A bounded dynamic programming approach to schedule operations in a cross docking platform

Gülgün Alpan, Rim Larbi, Bernard Penz

A bounded dynamic programming approach to schedule operations in a cross docking platform

Gülgün Alpan *, Rim Larbi, Bernard Penz

Laboratoire G-SCOP, Grenoble INP-UJF-CNRS, 46 avenue Félix Viallet, 38031 Grenoble, France

Cross docking is a logistic technique employed to reduce the inventory holding, order picking, transportation costs as well as the delivery time. Products arriving to the cross dock are unloaded from inbound trailers, possibly reconsolidated with other products arriving from different destinations and loaded into outbound trailers within less than 24 h. In this study, we consider a multiple receiving and shipping door cross dock environment. The objective is to find optimal (for reasonably small cross docks) or near optimal (for larger cross docking facilities) scheduling policies which minimizes the total costs related to the transshipment operations at the facility.

1. Introduction

A platform of cross docking is a consolidation point of inbound products and offers short cycle times. Materials arriving to the cross dock from suppliers are unloaded from the inbound trailer, sorted according to their destinations, possibly consolidated with other products to the same destination and reloaded into an outbound trailer within less than 24 h. Therefore, this technique is mainly used in the transportation industries or in the distribution of the perishable products.

In the literature on cross docking, we can find studies dealing with problems either on the strategical or the operational level. The solutions for strategical problems often require an investment and the decisions taken are not frequently modifiable. For instance cross dock network design (see Chen, Guo, Lim, & Rodrigues, 2006; Donaldson, Jonhson, Ratliff, & Zhang, 1998; Ratliff, Vate, & Zhang, 1998) or the layout of cross docking platforms (see Bartholdi & Gue, 2002; Bartholdi & Gue, 2004; Gue, 1999) make part of this category of problems.

The problems which are handled in the operational level are mainly on the real-time control of the cross docking platforms and hence the decisions are modified on a real-time basis. In this category of problems, we can cite the dock assignment problems, the objective of which is the assignment of inbound and outbound trucks on the docks in order to optimize a criterion (see Tsui & Chang, 1990; Tsui & Chang, 1992 for the minimization of the weighted distance between inbound and outbound trucks; see (Bartholdi & Gue, 2001) for the minimization of the congestion within the cross docking platform.) Scheduling of transshipment operations inside the cross docking platforms are also in this group of problems (see Alpan, Bauchau, Larbi, & Penz, 2008; Baptiste & Maknoon, 2007; Baptiste, Penz, & Larbi, 2007; Boysen, 2009; Boysen, Fliedner, & Scholl, 2008; Chen & Lee, 2009; Chen & Song, 2009; Larbi, Alpan, Baptiste, & Penz, 2007; Larbi, Alpan, & Penz, 2009; McWilliams, Stanfield, & Geiger, 2005; Sadykov, 2009; Song & Chen, 2007; Yu & Egbelu, 2008). These recent studies seek to find the best schedule of trucks so that either the time or the cost related performance measures of the cross dock is optimized. Among these studies, (Baptiste & Maknoon, 2007; Baptiste et al., 2007; Boysen et al., 2008; Chen & Lee, 2009; Larbi et al., 2007; Sadykov, 2009; Yu & Egbelu, 2008) consider the case where there is a single receiving and a single shipping dock. These studies give interesting insights on the solution structure for scheduling problems at the cross docking facilities, however no practical application is possible. In practice a cross docking facility has several receiving and shipping docks. The largest facilities can have several hundreds of docks. Therefore, for implementation purposes, it is important to study the multi door cross docking settings.

To the best of our knowledge, the scheduling problems in multi door cross docks are studied in Alpan et al. (2008), Boysen (2009), Chen and Song (2009), Larbi et al. (2009), McWilliams et al. (2005), Song and Chen (2007). Chen and Song (2009) presents the cross docking scheduling problem as a two-stage flow shop problem with parallel machines. Each stage corresponds to either inbound or outbound side of a cross dock, the machines and the set of jobs are analogous to inbound or outbound docks and the trucks to unload or load, respectively. Indeed, this analogy was first established by Chen and Lee (2009) where each stage of the problem contained only a

* Corresponding author. Tel.: +33 476574333.

E-mail addresses: gulgun.alpan@grenoble-inp.fr (G. Alpan), rim.larbi@grenoble-inp.fr (R. Larbi), Bernard.Penz@grenoble-inp.fr (B. Penz).

single machine (i.e. single inbound and outbound dock). They show that the problem is NP-hard in the strong sense. An extended version is proposed by Chen and Song (2009). In this later study, at least one of the stages is allowed to have more than one machine. In these studies, the objective is to find the best schedule of inbound and outbound trucks so that the makespan of operations is minimized. The same authors have studied a similar, yet simplified version of the same scheduling problem in Song and Chen (2007).

The industrial context may impose specific constraints on the cross docking operations. McWilliams et al. (2005) and Boysen (2009) hence consider some specific industrial settings. McWilliams et al. (2005) present the parcel hub scheduling problem common in parcel delivery industries such as the postal services. The objective of this study is very similar to the scheduling problems in multi door cross docking facilities, i.e. finding the best schedule of inbound trucks so that the makespan of the parcel transfer operations is minimized. However, the environmental setting has some specificities. One of the characteristics of a parcel hub compared to a classical cross docking platform is the type of materials handling system utilized. In a parcel hub, the flow of the materials is supported by a network of fixed conveyor belts. Temporary storage of pallets are not considered. The main focus is on the congestion of the fixed conveyor belts by the untimely unloading of the incoming parcels. Therefore, we consider the parcel hub scheduling as a special case of cross docking. Another special case is studied in Boysen (2009) for a food industry cross docking facility. In this case, the inventory holding is strictly forbidden. The author presents a dynamic programming approach as well as heuristics based on simulated annealing to schedule the inbound and outbound trucks. Three time-related objective functions are considered: minimization of total processing time, total flow time and the total tardiness.

The current article is an extended version of a previous study presented at the 38th International Conference on Computers and Industrial Engineering (Alpan et al., 2008). The cross docking environment under study is a multi receiving and shipping dock setting. In this paper, similar to the above presented literature, we would like to determine the best schedule of outbound trucks which should be present at the shipping doors at any given time, given a known sequence of inbound truck arrivals. Our problem differs from the previous work in two aspects: (i) In all of the previous work, the objective function is a time-related one, such as the minimization of makespan, or total tardiness. This is rather classical in scheduling and is an indirect way of considering operational costs. In this paper, we will directly focus on operational costs related to temporary storage of merchandize inside the cross dock and the costs related to pre-emption of loading operations at the docks. (ii) The second difference comes from the problem structure. Here we allow, pre-emption of loading operations which is not allowed in previous work. Furthermore, we explicitly model the temporary storage, which is either forbidden or not modeled explicitly in the existing literature on multi dock cross-dock scheduling problems. We note that in practice, temporary storage or pre-emption are solutions to increase the flexibility of cross docking operations.

The rest of the article is organized as follows. In Section 2 we will give a detailed description of the problem with the basic assumptions and the related input data to the problem. In Section 3, we will briefly present the dynamic program used for the resolution as well as some properties taken into consideration during the resolution phase (Alpan et al., 2008). This section will also include the performance limits of the DP model illustrated by numerical results. Section 4 is dedicated to the presentation of some bounds on the DP model in order to reduce its complexity. Performance of the resultant bounded dynamic programs are illustrated by numerical experiments in Section 5. And finally concluding remarks are given in Section 6.

2. Problem description

In this section we will introduce the basic operations that are realized in the cross docking platform under study. Basic notations used and the assumptions considered will also be given.

We consider a cross docking platform with $I \geqslant 1$ receiving and $O > 1$ shipping doors. The products which transit the facility are sent to one of the D different destinations. Each outbound truck serves a single destination, d, $d = \{1, \ldots, D\}$. Each inbound truck arriving to the platform may contain products for several destinations. If an outbound truck in destination to d is present at a shipping door, the products in destination to d are directly loaded from the inbound truck into the outbound truck. Otherwise, (i) the incoming products are either temporarily stored and a holding cost, h, is paid per unit product (ii) or one of the trucks occupying a shipping door is moved to a parking zone, liberating thus the door for loading another trailer (for example the one to destination d). In this latter case, a truck replacement cost, r, is paid. Naturally, the outbound trucks which are full and are leaving the platform are excluded from this penalty.

The major assumptions of the study are enumerated below.

