KAPITEL 9 / *CHAPTER 9*
GENERATING OF LINEAR DIVISOR METHODS' FULL FAVORING APPORTIONMENTS

ГЕНЕРИРОВАНИЕ ПОЛНОСТЬЮ БЛАГОСКЛОННЫХ РАСПРЕДЕЛЕНИЙ МЕТОДАМИ С ЛИНЕЙНЫМ ДЕЛИТЕЛЕМ

DOI: 10.21893/2709-2313.2021-04-11-037

Introduction

Often it is necessary to distribute a given number M of discrete entities of the same kind among n beneficiaries, in proportion to a numerical characteristic assigned to each of them V_i, $i = \overline{1,n}$. This is known as proportional apportionment (APP) problem [1-3]. The integer character of this problem usually causes a certain disproportion of the apportionment $\{x_i, i = \overline{1,n}\}$ [1, 4, 5], some beneficiaries being favored at the expense of the others. Such favoring leads to the increase of disproportionality of the apportionment. Therefore, reducing the favoring in question is one of the basic requirements when is choosing the APP method to be applied for apportionments.

As it is well known, the d'Hondt method [6] favors large beneficiaries (with larger V_i value) [1, 4, 7, 8], and Huntington-Hill method [9] favors the small ones (with smaller V_i value) [4, 10]. But which of the two favors beneficiaries to a larger extent? Preferences, in this sense, between methods, can help. Par example, in [11], five APP methods are placed „in the order as they are known to favor larger parties over smaller parties". However, the best way is to estimate this property quantitatively. One approach in this aim is proposed in [12]. Another, the " total (full) favoring", based on the definition of favoring of large beneficiaries or of the small ones by an APP method done in [1], is examined in [15, 16]. In [15], it was shown that the frequency of full favoring in apportionments, for the widely used Hamilton (Hare) [13], Sainte-Laguë (Webster) [14], d'Hondt (Jefferson), Huntington-Hill and Adapted Sainte-Laguë methods, is strongly decreasing on n, becoming approx. 0 at $n \geq 7 \div 10$. In [16], the conditions of linear divisor methods' (LDMs') apportionments compliance with the requirements of full favoring of large beneficiaries or of the small ones were determined. Also, the A1 algorithm for determining the LDMs' apportionments which fully favor beneficiaries was elaborated and some examples with the use of this algorithm were given. This algorithm is simple, but it doesn't guarantee the obvious solution. Aspects of the guaranteed generation of LDMs' apportionments, which fully favor large beneficiaries or the small ones, are examined in this paper.

9.1. Essence of favoring and of full favoring of beneficiaries in apportionments

The essence of favoring of beneficiaries in apportionments is described in such papers as [4, 7, 10]. In [12], for example, three notions of favoring of beneficiaries by

an APP method are distinguished:

a) favoring of a beneficiary in an apportionment;
b) favoring of large beneficiaries or of the small ones in an apportionment;
c) favoring of large beneficiaries or of the small ones overall by an apportionment method.

It is considered that a beneficiary i is favored if a larger number x_i of entities is distributed to him than would be due according to the V_i value, more precisely if $x_i > MV_i/V$, where $M = x_1 + x_2 + \ldots + x_n$ and $V = V_1 + V_2 + \ldots + V_n$. Of course, the lack of favoring is possible only if the equalities $a_i = MV_i/V$, $i = \overline{1,n}$ take place; here $a_i = \lfloor MV_i/V \rfloor$, where $\lfloor z \rfloor$ means the integer part of the real number z. In practice, such equalities rarely occur and that is why some beneficiaries are favored and others, respectively, are disfavored. The notation $\Delta M = M - a_1 + a_2 + \ldots + a_n$ will also be used.

In formalized form, the first, probably, definition of favoring of large beneficiaries or of the small ones by an APP method is given in [1]. But the requirements of this definition are very strong - no methods compliant to them and used in practice are known. At the same time, as mentioned in [15], these conditions can be used to identify the "full favoring" of large beneficiaries or of the small ones in concrete apportionments. Also, in [12], the conditions of the respective definition in [1] were simplified, reducing considerably the volume of needed calculations for computer simulation (see Definition 1).

