3D Bin Packing Problem in Practice

An INFORM Project

RWTH Aachen

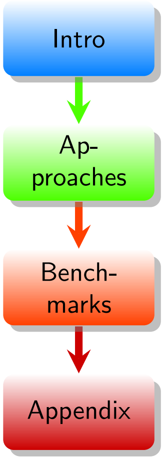
June 16, 2020

Team

Akshay Ganesh (M.Sc. DDS)
Clinton Charles (M.Sc. DDS)
Janos Piddubnij (M.Sc. Data Science)
Kalyan Keesara (M.Sc. DDS)
Luca Koczula (M.Sc. Industrial Engineering)
Moritz Dederichs (M.Sc. Computer Science)
Tobias Wagner (M.Sc. Industrial Engineering)
Yordan Manolov (M.Sc. Computer Science)

Outline

New outline suggestion



What exactly is bin packing?

How would you pack a set of articles in a given box(es) such that:

- All items are packed
- Dimensions of the box are not violated

Examples of potential Objectives:

- Minimize boxes used
- Minimize costs(packaging, delivery, etc.)
- Maximize Packing speed and safety of items

TODO When you include graphics, push the image to the git repo as well

Motivation and scope

- The global delivery market had an estimated value of 430 bn USD in 2019
- The global pallets market size was estimated to be 59,91 bn USD in 2018 with wood being the most used material type
- Improved efficiency = Better margins + Reduced Ecological Footprint
  TODO When you include graphics, push the image to the git repo as well

Optimization Task at hand

Formulate a model that optimizes 3D packaging, specifically addressing the following challenges faced by Customers :

- Packaging efficiency
- Complexity of execution
- Computation time and power
- Customization to the end user

Industry Benchmarks

How Amazon solves the problem:

- Software system displays the suitable box sizes to pick from
- Testing robots that create custom sized packaging
- Limitations :
  - Practical feasibility
  - At least a decade (or more) away before packing can be automated

Currently, only a hand full of small players offering solutions in this space

Business Use case @ INFORM

- Great addition to SyncroTess' Palette management module
- First mover advantage as a big player
- Reap the benefits of trickle-up spanning into transportation, maintenance and scheduling
- In-line with INFORM's characteristic on being environment-friendly

Split Model, Theoretical Background

- The 3D bin packing problem can be formulated as a linear program
- This approach is highly inefficient since the 3D-BPP is NP-hard
- Idea: Try to improve the runtime by splitting the problem into two separate sub-problems:
    - First: Assign items to a set of boxes according to their volume
    - Second: Place the items within the box according to some optimization criterion

Split Model, Box Assignment

- Objective: Minimize the number of boxes needed to fit all items of an order
- Items are considered to fit into a box if the combined volume of the items is less or equal to the volume of the box
- To simplify packing of items we decided to fill boxes only up to a certain volume ratio, currently 80%

Split Model, Packing Items

- Check if items can be placed within box limits
- Compute the exact placement coordinates for each item
- Minimize the sum over all z-coordinates to simulate gravity

Split Model, Restart

- Problem: Very rarely some items cannot be packed into the box that was assigned during the box assignment step
- Solution: Initiate a restart with the requirement that the number of boxes needed to pack these items is greater than the number of boxes assigned previously
  $\rightarrow$ Worst case: Each item is packed into a separate box

Split Model, Tweaks and Runtime Improvements

- Early termination is allowed if more than 5 seconds have passed since the last incumbent was found
- Multiprocessing has been implemented to allow processing of multiple orders simultaneously

Split Model, Packing Instructions

Two types of packing instructions available:

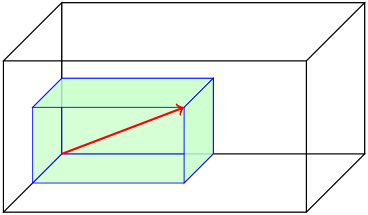
- Text instructions
- Visual instructions
- TODO: PROVIDE EXAMPLES HERE? WHEN TO SHOW THE LAYER PICTURES?

