Multiple Container Packing

John A George
Department of Management
University of Canterbury
New Zealand
j.george@mang.canterbury.ac.nz

Abstract

While the problem of packing single containers and pallets has been thoroughly investigated in theory and practice very little attention has been given to the efficient packing of multiple container loads. In practice such loads are normally packed using a single container method in a "greedy" fashion. This paper introduces the issues involved. It lays out three different strategies for solving the problem: (1) improved heuristic rules in the single container heuristics, (2) pre-allocation of items to the containers, and (3) solution using a models which packs all containers simultaneously. The principle models for strategy (3) are pattern selection IP models. The paper reports on experimental results for the problem of packing pipes in shipping containers using randomly generated data.

1 Introduction

Packing problem are significant both for their theoretical and practical importance. Dowsland and Dowsland (1992) gives a recent literature review which deals with algorithms covering bin packing, pallet loading, and two and three dimensional container packing. Such algorithms are almost exclusively concerned with the optimal packing of a single container or pallet load. In reality many loading problems are multiple container problems. At times the container represents a standard shipment of items and in some cases a single container solution will provide a once-for-all pattern. The multiple container problem is important when the packing of a container is affected by the packing of the previous ones. This will only occur where there are several item sizes to be loaded.

In practice, multiple container loads are solved with a single container algorithm used in a "greedy" fashion by choosing the best container load from the available load. This gives the locally best solution at each container but may not give the best overall solution. The algorithm has available a wide selection of items early in the process so it may perform extremely well for the early containers. It may, however, perform badly overall if the best combinations in the early containers leave items which pack together poorly in later ones. A slightly worse packing initially may give a better solution over all the containers. A poor overall solution is likely when some of the items are "awkward" to pack, e.g., when they are large compared to the size of the container or when they combine badly other items in the load. Thus an awkward item must be loaded using efficient combinations with other items.

The equivalent of the multiple container problem is not new as a cutting stock problem
it is normally called the "assortment problem". Gilmore and Gomory (1961) and (1962) and more recently Haessler and Sweeney (1991) give comprehensive statement of that problem. While there is a close similarity between the two problems they are not equivalent and many of the conclusions and techniques for the cutting stock problem do not translate directly to the packing problem. Because of the greater complexity of the latter the methodology for the assortment problem has not been applied to packing. To our knowledge the multiple container problem has not been investigated systematically for the packing problem. Prosser (1988) looks at multiple pallet packing using a genetic algorithm for small problems. Fraser and George (1993) outline briefly the problem within the context of the packing of paper products.

In this paper we attempt to provide systemic thinking to the problem and to suggest theoretical and practical approaches to deal with it. Some of the methods are tested for a particular example. We outline first an integer programming formulation. We outline three broad solution strategies. A particular application where pipes are packed into containers is outlined is used to test and compare the methods.

2. Mathematical Formulation

The problem can be formulated mathematically in terms of selecting container configurations ("pattern selection") akin to the cutting stock model of Gilmore and Gomory (1961), and Haessler and Sweeney (1991). It assumes that a set of all feasible packing patterns for the containers can be generated and, initially, this combination of patterns is able to stow the entire load. To guarantee globally optimality and feasibility all possible full and partial packing patterns must be included in the set of available patterns.

Let the set $S$ be the set of different types of items to be packed. Item $s \in S$ has $B_s$ units to be packed. Let $K$ be the set of packing patterns available to pack a container. Packing pattern $k \in K$ is defined as a packing configuration which contains $A_{sk}$ items for each $s \in S$. Each packing pattern is "feasible", i.e., it can be stowed by a known packing algorithm. Let $C_k$ be the proportion of the container filled by the items in pattern $k$. Let $p_k$ be the integer valued decision variable of the number of containers packed using pattern $k$. The objective criterion is to minimize the total number of containers required. Using the notation we have defined the formulation of the pattern selection problem is given as:

$$(1) \quad \text{Minimize } z = \sum_{k \in K} p_k$$

subject to

$$\sum_{k \in K} A_{sk} p_k = B_s \quad \forall \ s \in S$$

$$(3) \quad p_k \geq 0 \text{ and integer}$$

Constraints (2) require that the load must be stowed exactly using the container patterns selected. Usually, in practice, these constraints are too stringent because the patterns are only specified for full container loads because it is very difficult to specify all the associated partial container loads, and only a small subset of all of the possible packing patterns are used. So constraints (2a) replace (2) to ensure that a feasible solution exists. For convenience we will call the model with equations (1), (2a) and (3) BASIC_IP.
Solutions to BASIC IP can still cause implementation difficulties. If the constraint for item $s$ is slack then either too many units of $s$ must be shipped or some of the locations of units of item $s$ are not filled. The first assumes that the customer will accept excess units. The second assumes that a feasible packing schemes exist with the excess units of $s$ absent. Clearly, a customer may reject an excess load but, less obviously, even though an algorithm can pack a given container load it may not be able to pack a subset of that load. This does not occur for the cutting stock problem.

To overcome this difficulty we define a new model, EXTENDED IP, to give a solution which efficiently packs no more than the number of units of each item in the load. Any units unpacked by the solution are subsequently packed with the single container algorithm. This model is shown in the Appendix.

3 Alternative Solution Strategies

From existing practice, the current literature, and the ideas introduced in this paper, we identify three basic strategies for solving the multiple container problem.

Strategy 1: Sequential packing with a single container algorithm using a greedy approach.

This is the normal method for solving the problem. However, from a methodological viewpoint the approach has two important variations depending on the focus of the packing method: (a) a standard single container method, and (b) an algorithm which takes explicit account of the load being of a multiple container nature (see Fraser and George (1994)).

Commercial single container algorithms have many heuristic embedded in them and produce many possible solutions before selecting the "best". Thus it may not be difficult for an algorithm which has the appearances of being a type (a) method to be adapted to be a type (b). At the least users can test the various heuristic strategies already in the algorithm to see which perform better in the multiple container environment.

Strategy 2: Pre-allocation of items to containers so that each container is packed with those items allocated to it.

Here recognition is given to the difficulties of allocating the load across the containers. Rather than allow the greedy algorithm to choose the load, some a priori rules of load allocation are used. The rules for allocation can vary according to the problem. They should take into account the experience of the user in combining items together for efficient packing. However, they are also able to account for non-quantitative and intangible aspects of the problem.

Strategy 3. Simultaneous packing using a multiple container model.

There is very little in the packing literature on this approach. We have already introduced IP models which are inherently simultaneous. A further method which we foreshadow but do not been explored is a meta-heuristic which looks at the load as a whole. Such a heuristic may take the form of searching over the space of solutions to the problem. In this context a "solution" is defined in terms of the multiple container load not a single container load. To be effective the multiple container solutions must be generated
very quickly, even if some of them may not be very good. Then a search strategy across these solutions could be a viable method.

In practice, none of these approaches is expected to be entirely dominant. Each of the various strategic approaches to perform relatively better and be more appropriate in different circumstances.

4 Application to multiple container pipe packing

The application in this section is the second part of the problem previously described in detail in George, George and Lamar (1993). The problem involves loading different diameter plastic pipes in shipping containers. Each pipe is the full length of the container and pipes may be nested in other pipes. It can be divided into the three main parts: i) How to nest pipes optimally, ii) How to best pack pipes into one container, and iii) How to allocate pipes to various containers in order to minimise the number of containers.

We assume that parts (i) and (ii) have been successfully solved. Part (ii) is achieved by regarding the packing of the pipes as placing their circular cross-section against the vertical face of the container. So it becomes the 2-dimensional problem of packing circles into a rectangle. George et al devised a number of algorithms for solving the single container problem. We used their most effective algorithm, RANDOM.

We solved the IP models BASIC_IP and EXTENDED_IP using AMPL/ CPLEX on a 486DX2 50 Mhz computer. 100 randomly generated problems were solved for the same 6 item sizes. A set of 81 patterns were generated using a stratified generating technique.

