Acta Cybernetica **24** (2019) 187–195.

# Linear Time Ordering of Bins using a Conveyor System*

Géza Makay[a] and András Pluhár[b]

**Abstract**

A local food wholesaler company is using an automated commissioning system, which brings the bins containing the appropriate product to the commissioning counter, where the worker picks the needed amounts to 12 bins corresponding to the same number of orders. To minimize the number of bins to pick from, they pick for several different spreading tours, so the order of bins containing the picked products coming from the commissioning counter can be considered random in this sense. Recently, the number of bins containing the picked orders increased over the available storage space, and it was necessary to find a new way of storing and ordering the bins to spreading tours. We developed a conveyor system which (after a preprocessing step) can order the bins in linear space and time.

**Keywords:** material flow control, bin ordering, modified Yehuda-Fogel algorithm

## 1 Introduction

Automated material handling systems (AMHSs) are used in several different areas throughout the world: baggage handling, distribution, postal services, etc. The different market sectors have different goals and challenges, so it is very hard to create a common platform for all of them. There are several approaches to cope with this problem, see for example Haneyah et al. [4] and the references therein.

This paper is motivated by a collaboration of the authors and a local food wholesaler company. The company is using an automated commissioning system, where 3 PLC-controlled robots (each serving from 2 rows on their left and right) bring out and take the bins into 6048 storage spaces. The bins travel through a PLC-controlled conveyor system to the commissioning counters, where the workers

*This research was supported by the Ministry of Human Capacities, Hungary [grant 20391-3/2018/FEKUSTRAT] and by the National Research, Development and Innovation Office - NKFIH [grant SNN-117879].

[a]Bolyai Institute, University of Szeged, Hungary, E-mail: makayg@math.u-szeged.hu

[b]Department of Computational Optimization, University of Szeged, Hungary, E-mail: pluhar@inf.u-szeged.hu

DOI: 10.14232/actacyb.24.2.2019.2

pick one product for several orders at the same time, which is faster than picking the product for one order only. This method decreases the number of needed bins to pick from, therefore decreases the work of the workers, the robots and the conveyor system. There is a controlling system over the PLC, called MFC (Material Flow Control) system, which (among others) optimizes the order of bins arriving to the commissioning places. When a bin is full or the order's picking is complete, the worker places the commissioned bin to the conveyor system, and the MFC brings it to one of the storage places. This also means, that the commissioned bins are coming out of the commissioning place in more-or-less random order, but it is not a problem: when a spreading tour is complete, the robots bring them out in the correct order. A spreading tour is practically an order of customers visited by one truck or van, the customers are chosen so that the truck's load is utilized as much as possible. And the correct order of the bins for one spreading tour is the reverse order of the customers' visiting order on the tour, since when the bins are placed on a pallet in this order and put in a truck or van, the truck or van works as a LIFO (last in first out) stack.

Recently, the number of commissioned bins increased over the available storage space, and the company needed a new way of storing and ordering the bins to spreading tours without using the robots. The problem is twofold: we need an affordable storage and sorting hardware, but a hardware which is capable of ordering up to 100 bins (that is the maximum number of bins on one spreading tour). The conveyor system we use for commissioning is quite expensive, since each module has its own electric motor, rolling cylinders to pass the bins on and some of the modules (where the bins need to change traveling directions) even have belts and a pneumatic system to lift the bins. On the other hand a conveyor belt is relatively cheap: one motor and a belt for 40-50 bins. So we decided to use conveyor belts wherever possible for storage, and a mix of conveyor belts and modules for ordering.

In Section 2 we describe the mathematical model for this problem. One of the best ordering methods using conveyor belts and modules is the merge sort, so our main problem is to create monotone subsequences from the original sequence of bins to use this sorting method on this hardware. In Section 3 we overview the relevant literature and give an example for such partitioning using the Erdős-Szekeres Theorem [3]. The modified Yehuda-Fogel algorithm [5] is able to find this partioning faster, but since the moving of bins is much slower than this preprocessing step, the total time of ordering mainly depends on the former. In Section 4 we show a possible realization of the physical system, which is able to reorder the bins in linear time and space.

