The Potted Plant Packing Problem

René Schumann, Jan Behrens and Johannes Siemer

OFFIS, Escherweg 2 26121 Oldenburg Germany
rene.schumann|jan.behrens|johannes.siemer@offis.de

Abstract. In this article we describe a practical planning problem, the potted plant packing problem, and our first approaches to solve this problem. The transportation of plants is comparably expensive as they require careful treatment due to there sensitivity. For standardised transport, potted plants are loaded on transport trollies. The cost of transportation depends on the number of these trollies. In order to minimise transportation costs, effective packing of trollies is necessary. This packing problem is herein presented and a first proposal for a solution, based on problem decomposition and multi-dimension bin packing, is given.

1 Introduction

The packing problem presented in this article is a problem a customer of our research group asked us to solve. The transportation of potted plants is comparably expensive, because plants are sensitive products and their size to value ratio can be exceedingly low. The standardised mode of transport for potted plants is on transport trollies. Such a trolley, a so called 'CC-trolley', is shown in figure 1.

Fig. 1. A 'CC-trolley' for plants
Picture taken from [1].

As one can see in the picture all plants are placed on layers. These layers can be mounted into a trolley. Depending on the height of the plants on each layer a trolley can carry a varying number of layers.

The cost of transportation is mainly a function of the number of trollies used for the transport of one particular order. Effective trolley loading therefore is a key requirement for cost reduction. The resulting planning problem is a variation of a 3-D bin packing problem, with a number of additional constraints. We will present this problem in the following section. Thereafter we illustrate a possible solution through problem decomposition. We conclude with a short summary and an outlook on further work in the last section.

2 The packing problem

2.1 Detailed description of the packing problem

This section offers a detailed description of the potted plant packing problem. At first we describe the business process that leads to the transportation of an order:
Initially an order is given. Such an order consists of any number of order items. Each order item relates to a specific article and contains the numbers ordered. Each article, e.g. a potted plant, has various attributes like height, width, depth, weight and type of packing-unit (e.g. a flowerpot). There are of course further attributes, price for example, they are however neglected here as they are not related to the packing problem. The computation of the relevant attributes, especially the dimensions, is not trivial, as the dimensions of a plant are subject to change in time. The available article data is based on the estimated size at a point in time some weeks into the plant season. The actual size at any given time varies from this and has to be estimated with the help of further parameters like seasonal information.
Each plant is packed into a packing-unit, a flowerpot or a tray for example. Consequently a packing-unit can hold one or more plants of one article. The form of a packing-unit can differ from rectangular to circular. Apart from its form, a packing unit also has further attributes relevant to the packing problem, e.g. height, width, depth and weight. In theory, packing-units can differ for one article, depending on the plant supplier for subcontracted part orders. We assume that all plants of an order item were delivered by one supplier and thus have the same type of packing-unit. Such a simplification can be made without restricting the general approach as we could introduce a middle tier that would map packing articles - with only one packing unit associated - to actual articles later.
In practice sometimes subcontracted part orders are supplied by external nurseries. These might arrive pre-packed on cc-trollies at the central packing facility. Such trollies are typically left as is, because repackaging them would usually lead to a substantial additional effort. Except that, if they have additional space for entire layers available, this might be used for other order items of the same order.

After some pre-processing computation the relevant information for packaging of potted plants can be extracted from an order item. Recapitulating this information comprises of:

– An **order** consisting of a number of order items.
– Each **order item** consists of a number of identical packing-units.
– **Packing-units**: For each packing-unit, information about its dimension, weight and form are given.
– **Pre-packed trollies** may combine some order items on cc-trollies.

Our main objective is the computation of a valid packing plan, based on the order data, as well as providing detailed packing-instructions for the staff. Any packing-instruction, and therefore also every packing-plan, consists of a number of trolley-packing-instructions, one for each cc-trolley in the order. Such a trolley-packing-instruction itself consists of a number of layer packing-instructions. They define the exact position of the layer (mounting height) along with detailed packing instructions for all packing-units on the layer. Each single layer consists of a set of packing-units along with the exact position of every packing-unit on the layer. The position includes information about each packing-units height, because certain articles allow for stacking of further packing-units. This is shown in figure 2 where a second layer of flowerpots is stacked upon a first layer. The generation of a valid, stable and realis-

