



# On multi-level apportionment

Ulrike Schmidt-Kraepelin<sup>1</sup> · Warut Suksompong<sup>2</sup> · Steven Wijaya<sup>2</sup>

Received: 23 May 2025 / Accepted: 18 October 2025  
© The Author(s) 2025

## Abstract

Apportionment refers to the well-studied problem of allocating legislative seats among parties or groups with different entitlements. We present a multi-level generalization of apportionment where the groups form a hierarchical structure, which gives rise to stronger versions of the upper and lower quota notions. We show that running Adams' method level-by-level satisfies upper quota, while running Jefferson's method or the quota method level-by-level guarantees lower quota. Moreover, we prove that both quota notions can always be fulfilled simultaneously.

**Keywords** Apportionment · Quota · House monotonicity

## 1 Introduction

The problem of *apportionment* in mathematics and politics involves distributing a set of identical resources, often legislative seats, among a number of parties, regions, or groups with differing entitlements (Balinski and Young, 2001; Pukelsheim, 2014). These entitlements are typically derived from the populations of the regions or groups, or the votes that the parties receive in an election. The difficulty of apportionment stems from the fact that only integer numbers of seats can be apportioned whereas the proportional shares according to the entitlements are rarely integers. In addition to its long history, apportionment continues to receive significant interest from researchers nowadays (Brams et al., 2019; Cembrano et al., 2022; Brill et al., 2024; Mathieu and Verdugo, 2024).

---

✉ Warut Suksompong  
warut@comp.nus.edu.sg

<sup>1</sup> TU Eindhoven, Eindhoven, The Netherlands

<sup>2</sup> National University of Singapore, Singapore, Singapore

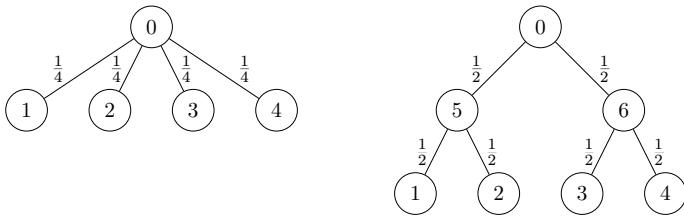
Since the exact proportional share of each group—also called the *quota*—cannot always be achieved, two natural and well-studied notions in apportionment are *upper quota* and *lower quota*.<sup>1</sup> Upper quota dictates that each group should receive at most its quota rounded up, while lower quota requires the group to obtain at least its quota rounded down. Prominent apportionment methods that fulfill at least one of the quota notions include *Adams' method*, *Jefferson's method*, and the *quota method*. Each of these methods can be formulated as an iterative procedure that allocates one seat at a time, with the recipient of each subsequent seat determined by the allocation thus far. Adams' method guarantees upper quota, while Jefferson's method ensures lower quota. The quota method is an adaptation of Jefferson's method that checks against upper quota violations, and thereby achieves both quota notions. Due to their iterative nature, all three methods satisfy an intuitive property called *house monotonicity*, which means that the number of seats that each group receives does not decrease as the total number of seats increases.<sup>2</sup>

In this paper, we study a generalized version of apportionment where instead of the groups being on a single level, they form an arbitrary hierarchical structure. This is applicable, for instance, when seats are divided among cities in a country and then among districts in each city. Ideally, we would like the quota notions to be satisfied not only for cities with respect to the country or for districts with respect to cities, but also for districts with respect to the country. Another example is when university personnel need to be allotted to faculties and further to departments within each faculty. We represent the hierarchical structure by a tree and each group by a node in the tree, and call the resulting problem *multi-level apportionment*. Note that canonical apportionment corresponds to the special case where the tree consists of a single level (besides its root). The system of *apparentment*, also known as *list alliances* (Leutgäb and Pukelsheim, 2009; Bochsler, 2010; Karpov, 2015), can be seen as two-level apportionment, as it allows groups to form a coalition and receive seats collectively in the main apportionment process, before distributing these seats among themselves. However, research on apparentment has primarily focused on the incentives for groups to form such coalitions, rather than on extending the quota notions and house monotonicity to a multi-level framework.

To illustrate the challenges that multi-level apportionment brings, consider the two trees in Fig. 1, and assume that there are six seats to be allocated. In the single-level tree on the left, each leaf node has a weight—representing its entitlement—of  $1/4$  relative to the root. Giving two seats each to nodes 1 and 2 and one seat each to nodes 3 and 4 is sufficient to satisfy both upper and lower quota. On the other hand, in the multi-level tree on the right, each leaf node again has weight  $1/4$  relative to the root, but nodes 1 and 2 belong to one group with weight  $1/2$  (represented by node 5) while nodes 3 and 4 belong to another such group (represented by node 6). If we use the same distribution of seats to nodes 1, 2, 3, 4 as before, node 5 will receive four seats

<sup>1</sup>Koriyama et al. (2013) suggested other reasons why one may want to deviate from strict proportionality.

<sup>2</sup>For a detailed overview of the theory of apportionment, we refer to Appendix A of the book by Balinski and Young (2001). In particular, the concept of satisfying upper and lower quota is called “staying within the quota” and discussed in Sect. 7 of Appendix A of that book.



**Fig. 1** Example tree structures of single-level apportionment (left) and multi-level apportionment (right)

while node 6 will receive only two. As a result, the upper quota of three for node 5 and the lower quota of three for node 6 relative to the root are both violated.

In spite of this apparent difficulty, we prove in Sect. 3 that for any multi-level apportionment instance, it is possible to distribute the seats so as to fulfill both upper and lower quota. Specifically, our notions of upper and lower quota require each node in the tree to satisfy quotas derived by comparing its seat allocation to that of *every* of its ancestors. This means that strong quota guarantees can be made regardless of the hierarchical structure and entitlements. In Sect. 4, we explore running Adams' and Jefferson's methods level-by-level and show that the resulting rules ensure upper quota and lower quota, respectively. As a consequence, either quota notion can be satisfied in conjunction with house monotonicity. In Sect. 5, we demonstrate that while an analogous extension of the quota method satisfies lower quota, the single-level check typically performed by the quota method is insufficient to guard against upper quota violations in the multi-level setting. On the other hand, modifying the check to span multiple levels, while guaranteeing upper quota, can result in a violation of lower quota. Finally, we conduct an experimental analysis in Sect. 6.

## 2 Preliminaries

An instance of multi-level apportionment consists of a rooted tree with  $n$  nodes for some positive integer  $n$ , entitlements (to be defined in the next paragraph), and a non-negative number of seats  $h$  to be allocated. Assume that the nodes are numbered  $0, 1, \dots, n - 1$ , with the root node having number 0.