## 2 Modeling the problem

The problem has quite a few connections to several areas in the literature. One of the most closely related topic is the parallel stack loading problem [1], which uses LIFO stack structures to store items in (preferably) decreasing order so that no blockages occur while unloading the stacks. However, generally blockages may

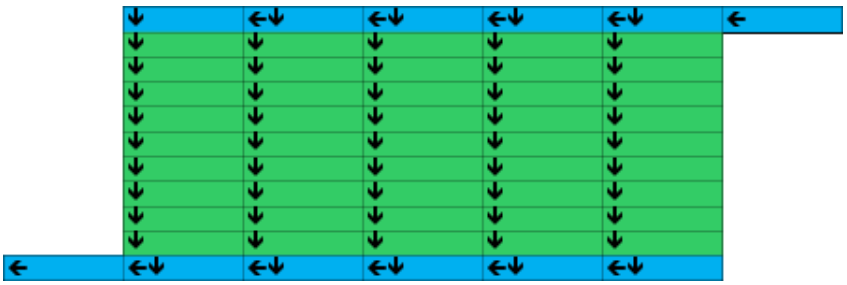

Figure 1: A storage setup

occur, in other words, the items cannot be stored such a way that they can be retrieved from the stacks in the desired order. In our case this is not allowed, we do need the correct order of bins. The main idea of the solution is to use parallel loading of LIFO and FIFO structures corresponding decreasing and increasing parts of the bin sequence to achieve this goal.

There are several methods to sort $n$ numbers quickly, an obvious one is quicksort which runs in an expected $n \log n$ time. However, this method exchanges elements far from each other, it would need a complicated hardware and it would take a long time to perform the exchange physically, therefore it is not suitable for bin ordering. On the other hand merge-sort is just perfect for sorting bins on a conveyor system. To see this first we describe a relatively cheap conveyor system capable of storing and transporting bins and performing merge-sort.

A conveyor system is made up from two major parts: a conveyor belt moving the bins in one direction, and a *direction changer module*. A direction changer module can pick up a bin from one direction using (for example) a belt, and by lowering the belt deposits the bin to the rolling cylinders, which move the bin forward in another direction. A simple storage system is shown on Figure 1. The green modules are conveyor belts for storage, while the blue modules are the direction changer modules capable of forwarding the bins from the blue modules to the green ones and from the green ones to the blue ones. The bins are coming in from the upper-right direction and leaving the storage system through the lower-left module. The bins can be sorted to the green modules by spreading tours, but within one spreading tour their order is still not specified, can be considered random.

Let us define the ordering problem. We assume that the bins are on a conveyor belt in random order, and we want them on a conveyor belt in the correct order. One step of a conveyor belt is when it moves all bins on it one step further. One step of a direction changing module is when the module drops the bin to the next belt/module and picks up the next bin from the previous belt/module. Physically a direction changer module cannot perform these actions in parallel, but it would make computation much more complicated if we took this into account. However it does not change the order of steps needed for reordering the bins: it adds a factor of 2 in the very worst case.

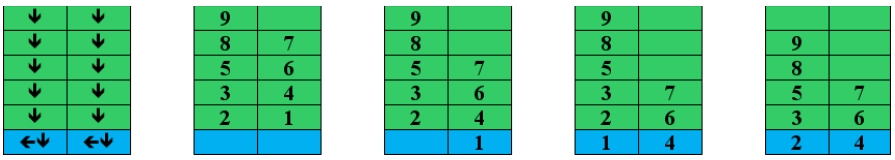


Figure 2: Merge-sort for 2 lines

# 3 The mathematical background of the problem

We proposed merge sorting for the physical hardware consisting of conveyor belts and direction changing modules. Our next theorem shows that an appropriate conveyor system is capable of performing merge-sort in linear time. Later we show that we can construct the monotone subsequences needed for the algorithm presented in this theorem.

**Theorem 1.** *Suppose, that we have $k$ lines of ordered set of bins represented here by increasing numbers, and we want to merge that $k$ lines into one ordered set of bins. We claim that this can be executed in at most $n + k + 1$ steps, where $n$ is the number of bins. One step here is a movement of all bins that need to be moved from one slot to the next one.*

*Proof.* We prove our claim by induction. For $k = 1$ the statement is trivial, as the bins are already in one ordered set.

An example of the case $k = 2$ is shown on Figure 2. The first column shows the direction in which the conveyor system can move the bins. On the second part the original setup of the two ordered list of numbers, and the rest shows the first couple of merging steps. Once a bin is at the lower-left module, it is the next in the merged order, it is considered sorted, and it steps out of the merging system to the left. In each step that number goes to the lower-left position, which is smaller, and its line moves forward. It is easy to see that the first number appears in the lower-left position in one or two steps, and after that a number comes in each step. So the total number of steps to clear the sorting system of all bins is either $n + 2$ or $n + 3$ which gives the result for $k = 2$.