- A_1: The products are identically conditioned in unit size pallets. Hence, all transshipment operations on every pallet are done in an identical unit time, τ. That is, a pallet can be unloaded, then either be directly loaded into an outbound truck or transfered to the storage area in τ time units. Similarly, a pallet which is temporarily stored will take τ time units to load into an outbound truck. Without loss of generality, we assume $\tau = 1$.
- A_2: The arrival sequence of inbound trucks, as well as their contents and the position of the merchandize in the truck are known. Without loss of generality, we assume that all departing inbound trucks are immediately replaced by a new one. We note that this assumption can be relaxed technically, by inserting empty inbound trucks in the arrival sequence to fill the time gap between inbound trucks.
- A_3: The inbound trucks are assigned to receiving doors based on a FIFO policy.
- A_4: There is sufficient workforce to load/unload all docked trailers at the same time. Hence, a trailer assigned to a dock does not wait for the availability of a material handler.
- A_5: The pallets in the inbound trucks have priority on the pallets already stored in the cross dock. This assumption is a logical consequence of the cost structure and assumption A_1. Since the holding cost h is per unit stored and the time of the storage is ignored, a pallet staying in the cross dock for the whole day will cost the same as the pallet stored for a few minutes. Furthermore, any pallet transferred directly from an inbound to an outbound truck will take only τ time units compared to 2τ time units for the stored items.
- A_6: The outbound truck fleet is well dimensioned with interchangeable standard trucks so that no time is lost waiting for the arrival of an outbound truck.
- A_7: All products arriving to the cross dock during the day should leave the cross dock the same day.

We note that by assumption A_1, we discretize time into intervals of length 1. Each time interval will be denoted by t, $t = 1, 2, \ldots, T$. By assumptions A_1 and A_4 we allow I arriving pallets to be handled at the same time at the inbound docks at a given time interval t. Furthermore, if several pallets to destination d are available at distinct receiving docks and an outbound truck to destination d is also available at the shipping dock at time interval t, these pallets can directly be loaded on the outbound truck at the

same time. However, if pallets to destination d are also available in the storage at the same time interval t, we will give priority to direct loading (by assumption $A5$); the already stored pallets will be loaded another time.

With assumptions A_1, A_2 and A_3, we can construct a fixed arrival sequence of pallets, denoted by S. For each receiving dock, we represent the sequence of pallets unloaded at this dock by their destination. By assumption A_1, each arriving pallet takes τ units to handle. Hence, the arrival sequence S is represented as a matrix of size $I \times T$, where T is the time at which the last pallet in the final inbound truck is accessed. The tth column of S gives the set of all pallets to be unloaded at time interval t, $t = \{1,\ldots,T\}$ at all receiving docks I. As a consequence, we know which pallet is available as an input to the cross dock at any time interval. We note that, by construction, there is a one-to-one correspondance between the time intervals and the sequence number of arriving pallets in S

The known parameters which will be used throughout the article is summarized in Table 1.

Given a known arrival sequence of pallets at the receiving docks at any time interval t, $t = \{1,\ldots,T\}$, the objective of this study is to find the optimum sequence of the set of outbound trucks that should be present at the shipping doors at any time interval t, $t = \{1,\ldots,T\}$ such that the sum of the total inventory holding and the truck replacement costs are minimized. The objective function, here, is a trade-off between the cost of storage and truck replacement (or pre-emption of loading). If we suppress one of these costs, the problem becomes easier. If $r = 0$, we can allow as many preemptions as needed to place the good outbound trucks at each time interval in order to avoid storage. Conversely, if $h = 0$, we can store pallets and charge them later when the outbound truck is present without paying penalties.

The problem described above as well as some of the assumptions are very close to "production cross docks" where the cross docking operations are placed at downstream of the production lines. In this context, the arrival sequence of products is fixed and predetermined by the production plan which takes into account the production constraints such as the resource or component availabilities from suppliers. The products coming out of the lines are handled by a FIFO policy. Since the incoming sequence is imposed by the production plan, we focus only on the outbound scheduling. Finally, if a long sequence of products to a given destination is coming out of the production lines and no trucks are available for this destination at the loading docks, the supervisor can decide to make a replacement to minimize the storage level inside the facility (i.e. truck replacement in our case). The above mentioned industrial context is given in Maknoon and Baptiste (2009) and Larbi et al. (2007). In these studies, the authors present the cross docking operations in a company producing household appliances.

3. An optimal solution based on dynamic programming

In this section we will present a dynamic programming (DP) based model to solve the problem described above. We consider that $D \geqslant O$ since the case with $O > D$ is trivial (see Alpan et al., 2008) and $O \geqslant I$ to ensure that traffic intensity inside the cross dock does not generate infinite stock. Note that, when the cross docks are designed, a common practice is to allocate about $O \geqslant 2I$ (see, Bartholdi & Gue, 2004 & Napolitano, 2000) to control the traffic intensity inside the facility.

Table 1
Input parameters.

Notation	Description
I	Number of receiving doors
O	Number of shipping doors
D	Number of destinations
h	Holding cost
r	Truck replacement cost
S	An input sequence of size T
C	Capacity of the outbound trucks
τ	Unit time of each transshipment operation

3.1. The mathematical model

Let X_t be the set of states at each time interval t, $t = \{1,\ldots,T\}$. We define a state $\bar{x}_t = (t, Z_t, \overline{Y}_t), \bar{x}_t \in X_t$, as a vector, where t is the time interval of the input sequence S, Z_t is the set of outbound trucks (or destinations) present at the shipping door during the time interval t and $\overline{Y}_t$ is a vector of dimension D which keeps track of the quantity of pallets temporarily stored for each destination, $d = \{1,\ldots,D\}$.

The cost incurred at time interval $t+1$ given that the system was in state $\bar{x}_t$ at time interval t and we have decided to assign the set of outbound trucks Z_{t+1} at the shipping docks at time $t+1$ is denoted by $C_{Z_{t+1}}(\bar{x}_t)$ and is calculated by Eq. (1).

$$C_{Z_{t+1}}(\bar{x}_t) = \sum_{i=1}^{D} max\{0, (y_i^{t+1} - y_i^t)\} \times h + \sum_{j=1}^{O} \mathbf{1}_{z_j^{t+1} \notin Z_t} \times r \tag{1}$$

where $\mathbf{1}$ is an indicator function which takes the value of 1 if $z_j^{t+1} \notin Z_t$ and 0 otherwise.

For each pallet in destination to d in the sequence S at time interval $t+1$, we may decide to assign or not an outbound truck to this destination at the shipping docks. If such a truck is already available at the docks at time t, we may keep it at time $t+1$ as well and transfer the pallets from receiving to shipping docks with 0 cost. If on the other hand, no such truck is already available at the shipping docks, we have two possibilities; either we store the pallet (i.e. $y_d^{t+1} = y_d^t + 1$ and incur h), or remove one of the current outbound trucks to affect the corresponding destination, d, to a shipping door (i.e. $\mathbf{1}_{z_j^{t+1} \notin Z_t} = 1$ and incur r).

As seen in Eq. (1), $C_{Z_{t+1}}(\bar{x}_t)$ sums up the total cost of storage for the pallets which are temporarily stored and the total cost of outbound trucks which are replaced during the time interval $t+1$, given that the system was in state $\bar{x}_t$ at time t and the set of outbound trucks Z_{t+1} are assigned at the shipping docks at time $t+1$.

Let $P(\bar{x}_{t+1})$ be the set of all states $\bar{x}_t$, $\bar{x}_t \subset X_t$, which are the predecessors of state $\bar{x}_{t+1}$. Using Eq. (1), the DP model to find the global cost upto time interval $t+1$ is given in Eq. (2).

$$f_{t+1}(\bar{x}_{t+1}) = \min_{\forall \bar{x}_t \in P(\bar{x}_{t+1})} \{C_{Z_{t+1}}(\bar{x}_t) + f_t(\bar{x}_t)\} \tag{2}$$

with $f_0(\bar{x}_0) = 0$.

The recursive function in Eq. (2) generates a graph where the number of nodes increases exponentially. Hence any rule which helps removing nodes without affecting the optimality of the solution can considerably increase the performance of the method. In the next section, we will present some rules and properties which are used to reduce the size of the graph generated by Eq. (2).

3.2. Properties to reduce the size of the DP model

Property 1. *A node n_1 of the graph generated by* Eq. (2) *is dominated by another node n_2 if and only if the following conditions are satisfied.*

1. *n_1 and n_2 have the same time stamp, t.*
2. *n_1 and n_2 have the same set of outbound trucks present at the shipping doors.*
3. *For each destination d, the quantity of the temporarily stored pallets for n_1 is greater than or equal to the quantity of the temporarily stored pallets for n_2.*
4. *Cost generated by n_1 is strictly greater than that of n_2.*

Proof. A node n_i in the generated graph represents a state of the system, i.e. $\bar{x}_t = (t, Z_t, \overline{Y}_t)$. The first two conditions in the property guarantee that the nodes are identical in terms of time interval t and Z_t and hence comparable. The third condition confirms that the possible temporary storage options in n_1 are included in n_2. And since, the cost in n_1 is strictly greater than that of n_2, we conclude that n_1 is dominated by n_2 and will never appear in an optimal solution. □

Property 2. *Let n_1 and n_2 be two nodes in the generated graph which have the same time stamp t. The nodes which succeed n_1 are exactly the same as those succeeding n_2 if and only if the vectors $\overline{Y}_t$ in n_1 and n_2 are identical.*

Proof. Since n_1 and n_2 have the same time interval t, they are comparable. We recall that, a given node n_i in the generated graph at level t represents the state $\bar{x}_t = (t, Z_t, \overline{Y}_t)$. The succeeding nodes of a given n_i are generated as follows:

1. Time is augmented by 1.
2. For a given n_i at time interval $t + 1$, all possible combinations of outbound trucks available at the shipping doors, i.e. Z_{t+1} are generated as a combination of O trucks out of D destinations (i.e. $C(D,O)$).
3. The vector $\overline{Y}_{t+1}$ is then obtained using the set of available trucks at the shipping doors Z_{t+1} and the state of the stock levels at time t, i.e. $\overline{Y}_t$.

