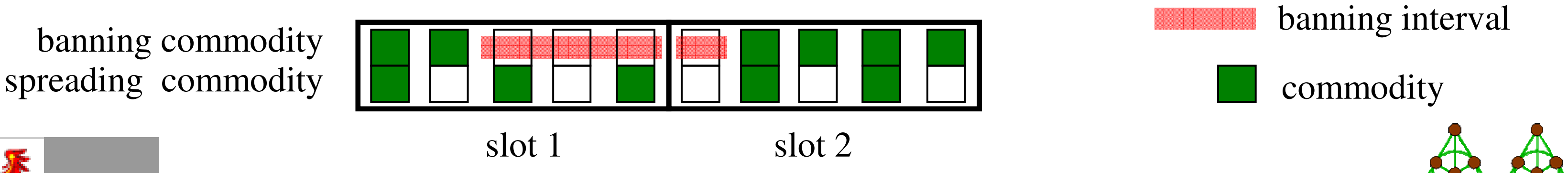
- Fully configurable by the choice of priorities

Order Clustering Algorithm

- Let $c_1,\ldots,c_u$ denote all clustering commodities and O(c) the set of all orders containing commodity c.

1. Assign to each clustering commodity its preferred slots
   - Other banning rules could imply varying ratio of a commodity
   - Spreading commodities must be evenly distributed over the slots



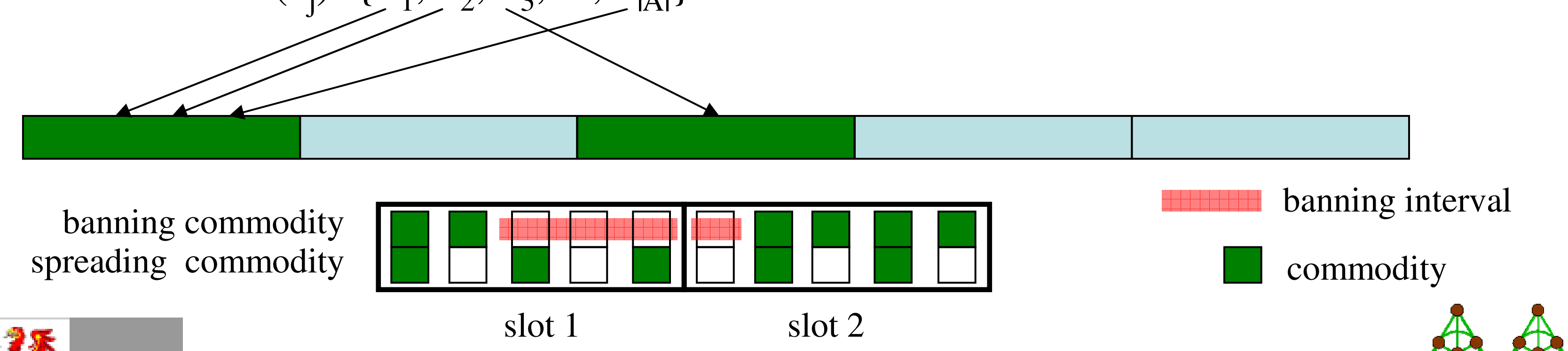


Order Clustering Algorithm

- Let $c_1,\ldots,c_u$ denote all clustering commodities and O(c) the set of all orders containing commodity c.

1. Assign to each clustering commodity its preferred slots
   - Other banning rules could imply varying ratio of a commodity
   - Spreading commodities must be evenly distributed over the slots
2. Assign each order to a preferred slot
   - Consider banning and spreading rules

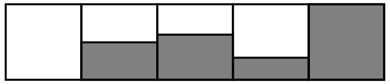
$O(c_j)$: $\{o_1, o_2, o_3,\ldots, o_{|A|}\}$



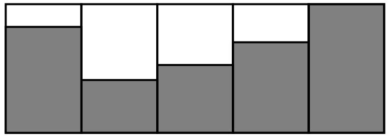
Best Vector Packing into Bins Problem

- Let $e_1,\ldots,e_r$ denote all spreading commodities
- For each set O(c_i) introduce a vector w_i (U is the set of non-clustering orders)

$$w_{i,j} \quad := \begin{cases} |O(c_i) \cap O(e_j)| & j = 1,\ldots,r \\ |O(c_i)| & j = r+1 \end{cases}$$



