Planning in Logistics: A survey

Pushkar Kolhe
Georgia Institute of Technology
pushkar@cc.gatech.edu

Henrik Christensen
Georgia Institute of Technology
hic@cc.gatech.edu

ABSTRACT

Planning is an essential part of any logistics system. The paper tries to generalize the view of a logistics planner by framing it as a knapsack problem. We show how the various variants of the knapsack problem compare for different types of industries. We also introduce the pallet stacking problem and survey some of the recent advances made towards this problem.

Keywords

Planning, Logistics, Bin Packing Problem, Warehouse Management

1. INTRODUCTION

Logistics integrates all the operations in an industry. It is a planning system that is supposed to model, analyze and optimize the entire supply chain. Naturally improving the logistics is a key step in industrial automation. Despite significant progress in improving efficiency in pick and place robots, there exists many important problems and much room for improvement. At present many solutions are designed for a particular manufacturer or have many assumptions. The article attempts to bridge the gap between current research and real world problems in logistics. We provide a general overview of the problem and give an introduction to some of the most basic techniques used to solve these problems.

Consider the special cases of a shipping industry or a automated storage and retrieval system where the goal is to improve throughput of the system under some constraints. This can be formulated as a constraint problem, specifically the knapsack problem (KP). We show that logistics operations can be studied theoretically and their performance can be guaranteed by a theoretical upper bound. It is possible the most optimal performance that the system can achieve. If an industry successfully models its logistics as a KP constraint problem, then it is possible to determine the worst case performance of their planner when it is compared to an optimal planner. New research tools from combinatorics can be used for improving a logistics planner.

We also consider the pallet stacking problem and show some of the recent advancement made in this field. We then list some open problems in logistics from our view point that can be improved solely by providing a good theoretical context to logistics.

2. HISTORICAL OVERVIEW

Most recent work in warehouse management systems has focused on satisfying needs of a particular industry. However there has been less work in improving logistics planners. On the other hand the pallet stacking problem, which is closely related to the bin packing problem, has made great advances in the field of combinatorics. It is a well known fact that mixed palletizing is NP-hard, but there are many polynomial and pseudo-polynomial planners which can give an approximate solution which can have a lower bound on its optimality. [31] gives a good overview of the mixed palletizing problem and proves that it is indeed a variant of the KP problem. There is also some work in optimizing robot motion to improve performance of mixed palletizing systems. For example, [28] shows how robot movements can be minimized while packing a pallet. There has also been significant development of software tools to solve this problem. [18] provides an overview of the combinatorial knapsack problem with a few examples for solving the palletizing problem.

The paper is divided into three parts. Section 3 shows how an entire warehouse management process can be formulated mathematically as a knapsack problem. We show how various variants of the knapsack problem can represent logistics for different types of industries. Section 4 introduces some of the most common knapsack problem solving techniques. Section 5 is dedicated to the pallet stacking problem where we show some of the most common techniques used for solving such kind of problems. We refer to a comprehensive list of references as we introduce new concepts.

3. LOGISTICS

Logistics is a key part of a warehouse management system. A logistics problem can be framed as a decision problem can be represented numerically as profit, loss or weight, cost value. This has two main advantages: 1. The entire warehouse management system can be framed mathematically and 2. Different approaches can be compared with respect to some benchmarks. Many other industrial problems can be formulated as knapsack problems: cargo loading, cutting stock, project selection and budget control.

Consider the simple example of managing logistics for a shipping industry. Packages are usually shipped through cargo trucks or planes. Every package has a certain weight but the cargo vehicle has a fixed capacity. The cost of sending a package usually does not depend on the weight, since dispatchers get paid on a contract. The dispatchers, therefore, try to maximize their profit by packing efficiently and shipping the maximum weight in a fixed volume. Such a problem can be framed as a optimization problem.

