Decision Support For Packing In Warehouses

Gurdal Ertek and Kemal Kilic

Sabanci University
Faculty of Engineering and Natural Sciences
Orhanli, Tuzla, 34956, Istanbul, Turkey

Decision Support For Packing In Warehouses

Gürdal Ertek and Kemal Kilic

Sabanci University
Faculty of Engineering and Natural Sciences
Orhanli, Tuzla, 34956, Istanbul, Turkey

Abstract. Packing problems deal with loading of a set of items (objects) into a set of boxes (containers) in order to optimize a performance criterion under various constraints. With the advance of RFID technologies and investments in IT infrastructures companies now have access to the necessary data that can be utilized in cost reduction of packing processes. Therefore bin packing and container loading problems are becoming more popular in recent years. In this research we propose a *beam search* algorithm to solve a packing problem that we encountered in a real world project. The 3D-MBSBPP (Multiple Bin Sized Bin Packing Problem) that we present and solve has not been analyzed in literature before, to the best of our knowledge. We present the performance of our proposed beam search algorithm in terms of both cost and computational time in comparison to a greedy algorithm and a tree search enumeration algorithm.

1 Motivation

Managers of a major automobile manufacturer located in Bursa, Turkey, are undertaking an efficiency improvement program at their spare parts warehouse. Order packaging is one of the crucial processes that require further improvements, particularly for those items that are urgently demanded by their major service parts dealers. Everyday over 80 major service dealers, located at different cities in Turkey, submit their requests of urgent orders to the warehouse. These orders are pooled daily in an information system until 3:00 p.m. Then, the workers start picking the requested items and packing them into adequately sized boxes. By 5:00 p.m., trucks of a third-party logistics firm collect the boxes from the warehouse to deliver them to their destinations. The effectiveness of the order handling process during the two hours between 3:00-5:00 p.m. depends on the efforts of a workforce team of eight to ten workers and on the accuracy of the decisions regarding the selection of appropriate box types and quantities. Currently, at 3:00 p.m., two experienced foremen spend approximately 20 minutes on a PC to decide on and record the choice of boxes for each order. The foremen strive, solely based on their experience and wisdom, for determining a suitable set of boxes such that the posterior extra work of reallocating the items between boxes can be avoided. However in the current state of the operations, the warehouse workers still spend approximately 20 critical minutes each day for such

reallocations. The warehouse management decided to initiate the development and implementation of a decision support system that would help the foremen in their decision making and would eliminate the item reallocation process. Our motivation for this research is based on a project targeted at realizing this vision of the warehouse managers.

The heart of the decision support system to be developed is a metaheuristic algorithm which makes the decisions of how many boxes are used of each box type and how each item is placed in its box. The objective is the minimization of the total cost of the boxes used. The time limitations of the packing process (which has to be completed within two hours) oblige us to develop an algorithm that offers real-time solutions. In this paper we present three alternative algorithms that can serve as the solution engine of the decision support system, and compare them with respect to solution and running time performance.

2 Problem Definition

Everyday, major service parts dealers send urgent order requests that have to be handled separately from regular orders. The sizes and item characteristics of the urgent orders depend on the service dealers and exhibit seasonality. In general the contents of the orders range from a few items to nearly a hundred items. In this paper, we consider the packing problem for a single order, since the more general problem of meeting the orders of all the dealers with minimal box cost is decomposable to the independent problems of preparing each of the orders.

We have a set of n rectangular objects, i.e., items, $(O = \{o_i\}_{i=1,..,n})$, each with a height (h_i), a width (w_i) and a depth (d_i), that can be packed into a set of m larger rectangular objects, i.e., boxes, $(C = \{c_j\}_{j=1,..,m})$, each with a height (H_j), a width (W_j) a depth $(D_j)$and a cost (C_j). Our objective is to allocate the items into the boxes so that the total cost of boxes that are used is minimized. All the items within an order must be included and should not overlap with each other inside the boxes. We consider only the orthogonal arrangements of the items and allow them to be rotated around all three axis. The problem is a member of the cutting and packing problems family, namely a bin packing problem. Further refined typology of the problem is provided in the next section.