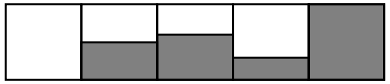
$$w_{u+1,j} := \begin{cases} |U \cap O(e_j)| & j = 1,\ldots,r \\ |U| & j = r+1 \end{cases}$$



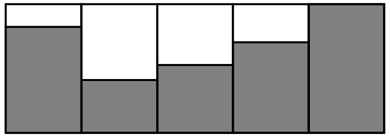
Best Vector Packing into Bins Problem

- Let $e_1,\ldots,e_r$ denote all spreading commodities
- For each set O(c_i) introduce a vector w_i (U is the set of non-clustering orders)

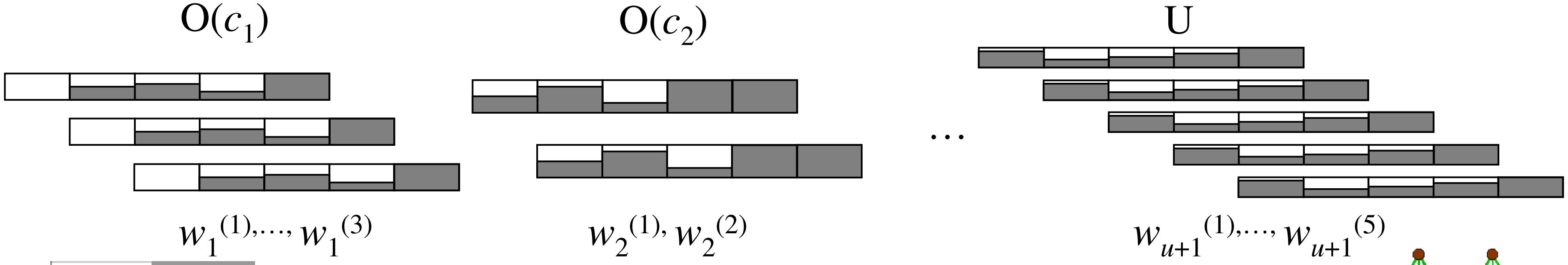
$$w_{i,j} := \begin{cases} |O(c_i) \cap O(e_j)| & j = 1,\ldots,r \\ |O(c_i)| & j = r+1 \end{cases}$$



$$w_{u+1,j} := \begin{cases} |U \cap O(e_j)| & j = 1,\ldots,r \\ |U| & j = r+1 \end{cases}$$