Definition 1. In an apportionment, large beneficiaries are fully favored if

$$\frac{x_i}{V_i} > \frac{x_j}{V_j}, \tag{1}$$

and small beneficiaries are fully favored if

$$\frac{x_i}{V_i} < \frac{x_j}{V_j} \tag{2}$$

whenever $x_i > x_j$, where i and j take values from the $\{1, 2, 3, \ldots, n\}$ ones [12].

Usually, in one and the same apportionment some large and some small beneficiaries can be favored and, nevertheless, mainly large or, on the contrary, small beneficiaries can be favored. Therefore, in [12] it is proposed to use two different notions: "favoring" of large or of small beneficiaries and "full favoring" of large or of small beneficiaries, the second being a particular case of the first one. The compliance of an apportionment with requirement (1) or with the (2) one is referred to "full favoring" of large beneficiaries or, respectively, of the small ones. The larger notion of „favoring" of large beneficiaries or of the small ones is used when in an apportionment are predominantly favored large beneficiaries or, on the contrary, the small ones in sense of [12].

In order to identify whether apportionments that fully favor large beneficiaries or the small ones can be obtained when applying an APP method, it is necessary to know the compliance requirements of the APP method in question with requirement (1) or, respectively, the (2) one.

9.2. Compliance of LDMs' apportionments with requirements (1) or (2)

From the multitude of APP methods, there some linear divisor methods [8] are examined. The conditions for compliance of LDMs' apportionments, at $c > 0$ and, separately, at $c \geq 1$, with requirements (1) or the (2) one were defined in [16]. Also, as shown in [16], at $c > 0$ no one of linear divisor methods is always compliant with the requirement (1) of full favoring of large beneficiaries and with the requirement (2) of full favoring of small beneficiaries. So, it is of interest to study apportionments that fully favor beneficiaries.

It is well known that d'Hondt method ($c = 1$) strongly favors large beneficiaries [4, 7]. At the same time, the lower is the value of c, the greater is the grade of favoring of large beneficiaries by the respective linear divisor method [8]. That's why below only cases with $c \geq 1$ are considered. Also, it is easier to analyze apportionments when beneficiaries are ordered by V_i, $i = \overline{1,n}$ values or the x_i, $i = \overline{1,n}$ ones. Evidently, for "proportional" apportionments examined in this paper, if $x_i > x_{i+1}$, $i = \overline{1,n-1}$ then the relations $V_i > V_{i+1}$, $i = \overline{1,n-1}$ occur, too.

The conditions for compliance of linear divisor methods' apportionments, at $c \geq 1$ and $x_i > x_{i+1}$, $i = \overline{1,n-1}$, with requirements (1) or the (2) one are, respectively, [16]:

$$\max\left\{V_{i-1}\frac{x_i}{x_{i-1}}, \max_{j=\overline{i+1,n}}\left[V_j\frac{c(x_i-1)+1}{cx_j+1}\right]\right\} < V_i < \min\left\{V_{i+1}\frac{x_i}{x_{i+1}}; \min_{j=\overline{1,i-1}}\left[V_j\frac{cx_i+1}{c(x_j-1)+1}\right]\right\} \quad i=\overline{1,n} \tag{3}$$

and

$$\max\left\{V_{i+1}\frac{x_i}{x_{i+1}}, \max_{j=\overline{1,i-1}}\left[V_j\frac{c(x_i-1)+1}{cx_j+1}\right]\right\} < V_i < \min\left\{V_{i-1}\frac{x_i}{x_{i-1}}; \min_{j=\overline{i+1,n}}\left[V_j\frac{cx_i+1}{c(x_j-1)+1}\right]\right\}, i=\overline{1,n}. \tag{4}$$

9.3. Generating apportionments that fully favor large beneficiaries

According to [16], one of the ways of determining apportionments which fully favor large/small beneficiaries, for linear divisor methods with $c \geq 1$, is the following. Starting from $\{V_1,\ x_i > x_{i+1},\ i = \overline{1,n-1}\}$ values and using formula (3) or, respectively, the (4) one and a special iterative algorithm, one can obtain the set of V_i, $i = \overline{1,n}$ values and, thus, the apportionment that corresponds to requirement (1) or, respectively, to the (2) one of Definition 1.