For each node  $i$ , denote by  $W_i \in (0, 1]$  its *entitlement* relative to its parent (for the root node 0, we let  $W_0 = 1$ ). We assume that entitlements are normalized, i.e.,  $\sum_{c \in C(i)} W_c = 1$  holds for every non-leaf node  $i$ , where  $C(i)$  denotes the set of  $i$ 's children. Moreover, for any node  $i$ , we let  $A(i)$  be the set of  $i$ 's ancestors. Formally,  $A(0) = \{0\}$ ,<sup>3</sup> and recursively down the tree, for each node  $i$  we define  $A(i) = \{j\} \cup A(j)$ , where  $j$  is the parent of  $i$ .

The  $h$  seats need to be allocated among the nodes in the tree. Let  $V_i$  be the number of seats allocated to node  $i$ . The root node always receives all  $h$  seats, and the number of seats allocated to any non-leaf node must be equal to the total number of seats allo-

<sup>3</sup>This choice will be more convenient for our purposes than defining  $A(0) = \emptyset$ .

cated to its children. That is,  $V_0 = h$  and  $V_i = \sum_{c \in C(i)} V_c$  for each non-leaf node  $i$ . A (*multi-level*) apportionment rule maps any (*multi-level*) apportionment instance to an allocation of the seats.

For each node  $i$ , denote by  $R_i$  the entitlement of node  $i$  relative to the root node, i.e.,  $R_i := \prod_{a \in A(i) \cup \{i\}} W_a$ .

**Definition 2.1** The *lower quota* of node  $i$  is defined as  $LQ_i := \max_{a \in A(i)} \lfloor \frac{R_i}{R_a} \cdot V_a \rfloor$ , and the *upper quota* as  $UQ_i := \min_{a \in A(i)} \lceil \frac{R_i}{R_a} \cdot V_a \rceil$ .

Note that both lower and upper quota capture the quota requirements of a node with respect to every of its ancestors in the tree. An allocation of seats is said to fulfill lower (resp., upper) quota if  $V_i \geq LQ_i$  (resp.,  $V_i \leq UQ_i$ ) for every  $i$ .<sup>4</sup> An apportionment rule satisfies lower (resp., upper) quota if, for every instance, it returns a seat allocation that fulfills lower (resp., upper) quota. As an example, for the right tree in Fig. 1 with  $h = 6$ , consider an allocation with  $V_0 = 6$ ,  $V_1 = V_2 = 2$ ,  $V_3 = V_4 = 1$ ,  $V_5 = 4$ , and  $V_6 = 2$ . This allocation violates lower quota because  $LQ_6 = \lfloor \frac{1}{2} \cdot 6 \rfloor = 3 > V_6$ ; it also violates upper quota because  $UQ_5 = \lceil \frac{1}{2} \cdot 6 \rceil = 3 < V_5$ .

Finally, an apportionment rule satisfies *house monotonicity* if it has the following property: for any instance, if the total number of seats to be allocated increases from  $h$  to  $h + 1$ , then every node receives no fewer seats than before. A common way to satisfy house monotonicity is to iteratively allocate one seat at a time. For such an iterative method, we denote by  $V_i^h$  the number of seats allocated to node  $i$  after  $h$  seats have been allocated in total.

### 3 Staying within the quotas

In single-level apportionment, it is easy to see that there always exists a seat allocation that fulfills both upper and lower quota.<sup>5</sup> The situation is less clear in multi-level apportionment, however, as the internal nodes can introduce several additional constraints on the permissible allocation. Nevertheless, we show that for any instance, there exists an allocation that respects both quotas, thereby demonstrating that staying within the quotas is always possible regardless of the hierarchical structure.

**Theorem 3.1** *For every multi-level apportionment instance, there exists a seat allocation that fulfills both upper and lower quota.*

<sup>4</sup> One could define lower and upper quota with respect to only the root node, i.e.,  $\lfloor R_i \cdot h \rfloor$  and  $\lceil R_i \cdot h \rceil$ , respectively. This leads to weaker quota notions, so our positive results carry over to these notions as well. Also, our quota notions are stronger than those defined by considering each node only with respect to its parent. For example, in the instance shown in Fig. 4, the allocation described in the proof of Proposition 5.6 satisfies the latter lower quota notion but not ours.

<sup>5</sup> For example, *Hamilton's method* first gives each node its lower quota, then assigns the remaining seats to the nodes with the highest fractional remainders of their quotas until all seats have been allocated.

Theorem 3.1 is an immediate consequence of Lemmas 3.2 and 3.3, which we state and prove next. Recall that a *full binary tree* is a binary tree in which every node has either 0 or 2 children.

**Lemma 3.2** *Suppose that for every multi-level apportionment instance with a full binary tree, there exists a seat allocation that fulfills both upper and lower quota. Then, the same holds for every multi-level apportionment instance (with an arbitrary tree).*

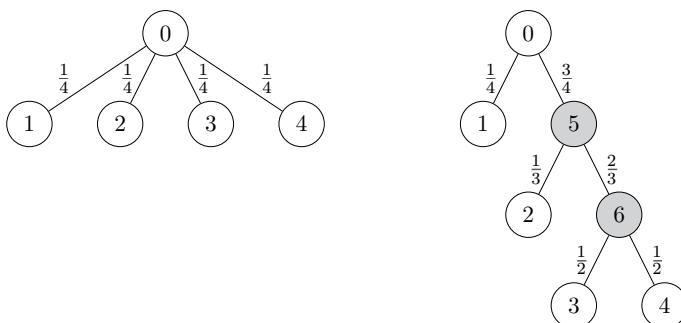
**Lemma 3.3** *For every multi-level apportionment instance with a full binary tree, there exists a seat allocation that fulfills both upper and lower quota.*

**Proof of Lemma 3.2** We claim that an arbitrary instance can be reduced to one with a full binary tree. Given an arbitrary instance, we can apply the following transformation rules for each node  $i$ :

- If  $|C(i)| = 0$  or 2, do nothing;
- If  $|C(i)| = 1$ , merge  $i$  with its only child, since the two nodes must receive the same number of seats;
- If  $|C(i)| > 2$ , introduce a new intermediate node  $j$  to become the parent of  $|C(i)| - 1$  children of  $i$ , and make  $i$  the parent of both  $j$  and the remaining child  $i^*$  of  $i$ . Set  $W_j = \sum_{c \in C(j)} W_c$ , and replace  $W_c$  by  $\frac{W_c}{1-W_{i^*}}$  for each  $c \in C(j)$ . Then, apply the same transformation rules to  $j$ .