3.1 The Knapsack Problem

We can formulate the above logistics problem as a knapsack problem (KP). Consider a knapsack with item set N, consisting on n items, where the j_{th} item has profit p_j and weight w_j, and the capacity is c. Then, the objective function can be formulated as:

$$\max \sum_{j=1}^{n} p_j x_j \tag{1}$$

subject to

$$\max \sum_{j=1}^{n} w_j x_j \leq c \tag{2}$$

$$x_j \in \{0,1\},\ j = 1, \ldots, n. \tag{3}$$

In equation (2), x_j can only take integral values and it denotes whether the j_{th} item is included in the knapsack or not. Finding the optimum solution vector x^* having an optimum profit z^* is non-trivial and known to be NP-hard.

However the logistics involved in shipping industries is not as simple as stated above. There can be several additional constraints. For example, a few packages might have to be delivered urgently and so packages also have a priority associated with them. Some packages have to be delivered within a particular time window and some package has exceptionally low weight, but high volume. Because of constraints like these, the definition of profit in equation (1) can change. This leads to many variations in the above problem.

In the above example, if optimal number of packages are transported by shipping the maximum possible weight in a given volume then $p_j = w_j$ in equation (1). This is called the **subset sum problem**. Solutions of the subset problems can be used for designing better lower bounds for scheduling problems [12].

Frequently, a large number of the boxes that are shipped have the same size and weight. To reduce the size of the knapsack problem, x_j can be used to represent a class of boxes rather than a single box. If there are b_j boxes of a class j, then equation (3) becomes:

$$0 \leq x_j \leq b_j, j = 1, \ldots, n. \tag{4}$$

This problem is then called the **bounded knapsack problem**. If b_j is large or unknown, then it is called the **unbounded knapsack problem**.

If you take into account the shipping example again and consider that the boxes have weight as well as volume and both have a maximum bound, then equation (2) should change. After generalizing the problem to include additional constraints we get a **d-dimensional knapsack problem**. We can rewrite equation (2) as:

$$\max \sum_{j=1}^{n} w_{ij} x_j \leq c_i,\ i = 1, \ldots, n. \tag{5}$$

The knapsack problem given by equations (2), (4) and (5) constitute the most generic of the mixed palletizing problems. In theoretical computer science, this problem is also referred to as the **3D bin packing problem**. If the mixed palletizer has to generate rainbow pallets, then equation (1) will be modified to reflect choice of at least two types of boxes together. This will yield the **quadratic knapsack problem** as:

$$\max \sum_{i,j=1}^{n} p_{ij} x_i x_j \tag{6}$$

3.2 More variants of KP

Consider the generic example of mixed palletizing we formulated in equations (2), (4) and (5). These equations can be further extended as variations of KP to represent entire logistics operations. For example, a shipping industry usually has several cargo vehicles making daily trips between popular locations. In that case the dispatcher has to decide if a particular package goes on a particular trip or the next one. If there are n items on the list of transportation requests and m trips available on a route, we use nm binary variables x_{ij} for representing if a particular item goes on the m^{th} trip.

The mathematical programming formulation of such a problem is called as the **multiple knapsack problem** and is given by:

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} p_j x_{ij} \tag{7}$$

subject to

$$\sum_{j=1}^{n} w_j x_{ij} \leq c_i,\ i = 1, \ldots, m, \tag{8}$$

$$\sum_{i=1}^{m} x_{ij} \leq 1,\ j = 1, \ldots, n, \tag{9}$$

$$x_{ij} \in \{0,1\},\ i,\ j = 1, \ldots, n. \tag{10}$$

Consider an example of an automated storage retrieval system. In such a system if the items were evenly distributed throughout the warehouse, they could be retrieved efficiently. Each pallet can store only one variant of each type of item, so that the overall utility value is maximized without exceeding the capacity constraint. This problem can be expressed as the following multiple-choice knapsack problem. Using the decision variable x_{ij} to denote whether variant j was chosen from the set N_i, the following model appears:

$$\max \sum_{i=1}^{m} \sum_{j \in N_i} p_{ij} x_{ij} \tag{11}$$

subject to

$$\sum_{i=1}^{m} \sum_{j \in N_i} w_{ij} x_{ij} \leq c, \tag{12}$$

$$\sum_{j \in N_i} = 1, \ i = 1, \ldots, m, \tag{13}$$

$$x_{ij} \in \{0, 1\}, \ i = 1, \ldots, m, \ j \in N_i. \tag{14}$$

Evaluating the efficiency of a warehouse system is an important factor in logistics. There are at least two ways in which we can measure the efficiency of a logistics scheme. Firstly, we can look at the output of the supply chain. For a shipping industry these may be factors like number of packages shipped, packages arriving late, wrongly delivered packages. These factors describe the quantity and quality of the items coming through the supply chain. Secondly, we can look at the planning algorithm and determine the efficiency of the logistics planner. Comparing the planner means comparing the algorithm that solves the effective knapsack problem which the logistics operation represents. Many such techniques are described in [18] and [25].