A simple algorithm (A1), which for small values of n can be easy implemented using a table processor, for example, Microsoft Excel, is described in [16]. But this algorithm doesn't guarantee the obvious solution. They guarantee the solution (if it exists), regardless of the value of n, the relatively laborious A2 and A3 algorithms, described below.

Let's begin with **algorithm A2** for the generating of **apportionments that fully**

favor large beneficiaries. Because initially only the $\{V_1,\ x_i > x_{i+1},\ i = \overline{1,n-1}\}$ values are known, in requirement (3) the factor $\max\limits_{j=\overline{i+1,n}} \left\lfloor V_j \frac{c(x_i-1)+1}{cx_j+1} \right\rfloor$ of the first inequality left side and the factor $V_{i+1}x_i/x_{i+1}$ of the second inequality right side are not known. Because of $x_i > x_{i+1}$, $i = \overline{1,n-1}$, one has $V_i > V_{i+1}$, $i = \overline{1,n-1}$ and basing on (1) to V_i, $i = \overline{2,n}$ can be preliminary assigned minimal possible values $V_i = \lfloor V_{i-1} x_i/x_{i-1} \rfloor + 1$, $i = \overline{2,n}$. If for at list one of the beneficiaries $i = \overline{2,n}$ occurs $V_i \geq V_{i-1}$, then the solution doesn't exist. Otherwise, taking into account requirement (3), the current V_i lower limit ($V_{i\,\min}$) can be determined as

$$V_{i\,\min} = \left\lfloor \max\left\{ V_{i-1}\frac{x_i}{x_{i-1}};\ \max_{j=\overline{i+1,n}} \left\lfloor V_j \frac{c(x_i-1)+1}{cx_j+1} \right\rfloor \right\} \right\rfloor + 1 \qquad (5)$$

and the V_i high limit ($V_{i\,\max}$) can be determined as

$$V_{i\,\max} = \left\lfloor \min\left\{ V_{i+1}\frac{x_i}{x_{i+1}};\ \min_{j=\overline{1,i-1}} \left\lfloor V_j \frac{cx_i+1}{c(x_j-1)+1} \right\rfloor \right\} \right\rfloor - 1. \qquad (6)$$

Similarly, instead of minimal possible values preliminary assigned to V_i, $i = \overline{2,n}$, one can use maximal possible values $V_i = V_{i-1} - 1$, $i = \overline{2,n}$ preliminary assigned to them. If for at least one of beneficiaries the relation $V_i = V_{i-1} - 1 < \lfloor V_{i-1} x_i/x_{i-1} \rfloor + 1$ occurs, that is $\lfloor V_{i-1}(x_{i-1} - x_i)/x_{i-1} \rfloor < 2$, then the solution doesn't exist.

Thus, **algorithm A2** for determining apportionments of full favoring of large beneficiaries by linear divisor methods consists in the following. Initially, the V_i, $i = \overline{2,n}$ values are determined using $g = 0$ (minimal possible values), after they are determined using $g = 1$ (maximal possible values) and the final values, for stability, are determined as the arithmetic mean of the first two values for each V_i, $i = \overline{2,n}$. Of course, one can use another value for g.

