Stack-up algorithms for palletizing at delivery industry*

J. Rethmann E. Wanke

University of Düsseldorf, Department of Computer Science
Universitätsstraße 1, D-40225 Düsseldorf, Germany
Email: {rethmann,wanke}@cs.uni-duesseldorf.de

Abstract

We study the following multi-objective combinatorial stack-up problem from delivery industry. Given a sequence q of labeled bins and two positive integers s and p. The aim is to stack-up the bins by iteratively removing one of the first s bins of the sequence and put it to one of the p stack-up places. Each of these places has to contain bins of only one label, bins of different labels have to be placed on different places. If all bins of a label are removed from q, the corresponding place becomes available for bins of another label.

We analyze the worst-case performance of simple algorithms for the stack-up problem that are very interesting from a practical point of view. In particular, we show that the so-called Most-Frequently on-line algorithm is $(2, 2)$-competitive and has optimal worst-case on-line performance.

Key words on-line algorithms, competitive analysis, complexity, approximation, control

1 Introduction

We consider the combinatorial problem of stacking up bins from a conveyor onto pallets. This problem originally appears in *stack-up systems* that play, especially in recent years, an important role at delivery industry and warehouses. The stack-up problem is, for example, strongly involved with the picking procedure at order-picking systems, which is a preceding working process immediately before the bins have to be stacked up. To explain more precisely the practical background of our research, let us first say some words about pick-to-belt order-picking systems.

At delivery industry, customers usually order a large amount of articles. These articles have to be packed into bins for delivery which is usually done by human workers, see [1, 3, 8, 12] for more details. Each order consists, in general, of several bins, because the articles ordered by a customer usually do not fit into a single bin. In the order-picking

*Short abstracts of this paper are published in the proceedings of ESA'97 [11] and ISAAC'98 [13].

system, the articles are picked from shelves and put into the bins. The initially empty bins enter the order-picking system order by order, but during the packing process the bins are mixed up between different orders. This rearrangement of the bins can not be avoided, because each bin has its own packing time that can not be estimated in advance. Additionally, the availability of all articles is not guaranteed each time. However, for delivery reasons, it is absolutely necessary to place all bins belonging to one costumer order onto the same pallet. Therefore, some bins have to be temporarily stored before they are moved onto pallets.

Bins arrive the stack-up system on the main conveyor of the order-picking system. At the end of the main conveyor they enter a *cyclic storage conveyor*. From the storage conveyor the bins are pushed out to *buffer conveyors*, where they are queued. The bins are picked-up by stacker cranes from the end of a buffer conveyor and moved onto pallets, which are located at some *stack-up places*. There is one buffer conveyor for each stack-up place. Automatic driven vehicles take full pallets from stack-up places, put them onto trucks and bring new empty pallets to the stack-up places, see also [12].

Many details of the architecture are unimportant to compute efficiently an order in which the bins can be palletized. We model the buffer and cyclic storage conveyors by one simple random access *storage* from which the bins can be picked-up and moved onto pallets. In real live, both conveyors are necessary to enable a smooth stack-up process irrespective of the real speed the cranes and conveyors are moving. Logistic experiences over 10 years lead to such high flexible conveyor systems at delivery industry. So we do not intend to modify the architecture of existing systems, but try to develop efficient algorithms to control them. Figure 1.1 shows a sketch of a simplified stack-up system.

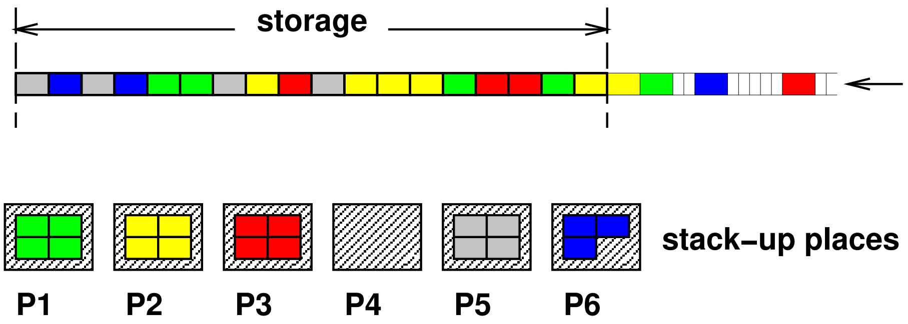

Figure 1.1: A sketch of a stack-up system.

