


Sorting Balls and Water: Equivalence and Computational Complexity

Takehiro Ito ✉ 

Graduate School of Information Sciences, Tohoku University, Japan

Jun Kawahara ✉ 

Kyoto University, Japan

Shin-ichi Minato ✉ 


Kyoto University, Japan

Yota Otachi ✉ 

Nagoya University, Japan

Toshiki Saitoh ✉ 

Kyushu Institute of Technology, Japan

Akira Suzuki ✉ 

Graduate School of Information Sciences, Tohoku University, Japan

Ryuhei Uehara ✉ 


Japan Advanced Institute of Science and Technology, Ishikawa, Japan

Takeaki Uno ✉

National Institute of Informatics, Tokyo, Japan

Katsuhisa Yamanaka ✉ 

Iwate University, Japan

Ryo Yoshinaka ✉ 

Graduate School of Information Sciences, Tohoku University, Japan

Abstract

Various forms of sorting problems have been studied over the years. Recently, two kinds of sorting puzzle apps are popularized. In these puzzles, we are given a set of bins filled with colored units, balls or water, and some empty bins. These puzzles allow us to move colored units from a bin to another when the colors involved match in some way or the target bin is empty. The goal of these puzzles is to sort all the color units in order. We investigate computational complexities of these puzzles. We first show that these two puzzles are essentially the same from the viewpoint of solvability. That is, an instance is sortable by ball-moves if and only if it is sortable by water-moves. We also show that every yes-instance has a solution of polynomial length, which implies that these puzzles belong to NP. We then show that these puzzles are NP-complete. For some special cases, we give polynomial-time algorithms. We finally consider the number of empty bins sufficient for making all instances solvable and give non-trivial upper and lower bounds in terms of the number of filled bins and the capacity of bins.

2012 ACM Subject Classification Mathematics of computing → Combinatorial algorithms

Keywords and phrases Ball sort puzzle, recreational mathematics, sorting pairs in bins, water sort puzzle

Digital Object Identifier 10.4230/LIPIcs.FUN.2022.16

Related Version *Previous Version:* <https://arxiv.org/abs/2202.09495>

Funding This research is partially supported by JSPS Kakenhi Grant Number JP18H04091, Japan. *Takehiro Ito:* Partially supported by JSPS KAKENHI Grant Numbers JP19K11814 and JP20H05793, Japan.

Jun Kawahara: JSPS KAKENHI Grant Number JP20H05794, Japan.

Yota Otachi: JSPS KAKENHI Grant Numbers JP18K11168, JP18K11169, JP20H05793, JP21K11752.

Toshiki Saitoh: JSPS KAKENHI Grant Number JP19K12098 and JP21H05857, Japan.

Akira Suzuki: Partially supported by JSPS KAKENHI Grant Numbers JP20K11666 and JP20H05794, Japan.

Ryuhei Uehara: JSPS KAKENHI Grant Numbers JP20H05961, JP20H05964, JP20K11673.

Katsuhisa Yamanaka: Partially supported by JSPS KAKENHI Grant Numbers 19K11812, Japan.



© Takehiro Ito, Jun Kawahara, Shin-ichi Minato, Yota Otachi, Toshiki Saitoh, Akira Suzuki, Ryuhei Uehara, Takeaki Uno, Katsuhisa Yamanaka, and Ryo Yoshinaka; licensed under Creative Commons License CC-BY 4.0

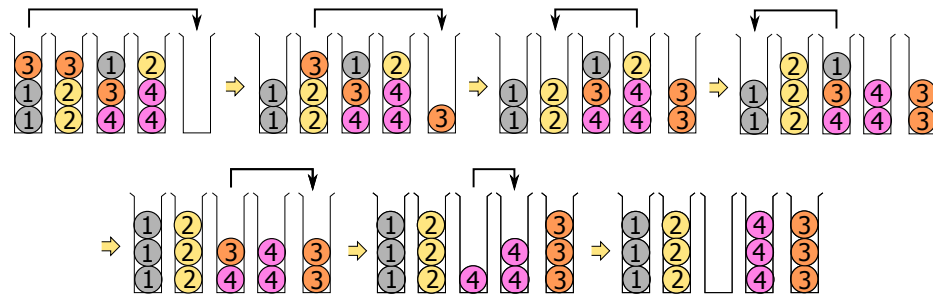
11th International Conference on Fun with Algorithms (FUN 2022).

Editors: Pierre Fraigniaud and Yushi Uno; Article No. 16; pp. 16:1–16:17

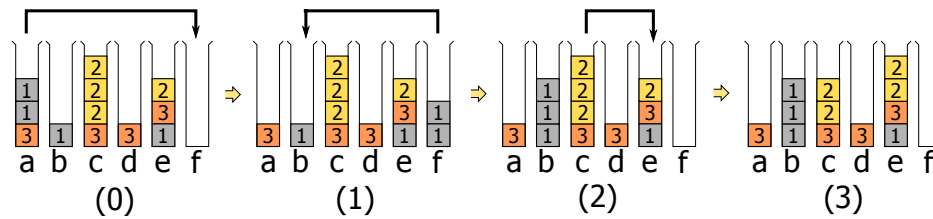
Leibniz International Proceedings in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** An example of the ball sort puzzle.



■ **Figure 2** Rules of the water sort puzzle. From the initial configuration (0), we obtain configuration (1) by moving two units of water of label 1 from a to f, configuration (2) by moving two units of water of label 1 from a to b, and configuration (3) by moving one unit of water of label 2 from c to e.

1 Introduction

The *ball sort puzzle* and the *water sort puzzle* are popularized recently via smartphone apps.¹ Both puzzles involve bins filled with some colored units (balls or water), and the goal is to somehow sort them. The most significant feature of these puzzles is that each bin works as a stack. That is, the items in a bin have to follow the “last-in first-out” (LIFO) rule.

In the ball sort puzzle, we are given hn colored balls in n bins of capacity h and k additional empty bins. For a given (unsorted) initial configuration, the goal of this puzzle is to arrange the balls in order; that is, to make each bin either empty or full with balls of the same color. (The ordering of bins does not matter in this puzzle.) The rule of this puzzle is simple: (0) Each bin works like a stack, that is, we can pick up the top ball in the bin. (1) We can move the top ball of a bin to the top of another bin if the second bin is empty or it is not full and its top ball before the move and the moved ball have the same color. An example with $h = 3$, $n = 4$, and $k = 1$ is given in Figure 1.

The water sort puzzle is similar to the ball sort puzzle. Each ball is replaced by colored water of a unit volume in the water sort puzzle. In the water sort puzzle, the rules (0) and (1) are the same as the ball sort puzzle except one liquid property: Colored water units are merged when they have the same color and they are consecutive in a bin. When we pick up a source bin and move the top water unit(s) to a target bin, the quantity of the colored water on the top of the bin to be moved varies according to the following conditions (Figure 2). If the target bin has enough margin, all the water of the same color moves to the target bin. On the other hand, a part of the water of the same color moves up to the limit of the target bin if the target bin does not have enough margin.

¹ These sort puzzles are released by several different developers. As far as the authors checked, it seems that the first one is released by IEC Global Pty Ltd in January, 2020.

In this paper, we investigate computational complexities of the ball and water sort puzzles. We are given $n + k$ bins of capacity h . In an initial configuration, n bins are full (i.e., filled with h units) and k bins are empty, where each unit has a color in the color set C . Our task is to fill n bins monochromatically, that is, we need to fill each of them with h units of the same color. We define **BSP** and **WSP** as the problems of deciding whether a given initial configuration can be reconfigured to a sorted configuration by a sequence of ball moves and water moves, respectively. We assume that instances are encoded in a reasonable way, which takes $\Theta(nh \log |C| + \log k)$ bits. Without loss of generality, we assume that each color $c \in C$ occurs exactly hj times for some positive integer j in any instance since otherwise the instance is a trivial no-instance. (This implies that $|C|$ is at most n .)

