Sorting Colored Balls in Colored Tubes

Ernst Althaus[0000−0002−2122−9520], Markus Blumenstock[0000−0003−3862−5922], Nick Johannes Peter Rassau[0009−0003−9896−9847], Felix Schuhknecht[0000−0002−0165−4116], and Anton Quentin Zimdars[0009−0005−0356−0008]

Johannes Gutenberg-Universität Mainz, 55128 Mainz, Germany
`{ernst.althaus,blumenstock,rassau,schuhknecht}@uni-mainz.de`

Abstract. We consider a game that was played in a German television show that is similar to the sorting balls puzzle. In it, we are assumed to move one colored ball after another in a set of colored tubes so that in the end, each ball is in the tube of its color. We are allowed to use one additional (uncolored) tube. We show general properties for solvability and that the problem of minimizing the number of moves is NP-hard, which is done by a reduction from the Feedback Arc Set Problem. Furthermore, we give an implementation of an algorithm to compute such a minimal sequence of moves. The algorithm is based on breadth-first search and accelerated by a lower bound on the number of moves from the current configuration to the final one that is obtained by solving a small instance of the Feedback Arc Set Problem. Our experiments show that instances with 7 colored tubes of height 4 can be solved in a reasonable amount of time and that the number of tubes is much more critical for the running time than the heights of the tubes.

1 Introduction

We consider an abstraction from a game played in a German television show, more precisely in “Frag doch mal die Maus”, episode 35, first shown in “Das Erste” on 20th April 2022. This game was also part of the exercises of the mathematics journal for high school students “Monoid” of the Institute of Mathematics of Johannes Gutenberg University Mainz. The game can be played at https://scratch.mit.edu/projects/691480451. Many other ball sorting apps are available, e.g., at https://play.google.com/store/search?q=ball+sorting.

In this game, one is given a set of colored tubes containing colored balls and an additional (typically empty) uncolored tube. One’s task is to find a sequence of moves such that in the end, each tube contains all balls of its color and the reserve tube is empty, where a move picks the topmost ball in one tube and puts it as the topmost ball of another tube given that the tube has space for an additional ball. In Figure 1 an example is shown.

In the show, this puzzle had to be solved as fast as possible, without developing a strategy in advance. In this paper, we consider the problem of finding a sequence to solve the puzzle with the fewest number of moves possible.

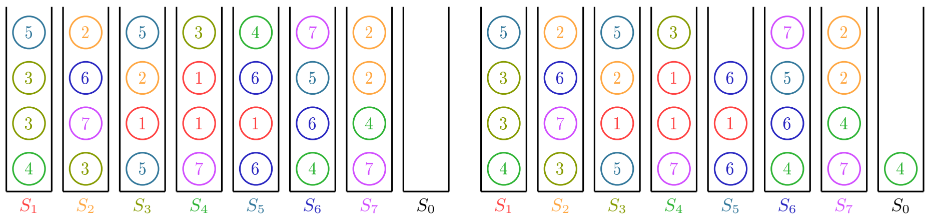

S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_0
S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_0

Fig. 1: In the upper left picture, a possible initial instance of the game is shown. As a first move, one has to pick a topmost ball from a tube and put it into the reserve-tube, as all colored tubes are full. In the upper right picture, we depict the instance that is obtained when picking the topmost ball of the fifth tube to the reserve tube.

We observe that the problem is very similar to the Feedback Arc Set Problem if the height of the reserve tube is unlimited. More precisely, we show that the problem is NP-complete by a reduction from the Feedback Arc Set Problem. This reduction can be reversed, i.e., the problem with an unlimited height of the reserve-tube can be solved by solving an appropriate instance of the Feedback Arc Set Problem. We use this reduction and the observation that smaller heights of the tubes can only increase the number of necessary moves, to obtain a lower bound on the number of necessary moves. We use lower bounds to avoid the enumeration of configurations in a breadth-first search in the configuration space. The feedback-arc-set-based lower bound is strong enough to solve puzzles with 7 tubes of height 4, which was not possible with the simpler lower bounds (sketched in Appendix C).

The rest of the paper is structured as follows. In the Section 2, we briefly give some related work. Before we show that the problem is NP-complete in Section 4, we formally define the problem and give some basic properties in Section 3. Afterwards, we describe possible lower bounds in Section 5 and the algorithm with its implementation in Section 6. Finally, in Section 7, we experimentally evaluate the algorithm in detail and conclude the paper. Additionally the Appendix contains further material that helps in understanding the concepts.

2 Related Work

A variant of the problem of sorting colored balls in tubes that are themselves colorless has been examined by Ito et al. [6], where the balls are restricted to be moved on top of balls of the same color or into empty tubes. They show this problem to be NP-complete by a reduction from the 3-Partition problem, which is NP-complete [4]. For the problem where balls may only be moved on top of balls of the same color or empty tubes, but the balls must end up in the tube of their color, a zero-knowledge proof was given by Ruangwises [10].

Another variant that is less similar to ours has tubes colored $1, \ldots, n$ with k balls colored $n - i + 1$ in tube i [12]. The goal is to move all balls of color

i into tube i for every color i. A move consists of swapping a pair of balls in adjacent tubes. An algorithm that uses a near-optimal number of moves for $k = 2$ was given by Ito et al. [5]. The problem had also been given in a contest by the German Mathematical Society and Springer Spektrum, but the submissions were not published in academic journals.

Another similar problem that has practical applications are containers in container terminals, which are stacked upon each other. To cast our ball sorting problem as a simple container problem, the containers that are to be loaded onto the same ship share a color and are to be put on top of each other in a designated stack of this color. In real-world applications, several objectives such as efficient use of storage space, avoidance of unproductive moves, and scheduling constraints are crucial [2]. In contrast to our problem, the order of the containers in the stack becomes important if the ship has a stowage plan. Stacking problems have been investigated in many variants, and several different methods have been used to solve them. For a survey see [9].

3 Definitions and Basic Properties

We consider the set $C = \{1, \dots, c\}$ of colors for some fixed $c \in \mathbb{N}$. A tube of height $h \in \mathbb{N}$ has a single color and is a carrier for a sequence of at most h balls. Every ball has a single color. A configuration S of a tube of height h is a sequence $(b_1, \dots, b_\ell)$ of colors with $\ell \leq h$, i.e., the tube contains ℓ balls with colors $b_1, \dots, b_\ell$ from bottom to top. We denote the length ℓ of the sequence by $|S|$.