Our aim in controlling stack-up systems is to avoid blocking situations. The system is *blocked*, if all stack-up places are occupied, and the storage is completely filled with bins not destined for the pallets on the stack-up places. To unblock the system, the missing bins have to be picked-up manually and moved to pallets by human workers.

Our model is the first attempt to capture important parameters necessary for an efficient and provable good algorithmic controlling of stack-up systems. The stack-up system that has initiated our research is located at Bertelsmann Distribution GmbH in Gütersloh, Germany. On certain days, several thousands of bins are stacked-up using a storage conveyor with a capacity of approximately 60 bins and 24 stack-up places, while approximately 32 bins are destined for any pallet.

From a theoretical point of view, we are given a sequence of bins $q = (b_1, \ldots, b_n)$ and two integers s and p. Each bin b_i of sequence q is destined for exactly one pallet. The bins have to be removed step by step from the first s positions such that at each intermediate step at most p pallets are open. A pallet t is called open, if not all but at least one bin for t is already removed from q. If a bin is removed then all bins to the right are shifted one position to the left. Integer s represents the capacity of the storage conveyor from which the bins can be picked up by the stacker cranes. Integer p represents the number of available stack-up places. A sequence q is called (s, p)-sequence, if it can be processed with a storage capacity of s bins and p stack-up places.

Given a sequence of bins q, the aim is to compute a processing of q minimizing simultaneously the used storage capacity s and the used number of stack-up places p. We distinguish between two optimization problems: minimizing the storage capacity s subject to a constraint on the number of stack-up places p, or minimizing the number of stack-up places p subject to a constraint on the storage capacity s. In real-world applications it is quite usual that more than one objective has to be minimized. Many approximation algorithms for several problems with multiple objectives can be found in literature, see for example in [6, 9, 14].

In this paper, we consider algorithms where a sequence of bins q and a storage capacity s is given to the input, while a number of stack-up places p and an (s, p)-processing of q is computed by the algorithm. We are basically interested in *on-line* algorithms that compute almost all decisions at a time where only a part of the complete sequence of bins is known. At each step, on-line algorithms only see the first s bins of the sequence and they know whether any bin they see is the last one for its destination.

We study the following two questions. Given a sequence of bins q and a storage capacity s, how many stack-up places $p_A(q, s)$ are used by algorithm A to process q with respect to storage capacity s, or given a sequence of bins q and a number of stack-up places p, how large has to be the storage capacity $s_A(q, p)$ such that the processing of q by algorithm A takes at most p stack-up places? Algorithm A is called an (c, d)-approximation algorithm if we have $p_A(q, c \cdot s) \leq d \cdot p$ or $s_A(q, d \cdot p) \leq c \cdot s$, respectively, for each (s, p)-sequence q.

The combinatorial stack-up problem seems to be not investigated by other authors up to now although it has important practical applications. However, the following facts are already known. In [12] it is shown that the stack-up decision problem is NP-complete [5], but can be solved efficiently if the storage capacity s or the number of stack-up places p is fixed. In [10] a polynomial time *off-line* approximation algorithm is introduced that yields a processing of any (s, p)-sequence with a storage capacity of $s \cdot \lceil \log_2(p + 1) + 1 \rceil$ bins and $p + 1$ stack-up places.

In this paper, we study the worst-case performance of simple stack-up algorithms that do not know the complete sequence of bins in advance. The worst-case study is done by competitive analysis [2, 4, 7], that is, we compare the performance of our algorithms with optimal off-line solutions. We prove that (off-line) stack-up algorithms use at most $s-1+p$

stack-up places to process (s,p)-sequences with respect to storage capacity s. Similarly, to process an (s,p)-sequence q with respect to p stack-up places a storage capacity of $s + p(f_q - 2) + 1$ bins is always sufficient, where f_q denotes the maximum number of bins per pallet in sequence q.

We show that the Most-Frequently algorithm – or MF algorithm for short – takes at most $p + p \cdot \log_2(s+1) + 1$ stack-up places to process an (s,p)-sequence with respect to a storage capacity of s bins. Moreover, the Most-Frequently algorithm needs at most a storage capacity of $(p+1)s - p$ bins to compute a processing for each (s,p)-sequence with at most p stack-up places. Furthermore, we prove that the Most-Frequently algorithm is a (2,2)-approximation algorithm and is therefore the best polynomial time algorithm for the stack-up problem known up to now that approximates both objectives within a certain small factor.

