University of Paderborn
Faculty of Business and Economics
Depatment of Information Systems
Decision Support & Operations Research Lab

Working Paper

WP1402

An Algorithm Selection Benchmark of the Container Pre-Marshalling Problem

Kevin Tierney

Paderborn, Germany
March, 2014

1. Introduction

The container pre-marshalling problem (CPMP) is a well-known, NP-hard problem in the container terminals literature Stahlbock and Voß (2008) that was first introduced in Lee and Hsu (2007). In the CPMP, stacks of containers at a container terminal are sorted by a crane such that containers that must leave the stacks first are placed on top of containers that must leave the stacks later. The crane can only access the top container on each stack, each stack has a maximum height, and the maximum number of stacks is fixed. The goal of the CPMP is to find the minimal number of container movements necessary to ensure that all of the stacks are sorted by the exit time of each container from the stacks.

Sorting containers prevents costly shuffling operations from delaying containers from leaving the stacks. Delayed containers subsequently delay ships, trains or trucks, resulting in a less efficient (i.e., more costly) supply chain.

A recent approach for solving the CPMP to optimality by Tierney et al. (2014) presents two state-of-the-art approaches, based on A* and an IDA*, respectively, to find optimal solutions to the CPMP. We use this approach to form a dataset for algorithm selection, and introduce 15 novel features to describe CPMP instances. This dataset contains two parameterizations of the A* algorithm and two parameterizations of the IDA* approach. In both cases, the parameterizations involve a symmetry breaking heuristic used within the search. The CPU time to optimality for each parameterized solver is given for instances from the work of Bortfeldt and Forster (2012) and from Caserta and Voß (2009).

2. The Pre-Marshalling Problem

Given an initial layout of a bay, the goal of the CPMP is to find the minimal number of container movements (or *rehandles*) necessary to eliminate all mis-overlays in the bay. Formally, a bay contains S stacks which are at most T tiers high. Each container in the bay is assigned a priority, $p_{st} \in \mathbb{N}_0$ $(s \in S, t \in T)$. We set $p_{st} = 0$ if there is no container at the position s, t. Containers with a smaller priority value are retrieved first, meaning they must be above containers with a larger priority value in a configuration with no mis-overlays. Thus, a bay has no mis-overlays iff $p_{st} \leq p_{s,t+1}$ for all $s \in S$, $1 \leq t < |T|$. As previously mentioned we focus on a single bay, thus no movements between bays are allowed and all containers are assumed to be of the same size.

Consider the simple example of Figure 1(a), which shows a bay composed of three container stacks where containers can be stacked at most four tiers high. Each container is represented by a box with its corresponding priority.[1] This is not an ideal layout as the containers with priority 2,

[1] We note that multiple containers may have the same priority, but in order to make containers easily identifiable, in this example we have assigned a different priority to each container.

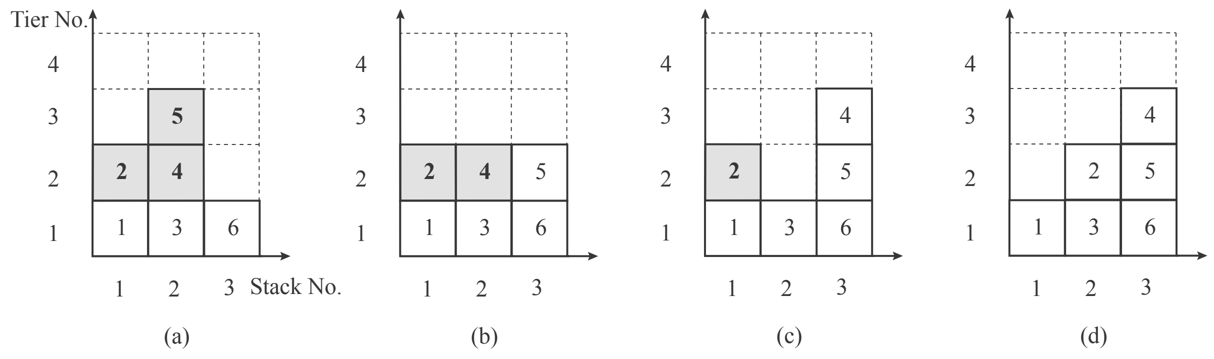

Figure 1 An example solution to the CPMP with mis-overlays highlighted. From Tierney et al. (2014).

4 and 5 will need to be relocated in order to retrieve the containers with higher priority (1 and 3). That is, containers with priority 2, 4 and 5 are mis-overlaid. Consider a container movement (f, t) defining the relocation of the container on top of the stack f to the top position of the stack t. The containers in the initial layout of Figure 1 (a) can reach the final layout (d) with three relocation moves: $(2, 3)$ reaching layout (b), $(2, 3)$ reaching layout (c) and $(1, 2)$ reaching layout (d) where no mis-overlays occur.

Pre-marshalling is important both in terms of operational and tactical goals at a container terminal. We refer to Tierney et al. (2014) for more information and a discussion of related work.

3. Algorithm Selection Dataset

In this section, we describe a set of novel features and an algorithm selection dataset for the CPMP.