A tube-rack $(T_0, T_1, \dots, T_c)$ with heights $H = (h_0, \dots, h_c)$ is a set of $c+1$ tubes of heights $(h_0, \dots, h_c)$. H is called the height profile, T_0 the reserve tube. A configuration of a tube rack with height profile H is a set of $c+1$ configurations $S = (S_0, \dots, S_c)$ for the tubes of a tube-rack with $|S_i| \leq h_i$.

For a configuration $S = (S_0, \dots, S_c)$, the move (i, j) is valid if $|S_i| \geq 1$ and $|S_j| < h_j$. The successor configuration is then obtained by removing the uppermost ball from S_i and adding it to S_j, i.e., if $S_i = (b_1, \dots, b_\ell)$ and $S_j = (\tilde{b}_1, \dots, \tilde{b}_{\tilde{\ell}})$, the new configuration is $S' = (S'_0, \dots, S'_k)$ with $S'_i = (b_1, \dots, b_{\ell-1})$, $S'_j = (\tilde{b}_1, \dots, \tilde{b}_{\tilde{\ell}}, b_\ell)$ and $S'_k = S_k$ for $k \notin \{i, j\}$.

A final configuration is a configuration $S = (S_0, \dots, S_c)$ with $S_0 = ()$ and $S_i = (i, \dots, i)$ for $1 \leq i \leq c$. Given a configuration of a tube-rack with heights $(h_0, \dots, h_c)$, we are interested in the minimum number of valid moves to reach a final configuration.

Definition 1. *Given a height profile H, a configuration S for the tube-rack with height profile H and an integer k, the Sorting Colored Balls Into Tubes Problem (SCBT) asks whether there is a sequence of at most k valid moves that brings S into a final configuration. We denote an instance of the SCBT by (H, S, k).*

In the television show the game was played with the following restriction, which we also consider from here on.

Definition 2. *Restricted SCBT (RSCBT): We assume that the height profile is $H = (h, \dots, h)$ for some $h \in \mathbb{N}$ and the given configuration $S = (S_0, \dots, S_c)$ satisfies $|S_0| = 0$ and $|S_i| = h$ for $1 \leq i \leq c$.*

Clearly, the RSCBT can only have a solution if the total number of balls of color i is h for all $1 \leq i \leq c$. We say that a ball b of color i is in final position if it is in tube i and all balls below it also have color i. Notice that if the reserve tube has smaller height than the highest colored tube, there are instances that are not solvable, as it is not possible to move a ball deeper in a tube than the height of the reserve tube, if all tubes except the reserve tube are full in the beginning.

Lemma 1. *The (R)SCBT has the following properties.*

1. *Each instance $((h, \dots, h), S, c \cdot h \cdot (2h + 1))$ of the RSCBT has a solution if the total number of balls of color i is h for all $1 \leq i \leq c$.*
2. *If the instance (H, S, k) of the SCBT has a solution and $H' \geq H$, then (H', S, k) has a solution, where $H' \geq H$ means the component-wise greater-or-equal.*
3. *Let $H = (\infty, \dots, \infty)$. If the instance (H, S, k) of the SCBT has a solution, there is a solution such that a ball that is in final position is not moved and any other ball is either moved exactly once into its final position or it is moved once into the reserve tube and once into its correct tube.*
   In particular, a ball that is in its correct tube is moved exactly twice if a ball of another color is below it and is not moved if all balls below it are from its color, i.e., if it is in final position.
4. *If the instance $((\infty, \dots, \infty), S, k)$ of the SCBT has a solution and $H = (\infty, h_1, \dots, h_c)$ with $h_i \geq \max(|S_i|, b_i)$, for b_i being the number of balls of color i, then (H, S, k) has a solution.*

Proof. 1. We move one ball after each other to a final position, where we need at most $2h + 1$ moves per ball as follows. We always start and end with an empty reserve tube. Balls in final position are not moved. In a non-final configuration, there is at least one topmost ball b of some color k that is not in final position, as all balls below a ball in final position are in final position. If b is in a different tube k' than k, we move all balls in tube k that are not in final position to the reserve tube, move b to tube k and move all balls but the bottommost from the reserve tube to tube k, and the bottommost ball is moved to tube k'. If b is on tube k, we start by moving a topmost ball b' from a different tube $k' \neq k$ to the reserve tube and b to tube k'. Then we proceed as before. Notice that we need at most $2h$ moves in the first case and $2h + 1$ in the second.

2. This is clearly true, as a sequence of valid moves for H is a sequence of valid moves for H'.
3. Consider a sequence M of valid moves. We construct a sequence M' of valid moves with $|M'| \leq |M|$ that has the desired properties. For the construction, we first scan M. Whenever a ball b is moved for the first time, we do the

same move if it is moved to a final position in M. Otherwise, we move it to the reserve tube. Notice that during the process, all the configurations of tubes except the reserve tube of M' are subsequences of the configurations of M. Hence, if a ball is moved to a final position in M, it is moved to a final position in M'. After this construction, a tube i can only contain balls of color i for $i \geq 1$. Hence, we can move all balls in the reserve tube to a final position with one further move. Furthermore $|M'| \leq |M|$, as a ball that is not moved in M is not moved in M', a ball moved once in M is moved once in M' and all other balls are moved twice in M'.

4. The construction of 3 only moves balls of color i into tube i and only into final position.

Notice that Lemma 1 (1) implies that the RCBT is in NP as the number of necessary moves is bounded by a polynomial.

4 Hardness

We first show the NP-completeness of the SCBT-problem for $H = (\infty, \dots, \infty)$ by a reduction from the Feedback Arc Set Problem, defined as follows.

Definition 3. *Given a directed multi-graph $G = (V, E)$ and an integer k, the Feedback Arc Set (FAS) problem asks whether there is a set $E' \subseteq E$ with $|E'| \leq k$ such that $G' = (V, E \setminus E')$ is acyclic*[1]*.*

The FAS is one of the 21 problems that were shown to be NP-complete by Karp [7].

Given an instance $(G = (V, E), k)$ of the FAS-problem, we construct an instance of the SCBT-problem as follows:

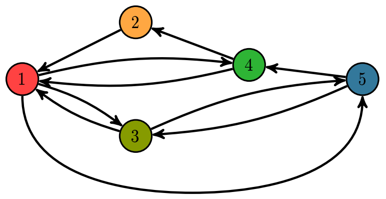
Construction 1 *For ease of notation, let $V = \{1, \dots, n\}$ and G having no loops (which are in every FAS and can be removed by modifying k accordingly). Let $c = n$, i.e., we have one tube S_u for each vertex $u \in V$. For each edge $(u, v) \in E$, we add a ball of color u in tube v (in arbitrary order). The constructed instance is $((\infty, \dots, \infty), (S_u)_{u=1}^{n}, |E| + k)$. Notice that this construction can be done in polynomial time (even in linear time).*

The construction is illustrated in Figure 2.

