Algorithm Configuration Applied to Heuristics for Three-Dimensional Knapsack Problems in Air Cargo

Marius Merschformann
Decision Support & Operations Research Lab
University of Paderborn
Warburger Str. 100, D-33098 Paderborn

Abstract

The problem of efficiently packing items into containers is of great importance in the air cargo industry. Hence, the algorithms used to solve the corresponding problem should also be efficient, including their configurations. We present an algorithm configuration scenario using a state-of-the-art algorithm from this area.

Introduction

Efficiently packing items into a set of containers is a basic process in many different applications. Especially in air cargo logistics, storage volume is expensive and the handling times at cargo hubs are short. Thus, there is demand for a decision support system capable of generating efficient so-called "build-up plans" that can be quantified and immediately executed, instead of planning the load allocation manually. As a special extension, more complex forms are handled through an approximation using "Tetris"-shapes (see Figure 1 and (Fasano 2013)). The resulting problem is called three-dimensional Tetris Multiple Heterogeneous Knapsack Problem according to (Wäscher, Haußner, and Schumann 2007). The algorithms of this paper aim to generate such plans to solve such problems in a reasonable amount of time.

The problem can be decomposed into three decisions. First, items must be allocated to available containers. Every item may be assigned to at most one suitable container. Secondly, the positions of items within containers must be determined. Finally, the orientation of the item is chosen from 24 distinctive 90° rotations for "Tetris"-shapes. These decisions are bounded by certain basic requirements like the non-overlapping of items and containers and more application specific ones like forbidden orientations and incompatibilities when handling dangerous goods. The objective of this problem is to utilize the available space as effectively as possible, similar to the classic knapsack problem.

We present an algorithm configuration scenario using SMAC (Hutter, Hoos, and Leyton-Brown 2011) to configure the parameters of a GRASP-like Greedy Adaptive Search Procedure (GASP) that solves the packing problem at hand.

Solution approach

In order to solve the previously described problem, researchers have created several heuristic construction algorithms based on GASP (Perboli, Crainic, and Tadei 2011). In this approach solutions are iteratively generated in parallel threads by inserting the items in an order corresponding to a score per item. This score is updated in each iteration. Three different techniques are used, in which two of them are also capable of handling "Tetris"-items. Since the handling of such items alters the behavior of the algorithm significantly, those techniques are split into two separate algorithms for tuning. The first technique is called Extreme Point Insertion (EPI) and uses so-called Extreme Points (EP) (see (Crainic, Perboli, and Tadei 2008)) at which new items are positioned. The list of available EPs is updated every time a new item is inserted. EPI-t refers to the "Tetris" variant of this approach. The second technique is called Push Insertion (PI), which uses fixed insertion points to initially allocate the items and subsequently pushes them towards the container's origin. PI-t refers to the "Tetris" variant of this approach. The last technique is called Space Defragmentation (SD) (see (Zhu et al. 2012)) and is an extension of the first approach. In addition to using EPs for insertion the technique focuses on increasing the density of the packing by pushing items away from newly allocated ones and subsequently all of them towards the containers origin. Thus, the items change their position constantly which makes an efficient integration of "Tetris" items difficult.

Algorithm configuration approach

The heuristic approaches described have a number of configurable parameters. We use the SMAC algorithm proposed by (Hutter, Hoos, and Leyton-Brown 2011). We perform configuration for the EPI, EPI-t, SD, PI and PI-t methods. While the basic algorithms are tuned on 100 instances only containing cuboid items, the "Tetris"-variants are tuned on 100 instances which also contain "Tetris"-shaped items. Both of the sets are split into a training set of 60 instances and a test set of 40 instances.

We tune each of the five algorithms for five days. Each single execution is limited with a five minute cutoff. The objective is set to solution quality, instead of time, because the time available for the solution process does not vary in the application field. The objective function value of the solution

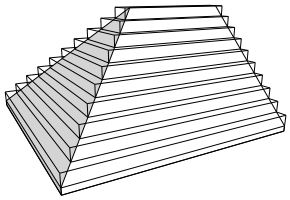

Figure 1: "Tetris"-approximation.

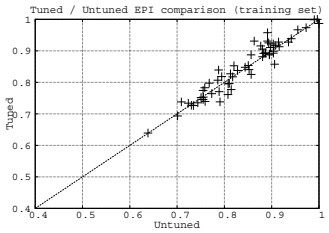

Figure 2: Results for EPI (training set)

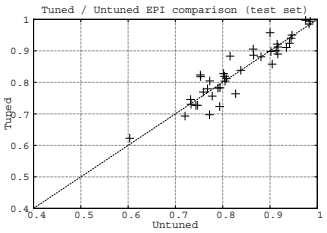

Figure 3: Results for EPI (test set)

is given by the utilized volume of the containers. The objectives are combined by the mean metric of SMAC. Also, due to the parallelization of the algorithms, determinism can not be guaranteed. We describe the parameters of the algorithms along with their domains as follows. Parameters not used by a specific algorithm are ignored for the corresponding run.

