Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 31 (2014) 773 – 777

Procedia Computer Science

Information Technology and Quantitative Management , ITQM 2014

A dynamic programming heuristic for optimizing slot sizes in a warehouse

Pavel Sukhov[a,*], Mikhail Batsyn[a], Petr Terentev[a]

[a]National Research University Higher School of Economics, 136 Rodionova, Nizhny Novgorod, 603093, Russia

Abstract

There exist a number of mathematical problems in the literature concerning warehouse storage optimization. However, to the best of our knowledge the problem of finding optimal slot sizes in pallet rack system minimizing the occupied storage space is not studied. More precisely, the problem is to optimally choose *k* of *m* pallet racks with pallets in a warehouse, find optimal slot sizes in pallet racks for these pallets, and reassign the pallets to the new slots in order to reduce the occupied space maximally. In this paper we suggest a dynamic programming heuristic which finds optimal slot sizes for the given number *k* of pallet racks in a warehouse. Our computational experiments are based on real-life data and demonstrate the efficiency of the suggested algorithm.

Keywords: storage assignment, slot sizes optimization, dynamic programming

1. Introduction

Warehouse optimization is a well known area of discrete optimization. Many practical problems concerning the location and design of warehouses are discussed. For detailed reviews of papers on warehouse design and operations see Berg (1999) [5], Koster et al. (2007) [1], and Gu et al. (2007) [3].

To the best of our knowledge, there is currently no research on slot sizes optimization. In practice it is usually possible to change the sizes of slots in a pallet racking, using, for instance, boltless selective pallet rack. Unfortunately, this possibility is not enough studied in the literature and applied in practice. The main reasons are the underestimated profit of such optimization and overestimated cost of warehouse reorganization. In this paper we suggest a heuristic for pallet racking reorganization and demonstrate the efficiency of this approach on real-life data. We study empirically how much space we can free by reorganizing the certain number of pallet racks.

The paper is organized as follows. In the next section we provide an integer linear programming formulation of the considered problem and show the computational complexity of this problem. In the third section we introduce our heuristic algorithm for solving this problem. In the fourth section the computational results for real-life data are given.

E-mail address: pavelandreevith@gmail.com.
*Corresponding author. Tel.: +7-831-436-1397; fax: +7-831-436-1397.

2. Problem formulation

The optimization problem we consider consists in optimal reorganization of pallet racking. Assume that we can reassign pallets located on some racks in order to reduce the wasted space above each pallet. Such a "reorganization" reduces the used space. To estimate the profit of such a reorganization we measure how much space we are able to reduce for the certain number of reorganized racks.

We ignore a beam height, because one can always assume, that the beam is a part of a pallet, so the height of the beam is a part of the pallet's height. Let us also assume, that all the pallets are already located in certain slots in a warehouse. We will call a single pallet rack a "box".

The problem can be described as follows. There are m boxes with a fixed height H and n ($n \geq 2$) pallets with an arbitrary height h_i ($1 \leq h_i \leq H$) in a warehouse. There is a number of slots in each box separated by load beams at arbitrary height. Every pallet is located in one slot and every slot contains at most one pallet. The goal is to reorganize exactly k boxes such that the total number of boxes occupied with pallets is minimized.

Below we provide an Integer Linear Programming (ILP) formulation of the problem. The following parameters and decision variables are used in the ILP model.

Parameters:
$h_i, i \in \{1, \ldots, n\}$ - pallet heights;
$x^0_{ij} \in \{0, 1\}, i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}$ - the initial assignment of pallets to boxes: x^0_{ij} is equal to 1 iff pallet i is located in box j. It is guaranteed, that $\forall i \sum_{j=1}^{n} x^0_{ij} = 1$;
H - the height of a box;
k - the number of boxes to be reorganized;
m - the total number of boxes in a warehouse.
Decision variables:

$$x_{ij} = \begin{cases} 1 & \text{if pallet } i \text{ is assigned to box } j \\ 0 & \text{otherwise} \end{cases}$$

$$y_j = \begin{cases} 1 & \text{if box } j \text{ is used for relocated pallets} \\ 0 & \text{otherwise} \end{cases}$$

$$z_j = \begin{cases} 1 & \text{if box } j \text{ is reorganized} \\ 0 & \text{otherwise} \end{cases}$$

The ILP model of the problem is as follows:

$$min \sum_{i=1}^{n} y_i \tag{1}$$

subject to

$$\sum_{i=1}^{n} h_i x_{ij} \leq H y_j \quad j = 1, \ldots, m \tag{2}$$

$$\sum_{j=1}^{m} x_{ij} = \sum_{j=1}^{m} z_j x^0_{ij} \quad i = 1, \ldots, n \tag{3}$$

$$\sum_{j=1}^{m} z_j = k \tag{4}$$

$$x_{ij} \in \{0, 1\} \quad i = 1, \ldots, n \quad j = 1, \ldots, m \tag{5}$$

$$y_j \in \{0, 1\} \quad j = 1, \ldots, m \tag{6}$$

$$z_j \in \{0, 1\} \quad j = 1, \ldots, m \tag{7}$$

The objective is to minimize the total number of boxes used for reallocated pallets (1). Constraint (2) reflects the limited capacity of boxes. The total height of the pallets located in one box cannot be greater than H. No pallets can be assigned to a box which is not used.