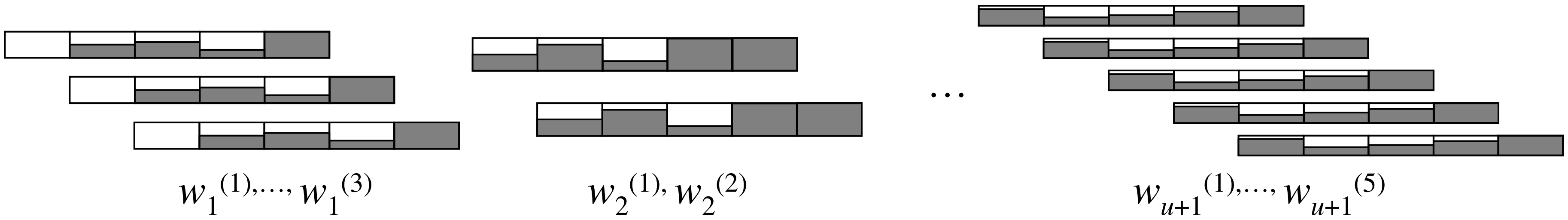


- Split these vectors according to the desired number of slots

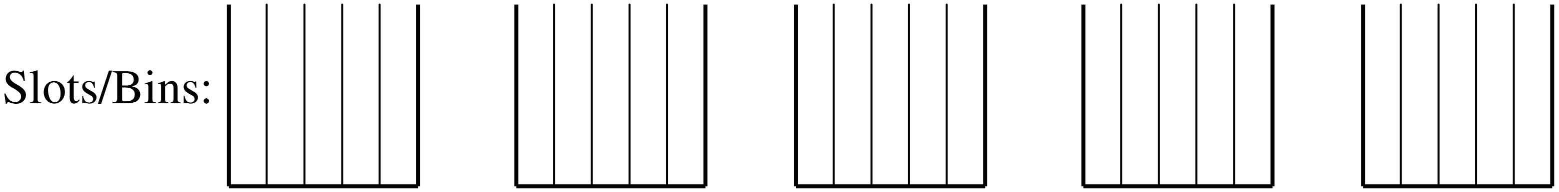

O(c_1)
O(c_2)
U
…
$w_1^{(1)},\ldots, w_1^{(3)}$
$w_2^{(1)}, w_2^{(2)}$
$w_{u+1}^{(1)},\ldots, w_{u+1}^{(5)}$

Best Vector Packing into Bins Problem

- Let $e_1,\ldots,e_r$ denote all spreading commodities

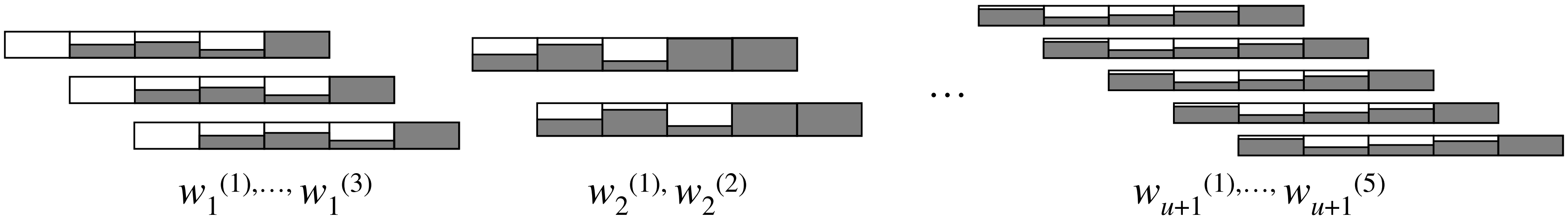

$w_1^{(1)},\ldots, w_1^{(3)}$
$w_2^{(1)}, w_2^{(2)}$
…
$w_{u+1}^{(1)},\ldots, w_{u+1}^{(5)}$

- Slots contain vectors indexed from 1 to r+1
- The last component is reserved to ensure that a slot is filled completely



Best Vector Packing into Bins Problem

- Let $e_1,\dots,e_r$ denote all spreading commodities

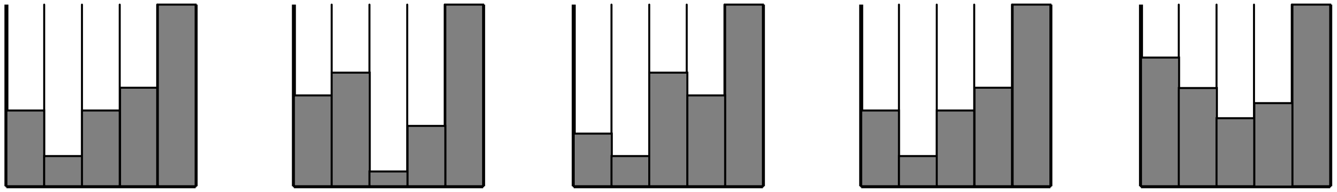


…
$w_1^{(1)},\dots, w_1^{(3)}$
$w_2^{(1)}, w_2^{(2)}$
$w_{u+1}^{(1)},\dots, w_{u+1}^{(5)}$

- Slots contain vectors indexed from 1 to r+1
- The last component is reserved to ensure that a slot is filled completely

