Computers & Industrial Engineering 66 (2013) 696–709

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Static and dynamic policies with RFID for the scheduling of retrieval and storage warehouse operations ☆

Francisco Ballestín [a,*], Ángeles Pérez [a], Pilar Lino [a], Sacramento Quintanilla [a], Vicente Valls [b]

[a] Dpto. de Matemáticas para la Economía y la Empresa, Facultad de Economía, Universitat de València, Avda Naranjos s/n, 46021 Valencia, Spain
[b] Dpto. de Estadística e Investigación Operativa, Facultad de Matemáticas, Universitat de València, C/Dr. Moliner, 50, 46100 Burjasot, Spain

ARTICLE INFO

Article history:
Received 11 February 2013
Received in revised form 20 September 2013
Accepted 22 September 2013
Available online 29 September 2013

Keywords:
Warehouse operations management
RFID technology
Order management policies
Simulation

ABSTRACT

Warehouses are essential components of logistics and supply chains. The performance of warehouse operations significantly affects the efficiency of the whole chain it belongs to. Radio frequency identification (RFID) is an emerging technology capable of providing real-time information about the location and properties of tagged object(s), such as people, equipment or products. The objective of this article is threefold, to propose and compare different offline and online policies for the scheduling of warehouse operations, to design a tool that allows the decision maker to compare policies and environments without putting them into practice, and to study the benefits that can be obtained if RFID is used in a particular type of warehouse. To this end, we have developed a stylised model that captures and generalises the main characteristics of the structure, routing and sequencing operations of a given real warehouse. The model incorporates several realistic features never or rarely discussed in the literature in the presence of RFID, for example, due dates in the orders that have to be performed and congestion in the warehouse due to the presence of multiple vehicles performing the orders. We have also developed a set of heuristic routing and sequencing procedures that take and, alternatively, do not take into account real time information, and compare their performance via simulation on a set of randomly generated, although realistic, warehouse scenarios. Computational results show the effect in terms of due data fulfilment and tardiness minimisation if the RFID technology is installed and offline and online management policies are considered.

1. Introduction

In the supply chain, a warehouse is an essential component for linking the upstream (production) and downstream (distribution) entities, and most of the warehouse operations are either labour- or capital-intensive. The performance of these operations not only affects the productivity and operational costs of a warehouse, but also the whole supply chain. Therefore, it is necessary to enhance the productivity and reduce the operational costs of the warehouse.

Warehouse management systems (WMSs) have been developed for handling warehouse resources and monitoring warehouse operations in order to increase their efficiency and effectiveness. However, most often the current WMSs have been designed for conventional material-tracking systems and are incapable of providing timely and accurate warehouse operation information because they contain no feature of real-time and automatic data retrieval. It is difficult to update daily operations of inventory levels, locations of forklifts and stock keeping units (SKUs) in real-time with systems that rely heavily on warehouse staff members to input operational information manually or through bar-code systems.

Radio frequency identification (RFID) is an emerging technology that is increasingly being used in business and industry (Ngai, Moon, Riggins, and Yi (2008) and Liao, Lin, and Liao (2011)); particularly in logistics and supply chain management (see for example Wang, Chen, and Xie (2010), Chow, Choy, Lee, and Lau (2006), García, Chang, and Valverde (2006), Chow, Choy, and Lee (2007) and Poon et al. (2009) and the literature reviews of Visich, Li, Khumawala, and Reyes (2009) and Sarac, Absia, and Dauzère-Pérès (2010)). RFID is intended to replace traditional barcodes in many ways. Its wireless tracking nature allows a reader to activate a transponder on a radio frequency tag attached to, or embedded in an item, allowing the reader to remotely read and/or write data on the RFID tag. With both identification and tracking characteristics, RFID may dramatically change an organisation's capability to obtain real-time information about the location and properties of tagged object(s), such as people, equipment or products. Real-time

☆ The manuscript was handled by the Area Editor Manoj Tiwari, Ph.D.

* Corresponding author. Tel.: +34 963828396; fax: +34 961625448.
E-mail addresses: francisco.ballestin@uv.es (F. Ballestín), angeles.perez@uv.es (Á. Pérez), pilar.lino@uv.es (P. Lino), maria.quintanilla@uv.es (S. Quintanilla), vicente.valls@uv.es (V. Valls).

0360-8352/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cie.2013.09.020

information can be obtained without RFID, but using this technology facilitates obtaining it. As stated in Lim, Bahr, and Leung (in press), "interest in RFID in warehousing is rather stagnant and relatively small in comparison to other research domains". According to this review, neither due dates nor congestion, both key concepts in our paper, have been studied in the context of RFID.

For warehouses that keep inventories, the basic warehouse operations are to receive Stock Keeping Units (SKUs) from suppliers, store the SKUs in storage locations, receive orders from customers, retrieve SKUs from storage locations, assemble them for shipment and ship the completed orders to customers (surveys can be found at De Koster, Le-Duc, & Roodbergen, 2007; Gu, Goetschalckx, & McGinnis, 2007; Gu, Goetschalckx, & McGinnis, 2010; Roodbergen & De Koster, 2001). Storing and retrieving are the most expensive operations because they tend to be either very labour intensive or very capital intensive. The routing and sequencing decision in storage/retrieval order operations determines the best sequence and route of locations for storing and/or retrieving a given set of items where the storage/retrieval location of an item is given. Oliveira (2007)studies the problem of scheduling the truck load operations in an automatic storage/retrieval system. The problem is modelled as a job-shop problem by considering the loads as jobs, the pallets of a load as job operations and the forklifts as machines. A genetic algorithm for makespan minimisation is also presented. Zäpfel and Wasner (2006) formulate a warehouse sequencing problem in a steel supply chain as a dynamic job shop sequencing problem and present a heuristic algorithm for its resolution. Ho and Sarma (2009) present an abstract warehouse model to evaluate the location assignment problem in warehouse systems. They rely on RFID to keep track of items even if they are distributed non-contiguously across a warehouse. The study in Fosso and Chatfield (2010) provides support for the enabling role of RFID technology in effecting warehouse process optimisation. Poon et al. (2011) describe a real-time warehouse operation planning system for solving small batch replenishment problems. Real-time production and warehouse operations are monitored by RFID.

One of the objectives of this research is to show how real-time information allows us to design better management policies for the scheduling of storage/retrieval warehouse operations. Another goal of the paper is to study the benefits than can be obtained if RFID is used in a particular type of warehouse. The work provides some insights that can be drawn from the RFID technology in the context of warehouse storage/retrieval operations management that can be used to decide whether or not this technology is installed. To address both issues, we develop a stylised model that captures and generalises the main characteristics of the structure and routing and sequencing operations of a given real warehouse in which the Stock Keeping Units (SKUs) are pallets and the storage/retrieval operations are performed by means of forklifts. The model incorporates several realistic features not previously dealt with in the literature in the presence of RFID. They are the following: (1) Due to the confined and narrow travel paths in a warehouse and also for security reasons, some restrictions on forklift movements have to be considered such as security distances, avoiding forklift intersection policies and aisle entrance blocking policies. The non-consideration of these constraints could lead to the proposal of unreliable plans of action and to meaningless results. (2) The expected arrival time of trucks to collect goods or the production requirements impose temporal limits on the fulfilment of retrieval operations. These temporal limits are reflected in the model as due dates on the fulfilment of groups of retrieval orders. It seems that in practice the fulfilment of these due dates is an objective more important than the usually considered objective of makespan minimisation.

We have developed a set of order sequencing and routing heuristic procedures that take, and, alternatively, do not take into account, real time information and compare their performance via simulation. We have considered two types of order management settings: static (S) and dynamic (D), depending on whether the forklift driver is provided at the start of the working period with an ordered list of orders to be fulfilled or is provided with a single order at the beginning of the working period and each time he arrives at the depot after he has completed the previous order.

Furthermore, to facilitate the accurate estimation of durations, the time to fulfil an order is calculated as the sum of several main times: the travel time, the waiting time, the depot time and the storage/retrieval time. Also to compute each of these main times we have considered the times needed to perform different elementary actions: travel time (to traverse an arc in a storage aisle or in a cross aisle); depot time (to assimilate a new order, to manually identify a pallet, to automatically identify a pallet and to deposit or remove a pallet from the depot ground); storage/retrieval time (to position the forklift in front of the storage location, to manually and automatically identify a pallet, to lift up the pallet to the proper level, to bring down the forks, to temporally deposit on the ground and pick up afterwards the carried pallet and one or more pallets already stored if necessary; to manoeuvre the forklift to a position that allows it to traverse the aisle). This detailed decomposition also facilitates the identification of features of the current warehouse status that influence the quality of the generated plans as is the case with area congestion (see Section 4).

The problem has been motivated by the study of a warehouse of a Spanish company that produces beauty products (shampoo, creams, lotions, and so on) for one of the most important supermarket chains in the country. This company is interested in knowing the benefits that can be obtained in terms of due data fulfilment and tardiness minimisation if the RFID technology is installed and some management policies are considered. To address this concern we considered the design of the warehouse and times to fulfil orders based on actual data from this company. However the layout corresponds to a rectangular layout whose use is widespread and times correspond to the usual speed of forklifts.

The rest of the paper is organised as follows: In Section 2 we describe the main characteristics of the type of warehouse we are considering. We also define a mathematical model for the optimisation problem dealt with in this paper. In Section 3, we introduce the types of order management settings and the technological environments we analyse throughout the paper. Section 4 is devoted to present the solution approach adopted. We describe the algorithm and the different policies proposed to solve our problem. An example is included to facilitate the understanding of the algorithm. We also present a simulator that simulates the structure and operations in a warehouse. The simulator will be used to evaluate the performance of a given order management policy. In Section 5 we carry out several experiments to analyse the performance of our procedures and the benefits of using the RFID technology. To do that we have also devised a procedure capable of generating test instances similar to our case. Finally, conclusions are given in Section 6.