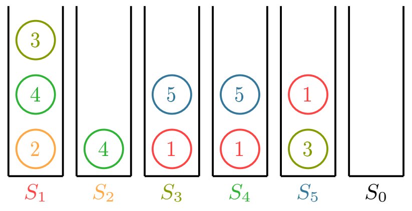
Lemma 2. *An instance (G, k) of the FAS has a solution if and only if the constructed instance of the SCBT-problem has a solution. Hence, the SCBT problem is NP-complete.*

Proof. Let $E' \subseteq E$ with $|E'| \leq k$ be a solution of the FAS. We construct a sequence of valid moves of length at most $k + |E|$ as follows. Let $\pi : \{1, \dots, n\} \to \{1, \dots, n\}$ be a topological order of the graph $(V, E \setminus E')$, i.e. a numbering of the

[1] The FAS is also NP-complete for simple directed graphs. For the constructions considered in this paper, it is preferable to consider multi-graphs.



a The graph of a Feedback Arc Set instance.



b The constructed SCBT instance.

Fig. 2: An example for Construction 1. The graph in (a) has a feedback arc set of size $k = 4$. The constructed SCBT instance in (b) with parameter $|E| + k = 14$ can be solved within 14 moves.

vertices such that $\pi(u) \leq \pi(v)$ for each edge (u, v). For $i = 1, \ldots, n$, we remove all balls from $S_{\pi(i)}$ and either move it to the reserve tube or to final position. More precisely, the balls in $S_{\pi(i)}$ that correspond to edges in E' are moved to the reserve tube and balls corresponding to edges $(\pi(i), u) \in E \setminus E'$ to tube S_u. Notice that tube S_u was processed before tube $S_{\pi(i)}$ as π gives a topological order and hence, the ball for $(\pi(i), u)$ ends up in final position. After that, we can move all balls from the reserve tube to the tube of its color.

Now consider a sequence M of valid moves of length $|E| + k$ leading to a final configuration. By Lemma 1, we may assume that each ball is either moved directly into final position or moved once to the reserve tube and then into final position. Let E' be the set of edges corresponding to balls that are moved twice. Notice that $|E'| \leq k$. Furthermore, we can argue that $(V, E \setminus E')$ has to be acyclic. Assume $(V, E \setminus E')$ contains a cycle. As a ball corresponding to edge $(u, v) \in E \setminus E'$ can be moved to final position (in tube S_u) only after all balls originally in tube S_u were removed (as these balls are not in final position), no edge of a cycle can be moved to final position as long as its predecessor has not been moved.

Notice that we can extend the construction above even for graphs with self-loops, but have to ensure that the ball for edge (u, u) is not in final position in the beginning. Hence, for every vertex, there has to be at least one outgoing edge that is not a loop which is placed in the tube below the ball for (u, u).

Furthermore, after removing all balls that are in the tube of their color (by Lemma 1(3), we know how often they are moved), we can use the inverse of Construction 1 to transform each instance of the SCBT-problem with height profile $(\infty, \ldots, \infty)$ and an empty reserve tube to a FAS instance.

To show that the RSCBT-problem is NP-complete, we use the fact that the FAS-problem is NP-complete even for Eulerian graphs [3], i.e., graphs in which each vertex has the same indegree and outdegree[2]. For tubes of bounded height, we use the fact that the Directed Feedback Vertex Set (DFVS) Problem is NP-complete [7], even if all in- and outdegrees are at most two [4, Problem GT7,

[2] As there is some error in their notation, we restate their proof in the appendix A.

proof by the authors unpublished] and a simple reduction from the DVFS to the FAS-problem that preserves this property (see Lemma 3(2)).

Definition 4. *Given a directed graph $G = (V, E)$ and an integer k, the Directed Feedback Vertex Set (DFVS) problem asks whether there is a set $V' \subseteq V$ with $|V'| \leq k$ such that $G[V \setminus V']$ is acyclic, where $G[V \setminus V']$ is the graph induced by $V \setminus V'$.*

The following result is well-known.

Lemma 3.

1. *The DFVS-problem is NP-complete, even for graphs where every vertex has indegree and outdegree of at most two.*
2. *The FAS-problem is NP-complete, even for graphs where every vertex has indegree and outdegree of at most two.*

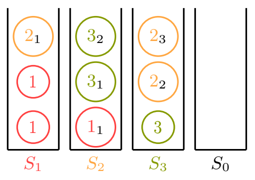
Using this result, we can show the NP-completeness of some restricted variants of the SCBT.

Lemma 4.

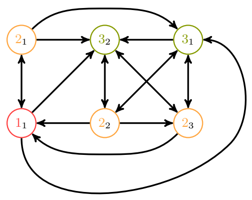
1. *The RSCBT-problem is NP-complete.*
2. *The SCBT-problem is NP-complete for the height profile $(\infty, 2, \ldots, 2)$.*
3. *The SCBT-problem is polynomial for height profiles $(h, 1, \ldots, 1)$ for all $h \in \mathbb{N}$.*

Proof.

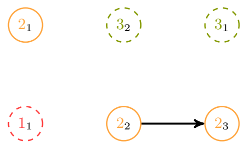
1. We reduce the FAS for Eulerian Graphs to the RSCBT-Problem. We start with the instance $((\infty, \ldots, \infty), (S_u)_{u \in V}, |E| + k)$ constructed for the SCBT-Problem from $((V, E), k)$ with Construction 1. This instance is then transformed to an instance of the RSCBT-problem by adding an appropriate number of balls with color i at the bottom of tube i. More precisely, let d be the maximum of k and any vertex degree in G. For $i \neq 0$, we add $d - |S_i|$ balls of color i at the bottom of tube S_i. Notice that this instance is valid for the RSCBT-problem and the reserve tube is high enough to store all balls corresponding to the edges of a feedback arc set if there is one of size k.
2. If we use Construction 1 for a graph with indegree and outdegree of at most two, we obtain an instance where we can reduce the height profile to $(\infty, 2 \ldots, 2)$ by Lemma 1(4).
3. Consider an instance of the SCBT-problem with height profile $(h, 1, \ldots, 1)$. This instance can be transformed to an instance of the FAS by reversing Construction 1. Each vertex of the resulting graph has indegree and outdegree of at most one and hence the graph consists of simple cycles (potentially loops), paths and isolated vertices. The minimal feedback vertex set contains exactly one edge for each cycle. The corresponding solution of the SCBT-problem as constructed in the proof of Lemma 2 can easily be modified for a reserve tube of height one as follows. We first move all balls corresponding



a Tube-rack with $h = 3$ and $c + 1 = 4$.



b Resulting graph used as input for DFVS-Solver.



c Graph after removing the nodes contained in the DFVS.