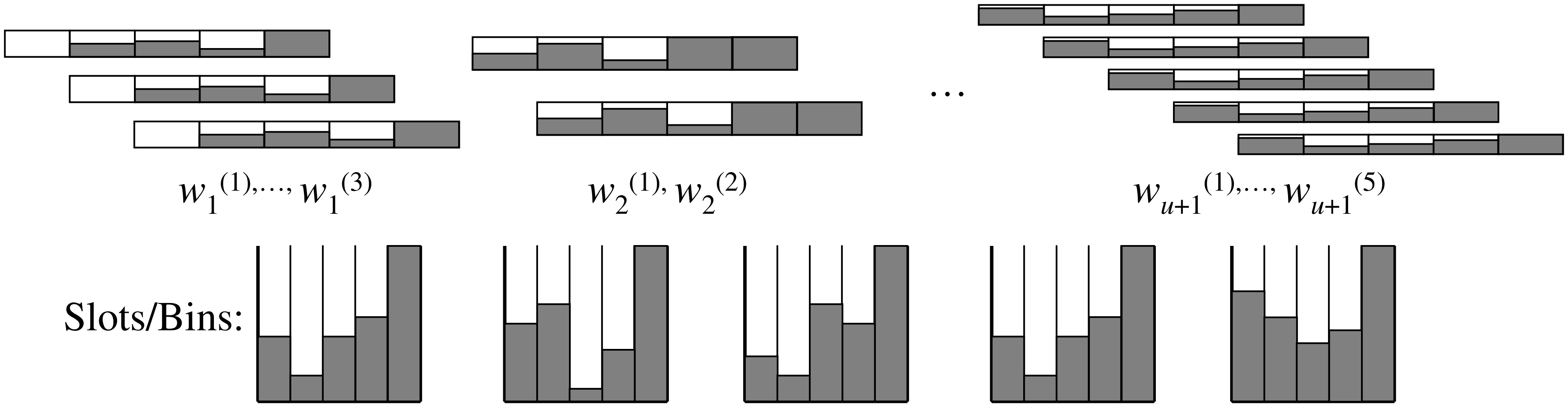
Desired slot contents: $$w^*_{s,j} := \begin{cases} \frac{|O(e_j)|}{S} \cdot b(e_j, s) & j = 1,\dots,r \\ \frac{n}{S} & j = r+1 \end{cases}$$

Slots/Bins:



Best Vector Packing into Bins Problem

- Let $e_1,\ldots,e_r$ denote all spreading commodities


$w_1^{(1)},\ldots, w_1^{(3)}$
$w_2^{(1)}, w_2^{(2)}$
…
$w_{u+1}^{(1)},\ldots, w_{u+1}^{(5)}$
Slots/Bins:

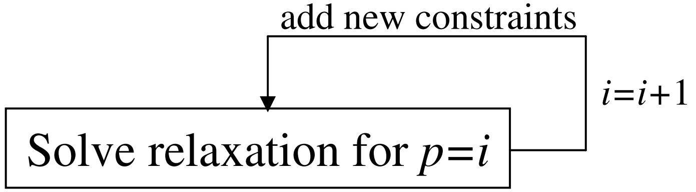
- Greedy approach: In each step choose a pair of vector and bin, so that the vector improves on the bins contents best possible

- This yields an assignment of an order o to its set of preferred slots $S^*(o)$.

Assignment of Orders to Slots

- Given for each order o a set $S^*(o)$ of preferred slots
- Assign each order to a preferred slot (respect other banning / spreading rules)

- Binary variable $x_{o,s}$ to indicate that o is assigned to slot s.
- Spreading objective: $\sum_{o:c\in o} x_{o,s} = O(c)/S\cdot b(c,s), \quad \forall s=1,\dots,S,\ c\in C_{Spr}$
- Clustering objective: $\sum_{o:c\in o,\ s\notin S^*(o)} x_{o,s} = 0, \quad \forall s=1,\dots,S,\ c\in C_{Clu}$
- Slot size constraint: $\sum_{o} x_{o,s} = n/S, \quad \forall s=1,\dots,S$
- Banning constraint: $\sum_{o:c\in o} x_{o,s} \le n/S-|B(c,s)|, \quad \forall s=1,\dots,S,\ c\in C_{Ban}$



add new constraints

i=i+1

Solve relaxation for p=i

- Number of variables can be reduced significantly

Results

- Easily configurable
  - simple rule definitions
  - transparent parameter tuning with priorities

- High performance
  - Operates with greedy subroutines and on small linear programs only
  - About 3 minutes on 1500 orders and realistic rules (Sun 450 MHz, 1GB)

- Color batch size increased by 50%

- The algorithm is currently used in all automobile plants of the Ford Motor Company across Europe

BTU

Thanks for your attention.