2. Warehouse description and model specification

In this section we describe the main characteristics of the warehouse we are working with. We also represent the warehouse as a graph, which will help us to develop a program that simulates the movements in the warehouse. Finally, we define a mathematical model for our problem.

2.1. Warehouse description and problem characteristics

In this paper we consider rectangular warehouse layouts (Fig. 1) with a certain number of parallel storage aisles and parallel cross

aisles. Storage locations are at both sides of the storage aisles at multiple levels. Cross aisles do not contain storage locations. A working zone (WZ) is defined as a section of a storage aisle between two cross aisles. We also need the concept of subworking zone (SWZ). If a working zone has only one entrance, there is only one SWZ, which is all the working zone. If there are two entrances to the working zone, we divide the working zone in two halves, and each of the halves is a different SWZ. The set of storage locations in one of the sides of a WZ is called a rack. The warehouse is divided in sections as is shown in Fig. 1. All racks in the same section have identical numbers of columns and all columns in the same rack have the same number of levels. The SKUs are homogeneously loaded pallets although different pallets may be loaded with possibly different items. The dimensions of the base of all pallets are the same although their heights can be different. The warehouse has a single receiving/dispatching area that we will call depot for simplicity, where workers deposit the collected pallets and remove the pallets to store; orders are assigned only at this area. The depot is located in the middle of the warehouse front; storage aisles run parallel to this front.

Rectangular warehouses (Caron, Marchet, and Perego (2000)) are the most popular ones in the literature. Two types can be considered. In the first one, common in picking problems, storage aisles run perpendicular to the warehouse front where the depot is located. In the second, storage aisles run parallel to the warehouse front, as in our model. Several versions of these layouts can be considered depending on which point of the front aisle the depot is located in (in a corner of the front aisle (Roodbergen and De Koster (2001), Roodbergen, Sharp, and Vis (2008)), in the middle of the front aisle (Caron et al. 2000) or even separating an in-zone and an out-zone as in Hsieh and Tsai (2006).

Storage locations can be single or double-depth. Single-depth storage locations can store up to 2 pallets one on top of the other in positions 2 and 1, respectively. Double-depth storage locations can store up to 4 pallets, two (positions 2 and 4) on top of the other two (positions 1 and 3). We will say that positions 1 and 2 are in the first depth and positions 3 and 4 are in the second depth. All storage locations in the same side of a given storage aisle have identical depth. Fig. 3 illustrates (front view and side view) two back to back double-depth racks with four levels, three storage locations in each one of them and 4 positions in each storage location.

Storage locations can be of different heights. However, all storage locations in the same level of the same rack are of the same height.

The warehouse is of the chaotic type where a pallet can generally be stored in any storage location although some restrictions are applied to certain pallets. The most important restrictions are that certain pallets should be stored below a maximum level and that there are pallets that cannot have another one on top of them.

Generally, in the depot the pallets are on the ground grouped together with others that are identical.

There is a set $F = \{f_1, \ldots, f_{nf}\}$ of available forklifts with the corresponding driver that can work simultaneously. There are four types of forklifts:

- Type 1: forklifts that can manage pallets only at the first level and at the first depth.
- Type 2: forklifts that can manage pallets up to the third level but only at the first depth.
- Type 3: forklifts that can work at any level but only at the first depth.
- Type 4: forklifts that can work at any level and any depth.

Forklift types are hierarchical in the sense that the locations reachable by a forklift of a type include the locations reachable by a forklift of any inferior type. All forklifts work at the same speed.

At a given time instant there is a set of orders to be fulfilled, $O = \{o_1, \ldots, o_{norders}\}$. Each order is determined by four elements, $o_i = (p(o_i), l(o_i), \mathrm{pos}(o_i), \mathrm{dd}(o_i))$, where: $p(o_i)$ is the pallet that must be stored (retrieved), $l(o_i)$ is the storage location where $p(o_i)$ has to be stored (retrieved from), $\mathrm{pos}(o_i)$ ($\mathrm{pos}(o_i) = 1,2,3,4$) is the position of $p(o_i)$ in the storage location $l(o_i)$ and $\mathrm{dd}(o_i)$ is the due date for order fulfilment. It is assumed that position $\mathrm{pos}(o_i)$ is feasible for pallet $p(o_i)$. Each retrieval order has associated a finite due date representing the instant of time where the associated pallet should be deposited in the depot as a maximum. Due dates are motivated, for example, by the expected arrival time of trucks to collect goods or by production requirements. The due date of a storing order is always $+\infty$.

Forklift drivers receive the orders at the depot one at a time. To fulfil a given retrieval/storing order, (a worker driving) a forklift capable of reaching the order storage location retrieves/stores a single pallet from/to the corresponding storage location and transports it/returns to the depot. The forklift always follows the shortest path to go and to return. We will denote $\mathrm{Fork}(o_i)$ as the set of forklifts capable of performing order o_i. Forklifts can traverse cross and storage aisles in both directions. A forklift can change direction in a storage aisle after retrieving or storing the pallet. Due to the confined and narrow travel paths in a warehouse and also for security reasons, some restrictions on forklift movements have to be considered. Two forklifts cannot go in parallel in the same direction in any aisle although they can go one after the other along the

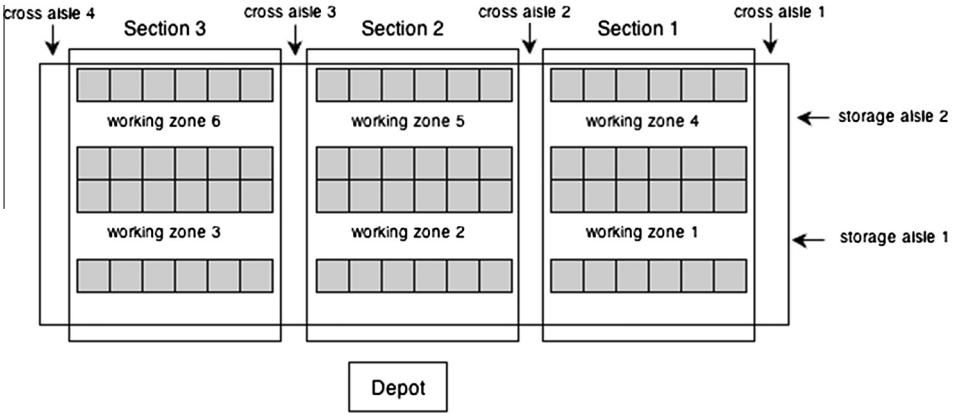

Fig. 1. The warehouse layout (plan view).

same aisle if they maintain a security distance. In the case of storage aisles, a distance of three storage locations would seem to be a reasonable security distance. In the case of cross aisles, the security distance is defined as at least one aisle intersection which must be between two forklifts. Two forklifts cannot intersect in the same aisle. If this is going to happen, one of them has to enter in an adjacent aisle and stop until the other forklift has passed (waiting time). Up to two forklifts can work simultaneously in the same aisle maintaining a security distance already mentioned. A forklift which enters a WZ at one end must exit from the same end. This means that the key question is in which subworking zone an order is situated. By SWZ(o_i) we will denote the subworking zone where the location of the order is situated. A forklift that wants to enter in an occupied SWZ must wait until the other forklift exits (waiting time). The term congestion is sometimes used in the literature to include these waiting times produced by "traffic problems" inside a warehouse, see for example, Chao-Hsien Pan and Wu (2012). As far as we know, congestion has not been studied in connection with RFID or to compare offline and online policies.

The time to fulfil an order, $t(o_i)$, is calculated as the sum of several times: the time spent at the depot (depot time), the travel time, the waiting time spent to avoid intersections and to comply with aisle entry restrictions and the time a forklift spends either storing or retrieving a pallet from the storage location $l(o_i)$ which depends on its position in the rack (storage/retrieval time). In this paper we consider that each of the above times is obtained as the sum of the durations of several more elementary actions and that these durations are random variables whose probability distributions are known. We further elaborate on this issue in Section 5.

The finishing time $f(o_i)$ of a storing order is the time instant at which the forklift that is performing the order o_i returns to the depot. In the case of a retrieval order, the finishing time is the time instant at which the corresponding pallet is placed in the depot.

Roughly speaking, the problem we study in this paper is the following: given a set of orders to be fulfilled, find a sequence of order fulfilment and a forklift for each order, with the objective of minimising the tardiness of the retrieval orders. Nevertheless, the model is clearly defined in Section 2.3. In order to do this, we need to represent the warehouse as a graph.

2.2. Managerial/graphical representation of the warehouse

The warehouse layout can be represented by an undirected weighted graph $G = (V,A)$ where V and A are the sets of vertices and arcs, respectively (see Figs. 1 and 2). The set of vertices V contains:

- A vertex for every two columns of storage locations placed one in front of the other in the same aisle.
- A vertex for each intersection of aisles.
- A vertex representing the depot.

The set of arcs contains:

- An arc for every pair of vertices representing adjacent columns of storage locations.
- An arc for every pair of adjacent aisle intersections.
- An arc for every pair formed by an aisle intersection and a storage location column located first in any aisle leaving the intersection.
- An arc connecting the depot vertex with each of the vertices representing the aisle intersections nearest to the depot.

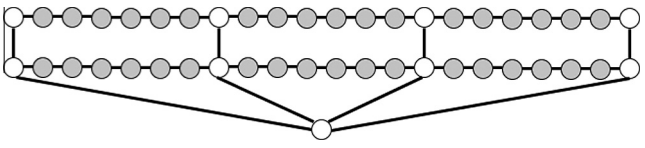

Fig. 2. Graph representation corresponding to the warehouse.