By construction, the first and the second parameters of $\bar{x}_{t+1} = (t, Z_{t+1}, \overline{Y}_{t+1})$ will be identical for all succeeding nodes of n_1 and n_2. Since $\overline{Y}_{t+1}$ is calculated using the set Z_{t+1} and $\overline{Y}_t$, the only way to have identical $\bar{x}_{t+1}$ is then possible if and only if $\overline{Y}_t$ are identical for n_1 and n_2. □

Property 3. *Let t be a time instant in the input sequence S of length T and d be a destination for which an outbound truck is present at the shipping door at time t. If a pallet to destination d is present at t + 1 as well, the truck to destination d is kept at the shipping doors during t + 1.*

Proof. If we choose to keep the outbound truck, the product in destination to d at time interval t can be loaded without paying any penalty. If on the other hand, we choose to remove this truck from the shipping door, the product is stored and at least h (and eventually also r if a truck to destination d has to be called up to pick up the merchandize) is paid. Hence it is optimal to keep the truck at the shipping door. □

We note that the above proof applies for infinite T and C (i.e. capacity of the truck). Optimality is not guaranteed when C is finite since there will be other decisions to consider in terms of the remaining capacity of the trucks. But these are mainly borderline effects. In Alpan et al. (2008), a series of experiments have proved that the relative deviation from optimal in case of the bounded truck capacity is negligible. Below, we summarize the reduction rules which are used in the rest of the paper:

1. All dominated nodes are eliminated based on Property 1.
2. All identical successors (i.e. double nodes) are identified in advance based on Property 2 and are eliminated.
3. All nodes violating Property 3 are also eliminated.

An example is given in Appendix A to illustrate the iterations of the DP model as well as the application of properties given above.

3.3. Numerical Results on the performance of the DP model

In this section, we will give some numerical results. The model given in the previous section is tested to observe the impact of the input parameters on the solution obtained. Since the dynamic program gives an optimal solution, the only performance measure taken into consideration here is the execution time of the program. The parameters considered are the length of the input sequence S, the number of destinations D, the number of receiving doors O and the ratio of cost parameters r/h. We will assume the capacity of each truck to be $C = 20$ throughout the experiments.

The results presented in this section are the mean results of 10 different instances when the execution time is reasonable and the mean of three different instances if the execution time is very long. In the input sequence S, the majority of the destinations are assumed to be well represented. The destinations with a few pallets are assumed to be rare. We note that, such a structure for the distribution of destinations is rather common in practice. Since the products arriving a cross dock should leave it in 24 h, the products for rare destinations are not cross docked in practice (Napolitano, 2000). Nevertheless, the interested reader can refer to Alpan et al. (2008) where other structures of input sequence S have been tested. The distribution of destinations will be represented by the probability of appearances. For instance, for $D = 4$, distribution (1,29,34,36) means that destination 1 will appear in the sequence with a probability 0.01 while destination 2 will appear with a probability 0.29 and so on. The input sequence is generated as follows: Given destinations distribution, for each inbound dock, the destination of an arriving pallet will randomly be assigned using the given distribution function. The instance generator is down-loadable from http://www.g-scop.fr/∼gaujalg. Destination distributions used are given in Table 2.

The model is programmed using JAVA and the tests are run on a 1.6 GHz and 512Mo RAM Personal Computer.

3.3.1. Tests on the length of the input sequence

We make the hypothesis that an acceptable execution time will be at most 15 min. Hence we varied T such that the execution time stays within the 15 min range. The other input parameters are $I = 4$, $O = 7$ and $D = 8$. Fig. 1 displays the evolution of the execution time of the program versus the length T of the input sequence S. Each point on the curve corresponds to the mean of 10 different test runs. As seen in Fig. 1, the critical time of 15 min is exceeded when T is almost 570. We also observe that the execution time augments exponentially as T is increased.

This phenomenon is explained as follows: for a given instance t in the sequence, if a pallet to be unloaded does not have its corresponding outbound truck at the shipping door, we should decide either to store the pallet temporarily or to affect the corresponding truck at the shipping door. To find the best decision, we often have to go in depth in the sequence to check the possible arrivals for the same destination. Therefore, the longer the sequence is the search space in the sequence is longer and the number of nodes to be

Table 2
Distribution of destinations.

D	Distribution of destinations (in %)
4	(1,29,34,36)
5	(1,15,25,28,31)
6	(1,14,19,21,22,23)
7	(2,7,10,17,20,21,23)
8	(2,7,12,14,15,16,17,17)
9	(2,4,11,12,12,13,14,16,16)

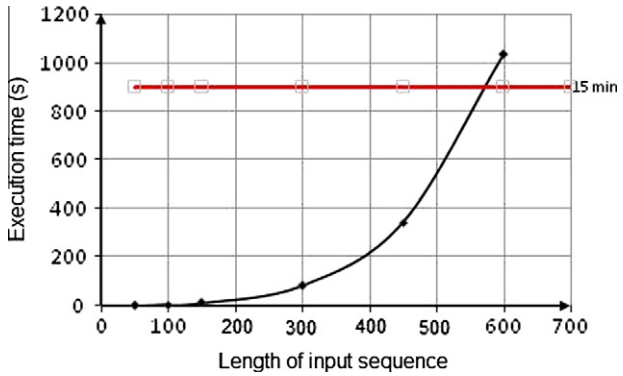

Fig. 1. Evolution of the execution time by the length of the sequences.

generated is considerably higher. For instance for $T = 570$ where the 15 min limit is attained, the total number of nodes generated is around 360,000 whereas for $T = 280$, this number is divided by 3.

3.3.2. Tests on the number of destinations

The test runs are done for $I = 1$, $O = 2$, $T = 50$ and $4 \leqslant D \leqslant 9$. The curves in Figs. 2 and 3 give the execution time and the number of generated nodes versus the number of destinations, respectively. As illustrated in Fig. 2, the execution time is reasonable until $D = 8$ and for $D > 8$ we have an exponential behavior for the execution time. This is explained by the curve in Fig. 3 where we observe an important augmentation in the number of nodes generated.

3.3.3. Tests on the number of receiving doors

The results given in Figs. 4 and 5 are obtained for input parameters $O = 9$, $D = 10$, $T = 100$ and the number of receiving doors varying between 1 and 7. The program is tested for $I = 8$ as well but did not give any solution after a very long execution period. We believe that this occurs because the number of receiving doors I is very close to the number of shipping doors O and hence the system is close to saturation. The input traffic of pallets is barely absorbed by the outgoing trucks and hence finding an optimal solution gets difficult as I approaches O.

3.3.4. Tests on the number of shipping doors

We tested the effects of the number of the shipping doors on the execution time for $I = 1$, $D = 9$ and $T = 50$. Table 3 gives the results obtained.

For 3, 4, 5 and 6 shipping doors the execution time is exploded. The major reason is that, for each node in the graph, the number of

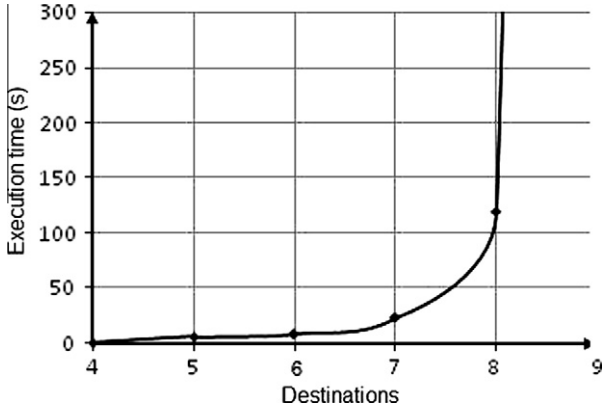

Fig. 2. Evolution of the execution time with respect to the number of destinations.

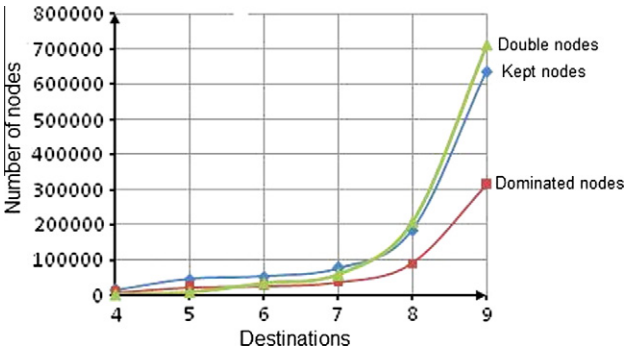

Fig. 3. Number of the nodes generated with respect to the number of destinations.