Table 1: Usage of patterns in the IP optimal solutions for 100 problems

<table>
<thead>
<tr>
<th>Patterns in order of efficiency</th>
<th>Number of patterns</th>
<th>Number of containers packed by patterns</th>
<th>Number of containers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1 - 9</td>
</tr>
<tr>
<td>1-2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 - 20</td>
<td></td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>21 - 40</td>
<td></td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>41 - 60</td>
<td></td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>61 - 81</td>
<td></td>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>11</td>
<td>44</td>
</tr>
<tr>
<td>Number of Containers</td>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

From Table 1 we draw two main conclusions.

i) The high efficiency patterns are likely to dominate the packing configurations actually used, e.g., 37% of all containers used the two most efficient patterns.

ii) Many patterns in the set are of little or no practical value, e.g., 55 of the patterns were used for less than 10 containers each.
Table 2: Deviation of major methods against best solution found.

<table>
<thead>
<tr>
<th>Deviation (Containers)</th>
<th>BASIC_IP</th>
<th>EXTENDED_IP</th>
<th>GREEDY</th>
<th>PREALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>84</td>
<td>80</td>
<td>37</td>
<td>27</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>20</td>
<td>58</td>
<td>53</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Total increase</td>
<td>16</td>
<td>20</td>
<td>68</td>
<td>97</td>
</tr>
</tbody>
</table>

Table 2 compares selected methods, BASIC_IP, EXTENDED_IP, GREEDY (Strategy 1 using RANDOM) and PREALL (a pre-allocation method). Each method is compared to the best solution found by any of these and several other methods. It gives the deviations in the number of containers from the best solution for each method over the 100 problems. The IP methods were superior to either of the other strategies. GREEDY provided the best solution less than 40% of the time. PREALL performed worst by using 97 more containers than the best solutions. More detailed analysis showed that the poor performance of these two methods occurred for large loads. Naturally we are loathe to suggest that these results can be generalised to other applications of the models. However, since they can be reasoned on the basis of the model assumptions we will not be surprised if further work substantiates our conclusions.

References


Appendix

For EXTENDED_IP several new parameters are needed. Let $LPopt$ be the optimal solution to the continuous relaxation of BASIC_IP. Let $W_s$ be the proportion of the container filled by a unit of item $s \in S$. Let $F$ be the proportional fit of an "average" full container and let $M$ be a positive constant. We define new variables $e_s$ as the unpacked units of item $s$. Two further binary variables $\delta_1$ and $\delta_2$ are defined as:

\[
\delta_1 = \begin{cases} 
1 & \text{if an additional full container above } [LPopt] \text{ is used}, \\
0 & \text{otherwise.}
\end{cases}
\]

\[
\delta_2 = \begin{cases} 
0 & \text{if there are no surplus units of item } s \text{ for all } s \in S, \\
1 & \text{otherwise.}
\end{cases}
\]

Having solved BASIC_IP to find $LPopt$ we know that at least $[LPopt]$ containers are required. The new model then packs at most $[LPopt]+1$ containers using patterns from the set $K$. Any units excess unpacked units are packed by the single container algorithm. In defining the excess we compare the $([LPopt]+1)$th container with the excess units after only $[LPopt]$ containers. To do this we assume we are indifferent between a full container and unpacked units filling $100F\%$ of the container, i.e., the $([LPopt]+1)$th container is defined as being $100F\%$ full. We also assume that having up to $[LPopt]+1$ containers exactly full is preferred to a solution having any excess units.

(4) Maximize $z = y$

subject to

(5) \[ \sum_{k \in K} p_k \leq [LPopt] + \delta_1 \]

(6) \[ \sum_{k \in K} A_{sk} p_k + e_s = B_s \quad \forall s \in S \]

(7) \[ \sum_{s \in S} W_s e_s + F \delta_1 \leq y + M (1 - \delta_2) \]

(8) \[ y \leq M \delta_2 \]

(9) \[ \sum_{s \in S} W_s e_s \leq M \delta_2 \]

(10) \[ e_s, y \geq 0 \]

(11) \[ p_k \geq 0 \text{ and integer} \]

(12) \[ \delta_i = [0, 1] \quad i = 1, 2. \]