We also show that the Most-Frequently algorithm has optimal on-line worst-case performance. More precisely, we show that for each on-line algorithm A there is an (s,p)-sequence q such that for all (s,p)-sequences q' it holds $p_A(q, s') \geq p_{\mathrm{MF}}(q, s') - 1$, where s' denotes some integer greater than or equal to s. To prove this, we first give a lower bound on the number of stack-up places each on-line algorithm takes. That is, for given integers s, p, and $s' \geq s$, we define step by step an (s,p)-sequence by playing the role of an adversary who defines the next part of the sequence depending on the on-line algorithm's choice. Second, we prove an upper bound on the number of stack-up places the Most-Frequently algorithm takes. More precisely, we show that for any given integer $c \geq 0$, the Most-Frequently algorithm takes at most $p + p \cdot \log_2(\frac{s}{c+1} + 1) + 1$ stack-up places to process each (s,p)-sequence with respect to storage capacity $s + c$. It is, in general, not surprising to obtain a polynomial time algorithm approximating both objectives within a small factor although no such algorithm is known for approximating one objective, but it is quite unusual that such a provable good performance can be achieved by a simple *on-line* algorithm.

The paper is organized as follows. In section 2, we introduce the preliminary notations for processing sequences of bins. We consider some stack-up strategies and we define the stack-up algorithms First-In, First-Done, Most-Frequently and Greedy. In section 3, we prove upper bounds for general stack-up algorithms, and in section 4 we prove upper bounds for our algorithms. In section 5, we prove lower bounds of on-line stack-up algorithms and compare them with the upper bounds of the Most-Frequently algorithm. Finally, some concluding remarks are given.

References

[1] N. Ascheuer, M. Grötschel, and A. Abdel-Aziz Abdel-Hamid. Orderpicking in an Automatic Warehouse: Solving Online Asymetric TSPs. Technical Report 35/98, Konrad-Zuse-Zentrum für Informationstechnik, D-14195 Berlin, Germany, 1998.

[2] Alan Borodin. *On-line Computation and Competitive Analysis.* Cambridge University Press, 1998.

[3] R. de Koster. Performance approximation of pick-to-belt orderpicking systems. *European Journal of Operational Research*, 92:558–573, 1994.

[4] A. Fiat and G.J. Woeginger, editors. *Online Algorithms: The state of the art*, volume 1442 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[5] M.R. Garey and D.S. Johnson. *Computers and Intractability.* W.H. Freeman and Company, San Francisco, 1979.

[6] J.H. Lin and J.S. Vitter. ϵ-Approximations with Minimum Packing Constraint Violation. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 771–781. ACM, 1992.

[7] M.S. Manasse, L.A. McGeoch, and D.D. Sleator. Competitive algorithms for on-line problems. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 322–333. ACM, 1988.

[8] H.D. Ratcliff and A.S. Rosenthal. Orderpicking in a Rectangular Warehouse: A Solvable Case of the Traveling Salesman Problem. *Operations Research*, 31:507–521, 1983.

[9] R. Ravi, M.V. Marathe, S.S. Ravi, D.J. Rosenkrantz, and H.B. Hunt. Many birds with one stone: Multi-objective approximation algorithms. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 438–447. ACM, 1993.

[10] J. Rethmann and E. Wanke. An approximation algorithm for stacking up bins from a conveyer onto pallets. In *Proceedings of the Annual Workshop on Algorithms and Data Structures*, volume 1272 of *Lecture Notes in Computer Science*, pages 440–449. Springer-Verlag, 1997.

[11] J. Rethmann and E. Wanke. Competitive analysis of on-line stack-up algorithms. In *Proceedings of the Annual European Symposium on Algorithms*, volume 1284 of *Lecture Notes in Computer Science*, pages 402–415. Springer-Verlag, 1997.

[12] J. Rethmann and E. Wanke. Storage Controlled Pile-Up Systems. *European Journal of Operational Research*, 103(3):515–530, 1997.

[13] J. Rethmann and E. Wanke. An optimal algorithm for on-line palletizing at delivery industry. In *Proceedings of the International Symposium on Algorithms and Computation*, volume 1533 of *Lecture Notes in Computer Science*, pages 109–118. Springer-Verlag, 1998.

[14] A. Warburton. Approximation of pareto optima in multiple-objective shortest path problems. *Operations Research*, 35(1):70–79, 1987.