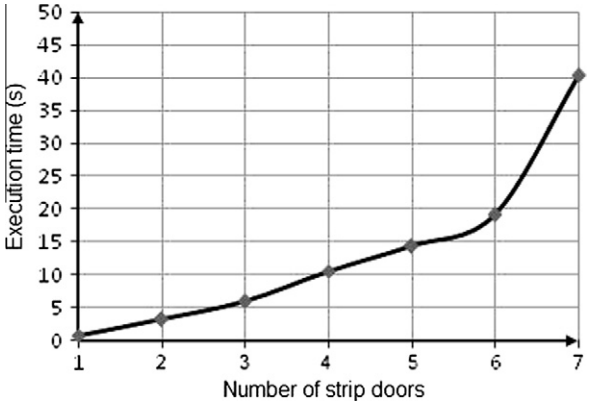

Fig. 4. Execution time as a function of number of receiving doors.

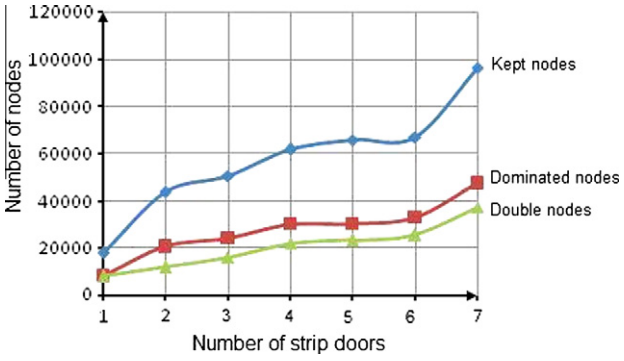

Fig. 5. Number of nodes generated as a function of number of receiving doors.

succeeding nodes (denoted NXT-S in the tables) varies between 84 and 126. Hence the required number of calculations is augmented.

We also observe that $O = 2$ and $O = 7$, the number of succeeding nodes are equivalent however the solution for $O = 7$ is obtained much faster than the solution for $O = 2$. The reason is that, it is much easier to affect nine destinations to seven shipping doors than to affect them to two shipping doors. These results show that the closer the number of shipping doors to the number of destinations, an optimal solution can be found faster. A cross docking platform is hence efficient if $|D - O|$ is low.

3.3.5. Tests on the ratio of cost parameters r/h

Finally, we tested the effects of the ratio of cost parameters r/h. The fixed parameters are $I = 1$, $O = 2$, $D = 4$ and $T = 50$. The results obtained are presented in Fig. 6.

Table 3
Effects of the number of shipping doors on the execution time (in seconds).

O	NXT-S	Time (s)
2	36	5363.772
3	84	>10,000
4	126	>10,000
5	126	>10,000
6	84	>10,000
7	36	8.485
8	9	0.481

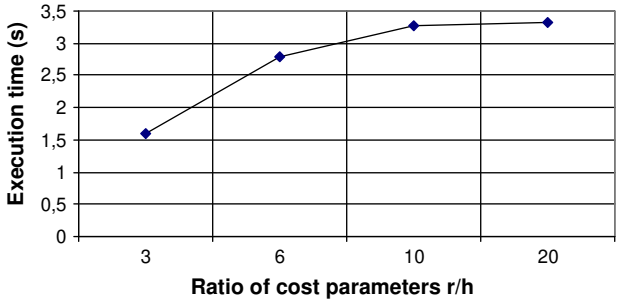

Fig. 6. Evolution of the execution time with respect to the ratio of cost parameters.

We observe that the execution time increases upto a point when the ratio r/h increases and than remains almost constant. This is most probably due to the fact that, as r gets much higher than h, we would prefer storing incoming products than replacing a truck. Increasing the number of units stored in the facility means that new nodes will be generated in the graph to keep track of these new levels of stock units. On the other hand, the computational complexity related to the replacement of the truck (i.e. $C(D,O)$) stays the same for all r since D and O are fixed. Therefore, when a certain limit of storage is attained (i.e. all possible combinations of storages are created), the execution time will reach to stability even if r is further increased.

Table 4 gives a summary of the limiting results obtained throughout the test runs. We observe that the optimal solution procedure presented above can handle a cross docking facility with about ten docks and nine destinations. In the next section, we will explore how bounding the DP model in terms of number of nodes generated at each stage can impact the performance of the model.

4. Bounding strategies on dynamic programming model

As we have seen via the numerical experiments presented in the previous section, the DP model fails to handle scheduling problems in the case of large cross docking systems (i.e.high number of destinations, products, input and output docks). This is basically due to the exponential increase in the number of nodes generated as the value of input parameters to the problem increases. In this section, we will present two strategies to limit the number of nodes generated at each stage of the DP model. If these bounds are efficient to remove unnecessary nodes (i.e. the ones which are not on the optimal path) then we can hope to increase the computational performance of the model while limiting the deviation from the optimal.

Table 4
A summary of limiting input parameters for reasonable execution times.

I	O	D	T	Time (s)
1	2	3	5000	1383.106
3	7	8	1000	1983.282
4	7	8	600	1034.437
2	4	6	50	4679.657
1	2	9	50	5363.772

4.1. Bounds on the storage allowed at each time period, t

As we have seen in Section 3 a state $\bar{x}_t$ in the original DP is represented by three components, namely the time interval t, set of outbound destinations Z_t and the amount of products already stored, $\overline{Y}_t$. Among these three components, especially Z_t and $\overline{Y}_t$, contribute to the complexity of the problem. Here, we will put a special emphasis on the storage since we believe that bounding the number of units which can be stored in the cross docking facility will not only have a positive effect on the performance of the DP model, but it may also improve the internal performance of the facility. For instance, smaller stocks in the facility will diminish the level of congestion inside the facility and hence augment the speed of the product flow inside the cross dock, which in turn will increase the productivity of the facility.

Therefore, in this section we will first present an upper bound on the level of inventory that can be accumulated in the cross docking facility, denoted S_{max} (see Eq. (3)). In Eq. (3), T, I, O are defined as before.

$$S_{max} = \frac{T(O-I)I}{O} \tag{3}$$

Eq. (3) is calculated similar to a classical EOQ model with finite production rate (see for instance, Silver, Pyke, & Peterson, 1998). Since we are looking for an upper bound, we make the assumption that at the beginning wrong decisions are made concerning the destinations to be assigned on the outbound docks, thus increasing inventory level with a rate of I upto S_{max}. Then, the inventory is progressively absorbed with a rate of $O-I$ since all products are to be removed at the end of the period T and the workforce capacity is infinite (see assumption $A4$). The algorithm below summarizes how we bound the original DP to reduce the number of nodes generated in the underlying graph.

Algorithm 1. : DP bounded on the level of stock

Let α be the percentage of S_{max} which will be allowed at the cross docking facility. Repeat the following steps for each node generated by the original DP.

1: At a given stage t, generate all nodes (or equivalently states $\bar{x}_t \in X_t$ as in the original DP).
2: If $\sum_{d=1}^{d=D} y_d > \alpha \times S_{max}$ where y_d represents each cella of the vector $\overline{Y}_t$, then remove this node from the graph.

4.2. Bounds on the number of states generated at each time period, t

In this section, we will propose to bound the DP model by imposing a limit on the number of nodes generated at each stage based on the cost incurred at each node. To this end, we first calculated a theoretical upper bound on the number of nodes generated at each step, denoted $UB(nodes)$ (see Eq. (4))

$$UB(nodes) = C(D,O) \times \sum_{i=0}^{S_{max}} C(i+D-1,i) \tag{4}$$

In Eq. (4), the first term on the right hand side corresponds to all possible configurations of destinations which can be present at the outbound docks. Similarly, the term $C(i+D-1,i)$ corresponds to all possible configurations of stocks which can be accumulated for each destination d for a given level of stock i. We note that the term $C(i+D-1,i)$ is originally proposed in Bruel and Balbo (1980)

to calculate the number of possible states of a closed queuing network composed of D single server queues and i customers. This is analogous to finding the number of states where the stock for each destination varies from 0 to i with the constraint that the sum of the stock for all destinations is equal to i. Finally, since the level of stock i may itself vary (i.e. $i = \{0,\ldots,S_{max}\}$), we sum over all possible storage levels. Algorithm 2 summarizes the bounding procedure based on this upper bound.

Algorithm 2. : DP bounded on the number of nodes

Let β be the percentage of nodes which will be kept at each stage. Repeat the following steps for each stage t of the original DP, $t = \{0,\ldots,T\}$.

1: For a new stage $t+1$, generate all the nodes which stem from stage t as in the original DP.
2: For each new node in stage $t+1$ calculate the global cost as in Eq. (2).
3: At stage $t+1$, keep only β percent of the nodes having the least global cost.

