International Journal of Management Science
Vol 17, No 2, November 2011

An Exact Splitting Algorithm for a 4-Class-Based Dedicated Linear Storage Problem

Moon Hee Yang[*]
Department of Industrial Eng., Dankook University

Chang Hwan Choi
Department of International Trade, Dankook University

Hee Kim
Department of Industrial Eng., Dankook University

(Received: October 15, 2010 / Revised: November 2, 2011 / Accepted: November 2, 2011)

ABSTRACT

In this paper, we address a layout design problem for determining an optimal 4-class-based dedicated linear storage layout in a class of unit load storage systems. Assuming that space requirement for a class is the sum of the maximum inventory levels of products assigned to the class, and that one-way travel time is a linear function of storage index, we formulate a 4-class-based dedicated linear storage problem PTL[4] and provide an exact splitting algorithm with $O(n\lceil \log n \rceil)$. Our algorithms could be applied to more than a 4-class-based dedicated storage layout problem with slight modification in order to reduce computational execution time.

Keywords: Storage system, Class-based Dedicated Storage Layout, AS/RS

1. Introduction

K-Class-based dedicated storage policy or simply K-class storage policy employs several zones in which lots from a class of products are stored randomly. Zones form a partition of storage locations. In designing a K-class-based storage system, one of design issues will be to maximize the throughput rate. Suppose that order quantities and demand rates of n products are given for storage and retrieval operations where

[*] Corresponding author, E- mail: myfriend@dankook.ac.kr

$1 \le K \le n$. In order to maximize the throughput rate, the first question will be naturally "How many classes should we use?" Suppose that the optimal number of classes is given. Then the remaining questions will be "How should we partition the products into classes?" and at the same time "How should we determine the zone size, i.e., the number of storage locations which should be reserved for a zone?" The difficulty for solving the class-based storage problem lies in that all of these questions should be investigated simultaneously. Tompkins and White (1984) pointed out that class-based storage with randomized storage within each class can yield both the throughput benefits of dedicated storage and the space benefits of randomized storage. Also they suggested that in order to achieve both benefits, three to five classes may be defined. However, in fact, there is no mathematical proof that three to five classes must be optimal in terms of the space and/or throughput benefits.

For the first time, Hausman (1976) addressed the layout design problem for determining a 2/3 class-based storage layout in a unit load AS/RS using their analytical results and simulation. They reported that, based on the EOQ model, an ABC curve, and a continuous square-in-time rack, significant reductions in the expected single and dual command travel time were obtainable from a class-based storage policy. Based on Hausman(1976)'s approach, many authors have been studied the class-based storage system mathematically. Assuming single command and rectangular racks, Kouvelis and Papanicolaou (1995) provided a statistical approach and the formula to obtain the optimal boundaries for a two-class-based rectangular-in-time AS/RS. Using the same assumption, to derive the optimal boundaries for a general n-class-based storage policy, Rosenblatt and Eynan (1989) suggested a simple recursive procedure and Van den Berg (1996) provided a dynamic programming. Assuming dual command and rectangular racks, Kouvelis and Papanicolaou (1995) provided the search procedure to derive optimal boundaries for zone sizing decisions and suggested to use their formulas in combination with the recursive procedure of Eynan and Rosenblatt (1993) for n classes.

On the other hand, Bazaraa (1975) formulated a quadratic assignment problem for minimizing dual command travel time in a dedicated storage system, which is a special case of class-based storage systems. Malmborg and Krishnakumar (1989) showed that this COI rule was optimal for person-aboard AS/RSs with respect to order picking costs if there were fixed inventory levels and a fixed balanced assignment of order pickers to items. Malmborg and Bhaskaran (1990) gave a proof of optimality for the COI (cube-per-order) index rule while taking into account the non-uniqueness of the COI layout if dual command scheduling was used in a dedicated storage system. However, Lee (1992) pointed that the COI rule could not be applied for to person-on-board systems due to the fact an order usually consisted of more than two independent items at different locations, and suggested a new heuristic that outperforms the COI rule. Assuming that the space requirement for a class is the maximum aggregated inventory level, Yang (2003) formulated a deterministic 2-class-based dedicated storage problem PTN[2] and conjectured that even PTN[2] is NP-hard. He provided an efficient heuristic algorithm as well as similar results found by those simulation models. Assuming that one-way travel time is a linear function of storage location index, Yang and Kim (2004) addressed a 3-class-based dedicated linear storage problem PTL[3] and provided an exact algorithm. Later Yang (2006) improved their algorithm in time complexity.

