Solving a scheduling problem at cross docking terminals

Ruslan Sadykov

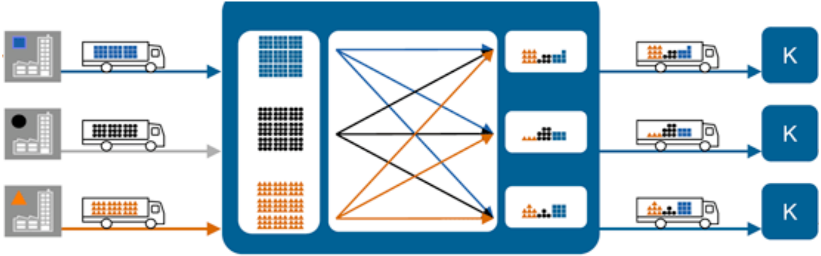
INRIA Bordeaux — Sud-Ouest

Institut de Mathématiques de Bordeaux

EURO XXIV
Lisbon, July 12, 2010

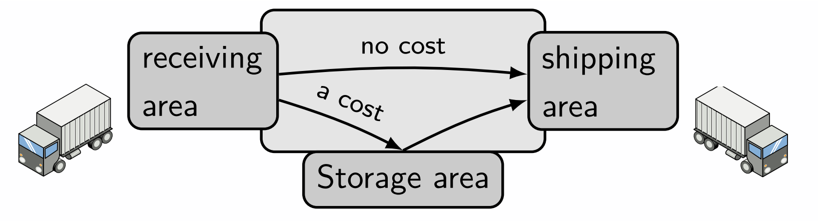
Contents

Cross docking



- Product of several types should be delivered from several production/distribution points to several costumers.
- The cross docking terminal serves to reallocate goods according to their destinations (costumers) in order to reduce the transportation costs.

Cross-docking scheduling problem



- If a product unit goes to the storage, a cost should be paid.
- An incoming (outgoing) truck leaves the door only if it is fully unloaded (loaded)
- We need to schedule the sequences of incoming and outgoing trucks and obtain a product transfer policy which minimizes the cost.

Negative result

The problem is NP-hard in the strong sense even if

- There is only one receiving door and one shipping door.
- Incoming trucks supply products of at most 2 types.
- Outgoing trucks demand products of one type.
- Storage costs are unitary.
- Storage capacity is unlimited.

Exact methods

- Yu and Egbelu (2008)
- Boysen, Fliender, and Scholl (2010)

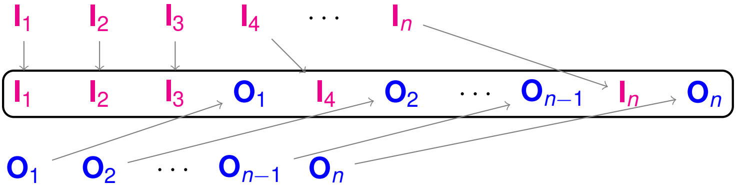
Contents

Cross docking scheduling problem: notations

- n incoming trucks, m outgoing trucks ($m = n$ for simplicity)
- T different products types
- An incoming truck $\mathbf{I}_i$ supplies a_{it} units of product type t
- An outgoing truck $\mathbf{O}_o$ demands b_{ot} units of product type t
- Each outgoing truck demands products of at most q types
- The cost of storing one unit of product type t is c_t
- The volume of a unit of product type t is d_t
- Storage capacity is D.

Cross docking scheduling problem: special case

- There is only one receiving door and one shipping door.
- The sequences of incoming and outgoing trucks are fixed:



- We need to find
  - an aggregate sequence of truck arrivals/departures,
  - a product transfer policy,

  which minimize the storage cost (maximizes the weighted number of product units transferred directly).
- Introduced by Maknoon, Baptiste, and Kone (2009) ($q = 1$).

A dominance rule

Observation

There exists an optimal policy in which, each time trucks $\mathbf{I}_k$ and $\mathbf{O}_j$ are at the doors, for each t, $\mathbf{I}_k$ transfers directly to $\mathbf{O}_j$ as many products of type t as possible.

Consequence

- We call a policy complying with the observation direct first.
- For each departure sequence of trucks, there is exactly one direct first policy.