From the parameter's description it follows that $\sum_{j=1}^{m} z_j x_{ij}^0 = 1$ iff pallet i is initially located in a box which is reorganized, and $\sum_{j=1}^{m} z_j x_{ij}^0 = 0$ otherwise. Constraint (3) requires, that pallet i should be assigned to a new box iff it is belongs to one of the reorganized boxes. Constraint (4) requires, that we should reorganize exactly k boxes in the warehouse.

This problem is strongly NP-hard. The special case of this problem for $k = m$ is equivalent to the Bin Packing Problem, where the height of boxes is the size of bins and the pallet sizes are the sizes of items. The Bin Packing Problem is known to be a strongly NP-hard problem (Garey & Johnson, 1979 [2]), therefore there is no pseudopolynomial algorithm for our problem unless $P = NP$.

3. Algorithm description

Let us consider the following problem. There given n pallets of arbitrary size and k boxes of size H. All the pallets should be placed to the boxes so that the total number of boxes occupied by the pallets is minimized. It is easy to see, that this problem can be reduced to the knapsack problem (Martello & Toth, 1990 [4]). Let us call the well-known dynamic scheme for solving the knapsack problem "KnapsackDynamic". We can repeat this dynamic scheme, applying it separately to each of k boxes in the way, shown in the pseudocode below.

We choose k boxes with the smallest total loading, take all pallets from these boxes and put them into the set A of pallets to be relocated. We fill a new box using the KnapsackDynamic procedure for this set, and then remove the allocated pallets from it. This procedure is repeated while the set of pallets is not empty. The corresponding pseudo-code is presented below. The following variables A_j are used in this pseudo-code: $A_j = \{i : x_{ij}^0 = 1\}$. A_j is the set of pallets initially assigned to box j.

```
**Data**: $k$;$n$;$m$;$H$;$[h_i]$;$[A_j]$
**Result**: $S = \{j_1, ..., j_k\}$ - the set of reorganized boxes;
$r$ - the number of boxes occupied with the relocated pallets;
$A'_j$, $j \in \{1, ..., r\}$ - the j-th set of the relocated pallets assigned to one box.
$S \leftarrow \emptyset$
$B \leftarrow \{1, ..., m\}$ // boxes for reorganization
**for** $l \leftarrow 1$ **to** $k$ **do**
    $j^* \leftarrow \operatorname{argmin}_{j \in B} \sum_{i \in A_j} h_i$
    $S \leftarrow S \cup \{j^*\}$
    $B \leftarrow B \setminus \{j^*\}$
**end**
$A \leftarrow \bigcup_{j \in S} A_j$ // the set of all pallets to be relocated
$r \leftarrow 0$
**while** $A \neq \emptyset$ **do**
    $r \leftarrow r + 1$
    $A'_r \leftarrow$ KnapsackDynamic$(H, A, [h_i])$
    $A \leftarrow A \setminus A'_r$
**end**
```

Algorithm 1: Heuristic Algorithm

4. Computational results

We use real-life data for computational experiments. A pallet racking consists of slots of arbitrary height. The heights and other parameters of slots are given in appendix. We do not provide the heights of pallets because

Table 1. Occupied space reduction

t	q	min(%)	average(%)	max(%)
20	10	11.76	17.76	26.67
20	15	11.54	17.84	30.00
20	20	12.50	19.40	24.00
30	5	10.00	14.98	28.57
30	10	13.04	18.17	21.05
30	15	15.79	21.92	28.57
40	5	11.11	18.05	23.08
40	10	14.29	18.11	27.27
40	15	14.63	19.88	22.22
50	5	7.14	16.50	25.00
50	10	14.28	20.16	27.27
50	20	18.60	22.01	26.47

there are 4355 pallets in the considered instance. The following random generation is used for computational experiments . The generator works as follows:

1. Uniformly choose t pallet types from the given list;
2. For each chosen type select the quantity of pallets uniformly from set $\{0, 1, ..., q\}$;
3. Assign each pallet to the smallest possible slot

We apply our algorithm to the generated instances. The optimization of such pallet assignments is possible due to the difference between the heights of pallets and slots. For each t and q the computation is repeated 100 times. In case where it is impossible to assign the generated pallets set to the given storage, the generation is rerun.