In this paper, using the same assumptions as Yang (2006), we deal with a deterministic 4-class-based dedicated linear storage problem. In section 2, for convenience to readers, we formulate a K-class linear storage problem PTN[K] generally. Relaxing one-way nonlinear travel time to linear, we formulate a deterministic K-class-based dedicated linear storage problem PTL[K] and give a necessary condition to PTL[K]. In section 3, based on the necessary condition and previous results, we provide an exact splitting algorithm with $O(n\lceil \log n \rceil)$, which can be used as an efficient heuristic algorithm to PTN[4]. Throughout this paper, in order to denote optimality for a decision variable, a superscript (*) will be used at the upper right side of each symbol.

2. Problem Statement and Fundamental Properties

Our storage system consists of M storage locations each of which accommodates only one unit load. The storage/retrieval operation is based on a K-class-based storage policy and within each zone random assignment rule (RAN rule) is used. The expected one-way travel time from a pick-up/deposit (P/D) station to storage location j is constant and given as t_j for j = 1, ⋯, M. If there n_p i/o points, by assuming that any pallet stored in location j travels to i/o point s with probability p_{sj}, t_j can be obtained as $t_j = \sum_{s=1}^{n_p} p_{sj} t_{sj}$ where t_{sj} is the one-way travel time from i/o point s to storage location j. Without loss of generality, it can be assumed that $t_1 \le t_2 \le \cdots \le t_M$.

An arriving replenishment lot of a product i, the size of which is r_i in unit load, contains a single product and are assigned randomly to open storage locations in one of K separate zones by using an storage/retrieval (S/R) machine or a lift-truck operator which or who carries only one unit load at a time. The average number of retrievals per unit time is given as d_i unit loads/unit time. We assume that retrievals are performed on first-in first-out basis.

Let C_k be the set (or class) of products assigned to zone k for k = 1, 2, ⋯, K. Then the average demand rate from zone k, D_k is obtained as $\sum_{i\in C_k} d_i$. Let A_k and $|A_k|$ be a set of storage locations assigned to zone k and the cardinality of a set A_k respectively. Given the t_j-nondecreasing ordering, we assign the first $|A_1|$ storage locations to A_1, and assign the next $|A_2|$ storage locations to A_2 and so on where $|A_k|$ denotes the cardinality of set A_k. Let T_k be the expected SC (Single Command) travel time to zone k. Then T_k can be expressed as

$$T_k = \frac{2}{|A_k|}\sum_{j\in A_k} t_j \tag{1}$$

Since the steady-state probability of visiting zone k is $\frac{D_k}{D}$, the expected SC travel time $E(SC_K)$ can be expressed as

$$E(SC_K) = \sum_{k=1}^{K} \frac{D_k}{D} T_k \tag{2}$$

where $D = \sum_{k=1}^{K} D_k$. Replacing T_k with Equation (2), $E(SC_K)$ can be further reduced to

$$E(SC_K) = \frac{2}{D}\sum_{k=1}^{K} \frac{D_k}{|A_k|}\sum_{j\in A_k} t_j = \frac{2}{D}\sum_{k=1}^{K} CAI_k \sum_{j\in A_k} t_j \tag{3}$$

where $CAI_k = \frac{D_k}{|A_k|}$, which will be called the CAI (class activity index) of class k. Since it is possible to view t_j as a cost for storage or retrieval of a pallet to or from storage location j, $E(SC_K)$ can be interpreted as the average cost for storing a pallet to an arbitrary location and retrieving the pallet to an i/o point. Hence, the minimization of $E(SC_K)$ is exactly equivalent to the minimization of the average cost for storage and retrieval of a pallet if t_j is replaced with a cost for storing in location j.