1.1 Our results

We first prove that the problems **BSP** and **WSP** are actually equivalent. By their definitions, a yes-instance of **WSP** is a yes-instance of **BSP** as well. Thus our technical contribution here is to prove the opposite direction. As a byproduct, we show that a yes-instance of the problems admits a reconfiguration sequence of length polynomial in $n + h$ as a yes-certificate. This implies that the problems belong to NP. We also show that **BSP** and **WSP** are solvable in time $O(h^n)$.

We next show that **BSP** and **WSP** are indeed NP-complete. By slightly modifying this proof, we also show that even for some kind of *trivial* yes-instances of **WSP**, it is NP-complete to find a shortest reconfiguration to a sorted configuration.

We show that if the capacity h is 2 and the number of colors is n , then **BSP** and **WSP** are polynomial-time solvable. In this case, we present an algorithm that finds shortest reconfiguration sequences for yes-instances.

We also consider the following question: how many empty bins do we need to ensure that all initial configurations can reach a sorted configuration? Observe that **BSP** and **WSP** are trivial if $k \geq hn$. It is not difficult to see that $k \geq n$ is also sufficient by using an idea based on bucket sort. By improving this idea, we show that $k \geq \lceil \frac{h-1}{h} n \rceil$ is sufficient for all instances. We also show that some instances need $k \geq \lfloor \frac{19}{64} \min\{n, h\} \rfloor$ empty bins.

1.2 Related studies

Various forms of sorting problems have been studied over the years. For example, sorting by reversals is well investigated in the contexts of sorting network [6, Sect. 5.2.2], pancake sort [2], and ladder lotteries [8]. There is another extension of sorting problem from the viewpoint of recreational mathematics defined as follows. For each $i \in \{1, \dots, n\}$, there are h balls labeled i . These hn balls are given in n bins in which each bin is of capacity h . In one step, we are allowed to swap any pair of balls. Then how many swaps do we need to sort the balls? This sorting problem was first posed by Peter Winker for the case $h = 2$ in 2004 [7], and an almost optimal algorithm for that case was given by Ito et al. in 2012 [4].

The ball/water sorting puzzles are interesting also from the viewpoints of not only just puzzles but also combinatorial reconfiguration. Recently, *reconfiguration problems* attracted attention in theoretical computer science (see, e.g., [5]). These problems arise when we need to find a step-by-step transformation between two feasible solutions of a problem such that all intermediate results are also feasible and each step abides by a fixed reconfiguration rule, that is, an adjacency relation defined on feasible solutions of the original problem. In this sense, the ball/water sort puzzles can be seen as typical implementations of the framework of reconfiguration problems, while their reconfiguration rules are non-standard.

In most reconfiguration problems, representative reconfiguration rules are *reversible*; that is, if a feasible solution A can be reconfigured to another feasible solution B , then B can be reconfigured to A as well (see e.g., the puzzles in [3]). In the ball sort puzzle, we can reverse the last move if we have two or more balls of the same color at the top of the source bin, or there is only one ball in the source bin. Otherwise, we cannot reverse the last move. That is, some moves are reversible while some moves are one-way in these puzzles. (In Figure 1, only the last move is reversible.) In the water sort puzzle, basically, we cannot separate units of water of the same color once we merge them, however, there are some exceptional cases. An example is given in Figure 2; the move from configuration (2) to configuration (3) is reversible, but the other moves are not.

2 Equivalence of balls and water

This section presents fundamental properties of BSP and WSP, from which we will conclude that

- a configuration is a yes-instance of BSP if and only if it is a yes-instance of WSP, and
- BSP and WSP both belong to NP.

2.1 Notation used in this section

Let B and C be finite sets of *bins* and *colors*, respectively. The set of sequences of colors of length at most h is denoted by $C^{\leq h}$. A sequence of $c \in C$ of length l is called *monochrome* and denoted by c^l . The empty sequence is denoted as ε , which is, in our terminology, also called monochrome. A *configuration* is a map $S: B \rightarrow C^{\leq h}$. An instance of BSP and WSP is a configuration S such that $|S(b)| \in \{h, 0\}$ for all $b \in B$. A bin b is *sorted* under S if $S(b) \in \{\varepsilon, c^h\}$ for some $c \in C$. A configuration S is *sorted* or *goal* if all bins are sorted under S . Sometimes we identify $b \in B$ with $S(b)$ when S is clear from the context.

The i th element of a sequence α is denoted by $\alpha[i]$ for $1 \leq i \leq |\alpha|$. The *top color* of a color sequence α is $\text{TC}(\alpha) = \alpha[|\alpha|]$ if α is not empty. In case α is empty, $\text{TC}(\alpha)$ is defined to be ε . A *border* in α is a non-negative integer i such that either $1 \leq i < |\alpha|$ and $\alpha[i] \neq \alpha[i+1]$ or $i = 0$, and the set of borders of α is denoted by $\mathcal{D}(\alpha)$. We count non-trivial borders under S as $D(S) = \sum_{b \in B} |\mathcal{D}(S(b)) \setminus \{0\}|$. For example, in Figure 2, the values of D are 4, 3, 3, 3 in (0), (1), (2), (3), respectively.

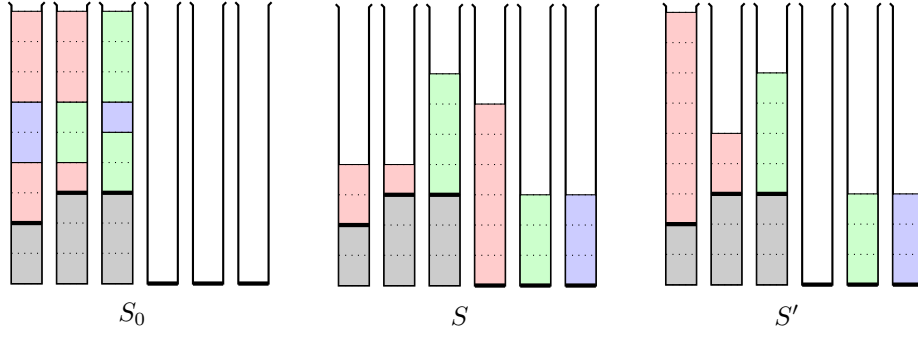
We now turn to define a move of the sort puzzles with the terminologies introduced above. Intuitively, we pick up two bins b_1 and b_2 from B , and pour the top item(s) from b_1 to b_2 if b_2 is empty or the top item of b_2 has the same color. The quantity of the items are different in the cases of balls and water. In the ball case, a unit is moved if possible. On the other hand, two or more units of water of the same color move at once until all units are moved or b_2 becomes full. We define the move more precisely below.

A *ball-move* can modify a configuration S into S' , denoted as $S \rightarrow S'$, if there are $b_1, b_2 \in B$ and $c \in C$ such that

- $S(b_1) = S'(b_1) \cdot c$, $\text{TC}(S(b_2)) \in \{c, \varepsilon\}$, $S'(b_2) = S(b_2) \cdot c$, and
- $S(b) = S'(b)$ for all $b \in B \setminus \{b_1, b_2\}$.

A *water-move* can modify S into S' , denoted as $S \Rightarrow S'$, if there are $m \geq 1$, $b_1, b_2 \in B$ and $c \in C$ such that

- $S(b_1) = S'(b_1) \cdot c^m$, $\text{TC}(S(b_2)) \in \{c, \varepsilon\}$, and $S'(b_2) = S(b_2) \cdot c^m$,
- either $\text{TC}(S'(b_1)) \neq c$ or $|S'(b_2)| = h$, and
- $S(b) = S'(b)$ for all $b \in B \setminus \{b_1, b_2\}$.



