A Heuristic Algorithm for Carton to Pallet Loading Problem

Vu T. Le1, Doug Creighton, Saeid Nahavandi
Intelligent Systems Research Lab, Deakin University, Geelong, Australia

Abstract— **This paper presents an algorithm used to solve a carton to pallet packing problem in a drink manufacturing firm. The aim was to determine the cartons loading sequence and the number pallets required, prior to dispatch by truck. The algorithm consists of a series of nine parts to solve this industrial application problem. The pallet loading solution relatively computationally efficient and reduces the number pallets required, compared to other 'trail and error' or manual spreadsheet calculation methods.**

Index Terms—**Multi-item, multi-class, multi-group, bin packing, pallet allocation.**

I. Introduction

The bin-packing problem is very popular amongst researchers [1-3, 5-7]. The problem usually involves the packing of objects of different sizes into some finite confined space. That is, there consists a set of objects to be placed into a set of bins. In this work we study a drink carton to pallet packing problem in a busy drink manufacturer. The large scale of the drinks manufacturing operation means that many trucks would come to pickup the finish drink products and deliver to the assigned destination every working day of the week. The amount of stock required for dispatch is such that it was necessary to design rules to minimize the loading calculation time, while reducing the number of pallet required before loading onto truck. This in turn reduces the number of delivery trucks, and consequently long-term transportation costs.

Most bin-packing problems involve the assignment of objects with different dimensions. These need to be arranged and stacked in a way that minimises the total bin usage or fits as many objects into a determined dimension bin. One of the earlier authors who looked at these bin-packing problem was Korf [2]. He developed an algorithm to packing unoriented rectangles into unit square bins. Korf found that his algorithm provides a similar result for packing 24 squares into a 70x70 square bin to the earlier hand calculated solution by Gardner [3]. Manyem [4] studied a one-dimensional, NP-hard online bin-packing problem. He provided several modified heuristic algorithms to solve different what-if on arrival object length scenarios. Items with a longer length have the priority, being placed at the bottom of the bin. A population-based heuristic search algorithm that could be used for both three-dimensional pallet loading and two and three-dimensional multiple-destination bin packing problems was presented by VerWeij [5]. He demonstrated that the algorithm improved the packing result as the running time increases.

A NP-Hard Multi-Pallet Loading problem was studied by Terno et. al. [6], where a branch and bound algorithm was developed to arrange a set of distinct rectangular shaped products onto a restricted pallet space and to minimize the total number of pallets required in the system. Through experiments, the authors have shown that their designed algorithms were efficient. The only problem is that in some test cases it permitted the loading of rectangle product pieces in undesired overhanging positions. An earlier publication from the same authors, Scheithauer and Sommerwei [7], involved the development of a heuristic algorithm. The algorithm was based on the G4-heuristic algorithm for pallet loading problem to pack and aligning smaller rectangular shape into a larger rectangle shape while maximizing the utilization area. Another paper in the pallet loading problem chain, by Scheithauer and Terno [8], provides a heuristic algorithm for two-dimensional pallet loading problem. Other authors who studied the pallet loading including NeliBen [9], who developed a hybrid of recursion heuristic and branch and bound algorithms to allocate rectangular box onto pallet by computing the upper bound solution. Through experiments Naliben [9] has shown that the developed algorithms were able to solve problem instances for which other known other heuristic algorithm fail.

A simple but efficient algorithm for pallet loading problem is presented in this paper. Experimental results and discussion from factory floor data running the algorithm are also given.

II. The Basic Rule And Structure

Given that there is a set of all products, includes boxes of different sizes and quantities, where each set of similar products has a unique product number. A subset of product number must be prepared for loading onto pallets, before being loaded on to trucks. In this carton to pallet loading problem, a full pallet is defined as *100% loaded*. Each box of different products, when allocated to a pallet, will take up a certain percentage on the pallet, given by a volume product cube percentage value. The product structure for each product is given in Table 1.