Contents

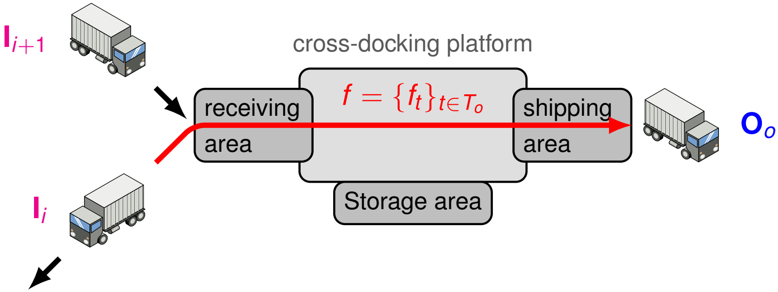
Dynamic programming states: first group

$\mathbf{S}^{out}(i, o, f)$ — departure sequence is

$$\ldots, \mathbf{O}_{o-1}, \mathbf{I}_i, \ldots$$

$f = \{f_t\}_{t \in T_o}, \quad 0 \leq f_t \leq \min\{a_{it}, b_{ot}\},$

f_t — number of products of type t transferred from $\mathbf{I}_i$ to $\mathbf{O}_o$



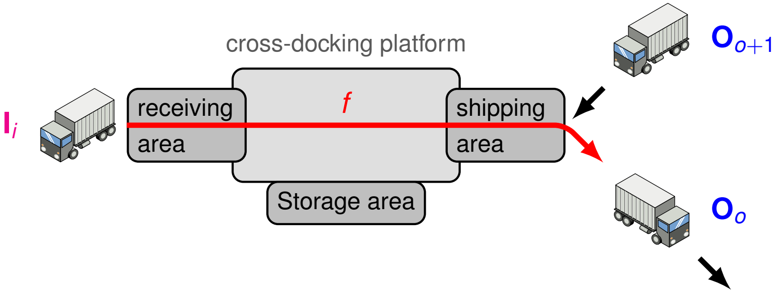
Dynamic programming states: second group

$\mathbf{S}^{inc}(i, o, f)$ — departure sequence is

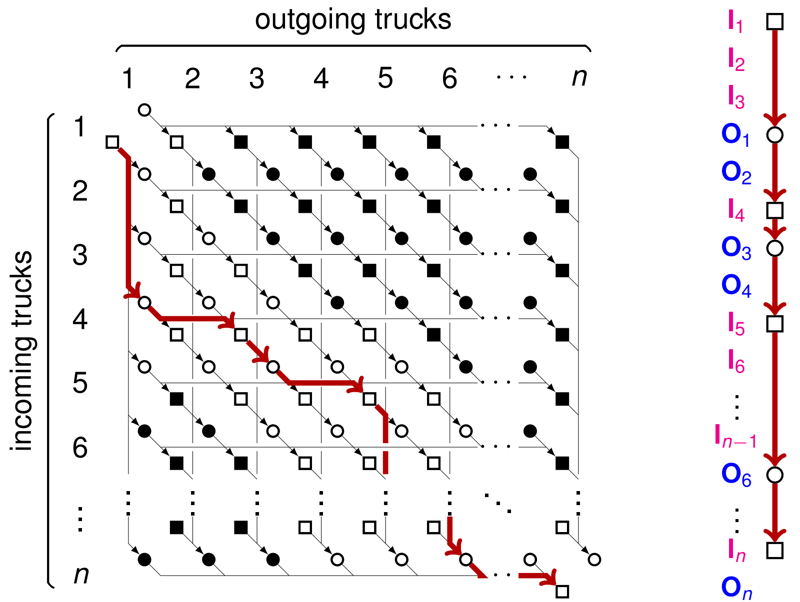
$$\ldots, \mathbf{I}_{i-1}, \mathbf{O}_o, \ldots$$

$f = \{f_t\}_{t \in T_o}, \quad 0 \leq f_t \leq \min\{a_{it}, b_{ot}\},$

f_t — number of products of type t transferred from $\mathbf{I}_i$ to $\mathbf{O}_o$



The underlying graph for the dynamic programming



Number of the dynamic programming states

- In a state $\mathbf{S}(i, o, f)$, for every type t,

$$0 \leq f_t \leq \min\{a_{it}, b_{ot}\}.$$

- Then, the overall number of states is a pseudo-polynomial of n and an exponential of q:

$$\mid \mathbf{S} \mid = \sum_{i=1}^{n} \sum_{o=1}^{n} \prod_{t:b_{ot}>0} \big(\min\{a_{it}, b_{ot}\} + 1 \big) = O(n^2 \cdot AB^q),$$

where $AB = \max_{i,o,t} \min\{a_{it}, b_{ot}\}$.