5. Numerical results on the performance of bounds

In this section we will present the results of some numerical experiments carried out to evaluate the performance of bounding procedures. Different experiments are generated by varying the number of inbound doors I, number of outbound doors O and the number of destinations D. The experiments are performed on small size problems in order to test the quality of the solutions in comparison to the optimal solution given by the DP model. For each experiment, 10 instances are generated and the results are given as an average of the results obtained for each instance. We considered the unit storage cost $h = 1$ and the truck replacement cost $r = 10$.

The algorithms are programmed in Java and the tests are performed using a Pentium 4 (1.6 GHz, 512 RAM) computer as is the case for the DP model.

The results are given in the following tables. The first column gives the name of the method used. The second column gives the total number of nodes generated using the method. The third column represents the execution times (in seconds) and finally the last column gives the percent difference of total cost obtained by the bounding procedure and the optimal or absence of some destinations at the outbound dock. $H^{\alpha=0.1}_{stock}$, $H^{\alpha=0.2}_{stock}$, $H^{\alpha=0.3}_{stock}$, $H^{\alpha=0.4}_{stock}$ gives the experimental results for Algorithm 1 where the storage capacity is respectively limited to 10%, 20%, 30% and 40% of S_{max} given in Eq. (3). $H^{\beta=0.7+0.5i}_{nodes}$ with $i \in \{0,1,2,3,4,5,6\}$ corresponds to results for Algorithm 2 where the number of nodes is bounded by $\beta = 0.7 + 0.5i$.

5.1. Experiments for variable D

In this section, we will present the results when I, O and T are fixed at 1, 2 and 50, respectively, and the number of destinations, D is varied. (see Tables 5–7).

We first observe that, the results of H_{stock} is more sensitive to the changes in the parameter α than H^{β}_{nodes} is for parameter β for the ranges chosen in these experiments: The number of nodes removed do not change a lot when β varies between 0.7 and 0.95. On the other hand the performance of H_{stock} is very much affected by the value of α.

For bounding procedure H_{stock}, when the storage space is highly restricted (i.e. the case where $\alpha = 0.1$), the percent difference of H_{stock} with the optimal solution is in the order of 30% with very low execution times (only a few seconds). The improvement in execution times is especially appreciated for higher number of destinations. For instance, the execution time of $H^{\alpha=0.1}_{stock}$ is 25 times lower than that of the optimal solution when $D = 4$. It is in the order of 854 times for $D = 5$ and increases to 2343 times for $D = 6$ while the quality of the solution stays within the limit of 30%.

We also observe that as the storage capacity augments the solution gets closer to the optimal solution, however the execution times augment significantly as well. For the case where $\alpha = 0.4$ the solution space is almost the same as the DP model, but the execution times are even worse, since the Algorithm 1 requires some execution time in addition to the generation of the graph underlying the DP model.

For the second bounding procedure H^{β}_{nodes}, the difference of the solution with the optimal is inferior to 5%. We can observe that bounding the number of nodes to $\beta = 0.7$ reduces the execution

Table 5
Results for I, O, D, $T = (1, 2, 4, 50)$.

	Nb nodes gen.	Exec. times (s)	% diff. to optimal
DP	39,946	4.2963	
$H^{\alpha=0.1}_{stock}$	4576.1	0.071	18.63
$H^{\alpha=0.2}_{stock}$	20,878	1.03	2.5
$H^{\alpha=0.3}_{stock}$	37564.3	3.69	0
$H^{\alpha=0.4}_{stock}$	39,926	4.26	0
$H^{\beta=0.7}_{nodes}$	27692.8	2.2	3.87
$H^{\beta=0.75}_{nodes}$	30070.1	2.54	1.93
$H^{\beta=0.8}_{nodes}$	31174.9	2.77	1.93
$H^{\beta=0.85}_{nodes}$	32598.4	3.07	0.78
$H^{\beta=0.9}_{nodes}$	33599.2	3.22	0.6
$H^{\beta=0.95}_{nodes}$	34963.3	3.55	0.6

Table 6
Results for I, O, D, $T = (1, 2, 5, 50)$.

	Nb nodes gen.	Exec. times (s)	% diff. to optimal
DP	342611.7	445.52	
$H^{\alpha=0.1}_{stock}$	14443.6	0.52	28.29
$H^{\alpha=0.2}_{stock}$	109057.7	27.56	9.49
$H^{\alpha=0.3}_{stock}$	282524.8	264.22	3.83
$H^{\alpha=0.4}_{stock}$	334468.4	398.96	3.69
$H^{\beta=0.7}_{nodes}$	214468.5	182.48	3.58
$H^{\beta=0.75}_{nodes}$	229,831	185.13	3.92
$H^{\beta=0.8}_{nodes}$	232981.9	196.93	3.13
$H^{\beta=0.85}_{nodes}$	243599.3	211.67	2.81
$H^{\beta=0.9}_{nodes}$	248379.2	225.76	1.57
$H^{\beta=0.95}_{nodes}$	252048.1	234.10	0.7

Table 7
Results for I, O, D, $T = (1, 2, 6, 50)$.

	Nb nodes gen.	Exec. times (s)	% diff. to optimal
DP	1093904.8	4998.24	
$H^{\alpha=0.1}_{stock}$	32591.2	2.13	30.5
$H^{\alpha=0.2}_{stock}$	352422.2	346.16	4.72
$H^{\alpha=0.3}_{stock}$	950888.2	3548.95	1.55
$H^{\alpha=0.4}_{stock}$	1085131.2	4909.53	0.36
$H^{\beta=0.7}_{nodes}$	896934.6	3531.78	4.83
$H^{\beta=0.75}_{nodes}$	901099.4	3630.62	4.83
$H^{\beta=0.8}_{nodes}$	908,263	3705.39	1.19
$H^{\beta=0.85}_{nodes}$	927,230	3758.75	1.19
$H^{\beta=0.9}_{nodes}$	938,175	3817.03	1.19
$H^{\beta=0.95}_{nodes}$	942,595	3861.32	1.19

time to 70% of the execution time of DP. When β augments, the solution is closer to the optimal but the execution time augments.

However it seems like H_{stock} is better than H^{β}_{nodes} for the cases where the number of destinations is high. In Table 7, we observe that H_{stock} for $\alpha = 0.2$, the quality of the solution is similar to $H^{\beta=0.7}_{nodes}$, however, the execution time is 10 times better than that of $H^{\beta=0.7}_{nodes}$. In general, H_{stock} seems to perform better than H^{β}_{nodes} for $0.2 \leqslant \alpha \leqslant 0.3$ if we admit a maximum of 5% deviation from optimal.

5.2. Experiments for variable O

In this section we have tested the two bounding procedures and the DP when *I*, *D* and *T* are fixed at 1, 6 and 50, respectively, and the number of outbound doors, *O* is varied. The results are given in Tables 7–9.

We can observe that for the two bounding procedures the difference to optimal solution decreases when the difference between the number of outbound doors *O* and the number of destinations *D* decreases. Similar to the previous tests, the quality of the solutions augments as α and β are increased, but the execution times get worse. We note that, the input data is already very easy for the DP model in the case of Table 9. Hence, it is preferable to use directly the DP model which is sure to give the optimum solution. However, the interest of bounding procedures are especially appreciated for difficult instances such as the one given in Table 8. Here we observe that the execution time can be divided by 2 if we accept a solution 0.73% worse than the optimal one (case of $H^{\beta=0.73}_{nodes}$) and by 12 if we accept a solution 2.7% worse than the optimal one (case of $H^{\alpha=0.2}_{stock}$).

Table 8
Results for *I*, *O*, *D*, *T* = (1, 3, 6, 50).

	Nb nodes gen.	Exec. times (s)	% diff. to optimal
DP	721097.5	1482.8	
$H^{\alpha=0.1}_{stock}$	32077.2	1.4	31.91
$H^{\alpha=0.2}_{stock}$	258826.7	119.6	2.7
$H^{\alpha=0.3}_{stock}$	634467.7	1070.9	0
$H^{\alpha=0.4}_{stock}$	709198.7	1400.9	0
$H^{\beta=0.7}_{nodes}$	503,070	692.4	0.73
$H^{\beta=0.75}_{nodes}$	507146.5	711	0.73
$H^{\beta=0.8}_{nodes}$	514943.7	722.2	0.73
$H^{\beta=0.85}_{nodes}$	515951.2	708.8	0.73
$H^{\beta=0.9}_{nodes}$	519652.2	743	0
$H^{\beta=0.95}_{nodes}$	530,659	778	0

Table 9
Results for *I*, *O*, *D*, *T* = (1, 4, 6, 50).

	Nb nodes gen.	Exec. times (s)	% diff. to optimal
DP	74830.2	8.97	
$H^{\alpha=0.1}_{stock}$	12225.7	0.24	46.33
$H^{\alpha=0.2}_{stock}$	53035.2	3.83	0
$H^{\alpha=0.3}_{stock}$	74,473	8.81	0
$H^{\alpha=0.4}_{stock}$	74830.2	8.82	0
$H^{\beta=0.7}_{nodes}$	60603.2	5.63	0
$H^{\beta=0.75}_{nodes}$	60803.5	5.54	0
$H^{\beta=0.8}_{nodes}$	60915.2	5.62	0
$H^{\beta=0.85}_{nodes}$	60938.2	5.68	0
$H^{\beta=0.9}_{nodes}$	61378.7	5.68	0
$H^{\beta=0.95}_{nodes}$	62285.7	6.07	0

Table 10
Results for *I*, *O*, *D*, *T* = (2, 4, 6, 50).