The computational results are presented in Table 1. Parameters t and q are described above, "min" is the minimum obtained reduction of the initial storage space, "max" is the maximum reduction, and "average" is the average reduction over all 100 runs.

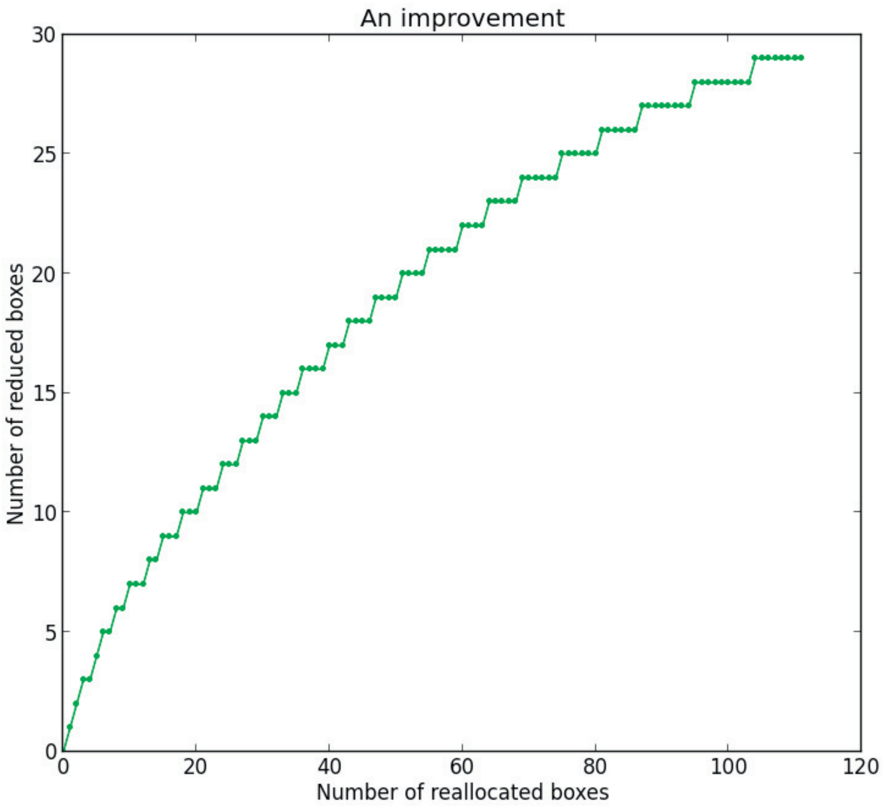
In Table 2 the information on the considered warehouse is given. The first three columns show a slot size. The fourth column gives the height of a slot after removing the height of a beam. In the last column there is the quantity of slots with the given parameters.

Table 2. Slot types

length	width	height	real height	quantity
1.2	0.8	0.70	0.672	562
1.2	0.8	1.00	0.960	19
1.2	0.8	1.20	1.152	16
1.2	0.8	1.50	1.440	54
1.2	0.8	1.55	1.488	249
1.2	0.8	2.00	1.920	92
1.2	0.8	2.10	2.016	16
1.2	0.8	2.30	2.208	211

The obtained improvement depends on the number of reorganized boxes as it is shown in Figure 1. This graph shows an approximately logarithmic dependency of the number of freed boxes from the number of reorganized boxes. This means that reorganization of a small number of boxes gives a profit, which decreases when the number of reorganized boxes increases. Such a dependency can be obtained for other instances with different number of boxes.

Fig. 1. Example for m = 120 boxes



Acknowledgments

Mikhail Batsyn and Pavel Sukhov are supported by LATNA Laboratory, National Research University Higher School of Economics (NRU HSE), Russian Federation government grant, ag. 11.G34.31.0057.

References

[1] de Koster, R., T. Le-Duc, K.J. Roodbergen. *Design and control of warehouse order-picking: A literature review.* Eur. J. Oper. Res., 182 481-501, 2007.
[2] Garey M. R., D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman., ISBN 0-7167-1045-5. A4.1: SR1, p. 226., 1979.
[3] Gu, J., M. Goetschalckx, L.F. McGinnis. *Research on warehouse operation: A comprehensive review.* Eur. J. Oper. Res., 177 1-21, 2007.
[4] Martello S., P. Toth *Knapsack Problems: Algorithms and Computer Implementation.* John Wiley and Sons 1990.
[5] van den Berg, J.P. *A literature survey on planning and control of warehousing systems.* IIE Trans., 31 751-762, 1999.