■ **Figure 3** The configuration S can be obtained from the initial configuration S_0 and can be modified into the tight one S' . The top-borders TB_S of S are shown with thick lines. This figure does not care the units below those borders. All of S_0 , S , and S' have $\mathcal{F}_{\text{red}}(TB_S) = 9$ red units, $\mathcal{F}_{\text{green}}(TB_S) = 7$ green units, and $\mathcal{F}_{\text{blue}}(TB_S) = 3$ blue units above the borders in total. The colors red, green, and blue require $\mathcal{M}_{\text{red}}(TB_S) = 0$, $\mathcal{M}_{\text{green}}(TB_S) = 1$, and $\mathcal{M}_{\text{blue}}(TB_S) = 1$ monochrome bins, respectively, which coincide with the number of the monochrome bins of respective colors in S' .

The reflexive and transitive closures of \rightarrow and \Rightarrow are denoted as \rightarrow^* and \Rightarrow^* , respectively. Clearly any single water-move can be simulated by some number of ball-moves. Thus we have $\Rightarrow^* \subseteq \rightarrow^*$.

2.2 Fundamental Theorems of BSP and WSP

Hereafter we fix an arbitrary initial configuration S_0 . Some notions introduced below are relative to S_0 , though it may not be explicit. The *top-border table* of a configuration S is a map $TB_S: B \rightarrow \{0, 1, \dots, h-1\}$ defined as $TB_S(b) = \max \mathcal{D}(S(b))$.

The main object of this section is to observe that top-border tables play an essential role in analyzing the solvability of an instance. Note that $TB_S(b) = 0$ if and only if b is monochrome. Once we have reached a configuration with $TB_S(b) = 0$ for all $b \in B$, we can achieve a goal configuration by trivial moves. Figure 3 illustrates some notions introduced in this section.

By the definition of moves, one can easily observe the following properties.

► **Observation 1.** *If $S \rightarrow^* T$, the following holds for all bins $b \in B$:*

1. *Moves monotonically remove borders from top to bottom:*

$$\mathcal{D}(T(b)) = \{i \in \mathcal{D}(S(b)) \mid i \leq TB_T(b)\};$$

2. *The contents below the top borders of T do not change:*

$$T(b)[i] = S(b)[i] \text{ for all } i \leq TB_T(b);$$

3. *If b is not monochrome in T , the colors of all the units above the border in T are the color of the unit just above that border in S :*

$$TC(T(b)) = S(b)[TB_T(b) + 1] \text{ if } TB_T(b) \neq 0.$$

We first give a necessary condition for $S_0 \rightarrow^* S$.

Recall that throughout the game play, the total number of units of each color does not change, i.e., if $S_0 \rightarrow^* S$, it holds for every color $c \in C$,

$$\sum_{b \in B} |\{i \mid 1 \leq i \leq h \text{ and } S_0(b)[i] = c\}| = \sum_{b \in B} |\{i \mid 1 \leq i \leq |S(b)| \text{ and } S(b)[i] = c\}|.$$

16:6 Sorting Balls and Water: Equivalence and Computational Complexity

Since the units under the top border of each bin have not been moved (Observation 1-2), the total numbers of units of each color c above the borders of TB_S coincide in S_0 and S . We count those units of color c as

$$\mathcal{F}_c(\text{TB}_S) = \sum_{b \in B} |\{i \mid \text{TB}_S(b) < i \leq h \text{ and } S_0(b)[i] = c\}|.$$

Those units may have been moved, but units of color c can go only to empty bins or bins whose top color is c in S . Let us partition the bins into $|C| + 1$ groups with respect to S , where

- $B_\varepsilon(\text{TB}_S) = \{b \in B \mid \text{TB}_S(b) = 0\}$: monochrome bins,
- $B_c(\text{TB}_S) = \{b \in B \setminus B_\varepsilon \mid S_0(b)[\text{TB}_S(b) + 1] = c\}$: non-monochrome bins with the top color c .

Note that if $b \notin B_\varepsilon(\text{TB}_S)$, the top color of b in S is $\text{TC}(S(b)) = S(b)[\text{TB}_S(b) + 1] = S_0(b)[\text{TB}_S(b) + 1]$. Since each bin $b \in B_c$ may have at most $h - \text{TB}_S(b)$ units of color c above the border $\text{TB}_S(b)$, there are

$$\mathcal{G}_c(\text{TB}_S) = \sum_{b \in B_c} (h - \text{TB}_S(b))$$

units of color c can be on the top layer of non-monochrome bins. Thus, we must have at least

$$\mathcal{M}_c(\text{TB}_S) = \max \left\{ 0, \left\lceil \frac{\mathcal{F}_c(\text{TB}_S) - \mathcal{G}_c(\text{TB}_S)}{h} \right\rceil \right\}$$

monochrome bins in B_ε which have some units of color c . Therefore, it is necessary that

$$\sum_{c \in C} \mathcal{M}_c(\text{TB}_S) \leq |B_\varepsilon(\text{TB}_S)|. \quad (1)$$

We remark that the values of the functions introduced above, namely \mathcal{F}_c , B_c , \mathcal{M}_c , are all determined by the top-border table TB_S . Those values coincide for S and S' if $\text{TB}_S = \text{TB}_{S'}$. Let us say that S is *consistent with S_0* if and only if TB_S satisfies Eq. (1). Moreover, S is *tight* if S has exactly $\mathcal{M}_c(\text{TB}_S)$ monochrome bins with at least one unit of every color $c \in C$.

► **Lemma 2.** *Suppose S is consistent with S_0 . There is a tight configuration S' such that $\text{TB}_S = \text{TB}_{S'}$ and $S \Rightarrow^* S'$.*

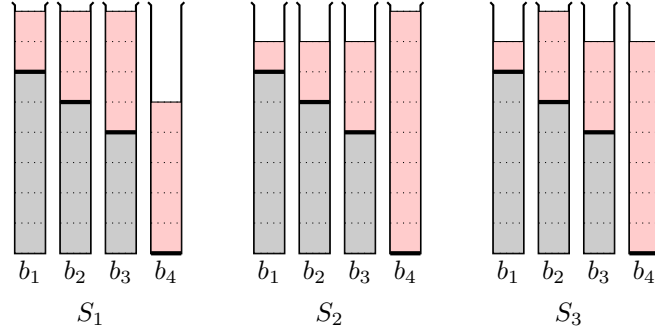
Proof. Recall that it is necessary that S has at least $\mathcal{M}_c(\text{TB}_S)$ monochrome bins of each color c . If S has more than that, one can move all the units of any of the monochrome bins of color c to other non-empty bins. The definition of \mathcal{M}_c guarantees that we can repeat this until we have just $\mathcal{M}_c(\text{TB}_S)$ monochrome bins of color c . ◀

Of course TB_S being consistent with S_0 does not imply $S_0 \rightarrow^* S$. Suppose $S_0 \rightarrow^* S \rightarrow S'$ with $\text{TB}_S \neq \text{TB}_{S'}$, where S' has one less border than S by moving a unit above the top border of some bin b to some other bins. Note that $S(b)$ is not monochrome. During the move, the bin b can keep units below its top border only. Therefore, the number of necessary monochrome bins of each color c for this move is

$$\mathcal{M}_c^b(\text{TB}_S) = \max \left\{ 0, \left\lceil \frac{\mathcal{F}_c(\text{TB}_S) - (\mathcal{G}_c(\text{TB}_S) - (h - \text{TB}_S(b)))}{h} \right\rceil \right\}$$

and it is necessary that

$$\sum_{c \in C} \mathcal{M}_c^b(\text{TB}_S) \leq |B_\varepsilon(\text{TB}_S)|.$$



■ **Figure 4** All configurations S_1 , S_2 and S_3 are tight and have the same top-border table $\tau = \text{TB}_{S_1} = \text{TB}_{S_2} = \text{TB}_{S_3}$, where $\tau(b_1) = 7$, $\tau(b_2) = 5$, and $\tau(b_3) = 4$. In those configurations, one can move the units above the top border of any of b_1 or b_2 , but it is impossible for b_3 . This solely depends on the top-border table τ and the total number $\mathcal{F}_{\text{red}}(\tau) = 16$ of red units above the top borders. It is independent of how those red units are distributed over the bins in the current configuration.