Fig. 3: (a) shows an example tube-rack configuration with $h = 4$ and $c + 1 = 4$. The big number in the balls gives its color, the subscript is an arbitrary numbering of the balls of a color to have an unique name for the corresponding vertices in the constructed graph shown in (b), where the balls in final position do not have an index as they will not have a corresponding vertex in the constructed graph. The output of the DFVS-Solver for the graph in (b) is $\{1_1, 3_1, 3_2\}$ and removing those results in (c). The lower bound is the number of vertices plus the size of the feedback vertex set, i.e., 9 in the example.

to the edges of the paths from the beginning of the path to the end to their final position. For each cycle, we move one ball to the reserve tube, handle the remaining path as before and then moving the ball from the reserve tube to the tube of its color. After that, we handle the next cycle.

5 Lower Bounds on the Number of Necessary Moves

We want to find the smallest k, such that (H, S, k) is a yes-instance by a branch-and-bound algorithm. Hence, we need lower bounds, i.e. sufficiently larger numbers k such that $(H, S, k - 1)$ is a no-instance, which are described next. Notice that we keep k in the notation of the instance although we are interested in the optimization problem.

Given an instance (H, S, k) of the SCBT, a lower bound can be obtained by computing the minimal k such that $((\infty, \ldots, \infty), S, k)$ is a yes-instance. Notice that the problem is NP-complete, but instances considered here are quite small such that we were able to solve them fast in practice. As there are many efficient implementations of the directed feedback vertex set problem (DFVS), we reduced the FAS-instance to a DFVS-instance and solved it with an existing solver. Given a graph $G = (V, E)$ for the FAS-problem, we create the graph $G' = (E, E')$ with $ee' \in E'$ if the target vertex of e is the source of e' in G, which has a DFVS of size k if and only if G has a FAS of size k (see Figure 3).

Before using this lower bound, we developed and experimentally tested other lower bounds which were outperformed by the DFVS-based lower bound, as outlined in appendix C.

6 Algorithm and Implementation

The algorithm is illustrated with pseudocode shown in Algorithm 1. For an instance of the RSCBT (H, S, k) we start a breadth-first search (BFS) with the given tube-rack configuration S (line 2). The BFS explores all possible configurations in waves and we only need to store the tube-rack configurations of the current level i, to generate the next level $i+1$ and the previous level $i-1$ to ensure that we dont go back to configurations we already visited.

To generate the next level $i+1$ from the current level i, we iterate over all tube-racks in the current level i (line 6) and for each tube-rack configuration S' we calculate the lower-bound LOWER-BOUND(S') (line 7) for required moves to reach a final configuration. Afterwards all tube-rack configurations NEIGHBORS(S') (line 10/14) that can be reached from S', by moving one ball, are added into the in-bound respectively out-of-bound set.

Let μ be an upper bound on the number of moves from the start configuration to the final one. If LOWER-BOUND$(S') + i$ is greater than μ it is not possible to reach a final configuration through S', which transfers to all tube-rack configurations NEIGHBORS$(S') \setminus$ (level $i \cup$ level $i-1$) derived from S' that were not contained in the current level i or previous level $i-1$. Those tube-rack configurations are stored in an out-of-bound set (lines $14-16$).

In the case of LOWER-BOUND$(S') + i$ being smaller or equal to μ we insert NEIGHBORS$(S') \setminus$ (level $i \cup$ level $i-1$) into an in-bound set (lines $10-12$).

If LOWER-BOUND(S') is equal to 0 for any tube-rack configuration in the current level i, we have found a final configuration within μ moves and the algorithm terminates (lines $8-9$) with TRUE.

After iterating over all tube-rack configurations of the current level i we remove all tube-rack configurations from the in-bound set that are contained in the out-of-bound set, the resulting set is the next level $i+1$ (line 17). The reason for storing the out-of-bound set and making the set difference at the end is that it might happen, that a tube-rack configuration T is derived from multiple tube-rack configurations S'_1 and S'_2 in the current level i, where LOWER-BOUND$(S'_1)+i$ is greater than μ and LOWER-BOUND$(S'_2) + i$ is smaller or equal to μ. In such a case we want to use the knowledge gained through LOWER-BOUND(S'_1) that T is not able to reach a final configuration within the remaining $\mu - i$ amount of moves. When no tube-rack configurations are remaining for the next iteration it is not possible to reach a final configuration with μ moves from S and the algorithm terminates with FALSE (lines $18-19$). If this is not the case, the sets are prepared for the next iteration $i+1$: the current level set becomes the previous level set (line 20) and the next level set becomes the current level set (line 21).

To reduce the amount of tube-rack configurations we have to proceed during the algorithm, we define an equivalence relation on tube-rack configurations and for any tube-rack configuration we store only the representative tube-rack configuration of its equivalence class. Two configurations are equivalent if they can be transformed into each other by a permutation of the colors, which is a recoloring of the balls and a reordering of the tubes accordingly. From all tube-rack configurations in an equivalence class we select the one with the lexicographically smallest bitset representation as the representative of the equivalence class. It is not necessary to generate the whole equivalence class to find the representative, as we can construct it efficiently from any member of the equivalence class.

Our implementation of the algorithm is written in C++ and available at https://gitlab.rlp.net/rassau/sorting-colored-balls.