There are currently 22 different box types used for the urgent deliveries in the warehouse. However most of these are designed for special items and need not be considered in our study. Since only six of these box types are frequently used we have carried out our experiments for only six box types in our study, as well. In 2005, the cost of the boxes that are used for the shipment of the urgent orders alone was in the scale of tens of thousands of dollars. When the cost of labor is added to this figure, the total cost figure for handling urgent orders ramps even further. Thus, the decision support system we plan to develop and implement has the potential to bring significant savings since it will reduce both the cost of boxes and the cost of labor. Even though our research currently focuses only on the urgent orders, it is the forerunner of a more extensive research where the

problem of efficiently packing regular orders (which have higher costs) will be handled.

In the warehouse the fragile items are packed separately. This constraint will be handled in the application with a simple if-then rule and is neglected in the scope of the paper. Therefore, we assume at this stage that all the items of an order can be packed together. Thousands of items with varying shapes and dimensions are present in the warehouse. However, a majority (%80-%85) of these items come in rectangular boxes. For the rest of our paper we assume that all the items have rectangular shape. This assumption was approved by the warehouse managers to be non-restrictive and is required for our computations.

3 Relevant Literature

The strong duality relation between the packing problems and the cutting problems makes it impossible to separate the research conducted on either of the two problems. While the packing problems deal with optimal arrangement of a set of smaller items (O) within larger items (C), the cutting problems aim to cut a larger item (stock) into smaller pieces.

The cutting and packing problems are some of the most extensively studied combinatorial optimization problem in the literature. In the editorial of the Special Issue of the European Journal of Operational Research on Cutting and Packing, Biscкhoff and Wäscher [2] present three reasons for the popularity of these problems. First of all, the cutting and packing problems have a wide range of applications in industry. This is quite true when one considers that the applications extend from production and distribution problems in supply chain logistics (e.g., cutting metal sheets into smaller pieces, order packaging in warehouses, loading the containers to trucks or ships, etc.), to applications as diverse as computer science and finance (e.g., memory allocation of processors [4], and capital budgeting problems [7]).

Second reason is the fact that the majority of such problems are strongly NP-Hard. Therefore, they receive a great deal of attention from researchers who focus on exact algorithms, as well as those who develop heuristic solutions for computationally complex problems.

The third reason is the diversity of the real world problems. Even small nuances in the objective function or the packing/cutting constraints result in new problem structures. In the knapsack packing problem, the space is limited and the goal is to choose the most valuable *(i.e., a set of)* small objects that would maximize the utility of the load packed into the larger object. On the other hand, the bin packing problems deal with the problem of packing *all* of the smaller objects so that the number of larger objects required is minimized. A special case is the pallet loading problem in which the smaller objects are usually *identical* or *weakly heterogenous* (relatively few small item types) and the objective is minimizing the number of *identical* (generally speaking) pallets required.

Since the seminal papers of Gilmore and Gomory in early 60's, over a thousand papers are published in problems related to cutting and packing problems. Such an extensive literature with many varieties of the problems necessitates a good typology. To our knowledge, two well-accepted typologies are present in the literature [5, 15]. The most commonly accepted (and the earliest) is due to Dyckhoff [5] in which he proposes four criteria according to which the problems can be categorized; (1) Dimensionality (1D, 2D, 3D, ND) (2) Kind of assignment (all large objects and selected small objects - **B**, or all small objects and selected large objects -**V**) (3) Assortment of large objects (**O**ne, **I**dentical, **D**ifferent) and (4) Assortment of small objects (**F**ew items of few sizes, many items of **R**elatively few sizes, **M**any items of many sizes, **C**ongruent items).

In a recent paper, Wäscher et al. [15] argues that the classification scheme of Dyckhoff has major drawbacks in certain aspects such as inconsistency, lack of uniqueness and homogeneity. They improve the classification and propose a new typology similar to the Dyckhoff's in certain aspects but complements its deficiencies.

Given the overwhelming size of the literature, we refer the interested readers to one of the following extensive reviews: A historical development of the cutting and packing problems, with a brief discussion of the seminal and most important papers in major problem categories are presented by Dyckhoff et al.[7]. In an extensive review of the literature, Dyckhoff and Finke [6] investigate 308 papers that were published prior to 1992. They also provide a classification of the literature based on the typology proposed by Dyckhoff. In [15], 413 papers are reviewed and a classification based on the improved typology is provided.