Those conditions depend on top-border configuration of S . For two top-border tables τ and τ' and a bin $b \in B$, we write $\tau \stackrel{b}{\Rightarrow} \tau'$ if

- $\tau(a) = \tau'(a)$ for all $a \in B \setminus \{b\}$,
- $\tau(b) > 0$ and $\tau'(b) = \max\{i \in \mathcal{D}(S_0(b)) \mid 0 \leq i < \tau(b)\}$,
- $\sum_{c \in C} \mathcal{M}_c^b(\tau) \leq |B_\varepsilon(\tau)|$.

We write $\tau \Rightarrow \tau'$ if $\tau \stackrel{b}{\Rightarrow} \tau'$ for some $b \in B$. Figure 4 may help intuitive understanding the above argument.

► **Theorem 3.** *Suppose a configuration S is consistent with S_0 . Then, there is a configuration T such that $S \Rightarrow^* T$ if and only if $\text{TB}_S \Rightarrow^* \text{TB}_T$, where \Rightarrow^* is the reflexive and transitive closure of \Rightarrow .*

Proof. Suppose $S \Rightarrow T$ and $\text{TB}_S \neq \text{TB}_T$. Then TB_S and TB_T must satisfy the condition for $\text{TB}_S \Rightarrow \text{TB}_T$.

Suppose $\text{TB}_S \Rightarrow \text{TB}_T$. We assume without loss of generality that S is tight by Lemma 2. The condition $\sum_{c \in C} \mathcal{M}_c^b(\tau) \leq |B_\varepsilon(\tau)|$ implies that units of color $c = S_0(b)[\text{TB}_S(b) + 1]$ in concern can be distributed to bins other than b . If $\mathcal{M}_c^b(\text{TB}_S) = \mathcal{M}_c(\text{TB}_S)$, one can move the units in b above $\text{TB}_T(b)$ to other bins whose top color is c . If $\mathcal{M}_c^b(\text{TB}_S) > \mathcal{M}_c(\text{TB}_S)$, one can move those units to an empty bin, which must be available in S , since S is tight. ◀

► **Corollary 4.** *An initial configuration S_0 is a yes-instance of BSP if and only if it is a yes-instance of WSP.*

Proof. Since any water-move can be simulated by some ball-moves, it is enough to show the converse. We claim that, for any configurations S and S' such that $\text{TB}_S = \text{TB}_{S'}$, if $S \rightarrow T$, then there is T' such that $S' \Rightarrow^* T'$ and $\text{TB}_T = \text{TB}_{T'}$.

By Theorem 3, either $\text{TB}_S = \text{TB}_T$ or $\text{TB}_S \Rightarrow \text{TB}_T$. If $\text{TB}_S = \text{TB}_T$, then $T' = S'$ proves the claim. Otherwise, the claim immediately follows from Theorem 3. Thus, if $S_0 \rightarrow^* T$ and T is a goal configuration, then one can have $S_0 \Rightarrow^* T'$ for some T' with $\text{TB}_T = \text{TB}_{T'}$. By modifying T' tight by Lemma 2, we get a goal configuration. ◀

If $\tau \stackrel{b}{\Rightarrow} \tau'$, from the values $\mathcal{F}_c(\tau)$ and $\mathcal{G}_c(\tau)$, one can compute $\mathcal{F}_c(\tau')$ and $\mathcal{G}_c(\tau')$ in constant time, by preprocessing S_0 , using the following equations:

$$\mathcal{F}_c(\tau') = \begin{cases} \mathcal{F}_c(\tau) + (\tau(b) - \tau'(b)) & \text{if } S_0(b)[\tau'(b) + 1] = c, \\ \mathcal{F}_c(\tau) & \text{otherwise,} \end{cases}$$

$$\mathcal{G}_c(\tau') = \begin{cases} \mathcal{G}_c(\tau) - (h - \tau(b)) & \text{if } S_0(b)[\tau(b) + 1] = c, \\ \mathcal{G}_c(\tau) + (h - \tau'(b)) & \text{if } S_0(b)[\tau'(b) + 1] = c \text{ and } \tau'(b) > 0, \\ \mathcal{G}_c(\tau) & \text{otherwise.} \end{cases}$$

► **Corollary 5.** *BSP and WSP belong to NP.*

Proof. By Theorem 3, S_0 is a yes-instance if and only if there is a bin sequence (b_1, \dots, b_m) with $m = D(S_0)$ ($= \sum_{b \in B} |\mathcal{D}(S_0(b)) \setminus \{0\}|$) that admits a top-border table sequence (τ_0, \dots, τ_m) such that $\tau_0 = \text{TB}_{S_0}$ and $\tau_{i-1} \stackrel{b_i}{\Rightarrow} \tau_i$ for all i . Each τ_i is uniquely determined by τ_{i-1} and b_i and one can verify $\tau_{i-1} \stackrel{b_i}{\Rightarrow} \tau_i$ in constant time, by maintaining the values of \mathcal{F}_c and \mathcal{G}_c (or just $\mathcal{F}_c - \mathcal{G}_c$). ◀

A solution for an instance is essentially an order of borders of the instance to remove.

► **Corollary 6.** *BSP and WSP can be solved in $O(h^n)$ time.*

Proof. There are at most $\prod_{b \in B} |\mathcal{D}(S_0(b))| \leq h^n$ distinct top-border tables. ◀

► **Corollary 7.** *For every yes-instance of BSP, there exists a sequence of ball-moves to a goal configuration of length at most $(2h - 1)hn$.*

Proof. In BSP, $h - 1$ ball-moves are enough to eliminate a border from a tight configuration, if there is a way to eliminate the border. To make the obtained configuration tight takes at most h moves, since the obtained configuration may have at most one monochrome bin to empty when the previous configuration is tight. Since there can be at most $(h - 1)n$ borders to eliminate, every yes-instance has a solution consisting of at most $(2h - 1)hn$ ball-moves. ◀

► **Corollary 8.** *For every yes-instance of WSP, there exists a sequence of water-moves to a goal configuration of length at most $2(h - 1)nl$ where $l = \min\{h, n\}$.*

Proof. In WSP, $l = \min\{h, n\}$ water-moves are enough to eliminate a border from a tight configuration, if there is a way to eliminate the border. To make the obtained configuration tight, it takes again l moves. To see this, observe that the obtained configuration may have at most one monochrome bin to empty as in the ball case. If one water-move does not empty the source monochrome bin, it must make the target bin full. Since one cannot make more than n full bins, to empty a monochrome bin takes at most l water-moves. Therefore, every yes-instance has a solution consisting of at most $2(h - 1)nl$ water-moves. ◀

3 NP-completeness

In this section, we show that BSP and WSP are NP-complete even with two colors. By Corollaries 4 and 5, it suffices to show that WSP with two colors is NP-hard. By slightly modifying the proof, we also show that given a trivial yes-instance of WSP, it is NP-complete to find a shortest sequence of water-moves to a sorted configuration, where an instance is trivial in the sense that it contains many (say hn) empty bins.

► **Theorem 9.** *BSP and WSP are NP-complete even with two colors.*

Proof. As mentioned above, it suffices to show that WSP with two colors is NP-hard. We prove it by a reduction from the following problem 3-PARTITION, which is known to be NP-complete even if B is bounded from above by some polynomial in m [1].

3-PARTITION

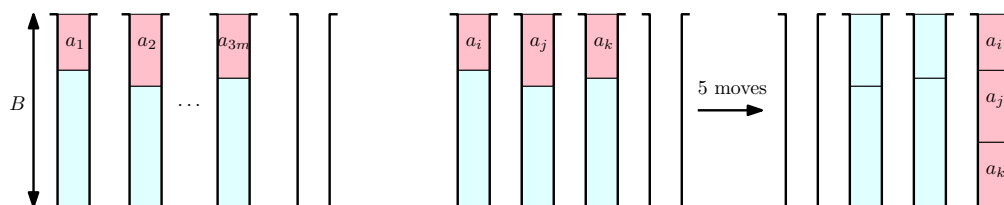
Input: Positive integers $a_1, a_2, a_3, \dots, a_{3m}$ such that $\sum_{i=1}^{3m} a_i = mB$ for some positive integer B and $B/4 < a_i < B/2$ for $1 \leq i \leq 3m$.