Algorithm 1 BFS using RAM

```
Input: RSCBT $(H, S, \mu)$
 1: $previous\_level \leftarrow \emptyset$
 2: $current\_level \leftarrow \{S\}$
 3: for $i = 0, \ldots, \mu - 1$ do
 4:     $ib\_neighbors \leftarrow \emptyset$                                   ▷ in-bound neighbors
 5:     $oob\_neighbors \leftarrow \emptyset$                                 ▷ out-of-bound neighbors
 6:     for all $S' \in current\_level$ do        ▷ Distributed over threads, followed by binary merge
 7:         if $i + \text{LOWER\_BOUND}(S') \leq \mu$ then
 8:             if $\text{LOWER\_BOUND}(S') = 0$ then
 9:                 return true                                     ▷ reached a final configuration
10:             for all $neighbor \in \text{NEIGHBORS}(S')$ do
11:                 if $neighbor \notin previous\_level$
                    $\wedge\ neighbor \notin current\_level$
                    $\wedge\ neighbor \notin oob\_neighbors$ then
12:                     $ib\_neighbors \leftarrow ib\_neighbors \cup \{neighbor\}$
13:         else
14:             for all $neighbor \in \text{NEIGHBORS}(S')$ do
15:                 if $neighbor \notin previous\_level$
                    $\wedge\ neighbor \notin current\_level$ then
16:                     $oob\_neighbors \leftarrow oob\_neighbors \cup \{neighbor\}$
17:     $next\_level \leftarrow ib\_neighbors \setminus oob\_neighbors$
18:     if $next\_level = \emptyset$ then
19:         return false
20:     $previous\_level \leftarrow current\_level$
21:     $current\_level \leftarrow next\_level$
Output: If $S$ can be reconfigured to a final tube-rack configuration with at most $\mu$ moves.
```

7 Experiments

In the following, we will experimentally evaluate our parallel implementation of RSCBT. We run all of the following experiments on an Intel Core i9-12900K (Alder Lake architecture) with 8 performance cores (with SMT) and 8 efficiency cores (without SMT). The machine is equipped with 128 GB of DDR4-3200 RAM. For the disk-based execution, we use a Samsung 980 Pro M.2 SSD to store the working-set. As operating system, we use Arch Linux in version 6.7.3-arch1-1. Unless stated otherwise, we configure RSCBT to use 16 threads and keep the working-set in main memory.

For each parameter configuration $(h, c+1)$ and metric of interest, we perform ten runs on ten different randomly generated initial tube-rack configurations and report the median. If k runs of a batch did not finish (due to running out of memory), we use the maximum value we observed in the other $10 - k$ runs for computing the median.

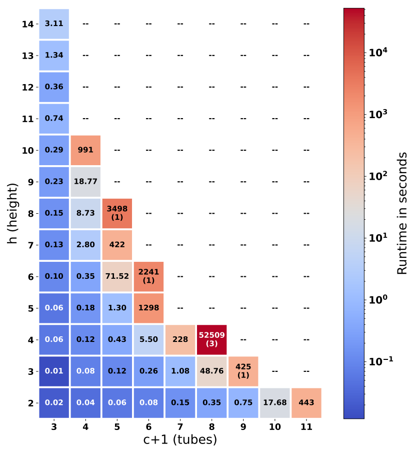
7.1 Varying Height and Number Of Colors

We begin with independently varying the height h and the amount tubes $c{+}1$ and observe its impact on the runtime, space usage, needed moves till completion, and maximum number of elements we have to maintain at a single time. Figure 4 shows the results. As mentioned above, there are some runs which did not finish due to running out of memory, we denote this with (k) in Figure 4 next to the value and if none of the ten runs completed, we show an empty box.

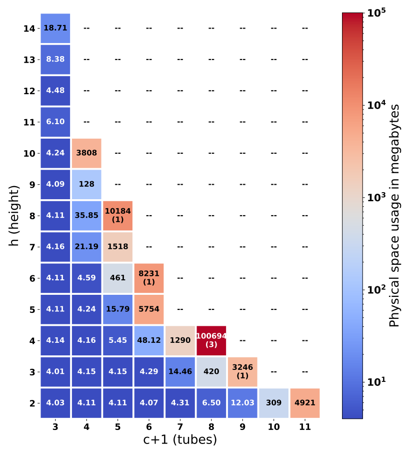
We start by inspecting the end-to-end runtime of RSCBT in Figure 4a, where we color the observed times on a logarithmic scale. We can observe that for a large number of parameter configurations, the algorithm is able to finish quickly within one second. However, we can also see that close to the border of feasibility, the runtime increases significantly. Obviously, there is a correlation between h and c, where increasing one parameter essentially limits the other one. Interestingly, increasing the amount of tubes puts more pressure on the algorithm than increasing the height h: The largest feasible height is $h = 14$, while the largest feasible amount of tubes is $c + 1 = 11$. A reason for this is that the amount N of all tube-rack configurations which are possible with a height of h and c colors can be described by $N = \frac{(h+c)!}{c!} \cdot \frac{(hc)!}{(h!)^{c+1}}$, where the numerator is symmetric for h and c, but the denominator is not (see Appendix B). The denominator is smaller for a larger c than for a larger h. This results in a larger N for a larger c than for a larger h and explains why the difficulty of the RSCBT increases differently for h and c. Two additional reason are that the lower bound we are using works better on higher tube-racks and that c determines the exponential growth rate of the levels during the BFS, because the number of colors c is equal to the minimal amount of neighbors of a tube-rack configuration.

Next, in Figure 4b and Figure 4c, we look at the maximum amount of physical memory in use by the process as well as the maximum number of elements kept in memory at a time. As we can see, they strongly correlate and also significantly increase right at the border of feasibility. Because c determines the exponential growth rate of the levels during the BFS, the memory footprint increases much stronger with increases of c than with increases of h. It is also worth mentioning that the deviation of the difficulty for a single parameter configuration, i.e., a specific height and a specific amount of colors, can be quite high for different initial tube-rack configurations. As an example, for $h = 10$ and $c = 4$ a tube-rack configuration occurred with a maximum space consumption of 76.9 GB whereas for another tube-rack configuration the maximum was only 0.124 GB. Hence, it can happen that a specific parameter configurations is sufficient for solving some initial tube-rack configuration, while it is not for others.

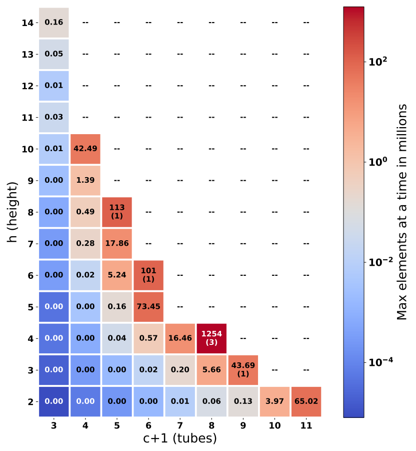
In Figure 4d, we show the number of moves needed to solve the problem for a particular parameter configuration. We can see that in contrast to the other three metrics, there is a rather linear increase in the number of required moves observable. The number of needed moves is observed to be correlated to the amount of balls, which is $h \cdot c$, contained in a certain tube-rack configuration.