Based on the typology of Dyckhoff [5], the problem studied in this paper is categorized as **3D/V/D/M**. None of the papers published prior to 1992 falls into this category. Actually, even three-dimensional packing problems were rarely studied prior to 1992. According to the typology of Washner[15], our problem is **3D-MBSBPP** (Multiple Bin Sized Bin Packing Problems). Again there are no papers published in this category between 1994-2004 [15]. Out of the 413 papers only four papers were published in the MBSBPP category and among these four papers, three of them were single-dimensional and the fourth one was two-dimensional. The closest category is 3D-MHLOPP (3D-Multiple Heterogenous Large Objects Placement Problems), which is basically a generalization of 3D-multiple container or pallet loading problems.

The object placement problems are categorized as the problems in which smaller objects have limited number of types (*weakly homogenous*) so that the proposed solutions exploit this constraint. The papers of Ivancic et al. [9] and Eley [8] fall into the 3D-MHLOPP category. Considering relatively few small number of items allows limited number of preassigned patterns of these small items and eases the computations. Being published in 2005, the work of Brunetta and Grégoire [3] was not included in the review, but also falls into 3D-MHLOPP category. Motivated from a real case in a biscuit factory with 15 different smaller objects size (in this case the smaller objects are boxes of sweeties), they propose

a tree-search algorithm that implicitly explores the solution space in order to maximize volumetric utilization of multiple sized containers.

Another notable relevant paper is due to Martello et al. [11], in which the authors propose an exact solution for 3D-(Single Bin Sized) BPP.

To the best of our knowledge, our paper is the first work that proposes solution approaches to the 3D-MBSBPP.

4 Methodology

Let us assume that we have an order request $O = \{o_1, o_2, .., o_n\}$, a set of n items with dimensions (h_i, w_i, d_i) and that we have m boxes ($C = \{c_1, c_2, .., c_m\}$) with dimensions (H_j, W_j, D_j) and costs C_j. The problem is identifying the number of boxes of each type that will be used to fully ship a given order while minimizing the cost of boxes. We propose three solution algorithms, namely a greedy algorithm (G), a beam search algorithm (BS) and a tree search based implicit enumeration algorithm (TS).

The greedy algorithm (G) is based on a sequential assignment methodology. In this methodology, while there are objects that are not packed into a box, a single sized bin container loading problem (SSBCLP) is solved independently for each different box size, and the best box is selected according to a performance index ($I_{j,l}$). We used the tree search heuristic proposed by Pisinger [13], which loads the objects to the boxes layer by layer (guillotine cuttable) as the container loading algorithm (CLA). After the boxes are loaded independently, the best box is selected based on an index which is a function of the *filling ratio* and the *cost per volume* of the boxes. Next, those objects that are assigned to a box are deleted from the waiting list and the same procedure is applied sequentially. The filling ratio is defined as follows:

Let W_l denote a subset of item set (O) that contains the items that are not still packed in a box at level l. Let $A_{j,l}$ be a subset of W_l, which contains all of the items that are packed in the j^{th} box by the container loading algorithm (CLA) at level l. The filling ratio, $q_{j,l}$, of the j^{th} box type at level l is calculated as follows;

$$q_{j,l} = \frac{\sum_{i=1}^{n} (V_i \times X_{i,j,l})}{V_j} \tag{1}$$

Where V_i is the volume of the i^{th} object ($V_i = h_i \times d_i \times w_i$), V_j is the volume of the j^{th} box type ($V_j = H_j \times D_j \times W_j$) and $X_{i,j,l}$ is the binary variable that equals to 1 if the CLA loads the i^{th} object to the j^{th}container at the l^{th} level, and 0 otherwise. Considering the overall objective of minimizing the total cost of the boxes that are used for the shipment, we also included the cost per volume ($p_{j,l}$) in our performance index. Cost per volume for each container size is calculated as follows:

$$p_{j,l} = \frac{C_j}{\sum_{i=1}^{n} (V_i \times X_{i,j,l})} \tag{2}$$

The proposed performance index is calculated as follows;

$$I_{j,l} = \frac{p_{j,l}}{q_{j,l}} \tag{3}$$

After the performance indices of each of the boxes are calculated, the box with the lowest index is selected (c_{j^*}) as the next assignment. Later, the objects $A_{j^*,l}$ that are loaded to the selected box are deleted from W_l and the remaining items are considered as W_{l+1} (i.e., $W_{l+1} = W_l \backslash A_{j^*,l}$). This procedure is repeated until $W_{l+1} = \emptyset$.