An example of the reduction is shown in Fig. 2. Observe that all quota constraints between nodes in the original tree are still present in the new tree. Therefore, a seat allocation that fulfills both quotas in the new tree does so in the original tree as well.  $\square$

**Proof of Lemma 3.3** We proceed by induction, showing that if a satisfactory seat allocation can be found for all full binary trees with  $n$  nodes, then the same is true for all



**Fig. 2** Illustration of reducing an arbitrary tree to a full binary tree in the proof of Lemma 3.2. Shaded nodes are the introduced intermediate nodes

full binary trees with  $n + 2$  nodes (note that the number of nodes in a full binary tree is always odd). The base case  $n = 1$  holds trivially.

Consider an arbitrary instance with  $n + 2 \geq 3$  nodes. Let  $i$  be a node such that both of its children, denoted by  $x$  and  $y$ , are leaf nodes—such a node  $i$  always exists in a full binary tree when  $n > 1$ . We have  $W_x, W_y > 0$  and  $W_x + W_y = 1$ . By the induction hypothesis, in the tree with  $x$  and  $y$  removed, there exists a seat allocation (with the same number of seats) that fulfills both quotas.

For any two nodes  $c$  and  $a$  such that  $a \in A(c)$ , let  $e_{c,a} = \frac{R_c}{R_a} \cdot V_a$ , and recall that  $LQ_c = \max_{a \in A(c)} \lfloor e_{c,a} \rfloor$  and  $UQ_c = \min_{a \in A(c)} \lceil e_{c,a} \rceil$ . Fix  $c \in \{x, y\}$ , let  $e_{\max} := \max_{a \in A(c)} e_{c,a}$  and  $e_{\min} := \min_{a \in A(c)} e_{c,a}$ , and suppose that the maximum and minimum are achieved at ancestors  $a_1$  and  $a_2$  of  $c$ , respectively. We will show that  $e_{\max} - e_{\min} < 1$ . This is clear if  $a_1 = a_2$ , so assume that  $a_1 \neq a_2$ . If  $a_1$  occupies a higher position in the tree than  $a_2$ , we have

$$0 \leq e_{\max} - e_{\min} = \frac{R_c}{R_{a_1}} \cdot V_{a_1} - \frac{R_c}{R_{a_2}} \cdot V_{a_2} = \frac{R_c}{R_{a_2}} \left( \frac{R_{a_2}}{R_{a_1}} \cdot V_{a_1} - V_{a_2} \right).$$

Notice that  $\frac{R_c}{R_{a_2}} < 1$  since  $a_2$  is an ancestor of  $c$ , and  $\frac{R_{a_2}}{R_{a_1}} \cdot V_{a_1} - V_{a_2} < 1$  since lower quota for  $a_2$  with respect to  $a_1$  is satisfied. Hence,  $e_{\max} - e_{\min} < 1$  in this case. On the other hand, if  $a_2$  occupies a higher position in the tree than  $a_1$ , we have

$$0 \leq e_{\max} - e_{\min} = \frac{R_c}{R_{a_1}} \cdot V_{a_1} - \frac{R_c}{R_{a_2}} \cdot V_{a_2} = \frac{R_c}{R_{a_1}} \left( V_{a_1} - \frac{R_{a_1}}{R_{a_2}} \cdot V_{a_2} \right).$$

Notice that  $\frac{R_c}{R_{a_1}} < 1$  since  $a_1$  is an ancestor of  $c$ , and  $V_{a_1} - \frac{R_{a_1}}{R_{a_2}} \cdot V_{a_2} < 1$  since upper quota for  $a_1$  with respect to  $a_2$  is satisfied. Hence,  $e_{\max} - e_{\min} < 1$  always holds. Since  $LQ_c = \lfloor e_{\max} \rfloor$  and  $UQ_c = \lceil e_{\min} \rceil$ , it follows that  $LQ_c \leq UQ_c$ . Thus, there exists a number of seats  $V_c$  that can be allocated to node  $c$  so that both quotas are satisfied for  $c$  with respect to all of its ancestors.

To complete the proof, it remains to show that we can choose  $V_x$  and  $V_y$  so that  $V_i = V_x + V_y$ . Since  $R_x + R_y = R_i$ , it holds that  $e_{x,a} + e_{y,a} = e_{i,a}$  for each  $a \in A(i) \cup \{i\}$ . Thus,

$$\lfloor e_{x,a} \rfloor + \lfloor e_{y,a} \rfloor \leq \lfloor e_{i,a} \rfloor \leq V_i \leq \lceil e_{i,a} \rceil \leq \lceil e_{x,a} \rceil + \lceil e_{y,a} \rceil \quad (1)$$

for each  $a \in A(i) \cup \{i\}$ .

Now, for any  $c \in \{x, y\}$  and any  $a_1, a_2 \in A(c)$  (note that  $A(c) = A(i) \cup \{i\}$ ), we know that  $\frac{e_{x,a_1}}{e_{y,a_1}} = \frac{e_{x,a_2}}{e_{y,a_2}} = \frac{R_x}{R_y}$ , which means that  $e_{x,a_1} \leq e_{x,a_2}$  implies  $e_{y,a_1} \leq e_{y,a_2}$ . From the definition  $LQ_c = \max_{a \in A(c)} \lfloor e_{c,a} \rfloor$ , we have

$$LQ_x + LQ_y = \max_{a_1 \in A(c)} \lfloor e_{x,a_1} \rfloor + \max_{a_2 \in A(c)} \lfloor e_{y,a_2} \rfloor = \max_{a \in A(c)} (\lfloor e_{x,a} \rfloor + \lfloor e_{y,a} \rfloor),$$

where the second equality holds by the previous sentence. Similarly,  $UQ_x + UQ_y = \min_{a \in A(c)} (\lceil e_{x,a} \rceil + \lceil e_{y,a} \rceil)$ . Combining these with (1), we find that

$$LQ_x + LQ_y \leq V_i \leq UQ_x + UQ_y.$$

Together with the previously established claims that  $LQ_x \leq UQ_x$  and  $LQ_y \leq UQ_y$ , this implies that we can choose  $V_x$  and  $V_y$  so that  $V_i = V_x + V_y$ , as desired.  $\square$

## 4 Evaluating Adams and Jefferson

While Theorem 3.1 ensures that both upper and lower quota can be fulfilled in any multi-level apportionment instance, it does not give rise to a method that satisfies house monotonicity. A simple way to create such a method is to take an arbitrary house-monotone single-level method and run it on a multi-level instance one level at a time, starting at the root node and going down the tree—it is clear that if the total number of seats increases, no node receives fewer seats as a result. In this section, we show that if we take Adams' method (resp., Jefferson's method),<sup>6</sup> which satisfies upper quota (resp., lower quota) in single-level apportionment, the corresponding quota compliance is preserved in the multi-level setting.