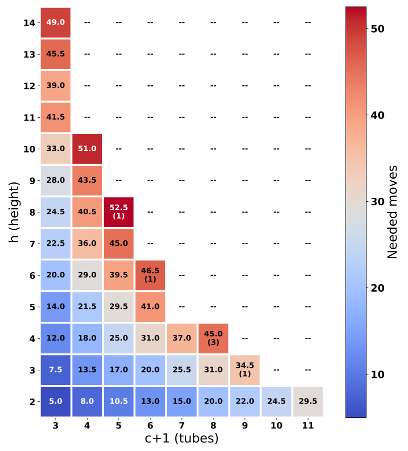
a Runtime



b Space usage



c Maximum number of elements at a time



d Number of needed moves

Fig. 4: Effect of varying the height h and the amount of tubes $c + 1$ for four different metrics. Each box shows the median of ten different initial tube-rack configurations.

8 Conclusion

We showed that the Sorting Colored Balls into Tubes problem is NP-complete and gave an algorithm that can solve instances with up to 7 colored tubes of height 4. The algorithm uses a lower bound which is based on solving a small instance of the NP-complete Feedback Vertex Set Problem. As future work, we want to investigate whether the problem remains hard when bounding the number of tubes or their heights, maybe using a parameterized complexity class. To improve the algorithm, even stronger lower bounds could be a possible key.

References

1. J. Chen, Y. Liu, and S. Lu. Directed Feedback Vertex Set Problem is FPT. In E. Demaine, G. Z. Gutin, D. Marx, and U. Stege, editors, *Structure Theory and FPT Algorithmics for Graphs, Digraphs and Hypergraphs*, volume 7281 of *Dagstuhl Seminar Proceedings (DagSemProc)*, pages 1–17, Dagstuhl, Germany, 2007. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
2. R. Dekker, P. Voogd, and E. van Asperen. Advanced methods for container stacking. *OR Spectrum*, 28(4):563–586, 2006.
3. H. Flier. *Optimization of Railway Operations.* PhD thesis, ETH Zürich, 2011.
4. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.
5. H. Ito, J. Teruyama, and Y. Yoshida. An almost optimal algorithm for Winkler's sorting pairs in bins. *Progress in Informatics*, (9):3–7, 2012.
6. T. Ito, J. Kawahara, S. Minato, Y. Otachi, T. Saitoh, A. Suzuki, R. Uehara, T. Uno, K. Yamanaka, and R. Yoshinaka. Sorting balls and water: Equivalence and computational complexity. *Theor. Comput. Sci.*, 978:114158, 2023.
7. R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.
8. I. Razgon and B. O'Sullivan. Directed Feedback Vertex Set is Fixed-Parameter Tractable. In E. Demaine, G. Z. Gutin, D. Marx, and U. Stege, editors, *Structure Theory and FPT Algorithmics for Graphs, Digraphs and Hypergraphs*, volume 7281 of *Dagstuhl Seminar Proceedings (DagSemProc)*, pages 1–14, Dagstuhl, Germany, 2007. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
9. I. Rekik and S. Elkosantini. A survey on container stacking problems based on a conceptual classification scheme: limitations and future trends. *Asian Journal of Management Science and Applications*, 7(1):23–58, 2022.
10. S. Ruangwises. Physical zero-knowledge proof for ball sort puzzle. In G. Della Vedova, B. Dundua, S. Lempp, and F. Manea, editors, *Unity of Logic and Computation*, pages 246–257, Cham, 2023. Springer Nature Switzerland.
11. P. Tittmann. *Abzählen von Objekten*, pages 1–35. Springer Berlin Heidelberg, Berlin, Heidelberg, 2019.
12. P. Winkler. *Mathematical Puzzles: A Connoisseur's Collection.* Ak Peters Series. Taylor & Francis, 2003.

A Corrected Proof that FAS is NP-complete on Eulerian Graphs

Definition 5. *Let $G = (V, E)$ be a directed graph. Let $<_L$ be a linear order of its vertices V. If $u <_L v$, an edge (u, v) is forward and otherwise backward (a loop (u, u) is backward).*

For a vertex $u \in V$ and its direct successor $v \in V$ in $<_L$, we define the u-v-cut

$$\delta(u, v) := \delta^+(u, v) \cup \delta^-(u, v),$$

which consists of the forward and backward edges crossing the cut,

$$\delta^+(u, v) := \{(x, y) \in E \mid x \leq_L u <_L v \leq_L y\},$$
$$\delta^-(u, v) := \{(x, y) \in E \mid y \leq_L u <_L v \leq_L x\}.$$

If G is a directed graph (possibly with loops), there is a linear order of G with at most k backward edges if and only if there is a feedback arc set of G with at most k edges. This is due to the fact that a directed graph is acyclic if and only if it can be topologically ordered. Therefore, every FAS can be identified with such an order.

FAS in directed graphs is NP-complete [4]. We show NP-completeness if the graph is restricted to be Eulerian, i.e., in every vertex, the indegree equals the outdegree. In order to prove this, we need two lemmata. The first lemma, claimed by Flier without proof, fails if one uses his original definition of a u-v-cut, which allows u and v to be non-consecutive.[3]

Lemma 5. *Let $u <_L v$ be two arbitrary consecutive vertices in a minimum FAS of $G = (V, E)$. Then $|\delta^-(u, v)| \leq |\delta^+(u, v)|$.*

Proof. For the sake of contradiction, suppose that there is a u-v-cut with $|\delta^-(u, v)| > |\delta^+(u, v)|$ in a minimum FAS order. Let the vertices of V be in order $(v_1, \ldots, v_i = u, v = v_{i+1}, \ldots, v_n)$. Move all vertices up to u to the end to obtain the order $(v = v_{i+1}, \ldots, v_n, v_1, \ldots, v_i = u)$. This is an FAS with less backward edges, which contradicts minimality: The edges between the vertices $v_{i+1}, \ldots, v_n$ and the edges between the vertices $v_1, \ldots, v_i$ keep their forward/backward direction. The remaining edges are exactly the set $\delta(u, v)$ in the original order. The forward edges $\delta^+(u, v)$ become backward edges, and the backward edges $\delta^-(u, v)$ become forward edges. By our assumption, the number of backward edges decreases.

Flier uses the following construction for his NP-hardness proof.