0-7803-9094-6/05/$20.00 ©2005 IEEE

Table 1. Product data structure

Data Structure Fields
Product Number
Product Description
Product Cube Percentage (%)
Product Group ID

In order to reduce the number of combinations in the algorithm, grouping of products with a similar cube size box together methodology was developed and provided as the input data. A unique *Product Group ID* assigned to each such group as shown in Table 1.

In Table 1 the *Product Group ID* referencing from the product group structure such that similar volume size drinks of different types are grouped together. The data schema of each of this product group is given in Table 2.

Table 2. Product group data structure

Data Structure Fields
Product Group ID
Product Group Description
Product Group Minimum Percent {%)
Product Class ID

The *Product Group Minimum Percent* is a rule that simply said that if the quantity of boxes of the same group contains greater than or equals to this percentage then the loaded pallet is acceptable.

In Table 2 the *Product Class ID* is a reference from the class schema. The class schema was derived such that, since some groups of soft-drink have a package box size similar to some group of beer and so on, therefore one could loaded different groups of products on the same pallets. The reason being that since one would like to keep all similar product boxes together as a whole. The product class data structure is given in Table 3.

Table 3. Product class data structure

Data Structure Fields
Product Class ID
Product Class Description
Product Class Minimum Percent (%)

In Table 3, the *Product Class Minimum Percent* means that if a loaded pallet of the same class is greater than this percentage then the pallet is acceptable.

The Tables 1 to 3 stored the standard sets of constant input data. These tables are the basic input framework that are used by the bin-packing algorithm.

III. THE ALGORITHM

The developed algorithm sequentially allocates product boxes to pallets using ten sub steps. These are summarised in Figure 1.

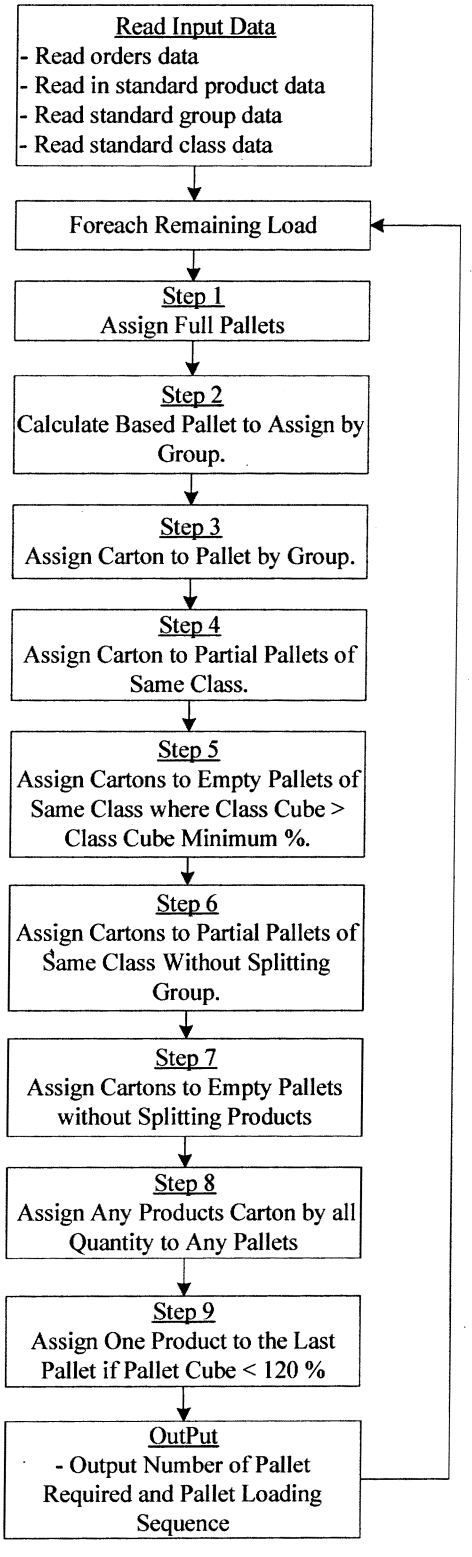

Figure 1: Overall box to pallet loading algorithm

From Figure 1 above, each load is the unique identifier that concatenate between the delivery date and location, which was read in from the input order data. The structure of the load identifier is it departing date and it's delivery location. The details of each step are described in the following proceeding sections.