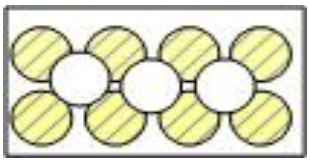

Fig. 2. A simple example of stacked flowerpots

able packing-instruction requires the observance of a number of constraints. Usually a differentiation is made between soft constraints (e.g. keep order items together) and hard constraints like the fixed size of layers, or the fact that an item can not span over more than one layer.
When plants are stacked, stability is an important aspect. Another facet of the stack packaging is that it has to be ensured that those items at the bottom are not damaged by the stacked items. This constraint can be formulated as an additional artificial attribute of an article. Each article has an inner region which is of bounds for stacking. If this region has the same size as the packing-unit itself, the packing-unit is called non stackable. This concept is displayed in figure 3. The overall height of a loaded trolley is of course limited too, because the trollies are transported on truck. Thus the available height varies from approx. 240 to 260 cm. For stability reasons the position of a layer in a cc-trolley must correspond to the weight of that layer, e.g. a heavy layer should be placed lower within the trolley. Furthermore each layer as well as the cc-trolley itself has a maximum payload.
The definition of rules leading to valid stacking strategies and packing plans and so

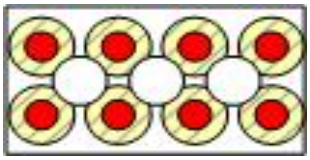

Fig. 3. stacked flowerpots with non stackable regions (displayed in dark color)

ensuring the adherence of all constraints, is not a trivial task.
As a further - soft - constraint it is desirable to keep order items together, therefore one order item should be loaded onto one layer - if fit - or at least contiguous onto subsequent layers and trollies. This is desirable because usually all packing-units of an order item will be provided together at the loading zone and the number of cc-trollies concurrently in loading is limited. The overall goal is to provide said packing-instruction for any given order, with the minimisation of needed cc-trollies - and therefore costs of transport - as the objective function.

2.2 The formalised potted-plant packing problem

One general approach when describing planning problems is to use a tuple of parameters, which are capable of describing the entire problem. Scheduling problems for example are described in such a way by Keng et al. [2] or Sauer [3]. The potted-plant packing problem can be formalised in this way, too. Its tuple contains the following parameters.

- **resources (trollies:)** Because one could argue that this problem is a complex bin-packing problem with a virtually unlimited number of trollies on which to pack the entire order, it may be doubtful why resources should be modelled explicitly. There are however two main reasons for doing so:
  - It is possible to use add-on modules with the trollies to increase their height. The number of available add-on modules however does not necessarily correspond to the number of available trollies
  - Any order might include a number of pre-packed cc-trollies which may not be loaded entirely. Thus a planning might start with any number of pre-packed trollies with limited capacity left.
- **objects (plants:)** These are the objects the algorithm is supposed to pack. We actually do not focus on the plants themselves, but on an artificial entity called packing-unit instead. We use this simplification because a typical instance of the potted-plant packing problem can have some 50.000 different plants. In using an abstraction such as the packing-units instead, we can greatly reduce the number of different entities the algorithm has to respect.
- **order (order items:)** An order comprises of a set of order items, each declaring a number of specific packing-units which have to be packed on trollies. Additionally the order details the customer as well as the delivery and packing time. The

later is relevant in computing the height of the different plants, since plants grow over time. An order furthermore specifies the maximum height any cc-trollies is allowed to have.

- **hard-constraints:** This list is imperfect so far.
  - stability of packed plants
  - safety-aspects for stacking plants, ensuring no plant is damaged by stacking
  - any single packing-unit has to be stored in its entirety on one layer
  - a layer has to be mounted correctly into one trolley
  - the overall trolley height is lesser than or equal to the allowed maximum
  - any single layer can be mounted into one trolley only
  - a packing-unit can not be stored bottom up
- **soft constraints**: The packing-units of an order item are placed contiguously on layers.
- **objective function:** The objective function is to minimise the number of trollies needed for packing all potted plants of an order onto trollies.

3 Proposed solution

In this section we present our concept of an algorithm capable of solving the described potted plant packing problem. As this paper addresses work in progress, this algorithm has not been implemented so far and can therefore not be presented with actual test results.
The original potted-plant packing problem, which is a 3-D bin packing problem, is decomposed in a 2-D bin packing and a 1-D bin packing problem. We are able to do so, because one can first compute the layers, which is a 2-D bin packing problem with a modified objective function. After completing this first step the layers have to be fitted into trollies, this corresponds to a 1-D bin packing problem. The following section describes the algorithm in more detail.