3.3 Comparing Algorithms

Algorithms that solve the knapsack problem are not similar. Algorithms that find the most optimum solution often do so by doing a exhaustive search. They are computationally very inefficient when compared to approximate algorithms which are computationally far more efficient. However, approximate algorithms have the drawback that they can only find a near optimum solution. It is also important to note the complexity for each algorithm. Simple algorithms which are easy to implement are desired. However, performance can be improved by adding appropriate complexity, for example, storing items in a tree instead of a list improves running time.

Algorithms that solve NP-hard problems can be divided into two parts: **exact algorithms** and **approximate algorithms**. Exact algorithms find the most optimal solution of a given problem whereas approximate algorithms find an approximate solution. Usually, approximate algorithms are faster because they only find a near optimal solution. Because of this, running time becomes an important criteria in comparing exact algorithms while approximate algorithms can be compared by finding out how close they come to the most optimal solution.

It is not always possible to do an exhaustive search over a problem space. Algorithms are usually run over several data sets and their performance is determined analytically. One of the most common way to check is to compare running time of the algorithm after the data set is doubled. The time required to find the solution should not increase exponentially for polynomial or pseudo polynomial algorithms.

The most common way to measure the performance of an algorithm is to perform the worst-case analysis. It is denoted by the big-Oh notation as described in [5]. The most efficient algorithms have a polynomial running time bounded by $O(n)$, $O(n \log n)$, $O(n^k)$. Pseudo-polynomial algorithms have running time bounded by $O(nc)$ which is better than $O(2^n)$ or $O(3^n)$ for non-polynomial algorithms [18].

3.4 Designing the KP problem

It is not obvious how a logistics problem can be formulated as a KP problem. Answering this question involves further research into studying and understanding industry specific parameters. The problem is to design the profit, that is (1), and weight constraints, that is (2), for a logistics problem.

Profit depends on several factors. In a shipping industry it will depend on the throughput, that is number of shipments per hour, the correctness of the delivery and the total number of cargo vehicles used. The relationship between profit and all these factors is non-linear, in the sense that they depend on each other or improving throughput comes through compromising the number of vehicles used. Similar problems arise when designing the relationship between maximum capacity and actual industrial constraints.

However, the simplest way to formulate the problem is to simply formulate profit or capacity as a linear combination of all the parameters that influence profit and capacity. The weights used will decided by the industry depending upon their specific needs and requirements. Similarly a system of equations can be formulated for a warehouse. The main idea is to formulate the logistics in such a way that it can tell us back what are the specific operations that can be compromised upon and still performing near the optimum.

4. LOGISTICS PLANNERS

As we have seen in Section 3.2 a logistics system can be expressed as a Knapsack problem. This gives us many insights into logistics. There are lower bounds available from theoretical computer science, which give a theoretical indication that the running time of an exact algorithm for (KP) can not beat a certain threshold under reasonable assumptions on the model of computation. Many (KP) instances can be solved within reasonable time by exact solution methods. This fact is due to algorithms like primal-dual dynamic programming recursions, the concept of solving a core, and the separation of cover inequalities to tighten the formulation [18].

4.1 Basic Algorithms

The greedy algorithm is perhaps the most basic algorithm that can be used to solve the KP problem. We also explain the basic idea behind exact algorithms like branch and bound, dynamic programming and approximate algorithms like polynomial time approximate schemes (PTAS) and fully PTAS.