- But the number of direct first states (which correspond to a direct first policy) is polynomial.

Complexity of the dynamic programming algorithm

Theorem

The total number of the direct first states $\mathbf{S}^{out}$ is $O(qn^3)$ (same holds for $\mathbf{S}^{inc}$).

- Complexity of checking whether a state $\mathbf{S}(i, o, f)$ has been already visited is $O\big(q \log(qn^2)\big) = \rho$.
- Complexity of making all moves from a state $\mathbf{S}(i, o, f)$ is $O\big(n(q + \rho)\big) = O(nq \log n)$.

Theorem

The complexity of the dynamic programming algorithm is

$$O\big(q^2 n^4 (q + \log n)\big)$$

Contents

Test instances

Parameters

- $n = 100, 200, 400, 800$
- $q = 1, 2, 4, 8$
- $\mid T \mid = 10q$
- $a_{it} \in U[1, 1000]$
- $c_t \in U[1, 10]$
- storage capacity is unlimited

Number of instances

10 instances generated for each pair (n, q)

Numerical results

$|\,\mathbf{S}\,|$ — number of the created states, in thousands

RT — average running time, in seconds

n	$q = 1$		$q = 2$		$q = 4$		$q = 8$	
	$\mid \mathbf{S} \mid$	RT	$\mid \mathbf{S} \mid$	RT	$\mid \mathbf{S} \mid$	RT	$\mid \mathbf{S} \mid$	RT
100	13	0.01	18	0.02	24	0.03	36	0.06
200	77	0.13	107	0.19	168	0.40	286	0.92
400	365	1.37	533	2.09	877	4.22	1’549	10.05
800	1’626	15.97	2’444	22.53	4’175	41.56	7’477	93.52

When n doubles, running time is 11.3 times larger on average.

When q doubles, running time is 1.9 times larger on average.

Conclusions and perspectives

Conclusion

- We presented a polynomial dynamic programming algorithm for the problem.
- Note that the complexity question was open (even for $q = 1$).

Perspectives

- Linear Programming formulation?

Contents

Number of different values for f_t

- Value f_t is canonical in a state $\mathbf{S}(i, o, f)$ if

$$f_t \in \{0, \min\{a_{it}, b_{ot}\}\}.$$

- From any state $\mathbf{S}^{out}(i, o, f)$, we can pass to at most one direct first state $\mathbf{S}^{inc}(i', o, f')$, $i' > i$, with a non-canonical value f_t.
- From any state $\mathbf{S}^{inc}(i, o, f)$, we can pass to at most one direct first state $\mathbf{S}^{out}(i, o'', f'')$, $o'' > o$, with a non-canonical value f_t.
- Therefore, any state with a canonical value f_t “generates” at most $2n$ direct first states with non-canonical values f_t.
- Then, the number of different values for f_t in all direct first states is $O(n^3)$.

Number of the direct first DP states

Lemma

For fixed i^* and o^*, there are no two direct first states $\mathbf{S}^{out}(i^*, o^*, f')$ and $\mathbf{S}^{out}(i^*, o^*, f'')$ such that $f'_{t_1} < f''_{t_1}$ and $f'_{t_2} > f''_{t_2}$.

Consequence

For fixed i^* and o^*, direct first states $\mathbf{S}^{out}(i^*, o^*, f)$ can be lexicographically ordered:

$$\mathbf{S}^{out}(i^*, o^*, f') \prec \mathbf{S}^{out}(i^*, o^*, f'') \Leftrightarrow f'_t \leq f''_t, \forall t.$$

Theorem

The total number of the direct first states $\mathbf{S}^{out}$ is $O(qn^3)$ (same holds for $\mathbf{S}^{inc}$).