Question: Is there a partition of $\{1, 2, \dots, 3m\}$ into m subsets A_1, A_2, \dots, A_m such that $\sum_{i \in A_j} a_i = B$ for $1 \leq j \leq m$?

In the reduction, we use two colors red and blue. A non-empty bin is *red* (*blue*) if it contains red (blue, resp.) units only. A non-empty bin is *red-blue* if its top units are red and the other units are blue. From an instance $\langle a_1, \dots, a_{3m} \rangle$ of 3-PARTITION, we define an instance S of WSP as follows (see Figure 5):

- the capacity of each bin is B ;
- for each $i \in [3m]$, it contains a full red-blue bin b_i with a_i red units and $B - a_i$ blue units;
- it contains one empty bin.

We show that S is a yes-instance of WSP if and only if $\langle a_1, \dots, a_{3m} \rangle$ is a yes-instance of 3-PARTITION.



■ **Figure 5** The reduction from 3-PARTITION to WSP.

To show the if direction, assume that $\langle a_1, \dots, a_{3m} \rangle$ is a yes-instance of 3-PARTITION. We can construct a sequence of water-moves from S to a sorted configuration as follows. Let A_1, \dots, A_m be a partition of $\{1, \dots, 3m\}$ such that $\sum_{i \in A_j} a_i = B$ for $1 \leq j \leq m$. Using one empty bin, we can sort $b_{j_1}, b_{j_2}, b_{j_3}$ with $A_j = \{j_1, j_2, j_3\}$ as follows (see Figure 5). We first move all red units to the empty bin. Since the number of red units is $a_{j_1} + a_{j_2} + a_{j_3} = B$, the bin becomes full. Now b_{j_1}, b_{j_2} , and b_{j_3} are blue bins containing $(B - a_{j_1}) + (B - a_{j_2}) + (B - a_{j_3}) = 2B$ blue units in total. We move the units in b_{j_3} to b_{j_1} and b_{j_2} . After that, b_{j_1} and b_{j_2} become full and b_{j_3} is empty. Using this new empty bin, we can continue and sort all bins.

To show the only-if direction, we prove the following slightly modified statement. (What we need is the case of $\rho = \beta = 0$.)

Let ρ and β be non-negative integers, and S_0 be the instance of WSP obtained from S by adding ρ full red bins and β full blue bins. If S_0 is a yes-instance of the decision problem, then $\langle a_1, \dots, a_{3m} \rangle$ is a yes-instance of 3-PARTITION.

Assume that there is a sorting sequence S_0, \dots, S_ℓ , where S_ℓ is a sorted configuration. We use induction on m . (We need the dummy bins for the induction step.) The case of $m = 1$ is trivial since all instances of 3-PARTITION with $m = 1$ are yes-instances. Assume that $m \geq 2$ and that the statement holds for strictly smaller instances of 3-PARTITION.

Observe that S_0 contains $3m + \rho + \beta + 1$ bins and $(3m + \rho + \beta)B$ units of water ($(m + \rho)B$ red units and $(2m + \beta)B$ blue units). Since the capacity of bins is B , every S_i contains at most one empty bin, and if there is one, then the other bins are full. Observe also that each bin in each S_i is either red, blue, red-blue, or empty.

16:10 Sorting Balls and Water: Equivalence and Computational Complexity

We say that a move *opens* a bin b_i if the move makes b_i red units free for the first time. Observe that each red unit in b_1, \dots, b_{3m} has to move at least once since otherwise non-monochromatic bins will remain. Thus, all b_1, \dots, b_{3m} will eventually be opened. Let $b_p, b_q,$ and b_r be the first three bins opened in the sorting sequence in this order.

Assume that b_r is opened by the move from S_{j-1} to S_j . Let $\alpha = a_p + a_q + a_r$.

▷ **Claim 10.** S_j satisfies the following conditions.

1. There are $\rho + 1$ red bins, and they contain at least $\rho B + \alpha$ units.
2. There are $\beta + 3$ blue bins, and they contain $(\beta + 3)B - \alpha$ units.
3. There are $3m - 3$ red-blue bins, and they contain $(2m - 3)B + \alpha$ blue units and at most $mB - \alpha$ red units.
4. There is no empty bin. (This follows for free from the other conditions.)

Proof. Observe that we cannot create a new red-blue bin by any sequence of moves. Thus the number of red-blue bins is $3m - 3$. Since each red-blue bin b_i contains exactly $B - a_i$ blue units and at most a_i red units, the red-blue bins contain exactly $(2m - 3)B + \alpha$ blue units and at most $mB - \alpha$ red units in total.

The blue bins contain exactly $(\beta + 3)B - \alpha$ units. Since b_r contains exactly $B - a_r$ units, the other blue bins contain $(\beta + 2)B - (a_p + a_q)$ units, and thus the number of blue bins is at least $\lceil ((\beta + 2)B - (a_p + a_q))/B \rceil + 1 \geq \beta + 3$, where the inequality holds as $a_p + a_q < B$.

Since the red bins contain at least $\rho B + \alpha$ units, the number of red bins is at least $\rho + \lceil \alpha/B \rceil \geq \rho + 1$. Recall that the total number of bins is $3m + \rho + \beta + 1$. Thus, the number of red bins is exactly $\rho + 1$ and the number of blue bins is exactly $\beta + 3$. ◁

▷ **Claim 11.** $\alpha = B$.

Proof. In S_j , the $\rho + 1$ red bins contain at least $\rho B + \alpha$ units. This implies that $\alpha \leq B$. Suppose to the contrary that $\alpha < B$. We show that under this assumption, the conditions in Claim 10 cannot be violated by any sequence of moves. This contradicts that the final state S_ℓ does not contain any red-blue bin.

Let T be a configuration satisfying the conditions in Claim 10 and T' be a configuration obtained from T by one move. It suffices to show that T' still satisfies the conditions in Claim 10. Assume that the move is from a bin b to another bin b' .

First consider the case where we moved blue units. In this case, b and b' have to be blue bins. Hence, to show that all properties are satisfied, it suffices to show that b is not empty after the move. Indeed, if b becomes empty after the move, then $\beta + 2$ blue bins contain $(\beta + 3)B - \alpha > (\beta + 2)B$ units in total. This contradicts the capacity of bins.

Next consider the case where we moved red units. The bins b and b' are red or red-blue. If b becomes empty (when it was red) or blue (when it was red-blue), then the total number of red bins and red-blue bins becomes $3m + \rho - 3$ but they still contain $(3m + \rho - 3)B + \alpha$ units ($(m + \rho)B$ red and $(2m - 3)B + \alpha$ blue). This contradicts the capacity of bins. Thus, we know that the type of b does not change by the move. If the move is either from red to red, from red-blue to red-blue, or from red-blue to red, then the all conditions are satisfied. Assume that the move is from red to red-blue and that the red-blue bins contain more than $mB - \alpha$ red units after the move. Now the $3m - 3$ red-blue bins contain more than $(2m - 3)B + \alpha + mB - \alpha = (3m - 3)B$ units. This again contradicts the capacity of bins. ◁

By the claims above and the capacity of bins, S_j satisfies the following conditions.

1. There are $\rho + 1$ full red bins.
2. There are $\beta + 3$ blue bins, and they contain $(\beta + 2)B$ units.

3. There are $3m - 3$ full red-blue bins, and they contain $(2m - 2)B$ blue units and $(m - 1)B$ red units.
4. There is no empty bin.

Observe that each red-blue bin b_i in S_j is not opened so far, and thus contains $B - a_i$ blue units (and a_i red units as it is full).