4.1.1 The Greedy Algorithm

This is by far the most popular algorithm currently used to solve the bin packing problem. For every item, an efficiency factor is calculated as,

$$e_j := \frac{p_j}{w_j}. \tag{15}$$

We want the first bin to have the maximum profit to weight ratio. All items are arranged in a descending order based on its efficiency factor. Items are selected to be in the knapsack in this order until equation (2) is not violated.

The Greedy solution is arbitrarily bad as compared to the optimal solution, but it can yield at least a 0.5 of the optimal solution [18].

4.1.2 Branch and Bound

The general idea of the branch and bound technique is to intelligently enumerate the entire solution space and pick the best solution. It basically consists for two fundamental principles: *branching* and *bounding*. In the branching part, the solution space is divided and the optimum solution is found locally. In the bounding part, the algorithm derives upper and lower bounds of the solution space. The upper bound is found trivially in $O(n)$ by relaxing the KP integral constraint given in equation (3). Thus, $0 \leq x_j \leq 1$. The upper bound is used to prune parts of solution space whose optimum value is less than this value. The lower bound is the most optimum solution if no other local solution space has a greater lower bound.

To make the search process more efficient, the entire solution space can be divided so that it forms a tree structure. This way the search for the most optimum value can be done with a recursive depth-first or breadth-first search.

4.1.3 Dynamic Programming

Instead of optimizing the knapsack problem over all items, dynamic programming only optimizes the knapsack for a small subset of items. Then it adds an item iteratively to the problem and the solution. (KP) has the property of an optimal substructure, that is, if x^* is an optimum solution of a knapsack with capacity c, then $x^* - j$ is an optimum solution of a knapsack with capacity $c - w_j$. KP has the property of an *optimal substructure* as described in [5].

A simple dynamic programming approach to solve this problem would involve using the *Bellman recursion*. Consider l items which are a subset of the original j items. We formally solve the KP problem for l items and a knapsack capacity of $d \leq c$. The optimal solution at this point is given by $z_{j-1}(d)$. For a new item j, if $d \geq w_j$ and $z_{j-1}(d - w_j) > z_{j-1}(d)$, then it is added into the knapsack.

4.1.4 Polynomial Time Approximation Schemes (PTAS)

Algorithms that solve the knapsack problem either compromise on running time to get an optimal solution or run in pseudo-polynomial time to get an approximate solution. PTAS algorithms are also more formally known as the *ε-approximation scheme*. ε will determine how close the solution is to the optimal solution. Running time will increase if a solution near the optimal is desired.

These algorithms have the basic idea of guessing a certain set of items included in the optimal solution by going through all the possible candidate sets and then filling the remaining capacity in a greedy way [18]. In a simple scheme, the Greedy algorithm can be extended so that a subset of item are compared before inserting them in the bin. Item with maximum efficiency factor given by equation (15) within the subset is selected. The size of the subset determines the running time of this algorithm. The classical PTAS in [27] requires $O(n^{\frac{1}{\varepsilon}})$ time. The CKPP algorithm given in [4] had an improved runtime over the one in [4]. They explored the monotonicity of the arranged items for the Greedy algorithm to create subsets. This reduced their search space and time. The running time for their algorithm is $O(n^{\frac{1}{\varepsilon}-2})$ for $\varepsilon < \frac{1}{3}$.

4.1.5 Fully Polynomial Time Approximation Schemes (FPTAS)

Dynamic programming techniques discussed in 4.1.3 can be modified so that they can run in polynomial time. The earliest FPTAS technique for KP was given in [15]. Later [21] and [16] introduced new partition techniques for partitioning the profit space which improved the FPTAS algorithm further. Some of the most recent work in improving on these algorithms was done by [17, 16].

The basic idea behind the FPTAS technique is to scale the profits values or the weight values and then apply a dynamic programming technique. The optimal solution value of the scaled instance will always be at least as large as the scaled profits of the items of the original optimal solution [18]. Usually, the upper bound on the most optimal solution is found out using a Greedy method and the scaling factor is determined whose value determines the approximation of the solution. Most of the earlier solutions were impractical because they compromised memory to get better running time. However, some of the new techniques given in [17, 16] have a running time of approximately $O(n \log n \log \frac{1}{\varepsilon})$ and space complexity of $O(n + \frac{1}{\varepsilon^2})$.