Even though C_k is given, the problem of determining the space requirement for class k, i.e., $|A_k|$, or equivalently minimizing the maximum aggregate stock level for multiple products is not easy. This problem is well known to be NP-hard (Hall, 1987). Probabilistically the zone size $|A_k|$ must be $\sum_{i\in C_k} r_i$ if no space shortage is allowed whenever replenishment lots arrive. Hence $|A_k|$ can be assumed to be $\sum_{i\in C_k} r_i$. This assumption implies that the total space requirement for all the classes is constant regardless of the number of classes and the particular partition, i.e., $\sum_{k=1}^{K}|A_k| = \sum_{i=1}^{n} r_i =$ R. Since a class contains at least one product, it is assumed that $|C_k|\geq 1$. Our K-class storage problem denoted by PTN[K] can be stated as follows:

PTN[K]: Given n products with $\{(r_i, d_i),\ i=1,\cdots,n\}$, storage locations with $\{t_j,\ j=1, 2,\cdots, M\}$, and an integer $K\leq n$, find an optimal solution $P_N^*(K)=\{C_1^*,\cdots,C_K^*\}$ such that we

$$\text{Minimize } Z_N(K) = \frac{2}{D}\sum_{k=1}^{K}\frac{D_k}{|A_k|}\sum_{j\in A_k} t_j$$

subject to $|C_k|\geq 1$.

$$D_k = \sum_{i\in C_k} d_i \qquad \text{for } k = 1, 2, \cdots, K$$

$$|A_k| = R_k = \sum_{i\in C_k} r_i \qquad \text{for } k = 1, 2, \cdots, K$$

It can be observed that the total enumeration algorithm requires $O(2^n)$ when K = 2. Yang (2003) conjectured strongly that even PTN[2] is NP-complete and provided a heuristic algorithm with $O(n^2)$ for PTN[2]. However, if we assume that one-way travel time is a linear function of storage index j, i.e., t_j = pj+q, j = 1, 2, ⋯, M for constants p and q, then Equation (1) can be reduced as

$$E(SC_K) = Z_L(K) = \frac{p}{D}\left\{\sum_{k=1}^{K-1}\sum_{m=k+1}^{K}(R_k D_m - D_k R_m)\right\} + p(R+1) + 2q \tag{4}$$

Thus PTN[K] can be relaxed to PTL[K] as follows.

PTL[K]: Given n products with $\{(r_i, d_i), i = 1, \cdots, n\}$, storage locations with $\{t_j = pj + q, j = 1, 2, \cdots, M\}$, and an integer $K \le n$, find an optimal solution $P_L^*(K) = \{C_1^*, \cdots, C_K^*\}$ such that we

$$\text{Minimize } Z_L(K) = \frac{p}{D}\left\{\sum_{k=1}^{K-1}\sum_{m=k+1}^{K}(R_k D_m - D_k R_m)\right\} + p(R+1) + 2q$$

subject to $|C_k| \ge 1$.

$$D_k = \sum_{i \in C_k} d_i \qquad \text{for } k = 1, 2, \cdots, K$$

$$|A_k| = R_k = \sum_{i \in C_k} r_i \qquad \text{for } k = 1, 2, \cdots, K$$

It can be observed that $Z_L(K)$ is independent of t_j. In order to facilitate our analysis, define PAI(i) to be d_i/r_i, which will be called the product activity index of product i. Define a PAI-nonincreasing ordering denoted by O = {1, 2, ⋯, n} to be an ordering which satisfies $d_i/r_i \ge d_{i+1}/r_{i+1}$ for i = 1, 2, ⋯, (n-1). For a given O, define $\mathrm{CAI}(a, b) = \sum_{i=a}^{b} d_i / \sum_{i=a}^{b} r_i$ for two positive integers a and b where $a \le b$.

When K = 2, Yang (2003) provided a necessary and sufficient condition to PTL[2] as follows and provided a greedy exact algorithm with $O(n)$ based on his condition.

Proposition 1: (Necessary and sufficient condition to PTL[2]) $P_L^*(2)$ is optimal to PTL[2] if and only if $P_L^*(2)$ satisfies $\mathrm{PAI}(i_1) \ge \mathrm{CAI}(1, n) = \frac{D}{R} > \mathrm{PAI}(i_2)$ where $i_k \in C_k^*$ for k = 1, 2 and $P_L^*(2)$ is one of the partitions based on a PAI-nonincreasing ordering.

Proposition 1 implies that if given a PAI-nonincreasing ordering O, we assign the first N_1 products of O to C_1^* and the remaining products of O to C_2^*, then $\{C_1^*, C_2^*\}$ is optimal. Yang and Kim (2004) addressed PTL[3] and derived a necessary condition to PTL[3] as follows and provided an exact algorithm with $O(n\log\lceil n\rceil)$ based on their condition.