	Nb nodes gen.	Exec. times (s)	% diff. to optimal
DP	465526.8	1200	
$H^{\alpha=0.1}_{stock}$	14334.3	0.72	30.54
$H^{\alpha=0.2}_{stock}$	122052.1	47.11	4.46
$H^{\alpha=0.3}_{stock}$	377,324	707.98	1.25
$H^{\alpha=0.4}_{stock}$	449885.1	1073.8	1.54
$H^{\beta=0.7}_{nodes}$	418374.1	910.92	4.76
$H^{\beta=0.75}_{nodes}$	418654.8	917.76	2.38
$H^{\beta=0.8}_{nodes}$	428819.3	970.2	0
$H^{\beta=0.85}_{nodes}$	434,570	1006.9	0
$H^{\beta=0.9}_{nodes}$	448918.8	1089.2	0
$H^{\beta=0.95}_{nodes}$	449916.6	1106	0

Table 11
Results for *I*, *O*, *D*, *T* = (3, 4, 6, 50).

	Nb nodes gen.	Exec. times (s)	% diff. to optimal
DP	780573.5	1929.5	
$H^{\alpha=0.1}_{stock}$	17608.5	0.75	35.94
$H^{\alpha=0.2}_{stock}$	189,263	65.32	15.01
$H^{\alpha=0.3}_{stock}$	587,252	962.1	0
$H^{\alpha=0.4}_{stock}$	742,913	1700.82	0
$H^{\beta=0.7}_{nodes}$	663,215	1318.23	0
$H^{\beta=0.75}_{nodes}$	665,497	1315.04	0
$H^{\beta=0.8}_{nodes}$	702,851	1500.45	0
$H^{\beta=0.85}_{nodes}$	735,555	1669.89	0
$H^{\beta=0.9}_{nodes}$	749,383	1779.79	0
$H^{\beta=0.95}_{nodes}$	763,153	1779.8	0

5.3. Experiments for variable I

In this section we have tested the two bounding procedures and the DP when *O*, *D* and *T* are fixed at 4, 6 and 50, respectively, and the number of inbound doors, *I* is varied. The results are given in Tables 9–11.

For these series of tests, we make the same observations as before: the solutions are closer to the optimal when α and β increase but the execution times suffer. Similarly, for similar performance in terms of cost, H_{stock} seems to perform better than H_{nodes} (see for $\alpha = 0.2$ and $\beta = 0.7$ in Table 10).

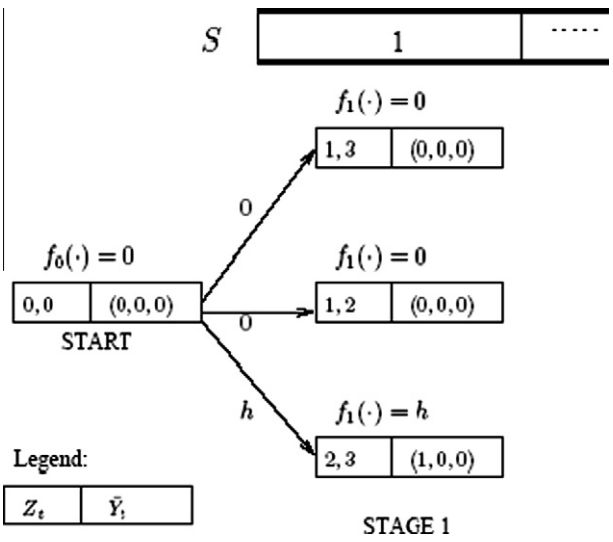

Fig. A.7. First iteration of the DP model.

6. Conclusion

In this article, we proposed a bounded dynamic programming approach to schedule the internal operations in a cross docking facility. The contributions are two-fold. First of all, using dynamic programming technique, we propose an optimal truck schedule to minimize the cost of operations in a multiple-dock cross docking facility. Secondly, we have seen through experimental results that

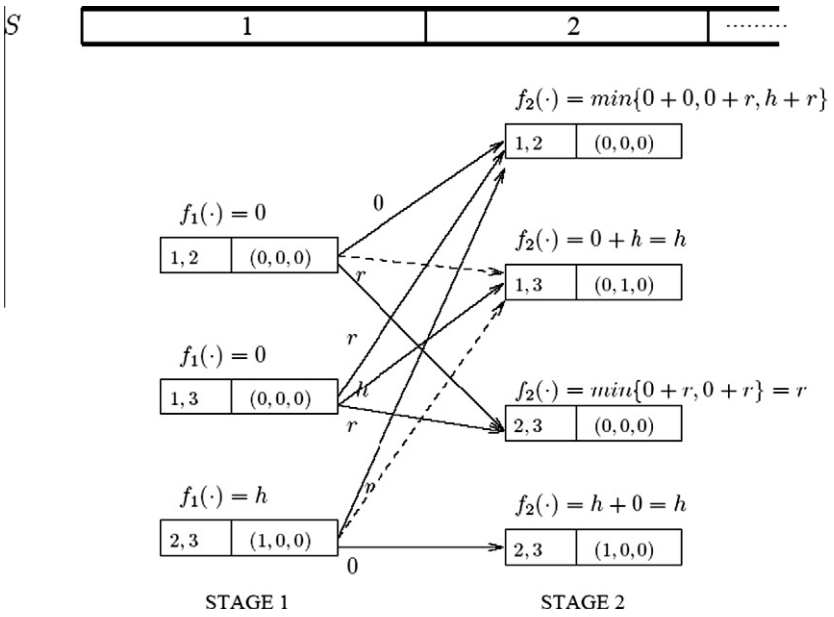

Fig. A.8. Second iteration of the DP model.

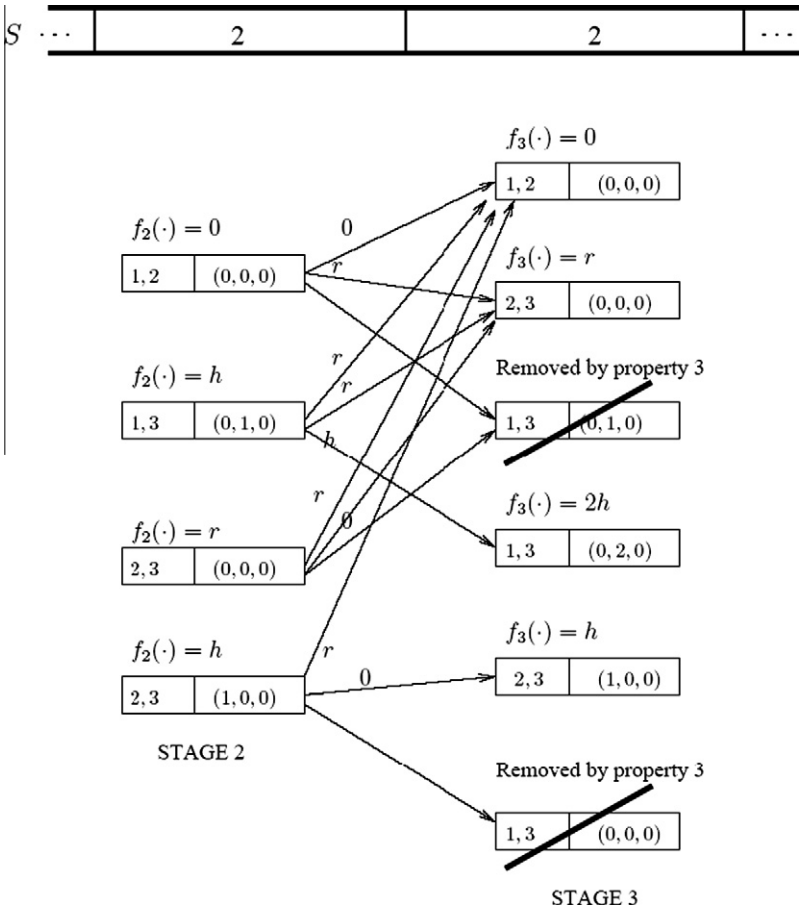

Fig. A.9. Third iteration of the DP model.

by generating intelligent bounds, we can solve scheduling problems much faster than a classical DP model without much degrading the solution. This opens up new possibilities for scheduling of internal operations in larger cross docking facilities.

The numerical results on bounds have shown that the parameters of bounding procedures affects considerably the final solution. For instance, in the case of H_{stock}, the procedure seems to be well performing when $0.2 \leqslant \alpha \leqslant 0.3$. On the other hand for H^{β}_{nodes}, $\beta \geqslant 0.7$ seems to be not very restrictive for certain cases. Unfortunately, it is difficult to provide a general procedure or closed-formula which give the best or most reasonable parameter values. Indeed, α and β are very much dependent on the input data. We believe that these parameters can be included in a long term decision process. Usually, the global flow of materials (e.g. number of pallets per hour or day) which can be handled and/or usually handled are known. Based on historical data, the supervisor can use our method, to observe a window of best performing α and β values specific for their platform. These parameters can then be used on a real-time basis.