Suppose that we know the claim of the theorem for $k$ lines. Let us consider the case of $k + 1$ lines. From our induction assumption we know that while merge-sorting the right $k$ lines, the smallest number will appear in at most $k + 1$ steps in the lower-left position of that $k$ lines. Then the merging continues as we have already shown for two lines, i.e. it takes at most one more step for the smallest of the $k + 1$ lines to appear at the lower-left position of the whole table, then the numbers are coming continuously, so all together the number of steps is at most $n + k + 2$. □

To use merge-sorting one needs already sorted numbers, i.e. if we have $n$ numbers then we would need monotonic subsequences of that, as they can be stored on

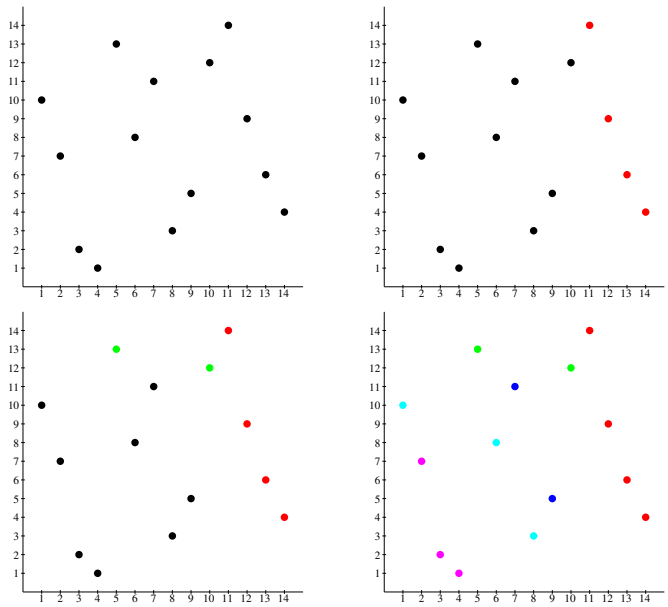

Figure 3: Illustration for finding long monotonic sequences

different lines and then merged. First we show a heuristics to find long monotone subsequences based on the proof of the Erdős and Szekeres Theorem [3].

**Theorem 2.** *For given $r$ and $s$ any sequence of distinct numbers with length at least $(r-1)(s-1)+1$ contains either a monotonically increasing subsequence of length $r$ or a monotonically decreasing subsequence of length $s$.*

As a special case we get that a sequence of $n$ numbers contains a monotonic subsequence of length $\lceil\sqrt{n}\,\rceil$ where $\lceil x\rceil$ is the smallest integer such that $x \leq \lceil x\rceil$.

Applying this result for the original sequence, then to the sequence from which the found monotonic subsequence is removed, etc, one can find monotonic subsequences of the original sequence so that all elements appear in at least one subsequence, i.e. it is a partition of the original sequence to monotonic subsequences. Let us see an algorithm to find a long enough monotonic subsequence. We demonstrate the algorithm on the sequence 10, 7, 2, 1, 13, 8, 11, 3, 5, 12, 14, 9, 6, 4. To illustrate this sequence, take the index of an element as a first coordinate, and the element itself as the second coordinate as shown in Figure 3a.

Let us find *peak elements* in the sequence: they are larger than any elements in the sequence after them. These elements correspond to points in the figure which do not have other points in their upper-right quarter. They are marked by red dots on Figure 3b and they are called the *Pareto border* or layer 1 of the point set. Peeling this Pareto border off, we find another set of peak elements (layer 2)

marked by green on Figure 3c. Continuing this procedure we color all points, all of them will be part of some layer. Now there are two possibilities according to the Theorem of Erdős and Szekeres.

- Either there is a long enough layer corresponding to a monotonically decreasing subsequence. Since we now have $n = 14$ elements,"long enough" means now $\lceil\sqrt{14}\,\rceil = 4$ elements. Layer 1 satisfies this property.
- Or (if all layers have less than 4 elements) then there must be at least 4 layers, which is also true now: we have 5 layers. In this case if we choose one point from layer 5 it is not a peak element considering the points of layer 5 and 4, so there must be a point in layer 4 which is in its upper-right quarter. Then this point is not peak in layer 5, 4 and 3, so there is a point in layer 3 in its upper-right quarter. Continuing this procedure we find one point from each layer forming a monotonically increasing subsequence of length 5.