1. Assign Full Pallets

The first step assigns any products order that has cartons quantity greater than or equal to 100% pallet utilization to a new pallet. Any left over cartons are allocated in later steps of the algorithm. The structure of the algorithm has been given in Figure 2.

```
calculate total product cube from input order file
for item = 1 to NoOfProduct
  oldProductList = OriginalProductList [i]
  CurrentBoxCube = oldProductList.CurrentCube
  while (CurrentBoxCube >=100)
      assign a new pallet
      CurrentBoxCube = CurrentBoxCube – 100
  add left over to product list
```

Figure 2: Assign similar full product orders to full pallet

2. Calculate Based Pallet to Assign by Group

Step 2 involves calculating the number of remaining cartons not allocated in Step 1. It also prepares the data for Step 3. The step starts with sorting the product by group and then by descending leftover cube percentage. If the group cube sum is greater than the minimum group cube specified in Table 2, it then sums the product group by cube, creating a virtual pallet count. The pseudo code defining this step is given in Figure 3.

```
sort left over product by group then by descending cube
for item i = 1 to LeftOverProductListCount
  Product =  LeftOverProductList[i]
  PGroup = ProductGroupList[i]
  PGroup.Group = Product.Group
  while (PGroup.Group == Product.Group)
    PGroup.Cube = Pgroup.Cube + Product .Cube
    i++
     if (i <= LeftOverProductListCount)
         Product = ProductGroupList[i]
    else
         break
  i--
  add product group data to product group list
for group i = 1 to ProductGroupListCount
  PGroup = ProductGroupList[i]
  if (PGroup.GroupCube > 100)
     PGroup .NoPalletRequired = round(GroupCube/100)
     PGroup.GroupCube -= GroupNoPalletRequired*100
  end if
  groupMinPercent = ProductGroupTable2.PGM
  if(PGroup.GroupCube < 100)
     if (PGroup.GroupCube >= groupMinPercent)
         PGroup.NoPalletRequired++
  ProductGroupList[i] = PGroup
```

Figure 3: Calculate base pallet to assign by group

3. Assign Carton to Pallet by Group

Step 3 groups the leftover cartons and assigns the number of cartons, by group to a new pallet. The pseudo code is given in Figure 4.

```
sort by group then by group cube descending
for group i= 1 to ProductGroupListCount
  PGroup = ProductGroupList[i]
   While(PGroup.NoPalletAssign< PGroup.NoPalletRequired)
     PGroup.NoPalletAssign++
     ProductGroupList[i] = PGroup
     CreateBasePalletData
     for item j = 1 to LeftOverProductListCount
       if(CurrentProductGroup = Pallet.Group)
         if (Pallet.Cube + CurrentProductCube <= 100)
           Pallet.Cube+= CurrentProductCube
           assign new or reassign to current pallet
           remove current group products
           j--
```

Figure 4: Assign carton to pallet by group

4. Assign Cartons to Partial Pallets of Same Class

This step assigns product to partial filled pallets of the same product class, where a complete product group can be assigned to that existing pallet. The step structure is shown in Figure 5.

```
calculate group list from left over product cartons
sort group list by class
calculate class list from left over group
sort class list by descending cube
sort pallet list descending order
for class i = 1 to ProductClassListCount
  PClass = ProductClassList[i]
  for pallet j = PalletListCount
    thisPallet = PalletList[j]
    if ( thisPallet.Cube < 100)
      if (thisPallet.Class == Pclass.Class)
        for group k = 1 to ProductGroupListCount
          PGroup = ProductGroupList[k]
          if (Pgroup.Class = Pclass.Class)
            if (thisPallet.Cube + Pgroup.Cube <= 100)
              thisPallet.Cube+= Pgroup.Cube
              PalletList[j] = thisPallet
                for item l = 1 to LeftOverProductListCount
                  Product = LeftOverProductList[i]
                  if (Product.Group == PGroup.Group)
                     remove current product
                     l--
```