Proposition 2: (Necessary condition to PTL[3]) If $P_L^*(3)$ is optimal to PTL[3], then $P_L^*(3)$ satisfies

$$\mathrm{PAI}(i_k) \ge \mathrm{CAI}(N^*_{k-1}+1, N^*_{k+1}) > \mathrm{PAI}(i_{k+1}) \quad \text{for } k = 1, 2 \tag{5}$$

where $N^*_0 = 0$, $i_k \in C^*_k$ for k = 1, 2, 3 and $P^*_L(3)$ is one of the partitions based on a PAI-nonincreasing ordering.

From Proposition 1 and 2, it may be conjectured that Equation (5) is also sufficient condition to PTL[3]. We will prove this in Appendix. From two previous propositions, a necessary and sufficient condition to PTL[K] may be conjectured as

$$\mathrm{PAI}(i_k) \ge \mathrm{CAI}(N^*_{k-1}+1, N^*_{k+1}) > \mathrm{PAI}(i_{k+1}) \quad \text{for } k = 1, \cdots, (K\text{-}1) \tag{6}$$

where $i_k \in C^*_k$ for k = 1, $\cdots$, K. However, this conjecture turns out to be wrong. It turns out that Equation (6) is not a sufficient condition but a necessary condition to PTL[K]. We will prove that Equation (6) is a necessary condition to PTL[K] as stated in the following theorem, and Equation (6) is not a sufficient condition by showing a counterexample in Appendix.

Theorem 3: (Necessary condition to PTL[K]) If $P^*_L(K)$ is optimal to PTL[K], then $P^*_L(K)$ satisfies Equation (6) and $P^*_L(K)$ is one of the partitions based on a PAI-nonincreasing ordering.

Proof: Consider any two adjacent classes C^*_k and C^*_{k+1} from $P^*_L(K)$. Then, for $i_k \in C^*_k$ and $i_{k+1} \in C^*_{k+1}$, Equation (6) holds true. If not, $E(SC_k)$ can be further reduced by applying Proposition 1 to $\{C^*_k, C^*_{k+1}\}$. This is a contradiction to that $P^*_L(K)$ is optimal. Equation (4) holds for k = 1, $\cdots$, (K-1). By rearranging products in each class by PAI-nonincreasing order, finally $P^*_L(K)$ will be one of the partitions based on a PAI-nonincreasing ordering. □

Theorem 3 implies that an optimal solution to PTL[K] can be obtained by enumerating all the partitions based on a PAI-nonincreasing ordering and that any candidate solution $X_L(K)$ to PTL[K] can be expressed as $(N_1, N_2, \cdots, N_{K-1})$, where N_k is the number of products assigned to classes 1 through k for k = 1, $\cdots$, K. Note that $N_K = n$. Furthermore, Equation (6) can be rewritten as

$$PAI(N_k^*) \geq CAI(N_{k-1}^* + 1, N_{k+1}^*) > PAI(N_k^* + 1) \quad \text{for} \quad k = 1, \cdots, (K\text{-}1) \tag{7}$$

Theorem 3 also implies that those products with the same PAI value can be assigned to the same class and that replacing those m products with a single product does not affect the optimal CAI values. Hence given a data set, $\{(r_i, d_i), i = 1, \cdots, n\}$, if there are m products with the same PAI value, say d/r, and if the number of different PAI values is greater than or equal to K, then replacing those m products with a single product having space requirement mr and average retrieval rate md does not affect the optimality to PTL[K]. Applying this property to an original input data set, we can obtain a modified input data set where PAI's are unique for each product. For this reason, we use the term "PAI-decreasing" instead of "PAI-nonincreasing" from now on. In real-life warehouse, some products, especially slow-moving items, are likely to have almost same PAI values. In this case, the products can be treated as a single product for an optimal solution. Without loss of generality, we assume that a PAI-decreasing ordering is given as $O = \{1, 2, \cdots, n\}$.

3. An Exact Splitting Algorithm for PTL[4] and An Example

3.1 An Exact Splitting Algorithm

Consider PTL[4]. Without loss of generality, assume that a PAI-decreasing ordering is given as $O = \{1, 2, \cdots, n\}$. From Theorem 3, an optimal solution to PTL[4], which can be represented by $X_L(4) = (N_1, N_2, N_3)$, is one of the partitions based on an O, and the enumeration of those partitions gives an optimal solution. We call this naive enumeration algorithm or NE algorithm. This algorithm clearly requires $O(n^3)$. However, the time complexity can be reduced to $O(n\lceil \log n \rceil)$ by the following algorithm, called as the exact splitting algorithm or ES algorithm.