The weight of an arc is the time a forklift spends traversing it. To execute an order o_i it is necessary to traverse the shortest route on the graph G, starting at the depot vertex, visiting the location $l(o_i)$ vertex and coming back to the depot vertex.

The aforementioned forklift movement restrictions can be worded now by simply saying that: There cannot be more than one forklift in a vertex or in an arc and that the minimum distance between two forklifts in the same storage aisle is three vertices.

2.3. Model definition

In this section, we provide a conceptual formulation of the problem as a resource project scheduling problem (Kolisch, 1994), with the aim of unambiguously establishing the constraints and objective involved. The main elements of a project are the activities to perform, the precedence relationships between them, and the resources that the activities require. Projects can be represented in a directed graph where the vertexes are the activities to perform and the arcs are the precedence relationships between the activities. We need therefore a graph, but a different one from the G defined in the previous section. The new graph, set of vertexes and set of arcs will be denoted respectively by Gm, Vm and Am, where the "m" is devised from "model".

The graph of our model, $Gm = (Vm,Am)$, has $n+2$ vertexes, including two dummy vertexes 0 and $n+1$ representing, respectively, the beginning and end of the project. Thus $Vm = \{0,1,\ldots,n+1\}$ is the set of project activities. To fulfil a given order a forklift has to perform a sequence of actions (traversing certain arcs, manoeuvring for storing/retrieving a pallet, etc.). In this way, each order o_i generates the following activities in the project (or vertexes in Vm), whose accomplishment will be equivalent to the execution of the order:

- two dummy activities (d_{i1} and d_{i2}) representing the beginning and end of the order. Their duration is equal to zero.
- an activity for each one of the arcs of the shortest route in G between the depot and the entrance to SWZ(o_i). They represent the trip of the forklift to SWZ(o_i). The activities of this shortest path will be denoted by: $a_{i1}, a_{i2}, \ldots, a_{\mathrm{in(i)}}$, where $n(i)$ denotes the number of arcs in this path. The duration of each one of these activities is equal to the time that a forklift takes in traversing the corresponding arc of G.
- an activity representing the actions associated to the order inside SWZ(o_i). It will be denoted by SWZ$_i$. It includes, for example, the action of travelling along the arcs of G in the storage aisle of SWZ(o_i) and the actions of storaging/retrieving the pallet from the location (see Section 5). The duration of this activity is the sum of the duration of all these actions.
- an activity for each one of the arcs of the shortest route in G between the entrance to SWZ(o_i) and the depot. They represent the forklift return trip from SWZ(o_i). The activities of this shortest path are obviously: $a_{\mathrm{in(i)}}, a_{\mathrm{in(i)}-1}, \ldots, a_{i1}$. The duration of each one of these activities is equal to the time that a forklift takes in traversing the corresponding arc of G.

Fig. 4 illustrates the graph Gm corresponding to the example of Fig. 5, which will be used in Section 4.4., but only considering orders O3, O4, and O7. For example, d_{31} marks the start of order 3. The forklift has only to traverse a cross aisle (a_{31}) to arrive to the

subworking zone of the order. Then the forklift has only to traverse one more arc (a_{32}) to arrive to the location, the one that connects the aisle intersection and the first storage location column. Fig. 6 shows the arcs of G used in Gm. In the vertexes of Fig. 4 we show the description of the vertex for the sake of clarity; however in Gm they are numbered following a topological order. The precedence relationships between the vertexes (see Fig. 4) are all of the finish-start type and establish the sequence in which the actions of an order have to be fulfilled to perform the order.

In order to complete the project, each activity has to be processed in one of several modes. Each mode stands for a different way of performing the activity, depending on the resources used. The set of modes of activity j is denoted by $M(j)$.

There are two types of resources in our model. There are renewable resources (Kolisch, 1994) and cumulative resources (Neumann & Schwindt, 2002). Renewable resources are available in the same quantity for every single period. The different subworking zones of the warehouse can be considered as renewable resources with only one unit available of each of them. With these resources we model the fact that only one forklift can work in the same subworking zone at a given time. Since only one forklift can traverse the same arc at a given time, the arcs in G that belong to the shortest route of any of the orders are also considered as renewable resources with only one unit available of each of them. Cumulative resources permit that, when an activity starts (finishes), a certain quantity of a resource is consumed (replenished). There is a total quantity available. Each forklift can be considered as a cumulative resource with availability equal to one. These resources are used to assign the same forklift to all the activities that define a certain order and to force the selected forklift, once it has started an order, to execute all the activities associated with that order before starting any new order.

The modes associated with each one of the activities in Gm are the following:

- the dummy vertexes (d_{i1} and d_{i2}) have as many associated modes as the number of forklifts that can perform o_i. Each mode requires a unique cumulative resource: a forklift of Fork(o_i). Let r_{ikm} denote the demand of the cumulative resource k of activity i if the activity is performed in mode m. If $k \in$ Fork(o_i) is the forklift assigned to mode m, the dummy vertex d_{i1} will have $r_{ikm} = -1$, indicating a consumption of resource k – the forklift has started an order so it is occupied until the order finishes. The dummy vertex d_{i2} will have $r_{ikm} = +1$ indicating a replenishment of resource k – the forklift has finished an order, so it is free to be assigned to another one.
- a vertex of type SWZ_i has a unique mode that requires a unique resource: the subworking zone associated with the storage location of o_i (SWZ(o_i)).
- a vertex of type a_{ij} has a unique mode that requires a unique resource: the arc in G that a_{ij} represents.
- the dummy vertexes 0 and $n+1$ has a unique mode that does not require any resource.

Furthermore, we associate with the second dummy activity of each order a due date equal to the due date of the order it represents if the order is a retrieval one and equal to $+\infty$ if the order is a storage one.

A schedule or solution S of the new model is represented by two vectors (s,m). The vector $s = (s_0, s_1, s_2, \ldots, s_n, s_{n+1})$ of non-negative integers indicates the start time of each activity i, whereas $m = (m(0), m(1), m(2), \ldots, m(n), m(n+1))$ assigns to each activity i a mode $m(i) \in M(i)$. A schedule $S = (s,m)$ is a feasible schedule if the starting times fixed by S satisfy the generalised precedence relationships and the resource constraints. Therefore, S is a feasible schedule if S satisfies the following constraints:

$$m(i) \in M(i) \quad \forall i \in Vm \tag{1}$$

$$S_i - e_i \geq 0 \quad \forall i \in \text{Pred}(j) \tag{2}$$

$$\Sigma_{i \in A(s,t)} r_{ikm(i)} \leqslant 1 \quad \forall k \in \text{RR},\ 0 \leqslant t \leqslant \text{PDUR} \tag{3}$$

$$0 \leqslant \Sigma_{i \in D(s,t)} r_{ikm} \leqslant 1 \quad \forall k \in \text{CR}, 0 \leqslant t \leqslant \text{PDUR} \tag{4}$$

$$0 \leqslant s_i \leqslant \text{PDUR} \quad \forall i \in Vm \tag{5}$$

$$S_0 = 0 \tag{6}$$

where

- RR is the set of renewable resources.
- CR is the set of cumulative resources.
- $e_i = s_i + d_i$, is the finishing time of activity i.
- Pred(j) is the set of predecessor activities of activity j in graph Gm.
- The value r_{ikm} with $k \in$ RR represents the demand of the resource k by activity i if it is performed in mode $m \in M(i)$. It is defined as:

$$r_{ikm} = \begin{cases} 1 & \text{if activity } i \text{ uses one unit of resource } k \text{ in mode } m \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

- The value r_{ikm} with $k \in$ CR represents the demand of the resource k by activity i if it is performed in mode $m \in M(i)$. It is defined as:

$$r_{ikm} = \begin{cases} 1 & \text{if activity } i \text{ replenishes one unit of resource } k \text{ in mode } m \\ -1 & \text{if activity } i \text{ consumer one unit of resource } k \text{ in mode } m \\ 0 & \text{if activity } i \text{ does not use resource } k \text{ in mode } m \end{cases} \tag{8}$$

- $A(S,t) = \{i \in Vm / s_j \leqslant t \leqslant f_i\}$ is the set of activities that are in process at instant $t > 0$.

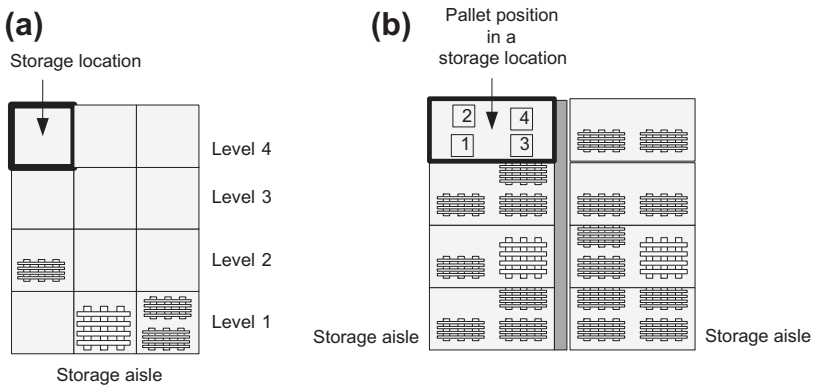

Fig. 3. (a) A rack seen from the front. (b) Back-to-back double-deep rack seen from the side.