New bounding procedures can also be tested. As we have explained in Section 4.1, set of outbound destinations Z_t contributes to the complexity of the problem as well. We may impose, for instance, f destinations to be already present in the set Z_t depending on the density of products in a portion of the input sequence. Such a procedure, will put a bound on the number of nodes generated since the possible combinations for Z_t will be $C(D-f, O-f)$, instead of $C(D,O)$. Similarly, hybrid bounds may be tested such as $H_{stock} + H^{\beta}_{nodes}$.

Finally, in this study, we have considered that input sequence is fixed. It will be interesting to allow a flexible input sequence. That is, the supervisor has the possibility to reschedule the inbound trucks. To this end, the techniques developed in this paper can be combined with some metaheuristics which modify the input sequence. We believe that, this may considerably reduce the solution space generated by the DP model.

Appendix A

In this appendix, we give an example to illustrate the DP model presented in Section 3. We consider a cross docking facility with $I = 1$, $O = 2$, $D = 3$, $r = 2$, $h = 1.5$ and an input sequence of $S = (1,2,2,3,1)$. This means that, at time $t = 1$, a pallet for destination 1 will be unloaded at the unique receiving dock. At time $t = 2$ a pallet for destination 2 will be unloaded, and so on. This sequence will be displayed on top of the generated graph to illustrate which destination is present at the receiving dock at a given stage.

The objective is to find which trucks should be present at the two shipping docks during the periods 1–5 such that the total inventory holding and truck replacement costs are minimized. In the following figures, each state $\bar{x}_t = (t, Z_t, \overline{Y}_t)$ in a given stage t is represented by a double compartment rectangle. The left compartment gives the set of trucks assigned at a shipping dock at time t, i.e. Z_t, and the right compartment gives the vector of destinations in which we keep track of the storage level for each destination. As seen in Fig. A.7, initially no trucks (destinations) are assigned on the two shipping docks and the storage area is empty and a total cost of $f_0 = 0$ is incurred initially. Then we start the DP recursion. We first calculate all possible combinations of truck (destination) assignments: $Z_1 = \{1,2\}$ or $Z_1 = \{1,3\}$ or $Z_1 = \{2,3\}$. For $Z_1 = \{1,2\}$ or $Z_1 = \{1,3\}$, no pallets are stored, since the arriving pallet to destination 1 can directly be loaded on the truck to destination 1 which is assigned to one of the available docks. Consequently, the storage

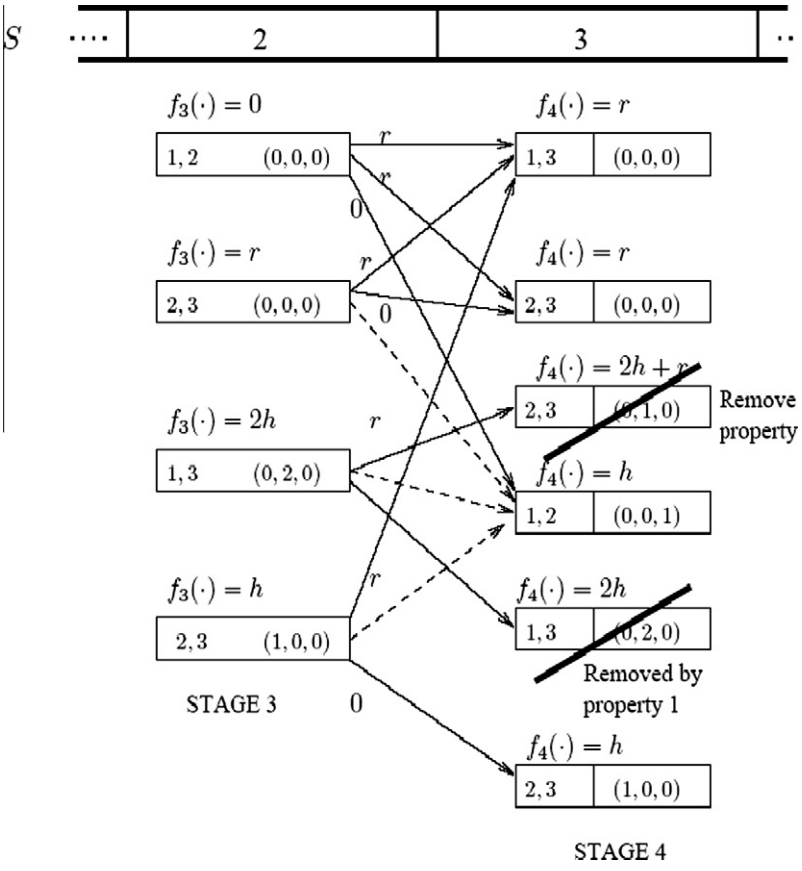

Fig. A.10. Fourth iteration of the DP model.

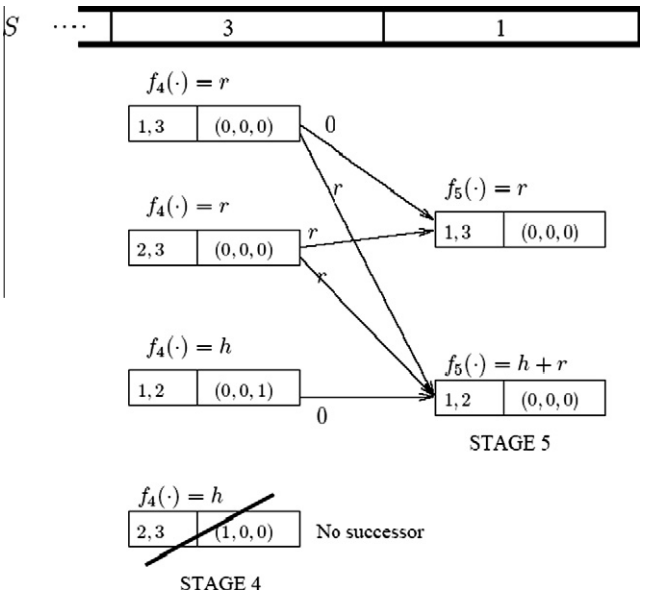

Fig. A.11. Fifth iteration of the DP model.

levels are all 0 for all destinations and no cost is incurred. If the assignment $Z_1 = \{2,3\}$ is made, on the other hand, the unloaded pallet to destination 1 will be temporarily stored (i.e. $\overline{Y}_1 = (1,0,0)$) and a cost of h is incurred. Since each state in stage 1 has a single predecessor with a cost of $f_0 = 0$, the total cost for each state at this stage will be equal to the costs evaluated on the arcs.

Fig. A.8 illustrates the second iteration of the DP model. First, we generate all possible succeeding states for every state in stage 1. Let's take state $\bar{x}_1 = (Z_1, \overline{Y}_1) = (\{1,3\}, (0,0,0))$. From this state, we can generate three succeeding states with the same possible combinations of truck (destination) assignments that we have observed in Fig. A.7: $Z_2 = \{1,2\}$, $Z_2 = \{1,3\}$, $Z_2 = \{2,3\}$. Similar to first iteration, we update the storage levels for each destination given these assignments. For $Z_2 = \{1,2\}$ and $Z_2 = \{2,3\}$, $\overline{Y}_2 = (0,0,0)$. This means that no pallets are stored. The pallet to destination 2 arriving at $t = 2$ can directly be loaded on the truck to destination 2 which is available at the shipping docks. On the other hand, for $Z_2 = \{1,3\}$ this arriving pallet will be stored temporarily since no trucks to destination 2 is available at the shipping docks at time $t = 2$. The arc from $\bar{x}_1 = (\{1,3\}, (0,0,0))$ to $\bar{x}_2 = (\{1,2\}, (0,0,0))$ respectively, $\bar{x}_2 = (\{2,3\}, (0,0,0))$ costs r since truck to destination 3 (resp. 1) is replaced by a truck to destination 2. The arc from $\bar{x}_1 = (\{1,3\}, (0,0,0))$ to $\bar{x}_2 = (\{1,3\}, (0,1,0))$ is evaluated by h, since a holding cost is paid for the stored pallet. Same procedure is followed for all $\bar{x}_1$ to find the succeeding states and the costs on the arcs to these succeeding states. By Property 2, state $\bar{x}_1 = (\{1,2\}, (0,0,0))$ will have the same potential succeeding states as $\bar{x}_1 = (\{1,3\}, (0,0,0))$. Finally by Property 3, the arcs from $\bar{x}_1 = (\{1,2\}, (0,0,0))$ to $\bar{x}_2 = (1,3,(0,1,0))$ (similarly, from $\bar{x}_1 = (\{2,3\}, (1,0,0))$ to $\bar{x}_2 = (\{1,3\}, (0,1,0))$) will not be generated. This is illustrated by dashed lines in Fig. A.8. Finally, the total cost from the beginning upto a given state in stage 2 is calculated using Eq. (2). Before proceeding to the next stage, we verify if a state can be eliminated by Property 1. State $\bar{x}_2 = (\{2,3\}, (1,0,0))$ is a potential candidate compared to $\bar{x}_2 = (\{2,3\}, (0,0,0))$ since it accumulates a higher number of pallets in the storage. However, we cannot eliminate it since the cost incurred at this stage is lower than $f_2(\{2,3\}, (0,0,0))$.