Since there is a huge number of different plant species, which differ only in aspects irrelevant for packing, the first step of our solution is the forming of plant groups. All plants classified in one group will have similar packing parameters. There could be a number of articles only differing in colour for instance, as colour is not relevant to packing it can be neglected and all articles differing in this respect only can be classified into the same packing group. Classification will go beyond the mere exclusion of irrelevant attributes however, it will also be necessary to look for similarities between different plants. Thus our classification algorithm will be looking at these parameters: width, height, depth, weight, type of packaging-unit, flexibility of article (e.g. can it be bend) and stack ability. Furthermore it must be possible to classify one article as belonging to different groups, with the assignment to a group being limited to a certain time frame (e.g. June to September). For each packing-unit the packing groups have to be computed from the existing data base, containing of the

– length,
– depth,
– height and
– weight

of the packing-units. Afore mentioned additional attributes (e.g. stack ability) will be filled at request, as the data does not allow for an automatic computation of these.
Because of the huge number of possible solutions and to enable the integration of existing knowledge on well packed layers at trollies, the use of packing patterns seems profiting. A packing pattern encodes a valid packing of packing-units of certain groups on a single layer. A packing pattern can contain one or more groups of packing-units. Figure 2 shows an example of a packing-pattern, containing packing instructions on one kind of packing group per layer only. In figure 4 another example with different kinds of packing groups is shown. A preference coefficient can be as-

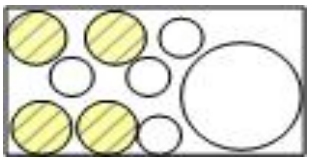

Fig. 4. packing-pattern with different packing groups

signed to any particular packing pattern to distinguish favourable and less favourable patterns from one another. Such a preference coefficient can for example express the degree of space utilisation per layer. A variety of packing-patterns is required for the proposed system to work. They can either be derived from expert knowledge or be computed offline, thus reducing online computing time. All packing patterns will be stored in a knowledge base. To compute such packing-patterns, a 2-D packing problem has to be solved. Therefore a heuristic like the G4 heuristic [4] can be used.
Besides the G4 heuristic, a first approach can be made using a much simpler strategy, already in use today by the packing staff. The basic idea of this strategy is to use existing knowledge on the number of similar plants that can be fitted on one layer. If, for instance, 21 plants of one type fit on a single layer then one equals $\frac{1}{21}$ of a layer. Accordingly computing the packing instruction of one layer is simple fractional arithmetic. This simple heuristic has the advantage of being faster and can therefore be used for online-computation of packing instructions. The results produced however deteriorate with increasing number of different plants or smaller number of similar plants. For instance, one can think of a packing-pattern for three large plants, as shown in figure 5. Obviously only three plants of this particular size fit on one layer - thus one plants size would be $\frac{1}{3}$ of a layer. It is also obvious though, that enough space for the packing of smaller plants remains on that layer (see figure 6).

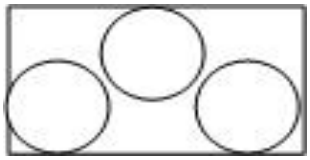

Fig. 5. packing-pattern with three large plants)

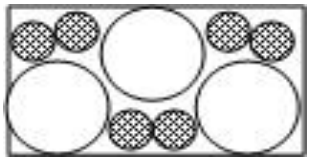

Fig. 6. packing-pattern with three large and six small plants

The first step of our actual runtime packing algorithm is finding those order items being delivered on pre-packed trollies and computing the storage capacity left available on them. The pre-packed order items will be neglected for the remainder of the planning process.

The second step of the algorithm is grouping all packing-units as per their classification, their packing-group. For each packing-group a queue is used, holding all corresponding packing-units. The order of packing-units in the queue is analogue to the order of items in the initial order.

To compute an initial packing plan, a simple heuristic is used. The queues are sorted by their group height in descending order. Thus the algorithm starts with the queue containing the tallest plants. The algorithm then accesses the knowledge base and tries to find packing-patterns for the packing-units of the group stored in the active queue. If no matching packing-pattern can be found, or if there are not enough packing-units to fill an entire layer the packing-units of the next queue are added to the pool and the procedure is repeated. If there is more then one applicable packing-pattern, the one with the highest preference coefficient is chosen. The algorithm will try to assign all items to complete packing-patterns without a rest of unassigned items. It is however doubtful, that this will be possible, we anticipate to usually have a remainder of items that is unassigned after the pattern filling procedure. Said remainder will then be packed on layers using the same algorithm used for the offline computation of the packing-patterns or an appropriate heuristic. While this second step will ask for much higher computation times, we still expect to see a low overall runtime as we anticipate a pattern filling rate of 80% at the least.

The packing pattern knowledge-base basically improves the performance of the planning tool through the reuse of sub-solutions. Future steps will see the adoption of self-learning strategies such as the extraction of patterns from commonly used online computed layers. Such patterns would then be added to the knowledge base.

Furthermore the preference coefficient could be automatically adjusted by the system for often used patterns, or those not used at all. The knowledge-base will start with patterns based on expert knowledge already existing and a set of computed patterns to broaden the base. It is expected to grow trough the aforementioned steps and the possible addition of further manually compiled patterns. The knowledge-base enables the planning system to use advantages learning systems offer, to improve computed packing instructions.