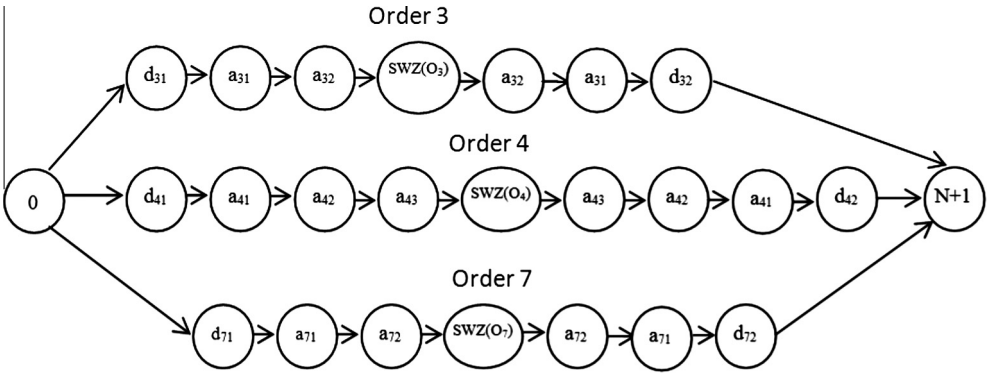

Fig. 4. Graph *Gm* corresponding to orders O3, O4 and O7.

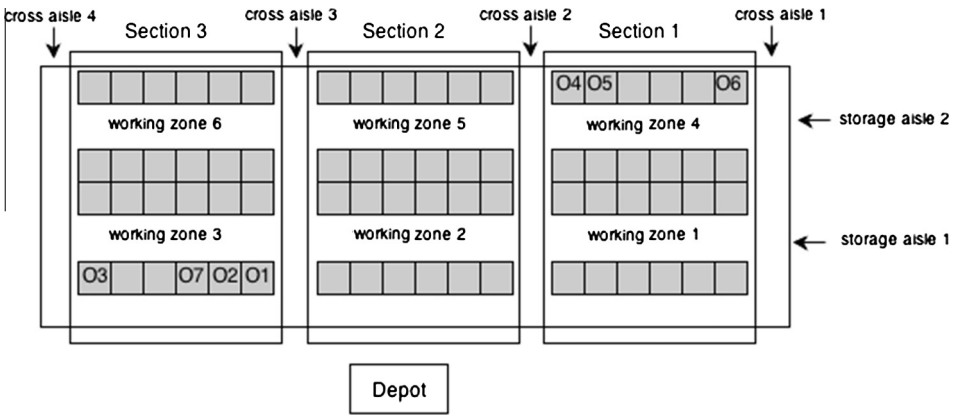

Fig. 5. Warehouse layout with the orders of the example (Section 4.4).

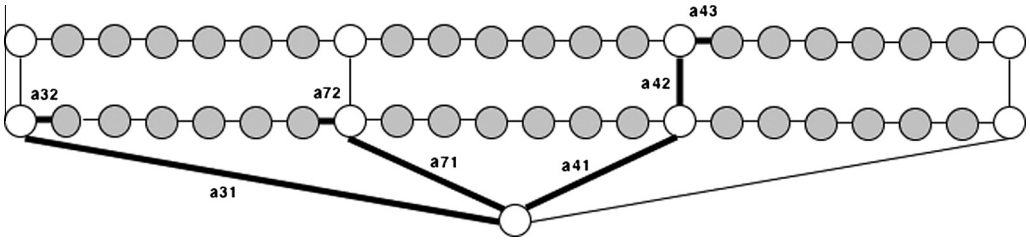

Fig. 6. Arcs in *G* used in graph of Fig. 4.

- $D(S,t) = \{i \in Vm / s_j \leqslant t\}$ is the set of activities which have occurred by instant $t > 0$.

Eq. (1) ensure that each activity i is assigned to a unique mode of $M(i)$. Eq. (2) ensure that activities start after their predecessors have finished. Eq. (3) guarantees that each renewable resource does not execute more than one activity simultaneously. Eq. (4) guarantee that the amount of a cumulative resource k used until time t is never more than one. Eqs. (5) and (6) fix the activity start times within the scheduling horizon PDUR (an upper bound of the project duration).

The objective of the problem is to find a feasible schedule S minimising the tardiness of the retrieval orders:

$$\text{Min} \sum_{i \in \text{Pred}(n+1)} \max(0, e_{i-dd(i)}) \tag{9}$$

In the previous model we have disregarded the space restriction between forklifts in the same aisle, because of simplicity. This type of restriction could be modelled by dividing each SWZ node into several nodes and assigning cumulative resources to each of them.

The problem we have just defined is a deterministic problem. However, all durations, as previously commented, are random variables (see Section 5 for a deeper description). We are therefore dealing with a stochastic optimisation model. Therefore, exact methods and commercial software are not suitable for solving medium or large instances. We have to resort to heuristic algorithms (see Section 4).

3. Storage and retrieval orders management environments

Due to the stochasticity of our problem, we cannot specify the start of the orders beforehand. Therefore, a solution for our problem consists of an *order management policy*, which indicates the procedure for determining the next order to be fulfilled and the forklift that is going to perform the order. The route followed to perform the order is always the shortest path in G between the depot and the order location. The policy is carried out in a specific

environment, which determines the moments where decisions are made and the conditions of the warehouse in terms of RFID technology. The actual time instants at which each order starts will depend on the actual durations of the different activities (journey from or to the depot, performing the order, etc.).

We have considered two types of order management decision settings: static (S) and dynamic (D). In a static setting, each forklift driver is provided at the start of the working period with an ordered list of orders in such a way that their storage locations can be reached by the forklift. The orders are fulfilled in the prescribed order, one after the other. The order lists are constructed following a prescribed priority rule.

In a dynamic setting, each forklift driver is provided with a single order to be fulfilled at the beginning of the working period and each time he arrives at the depot after he has completed the previous order. Given an available forklift, a non-assigned order whose storage location can be reached by the forklift is assigned to it, following a prescribed priority rule. This procedure is repeatedly applied until all the orders have been assigned. If at a given time instant there is no forklift capable of performing any of the non-assigned orders then the decision maker (the depot manager or the information system) waits until a currently assigned forklift finishes its assignment and becomes available again. In both cases, the forklifts traverse the routes complying with the movement constraints specified above and performing the required operations.

In this paper we also want to explore the impact of the RFID technology on the total tardiness T. We have based our comparisons upon results obtained using bar codes which are used daily. With RFID we are able to identify and locate diverse items that have tags attached to them automatically without the need for human intervention and in real time. However, when using bar code technology a person is required to physically put a reader next to the tag in order to read the information. It should be noted that in our particular case there are two types of operations where tags need to be read: To identify locations and to identify pallets.

When using bar code technology, the forklift driver should make the following tag readings:

1. In case of a retrieval order: Once the forklift gets the assigned storage location the driver has to get down from the forklift, read the tag attached to the storage location, get back into the forklift, load the pallet on the forklift and get down again to read the tag attached to the pallet. Once the retrieved pallet has been deposited on the depot ground the driver gets down again for identifying the deposited pallet.
2. In case of a storage order: The driver gets down from the forklift in the depot to identify the pallet to be stored, gets down again at the assigned storage location to identify it, stores the pallet in the storage location and, to further ensure the accuracy of the data, gets down again to identify the storage location in which the pallet has been stored.

If RFID tags are attached on the storage locations and pallets then the identification of these items is realized without human participation therefore saving time. The bar code solution requires the driver to carry about a hand reader and the RFID solution requires the forklift to have a RFID reader installed on it.

Taking into account this information, we have considered three technological environments. In the first one (BC), we assume that the bar code technology is deployed. In the second environment (RFID1), we assume that RFID tags are attached on the racks to identify the storage locations, whereas in the third (RFID2) tags are also attached to the pallets. In technological environment RFID1, there is no need to step out of the forklift for location identification. In addition to the above time savings, in technological environment RFID2 it is also unnecessary to exit the forklift to locate a particular pallet.

Combining the two types of decision settings (S,D) with the three technological environments (BC, RFID1, RFID2) we obtain six order management environments (SBC, SRFID1, SRFID2, DBC, DRFID1, DRFID2) for which we have developed a set of priority rules whose performance will be compared via simulation.

In our experiments, we assume that no errors in the data have occurred. In reality errors occur, and there are more errors with the use of bar codes than with RFID technology, another increase in efficiency in using RFID technology (García et al. 2006).

4. Solution approach

In this section we describe first the algorithms we have proposed for the static and dynamic settings, second the specific policies which will be applied in each of these settings and third the simulator we have developed to evaluate the performance of our procedures. We will finally present an example of our algorithms.

4.1. Algorithm description

The next schemes describe the algorithms applied to solve the problem in the dynamic and the static settings, respectively. They depend on the priority rule they are applied with. The priority rules are explained in Section 4.2. Due to the randomness of the problem, the performance of a policy can only be evaluated through different simulation runs; the developed simulator is described in Section 4.3.

4.1.1. Dynamic setting

- Step 0. Initialisation.
  1. tnow = 0 · O_{left} = ∅ O_{out} = ∅.
  2. Select an order for each forklift with the given priority rule.
  3. Perform that order with the assigned forklift.
  4. Eliminate the assigned orders from O_{left} and introduce them in O_{out}.

  Step 1.
  5. Let tnow' > tnow be the minimum time the next forklift f_i arrives at the depot. Eliminate from O_{out} the order f_i has performed. tnow = tnow'.
  6. If $O_{out} \cup O_{left} = \emptyset$, STOP.
  7. If $O_{left} \neq \emptyset$, use the given priority rule to choose an order for f_i, or to wait until the next forklift arrives. Eliminate the order from O_{left}.
  8. Perform the chosen order with the assigned forklift. Go to 4.

4.1.2. Static setting

Step 0. Initialisation.
Calculate with the given priority rule an ordered list of orders to be fulfilled for each forklift.
Step 1.
Perform each order with the assigned forklift, in the prescribed order. When a forklift returns after finishing an order it starts with the following one.

4.2. Priority rules

We next introduce separately the priority rules that we have devised for the static and dynamic order management settings. To evaluate the performance of a priority rule we need a program that simulates the conditions of the warehouse. This simulator is explained in Section 4.3.