Suppose that N_2 is fixed and given. Then it can be observed from Equation (2) that D_1, D_2, T_1 and T_2 can be determined by the combination of the first N_2 products of an O, and that D_3, D_4, T_3 and T_4 can be also determined by the combination of the last $(n - N_2)$ products of an O. Hence the minimization of $E(SC_4)$ given N_2 is equivalent to solving two split PTL[2]'s, one PTL[2] with the first N_2 products of an O and the other PTL[2] with the last $(n-N_2)$ products of an O. Given N_2, define $N_1^c(N_2)$ and $N_3^c(N_2)$ as an optimal solution of the first PTL[2] and the second PTL[2] respectively, and define $X_{LC}(4)=(N_1^c(N_2), N_2, N_3^c(N_2))$ as the conditional optimal solution given N_2, which can be determined from Proposition 1 such that

$$PAI(N_1^c(N_2)) \geq CAI(1, N_2) > PAI(N_1^c(N_2)+1) \tag{8}$$

$$PAI(N_3^c(N_2)) \geq CAI(N_2+1, n) > PAI(N_3^c(N_2)+1) \tag{9}$$

Clearly, $E(SC_4 \mid X_{LC}(4)) \leq E(SC_4 \mid X_L(4))$. Since N_2^* must be one of the values through two to (n-2), by changing N_2 from 2 to (n-2), we can find an optimal solution. In addition, since $N_1^c(N_2)$ and $N_3^c(N_2)$ must also satisfy Equation (6), we have

$$PAI(N_2) \geq CAI(N_1^c(N_2)+1, N_3^c(N_2)) > PAI(N_2+1) \tag{10}$$

Thus if $X_L(4)$ does not satisfy Equation (10), it will be discarded. Now, the ES algorithm can be summarized as follows.

```
ES Algorithm
Step 1: Take products by PAI-decreasing order.
Step 2: Set $E(SC_4^*) \leftarrow$ big value
      For $N_2 = 2$ to (n-2) do
      Begin
        Find $N_1^c(N_2)$ and $N_3^c(N_2)$ satisfying Equation (8) and Equation (9).
        If Equation (10) is true, then do
          Begin
            Compute $E(SC_4)$ using $X_L(4)=(N_1^c(N_2), N_2, N_3^c(N_2))$
            If $E(SC_4) < E(SC_4^*)$, then $E(SC_4^*) \leftarrow E(SC_4)$ and
              $X_L^*(4) \leftarrow (N_1^c(N_2), N_2, N_3^c(N_2))$
          End
      End
```

Proposition 4: ES algorithm solves PTL[4] in $O(n\lceil \log n \rceil)$.

Proof: Since ES algorithm enumerates a set of the solutions which satisfies Equation (6), it solves PTL[4]. Since Step 1 requires $O(n\lceil \log n \rceil)$, and the maximum number of iterations at Step 2 is $O(n)$ and each iteration requires $O(\lceil \log n \rceil)$, Step 2 requires $O(n\lceil \log n \rceil)$. Thus ES algorithm requires $O(n\lceil \log n \rceil)$. □

Our NE or ES algorithm can be used as a heuristic algorithm to PTN[4]. Since the set of solutions generated by the ES algorithm is a subset of the solutions generated by the NE algorithm, the minimum objective value to PTN[4] found by the NE algorithm must be less than that found by the SS algorithm. However, the ES algorithm is superior to the NE algorithm in terms of time complexity.

3.2 An Example

Suppose that the input data are given as $\{(r_i, d_i), i = 1, \cdots, 10\}$ = {(5, 3), (15, 4), (20, 5), (10, 2), (15, 3), (20, 4), (12, 2), (7, 1), (7, 1), (50, 6)} and $t_j = j$ for j = 1, ⋯, 161. Since PAI(4) = PAI(5) = PAI(6) and PAI(8) = PAI(8), we replace products 4, 5, and 6 with product A having PAI(A) as 9/45 and product 8 and 9 with product B having PAI(B) as 2/14. Without loss of generality, the modified data set can be obtained as $\{(r_i, d_i), i = 1, 2, 3, A, 7, B, 10\}$ = {(5, 3), (15, 4), (20, 5), (45, 9), (12, 2), (14, 2), (50, 6)}. For convenience, we convert $\{(r_i, d_i), i = 1, 2, 3, A, 7, B, 10\}$ as a revised data $\{(r_i, d_i), i = 1, 2, 3, 4, 5, 6, 7\}$. Using this revised data and following the ES algorithm, we have