The second algorithm we propose is a filtered beam search heuristic (BS). Beam search is a special type of tree search heuristic that mimics the infamous branch and bound algorithm. In the case of branch and bound one can prune the nodes after a guarantee of non-optimality, on the other hand in beam search the less promising nodes that do not seem to be leading to the optimal decisions are pruned. In this regard, beam search algorithm is faster but typically does not yield a solution as good as TS. Beam search was first used by Lowerre in a speech recognition program called HARPY [10]. Ow and Morton [12] present a thorough analysis of a filtered beam search methodology for different scheduling problems. Aktürk and Kilic [1] applied filtered beam search to a space mission scheduling problem and demonstrated that it outperforms other metaheuristics such as GRASP and Simulated Annealing for their problem.

A conventional filtered beam search has two decision parameters, the *beam width* (b) and the *filter width* (f). The most promising b nodes are determined based on a heuristic *global evaluation function,* which is typically the objective function value of a heuristic solution. Even though the heuristics utilized for this purpose are quite fast, they still might be undesirably time consuming and thus infeasible for real-time decision making, in particular for large scale problems. In order to overcome this problem, a second parameter f is introduced, which limits the number of the nodes where the *global evaluation function* is calculated. Under a *filtered* beam search scheme, at each level, firstly, the most promising f nodes are determined based on a *local evaluation function* and the rest of the nodes are pruned. Next among the filtered f nodes, the b most promising ones are selected based on a *global evaluation function.* In the proposed algorithm, the index introduced in Equation 3 is used as the local evaluation function, i.e. the filtering stage. On the other hand, the total cost of the boxes of the solution that is obtained by applying the greedy algorithm to the non-pruned nodes is utilized as the global evaluation function. Below we discuss the algorithm and the notation in detail.

Let $W_{t,b}$ be the set that contains the items that are still not packed in a box at level t for the b^{th} beam of the tree. We intentionally denote the level by t, rather than l, so as to distinguish the description of the current algorithm (BS)

from the previous (G). Let $S_{t,b}$ be the set of boxes that are already packed with objects prior to level t for beam b. The proposed algorithm is as follows;

Algorithm Beam Search.

Select f and b sizes
Initialize $t = 1$, $W_{1,b} = \{o_1, o_2, .., o_n\}$ $\forall\ b$,
While not *done do*

1. Procedure Filter_with_one_step
   (a) For all beams, $k = 1..b$ repeat the following
      i. For all $j = 1..m$ do the following
         Apply the container loading algorithm (CLA) with the set $W_{t,k}$
         Determine the local evaluation score, $I_{j,t,k}$
   (b) Select f boxes with the best (lowest) $I_{j,t,k}$
2. Let $F_{t,b}$ be the set of the indices of the boxes that are selected at step 1b on the b^{th} beam.
3. $done = true$
4. Procedure Evaluate_the_global_evaluation_score_of the_nodes
   (a) For all beams, $k = 1..b$ repeat the following
      i. For each $j \in F_{t,k}$ repeat the following
         Develop an auxiliary problem with the remaining items that are not packed yet.
         Apply the greedy algorithm (G) and find a heuristic solution
         Evaluate the total cost of boxes (including the ones that assigned prior to that node)
   (b) Select b nodes yielding the lowest cost evaluated in step 4(a)i
   (c) Let $B_{t,b}$ be the set of the indices of the boxes on the b^{th} beam that are selected at step 4b
   (d) For all beams, $k = 1..b$ repeat the following
      i. For each $j \in B_{t,k}$ do the following
         Augment the node (j^{th} box) to the k^{th} beam.
         Let $P_{t,j,k}$ be the set of the *items* assigned to the f^{th} box of the k^{th} beam at step 1(a)i
         Let $W_{t,b} = W_{t,b} \setminus P_{t,j,k}$
         If $W_{t,b} \neq \{\}$ then $done = false$
5. Let $t = t + 1$