4.2.1. Dynamic order management settings

A priority rule in a dynamic order management setting is a procedure for determining the next eligible order to be fulfilled by an available forklift. Given a forklift, an eligible order is a non-assigned order whose storage location can be reached by the forklift. The priority rules we have devised consist of a primary rule and a secondary rule that is applied in case of ties. Further ties are broken at random.

The rules we have used either as primary rules or as secondary rules are the following:

a. Due date. Select an order with minimum due date.
b. Forklift sub-utilisation. Select an order with minimum sub-utilisation. Given an order, o_i, there is a minimum type of forklift, $forklift\_type_i$, which can fulfil the order. Using a forklift of type $j > forklift\_type_i$ to fulfil order o_i implies a sub-utilisation of the resource that we measure as $j - forklift\_type_i$.
c. Working zone congestion. Select an order with minimum SWZ congestion. Given an order, its SWZ congestion is computed as the number of orders whose storage locations are in the same working zone as the storage location of the given order and which fulfilment has started but not finished yet. Remember that the fulfilment of an order finishes when the forklift that is performing the order returns to the depot.
d. Random. Select an order at random.

4.2.2. Static order management setting

A priority rule in a static order management setting can be seen as a procedure for assigning to each forklift an ordered list of orders that can be executed by the forklift or, equivalently, as a procedure for assigning to each order a forklift capable of performing the order.

To assign orders to forklifts we proceed as follows: We consider the set of orders one by one in order of increasing due date. Ties are broken at random. Given an order, we assign it to a forklift capable of performing the order by means of a primary rule and a secondary rule.

As primary rules, we have considered the following rules:

a. Number of orders balance. Select the forklift with minimum number of orders assigned to it up to this moment.
b. Duration balance. Select a forklift for which the sum of the estimated execution times of the orders already assigned to it is minimal. The estimated execution time of an order is an estimation of the time that the given forklift would spend to fulfil the order given that no other forklift is in the aisles and is obtained as the sum of the averages of three partial times: depot, travel and storage/retrieval times.
c. Random. Select a forklift at random.

As secondary rule, we have always applied the Forklift sub-utilisation rule: Select a forklift for which the sub-utilisation is minimal. Further ties are broken at random.

4.3. Warehouse operations simulator

We have developed a simulator that simulates the structure of and the storage/retrieval operations in a warehouse. The inputs are: the physical structure of the warehouse, the size of the fleet of forklifts, the possible location restrictions of the pallets, the initial state of the warehouse, the durations of the different types of operations and the list of orders to be fulfilled.

The physical structure of the warehouse is described by stating the number of cross and storage aisles and the number of levels in each storage rack. The input for the forklifts indicates the number of forklifts for each type. The possible restrictions on the storage position of a pallet are introduced as an input. For the experiments, we have considered two of the most popular ones: the maximum level at which a pallet can be stored and whether a pallet admits another pallet on top of it or not. The initial state of the warehouse is defined indicating for each occupied storage position the pallet that is stored in it. The simulator considers that the time spent by a forklift to fulfil an order is obtained as the sum of four partial times corresponding to four main forklift actions (the depot time, the travel time, the waiting time and the storage/retrieval time). To allow for a more accurate comparison between different strategies of operation management, a further disintegration of these main partial times is needed. Therefore, each of these times is in turn obtained as the sum of the durations of several more elementary actions. We refer again to Section 5 for a full description of the different times considered. The probability distributions of the durations of these elementary actions are inputs to the simulator.

The simulator simulates the movements of the forklifts inside the warehouse. Using the graph G representing the warehouse, it can calculate the location of the forklifts inside the warehouse at any time instant. To fulfil a given order, the selected forklift has to perform a sequence of actions (traversing certain arcs, manoeuvring for storing/retrieving a pallet, waiting for target reading, etc.) whose durations are random variables. Actual durations are generated for each action by drawing a new duration from its associated probability distribution. These generated durations are used to calculate the locations of the forklifts as time goes on. Also the simulator simulates appropriated actions to avoid forbidden movements of the forklifts.

The simulator maintains an information structure called the *Forklift Position Map*. The Forklift Position Map at time t consists of the graph G representing the warehouse plus the information of the position of every forklift in G at that instant. At time t, a forklift could be traversing an arc or in a node. The Forklift Position Map is updated only at certain points in time: each time a forklift reaches a node or finishes the storage/retrieval of a pallet. The updating of the Forklift Position Map takes into account the time used to comply with the forklift movement restrictions. If a forklift that is located at a certain vertex is going to enter into a temporarily forbidden arc then it is stopped at that vertex or in an adjacent arc until the conditions that make that arc forbidden disappear. At time 0, the Forklift Position Map coincides with G.

We will use the simulator to evaluate the performance of a given order management policy. For a given instance of the problem, the simulator simulates the duration of each action and makes the decisions at certain decision times according to the given policy. A simulation run will be the simulation of every action until every order has been completed. The value of T for the simulation run is computed. To a given instance, we associate the average value of T over 30 simulation runs. We further elaborate on this issue in Section 6.

4.4. Example of the algorithm

In the following part we describe an example of the algorithm described in Section 4.1. We will use the warehouse example depicted in Fig. 5. We will work with 7 orders and three forklifts, one of type 1, another of type 3 and another of type 4. Table 1 shows the data of the orders. Given the level and depth of the orders, only forklifts two and three can perform orders 2, 4, 5 and 7, and only forklift three can perform order 1.

S/R stands for storage and retrieval, respectively. "Assim." means the time needed to assimilate an order. "Depot time out" (in) is the time needed when the forklift leaves (arrives at) the depot. "Total duration" is the sum of all times for an order; note that

Table 1
Data of the orders of the example.

Number of order	S/R	Assim.	Depot time out	Depot time in	Travel time	Storage/retrieval time	Total duration	Due date	Depth	Level
1	R	0.5	0	1.25	0.024	2.083	3.881	8 min	2	6
2	R	0.5	0	1.25	0.032	2.083	3.897	10 min	1	6
3	R	0.5	0	1.25	0.024	1.166	2.964	6 min	1	1
4	S	0.5	1.25	0	0.04	1.583	3.413	∞	1	3
5	R	0.5	0	1.25	0.048	1.583	3.429	8	1	3
6	R	0.5	0	1.25	0.04	1.166	2.996	20 min	1	1
7	S	0.5	1.25	0	0.04	1.583	3.413	∞	1	3

the travel time must be counted twice. "Level" and "depth" contains information about the location of the order. More information about these times is given in Section 5.

To calculate the assignation in the static setting, we sort the orders according to their due dates, ties broken randomly, and obtain for example (3 1 5 2 6 7 4). We assign the orders in this order to the forklifts, taking into account the workload of the forklifts at the moment of the assignation and assigning the worst forklift capable of performing the order. Below we have the assignment we make. In parentheses we can see the workload of each forklift after each assignation. For example, we assign order 2 to forklift 2 because at that moment this forklift has a workload of 3.429 and forklift 1 has a workload of 3.881. Note that order 1 can only be done by forklift 3 and forklift 1 can only perform orders 3 and 6.

- F1: (0) 3 (2.964) 6 (5.96)
- F2: (0) 5 (3.429) 2 (7.326)
- F3: (0) 1 (3.881) 7 (7.294) 4 (3.413)

To evaluate the performance of this policy we would calculate several scenarios or runs, with different random values for each duration considered, taken out from their corresponding distributions (see Section 5). Then we would calculate the tardiness in each scenario and calculate the average over all scenarios. In this example we are only going to calculate one scenario. The values for each time we have used are the average times shown in the previous table, except in two cases. The retrieval time for orders 1 and 2 is going to be 3.383 and 3.083 min, respectively, to help us understand several of the features of the simulator and differences between the settings.

Let us see how the orders would be performed according to the scenario and the decision taken under the static setting. Table 2 shows the most important times for each order.

At time 0 we start orders 3, 5 and 1, with their forklifts 1, 2 and 3, respectively. When a forklift becomes available we assign the next order to it. For instance, when forklift 2 finishes order 5, it starts order 2. However, on its way to the SWZ it crosses forklift 1, which is returning from the SWZ. Forklift 2 has to enter in an adjacent aisle and stop until forklift 1 has passed. We therefore add 0.333 min to the time of this forklift for this concept (called waiting time). When the forklift 3 is performing order 7 and it arrives at the corresponding SWZ, forklift 2 is in the same SWZ. Forklift 3 cannot enter in the SWZ until forklift 2 leaves it. Once the run has finished, the tardiness can be calculated. In this case only order 2 is tardy, so $T = 12.159 - 10 = 2.159$ min. The flow time (total time in performing all orders) is 12.5 min.

If we apply the dynamic setting with these values, the first decisions are the same as in the static case. All the rows of the previous table except the last two are common. At time 0 each SWZ is empty, and therefore we choose according to the due date. To assign a forklift we use sub-utilisation. Therefore we also assign orders 3, 5 and 1 to forklifts 1, 2 and 3, respectively. When each forklift arrives at the depot it is decided which order it has to do next. Forklift 1 and 2 receive orders 6 and 2, respectively. When forklift 3 becomes available, orders 4 and 7 are still to be performed. At that moment (time 5.181), the SWZ of order 7 is occupied, because forklift 2 is performing order 2. Therefore the policy decides to use forklift 3 to perform order 4 in another SWZ. Forklift 3 becomes available before forklift 2, and is therefore responsible for order 7 as well. However, forklift 2 would have been assigned to order 7 if it had arrived earlier. In fact, forklift 1 is available since time 5.181. If this forklift could have done order 7 the dynamic case could have assigned forklift 1 to order 7, whereas the static case cannot do this. The next table (Table 3) shows the last two rows of the dynamic case. In this example the tardiness is also 2.159 min, but the flow time is 12.007 min.