Split Model, Implementation

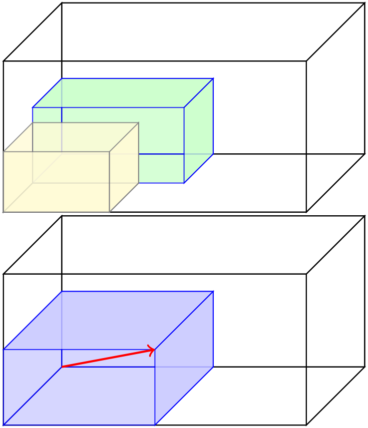
- Implemented using Python 3 and Gurobi.
- TODO: ADVANTAGES
- TODO: DISADVANTAGES

Extreme Points (EP) Heuristic, Theoretical Background

Based on [1], relies on the concept of an EP.



EP Heuristic, Updating EPs



Whenever a new item is to be placed into a box,

1. put the item next to the "walls" of the current EPs
2. if any fits, choose the smallest increase
3. new EPs are the maximum corner points of all items

EP Heuristic, Restrictions

Input: an order containing **N** items. Output: a list of **M** boxes packed with the input items.

Precondition: all items are sorted in decreasing order, see slide 34.

Postconditions:

- successor items are of sizes less or equal to predecessor ones
- all items keep their orientation after insertion into a box

Criteria:

- area
- height
- first height, then volume
- first volume, then height
- first area, then height
- first height, then area

EP Heuristic, Variants

Input: an order containing **N** items. Output: a list of **M** boxes packed with the input items.

EP-FFD: first fit.

- Put an item into a box
- If the current box is too full for the item, pick a new box
- Time complexity: $\Omega(3 \cdot M \cdot N) \subseteq O(N^2)$

EP-BFD: best fit. Before inserting the current item,

- if no previously packed box has enough room, place the item in a new box
- otherwise, pick one which maximizes f_m
- Time complexity: depends on f_m

The merit function f_m is one of: maximize free volume after item insertion, minimize packing size, minimize packing size leveled, maximize residual spaces - details on slide 32.

EP Heuristic, Implementation I

- Implemented using Python3, standard libraries for (algebraic) computation, data structures, sorting
- 100% FOSS [1], specifically:
- No 3rd party vendor lock-in
- Easy set-up
- Highly customizable (transpiling)
- Portable

Real-life results: see slide 27 . Sample output:

```
Processing order no. 0100036
For box B6811 of size B7, place the following items: [('AM', 2), ('AS2', 6)]
For box B1777 of size B7, place the following items: [('AS2', 5), ('AS1', 1)]
```

Figure: Textual output, adjacency list (conventions omitted)

EP Heuristic, Implementation II

Disadvantages: TODO put on last slide comparing all approaches

- Wasteful when items have significantly different dimensions
- No straightforward way to combine with OTS optimizers
- Depending on Python implementation: costly runtime overhead

[1] https://opensource.org/faq

GA, Theoretical Background

TODO

GA, Implementation

TODO

Real-life results: see slide TODO

PSFP, Theoretical Background

TODO

PSFP, Implementation

TODO
Real-life results: see slide TODO

Kickass benchmark, part1 I

TODO: order 1, benchmark results
example:

```
2020-07-09 10:57:20,570 Processed order no 099999
                          in 0.0008394280000629806 seconds
                          with 7 boxes and
                          avg. free volume of 14.185714285714287 percent
2020-07-09 10:57:20,570 Total time: 159.41650219099995
2020-07-09 10:57:20,570 Time elapsed:   159 seconds.

Process finished with exit code 0
```

Figure: EP heuristic using small, fixed-size boxes, on 100 000 orders

Kickass benchmark, part1 I

Approach	No. of boxes	Free volume (%)	Time (s)
Split model			
EP			
GA			
PSFP			
Split+GA			
Split+PSFP			
EP+PSFP			
GA+PSFP			

Table: Performance of all approaches on orders WHICH LOOK LIKE TODO

Kickass benchmark, part2

TODO: order 2, benchmark results

```
2020-07-09 10:57:20,567 Processed order no 099996
                     in 0.0017580510000024352 seconds
                     with 10 boxes and
                     avg. free volume of 50.49733333333334 percent
```

Figure: EP heuristic (at least first-fit) is too wasteful with many items of similar volume but different dimensions

Kickass benchmark, partN

TODO: order 3, benchmark results

Discussion I

Short on boxes? Pick . . .
Short on time? Pick . . .
Balance act? Pick . . .