Step 1: Since $\{PAI(i), i = 1, \cdots, 7\}$ = {0.6000, 0.2667, 0.2500, 0.2000, 0.1667, 0.1429, 0.1200}

$O \leftarrow$ {1, 2, 3, 4, 5, 6, 7}

Step 2: Set $E(SC_4^*) \leftarrow$ big value

When $N_2 = 2$, $N_1^c(2) \leftarrow 1$ since CAI(1, 2) = 0.3500, $N_3^c(2) \leftarrow 4$ since CAI(3, 7) = 0.1702

Equation (10) is false since PAI(2) = 0.2667, CAI(2, 4) = 0.2250, and PAI(3) = 0.2500.

When $N_2 = 3$, $N_1^c(3) \leftarrow 1$ since CAI(1, 3) = 0.3000, $N_3^c(3) \leftarrow 5$ since CAI(4, 7) = 0.1570

Equation (10) is true since PAI(3) = 0.250 0 ≥ CAI(2, 5) = 0.2174 > PAI(4) = 0.2000.

$E(SC_4) \leftarrow 129.7419$ using $X_L(4) \leftarrow (1, 3, 5)$.

Since $E(SC_4) <$ big value, $E(SC_4^*) \leftarrow 129.7419$ and $X_L^*(4) \leftarrow (1, 3, 5)$.

When $N_2 = 4$, $N_1^c(4) \leftarrow 3$ since CAI(1, 4) = 0.2471, $N_3^c(4) \leftarrow 6$ since CAI(5, 7) = 0.1316

Equation (10) is true since PAI(4) = 0.2000 ≥ CAI(4, 6) = 0.1831 > PAI(5) = 0.1667.

$E(SC_4) \leftarrow 130.7097$ using $X_L(4) \leftarrow (3, 4, 6)$.

Since $E(SC_4) > E(SC_4^*)$, we do not update $E(SC_4^*)$ and $X_L^*(4)$.

When $N_2 = 5$, $N_1^c(5) \leftarrow 3$ since CAI(1, 5) = 0.2371, $N_3^c(5) \leftarrow 6$ since CAI(6, 7) = 0.1250

Equation (10) is false since PAI(5) = 0.1667 , CAI(4, 6) = 0.1831, and PAI(6) = 0.1429.

Hence the optimal solution $X_L^*(4)$ can be obtained as (1, 3, 5) with the optimal objective value $E(SC_4^*) = 129.7419$ in the revised data form or equivalently, the optimal partition $P_L^*(4)$ can be obtained as {{1}, {2, 3}, {4, 5}, {6, 7}}. In the original data form, the optimal solution can be obtained as (1, 3, 7) or equivalently, the optimal partition can be obtained as {{1}, {2, 3}, {4, 5, 6, 7}, {8, 9, 10}}. Note that $P_L^*(4)$ can be used as a heuristic solution to PTN[4].

4. Concluding Remarks

In this paper, we addressed a layout design problem for determining an optimal 4-class-based dedicated linear storage layout in a class of unit load storage systems. We provided some fundamental results including a sufficient condition to PTL[3], a necessary condition to PTL[K], a counterexample that the necessary and sufficient condition to PTL[3] is no longer extended to PTL[K] when $K \geq 4$. Based on these properties, we provided an exact splitting algorithm with $O(n\lceil \log n \rceil)$, which could be used as an efficient heuristic algorithm to PTN[4].

The assumption that the space requirement for a class is the sum of the maximum inventory levels of products assigned to the class may not be realistic and needs to be replaced with a reasonable mathematical formula for the space requirement. However, even though C_k is given, the problem of determining the space requirement for class k, or equivalently minimizing the maximum aggregate stock level for multiple products is not easy. This problem is well known to be NP-hard. No matter what kind of mathematical formula for the space requirement is provided, our algorithms could be either generalized or applied to more than a 4-class-based dedicated storage layout problem with slight modification in order to achieve both benefits, the throughput benefits of dedicated storage and the space benefits of randomized storage as Tompkins and White suggested.