5. Computational experiments

In this section we present the experiments we have performed to test the procedures developed in Section 4. First of all we need a set of random instances generated according to the parameters

Table 2
Most important times for each order in the example.

Order (F)	Forklift available	Order starts	Leaves depot	Arrives at SWZ	Enters SWZ	Leaves SWZ	Arrives at depot	End of order
3 (1)	0	0.5	0.5	0.524	0.524	1.69	1.714	2.964
5 (2)	0	0.5	0.5	0.548	0.548	2.131	2.179	3.429
1 (3)	0	0.5	0.5	0.524	0.524	3.907	3.931	5.181
6 (1)	2.964	3.464	3.464	3.504	3.504	4.67	4.71	5.96
2 (2)	3.429	3.929	3.929	4.294	4.294	10.877	10.909	12.159
7 (3)	5.181	5.681	6.931	6.971	7.377	8.96	9	9
4 (3)	9	9.5	10.75	10.79	10.79	12.373	12.413	12.413

Table 3
Last two rows of the dynamic setting.

Order (F)	Forklift available	Order starts	Leaves depot	Arrives at SWZ	Enters SWZ	Leaves SWZ	Arrives at depot	End of order
4 (3)	5.181	5.681	6.931	6.971	6.971	8.554	8.594	8.594
7 (3)	8.594	9.094	10.344	10.384	10.384	11.967	12.007	12.007

defined by our case. To create this set we have programmed an instance generator capable of generating instances similar to our example.

5.1. Instance generator

In the following, we describe the procedure we have devised to generate test instances. The structure of a warehouse of the type described in Section 2, the set of orders to be fulfilled and the distributions that describe the behaviour of the times spent in performing the different logistic warehouse operations can be completely described by specifying the values of a set of parameters. To generate an instance we choose a value for each parameter following a procedure that involves randomly generating a number from a range of possible values. These ranges have been fixed based on real data of the considered warehouse, so that realistic instances of different complexity are generated. In what follows we introduce each parameter and the procedure we use to randomly assign a value to it.

ncaisles and *nsaisles* denote the number of cross and storage aisles, respectively. Their ranges of possible values are {3,4,5} and {6,8,10} respectively.

For each section, *ncolumns* denotes the common number of columns of all racks in the section. For each rack, *nlevels* denotes the number of levels of the rack. The set of integer numbers belonging to the intervals [12,18] and [5,7] are the ranges of possible values of *ncolumns* and *nlevels*, respectively.

For each level of each rack, *levelheight* denotes the common height of the storage locations on that level, expressed in centimetres. Its possible values are {125,145,155,180,240}.

For a given side of a storage aisle, *sldepth* denotes the common depth of all storage locations in that side of the storage aisle. The value of *sldepth* can be either 2 or 4.

For a given pallet, *palletheight* denotes its height. Its range of possible values is {60,70,80,90,100,110,120,130,140}.

We have fixed the number of forklift types, *nforklifttypes*, as 4 for all instances, of the types described in Section 2. The total number of forklifts, *totalnforklifts*, is generated as a percentage, *%nWZ*, of the number of WZ ($nWZ = nsaisles * (ncaisles - 1)$) with a minimum of 4: $totalnforklifts = min(4, \text{int}(\%nWZ * nWZ/100))$. The number of subworking zones is the number of working zones divided by two. As traffic congestion inside the warehouse may influence the performance of the procedures we want to test, we have considered two different intervals, [30,40] and [40,50], from which the value of *%nWZ* may be randomly generated. The procedure to assign forklifts to forklift types is the following:

1. Assign one forklift to each type.
2. Assign a percentage, randomly generated from the interval [25,40], of the not yet assigned forklifts to Type 4 (forklifts that can work at any level and any depth).
3. Each not yet assigned forklift is randomly assigned to a type.

We have used the following procedure to generate the initial state of the warehouse (which pallets are stored in which positions of which locations). The procedure proceeds location by location and for a given location works as follows:

1. With probability *%occupancy*, it is decided whether the given location stores a pallet or not. If the decision is no, the consideration of the current location stops. If the decision is yes, we store in position 1 a pallet whose height has been randomly chosen among the set of pallet heights which are less or equal than the height of the given location. With probability *%ontop* it is decided whether the stored pallet can have another one on top of it or not. For both parameters, the value is 50%.
2. If the pallet in position 1 can have another pallet on top and the free height on top of it is greater or equal to 60 cm, it is decided with probability *%occupancy* whether a new pallet is stored in position 2 or not. In the affirmative case, we store in position 2 a pallet whose height has been randomly chosen among the set of pallet heights which are less or equal than the free height above the pallet in position 1.
3. With probability *%occupancy*, it is decided whether a new pallet is stored in position 3 or not. If the decision is no, the consideration of the current location stops. If the decision is yes and the position 2 is occupied, we store a pallet in position 3 following the same rules as for position 1. If position 2 is empty, the height of the pallet stored in position 3 must be greater than the free height above the pallet in position 1.
4. To store or not a pallet in position 4 we proceed similarly as with pallet 2 with just one extra consideration: If position 2 were empty and the chosen pallet could be feasibly stored in position 2, the new pallet would be stored in position 2 and position 4 would remain empty.

norders, *nrorders* and *nsorders* denote the number of total, retrieval and storage orders, respectively. *norders* is calculated as $factor * nrec$, where *factor* is an integer number randomly generated from the interval [80,90]. The aim of this way of computing *norders* is to generate order lists that can be fulfilled in approximately 6 h of work. *nrorders* is calculated as $nrorders = \text{int}(\%norders * norders/100)$ where *%norders* is an integer number randomly generated from the interval [45,55]. *nsorders* is calculated as $nsorders = norders - nrorders$.

Given a retrieval order, o_s, $l(o_s)$ and $pos(o_s)$ will be randomly chosen among those already occupied in the warehouse.

Only retrieval orders have due dates. Pallets are usually retrieved from the racks either to be loaded into a lorry or to satisfy production requirements. Therefore, it seems more realistic to assign due dates to groups of orders (all the orders in the same group will have the same due date) rather than to individual orders.

The groups of retrieval orders are generated one by one. The size of a group is an integer number randomly extracted from the interval $[1, \min(rest, nrorders/4]$ with a maximum value of 64 (this figure is related with the maximum capacity of the lorries) where *rest* is the number of retrieval orders which have no due date assigned yet. Initially, $rest = nrorders$.

Given a group of retrieval orders, the common due date *FF* assigned to all the orders in the group is generated as an integer number randomly chosen from the interval $[mintight * averagetime, maxtight * averagetime]$ where *mintight* and *maxtight* are factors that control the tightness of the due dates. Half of the instances are generated with values $(mintight, maxtight) = (0.1, 0.5)$ and the other half with values $(mintight, maxtight) = (0.15, 0.55)$. Also, $averagetime = (\text{int})(norders * average\_order\_fulfilment\_time/totalnforklifts)$ where *average_order_fulfilment_time* is an estimation of the average time needed to perform an order. It is calculated as the sum of the averages of the depot and travel times plus an estimation of the average of the storage/retrieval times.

The storage orders are also generated in groups. The number of groups is an integer number randomly generated from the interval $[1, nsorders]$. The *nsorders* storage orders are randomly assigned to the groups with a minimum of one order per group. All the pallets referred to by the storage orders in a group have the same height and the same maximum level at which they can be stored. Both parameters are randomly chosen among the set of possible values. Also, it is probabilistically decided with probability *%ontop* whether the pallets can have another one on top of them or not. Given a storage order, the location and position where the pallet has to be stored is randomly chosen among those which can feasibly store the pallet.

Table 4
Parameters of the generator and their ranges.

Parameter	Range	Parameter	Range
ncaisles	3,4,5	*nforklifttypes*	4
nsaisles	6,8,10	*%nWZ*	[30,40],[40,50]
nlevels	[5,7]	*%occupancy*	50%
ncolumns	[12,18]	*%ontop*	50%
levelheight	125,145,155,180,240	*factor*	[80,90]
sldepth	2,4	*%norders*	[45,55]
palletheight	60,70,80,90,100,110,120,130,140	*mintight-maxtight*	0.1–0.5 or 0.15–0.55

Table 4 shows the input parameters of the generator and their ranges of possible values.

The complexity of the instances generated by the aforementioned generator is mostly influenced by the parameters *ncaisles*, *nsaisles*, *%nWZ*, *mintight-maxtight*. They have either two or three options in their ranges. For each combination of the possible options ($3 \times 3 \times 2 \times 2 = 36$), we have randomly generated 5 instances making a total of 180 instances of different complexity that constitute the test instance set that have been used in the computational experiments which will be presented in the next section.

In order to perform the simulations it is necessary to make a reliable analysis of the times required in the different logistic actions taking place in a warehouse. The operation times in minutes have been estimated from a real warehouse of a Spanish company. As we have already said we consider that the time to fulfil an order, $t(o_i)$, is calculated as the sum of several main times: the travel time, the waiting time, the depot time and the storage/retrieval time. Each of the above times is obtained as the sum of the durations of several more elementary actions and these durations are random variables whose probability distributions are known. The elementary actions and the probability distributions of their durations are described next.

To compute the travel time we have considered two elementary actions: to traverse an arc for which at least an extreme vertex represents a column of storage locations and to traverse any other arc. The durations of both actions are uniformly distributed between 0.006 and 0.01, and between 0.012 and 0.02, respectively.

We have assumed that all other durations are computed as: $duration = 0.8\mu + \text{Exp}(\mu - 0.8\mu)$ where $\mu > 0$ may be different for each elementary operation. Notice that *duration* can never be equal to zero and the mean value of *duration* is always μ.