Figure 5: Assign carton to existing pallet by class

5. Assign Cartons to Empty Pallets of Same Class where Class Cube > Class Cube Minimum %

This section of the algorithm attempts to assign *Class Cube Volume Percentage* to empty pallets, where the *Product Class Minimum Percentage* is satisfied. The program structure is given in Figure 6.

```
calculate group list from left over product cartons
sort group list by class then by cube
calculate class list from left over group
sort class list by descending cube
for class i = 1 to ProductClassListCount
  PClass = ProductClassList[i]
  classMinPercent = ProductClassTable3.PCM
  if(Pclass > classMinPercent)
        create base pallet data
        for group j = 1 to ProductGroupListCount
            PGroup = ProductGroupList[k]
            if (Pgroup.Class = Pclass.Class)
              if (Pallet.Cube + Pgroup.Cube <= 100)
                Pallet.Cube+= Pgroup.Cube
                assign new or reassign to current pallet
                remove current class products
```

Figure 6: Assign class to a new pallet where class cube greater than minimum class cube

6. Assign Cartons to Partial Pallets of Same Class Without Splitting Group

The procedure then continues by assigning the product cartons to partial pallets of the same product class, where the complete article can be assigned to that pallet. The pseudo code is given in Figure 7.

```
calculate group list from left over product cartons
sort group list by class
sort pallet list descending order
calculate class list from left over group
sort class list by descending cube
for class i = 1 to ProductClassListCount
  PClass = ProductClassList[i]
  for pallet j = PalletListCount
    thisPallet = PalletList[j]
    if( thisPallet.Cube < 100)
      if (thisPallet.Class == Pclass.Class)
        for group k = 1 to ProductGroupListCount
          PGroup = ProductGroupList[k]
          if (Pgroup.Class = Pclass.Class)
            if (thisPallet.Cube + Pgroup.Cube <= 100)
              thisPallet.Cube+= Pgroup.Cube
              PalletList[j] = thisPallet
                for item l = 1 to LeftOverProductListCount
                  Product = LeftOverProductList[i]
                  if (Product.Class == PGroup.Class)
                    remove current product
                    l--
```

Figure 7: Assign carton to partial pallets of same class

7. Assign Cartons to Empty Pallets without Splitting Products

This step assigns cartons to empty pallets of any class, where the pallet are not full, in ordered of descending product cube percentage. The products would be assigned only if the pallet has space for the full remaining product cube percentage. If there was only one product left then assign it to any of the pallet that is less than 120 %. The underlying pseudo code is shown in Figure 8.

```
sort group list by class()
calculate class list from left over group
sort class list by descending cube
for class i = 1 to ProductClassListCount
    PClass = ProductClassList[i]
    create base pallet data
    for item j = 1 to LeftOverProductListCount
        Product = LeftOverProductList[j]
          if (Pallet.Cube + Product.Cube <= 100)
            Pallet.Cube+= Product.Cube
            assign new or reassign to current pallet
            RemoveCurrentProduct(j)
            j--
          else
            curCube = Pallet.Cube + Product.Cube
            if (ProductListCount == 1) AND (curCube <= 120)
              assign current pallet
              remove current product(j)
              j--
```