By removing this long enough monotonic subsequence from the sequence and repeating this procedure one can arrive (for example) to this partitioning: 10, 7, 2, 1, 13, 8, 11, 3, 5, 12, 14, 9, 6, 4. Our algorithm takes $O(n^3)$ time to complete, and it shows the basic method for finding a partitioning of a sequence to monotonic subsequences. The partitioning guaranteed by the Erdős and Szekeres Theorem can be constructed by an asymptotically better algorithm developed by Yehuda and Fogel [6].

**Theorem 3.** *A sequence of $n$ numbers can be partitioned into $2\lfloor\sqrt{n}\rfloor$ monotone subsequences in time $O(n^{1.5})$. All the subsequences can be chosen to be of size $\lceil\sqrt{n}\,\rceil$ or less.*

Brandstädt and Kratsch [2] gave the smaller bound of $\left\lfloor\sqrt{2n+1/4}-1/2\right\rfloor$ on the number of subsequences and proved that it is a tight bound. Recently Yang et al. [5] modified the Yehuda-Fogel algorithm to provide at most $\left\lfloor\sqrt{2n+1/4}-1/2\right\rfloor$ monotonic subsequences of size no more than $\lceil\sqrt{n}\,\rceil$ in $O(n^{1.5})$ time.

If our algorithm has to comply with the second part of the theorem above, then in our example we can simply ignore the extra elements in the monotonic subsequences we find during the procedure. Then we would get the following partitioning for example: 10, 7, 2, 1, 13, 8, 11, 3, 5, 12, 14, 9, 6, 4.

## 4 The physical realization of the sorting system

Now we have the theoretical background to design the physical sorting system.

We assume, that the bins are on a conveyor belt one after the other. Their desired order is represented by numbers. First we apply the modified Yehuda-Fogel algorithm to partition them into monotone subsequences. Since the number of bins are relatively small (in our case under 100), their algorithm runs within milliseconds on a modern computer. Then let us feed the bins from the lower-right module into the construction shown on Figure 4. This construction has

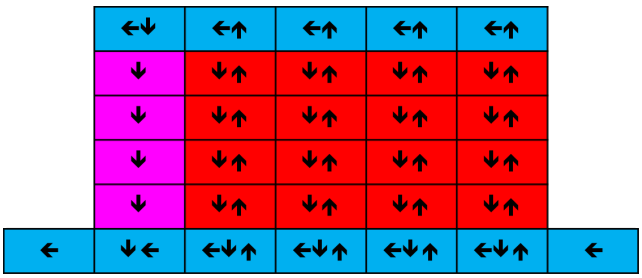

Figure 4: Construction of the sorting conveyor system

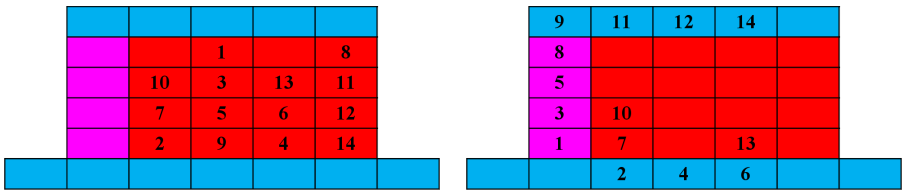


Figure 5: Using the sorting conveyor system

$\left\lfloor \sqrt{2n+1/4} - 1/2 \right\rfloor$ columns of red sorting conveyor belts, each belt has the length $\lceil \sqrt{n} \rceil$ (the figure shows the numbers for our example sequence: for $n = 14$ we have $\left\lfloor \sqrt{2n+1/4} - 1/2 \right\rfloor = 4$ columns of length $\lceil \sqrt{n} \rceil = 4$). Each monotonic subsequence goes into its own red conveyor belt: there are enough belts and they are long enough to accommodate them (see Figure 5a). This partitioning of the bins takes at most $n + \left\lfloor \sqrt{2n+1/4} - 1/2 \right\rfloor + \lceil \sqrt{n} \rceil$ bin moving steps. Monotonically increasing subsequences go up in their red conveyor belts, while monotonically decreasing sequences remain down. Then the system merge-sorts the monotonically increasing subsequences by moving them upward out of the red conveyor belts (see Theorem 1). At the same time the system merge-sorts the monotonically decreasing subsequences by moving them downward out of the red conveyor belts, which makes the merged sequence monotonically increasing (see Figure 5b). Note that on the figure the monotonically decreasing sequences are in the middle of sorting as they need to wait for the monotonically increasing sequence to arrive. Finally the system merge-sorts these two subsequences to produce the final sorted sequence. The last sortings take at most $n + \left\lfloor \sqrt{2n+1/4} - 1/2 \right\rfloor + \lceil \sqrt{n} \rceil + 1$ steps. So all together the sorting needs $2\left(n + \left\lfloor \sqrt{2n+1/4} - 1/2 \right\rfloor + \lceil \sqrt{n} \rceil\right) + 1$ time to complete, that is, it is linear in the number of bins.