It is assumed that $\mu = 1/3$ for the waiting time. To compute the depot time we have considered four elementary times: the time needed to assimilate the new order ($\mu = 0.5$), the time needed to manually identify a pallet ($\mu = 0.25$), the time needed to automatically identify a pallet (1/60) and the time needed to deposit or remove a pallet from the ground ($\mu = 1$).

To compute the storage/retrieval time we have considered the following elementary actions: Once a forklift gets the prescribed storage location it has to be properly positioned in front of the storage location ($\mu = 0.5$). The durations of the manual and automatic pallet identifications are distributed as in the case of the depot time. Once the forklift has been properly positioned the forks have to be lifted up to the proper level, the pallet has to be either retrieved or stored and the forks have to be brought down. The value of μ for this compound operation depends on the level: level 1 ($\mu = 0.083$), levels 2 or 3 ($\mu = 0.5$) and level 4 or greater ($\mu = 1$). When the storage location is not directly accessible some pallets (the pallet carried on the forklift and one or more pallets already stored) have to be temporally deposited on the ground and picked up afterwards. The value of μ for any of these two operations is $\mu = 0.167$. After the pallet has been stored or retrieved from the storage location the forklift has to manoeuvre to a position that allows it to traverse the aisle ($\mu = 0.083$).

5.2. Computational experiments

In this subsection, we present the computational experiments carried out to analyse: (1) the performance of the different priority rules devised and (2) the benefits of using the RFID technology. The algorithms have been coded in C and the tests have been carried out on an Intel Core Duo 3.16 GHz 4 GB RAM personal computer. The algorithms have been run on the 180 test instance set described above. Each algorithm has been run 30 times on each test instance. *Aver_tardiness* (*Aver_flow_time*) denotes the average of the tardiness (flow times) over all instances and all execution repetitions, measured in minutes.

5.2.1. Priority rule performance analysis in non-RFID environments

In this subsection, we present the computational results obtained in the computational experiments carried out to analyse the performance of the priority rules introduced in Section 4.3 when applied in the order management environments SBC and DBC. Regarding environment DBC, we have tested all possible ordered pairs of priority rules, considering the first rule as the primary rule and the second as the secondary rule. However, for simplicity, we only present the results corresponding to the best pair and also those corresponding to pairs that yield interesting results. To study the significant differences between algorithms we have used the sign test (see Dixon & Mood, 1946; Fisher & Boyd, 1925). Almost all differences turned out to be significative and we will point out the ones that are not.

Tables 5 and 6 show the values of *Aver_tardiness* and *Aver_flow_time* for the environments SBC and DBC, respectively. In Table 5, the priority rules are named by means of the names of their primary rules as their secondary rules are the same.

Regarding the SBC environment, the priority rule *duration balance* yields the best results with respect to both *Aver_tardiness*

Table 5
SBC environment.

Priority rule	Aver_tardiness	Aver_flow_time
duration balance	1453.25	391.58
# orders balance	1594.21	399.86
random	11215.07	574.29

Table 6
DBC environment.

Primary rule	Secondary rule	Aver_tardiness	Aver_flow_time
SWZ congestion	dd	745.45	357.55
SWZ congestion	random	31702.68	400.32
dd	SWZ congestion	812.30	360.73
dd	random	1392.18	385.02
dd	sub-utilisation	1485.90	381.31
sub-utilisation	random	6822.18	381.90
random	random	36021.47	423.00

and *Aver_flow_time*. Notice that rule *dd* with *random* as a tie-brake rule yields a very high value of *Aver_tardiness* (11215.07).

Regarding the DBC environment, the priority rule *(SWZ congestion,dd)* yields the best results with respect to both *Aver_tardiness* and *Aver_flow_time*. It is interesting to notice that the rule *SWZ congestion* alone, *(SWZ congestion,random)*, yields very bad results and also that the *SWZ congestion* concept is not applicable in an SBC environment. The difference between (dd,random) and (dd,sub-utilisation) is not significative. Notice also that the values of *Aver_flow_time* for the different priority rules do not differ too much. However, all differences are significant, except for (dd,random) with (dd,sub-utilisation) and for (sub-utilisation,random) with (dd,random). It is also interesting to notice that the *Aver_tardiness* values for the rules (*dd,random*) and (*dd,sub-utilisation*) (1392.18,1485.90) are similar to the best value of *Aver_tardiness* for the SBC environment (1453.25). In both cases, the statistical test significantly favours the dynamic case. At any rate, this similarity seems to indicate that just changing from a static to a dynamic policy does not guarantee an important improvement of *Aver_tardiness*. On the contrary, it seems necessary to devise new priority rules that dynamically take advantage of the current status of the warehouse.

5.2.2. Using the RFID technology

In this subsection, we present the computational results obtained in the computational experiments carried out to analyse the benefits that can be drawn from the RFID technology in the context of storage/retrieval operations management.

The experiments have consisted of applying the same rule, *duration balance*, ((*SWZ congestion,dd*)) in the order management environments SBC, SRFID1 and SRFID2 (DBC, DRFID1 and DRFID2). To perform a deeper analysis of the computational results, we have partitioned the test instance set into four subsets of instances of a priori increasing difficulty. To carry out this classification, we have computed a *difficulty index* for each instance. The *difficulty index* is computed as follows: Given an instance, we apply 30 times the rule *duration balance* in the order management environment SBC, compute the average of the tardiness over all execution repetitions (this figure will be called the *tardiness of the instance* in environment SBC) and divide this figure by the number of orders in the instance. We have partitioned the instance set into four subsets that group together the instances in which *difficulty indexes* belong to the intervals [0,0.1], (0.1,1], (1,4] and >4, respectively. The subsets receive the names of the intervals used in their definitions. Although we could have chosen different intervals to partition the instance set we believe that our selection is meaningful as the *Aver_tardiness* for the four instance subsets are of different orders of magnitude. We have chosen SBC as the reference not only because this is the environment that yields the worse results but also because it is the order management environment more frequently used in manually run warehouses.

Table 7 shows the values of *Aver_tardiness* for the four instance subsets and the six order management environments previously mentioned. The number of instances in each subset is denoted by #instan.

We can see in Table 7 that within each column *Aver_tardiness* increases from top to bottom. Broadly speaking, this means that the due dates are more difficult to satisfy as the *difficulty index* increases. We can also observe that if an environment yields a better result than another environment for a given subset then it also yields better or equal results for the other subsets. Therefore, we can broadly order the environments from less efficient to more efficient in the following way: SBC > SRFID1 > DBC > DRFID1 > SRFID2 > DRFID2. The differences between some pairs are not significative, especially in the interval [0,0.1]. Namely, we cannot distinguish between algorithms DBC and SRFID1, DRFID1 and SRFID2, DRFID2 and SRFID2 in that interval, and between DRFID1 and SRFID2 in interval (0.1,1]. The rest of the comparisons in the table are significative. According to these results, the dynamic environment DBC is significantly more efficient than the static environment SBC. Also, the efficiency clearly improves using the RFID technology reaching the greatest efficiency when both racks and pallets are tagged. It is interesting to note that DRFID1 > SRFID2, at least with bigger tardiness. This seems to indicate that the benefits of working on a dynamic instead of a static setting are outperformed when both racks and pallets are RFID tagged instead of only racks. It is worth noting that the tardiness is null for 106 instances in environment DRFID2. Also that the maximum value of the tardiness in environments SBC and DRFID2 are 15977.89 and 3774.8, respectively.

We now present the tardiness computational results as *percentage improvements* with respect to SBC. Given an environment *X*, where *X* can be any of the environments SRFID1, SRFID2, DBC, DRFID1 or DRFID2, and given a subset *Y*, where *Y* can be any of the subsets [0,0.1], (0.1,1], (1,4] or >4, the *percentage improvement* relative to *X* and *Y* is computed as follows: For each instance in *Y*, the *tardiness of the instance* in environments SBC and *X* are computed and the improvement in percentage of the *tardiness of the instance* in environment *X* with respect to that in environment SBC is also computed. The average of these figures over all *non-SBC-dd-satisfying* instances in *Y* is called the *percentage improvement* with respect to SBC relative to *X* and *Y*. An instance in which *tardiness* in environment SBC is less than 1 is called a *SBC-dd-satisfying* instance. Table 8 shows these *percentage improvements* for all possible values of *X* and *Y*. It is worth noting that there are 23

Table 7
Average tardiness by instance subset.

	#Instan	SBC	SRFID1	SRFID2	DBC	DRFID1	DRFID2
[0,0.1]	56	16.72	1.81	0.09	1.14	0.09	0.00
(0.1,1]	53	414.43	129.74	7.36	56.58	13.29	0.38
(1,4]	43	1740.19	916.30	177.93	709.17	320.07	50.82
>4	28	5851.99	4178.36	1919.54	3593.73	2433.78	961.54

Table 8
Tardiness percentage improvements with respect to SBC by instance subset.

	SRFID1 (%)	SRFID2 (%)	DBC (%)	DRFID1 (%)	DRFID2 (%)
[0,0.1]	89.60	99.80	88.37	98.71	100.00
(0.1,1]	72.53	98.62	86.68	96.71	99.92
(1,4]	48.37	90.40	59.13	81.81	97.11
>4	29.71	69.66	37.08	57.91	84.61

Table 9
Flow time percentage improvements with respect to SBC by instance subset.

	SRFID1 (%)	SRFID2 (%)	DBC (%)	DRFID1 (%)	DRFID2 (%)
[0,0.1]	6.39	16.98	9.60	14.92	24.59
(0.1,1]	6.26	16.92	8.91	14.25	24.08
(1,4]	6.20	16.75	8.25	13.57	23.43
>4	6.03	16.41	7.52	12.80	22.43

Table 10
Average total tardiness improvement in dynamic versus static environments by instance subset.