The algorithm will result in an initial plan, this initial plan than serves as the starting point for further improvement strategies. Modifications can be made for example by moving a group of packing-units on the layers. Another modification may be the exchange of packing-patterns used. This may improve the density of packed layers and thus after some modification steps results in reducing the number of needed layers. The objective function for the improvement strategy is not the reduction of layers, like it would be in classic 2-D bin packing problems. Since a minimal number of layers does not guarantee a minimal number of cc-trollies. Assume the plants are placed on n layers. A solution leading to a close to minimal number of cc-trollies then can be computed using the following objective function for the improvement strategy $h(\mathcal{E})$:

$$h(\mathcal{E}) = min \sum_{1 \leq i \leq n} h(E_i)$$

$$h(E_i) = max_{1 \leq j \leq max\_count\_plantson\_layer} height(p_j)$$

This objective function leads to a minimal overall height of an order. The overall height of an order $h(\mathcal{E})$ influences the number of needed cc-trollies, since every trolley can be loaded only up to an upper bound.
The final step of the packing algorithm is the optimised distribution of the layers onto the trollies. This problem is equivalent to a 1-D bin packing problem. The objective is once again the minimisation of the number of cc-trollies needed to transport all layers. A point that differs from classic bin packing is the fact that available resources (trollies) may differ. There may be some trollies containing pre-packed layers and thus limited free capacity. There may also be a number of trollies using add-on modules for additional capacity. Finally there is a virtual unlimited number of normal trollies. Other aspects of the 1-D trolley packing problem, like the maximum payload restriction are alike for all trollies and can easily be computed by simple addition of the weights per layer. The layers each have a known height, and so correspond to the heights that have to be distributed to the bins. We are convinced that an existing algorithm solving the common bin packing problem can be adopted to solve our modified problem. One possible adoption would be to group layers in such a way that their overall height is close to or equal to the maximum height allowed per trolley. The layer containing those packing-units with the greatest height is placed at the topmost valid position within the trolley. This is done in order to use as much as possible of the available loading height (e.g. on the lorry), as opposed to the maximum storage

height on the trolley itself. The remaining layers are placed in the trolley, sorted by their weight in descending order for stability reasons.

4 Conclusion and Further work

In this article we presented a real-world planning problem, the potted plant packing problem. This problem is a special instance of a 3-D bin packing problem. We described its particularities and focused on selected key aspects like stacking plants. We then presented a first formalisation of the problem. According to this formalisation a packing problem can be described by a 6-tuple consisting of resources, objects, order items, hard constraints, soft-constraints and an objective function.
Finally we presented the concepts of our proposed solution to the potted-plant packing problem. One of our main ideas is the usage of packing-patterns and a knowledge base to storing them. This allows for reuse of sub-solutions and should reduce online computation time considerably. The later is in fact a critical factor when attempting to solve a real-world problem especially as our costumer plans to integrate the packing planning tool in his software-architecture - if the quality of computed plans is satisfying.
We furthermore think that using the concept of a knowledge base offers great potential for developing the system into a self-learning system. Such self-learning, self-adapting functionality would reduce the administration effort for maintaining the knowledge base, e.g. create new packing patterns if the typical categories mix within an order changes over time, while at the same time increasing the quality of the packing plans produced.
Due to the algorithm design we decompose the 3-D bin packing problem into a 2-D bin packing problem for layer packing and a 1-D bin packing problem, for distribution of the layers into the trollies. So far we expect the implementation of the layer pa king algorithm with its mix of offline and online computation parts to be the most challenging point.
So far our project is still in a conceptual phase. The presented solution has to be implemented and has to be proven with real world data and in day to day business. We hope to have a first set of test results by October, which we then plan to present in a forthcoming version of this paper.

References

1. Foko Lübsen und Sohn Internationale Spedition: Hompage Focko Lüpsen & Sohn GmbH, http://www.luepsen.de/seite01e.htm. (2005) accessable on 02.05.05.
2. Keng, N.P., Yun, D., Rossi, M.: Interaction sensitive planning system for jop-shop scheduling. In Oliff, M.D., ed.: Expert Systems and Intelligent Manufacturing. Elsevier (1988)
3. Sauer, J.: Knowledge-based systems in scheduling. In Leondes, T.L., ed.: Knowledge-Based Systems Techniques and Applications. Academic Press, San Diego (2000) 1293–1325
4. Scheithauer, G., Terno, J.: The G4-Heuristic for the Pallet Loading Problem. Journal of the Operational Reseach Society **47** (1997) 511–522