4.2 All capacities problem

In several planning problem, the exact capacity is not known in advance, but it may be changed based on the proposed solutions. For example, carrying huge amounts of loads also increases transportation cost. There is obviously a non-linear relationship between actual profit and profit as defined by the KP problem. A natural way to overcome this problem is to calculate optimal solutions for each capacities including $c_{max} > c$ to find the most optimal solution. As we have stated earlier KP has the property of an optimal substructure [5]. The dynamic programming techniques exploit this property to solve the KP problem along with the all capacities problem.

5. 3D BIN PACKING PROBLEM

Packaging or storing is an integral part of many warehouse systems. Hence, many logistics planners are designed to optimize packing and storing of goods or items. As we have seen in Section 3.1, this problem can be represented as a variant of the knapsack problem. It is a well studied problem in literature. It is closely related to other three dimensional container loading problems: *Knapsack loading*, where the problem is to find a subset of items that will fit into a single bin; *Container Loading*, where the problem is to find a feasible arrangement of items in which the height of the bin filled is minimum and *Bin Packing*, where the items are packed into finite sized bins and the problem is to find the solution with the minimum number of bins.

To formulate orthogonal packaging or cutting constraints for packaging into the bin packing problem it has to be expressed as an integer programming problem, that is, condition (3) should hold. Many methods add constraints to this problem to get a structured pallet. Usually a pallet layer can be formulated as an integer programming problem and a feasible packing can be found. This is referred to as the cutting problem. There are two widely studied cutting problems: *guillotine* and *non-guillotine* cutting problems. Guillotine patterns refer to pattern that are cuttable. Most bin packing algorithms will have a two step process: one which selects the most profitable items to go in a bin followed by a feasibility check to see if these items can fit the bin.

Integer Programming formulations for packaging have been studied by [1, 13, 3]. Algorithms that solve the 2DKP using a branch and bound algorithm and then run a feasibility check which checks for overlaps for every new assignment are described in [9, 10, 8]. An enumeration technique for checking feasibility of an assignment is shown in [24]. Recent work in using advanced graphical technique to feasibility check an assignment called sequence triple is introduced in [7]. [6] is some earlier work that discusses some exact algorithms and heuristic techniques to solve the packaging problem.

5.1 Heuristics for the packing problem

The problem we consider in this section is that of selecting a subset of items and assigning coordinates (x_j, y_j, z_j) to each item, such that no item goes outside the bin, no two items overlap and the total volume of the items does not exceed the maximum capacity. We assume that the origin of the coordinate system is in the left-bottom-back corner of the bin. We have the obvious constraints

$$\begin{aligned} 0 \leq x_i \leq W - w_i, \\ 0 \leq y_i \leq H - h_i, \\ 0 \leq z_i \leq D - d_i. \end{aligned}$$

To ensure that no two packed boxes i, j overlap we will add more constraints

$$\begin{aligned} x_i + w_i \leq x_j, \\ y_i + h_i \leq y_j, \\ z_i + d_i \leq z_j, \\ x_j + w_j \leq x_i, \\ y_j + h_j \leq y_i, \\ z_j + d_j \leq z_i. \end{aligned}$$

These ensure that the boxes i and j are packed and that they must be located on the left, right, up, down, above or below each other. In the packaging sense, these constraints are enough. Many methods that solve the bin packing problem [7, 22] use the above constraints. However, in practice there are many other issues that concern the stability of a bin or pallet. These can be added as additional constraints to the above integer programming problem. The stability of a pallet is more when it has a lower center of mass, the distribution of pressure and weight is even and there are interlocking boxes. Interlocking can be guaranteed by maximizing the surface area of a box that will touch other boxes. Many of these factors are application specific but the problem of assigning constraints to maximize stability of a pallet remains an open problem.

Heuristic techniques use statistics to determine the most optimal packaging. As the size of the problem increases, exact algorithms' runtime complexity increases exponentially. Here, heuristic solutions are very popular. [7, 29] use a heuristic simulated annealing procedure to pack boxes together. [22, 14] presents a good overview of heuristic techniques to solve the bin packing problem.