- *ItemOrder*: The initial order of the items when inserting. $[0, \ldots, 23] \subset \mathbb{Z}$, (EPI, EPI-t, PI, PI-t, SD)
- *BestFit*: A boolean parameter defining whether to use a merit-function or not. If the value is false, every item is inserted at the first valid insertion point, otherwise the best available one is used as indicated by the merit-function. $\{true, false\}$, (EPI, EPI-t)
- *MeritType*: A conditional parameter identifying the specific merit-function to use, if *BestFit* is activated. $[0, \ldots, 6] \subset \mathbb{Z}$, (EPI, EPI-t)
- *InflateAndReplaceInsertion*: Defines whether to use or skip the inflate-and-replace strategy of the SD technique. $\{true, false\}$, (SD)
- *NormalizationOrder*: The order by which the container is normalized, i.e. items are pushed to the corresponding directions consequentially until no further push is possible. {xyz,zyx,zxy,yzx,xzy,yxz}, (PI, PI-t, SD)
- *ScoreBasedOrder*: Deactivates the score-based sorting of items and substitutes it with a complete random approach. $\{true, false\}$, (EPI, EPI-t, PI, PI-t, SD)
- i^{RD}: The maximal distance without an improvement on the objective value before the score is reinitialized. $[50, \ldots, 2500] \subset \mathbb{Z}$, (EPI, EPI-t, PI, PI-t, SD)
- r^S: A "salt"-value applied randomly for every item, hence, increasing this value increases a certain random influence on the method. $[0, \ldots, 0.9] \subset \mathbb{R}$, (EPI, EPI-t, PI, PI-t, SD)
- m^I: The initial score modification. $[0.01, \ldots, 1] \subset \mathbb{R}$ (EPI, EPI-t, PI, PI-t, SD)
- m^{max}: The maximum score modification $[1, \ldots, 5] \subset \mathbb{R}$, (EPI, EPI-t, PI, PI-t, SD)
- s^P: The number of minimal swaps $[1, \ldots, 4] \subset \mathbb{Z}$, (EPI, EPI-t, PI, PI-t, SD)
- s^{max}: The number of maximal swaps $[4, \ldots, 8] \subset \mathbb{Z}$, (EPI, EPI-t, PI, PI-t, SD)

Computational results

Overall, the configuring of the parameters did not obtain significantly improved configurations. As depicted by the following graphs for the EPI algorithm, the tuning has a slight positive (but also negative impact) depending on the particular instances. This can be seen by comparing the result of the default parameters depicted on the x-axis with the tuned configuration depicted on the y-axis. The respective axes identify the obtained evaluation values (relative volume utilization) by the two configurations. For both the training set (Figure 2) and the test set (Figure 3) only a very slight positive trend is seen. The results for the other algorithms are very similar. Thus, the trend is too small to verify a positive effect of the configuration. This might be due to the very heterogeneous instances requiring different algorithm settings. Hence, in future it may be useful to integrate instance features to find better configurations depending on those. The result would be an automatic algorithm configuration routine before solving the respective instance exploiting the results of the tuning.

Conclusion

We presented an algorithm configuration scenario for three dimensional knapsack problems in the air-cargo industry. Although the results of the configuration were not significantly better than the default configuration, there is hope that other strategies for configuring these algorithms could provide better performance.

References

Crainic, T. G.; Perboli, G.; and Tadei, R. 2008. Extreme point-based heuristics for three-dimensional bin packing. *INFORMS Journal on Computing* 20(3):368–384.

Fasano, G. 2013. A global optimization point of view to handle non-standard object packing problems. *Journal of Global Optimization* 55(2):279–299.

Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2011. Sequential model-based optimization for general algorithm configuration. In *Proceedings of LION-5*, 507–523.

Perboli, G.; Crainic, T. G.; and Tadei, R. 2011. An efficient metaheuristic for multi-dimensional multi-container packing. In *Automation Science and Engineering (CASE), 2011 IEEE Conference on*, 563–568.

Wäscher, G.; Haußner, H.; and Schumann, H. 2007. An improved typology of cutting and packing problems. *European Journal of Operational Research* 183(3):1109–1130.

Zhu, W.; Zhang, Z.; Oon, W.-C.; and Lim, A. 2012. Space defragmentation for packing problems. *European Journal of Operational Research* 222(3):452–463.