- Want to play? Live demo:
- Not satisfied? Repository available at TODO URL
- Also: will be uploaded to OSDN/GitHub/Sourceforge? (without copyrighted material) at TODO URL

Questions?

EP heuristic, merit functions I

Convention: the parts after *max* are introduced by the complexity of each merit function.

1. Maximize free volume: pick the box which would be left with the most free volume after accommodating the item. Time complexity: $O(N^2 + max\{N, N\}) = O(N^2)$
2. Minimize the maximum packing size: choose the box where either the item is placed on top or, if not possible, the box with the most free surface. Time complexity: as above
3. Level the EPs: choose the box whose EPs will have the least increase in height. Time complexity: as above

EP heuristic, merit functions II

4. Maximize the utilization of the EPs Residual Space (RS). RS is roughly the same concept as EMSs in the Genetic Algorithm, namely the cubes defined by projecting the EPs to the walls of a box. Pick the box with the smallest RS still fitting the item.
   Time complexity: $O(N^2 + max\{N, N^2\}) = O(N^2)$
   Back to EP on slide 20

EP heuristic, why sort? I

Example assuming sorting by area.

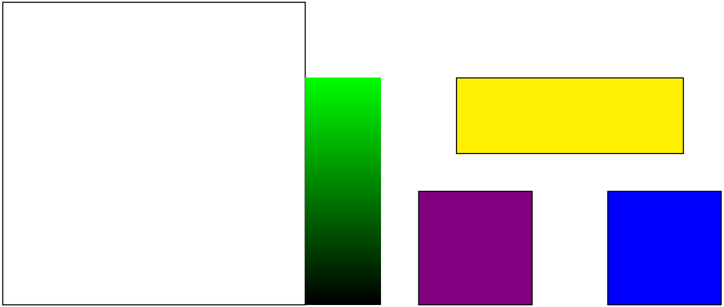
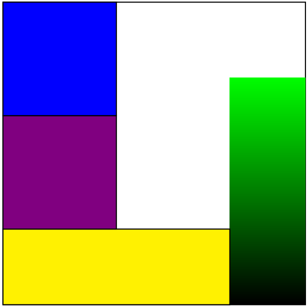


Figure: A box big enough for 4 items

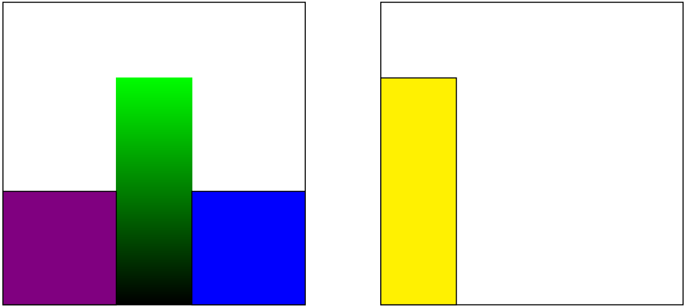
EP heuristic, why sort? II

With sorting:



EP heuristic, why sort? III

Without sorting (worst case):



Back to EP description on slide 17

Details

TODO Here come corner cases and issues. Use if questions arise Example: why does EP heuristic mandate a decreasing order of items? Why does GA use exactly N chromosomes and M mutations . . .

References

- Teodor Gabriel Crainic, Guido Perboli and Roberto Tadei (2008)
  Extreme Point-Based Heuristics for Three-Dimensional Bin Packing
  INFORMS Journal on Computing Vol. 20 pages 368–384
- Wissam F. Maarouf, Aziz M. Barbar and Machel J. Owayjan (2008)
  A New Heuristic Algorithm for the 3D Bin Packing Problem.
  In: Elleithy K. (eds) Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering. Springer, Dordrecht
- Xueping Li, Zhaoxia Zhao and Kaike Zhang (2014)
  A genetic algorithm for the three-dimensional bin packing problem with heterogeneous bins
  Proceedings of the 2014 Industrial and Systems Engineering Research Conference