Construction 2 *Let $G = (V, E)$, $k \in \mathbb{N}$ be the FAS instance. Let $k' = k + |E|$. Create a new vertex $v_0 \notin V$. Let $E' = E$ initially. For every edge $(x, y) \in E$, create an edge (v_0, x) and an edge (y, v_0) in E'. The resulting graph $G' =$*

[3] A counterexample is a directed cycle on three vertices, where the first vertex of an optimum order is chosen as u and the last vertex is chosen as v. This u-v-cut is crossed by the single backward edge and none of the two forward edges.

$(V \cup \{v_0\}, E')$ is Eulerian: v_0 has exactly $|E|$ ingoing and $|E|$ outgoing edges, and every vertex $v \neq v_0$ receives an additional $\text{outdeg}_G(v)$ *ingoing edges and* $\text{indeg}_G(v)$ *outgoing edges. The construction of instance (G', k') can be performed in polynomial time.*

Lemma 6. *Let $<_L$ be an FAS order of $G' = (V \cup \{v_0\}, E')$ from Construction 2. Then*

1. *If v_0 is the first or last vertex in the order, it contributes exactly $|E|$ backward edges.*
2. *If v_0 is neither the first nor the last vertex in $<_L$, let u be the predecessor and v be the successor of v_0. Then v_0 contributes exactly $|E| + |\delta^+(u,v)| - |\delta^-(u,v)|$ backward edges.*

Proof. 1. If v_0 is the first vertex in $<_L$, then for each $(x,y) \in E$, the edge (v_0, x) is a forward edge, and the edge (y, v_0) is a backward edge. Therefore, v_0 contributes exactly $|E|$ forward and $|E|$ backward edges. If v_0 is placed last, the same holds with opposite directions.

2. If v_0 is neither the first nor the last vertex in $<_L$, let u be the predecessor and and v the successor of v_0. We assign the edges of v_0 to the edges of E as in the construction: (v_0, x) and (y, v_0) are assigned to edge $(x,y) \in E$. We think of v_0 as contributing backward edges with a baseline of $+1$ per edge $(x,y) \in E$. Only if v_0 is between x and y, it contributes 0 or $+2$. We show that the contribution is

$$c(x,y) = \begin{cases} +2, & \text{if } (x,y) \in \delta^+(u,v), \\ +1, & \text{if } (x,y) \notin \delta(u,v), \\ +0, & \text{if } (x,y) \in \delta^-(u,v), \end{cases} .$$

Firstly, consider the cases in which $(x,y) \in E$ is a forward edge, i.e., $x <_L y$:

– If $x <_L y <_L v_0$, then (v_0, x) is a backward edge and (y, v_0) is a forward edge.
– If $v_0 <_L x <_L y$, then (v_0, x) is a forward edge and (y, v_0) is a backward edge.
– If $x <_L v_0 <_L y$, then (v_0, x) is a backward edge and (y, v_0) is a backward edge.

This means that v_0 contributes two backward edges for (x,y) if $(x,y) \in \delta^+(u,v)$, otherwise it contributes a single backward edge.

Secondly, consider the cases in which $(x,y) \in E$ is a backward edge (possibly a loop), i.e., $y \leq_L x$:

– If $y \leq_L x <_L v_0$, then (v_0, x) is a backward edge and (y, v_0) is a forward edge.
– If $v_0 <_L y \leq_L x$, then (v_0, x) is a forward edge and (y, v_0) is a backward edge.
– If $y <_L v_0 <_L x$, then (v_0, x) is a forward edge and (y, v_0) is a forward edge.

This means that v_0 contributes no backward edges for (x, y) if $(x, y) \in \delta^-(u, v)$, otherwise it contributes a single backward edge.

In summary, we have shown that c counts contributions as claimed. Therefore, the number of backward edges contributed by v_0 is exactly

$$\sum_{(x,y)\in E} c(x, y) = |E| + |\delta^+(u, v)| - |\delta^-(u, v)|. \tag{1}$$

Theorem 3. *FAS on Eulerian directed graphs is NP-complete.*

Proof. Since FAS on Eulerian directed graphs is a special case of FAS in general directed graphs, the identity function is a Karp reduction (polynomial-time many-one reduction) from the former to the latter. Since the latter is in NP, so is the former.

We prove that (G, k) is a yes-instance if and only if $(G', k + |E|)$ is a yes-instance, i.e., Construction 2 is a Karp reduction from FAS in general directed graphs to FAS in Eulerian directed graphs. Since FAS in general graphs is NP-hard, so is FAS in Eulerian directed graphs.

“$\Rightarrow$”: If (G, k) is a yes-instance, then let $<_L$ be the linear order of an FAS of at most k edges. Take this order with v_0 placed before the first vertex as a linear order for G'. By Lemma 6, v_0 contributes exactly $|E|$ backward edges, so there is an FAS with at most $k + |E|$ backward edges.

“$\Leftarrow$”: If $(G', k+|E|)$ is a yes-instance, consider a minimum FAS of G' with at most $k + |E|$ backward edges. If v_0 is placed first or last, it contributes exactly $|E|$ backward edges. By dropping v_0 and its edges from consideration, we obtain an FAS of G consisting of most k backward edges, i.e., (G, k) is a yes-instance.

If v_0 is neither placed first nor last in the minimum FAS order of G', every u-v-cut has $|\delta^-(u, v)| \leq |\delta^+(u, v)|$ by applying Lemma 5 to G'. By Lemma 6 the number of backward edges contributed by v_0 is exactly $|E|+|\delta^+(u, v)|-|\delta^-(u, v)|$.

By dropping v_0 and its edges from consideration, we obtain an FAS of G whose number of backward edges is at most

$$k + |E| - (|E| + |\delta^+(u, v)| - |\delta^-(u, v)|) = k + |\delta^-(u, v)| - |\delta^+(u, v)| \leq k,$$

i.e., (G, k) is a yes-instance.

B Amount of possible configurations

To get an insight into the amount N of distinguishable tube-rack configurations possible for a height of h and c colors, N can be calculated. For this N is divided into two factors $N = N_{\text{empty}} \cdot N_{\text{color}}$ with N_{empty} as the possibilities to place the h empty slots and N_{color} as the possibilities to place the hc colored balls. N_{empty} is equal to the possibilities to take h elements from a set of $c+1$ elements with re-insertion of the taken elements and not mentioning the order. With the usage of [11, Section 1.2.3], N_{empty} can be written as:

$$N_{\text{empty}} = \binom{h+c}{h} = \binom{h+c}{c} = \frac{(h+c)!}{h! \cdot c!}. \tag{2}$$

N_{color} is equal to take c times h balls from the set with all $c \cdot h$ balls without re-insertion and with no respect to the order.

$$N_{\text{color}} = \prod_{i=0}^{c-1} \binom{(c-i) \cdot h}{h} = \frac{(c \cdot h)!}{(h!)^c}. \tag{3}$$

By combining Equation (2) and (3), N can be written as

$$N = N_{\text{empty}} \cdot N_{\text{color}} = \frac{(h+c)!}{c!} \cdot \frac{(hc)!}{(h!)^{c+1}} \tag{4}$$

C Comparison of the Lower Bounds and Illustration of their Computation

Beside the DFVS-based lower bound, we developed, implemented and experimentally evaluated further lower bounds described next.