We start with Adams' method, whose multi-level version is displayed as Algorithm 1. The algorithm iteratively allocates one seat at a time based on the existing allocation. We first prove an auxiliary lemma.

---

**Input :** List  $V^h$  representing the allocation for  $h$  seats  
**Output:** List  $V^{h+1}$  representing the allocation for  $h + 1$  seats

- 1  $V^{h+1} \leftarrow V^h$  // initialize  $V^{h+1}$  to be  $V^h$
- 2  $i \leftarrow 0$  // root node
- 3  $V_i^{h+1} \leftarrow V_i^h + 1$  // root node always receives the new seat
- 4 **while**  $i$  is not a leaf node **do**
- 5     Let  $c$  be a child of  $i$  with the smallest  $\frac{V_c^h}{R_c}$ , breaking ties arbitrarily
- 6      $V_c^{h+1} \leftarrow V_c^h + 1$
- 7      $i \leftarrow c$

---

**Algorithm 1** Multi-level Adams' method

**Lemma 4.1** *Let  $h \geq 1$ . In Algorithm 1, suppose that the nodes that receive the  $h$ -th seat are  $a_0, a_1, \dots, a_k$  from top to bottom, where  $a_0 = 0$ . Then,  $\frac{V_{a_j-1}^h}{R_{a_j-1}} \geq \frac{V_{a_j}^h}{R_{a_j}}$  for each  $j \in \{1, 2, \dots, k\}$ .*

---

<sup>6</sup>The iterative form of these methods that we use here is described, e.g., by Balinski and Young (2001, p. 100, Prop. 3.3).

**Proof** Fix any  $j \in \{1, 2, \dots, k\}$ , and assume for contradiction that  $\frac{V_{a_{j-1}}^h}{R_{a_{j-1}}} < \frac{V_d^h}{R_d}$  for all  $d \in C(a_{j-1})$ . Then, we have

$$V_{a_{j-1}}^h = \sum_{d \in C(a_{j-1})} V_d^h > \sum_{d \in C(a_{j-1})} R_d \cdot \frac{V_{a_{j-1}}^h}{R_{a_{j-1}}} = V_{a_{j-1}}^h \cdot \frac{\sum_{d \in C(a_{j-1})} R_d}{R_{a_{j-1}}} = V_{a_{j-1}}^h,$$

a contradiction. Hence,  $\frac{V_{a_{j-1}}^h}{R_{a_{j-1}}} \geq \frac{V_d^h}{R_d}$  for some  $d \in C(a_{j-1})$ . Since the algorithm chooses a node  $a_j \in C(a_{j-1})$  with the smallest  $\frac{V_{a_j}^h}{R_{a_j}}$ , we have  $\frac{V_{a_{j-1}}^h}{R_{a_{j-1}}} \geq \frac{V_{a_j}^h}{R_{a_j}}$ . It follows that  $\frac{V_{a_0}^h}{R_{a_0}} \geq \frac{V_{a_1}^h}{R_{a_1}} \geq \dots \geq \frac{V_{a_k}^h}{R_{a_k}}$ .  $\square$

**Theorem 4.2** *Algorithm 1 satisfies upper quota and house monotonicity in multi-level apportionment.*

**Proof** House monotonicity is clear from the iterative nature of the algorithm. We prove by induction on  $h$  that, for any  $h \geq 0$ , the allocation of  $h$  seats fulfills upper quota. The base case  $h = 0$  is trivial. Assume that  $h$  seats have been allocated for some  $h \geq 0$ , and consider the allocation of the  $(h + 1)$ -th seat. Let the nodes that receive this seat be  $a_0, a_1, \dots, a_k$  from top to bottom, where  $a_0 = 0$ . It follows from Lemma 4.1 that  $\frac{V_{a_{j-1}}^h}{R_{a_{j-1}}} \geq \frac{V_{a_j}^h}{R_{a_j}}$  for each  $j \in \{1, 2, \dots, k\}$ .

By the induction hypothesis, after the  $(h + 1)$ -th seat is allocated, the only potential upper quota violations are those for some node  $a_j$  with respect to its ancestor  $a_\ell$ , where  $\ell < j$ . From the previous paragraph, we know that  $\frac{V_{a_\ell}^h}{R_{a_\ell}} \geq \frac{V_{a_j}^h}{R_{a_j}}$ . Since  $V_{a_\ell}^{h+1} = V_{a_\ell}^h + 1$  and  $V_{a_j}^{h+1} = V_{a_j}^h + 1$ , we have  $\frac{V_{a_\ell}^{h+1}-1}{R_{a_\ell}} \geq \frac{V_{a_j}^{h+1}-1}{R_{a_j}}$ , which implies that  $V_{a_j}^{h+1} \leq \frac{R_{a_j}}{R_{a_\ell}} \cdot V_{a_\ell}^{h+1} + 1 - \frac{R_{a_j}}{R_{a_\ell}} < \frac{R_{a_j}}{R_{a_\ell}} \cdot V_{a_\ell}^{h+1} + 1$ . It follows that  $V_{a_j}^{h+1} \leq \left\lceil \frac{R_{a_j}}{R_{a_\ell}} \cdot V_{a_\ell}^{h+1} \right\rceil$ , which means that upper quota for  $a_j$  with respect to  $a_\ell$  is satisfied upon the allocation of the  $(h + 1)$ -th seat, completing the proof.  $\square$

We now turn our attention to Jefferson's method, whose multi-level version is described as Algorithm 2. The only difference from Algorithm 1 is in Line 5, where the ratio  $\frac{V_c^h+1}{R_c}$  is used instead of  $\frac{V_c^h}{R_c}$ . Again, we first establish an auxiliary lemma.