	DRFID1 versus SRFID1				DRFID2 versus SRFID2			
	Non-dd-satisfying		dd-satisfying		Non-dd-satisfying		dd-satisfying	
	#inst	Imp (%)	#inst	Stat/dinam	#inst	Imp (%)	#inst	Stat/dinam
[0,0.1]	13	88.99	43	0.151/0.04	2	100.00	54	0.013/0
(0.1,1]	53	90.92	0	–	28	92.71	25	0.258/0.0068
(1,4]	43	67.76	0	–	43	85.30	0	–
>4	28	41.42	0	–	28	56.20	0	–

SBC-dd-satisfying instances in subset [0,0.1] and none in the other subsets. Also, that the *tardiness* of any of those 23 instances in any environment different from SBC has been always zero. We can also observe that the *percentage improvement* decreases as the *difficulty index* increases. Notice that the efficiency order among the environments induced by the data in Table 8 is the same as that induced by the data in Table 7.

Table 9 shows the flow time computational results as percentage improvements with respect to SBC. These percentage improvements have been computed similarly as in the case of the tardiness but all the 180 test instances are now included in the computations. As we can see, the percentage improvements obtained by only RFID tagging the racks are about a 5% or 6%. Tagging the pallets as well implies a further 10% improvement. We can observe that the *flow time percentage improvement* decreases as the *difficulty index* increases. Also that if we order the environments in increasing *flow time percentage improvement* we obtain the same order as before.

It is also interesting to compare the tardiness obtained in dynamic and static environments. The left part of Table 10 compares the tardiness computational results in environment DRFID1 with those in environment SRFID1. We have distinguished between *non-SRFID1-dd-satisfying* (column Non-dd-satisfying) and *SRFID1-dd-satisfying* (column dd-satisfying) instances. For the first group of instances we have indicated the number of *non-SRFID1-dd-satisfying* instances that belong to each instance subset (column #instan) and the tardiness *percentage improvement* with respect to SRFID1 relative to DRFID1 by instance subset (column Imp). For the second group of instances we have indicated the number of *SRFID1-dd-satisfying* instances that belong to each instance subset (column #inst) and the tardiness obtained in environments SRFID1 and DSRFID1 (column Stat/dinam). Similarly, the right part of Table 10 compares the tardiness computational results in environment DRFID2 with those in environment SRFID2. Table 10 clearly shows the benefits of working in a dynamic instead of a static environment whether only the racks are tagged or whether both the racks and the pallets are tagged. It is also interesting to note that there are 43 and 79 *SRFID1-dd-satisfying* and *SRFID2-dd-satisfying* instances, respectively. It is worth noting that if the *difficulty index* is computed with respect to the environment DRFID2 then 146, 23, 10 and 1 instances have their difficulty indexes in the intervals [0,0.1], (0.1,1], (1,4] and >4, respectively. Therefore, there are 169 *DRFID2-dd-satisfying* instances.

6. Conclusions

In this paper we have explored the impact of the RFID technology on the management of the storage/retrieval operations in a warehouse. We have considered six order management environments as the result of combining two types of policies (static, dynamic) with three technological environments (no RFID tags, RFID tags are attached on racks, RFID tags are attached on racks and pallets) in all possible ways. A solution for the problem has consisted of an *order management policy* that, given a set of orders to be fulfilled, indicates a sequence of order fulfilment and assigns a route and a forklift for each order. The objective has been the minimisation of the expected tardiness. We have introduced a set of order management policies whose performances have been compared via simulation on a set of randomly generated realistic test instances of different complexity. Given an instance and an order management policy, the developed simulator simulates the fulfilment of the storage/retrieval orders by simulating the movements of the forklifts inside the warehouse and making decisions at certain decision times according to the given policy.

A first conclusion that can be drawn from this work is that the dynamic order management environments are significantly more efficient, as far as tardiness and total flow time is concerned, than the static ones mainly because a dynamic environment allows us to incorporate information about the current status of the warehouse into the decision making process. In particular, the priority rules that take into account the concept of expected congestion in the working zones and the due dates yield the best results.

Regarding the RFID technology the main conclusion is that using this technology instead of a technology that requires human intervention to identify and locate items greatly improves the efficiency of an order management policy. We can conclude also that RFID tagging both racks and pallets is more efficient than tagging only racks. In fact, we consider that applying a dynamic policy in a warehouse where both racks and pallets are RFID tagged is an excellent option in practice.

Another issue that we are considering as a future work is to investigate the benefits of also attaching RFID tags on the warehouse floor at the depot, at the aisle intersections and in front of the columns. These additional tags would allow the automatic identification and location of the forklifts inside the warehouse in real time. This information could be used to dynamically assign routes to the forklifts taking into account the current status of the warehouse, for example, taking into account the current working zone congestion.

Acknowledgements

This research was partially supported by the Ministerio de Ciencia e Innovación under contract DPI2007-63100 and under contract MTM2011-23546.

References

Caron, F., Marchet, G., & Perego, A. (2000). Layout design in manual picking systems: A simulation approach. *Integrated Manufacturing Systems, 11*, 94–104.

Chao-Hsien Pan, J., & Wu, M.-H. (2012). Throughput analysis for order picking system with multiple pickers and aisle congestion considerations. *Computers & Operations Research, 39*, 1661–1672.

Chow, H. K. H., Choy, K. L., & Lee, W. B. (2007). dynamic logistics process knowledge-based system – An RFID multi-agent approach. *Knowledge-Based Systems, 20*(4), 357–372.

Chow, HKH., Choy, KL., Lee, WB., & Lau, KC. (2006). Design of a RFID case-based resource management system for warehouse operations. *Expert Systems with Applications, 30*(4), 561–576.

De Koster, R., Le-Duc, T., & Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research, 182*(2), 481–501.

Dixon, W. J., & Mood, A. M. (1946). The statistical sign test. *Journal of the American Statistical Association, 41*, 557–566.

Fisher, R. A. (1925). *Statistical methods for research workers*. Edinburgh: Oliver and Boyd.

Fosso, S., & Chatfield, A. T. (2010). RFID-enabled Warehouse Process Optimization in the TPL Industry. In *Proceedings of the 43rd Hawaii international conference on system sciences*.

García, A., Chang, Y. S., & Valverde, R. (2006). Impact of new identification and tracking technologies on a distribution center. *Computers & Industrial Engineering, 51*(3), 542–552.

Gu, J., Goetschalckx, M., & McGinnis, L. F (2007). Research on warehouse operation: A comprehensive review. *European Journal of Operational Research, 177*, 1–27.

Gu, J., Goetschalckx, M., & McGinnis, L. F (2010). Research on warehouse design and performance evaluation: A comprehensive review. *European Journal of Operational Research, 203*, 539–549.

Ho, S.S., & Sarma, S. (2009). The fragmented warehouse: Location assignment for multi-item picking. In *2nd International Logistics and Industrial Informatics, 2009. LINDI 2009*. (pp. 1–6).

Hsieh, L.-F., & Tsai, L. (2006). The optimum design of a warehouse system on order picking efficiency. *International Journal of Advanced Manufacturing Technology, 28*, 626–637.

Kolisch, R. (1994). *Project scheduling under resource constraints*. Spronger-Verlag Company.

Liao, W.-P., Lin, T. M. Y., & Liao, S.-H. (2011). Contributions to Radio Frequency Identification (RFID) research: An assessment of SCI-, SSCI-indexed papers from 2004 to 2008. *Decision Support Systems, 50*, 548–556.

Lim, M. K., Bahr, W., & Leung, S. (2013). RFID in the warehouse: a literature analysis (1995–2010) of its applications, benefits, challenges and future trends. *International Journal of Production Economics* (in press).

Neumann, K., & Schwindt, C. (2002). Project scheduling with inventory constraints. *Mathematical Methods of Operations Research, 56*, 513–533.

Ngai, E. W. T., Moon, K. K. L., Riggins, F. J., & Yi, C. Y. (2008). RFID research: An academic literature review (1995–2005) and future research directions. *International Journal of Production Economics, 112*, 510–520.

Oliveira, J. A. (2007). Scheduling the truckload operations in automatic warehouses. *European Journal of Operational Research, 179*, 723–735.

Poon, T. C., Choy, K. L., Chan, F. T. S., Ho, G. T. S., Gunasekaran, A., Lau, H. C. W., et al. (2011). A real-time warehouse operations planning system for small batch replenishment problems in production environment. *Expert Systems with Applications, 38*, 8524–8537.

Poon, TC., Choy, KL., Chow, HKH., Lau, HCW., Chan, FTS., & Ho, KC. (2009). A RFID case-based logistics resource management system for managing order-picking operations in warehouses. *Expert Systems with Applications, 36*(4), 8277–8301.

Roodbergen, KJ., & De Koster, R. (2001). Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research, 39*(9), 1865–1883.

Roodbergen, K. J., Sharp, G. P., & Vis, I. F. A. (2008). Designing the layout structure of manual order picking areas in warehouses. *IIE Transactions, 40*, 1032–1045.

Sarac, A., Absia, N., & Dauzère-Pérès, S. (2010). A literature review on the impact of RFID technologies on supply chain management. *International Journal of Production Economics, 128*, 77–95.

Visich, J. K., Li, S., Khumawala, B. M., & Reyes, P. M. (2009). Empirical evidence of RFID impacts on supply chain performance. *International Journal of Operations & Production Management, 29*, 1290–1315.

Wang, H., Chen, S., & Xie, Y. (2010). RFID-based digital warehouse management system in the tobacco industry: A case study. *International Journal of Production Research, 48*, 2513–2548.

Zäpfel, G., & Wasner, M. (2006). Warehouse sequencing in the steel supply chain as a generalized job shop model. *International Journal of Production Economics, 104*, 482–501.