The space required to order $n$ bins is $\left(\left\lfloor \sqrt{2n+1/4} - 1/2 \right\rfloor + 1\right)(\lceil \sqrt{n} \rceil + 2)$

which is approximately $n\sqrt{2}$ so the space requirement is also linear in the number of bins.

In a physical setup one does not need the complete sorted sequence of bins standing on a conveyor belt, for example the bins can be moved to a pallet while they are still coming out of the sorting system. Therefore the actual time from the start of the sorting to the point when one can start working and (according to Theorem 1) can continuously work with the sorted sequence is at most $n + 2\left\lfloor\sqrt{2n+1/4} - 1/2\right\rfloor + 2\lceil\sqrt{n}\,\rceil + 1$.

This sorting conveyor system can sort more than $n$ bins, if the sequence can be partitioned so that its monotonic subsequences fit into the sorting system. This can be achieved (for example) by pre-merge-sorting the bins while they are coming out of the storage conveyor belts. With this change one can achieve that all monotonic subsequences are decreasing, and hence we do not even need the upper (blue) and the left (purple) modules and belts containing $\left\lfloor\sqrt{2n+1/4} - 1/2\right\rfloor + 2$ expensive direction changing modules: a cut in the costs. Moreover, the lengths of the subsequences can be maximized, so the sorting system can be completely occupied during the sorting.

The sorting can run even faster by installing several (say: $m$) of these kind of sorting systems, dividing the original sequence into $m$ subsequences of more-or-less equal size, sorting them parallel in the sorting systems and then merging the sorted subsequences into one sequence.

## 5 Conclusions

The problem of ordering bins seems to be neglected in the literature. There are several sorting agents, which can collect items to different containers based on some of their property, but they do not order those items within one container. We constructed a system which solves the ordering problem, so that the number of bin moving steps and the required space is linear in the number of bins. Searching for the modified Yehuda-Fogel algorithm (which is essential for this method to work) did not return an application similar to the one presented in this paper, so this method of ordering the bins looks novel. The system itself is a generalization of the parallel loading of LIFO stacks by adding FIFO capabilities to the stacks.

Although the physical ordering of $n$ bins takes linear time in the number of bins, we should not forget that we need the preliminary task of partitioning the sequence into monotone subsequences, and it takes $O(n^{1.5})$ time. However, the physical ordering of bins is much slower: our conveyor system moves the bins by at most 1 module/second speed. If we had, say, 1 million bins (which is not realistic in practice), it would take days to order them even when using much faster conveyor belts and modules, so the time needed for the partitioning is negligible compared to this time frame.

## References

[1] Boysen, Nils and Emde, Simon. The parallel stack loading problem to minimize blockages. *European Journal of Operational Research*, 249(2):618–627, 2016. DOI: 10.1016/j.ejor.2015.09.033.

[2] Brandstädt, Andreas and Kratsch, Dieter. On partitions of permutations into increasing and decreasing subsequences. *Elektronische Informationsverarbeitung und Kybernetik*, 22(5/6):263–273, 1986.

[3] Erdős, Pál and Szekeres, György. A combinatorial problem in geometry. *Compositio Mathematica*, 2:463–470, 1935.

[4] Haneyah, S.W.A., Schutten, J.M.J., Schuur, P.C., and Zijm, W.H.M. Generic planning and control of automated material handling systems. *Computers in Industry*, 64(3):177–190, 2013. DOI: 10.1016/j.compind.2012.11.003.

[5] Yang, Bing, Chen, Jing, Lu, En-Yue, and Zheng, Si-Qing. Design and performance evaluation of sequence partition algorithms. *Journal of Computer Science and Technology*, 23(5):711–718, 2008. DOI: 10.1007/s11390-008-9183-2.

[6] Yehuda, Reuven Bar and Fogel, Sergio. Partitioning a sequence into few monotone subsequences. *Acta Informatica*, 35(5):421–440, 1998. DOI: 10.1007/s002360050126.

*Received 22nd March 2019*