The third algorithm we propose is a depth-first tree search algorithm (TS) that implicitly enumerates the solution space. The nodes at each level of the tree hold information on which box types have been used until themselves, and a list of the remaining items to be allocated. The nodes are generated from their parents for each feasible box type. The best solution at any stage of the algorithm is used as an upper bound, and any children nodes that have a worse solution than the upper bound are pruned all-together with their subtrees. When a leaf node is reached the algorithm evaluates the cost of the complete allocation, compares it to the best solution until then, returns back to the leaf node's parent, and continues with generating the next node that allocates the remaining items in the parent to next box type.

	Case I			*Case II*			*Case III*		
	G	*BS*	*TS*	*G*	*BS*	*TS*	*G*	*BS*	*TS*
$n = 20$	2.20	1.78	1.78	3.83	3.54	3.54	7.04	6.62	6.56
$n = 40$	3.64	3.00	3.00	6.77	6.17	6.16	12.69	12.25	$*$
$n = 60$	5.04	4.25	$*$	9.32	8.76	$*$	19.23	12.11	$*$

Table 1. The performance of the algorithms in terms of the total costs (appr. in $).

5 Experimental Results and Discussion

The algorithms presented in the previous section were coded with C under the MS Visual Studio .NET environment and run on a PC with Intel Centrino 1700 Mhz processor and 512 MB of RAM. As a subroutine of our program we have used the C implementation of Pisinger's algorithm for container loading (CLA) [13], which we acquired from the Internet [14]. In this section we present the results of our experiments and compare the performances of the proposed algorithms with respect to costs of the boxes and computational times. The company currently doesn't store the dimensions of the items in its warehouse and is in the process of acquiring this data from its suppliers. Thus we generated the item sizes randomly within given bounds so as to represent various settings. We alos opted for using box types that we generated, whose volumes are roughly uniformly distributed. To determine the box costs, we initially plotted the costs of the 22 box types used in the warehouse against their volumes and found out that a power function in the form of $C(V) = aV^b$ fits the data extremely well. In this function $C(V)$ refers to the cost of the box, V refers to the volume of box, and a and b are constant real numbers. The costs of the generated box sizes were estimated according to this function.

The experiments were designed to cover the environment with regard to two aspects. Firstly, the order size, i.e., the number of items to be packed. For this purpose three different problem sizes were considered: Small order size ($n = 20$), medium order size ($n = 40$) and large order size ($n = 60$). The second aspect is the dimensions of the items compared to the dimensions of the boxes. Again three different problems are generated: Orders with smaller items (Case I), medium items (Case II) and larger items (Case III), in which the three dimensions of the items are randomly generated within the ranges [10-25], [15-30] and [20-35], respectively (in centimeters). For each experiment (altogether $3 \times 3 = 9$ cases), 10 instances were randomly generated. The execution of the TS was terminated after 5 minutes in solving more difficult problem instances. The results of the algorithms in terms of the average costs (approximately in $) and the computational times (in seconds) are tabulated in Tables 1 and 2. For illustrative purposes, we depict the solution of the TS for a problem instance with 40 items in Figure 1. In this solution two large boxes of the same type, and two smaller boxes of different types are used to fully pack the sample order.

As expected, GS has the worst performance in terms of the total costs, but is very fast. On the other hand, the implicit enumeration based TS requires exten-

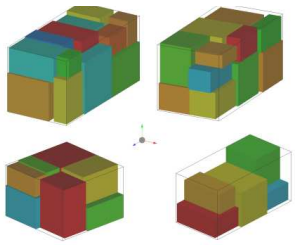

Fig. 1. The solution of the TS algorithm for a test instance with 40 items.

	Case I			*Case II*			*Case III*		
	G	*BS*	*TS*	*G*	*BS*	*TS*	*G*	*BS*	*TS*
$n = 20$	0.28	1.12	0.31	0.47	3.71	1.05	0.09	2.85	14.52
$n = 40$	2.85	19.07	3.88	0.65	13.77	30.35	0.19	8.60	*
$n = 60$	3.80	40.99	*	1.11	24.02	*	0.33	20.07	*

Table 2. The performance of the algorithms in terms of the CPU time (seconds).

sive computational time and is not feasible to be used for orders with many items or with larger items. Both of these situations require usage of more boxes, which explodes the number of nodes in the search tree of TS. The greedy algorithm (G) and the beam search algorithm (BS) handle such situations efficiently, with G naturally running much faster.