5.2 On-Line bin packing problem

Consider a bin packing example, where boxes are coming down a conveyor and a pick and place robot is arranging the boxes onto a pallet. In this scenario the data of the items and their number is unknown. As soon as the item is seen by the sensor the on-line planner has to decide whether to pack the item or discard it. In literature on-line algorithms are analyzed for their worst case when their solutions is compared with the optimal solution given by an exact algorithm with complete input data. This analysis is called the *competitive analysis* and it is widely studied and surveyed in [2, 11].

[18] provides a brief overview of this problem and some solutions. A programming compiler usually can optimize the program by modeling breaks in the program. This way it can avoid processor cache misses and improve efficiency of the processor. Similarly a KP algorithm can also create a stochastic model of the distribution of profit and weights of items. This can give rise to a simple on-line greedy algorithm. From the a-priori knowledge of the distribution one can determine a threshold for efficiency given by equation (15). The planner will pack all items that have efficiency more than this threshold. If the knowledge about distribution of profits and weights is known a lower bound can be determined on the performance on this algorithm as shown in [23].

5.2.1 Time Dependent On-Line bin packing problem

In this model the time dependence is explicitly taken into account. For example on a shipping yard, transportation requests are made randomly and they have to be accepted or rejected without delay. If they are accepted they consume a resource and gain some price, but if they are rejected then their is no resource lost but price is lost due to cost in storage space, customer goodwill etc.. [18]. These problems are formally known as *dynamic and stochastic knapsack problems* and they were extensively studied in [20, 26, 19].

In stochastic models discussed in the literature the requests arrive by a stochastic process, usually a Poisson process. The entire setup can be modeled as a Markov decision process. The time dependence does not allow us to compare performance of these algorithms with the classical knapsack problem. However, the achieved results contain general characterizations of optimal policy and optimal threshold policy for greedy algorithms. Some recursive algorithms in the context of freight transportation and scheduling in batch processes are discussed in [19]. A similar problem in the context of airline yield management problem is discussed in [30]. They also extend their work to include a stochastic version of the multidimensional knapsack problem (d-KP).

Recent work in designing algorithms for solving the palletizing problem on-line uses statistical methods. The idea behind these methods is very intuitive. In greedy methods items need to be arranged in the descending order of their efficiency and then they are added in the bin in order. In the on-line variation, statistical methods are used to predict the rank of an item around the last n items based on previous observations. If the efficiency of an item is below the threshold efficiency it is discarded, otherwise it is added to the bin.

6. OPEN PROBLEMS

Many algorithms and techniques to solve logistics and mixed palletizing are industry specific. On of the advantages of providing a theoretical context is to generalize this problem and use a theoretical framework to improve logistics. However there are many open questions. As we discussed in section 3.4 every industry is different and converting their entire logistics into constraints that can be fed to a combinatorics problem is an open question. In the context of mixed palletizing the most important question remains of improving stability. There are very few solutions that also accommodate stability into their problem formulation.

7. CONCLUSIONS

We showed how the logistics planner can be formulated as the knapsack problem. We can use this to determine theoretical performance measure over a logistics system. We also introduced a 3D bin packing algorithm and provided a comprehensive reference to some of the latest work in combinatorics for solving it.

We have tried to generalize the common planning problem in industries so that they can be studied theoretically. Our future work will involve understanding and surveying various industries and grounding their logistics planner in combinatorics. We also think that if the gap between industrial logistics and theoretical computer science research were closed, we can widely improve the scope of industrial automation.

8. ACKNOWLEDGEMENT

The work has been sponsored by the United States Department of Commerce/National Institute of Standards and Technology (NIST) under the grant - Logistics Design and Benchmarking Manufacturing System. Their support is gratefully acknowledged.

9. REFERENCES

[1] J. Beasley. Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research Society*, pages 297–306, 1985.

[2] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis.* Cambridge University Press Cambridge, 1998.

[3] M. Boschetti, A. Mingozzi, and E. Hadjiconstantinou. New upper bounds for the two-dimensional orthogonal non-guillotine cutting stock problem. *IMA Journal of Management Mathematics*, 13(2):95, 2002.