---

**Input** : List  $V^h$  representing the allocation for  $h$  seats  
**Output:** List  $V^{h+1}$  representing the allocation for  $h + 1$  seats

- 1  $V^{h+1} \leftarrow V^h$  // initialize  $V^{h+1}$  to be  $V^h$
- 2  $i \leftarrow 0$  // root node
- 3  $V_i^{h+1} \leftarrow V_i^h + 1$  // root node always receives the new seat
- 4 **while**  $i$  is not a leaf node **do**
- 5     Let  $c$  be a child of  $i$  with the smallest  $\frac{V_c^{h+1}}{R_c}$ , breaking ties arbitrarily
- 6      $V_c^{h+1} \leftarrow V_c^h + 1$
- 7      $i \leftarrow c$

---

**Algorithm 2** Multi-level Jefferson's method

**Lemma 4.3** Let  $h \geq 0$ . After  $h$  seats have been allocated by Algorithm 2, for any two nodes  $p$  and  $c$  such that  $c \in C(p)$ , it holds that  $\frac{V_c^h + 1}{R_c} \geq \frac{V_p^h + 1}{R_p}$ .

**Proof** First, we claim that whenever  $c$  has just received a seat—say, the  $g$ -th seat—it holds that  $\frac{V_c^g}{R_c} \geq \frac{V_p^g}{R_p}$ . Suppose for contradiction that  $\frac{V_c^g}{R_c} < \frac{V_p^g}{R_p}$ . We know that

$$\frac{V_p^g}{R_p} = \sum_{c' \in C(p)} \frac{V_{c'}^g}{R_p} = \sum_{c' \in C(p)} \frac{V_{c'}^g}{R_{c'}} \cdot \frac{R_{c'}}{R_p} = \sum_{c' \in C(p)} \frac{V_{c'}^g}{R_{c'}} \cdot W_{c'}.$$

Since  $\sum_{c' \in C(p)} W_{c'} = 1$  and  $\frac{V_c^g}{R_c} < \frac{V_p^g}{R_p}$ , there exists  $c^* \in C(p)$  such that  $\frac{V_{c^*}^g}{R_{c^*}} > \frac{V_p^g}{R_p} > \frac{V_c^g}{R_c}$ . Hence, just before  $c$  received this seat,  $\frac{V_{c^*}^{g-1}}{R_{c^*}} > \frac{V_c^{g-1} + 1}{R_c}$ . However, this means that the last seat allocated to  $c^*$  before this point should have been allocated to  $c$  instead, a contradiction that establishes the claim.

We now proceed to prove the lemma. Assume for contradiction that for some  $h$ , after  $h$  seats have been allocated,  $\frac{V_c^h + 1}{R_c} < \frac{V_p^h + 1}{R_p}$ . Let  $h' > h$  be the round in which  $c$  receives the next seat; it holds that  $\frac{V_c^{h'-1} + 1}{R_c} = \frac{V_c^h + 1}{R_c} < \frac{V_p^h + 1}{R_p} \leq \frac{V_p^{h'-1} + 1}{R_p}$ . Hence, after the  $h'$ -th seat has been allocated to  $c$ , we have  $\frac{V_c^{h'}}{R_c} < \frac{V_p^{h'}}{R_p}$ . However, this contradicts the claim in the previous paragraph, completing the proof.  $\square$

**Theorem 4.4** Algorithm 2 satisfies lower quota and house monotonicity in multi-level apportionment.

**Proof** House monotonicity is clear from the iterative nature of the algorithm. For lower quota, consider any pair of nodes  $a, b$  such that  $a$  is an ancestor of  $b$ . By applying Lemma 4.3 repeatedly along the path between  $a$  and  $b$ , we have that at any point

during the execution of the algorithm, it holds that  $\frac{V_b+1}{R_b} \geq \frac{V_a+1}{R_a}$ . This means that  $V_b \geq \frac{R_b}{R_a} \cdot V_a - 1 + \frac{R_b}{R_a} > \frac{R_b}{R_a} \cdot V_a - 1$ , so  $V_b \geq \left\lfloor \frac{R_b}{R_a} \cdot V_a \right\rfloor$ . Hence, lower quota for  $b$  with respect to  $a$  is satisfied.  $\square$

## 5 Applying the quota method

Given the results in Sect. 4, a natural next approach is to take an iterative method that satisfies both upper and lower quota in single-level apportionment, and apply it level-by-level in the multi-level setting. A well-known method with this property is the quota method (Balinski and Young 1975, Sect. 5), whose multi-level version is presented as Algorithm 3. The algorithm is the same as multi-level Jefferson (Algorithm 2), except that there is a check against upper quota violations with respect to the parent of each node (Line 5).

We first show that the multi-level quota method satisfies lower quota. The proof proceeds via a similar lemma as for multi-level Jefferson (Lemma 4.3), but proving the lemma is now more involved.

**Lemma 5.1** *Let  $h \geq 0$ . After  $h$  seats have been allocated by Algorithm 3, for any two nodes  $p$  and  $c$  such that  $c \in C(p)$ , it holds that  $\frac{V_c^h+1}{R_c} \geq \frac{V_p^h+1}{R_p}$ .*

**Proof** It suffices to prove that whenever  $c$  has just received a seat, it holds that  $\frac{V_c}{R_c} \geq \frac{V_p}{R_p}$ . Once we have this claim, we can finish the proof of the lemma as in the last paragraph of the proof of Lemma 4.3.

Suppose for contradiction that for some  $h$ , after the  $h$ -th seat has been allocated to  $c$ , it holds that  $\frac{V_c^h}{R_c} < \frac{V_p^h}{R_p}$ . As in the proof of Lemma 4.3, this means that there exists  $c^* \in C(p)$  such that  $\frac{V_{c^*}^h}{R_{c^*}} > \frac{V_p^h}{R_p} > \frac{V_c^h}{R_c}$ . Hence, the set  $I := \{i \in C(p) \mid \frac{V_i^h}{R_i} > \frac{V_p^h}{R_p}\}$  is non-empty and  $c \notin I$ . Let  $g \geq 0$  be the smallest integer such that  $V_i^g = V_i^h$  for all  $i \in I$ . Since  $c$  receives the  $h$ -th seat, we have  $g < h$ . Also,  $g = 0$  would imply that  $V_i^g = V_i^h = 0$  for all  $i \in I$ , contradicting the definition of  $I$ , so  $g > 0$ . Let  $J := \{j \in C(p) \mid V_j^h > V_j^g\}$ ; by definition, we know that  $c \in J$  and  $J \cap I = \emptyset$ . For each  $j \in J$ , since  $j \notin I$ , we have  $\frac{V_j^h}{R_j} \leq \frac{V_p^h}{R_p}$ . Also, by definition of  $g$ , there exists a node  $i^* \in I$  that receives the  $g$ -th seat. Thus, for each  $j \in J$ , we have  $\frac{V_{i^*}^g}{R_{i^*}} = \frac{V_{i^*}^h}{R_{i^*}} > \frac{V_p^h}{R_p} \geq \frac{V_j^h}{R_j} \geq \frac{V_j^g+1}{R_j}$ . Since  $j$  does not receive the  $g$ -th seat, it must be that receiving the  $g$ -th seat would violate the upper quota check for  $j$ , and so  $\frac{V_j^g}{R_j} \geq \frac{V_p^g}{R_p}$ .