One may surprisingly observe that TS runs faster than BS on the average for some instance sets of Case I and Case II. We believe that this is due to the fact that we have listed the box types in descending order with respect to their volumes in the input file. TS starts by allocating the items to the largest boxes and quickly obtains a good upper bound, which enables it to prune most of the search tree.

It should be noted that BS performs significantly better than G for $n = 60$ in Case III, with only a slight increase compared to its running time for $n = 40$. BS yields results very close to TS and returns results fastly enough (¡25 seconds on the average) to be used in the company. In implementing the decision support system, one should specify the ranges of order sizes and items sizes in which each algorithm should be run.

6 Conclusion and Future Work

In this paper, being motivated by a real world case, we described the 3D-MBSBP problem and proposed three algorithms to solve it. Future stages of our research will involve implementing a functional and usable decision support system with a friendly GUI (Graphical User Interface), with the developed algorithms residing in it as the solution generating engine. Another future work is the improvement of our algorithms and software implementations to run more efficiently and produce

better solutions. Finally, we are envisioning the extension of our work to scale up to the rest of the warehouse and yield significant monetary and ergonomic benefits.

7 Acknowledgement

The authors would like to thank Professor David Pisinger at University of Copenhagen for making the code of his container loading algorithm available on the Internet. The authors would also like to thank Sabanci University undergraduate student Cem Sözeri for the 3D illustration of the sample solution, and Mr. Sinan Südütemiz and Mr. Rıza Altuğ from the company for their assistance during the project.

References

1. Akturk, M.S., Kilic, K.: Generating short-term observation schedules for space mission projects. Journal of Intelligent Manufacturing, **10**(5), (1999) 387–404
2. Biskhof, E. E., Wäscher, G.: Cutting and packing. European Journal of Operational Research. **84**, (3) (1995) 503–505
3. Brunetta, L., Grégoire, P. A.: General purpose algorithm for three-dimensional packing. INFORMS Journal on Computing. **17**, (3) (2005) 328–338
4. Chung, F., Graham, R. and Varghese, G.: Parallelism versus memory allocation in pipelined router forwarding engines. ACM Symposium on Parallel Algorithms and Architectures, Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures, Barcelona, Spain (2004) 103–111
5. Dyckhoff, H.: A typology of cutting and packing problems. European Journal of Operational Research. **44**, (1990) 145–159
6. Dyckhoff, H., Finke, U.: Cutting and Packing in Production and Distribution. Springer Verlag, Berlin (1992)
7. Dyckhoff, H., Scheithauer, G., Terno, J.: Cutting and Packing. Annotated Bibliographies in Combinatorial Optimization. Dell'Amico, M., Maffioli F., and Martello, S. (eds) Wiley (1997) 393–412
8. Eley, M.: A bottleneck assignment approach to the multiple container loading problem. OR Spectrum **25** (2003) 45–60
9. Ivancic, N.J., Mathur, K., Mohanty, B.B.: An integer-programming based heuristic approach to the three-dimensional packing problem. Journal of Manufacturing and Operations Management **2**, (1989) 268–298
10. Lowerre, B.: The HARPY speech recognition system. Ph.D. Thesis, Carnegie Mellon University, PA (1976).
11. Martello, S., Pisinger, D., Vigo, D.: The three-dimensional bin packing problem. Operations Research, **48** (2), (2000) 256–267
12. Ow, P. S., Morton, T. E.: Filtered beam search in scheduling. International Journal of Production Research. **26**, (1988) 35–62
13. Pisinger, D.: A tree-search heuristic for the container loading problem. Ricerda Operativa. **28** (87), (1998) 31–48.
14. `http://www.diku.dk/~pisinger/`
15. Wäscher, G., Haußner, H., Schumann, H.: An improved typology of cutting and packing problems, Working Paper No. 24. Otto von Guericke University, Magdeburg (2006)