Computers & Industrial Engineering 63 (2012) 1003–1012

Contents lists available at SciVerse ScienceDirect

Computers & Industrial Engineering

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

Toward perpetually organized unit-load warehouses

Héctor J. Carlo [a,*], Germán E. Giraldo [b]

[a] Industrial Engineering Department, University of Puerto Rico – Mayagüez, Call Box 9000, Mayagüez, PR 00681, United States
[b] Projects Unit, Colombian School of Engineering, Bogotá, Colombia

ARTICLE INFO

Article history:
Received 19 September 2011
Received in revised form 11 May 2012
Accepted 22 June 2012
Available online 3 July 2012

Keywords:
Warehouse
Storage location assignment
Optimization
Facilities planning and design
Facility logistics

ABSTRACT

Determining the optimal storage assignment for loads in a unit-load warehouse has been extensively addressed in the facility logistics literature. However, the process of implementing a particular storage assignment given the current assignment of loads has not received much attention. Typically, unit-load warehouses use downtime or overtime to remove loads from their current location and move them to the suggested location. This study presents *Rearrange-While-Working* (RWW) as a strategy to optimize the process of rearranging a warehouse while serving a list of move requests. This study examines three scenarios: (1) one empty location in the warehouse and the material handling equipment (MHE) is idle; (2) one empty location in the warehouse under the RWW strategy; and (3) when there are multiple empty locations in the warehouse under RWW. In the first scenario, the MHE can make any movement desired as it is idle. For the other two scenarios it is assumed that the MHE is not idle so loads can only be moved when requested to perform a move request. Due to the complexity of the problems, several heuristics are proposed. Experimental results indicate that the proposed heuristics perform satisfactorily in terms of solution quality and computational time.

1. Introduction

An important operational decision in unit-load warehouses is to determine the best storage location for each product in order to minimize the total material handling effort (or cost). This problem is known in the literature as the storage location assignment problem (SLAP) (Hausman, Schwarz, & Graves, 1976). In general, given a storage assignment policy the SLAP can be easily solved. Gu, Goetschalckx, and McGinnis (2007) and Roodbergen and Vis (2009) provide overviews of the most common storage assignment policies and the existing solution approaches for the SLAP. In this paper, we assume a full-turnover-based dedicated storage assignment policy (Roodbergen & Vis, 2009), where there is only one load (i.e. pallet) of each product.

Typically, when loads are first stored in the warehouse, the SLAP is solved to optimize the assignment of loads (or products) to storage location. However, it is inevitable that the demand profiles change over time due to competition, introduction of new products, product maturity, and seasonality, among others. The initial storage location assignment used eventually becomes inefficient as the demand changes. Typically, at some point companies will identify some downtime (idle time or overtime) to reorganize the warehouse by repositioning items according to the solution of the SLAP with the updated demand. In this paper we propose the *Rearrange-While-Working* (RWW) strategy to methodically reorganize the warehouse while serving a set of move requests (without requiring idle time). The basic idea in RWW is to consider all available empty storage locations when (re-)storing a load.

Although the work presented in this paper applies to any general warehouse with one material handler, for convenience, we assume that the material handling equipment (MHE) is an end-of-aisle Automated Storage/Retrieval System (AS/RS) or a Miniload AS/RS. The system is composed of a single storage/retrieval (S/R) machine that has a pickup and deposit point that coincides with a (split-case) picking workstation. In other words, the retrieved loads (i.e. pallets) are taken to a workstation where a picker will take the required amount of products from the unit-load while the AS/RS waits. Once the picking is completed, the AS/RS will store the current load and retrieve the next load according to pre-determined sequence in an order list. The AS/RS is assumed to operate under a dual command cycle. Hence, the AS/RS will first store a load, travel empty to the location of the next load to be retrieved, and retrieve this load.

In RWW, instead of returning each load to its original location, a decision is made to determine the best (open) storage location to which the load should be assigned. The strategic reassignment of loads is made to methodically rearrange the loads until the (updated) SLAP-optimal solution is obtained. In RWW, when a load is retrieved to fulfill a particular order, its assigned storage location is permanently changed to systematically update the rack arrangement. We

* Corresponding author.
E-mail addresses: hector.carlo@upr.edu (H.J. Carlo), german.giraldo@escuelaing.edu.co (G.E. Giraldo).

assume that the initial (current) and final (*i.e.* optimal SLAP) storage location assignments are known. Hence, the problem is to rearrange the warehouse from an initial arrangement to a final (SLAP-optimal) arrangement by repositioning the loads as they are required by the picker.

The remainder of this paper is organized as follows: Section 2 presents a review of the pertinent literature. In Section 3 presents a more detailed description of RWW and the modeling assumption made. Section 4 considers repositioning loads with an idle AS/RS with only one empty location. Section 5 examines the RWW strategy for an AS/RS with only one empty location. Section 6 considers the RWW strategy with multiple empty locations. Section 7 revisits the modeling assumptions made. Lastly, Section 8 presents the conclusions and future work.

2. Literature review

The SLAP can be classified according to the amount of information that is known about the arrival and departure of the loads stored in the warehouse: (1) item information (SLAP/II), (2) product information (SLAP/PI), or (3) no information (SLAP/NI) (Gu et al., 2007).

The SLAP/II problem assumes that the complete information about the arrival and departure times of individual items is known. A commonly used policy for the SLAP/II is the Duration-of-Stay (DOS) policy where items with the expected shortest visit are assigned to the most desirable locations (Goetschalckx & Ratliff, 1990).