---

**Input :** List  $V^h$  representing the allocation for  $h$  seats  
**Output:** List  $V^{h+1}$  representing the allocation for  $h + 1$  seats

- 1  $V^{h+1} \leftarrow V^h$  // initialize  $V^{h+1}$  to be  $V^h$
- 2  $i \leftarrow 0$  // root node
- 3  $V_i^{h+1} \leftarrow V_i^h + 1$  // root node always receives the new seat
- 4 **while**  $i$  is not a leaf node **do**
- 5     Let  $c$  be a child of  $i$  with the smallest  $\frac{V_c^h + 1}{R_c}$  subject to  $\frac{V_c^h}{W_c} < V_i^h + 1$ , breaking ties arbitrarily
- 6      $V_c^{h+1} \leftarrow V_c^h + 1$
- 7      $i \leftarrow c$

---

**Algorithm 3** Multi-level quota method

Summarizing what we have so far, we know that  $\frac{V_j^h}{R_j} \leq \frac{V_p^h}{R_p}$  and  $\frac{V_j^g}{R_j} \geq \frac{V_p^g}{R_p}$  for all  $j \in J$ . Combining them yields  $\frac{V_j^h - V_j^g}{R_j} \leq \frac{V_p^h - V_p^g}{R_p}$  for all  $j \in J$ , or equivalently,  $V_j^h - V_j^g \leq \frac{R_j}{R_p}(V_p^h - V_p^g)$ . Summing this up across all  $j \in J$ , we get  $\sum_{j \in J}(V_j^h - V_j^g) \leq \frac{\sum_{j \in J} R_j}{R_p} \cdot (V_p^h - V_p^g)$ . Since  $\sum_{j \in J}(V_j^h - V_j^g) = V_p^h - V_p^g$  by definition of  $J$ , and  $V_p^h - V_p^g > 0$  because  $c$  receives the  $h$ -th seat, this simplifies to  $R_p \leq \sum_{j \in J} R_j$ . However, since  $I$  is non-empty, we must have  $\sum_{j \in J} R_j < R_p$ , a contradiction.  $\square$

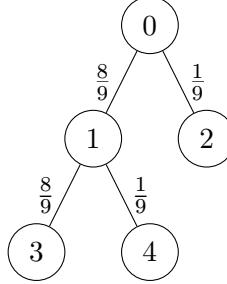
With Lemma 5.1 in hand, we can establish lower quota in exactly the same way as for multi-level Jefferson (Theorem 4.4).

**Theorem 5.2** *Algorithm 3 satisfies lower quota and house monotonicity in multi-level apportionment.*

Unfortunately, even though the algorithm has a check against upper quota violations with respect to the parent, this is insufficient to guarantee upper quota in the multi-level setting.

**Proposition 5.3** *Algorithm 3 fails upper quota in multi-level apportionment.*

**Proof** Consider the instance where the tree structure is as shown in Fig. 3, and assume that there are  $h = 5$  seats to be allocated. Since  $\frac{4+1}{8/9} < \frac{0+1}{1/9}$  and  $\frac{4}{8/9} < 4 + 1$ , the algorithm allocates all five seats to node 1. By the same reasoning, the algorithm allocates all five seats to node 3. However, since  $\lceil 5 \cdot \frac{8}{9} \cdot \frac{8}{9} \rceil = 4$ , upper quota for node 3 with respect to node 0 is violated.  $\square$

**Fig. 3** Example tree structure for which Algorithm 3 fails upper quota

**Input :** List  $V^h$  representing the allocation for  $h$  seats

**Output:** List  $V^{h+1}$  representing the allocation for  $h + 1$  seats

```

1  $V^{h+1} \leftarrow V^h$            // initialize  $V^{h+1}$  to be  $V^h$ 
2  $i \leftarrow 0$                 // root node
3  $V_i^{h+1} \leftarrow V_i^h + 1$     // root node always receives the new seat
   /* Threshold value to prevent upper quota violation. When considering a node  $c$ , we have  $t = \min_{a \in A(c)} \frac{V_a^h + 1}{R_a}$ . */
4  $t \leftarrow V_0^h + 1$           // Note that, by definition,  $R_0 = 1$ .
5 while  $i$  is not a leaf node do
6   Let  $c$  be a child of  $i$  with the smallest  $\frac{V_c^h + 1}{R_c}$  subject to  $\frac{V_c^h}{R_c} < t$ , breaking ties arbitrarily
7    $V_c^{h+1} \leftarrow V_c^h + 1$ 
8    $i \leftarrow c$ 
9    $t \leftarrow \min(t, \frac{V_c^h + 1}{R_c})$ 
```

**Algorithm 4** Upper-compliant multi-level quota method

To fix the failure in Proposition 5.3, one could try to prevent upper quota violations of a node not only with respect to its parent, but also with respect to all of its ancestors. The resulting algorithm is shown as Algorithm 4. In order for a node  $c$  to be eligible to receive the next seat, the algorithm requires that  $\frac{V_c}{R_c} < \frac{V_a + 1}{R_a}$  for all  $a \in A(c)$ . Unlike the previous three algorithms, it is not trivial to see that Algorithm 4 is well-defined—that is, if a non-leaf node  $i$  receives a seat, then at least one of its children is also eligible to receive the seat. We establish this fact in the following proposition.

**Proposition 5.4** *Algorithm 4 is well-defined.*

**Proof** Suppose for contradiction that a node  $i$  receives a seat but none of its children is eligible to receive the seat. This means that  $\frac{V_c}{R_c} \geq \min_{a \in A(i) \cup \{i\}} \frac{V_a + 1}{R_a}$  for all  $c \in C(i)$ . If  $\min_{a \in A(i) \cup \{i\}} \frac{V_a + 1}{R_a} = \frac{V_i + 1}{R_i}$ , we would have

$$\frac{V_i}{R_i} = \sum_{c \in C(i)} \frac{V_c}{R_i} = \sum_{c \in C(i)} \frac{V_c}{R_c} \cdot W_c \geq \frac{V_i + 1}{R_i} \cdot \sum_{c \in C(i)} W_c = \frac{V_i + 1}{R_i}, \quad (2)$$

a contradiction. Hence,  $\frac{V_c}{R_c} \geq \min_{a \in A(i)} \frac{V_a+1}{R_a}$  for all  $c \in C(i)$ . Using the same reasoning as in (2), we get  $\frac{V_i}{R_i} \geq \min_{a \in A(i)} \frac{V_a+1}{R_a}$ . However, this means that  $i$  should not have received the seat, a contradiction.  $\square$

Since Algorithm 4 prevents all possible upper quota violations when assigning each seat, upper quota is satisfied. In addition, house monotonicity is fulfilled due to the iterative nature of the algorithm.