Let us take a look at the sorting sequence from S_j to S_ℓ . Since there is no empty bin and all bins containing red units are full in S_j , we need to move blue units in blue bins first. Unless we make an empty bin, the situation does not change. Let S_h be the first configuration after S_j that contains an empty bin. By the capacity of bins and the discussion so far, S_h satisfies the following conditions.

1. There are $\rho + 1$ full red bins.
2. There are $\beta + 2$ full blue bins.
3. For each $i \in \{1, \dots, 3m\} \setminus \{p, q, r\}$, the bin b_i is a full red-blue bin that contains a_i red units and $B - a_i$ blue units.
4. There is one empty bin.

Without loss of generality, assume that $\{p, q, r\} = \{3m - 2, 3m - 1, 3m\}$. Let S' be the instance of WSP obtained from $a_1, \dots, a_{3(m-1)}$ by the reduction above, and S'_0 be the one obtained from S' by adding $\rho + 1$ full red bins and $\beta + 2$ full blue bins. Observe that S'_0 can be obtained from S_h by renaming the bins. Thus the sorting sequence from S_h to S_ℓ can be applied to S'_0 by appropriately renaming the bins. Therefore, we can apply the induction hypothesis to S'_0 and thus $a_1, \dots, a_{3(m-1)}$ is a yes-instance of 3-PARTITION. Since $\alpha = a_{3m-2} + a_{3m-1} + a_{3m} = B$, the instance a_1, \dots, a_{3m} is also a yes-instance of 3-PARTITION. ◀

Corollaries 7 and 8 and Theorem 9 imply that given an integer t and a configuration S , it is NP-complete to decide whether there is a sequence of length at most t from S to a sorted configuration under both settings in BSP and WSP. We observe a slightly stronger result for WSP.

► **Corollary 12.** *Given an integer t and an instance S of WSP, it is NP-complete to decide whether there is a sorting sequence for S with length at most t even if S is guaranteed to be a yes-instance.*

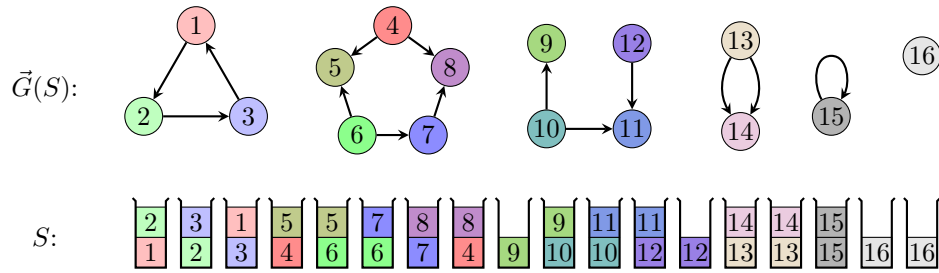
Proof. From an instance $\langle a_1, \dots, a_{3m} \rangle$ of 3-PARTITION, we first construct an instance of WSP as described in the proof of Theorem 9, and then add a sufficient number of empty bins to guarantee that the resultant instance is a yes-instance. This is always possible with a polynomial number of empty bins as we see in Section 5.² Let S denote the constructed instance. We set $t = 5m$.

The proof of Theorem 9 implies that if $\langle a_1, \dots, a_{3m} \rangle$ is a yes-instance, then S admits a sorting sequence of length $5m$. (See Figure 5.)

Conversely, assume that S admits a sorting sequence of length $5m$. Recall that each of the $3m$ red-blue bins in S contains a_i red units at the top and $B - a_i$ blue units at the bottom for some i . Since each full blue bin in the final configuration contains units from at least two original bins, we need at least one move for it. Thus we need at least $2m$ moves to make $2m$ full blue bins. This implies that we have at most $3m$ moves that involve red units. Actually, the number of such moves is exactly $3m$ since each red unit has to move at least

² It is not difficult to show that this instance only needs a constant number of bins. In fact, two empty bins are enough to perform a greedy algorithm for sorting.

16:12 **Sorting Balls and Water: Equivalence and Computational Complexity**



■ **Figure 6** Configuration S and its graph representation $\vec{G}(S)$.

once. Since $B/4 < a_i < B/2$ for all i , each of the m full red bins in the final configuration contains units from at least three original bins, and thus it needs at least three moves. If some red bin contains units from more than three original bins, then it needs at least four moves. This contradicts the assumption that we have at most $3m$ moves for red units. Thus we can conclude that each red bin in the final configuration contains units from exactly three original bins. This gives a solution to the instance of 3-PARTITION as the capacity of the bins is B . ◀

4 Polynomial-time algorithms when $h = 2$ and $|C| = n$

In this section, we focus on the special case of $h = 2$ and $|C| = n$. In popular apps, it is often the case that h is a small constant and $|C| = n$. The case $h = 2$ is the first nontrivial case in this setting. Under this setting, every ball-move is a water-move and vice versa, except for moving (a) unit(s) from a bin with two units of the same color to an empty bin, which is a vacuous move. Therefore, in this section we do not distinguish water-moves and ball-moves and simply call them moves. We prove that in this setting, all instances with $k \geq 2$ are yes-instances. We also show that we can find a shortest sorting sequence in $O(n)$ time (if any exists).

We say that a bin of capacity 2 is a *full bin* if it contains two units, and a *half bin* if it contains one unit.

► **Theorem 13.** *If $h = 2$, $|C| = n$, and $k \geq 2$, then all instances of WSP are yes-instances. Moreover, a shortest sorting sequence can be found in $O(n)$ time.*

Proof. For the first claim of the theorem, it is enough to consider the case $k = 2$. Under a configuration S , we say a color $c \in C$ is *sorted* if the two units of color c are in the same bin. For a configuration S , we define a directed multigraph $\vec{G}(S) = (V, A)$ as follows (see Figure 6). The vertex set V is the color set C . We add one directed edge from c to c' for each full bin that contains a unit of color c at the bottom and a unit of color c' at the top. That is, A consists of $S(b)$ for all full bins b . (We may add self-loops here.) For the directed multigraph $\vec{G}(S)$, we denote its underlying multigraph by $G(S) = (V, E)$, where E is a multiset obtained from A by ignoring the directions. Since $|C| = n$, each color appears twice in S . When S consists of full bins, $G(S)$ is a 2-regular graph with n edges, which means that $G(S)$ is a set of cycles. We call a self-loop a *trivial cycle*. If S also contains half bins, then $G(S)$ is a disjoint union of cycles and paths.

Let p_S denote the number of nontrivial directed cycles in $\vec{G}(S) = (V, A)$, q_S the number of vertices of indegree 2 in $\vec{G}(S) = (V, A)$, and r_S the number of vertices with no self-loop, i.e., the number of unsorted colors. We prove that if S has either two empty bins or one empty and two half bins, then a shortest sorting sequence has length $p_S + q_S + r_S$. Clearly the initial configuration, which has two empty bins, satisfies the condition.

We first prove that there exists a sorting sequence of length $p_S + q_S + r_S$. We use an induction on $p_S + q_S + r_S$. When $p_S + q_S + r_S = 0$, all colors are sorted and thus S is a goal. Now we turn to the inductive step, which consists of four cases.

- (Case 1)** Suppose S has no half bins and $q_S = 0$. Note that when S has no half bins, it has two empty bins. Then one can move the top unit of an arbitrary bin to an empty bin. This eliminates a directed cycle in the graph. Then we have one empty and two half bins. The claim follows from the induction hypothesis.
- (Case 2)** Suppose S has no half bins and there is a vertex c whose indegree is 2. By moving the two units of c to an empty bin, we use two moves, while decreasing both q_S and r_S . Then we have one empty and two half bins. The claim follows from the induction hypothesis.
- (Case 3)** Suppose S has two half bins one of which has a color of outdegree 0. Then, the other unit of that color is not below another unit in some bin. By moving that unit on top of the half bin, we can sort the color by one move. This decreases r_S by one, and does not change q_S . After this move, still we have two half bins. The claim follows from the induction hypothesis.
- (Case 4)** Otherwise, S has two half bins b and b' and both colors in the half bins have outdegree 1. Let c_0 be the color in b and (c_0, c_1, \dots, c_m) the sequence of vertices such that $(c_{i-1}, c_i) \in A$ for all i where c_m has outdegree 0. By the assumption, c_m is not the color of the unit of b' . Therefore, c_m has indegree 2, i.e., both units of c_m appear as the top units of full bins. We sort the color c_m using the empty bin. It takes two moves and decreases both r_S and q_S by one. We then sort c_{m-1}, \dots, c_0 in this order by m moves, which decreases r_S by m , where we require no additional empty bins and finally the bin b will be empty. We have one empty and two half bins. The claim follows from the induction hypothesis.