Figure 8: Assign carton to empty pallets without splitting

8. Assign Any Products Carton by all Quantity to Any Pallets

This section attempts to assign the remaining product cartons, sorted in descending cube size, onto partially filled pallets. The partially filled pallets are sorted by their cube percent in descending order. The products are assigned only if the pallet has space for all the remaining cartons of the same product. The pseudo code is given in Figure 9 below.

```
sort pallet list in ascending order
sort product list by descending cube
for pallet i = PalletListCount
   thisPallet = PalletList[i]
   for item j = 1 to LeftOverProductListCount
     if (thisPallet.Cube == 100)
        break
     Product = LeftOverProductList[j];
     If (thisPallet.Cube + Product.Cube <= 100)
       thisPallet.Cube =  thisPallet.Cube + Product.Cube
       PalletList[i] = thisPallet
       remove product list(j)
       j--
```

Figure 9: Assign any product to partial pallets

9. Assign One Product to the Last Pallet if Pallet Cube < 120 %

This step provide the last clean up, so that at this stage if there is only one carton left in the current load, then assign it to the last pallet that results in a total cube value of lesser than or equals to 120 %. If there are more than one box left after this then report un-allocate boxes to pallet error on this load, so that it could be reinvestigate and reassign outside the algorithm. The listing pseudo code is shown in Figure 10.

```
sort pallet list ascending order
sort product list by descending cube
thisPallet = PalletList[PalletListCount - 1]
for item i = 1 to LeftOverProductListCount
   Product = LeftOverProductList[j]
   if (thisPallet.Cube +  Product.Cube <= 120)
      thisPallet.Cube+= Product.Cube
      PalletList[PalletListCount - 1] = thisPallet
      remove product list(i)
      i--
       if (ProductListCount > 0)
       {
         report error
         break
       }
```

Figure 10: Assign product to last pallet

IV. Experiment and Result

The cartons to pallet allocation algorithm was implemented using the C# programming language. The experiments were performed on the developing platform that has a core CPU clock speed at 2.6 GHz running Windows XP Service Pack 2.

a) Experiment 1

In this experiment only one load input order file was

considered. The input order file is given in Table 4.

Table 4. Experiment 1: Input order file for selected load

Product Number	Order Quantity	Order Number
1822	28	1233
3680	12	1233
3685	5	1233
3945	2	1233
5661	40	1233
5702	1	1233
5705	15	1233
5732	21	1233
5734	42	1234
7342	5	1234
7601	42	1234
7602	80	1234
8668	110	1234

A feasible loading and pallet utilization for a selected load, obtained within 50 milliseconds, is presented in Table 5.

Table 5. Pallet loading results for selected load

On Pallet	This Pallet Class	Order Number	Product Number	Qty On Pallet	Pallet Utilization	Program Section
1	A	1234	7602	78	100%	1
2	B	1234	8668	108	100%	1
3	A	1234	7601	42		3
3	A	1233	5705	15		3
3	A	1234	7342	5	83.33%	3
3	A	1234	7602	2		3
3	A	1233	5702	1		3
4	C	1234	5734	42	87.33%	3
4	C	1233	3945	2		6
5	C	1233	1822	28	98%	5
5	C	1233	5732	21		5
6	A	1233	5661	40		5
6	A	1233	3680	12	71.38%	8
6	A	1233	3685	5		8
6	A	1234	8668	2		8

From Table 5, the important columns to be observed are the "On Pallet", "Qty On Pallet", Pallet Utilization" and "Program Section" respectively. On each row in this table it tell us that, on the current pallet, how many cartons was added, what was the total utilized percentage on the pallet and what algorithm section was run respectively. The result obtained, listed in Table 5, shows that the developed algorithm has split the order cartons of a particular product and rearranged them feasibly onto pallets ready for loading. It can be seen from the pallet utilization results that the developed algorithm, although not optimal, generates a promising feasible result within 50 milliseconds of the running time.

b) Experiment 2

In this experiment the affect of the number of input load increases is considered. The purpose was to test the reliability of the algorithm in solving these problems. Due to the size of the test dataset only computational results are provided. The experimental result for the increase in the number of orders and load numbers is given in Table 6. The plot of number of orders versus the algorithm solving time was plotted and shown in Figure 11. The number of pallet created for a given input cartons was plotted and has been given in Figure 12.