**Theorem 5.5** *Algorithm 4 satisfies upper quota and house monotonicity in multi-level apportionment.*

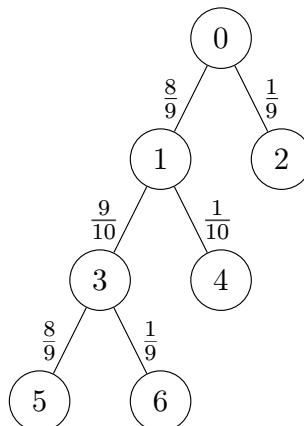
However, the upper quota check across multiple levels can cause lower quota to be violated.

**Proposition 5.6** *Algorithm 4 fails lower quota in multi-level apportionment.*

**Proof** Consider the instance where the tree structure is as shown in Fig. 4, and assume that there are  $h = 5$  seats to be allocated. As in the proof of Proposition 5.3, the algorithm allocates all five seats to node 1. Without the upper quota check, node 3 would receive all five seats. However, because  $\frac{3}{(8/9) \cdot (9/10)} < 3 + 1$  while  $\frac{4}{(8/9) \cdot (9/10)} \geq 4 + 1$ , it only receives the first four seats. Similarly, because  $\frac{2}{(8/9) \cdot (9/10) \cdot (8/9)} < 2 + 1$  while  $\frac{3}{(8/9) \cdot (9/10) \cdot (8/9)} \geq 3 + 1$ , among the four seats that node 3 receives, only the first three are passed on to node 5. Since  $\lfloor 5 \cdot \frac{9}{10} \cdot \frac{8}{9} \rfloor = 4$ , this means that lower quota for node 5 with respect to node 1 is violated.  $\square$

In light of our results, an interesting question is whether there exists a rule that satisfies both upper and lower quota as well as house monotonicity in multi-level apportionment, as the quota method does in single-level apportionment (Balinski and

**Fig. 4** Example tree structure where Algorithm 4 fails lower quota



Young 1975, Thm. 3). This question is open even when we relax our quota notions to require compliance only relative to the root node (cf. Footnote 4).

## 6 Experimental evaluation

To complement our theoretical results, we experimentally evaluate Adams', Jefferson's, Quota, and Upper-Compliant (UC) Quota methods (Algorithms 1–4, respectively). In particular, we run experiments with trees of heights 3, 4, 5, and 6, using the following two tree types:

- Full binary tree. Following our claim in Lemma 3.2, an arbitrary instance can be reduced to a full binary tree instance without relaxing any constraints, thereby making full binary trees natural for evaluation. We run tests with  $n \in \{15, 31, 63, 127\}$ , which correspond to perfect binary trees. However, since Jefferson's and Quota methods can be shown to be equivalent when a node has only two children, we also consider another tree type, as explained next.
- Full 4-ary tree. To keep the value of  $n$  relatively small, we generate trees in the following manner: if we list the nodes at each level ordered from left to right, then except for the last level, nodes with even indices (0-based) have four children while the remaining nodes have zero children. We run tests with  $n \in \{29, 61, 125, 253\}$ ; these values arise because the number of nodes at each level are 1, 4, 8, 16, 32, 64, and 128, respectively.

For each combination of tree type and  $n$ -value, we generate 100,000 instances with random entitlements. We assign entitlements as follows: for each non-leaf node, we generate a random array with length equal to the number of its children, where each entry is an integer between 1 and 10 (inclusive). The entitlement of each child equals its corresponding value divided by the sum of the array. This keeps the ratio of entitlement between any two sibling nodes at most 1 : 10 and the assigned entitlements relatively small, which helps avoid precision errors when working with floating-point numbers.

We run the four algorithms for each generated instance and analyze the allocation results at  $h \in \{100, 500\}$  with respect to the following four metrics:

- Lower quota violation rate. This is calculated by dividing the number of nodes that violate their lower quota (as defined in Sect. 2) by the value of  $n$ , then averaging across all generated instances. The results are presented in Tables 1 and 2, with Jefferson's and Quota methods omitted because Theorem 4.4 and 5.2 show that they always satisfy lower quota.

**Table 1** Average lower quota violation rate (% of nodes) for binary tree

Method	$n = 15$		$n = 31$		$n = 63$		$n = 127$	
	$h = 100$	$h = 500$						
Adams	1.2747	1.0580	1.4816	1.3939	1.6183	1.5440	1.7694	1.6135
UC quota	0.0120	0.0087	0.0300	0.0310	0.0502	0.0513	0.0609	0.0655

**Table 2** Average lower quota violation rate (% of nodes) for 4-ary trees. Note that while some values for the Upper-Compliant Quota method appear as 0.0000, they are non-zero but rounded to four decimal places

Method	$n = 29$		$n = 61$		$n = 125$		$n = 253$	
	$h = 100$	$h = 500$						
Adams	1.6681	1.4281	1.5177	1.5740	0.7814	1.4439	0.3863	0.8205
UC Quota	0.0001	0.0001	0.0000	0.0001	0.0000	0.0001	0.0000	0.0001

**Table 3** Average upper quota violation rate (% of nodes) for binary tree

Method	$n = 15$		$n = 31$		$n = 63$		$n = 127$	
	$h = 100$	$h = 500$						
Jefferson	0.9587	1.1367	1.3113	1.4119	1.4798	1.5552	1.4998	1.6120
Quota	0.9587	1.1367	1.3113	1.4119	1.4798	1.5552	1.4998	1.6120

**Table 4** Average upper quota violation rate (% of nodes) for 4-ary tree

Method	$n = 29$		$n = 61$		$n = 125$		$n = 253$	
	$h = 100$	$h = 500$						
Jefferson	1.5021	1.6556	1.1891	1.5936	0.6871	1.2565	0.3452	0.7668
Quota	0.4949	0.4806	0.4916	0.5660	0.2994	0.5209	0.1502	0.3387