The third iteration is illustrated by Fig. A.9. Here we note that the dashed arcs will not be generated by Property 3. That is, we are keeping the truck to destination 2. As a result, the states $\bar{x}_3 = (\{1,3\}, (0,0,0))$ and $\bar{x}_3 = (\{1,3\}, (0,0,0))$ are removed from the graph.

Fig. A.10 illustrates the fourth stage operations. Here, we remark that state $\bar{x}_4 = (\{2,3\}, (0,1,0))$ is dominated by $\bar{x}_4 = (\{2,3\}, (0,0,0))$: it has more pallets in storage and the total cost is higher. By Property 1, it can be removed from the graph. Same argument is valid for states $\bar{x}_4 = (\{1,3\}, (0,2,0))$ and $\bar{x}_4 = (\{1,3\}, (0,0,0))$. The former is dominated by the latter and is removed by Property 1.

The final iteration is given in Fig. A.11. We note that, only the states for which the storage level is equal to 0 for all destinations are considered at the fifth and final iteration of the DP model (by assumption *A*7). We observe that $\bar{x}_4 = (\{2,3\}, (1,0,0))$ has no succeeding nodes. We cannot connect this state to states $\bar{x}_5 = (\{1,3\}, (0,0,0))$ and $\bar{x}_5 = (\{1,2\}, (0,0,0))$ since the arriving pallet to destination 1 has priority on the already stored one (assumption *A*5). We cannot transfer pallet 1 from storage to the truck at the same time as the pallet 1 from inbound to outbound truck. Hence this node will be removed from the graph.

Finally, the overall graph generated by the DP model is illustrated in Fig. A.12. Backtracking the path with the minimum cost gives the optimal solution (highlighted by bold arrows in Fig. A.12). As we can observe at the bottom of the figure, the best schedule of trucks is assigning destination 1 to one of the

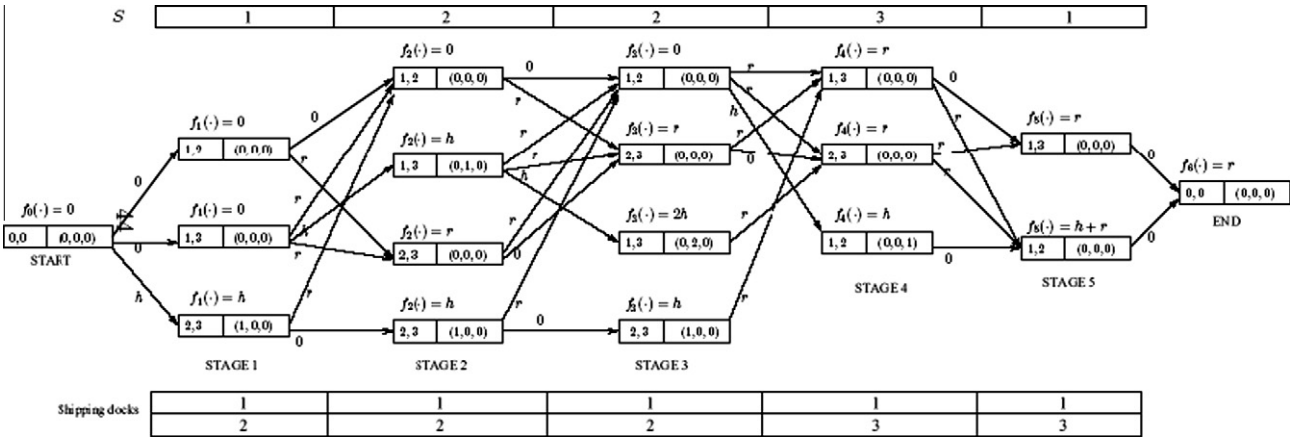

Fig. A.12. The generated graph and the optimal sequence of outbound trucks.

shipping docks throughout the whole scheduling period. The second dock can then be used by a truck to destination 2 during the periods 1, 2 and 3. Then, this truck is replaced by a truck to destination 3 at time $t = 4$. The total cost of this schedule is equal to $r = 2$.

References

Alpan, G., Bauchau, S., Larbi, R., & Penz, B. (2008), Optimal operations scheduling in a crossdock with multi strip and multi stack doors. In *38th International conference on computers and industrial engineering – CIE38* (Vol. 2, pp. 1168–1176). Beijing, China.

Baptiste, P., & Maknoon, M. Y. (2007), Cross-docking: Scheduling of incoming and outgoing semi-trailers. In *International conference on production research – ICPR'07*. Valparaiso, Chile.

Baptiste, P., Penz, B., & Larbi, R. (2007), Polynomial time solution methods for some cross-docking problems. In *International conference on industrial engineering and systems management – IESM'07*. Beijing, China.

Bartholdi, J. J., & Gue, K. R. (2001), Staging freight in a crossdock. In *International conference on industrial engineering and production management – IEPM'01*. Québec City, Canada.

Bartholdi, J. J., & Gue, K. R. (2002). Reducing labor costs in an LTL crossdocking terminal. *Operations Research, 48*(6), 823–832.

Bartholdi, J. J., & Gue, K. R. (2004). The best shape for a crossdock. *Transportation Science, 38*(2), 235–244.

Boysen, N. (2009). Truck scheduling at zero-inventory cross docking terminals. *Computers and Operations Research, 37*(1), 32–41.

Boysen, N., Fliedner, M., & Scholl, A. (2008). Scheduling inbound and outbound trucks at cross docking terminals. *OR Spectrum, 32*(1), 135–161.

Bruel, S., & Balbo, G. (1980). *Computational algorithms for closed queueing networks*. North Holland.

Chen, P., Guo, Y., Lim, A., & Rodrigues, B. (2006). , Multiple crossdocks with inventory and time windows. *Computers and Operations Research, 33*, 43–46.

Chen, F., & Lee, C.-Y. (2009). , Minimizing the makespan in two-machine cross-docking flow shop problem. *European Journal of Operational Research, 193*(1), 59–72.

Chen, F., & Song, K. (2009). Minimizing makespan in two-stage hybrid cross docking scheduling problem. *Computers and Operations Research, 36*, 2066–2073.

Donaldson, H., Jonhson, E. L., Ratliff, H. D., & Zhang, M. (1998), Schedule-driven cross-docking network. Technical report, Georgia Institute of Technology.

Gue, K. R. (1999). The effects of trailer scheduling on the layout of freight terminals. *Transportation Science, 33*(4), 419–428.

Larbi, R., Alpan, G., Baptiste, P., & Penz, B. (2007), Scheduling of transshipment operations in a single strip and stack doors crossdock. In *International conference on production research – ICPR'07*. Valparaiso, Chile.

Larbi, R., Alpan, G., & Penz, B. (2009), Scheduling transshipment operations in a multiple inbound and outbound door crossdock. In *39th International conference on computers and industrial engineering – CIE39*. Troyes, France.

Maknoon, M. Y., & Baptiste, P. (2009). Cross-docking: increasing platform efficiency by sequencing incoming and outgoing semi-trailers. *International Journal of Logistics Research and Applications, 12*(4), 249–261.

McWilliams, D., Stanfield, P., & Geiger, C. (2005). Parcel-hub scheduling problem: A simulation based solution approach. *Computers and Industrial Engineering, 49*, 393–412.

Napolitano, M. (2000). *Making the move to cross docking: A practical guide to planning, designing, and implementing a cross dock operation*. WERC Publisher.

Ratliff, H. D., Vate, J. V., & Zhang, M. (1998), Network design for load-driven cross-docking systems. Technical report, Georgia Institute of Technology.

Sadykov, R. (2009), A polynomial algorithm for a simple scheduling problem at cross docking terminals. Technical report, INRIA – 00412519.

Silver, E., Pyke, D., & Peterson, R. (1998). *Inventory management and production planning and scheduling*. Wiley.

Song, K., & Chen, F. (2007), Scheduling cross docking logistics optimization problem with multiple inbound vehicles and one outbound vehicle. In *International conference on automation and logistics*. Jinan, China.

Tsui, L. Y., & Chang, C.-H. (1990). A microcomputer based decision support tool for assigning dock doors in freight yards. *Computers and Industrial Engineering, 19*, 309–312.

Tsui, L. Y., & Chang, C.-H. (1992). Optimal solution to a dock door assignment problem. *Computers and Industrial Engineering, 23*, 283–286.

Yu, W., & Egbelu, P. J. (2008). Scheduling of inbound and outbound trucks in cross docking systems with temporary storage. *European Journal of Operational Research, 184*(1), 377–396.