[4] A. Caprara, H. Kellerer, U. Pferschy, and D. Pisinger. Approximation algorithms for knapsack problems with cardinality constraints. *European Journal of Operational Research*, 123(2):333–345, 2000.

[5] T. Cormen. *Introduction to algorithms.* The MIT press, 2001.

[6] K. Dowsland and W. Dowsland. Packing problems. *European Journal of Operational Research*, 56(1):2–14, 1992.

[7] J. Egeblad and D. Pisinger. Heuristic approaches for the two-and three-dimensional knapsack packing problem. *Computers & Operations Research*, 36(4):1026–1049, 2009.

[8] S. Fekete and J. Schepers. A new exact algorithm for general orthogonal d-dimensional knapsack problems. In *AlgorithmsâĂŤESA'97*, pages 144–156. Springer, 1997.

[9] S. Fekete and J. Schepers. On more-dimensional packing III: Exact algorithms. *Discrete Applied Mathematics*, 1997.

[10] S. Fekete, J. Schepers, and J. van der Veen. An exact algorithm for higher-dimensional orthogonal packing. *Arxiv preprint cs/0604045*, 2006.

[11] A. Fiat and G. Woeginger. *Online algorithms: The state of the art.* Springer Berlin, 1998.

[12] C. Guéret and C. Prins. A new lower bound for the open-shop problem. *Annals of Operations Research*, 92:165–183, 1999.

[13] E. Hadjiconstantinou and N. Christofides. An exact algorithm for general, orthogonal, two-dimensional knapsack problems. *European Journal of Operational Research*, 83(1):39–56, 1995.

[14] E. Hopper and B. Turton. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *European Journal of Operational Research*, 128(1):34–57, 2001.

[15] O. Ibarra and C. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM (JACM)*, 22(4):463–468, 1975.

[16] H. Kellerer and U. Pferschy. A new fully polynomial approximation scheme for the knapsack problem. *Approximation Algorithms for Combinatiorial Optimization*, pages 123–134, 1998.

[17] H. Kellerer and U. Pferschy. Improved dynamic programming in connection with an FPTAS for the knapsack problem. *Journal of Combinatorial Optimization*, 8(1):5–11, 2004.

[18] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack problems.* Springer Verlag, 2004.

[19] A. Kleywegt and J. Papastavrou. The dynamic and stochastic knapsack problem. *Operations Research*, 46(1):17–35, 1998.

[20] A. Kleywegt and J. Papastavrou. The dynamic and stochastic knapsack problem with random sized items. *Operations Research*, pages 26–41, 2001.

[21] E. Lawler. Fast approximation algorithms for knapsack problems. In *18th Annual Symposium on Foundations of Computer Science, 1977.*, pages 206–213, 1977.

[22] A. Lodi, S. Martello, and D. Vigo. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing*, 11(4):345–357, 1999.

[23] A. Marchetti-Spaccamela and C. Vercellis. Stochastic on-line knapsack problems. *Mathematical Programming*, 68(1):73–104, 1995.

[24] S. Martello, M. Monaci, and D. Vigo. An exact approach to the strip-packing problem. *INFORMS Journal on Computing*, 15(3):310, 2003.

[25] S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations.* 1990.

[26] J. Papastavrou, S. Rajagopalan, and A. Kleywegt. The dynamic and stochastic knapsack problem with deadlines. *Management Science*, 42(12):1706–1718, 1996.

[27] S. Sahni. Approximate algorithms for the 0/1 knapsack problem. *Journal of the ACM (JACM)*, 22(1):115–124, 1975.

[28] S. Taboun and S. Bhole. A simulator for an automated warehousing system. *Computers & Industrial Engineering*, 24(2):281–290, 1993.

[29] R. Tsai, E. Malstrom, and W. Kuo. A three dimensional dynamic palletizing heuristic. *Progress in Material Handling and Logistics*, 2:181–201.

[30] R. Van Slyke and Y. Young. Finite horizon stochastic knapsacks with applications to yield management. *Operations Research*, 48(1):155–172, 2000.

[31] H. Yaman and A. Sen. Manufacturer's mixed pallet design problem. *European Journal of Operational Research*, 186(2):826–840, 2008.