In the SLAP/PI the information available is at the product level (instead of the item level, where items are instances of products). At the same time, products may be classified into product classes, typically according to physical characteristics or requirements. The SLAP/PI seeks to assign product classes to storage locations. After product classes are assigned to storage locations, the item location within its class is determined by simple rules (e.g. randomly). As described by Hausman et al. (1976), the special case where the number of classes equals the number of products (n) is called *Dedicated* (or Fixed Slot) *Storage* as each product would have a specific set of storage locations for storage. If there is only one class, the storage policy is referred to as *Randomized* or *Floating Slot* as any item could go to any storage location. On the other hand, when the number of classes is between two and $n - 1$, it is known *Class-Based Storage*. A commonly used policy for the SLAP/PI is to assign classes with small cube-per-order index (COI) to the most desirable locations (Heskett, 1963).

In the SLAP/NI, no information is available on the characteristics of the arriving items. Hence, only simple storage policies can be developed (e.g. Closest-Open-Location). For a survey of the literature pertaining the SLAP and the reader is referred to Jeroen and Van den Berg (1999), Gu et al. (2007), and Roodbergen and Vis (2009).

Besides the classification of the SLAP problem according to the amount of information known, the SLAP can be further classified as static or dynamic. In the static version of the problem the material flows are assumed constant over the planning horizon. On the other hand, the dynamic version continuously adjusts storage assignments based on material flows. Most of the existing literature focuses on the static version of the problem (Gu et al., 2007). An interesting compromise was proposed by Christofides and Colloff (1973) (i.e. *warehouse rearrangement*), Linn and Wysk (1990a, 1990b) (i.e. *restoring policy*), and Muralidharan, Linn, and Pandit (1995) (i.e. *shuffling*). The basic idea in all of these studies is to reposition loads during idle times in order to actualize the storage location assignment. Christofides and Colloff (1973) proposed a two-stage algorithm to sequence load movements to minimize the material handling effort required to rearrange the products in a dedicated warehouse. Linn and Wysk (1990a, 1990b) suggested a restoring policy for AS/RSs using Class-Based Storage where idle times are used to move fast-moving items closer to the input/output point (I/O). The authors do not provide details or computation results. Muralidharan et al. (1995) formulated the problem of updating the warehouse configuration under Class-Based Storage for AS/RSs as a Precedence Constrained Selective Asymmetric Travelling Salesman Problem. It is assumed that the S/R becomes idle for a limited amount of time in which reshuffling is performed. The objective is relocated loads to their corresponding classes in order to maximize the total weighed savings in terms of service time (referred to as profit). Given the computational complexity of the problem, the authors proposed two heuristics: Shuffling with Nearest Neighbor Heuristic (SNN) and Shuffling with Insertion (SI). Based on simulation results the authors conclude that using idle times to update the warehouse (AS/RS) configuration increases the AS/RS operating efficiency. Clearly, these (reshuffling) policies are not effective in highly utilized AS/RSs. It is worth mentioning that the proposed RWW strategy could be complemented by using these reshuffling policies during idle time.

Han, McGinnis, Shieh, and White (1987) and Lee and Schaefer (1996) study the problem of sequencing the retrievals where loads may be stored in any open location on the rack. The main objective of these studies was to find a sequence of AS/RS dual cycles that minimize the total travel time required to fulfill all the requests. There are two main differences between Han et al. (1987), Lee and Schaefer (1996) and our study are: (1) they allow order (or move request) sequencing while we assume move requests are performed according to a pre-determined sequence; (2) their objective is to minimize the total travel time to perform a set of move requests, while our objective is to reorganize the rack from its current arrangement to the SLAP arrangement. Han et al. (1987) and Lee and Schaefer (1996) pursue a short term objective (minimize travel time to perform a set of move requests) while our objective is longer term (maintain an organized rack).

Several authors have addressed the puzzle problem, where $n - 1$ tiles and one open space need to be organized in a specific pattern. Tiles can only be moved to the empty space, creating a new open space. Archer (1999) shows that there is no single solution to the problem of the 15-puzzle. Reinefeld (1993) shows a comprehensive assessment of the number of possible configurations for the 8-puzzle problem, where all possible movements are evaluated using a database. Gue and Kim (2007) presents a model for reorganization of articles in a shelf, based on the 15 pieces puzzle game (15-puzzle). Like the traditional 15-puzzle problem, an array of 15 numbers must be ordered numerically, taking as its starting point a random arrangement, where only adjacent locations articles can be mobilized to the empty location. The study shows results for an array of high-density storage, where only an empty space available for movement of items, and experimental results for multiple storage spaces. The problem with 1 empty space (*gap*) is solved by a simple heuristic. The problem with 2 or more empty spaces is solved by dynamic programming.

Other authors have addressed other problems related to reorganizing items for different applications which have some analogy to our problem. These applications include reorganizing containers in a container terminal (e.g. Kim & Bae, 1998), sorting vectors (e.g. Drozdek, 2005), and disk defragmentation (e.g. Texas Instruments Incorporated, 1995), among others.

3. RWW description and modeling assumptions

As discussed in Section 1, in the *Rearrange-While-Working* (RWW) strategy we seek to update the arrangement of a warehouse from the current arrangement to the SLAP-optimal arrangement while serving a list of move requests. In RWW, when a load is retrieved, its assigned storage location is permanently changed to systematically update the warehouse arrangement. We assume