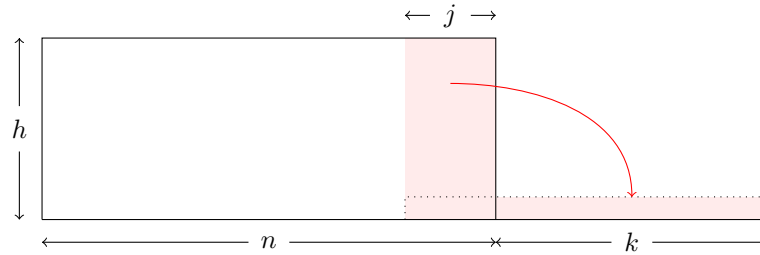
We next prove that there exists no sorting sequence of length less than $p_S + q_S + r_S$ with regardless of the number k of empty bins. Note that if all colors are sorted, $p_S + q_S + r_S = 0$. We show that any move decreases the potential by at most one.

- (Case 1)** If the color of the moved unit belongs to a directed cycle, this reduces p_S by one but not q_S . The other unit of the same color has a unit of another color on its top. Therefore this cannot reduce r_S .
- (Case 2)** If the color of the moved unit has indegree 2, this reduces q_S by one but none of p_S or r_S .
- (Case 3)** Otherwise, any other kind of moves cannot reduce p_S or q_S . Clearly one move cannot reduce r_S by two.

Since $p_S + q_S + r_S = O(n)$, we can sort any instance in $O(n)$ moves. Each feasible move can be found in $O(1)$ time, which completes the proof. ◀

► **Theorem 14.** *If $h = 2$, $|C| = n$, and $k = 1$, then WSP can be solved in $O(n)$ time. For a yes-instance, a shortest sorting sequence can be found in $O(n)$ time.*

Proof. For a configuration S , we again use the directed multigraph $\vec{G}(S) = (V, A)$ and its underlying multigraph by $G(S) = (V, E)$ defined in the proof of Theorem 13. Let S_0 be a given instance of WSP with $h = 2$, $k = 1$, and $|C| = n$. To simplify, we assume that S_0 has no full bin that contains two units of the same color without loss of generality. (Hereafter, when we have a full bin that contains two units of the same color, we remove it from the configuration and never touch. In terms of $\vec{G}(S)$, $\vec{G}(S)$ has no self-loop, and once $\vec{G}(S)$



■ **Figure 7** Refinement of the naive bucket sort.

produces a vertex with a self-loop, we remove it from \vec{G} .) Observe that if a configuration S consists of n' full bins and one empty bin, then $G(S)$ is a 2-regular multigraph with n' edges. That is, $G(S)$ is a set of cycles of size at least 2 since we remove vertices with self-loops.

Now we focus on a cycle C' in G . When C' is a directed cycle in \vec{G} (as C_3 in Figure 6), it is easy to remove the vertices in C' from G by using one empty bin. When C' contains exactly one vertex v of outdegree 2 in \vec{G} , C' contains another vertex u of indegree 2. In this case, we first move two units of color u to the empty bin, and we can sort the other bins using two half bins. Lastly, we can get two units of color v together, and obtain an empty bin again. Now we consider the case that C' contains (at least) two vertices v and v' of outdegree 2 in \vec{G} (as C_5 in Figure 6). In this case, we can observe that we eventually get stuck with two half bins of colors v and v' . That is, S is a no-instance if and only if $\vec{G}(S)$ contains at least one cycle C' that has at least two vertices of outdegree 2 in $\vec{G}(S)$.

In summary, we can conclude that the original instance S_0 can be sorted if and only if every cycle in $G(S_0)$ contains at most one vertex of outdegree 2 in $\vec{G}(S_0)$. The construction of $\vec{G}(S_0)$ and $G(S_0)$ can be done in $O(n)$ time, and this condition can be checked in $O(n)$ time. Moreover, it is easy to observe that sorting items in each cycle of length n' in $G(S_0)$ requires $n' + 1$ moves, and $n' + 1$ moves are sufficient. Therefore, the optimal sorting requires $n + \ell$ moves, where ℓ is the number of cycles in $G(S_0)$, which can be computed in $O(n)$ time. ◀

By the proofs of Theorems 13 and 14, we have the following corollary:

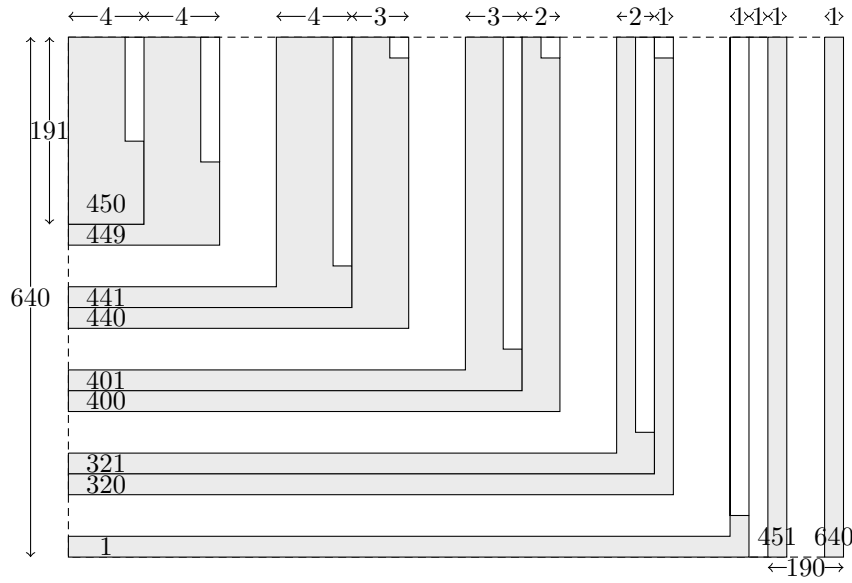
► **Corollary 15.** *If $h = 2$ and $|C| = n$, any yes-instance of WSP has a solution of length $O(n)$.*

5 The number of sufficient bins

In this section, we consider the minimum number $k(n, h)$ for n and h such that all instances of WSP with n full bins of capacity h with $k(n, h)$ empty bins are yes-instances. We can easily see that such a number exists and that $k(n, h) \leq n$ as follows. Let S be an instance with n full and k empty bins of capacity h such that $k \geq n$. For each color c used in S , let j_c be the positive integer such that S contains $j_c \cdot h$ units of color c . We assign j_c empty bins to each color c . Since $\sum_c j_c = n$, we can easily “bucket sort” S by moving water units in the n bins to empty bins following the assignment of the empty bins to the colors. On the other hand, as shown in Theorem 14, we have no-instances when $k = 1$ even if $h = 2$ (see Section 6 for a larger no-instance).

In the following, we improve the upper bound of $k(n, h)$ and show a lower bound.

► **Theorem 16.** $k(n, h) \leq \lceil \frac{h-1}{h} n \rceil$.



■ **Figure 8** When the first unit of color 1 has been moved, where $n = 640$ and $k = 190$.