3.1. Features

The features used in our dataset are given in Figure 2. A key aspect of these features is that they are extremely easy to compute. Even on large instances not even 0.001 CPU seconds were required on our Intel Core i7-4702HQ CPU. Furthermore, the features encompass a number of aspects of the problem. The first 5 features address the problem size and density of containers. Problems with a high container density are likely to pose different challenges than those where there is lots

1. Number of stacks
2. Number of tiers
3. Tiers/stacks
4. Container density
5. Empty stack percentage
6. Percent of all slots that are mis-overlayed
7. Bortfeldt & Forster lower bound

8–11. Min/max/mean/stdev container priority counts

12-15. Min/max/mean/stdev priority of top non-mis-overlayed container

Figure 2 Features for the CPMP.

of empty space. Features 6 and 7 look at the lower bound on the number of moves necessary to fix the bay in two different ways. Feature 6 simply counts the number of mis-overlayed containers, which is an obvious lower bound to the problem. Feature 7 provides the lower bound from Bortfeldt and Forster (2012), which analyzes indirect container movements in addition to the mis-overlays present in feature 7. Features 8 through 11 offer information on how many containers belong to each priority group. Some instances have a one-to-one mapping of groups to containers, whereas other instances have large numbers of containers in particular priority groups. Finally, features 12 through 15 attempt to uncover the structure of the priorities of the top non-mis-overlaid container on each stack. It is possible that low values of this feature could lead to more difficult problems, or at least problems where more moves are required, as low valued containers will have to trade places with higher valued ones.

We make no claim that this is an exhaustive list of features, indeed there are a number of other possibilities, such as probing features using various heuristics. At the very least, these features offer a good starting point for algorithm selection research into the CPMP.

3.2. Algorithms

The dataset consists of four algorithms, which are parameterizations of the A* and IDA* approaches. All algorithms solve the CPMP to optimality, returning the number of moves required to sort the bay. We do not give details about those algorithms or heuristics here, and refer curious readers to Tierney et al. (2014). We parameterize the unrelated move symmetry breaking heuristic described in the paper with a less than and greater than ordering constraint for both A* and IDA*, giving us four algorithms in total. The ordering used is completely arbitrary and has no bearing on the completeness of the algorithm, but we noticed that there are significant differences in performance depending on the type of constraint used. Furthermore, there are performance differences between A* and IDA* as well, as A* is able to make greedier decisions in the search tree than IDA*, but at the expense of high memory usage.

3.3. Datasets

We provide CPU time data for two well-known pre-marshalling datasets from Bortfeldt and Forster (2012) and Caserta and Voß (2009). We prune all instances in which all four parameterized algorithms timeout. Note that in Tierney et al. (2014) results are also presented for a dataset from Expósito-Izquierdo et al. (2012). We do not include this dataset here because the instances were easy for all algorithms involved, and generally only required a few seconds to solve. Thus, an algorithm selection approach is not necessary for them. In total, there are 267 instances from Bortfeldt and Forster (2012) and 260 instances from Caserta and Voß (2009). All runtime data was generated on an AMD Opteron 2425 HE processor running at 2100 MHz with a one hour timeout.

Solver	PAR-1	PAR-10	Timeouts
astar-symmulgt-transmul	1296.3	12239.8	178
astar-symmullt-transmul	1387.8	13253.4	193
idastar-symmulgt-transmul	916.4	7002.9	99
idastar-symmullt-transmul	1059.3	8068.1	114
VBS	227.6	227.6	0

Table 1 Scores for the solvers and the virtual best solver (VBS).

Table 1 provides solution time information for the dataset. We provide the penalized average runtime (PAR) score (in seconds) with a penalization of 1 (PAR-1, i.e., no penalization) and where the timeout is multiplied by 10 (PAR-10). The best solver in terms of timeouts, PAR1 and PAR10 is idastar-symmulgt-transmul, which is an IDA* algorithm using a greater than constraint for its unrelated move symmetry breaking constraint, as well as a transitive move prevention heuristic. The virtual best solver (VBS) provides the absolute best values that any algorithm selection approach could achieve on the dataset (i.e., the VBS selects the best solver for every instance). The runtimes show that there are clearly gains to be made for an algorithm selection approach over just using the single best solver, both in terms of PAR score and in terms of the number of timeouts.

4. Conclusion

We presented an algorithm selection dataset for the pre-marshalling problem, a well-known problem from the container terminals literature. We introduced novel features for the dataset, and showed that there is significant room for algorithm selection techniques to improve over the single best solver on the dataset. We hope that this dataset will help researchers in the field of algorithm selection test and evaluate their approaches.

Bibliography

Bortfeldt, A., F. Forster. 2012. A tree search procedure for the container pre-marshalling problem. *European Journal of Operational Research* **217**(3) 531–540.

Caserta, M., S. Voß. 2009. A corridor method-based algorithm for the pre-marshalling problem. M. Giacobini, A. Brabazon, S. Cagnoni, G. Caro, A. Ekárt, A.I. Esparcia-Alcázar, M. Farooq, A. Fink, P. Machado, eds., *Applications of Evolutionary Computing*, *Lecture Notes in Computer Science*, vol. 5484. Springer Berlin Heidelberg, 788–797.

Expósito-Izquierdo, C., B. Melián-Batista, M. Moreno-Vega. 2012. Pre-marshalling problem: Heuristic solution method and instances generator. *Expert Systems with Applications* **39**(9) 8337–8349.

Lee, Y., N.Y. Hsu. 2007. An optimization model for the container pre-marshalling problem. *Computers & Operations Research* **34**(11) 3295–3313.

Stahlbock, R., S. Voß. 2008. Operations research at container terminals: a literature update. *OR Spectrum* **30**(1) 1–52.

Tierney, K., D. Pacino, S. Voß. 2014. Solving the pre-marshalling problem to optimality with A* and IDA*. Tech. Rep. Working Paper #1401, Decision Support & Optimization Lab, University of Paderborn.