Table 6. Load numbers for increased carton orders

No. Of Order	No. Of Cartons	No. Of load	Pallet Required	Time (sec)	Manual Min Pallet Required	Manual Max Pallet Required
13	403	1	6	0.05	6	6
4000	31972	154	607	0.52	505	611
8000	61808	220	1109	1.09	969	1114
12000	92391	283	1611	1.82	1431	1617
16000	128964	500	2271	3.40	1965	2286
20000	159316	630	2776	5.15	2391	2793
24000	191361	736	3323	6.84	2869	3341
28000	222187	822	3862	8.79	3355	3881

In Table 6, the experiment results columns are the "Pallet Required" and "Time". The "Pallet Required" column shows the algorithm output *Total Number of Pallets Required.* The "Manual Min Pallet Required" column shows the total estimated minimum number of pallet required, when cartons were filled on to any pallets regardless of their sizes and types. The "Manual Max Pallet Required" was the manually calculated feasible pallet loading results. By comparing the "Pallet Required" column to the "Manual Max Pallet Required" column, it could be seen that as the number of cartons increases, the pallet differences between these two columns increases. The pallet differences when the number of cartons equals to 222187 is 19 pallets comparing a feasible manually calculated maximum solution result to the algorithm solution, which shows that the developed algorithm provides promising results that was much better than the manually calculated results. As it could also be seen from Table 6, that as the number of input cartons increases, the solving time only increases slightly.

From Figure 11, it can be seen that the solving time grows rapidly after the 12000 orders as it started turning into an exponential function. This is caused by a greater increase in the number of cartons between the 12000 orders and 16000 orders, as seen in Table 6 previously. Despite worse case exponential increases in the solving time, the solving time was less than 9 seconds for a 28000 lines of orders with a 222187 cartons, which is overall a very promising result. Although the solving time issue might takes it toll, this algorithm successfully handled actual factory orders data, where these sets of orders data are unlikely to be greater than 50000 lines. As the solving time were in the orders of number of seconds, this shows that the developed algorithm is reliable, hence solving time is not an issue to worry about.

In Figure 12, it was noticed that the computed number of pallets required grows linearly with the number of input cartons. This shows that, no matter how large the number of input cartons is, the number of output pallets change linearly with it. This also shows that the algorithm and the developed rule provide a linear packing behavior of cartons onto pallets.