Further research may be concentrated on the minimization of the expected dual command travel time at the cost of mathematical tractability and time complexity or on comparing the solution to PTN[4] with the optimal solution which minimizes the expected dual command travel time.

References

[1] Bazaraa, M. S., "Computerized Layout Design: a Branch and Bound Approach," *AIIE Transactions* 7, 4 (1975), 432-438.

[2] Eynan, A. and M. J. Rosenblatt, "An Interleaving policy in automated storage/retrieval systems," *International Journal of Production Research* 31, 1 (1993), 1-18.

[3] Hall, N. G., "A Multi-Item EOQ Model with Inventory Cycle Balancing," *Naval Research Logistics* 35, 3 (1988), 319-325.

[4] Hausman, W. H., L. B. Schwarz, and S. C. Graves, "Optimal storage assignment in automatic warehousing systems," *Management Science* 22, 6 (1976), 629-638.

[5] Kouvelis, P. and V. Papanicolaou, "Expected travel time and optimal boundary formulas for a two-class-based automated storage/retrieval system," *International Journal of Production Research* 33, 10 (1995), 2889-2905.

[6] Lee, M. K., "A storage assignment policy in a man-on-board Automated Storage/Retrieval System," *International Journal of Production Research* 30, 10 (1992), 2281-2292.

[7] Malmborg, C. J. and B. Krishnakumar, "Optimal storage assignment policies for multiaddress warehousing systems," *IEEE Transactions on Systems, Man and Cybernetics* 19, 1 (1989), 197-204.

[8] Malmborg, C. J. and K. Bhaskaran, "A revised proof of optimality for the cube-per-order index rule for stored item location," *Applied Mathematical Modeling* 14, 2 (1990), 87-95.

[9] Rosenblatt, M. J. and A. Eynan, "Deriving the optimal boundaries for class-based automatic storage/retrieval systems," *Management Science* 35, 12 (1989), 1519-1524.

[10] Tompkins, J. A. and J. A. White, *Facilities Planning*, John Wiley and Sons Inc., NY. (1984), 335-338.

[11] Van den Berg, J. P., "Class-based storage allocation in a single command warehouse with space requirement constraints," *International Journal of Industrial Engineering* 3, 1 (1996), 21-28.

[12] Yang, M., "Analysis and Optimization of a 2-class-based Dedicated Storage System," *Journal of the Korea Institute of Industrial Engineers* 29, 3 (2003), 222-229.

[13] Yang, M. and S. Kim, "Optimization of a 3-class-based Dedicated Linear Storage System," *Journal of the Korea Institute of Industrial Engineers* 30, 3 (2004), 190-196.

[14] Yang, M., "A Converging Exact Algorithm for Determining an Optimal 3-Class-Based Dedicated Linear Storage System," *International Journal of Management Science* 12, 1 (2006), 79-93.

<Appendix>

We prove that Equation (5) is also a sufficient condition. Consider PTL[3]. Before this proof, we need some definitions and two lemmas published in Yang (2006) as follows. Given O, any candidate solution to PTL[3] can be expressed as $X_L(3) = (N_1, N_2)$. Suppose that the first N_1 products of an O have been assigned to class 1. Then the minimization of $E(SC_3)$ given N_1 is equivalent to solving PTL[2] with the last $(n - N_1)$ products of an O. Let this "conditional best" solution given N_1 be expressed as $X_L^c(3) = (N_1, N_2^c(N_1))$, where $N_2^c(N_1)$ can be determined from Proposition 1 such that $PAI(N_2^c(N_1)) \geq CAI(N_1 + 1, n) > PAI(N_2^c(N_1) + 1)$. Clearly, we have, $E(SC_3 | X_L^c(3)) \leq E(SC_3 | X_L(3))$. In the similar manner, the conditional best solution $X_L^c(3) = (N_1^c(N_2), N_2) =$ given N_2 can be determined such that $PAI\,(N_1^c(N_2)) \geq CAI(1, N_2) > PAI\,(N_1^c(N_2) + 1)$. Now, Yang's converging exact algorithm (2006), called as ALGCONV[3], can be described as follows. Take any value of N_1 given an O and find $N_2^c(N_1)$. Regard $N_2^c(N_1)$ as N_2 and find $N_1^c(N_2)$. Repeat this process until there is no change in the obtained solution. Define the unchanged solution to be an equilibrium solution, $X_E(3) = (N_1^e, N_2^e)$. Define $X_L^*(3)$ to be an optimal solution to PTL[3].