Proof. It suffices to show the lemma for the case where $n = |C|$. We refine the bucket-sort base algorithm described above. See Figure 7. Suppose that we have at least $k = \lceil \frac{h-1}{h}n \rceil$ empty bins. Then, using those empty bins, one can make $j = \lfloor \frac{n}{h} \rfloor$ other bins monochrome, since $k \geq (h-1)j$. Now we have $k+j = n$ monochrome bins. When some of those bins have the same color, we merge them. Then we can dedicate different n bins to pairwise different colors, and we can use them to sort the remaining bins by the bucket sort. ◀

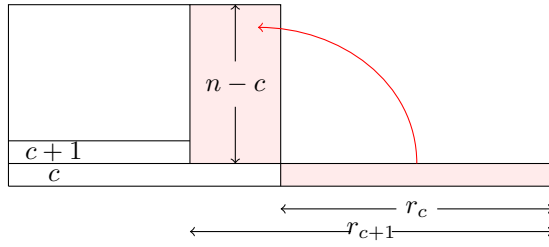
We note that $\lceil \frac{h-1}{h}n \rceil = n$ when $h > n$, which coincides with the naive bucket sort algorithm.

► **Theorem 17.** $k(n, h) \geq \lceil \frac{19}{64} \min\{n, h\} \rceil$.

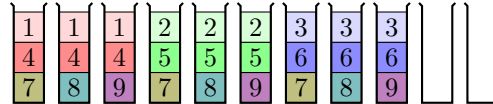
Proof. We show a configuration of $h = n$ bins where every bin has the same contents $(1, \dots, n)$. In the case where $n > h$, one can add $n - h$ extra bins whose contents are already sorted. In the case where $n < h$, one can arbitrarily pad the non-empty bins of the above configuration with $h - n$ extra units of each color at the bottom.

Suppose that it is solvable with k empty bins. We consider the very first moment when a unit of color 1 has been moved. Figure 8 shows an example configuration with $n = 640$ and $k = 190$. Note that any initially empty bin will always be monochrome. Let U be the set of colors c such that at least one unit of the color c occupies a bin which was initially empty, where $|U| \leq k$, and r_c be the number of units of color c which have been removed from the initial location. Particularly $r_1 = 1$. Then we have $r_{c+1} \geq r_c$ for all colors $c \geq 1$, since a unit of color c can be removed only after all the units above have been removed. Moreover, if $c \notin U$, units of color c which can be put only on units of the same color. That is, from both the source and target bins involved in the move, the unit of color $c + 1$ must have been removed. Each of the target bins accepts at most $(n - c)$ units of water color c on top of the unit initially located there. Therefore, for $c \notin U$, $r_c \leq (r_{c+1} - r_c)(n - c)$ (see Figure 9), i.e.,

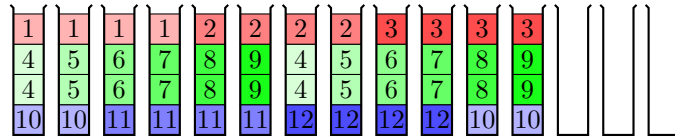
$$r_{c+1} - r_c \geq \left\lceil \frac{r_c}{n - c} \right\rceil. \tag{2}$$



■ **Figure 9** If $c \notin U$, $r_c \leq (r_{c+1} - r_c)(n - c)$.



(a)



(b)

■ **Figure 10** No-instances for (a) $h = 3, k = 2, n = 9$ and (b) $h = 4, k = 3, n = 12$.

We will give a lower bound of the size of U that admits a sequence of integers (r_1, \dots, r_n) with $1 \leq r_1 \leq \dots \leq r_n \leq n$ that satisfies Eq. (2) for $c \notin U$. Suppose that (r_1, \dots, r_n) satisfies the condition with $c \in U$ and $c_{c+1} \notin U$. Then, one can easily see that $U' = U \setminus \{c\} \cup \{c+1\}$ admits a sequence $(r_1, \dots, r_{c-1}, r'_c, r_{c+1}, r_n)$ with $r'_c = r_{c+1}$ that satisfies Eq. (2). Therefore, we may and will assume that $U = \{n - k + 1, \dots, n\}$.

For $c \leq n - k$, $r_1 = 1$ and $r_{c+1} - r_c \geq 1$ implies $r_c \geq c$. Hence, for $n/2 < c \leq n - k$,

$$r_{c+1} - r_c \geq \left\lceil \frac{r_c}{n - c} \right\rceil > \frac{n/2}{n - n/2} = 1$$

implies $r_c \geq 2c - n/2$. Hence, for $(5/8)n < c \leq n - k$,

$$r_{c+1} - r_c \geq \left\lceil \frac{r_c}{n - c} \right\rceil > \frac{(5/4)n - n/2}{n - (5/8)n} = 2$$

implies $r_c \geq 3c - (9/8)n$. Hence, for $(11/16)n < c \leq n - k$,

$$r_{c+1} - r_c \geq \left\lceil \frac{r_c}{n - c} \right\rceil > \frac{(33/16)n - (9/8)n}{n - (11/16)n} = 3$$

implies $r_c \geq 4c - (29/16)n$. On the other hand, $r_c \leq n$. That is, $c \leq n - k$ implies $c \leq (45/64)n$, i.e., $k \geq (19/64)n$. ◀

6 Concluding remarks

In this paper, we investigate the problems of solvability of ball and water sort puzzles. We show that both problems are equivalent and NP-complete. We also show that even for trivial instances of the water sort puzzle, it is NP-complete to find a shortest sorting sequence.

As discussed in Section 5, any instance can be sorted when k is large enough. Especially, as discussed in Section 4, $k = 2$ is enough for $h = 2$, while we have a no-instance for $k = 1$ and $h = 2$. On the other hand, there exist no-instances with $k = 2$ for $h = 3$ and $k = 3$ for $h = 4$ (Figure 10). We constructed them by hand and verified that they are no-instances by using a computer program. It is interesting to give the boundary of k for a given h , especially, $h = 3$, that allows us to sort any input. Does k depend on both h and n , or is it independent from n ?

Recently, a Japanese puzzle maker produces a commercial product of the ball sort puzzle.³ They extend the ball sort puzzle by introducing three different aspects as follows. (1) It contains three different capacities of bins. That is, it allows to use different size of bins. (2) The balls are not only colored, but also numbered. In some problems, the puzzle asks us to not only arrange the balls of the same color into a bin, but also they should be in order in each bin. In some other problems, the puzzle asks us to arrange the balls so that the sums of the numbers in every non-empty bins are equal. (3) It contains transparent balls. That is, these transparent balls are just used for arranging balls, and their final positions do not matter in the final state. Those extensions seems to be reasonable future work.

References

- 1 Michael R. Garey and David S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness*. Freeman, 1979.
- 2 William H. Gates and Christos H. Papadimitriou. Bounds for sorting by prefix reversal. *Discret. Math.*, 27(1):47–57, 1979. doi:10.1016/0012-365X(79)90068-2.
- 3 R. A. Hearn and E. D. Demaine. *Games, Puzzles, and Computation*. A K Peters Ltd., 2009.
- 4 Hiro Ito, Junichi Teruyama, and Yuichi Yoshida. An almost optimal algorithm for Winkler’s sorting pairs in bins. *Progress in Informatics*, 9:3–7, 2012.
- 5 Takehiro Ito, Erik D. Demaine, Nicholas J.A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412:1054–1065, 2011.
- 6 Donald E. Knuth. *The Art of Computer Programming: Sorting and Searching*, volume 3. Addison-Wesley Publishing Company, 2nd edition, 1998.
- 7 Peter Winkler. *Mathematical puzzles: A Connoisseur’s collection*, volume 143, pages 149–151. A K Peters, 2004.
- 8 Katsuhisa Yamanaka, Shin-ichi Nakano, Yasuko Matsui, Ryuhei Uehara, and Kento Nakada. Efficient Enumeration of All Ladder Lotteries and Its Application. *Theoretical Computer Science*, 411:1714–1722, 2010.

³ <https://www.hanayamatoys.co.jp/product/products/product-cat/puzzle/>