**Table 5** Average deviation from strict quota for binary tree

Method	$n = 15$		$n = 31$		$n = 63$		$n = 127$	
	$h = 100$	$h = 500$						
Adams	0.3009	0.2857	0.3251	0.3155	0.3457	0.3315	0.3683	0.3427
Jefferson	0.2850	0.2919	0.3156	0.3191	0.3296	0.3331	0.3288	0.3397
Quota	0.2850	0.2919	0.3156	0.3191	0.3296	0.3331	0.3288	0.3397
UC quota	0.2628	0.2673	0.2846	0.2864	0.2955	0.2955	0.2983	0.3001

**Table 6** Average deviation from strict quota for 4-ary tree

Method	$n = 29$		$n = 61$		$n = 125$		$n = 253$	
	$h = 100$	$h = 500$						
Adams	0.3468	0.3227	0.3581	0.3524	0.2757	0.3553	0.1807	0.2824
Jefferson	0.3216	0.3277	0.2896	0.3309	0.2078	0.2949	0.1249	0.2163
Quota	0.3083	0.3116	0.2815	0.3170	0.2038	0.2860	0.1230	0.2117
UC quota	0.2945	0.2967	0.2712	0.3002	0.1991	0.2739	0.1208	0.2057

- Upper quota violation rate. This is calculated by dividing the number of nodes that violate their upper quota (as defined in Sect. 2) by the value of  $n$ , then averaging across all generated instances. The results are presented in Tables 3 and 4, with Adams' and Upper-Compliant Quota methods omitted because Theorem 4.2 and 5.5 show that they always satisfy upper quota.
- Average deviation. This is calculated by averaging the absolute difference be-

**Table 7** Maximum deviation from strict quota for binary tree

Method	$n = 15$		$n = 31$		$n = 63$		$n = 127$	
	$h = 100$	$h = 500$						
Adams	0.7713	0.7438	0.9306	0.9165	1.0777	1.0638	1.2304	1.2050
Jefferson	0.7402	0.7575	0.9118	0.9241	1.0566	1.0685	1.1930	1.2060
Quota	0.7402	0.7575	0.9118	0.9241	1.0566	1.0685	1.1930	1.2060
UC quota	0.6555	0.6665	0.7589	0.7677	0.8279	0.8364	0.8803	0.8900

**Table 8** Maximum deviation from strict quota for 4-ary tree

Method	$n = 29$		$n = 61$		$n = 125$		$n = 253$	
	$h = 100$	$h = 500$						
Adams	0.9832	0.9447	1.1335	1.1171	1.1533	1.2537	1.1568	1.2801
Jefferson	0.9463	0.9701	1.0384	1.1066	1.0852	1.1990	1.1066	1.2412
Quota	0.8382	0.8512	0.9144	0.9356	0.9667	1.0028	0.9969	1.0452
UC quota	0.7700	0.7877	0.8208	0.8308	0.8889	0.8663	0.9373	0.9154

tween the number of allocated seats and the strict quota<sup>7</sup> across all nodes and generated instances. The results are presented in Tables 5 and 6.

- Maximum deviation. This is calculated by taking the maximum absolute difference between the number of allocated seats and the strict quota across all nodes and generated instances. The results are presented in Tables 7 and 8.

Our code is available at <https://github.com/stevenwjy/multi-level-apportionment>.

From the results, we make the following observations:

- As discussed earlier, Jefferson's and Quota methods yield identical results on binary trees.
- The violations and deviations tend to increase with  $n$  for perfect binary trees, but decrease with  $n$  for 4-ary trees. We hypothesize that the latter occurs because our construction makes 4-ary trees sparser as the height increases.
- As shown in Tables 1 and 2, the lower quota violation rate of the Upper-Compliant Quota method is substantially smaller than that of Adams' method.
- As shown in Table 4, the upper quota violation rate of the Quota method is smaller than that of Jefferson's method in 4-ary trees.
- As shown in Tables 5, 6, 7, and 8, the Upper-Compliant Quota method generally exhibits smaller average and maximum deviations than the other three algorithms.

**Acknowledgements** This work is partially supported by the Singapore Ministry of Education under grant number MOE-T2EP20221-0001 and by an NUS Start-up Grant. Part of this research was carried out while Ulrike Schmidt-Kraepelin was supported by the National Science Foundation under Grant No. DMS-1928930 and by the Alfred P. Sloan Foundation under grant G-2021-16778 while being in residence at the Simons Laufer Mathematical Sciences Institute (formerly MSRI) in Berkeley, California, during the Fall 2023 semester. We would like to thank Ayumi Igarashi for helpful discussions and the anonymous reviewer for valuable feedback.

<sup>7</sup>The *strict quota* of a node  $i$  is defined as  $R_i \cdot h$ , where  $R_i$  denotes the entitlement of node  $i$  relative to the root (as defined in Sect. 2).

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Balinski, M. L., & Young, H. P. (1975). The quota method of apportionment. *American Mathematical Monthly*, 82(7), 701–730.
- Balinski, M.L., & Young, H.P. (2001). *Fair Representation: Meeting the Ideal of One Man, One Vote*. Brookings Institution Press
- Bochsler, D. (2010). Who gains from appertainments under D'Hondt? *Electoral Studies*, 29(4), 617–627.
- Brams, S. J., Kilgour, D. M., & Pothoff, R. F. (2019). Multiwinner approval voting: an apportionment approach. *Public Choice*, 178(1–2), 67–93.
- Brill, M., Götz, P., Peters, D., Schmidt-Kraepelin, U., & Wilker, K. (2024). Approval-based apportionment. *Mathematical Programming*, 203(1–2), 77–105.
- Cembrano, J., Correa, J., & Verdugo, V. (2022). Multidimensional political apportionment. *Proceedings of the National Academy of Sciences*, 119(15), Article e2109305119.
- Karpov, A. (2015). Alliance incentives under the D'Hondt method. *Mathematical Social Sciences*, 74, 1–7.
- Koriyama, Y., Macé, A., Treibich, R., & Laslier, J.-F. (2013). Optimal apportionment. *Journal of Political Economy*, 121(3), 584–608.
- Leutgäb, P., & Pukelsheim, F. (2009). List appartenements in local elections - a lottery. *Homo Oeconomicus*, 26(3–4), 489–500.
- Mathieu, C., & Verdugo, V. (2024). Apportionment with parity constraints. *Mathematical Programming*, 203(1–2), 135–168.
- Pukelsheim, F. (2014). *Proportional Representation: Apportionment Methods and Their Applications*. Springer

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.