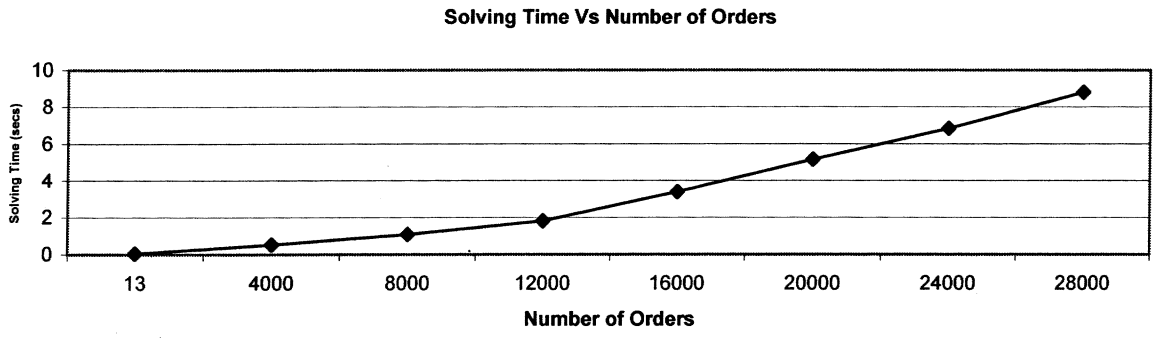

Figure 11: Solving time over input number of orders

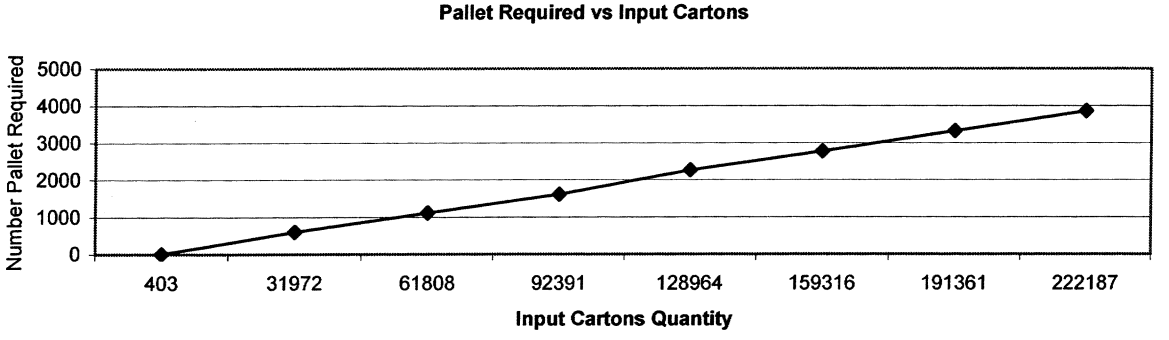

Figure 12: Number of pallet loaded verses input cartons

V. CONCLUSION

This paper describes an algorithm to efficiently solve the cartons to pallet allocation and loading problem. The carton to pallet software developed using this algorithm is currently used by the drink manufacturer involved in the study.

In this paper, through experiments, it was found that the developed algorithm efficiently solves carton to pallet loading problems very well and in a very manageable short computing time. A problem with 28000 orders, and 222187 cartons, was solved in 9 seconds and the load was allocated to 3862 pallets.

The developed algorithm finds a feasible solution, but does not seek to further improve on this. Future enhancement could be made to improve the algorithm by adding a branch and bound algorithm around Steps 4 to 7, to optimise the loading sequence and thus overall pallet utilization.

REFERENCES

[1] E. G. Coffman Jr., M. R. Garey, and D. S. Johnson, "Bin Packing Approximation Algorithms: A Survey" in *Approximation Algorithms for NP-Hard Problems*, D. Hochbaum (ed.), PWS Publishing Co., Boston MA. 1996.

[2] E. R. Korf, "Optimal Rectangle Packing: New Results", in Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004), pp. 142-149.

[3] M. Gardner, "Mathematical games: The problem of Mrs. Perkin's quilt and answers to last month's puzzles". *Scientific American* 215(3):264–272. 1966.

[4] P. Manyem, "Bin packing and covering with longest items at the bottom: online version". Australian Mathematical Society. 2002, http://anziamj.austms.org.au/V43/E044/Manyem.pdf, 2005.

[5] A. M. Verweij, "Multiple Destination Bin Packing". Institute of Information & Computing Sciences technical Reports. http://www.cs.uu.nl/research/techreps/aut/bram.html. 03-2005.

[6] J. Terno, G. Scheithauer, U. Sommerwei and J. Riehme. "An Efficient Approach for the Multi-Pallet Loading Problem". in *European Journal of Operational Research*, 2000, pp. 372-381.

[7] G. Scheithauer, U. SommerweiB. "4-Block Heuristic for the Rectangle Packing Problem". in *European Journal of Operational Research*, 1996, pp. 509-526.

[8] G. Scheithauer, U. SommerweiB. "A new heuristic for the Pallet Loading Problem". in *Operation Research Proceeding*, 1995, Springer-Verlag Berlin Heidelberg (1996), pp. 84-89.

[9] J. NeliBen. "New Approaches to the Pallet Loading Problem". Lehrstuhl für Angewandte Mathematik, insbesondere Informatik RWTH, Aachen, Germany. 1993.