Lemma A-1: $E(SC_3 | X_L^c(3))$ is a discrete convex function of N_1. That is,
(i) If $N_1 < N_1^e$, then $E(SC_3 | X_L^c(3))$ is a decreasing function of N_1.
(ii) If $N_1 > N_1^e$, then $E(SC_3 | X_L^c(3))$ is an increasing function of N_1.
Proof: See Lemma 1 in Yang (2006).

Lemma A-2: Any initial solution converges to $X_E(3)$ by minimizing $E(SC_3)$.
Proof: See Lemma 3 in Yang (2006).

Proposition A- 3: (Sufficient condition to PTL[3])
If $X_L(3)$ satisfies $PAI(i_k) \geq CAI(N_{k-1}^* + 1, N_{k+1}^*) > PAI(i_{k+1})$ for k = 1, 2, where $i_k \in C_k^*$ for k = 1, 2, 3, then $X_L(3)$ is an optimal solution to PTL[3].
Proof: Since $X_E(3) = (N_1^e, N_2^e)$ satisfies Equation (6). It is enough to show that $X_E(3) = X_L^*(3)$. Consider a solution $X_L(3) = (N_1, N_2)$. Then, there exists a conditional best solu-

tion $X_L^c(3) = (N_1, N_2^c(N_1))$ such that $Z_L(X_L(3)) \geq Z_L(X_L^c(3))$ for given N_1. From Lemma 2, any conditional best solution converges to $X_E(3)$ by minimizing $E(SC_3)$. Since $Z(X_L^c(3))$ is a decreasing function of N_1 in the range of $N_1 < N_1^e$ and an increasing function of N_1 in the range of $N_1 > N_1^e$ from Lemma 1, we have $Z(X_L^c(3)) \geq Z(X_E(3))$ for any value of N_1. It follows that $Z(X_L(3)) \geq Z(X_E(3))$. □

Now, we will prove that Equation (6) is not a sufficient condition to PTL[K] by showing a counterexample as follows. Let $E(SC_K|X)$ be the expected SC travel time given a solution X and a value of K.

Counter example: Consider the data as shown in Table 1. Let $t_j = j$ for j = 1, 2, ⋯, 164. Clearly the PAI-decreasing ordering is {1, 2, 3, 4, 5, 6, 7, 8, 9. 10}. It can be shown that $X_L(4)$ = (3, 5, 7) satisfies Equation (6) for k = 1, 2, 3. However, it is not optimal since $E(SC_3|X_L(4))$ = 135.66 > $E(SC_3|X_L^*(4))$ = 134.93 where $X_L^*(4)$ = (1, 4, 6). Now, using the above counterexample, we will construct a counterexample for K = 5 as follows. Consider a product with (r_{11}, d_{11}). Given $X_L(4)$, determine (r_{11}, d_{11}) such that Equation (6) is satisfied by assigning only this product to class 5. That is, (r_{11}, d_{11}) must satisfy Equation (7); PAI(10) = $\frac{1.2}{8} \geq$ CAI(8, 11) = $\frac{3+2+1.2+d_{11}}{15+11+8+r_{11}}$ > PAI(11) = $\frac{d_{11}}{r_{11}}$. Clearly, (r_{11}, d_{11}) = (14, 1) can be a solution for the above equation. Let $X_L(5)$ = (3, 5, 7, 10). Then $X_L(5)$ satisfies Equation (6) but $X_L(5)$ is not optimal because $E(SC_5|X_L(5)) > E(SC_5|X_L^*(5))$ where $X_L^*(5)$ = (1, 4, 6, 10). In the similar manner, the counterexample for $K \geq 6$ can be constructed.

Table A-1. Input Data for a Counterexample where R = 164 and D = 56.7

Product	r_i	d_i	PAI(i)	Product	r_i	d_i	PAI(i)
1	15	9.0	0.6000	6	24	7.5	0.3125
2	12	5.4	0.4500	7	12	3.1	0.2583
3	25	11.0	0.4400	8	15	3.0	0.2000
4	12	4.5	0.3750	9	11	2.0	0.1818
5	30	10.0	0.3333	10	8	1.2	0.1500