- Simple lower bound: By Lemma 1(4), we know how often a ball that is in its tube has to be moved. All other balls have to be moved at least once (see Fig. 5.
- Lookup-based lower bound: First, we ignore the color for a subset of the balls. From the final configuration, we do a breadth-first search and store the distance to the final configuration for all configurations (sometimes only up to a maximal distance k). Given an instance of the SCBT-problem, we can compute a lower bound by computing the minimal distance of an instance obtained by ignoring the color of some balls to the final configuration (see Fig. 6

Figure 7 compares the quality of the different lower bounds. It clearly indicates that the DFVS-based lower bound outperforms the others.

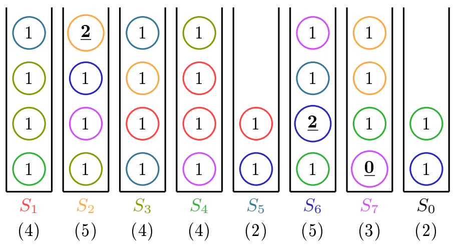
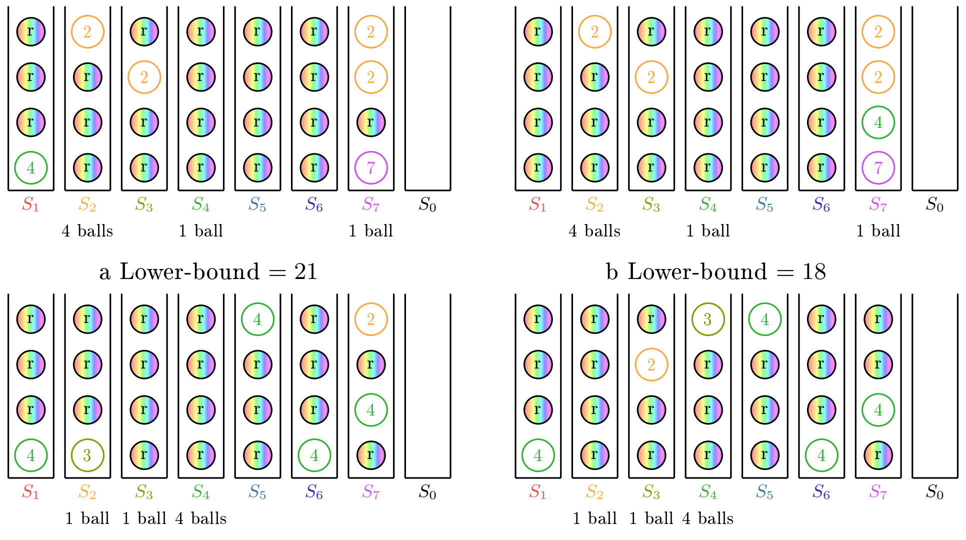


Fig. 5: Simple-lower-bound for a tube-rack configuration with 8 tubes where every tube has a height of 4. The number within a ball indicates how often it has to be moved. Adding these numbers for all balls gives the lower bound of 29.



a Lower-bound = 21

b Lower-bound = 18

c Lower-bound = 29

d Lower-bound = 30

Fig. 6: We show how to compute lower bounds if we ignore some colors of balls for the tube-rack configuration we saw in Figure 1. We assume that we have computed the distance of all configurations of balls where we ignore the color of all balls except 4 of one color and two further balls of two further colors. We call this the pattern 4110000, as we keep 4 balls of one color, 1 ball of two further colors and 0 balls of the remaining colors. In subfigures (a) and (b) we keep the color of all balls of color 2 and of one of the four balls of color 4 and 7. For those resulting tube-rack configurations we can ask a lookup table for a lower-bound. As the choice in subfigure (b) gives the lowest values over all choices, 18 is a valid lower bound. To get a valid lower-bound for the original tube-rack configuration it is necessary to take the minimum of the lower-bounds for all possible ball selections of a ball of color 4 and color 7. In subfigures (c) and (d), we keep all the color of balls of color 4 and one of color 2 and one of color 3. Again, a lower bound is obtained by considering all possible choices of a ball of color 2 and 3. As the choice in subfigure (c) gives the lowest possible value 29 is a valid lower bound. Finally the maximum over all possible color selections can be taken as lookup-based-lower-bound. In the example, choosing color 4 for the four balls and colors 2 and 3 for the further balls gives the best lower bound and hence, the look-up based lower bound for the pattern 4110000 is 29.

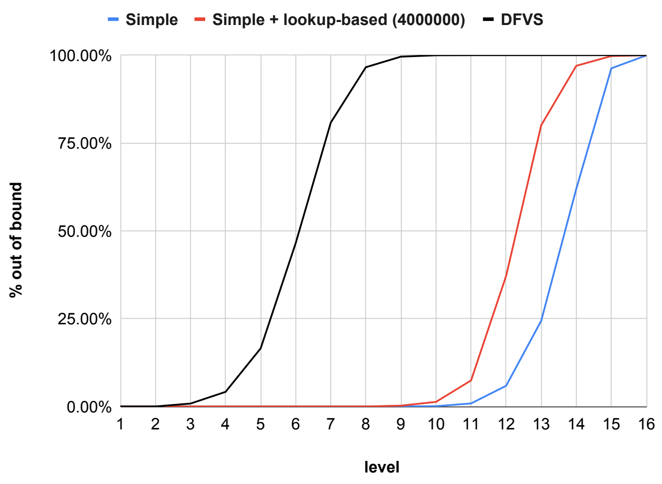


Fig. 7: Comparing of the filtering of the different lower bounds for demonstrated for the tube-rack configuration of height $h = 4$ and $c + 1 = 8$ we already saw in Figure 1. With the simple lower bound a very effective filtering starts at distance/level 15 from the starting configuration, for the lookup-based lower bound roughly one level earlier. With both lower bounds, we were not able to solve the instance. For the DFVS-based lower bound, the effective filtering starts already at level 8.