1. Initial data are: c, n, V_1, $x_i > x_{i+1}$, $i = \overline{1,n-1}$. $g = 0$.
2. Determining the values of sizes V_i, $i = \overline{2,n}$ basing on requirement (1) only. Thus, for $i = \overline{2,n}$, assigning to V_i a value in the range $[\lfloor V_{i-1} x_i/x_{i-1} \rfloor + 1;\ V_{i-1} - 1]$:
   2.1. If $V_{i-1} - \lfloor V_{i-1} x_i/x_{i-1} \rfloor < 2$, then the solution doesn't exist. Stop.
   2.2. $V_i = \lfloor V_{i-1} x_i/x_{i-1} \rfloor + 1 + g(V_{i-1} - \lfloor V_{i-1} x_i/x_{i-1} \rfloor - 2)$.
3. Concretizing the values of sizes V_i, $i = \overline{2,n}$ by taking into account the value of V_1 and the requirements of the used apportionment method. Because of Step 2.2, the V_1 value is compliant with requirement (6), that is the relation $V_1 \leq V_{1\,\max}$ occurs. Checking the compliance of V_1 value with requirement (5), that is if the relation $V_1 \geq V_{1\,\min}$ occurs. Thus, for $i = \overline{2,n}$:
   3.1. If $V_1 > V_i[c(x_1 - 1) + 1]/(cx_i + 1)$, then V_1 and V_i are compliant, $i := i + 1$ and repeat Step 3.1.
   3.2. The V_i value is too large. If $g = 0$, then the solution doesn't exist, because the V_i preliminary value is the minimal possible one. Stop.
   3.3. Reducing the V_i value, but assigning to it a maximal possible value with refer to V_1. Thus, $V_i = \lceil V_1(cx_i + 1)/[c(x_1 - 1) + 1] \rceil - 1$. If $V_i > V_{i-1} x_i/x_{i-1}$, then $i := i + 1$ and go to Step 3.1.
   3.4. The V_{i-1} value is too large. If $i = 2$, then the solution doesn't exist, because $V_{i-1} = V_1$. Stop.

3.5. Reducing the V_{i-1} value (and may be those of the V_j, $j = \overline{2, i-2}$ ones), but assigning to it a maximal possible value with refer to V_i. Thus, for $j = \overline{i-1,1}$ in decreasing order, that is $V_j = \lceil V_{j+1} x_j / x_{j+1} \rceil - 1$. If $V_j > V_{j-1} x_j / x_{j-1}$, then $i := i + 1$ and go to Step 3.1.

3.6. If $V_2 \le V_1 x_2 / x_1$, then the solution doesn't exist because the V_1 value can't be reduced. Stop.

4. The preliminary values of sizes V_i, $i = \overline{2,n}$ were determined, being the maximal possible one for the used g value; concretizing them (Steps 5-8). Thus, $i = 2$. $V_{n+1} = 0$.

5. Determining $V_{i\,\min}$ according to (5) and $V_{i\,\max}$ according to (6). Assigning to V_i a value in the range $[V_{i\,\min}, V_{i\,\max}]$, the $V_{i\,\max}$ value being the maximal possible one for the used g value.

5.1. If $V_{i\,\min} \le V_{i\,\max}$, then go to Step 5.4.

5.2. The $V_{i\,\min}$ value is too large. If g = 0, then the solution doesn't exist, because the $V_{i\,\min}$ value can't be reduced. Stop.

5.3. Reducing the V_{i+1} value, but assigning to it a maximal possible value with refer to V_i, that is $V_{i+1} = \lceil V_i(cx_{i+1} + 1)/[c(x_i - 1) + 1] \rceil - 1$. If $V_{i+1} \le V_i x_{i+1} / x_i$, then the V_{i+1} value is too small. The solution doesn't exist, because V_{i+1} can't be increased. Stop.

5.4. $V_i = V_{i\,\min} + g(V_{i\,\max} - V_{i\,\min})$.

6. If $i < n$, then actualizing V_k, $k = \overline{i+1,n}$ by assigning to them values in the range $[\lfloor V_{k-1} x_k / x_{k-1} \rfloor + 1;\ V_{k-1} - 1]$. Thus, for $k = \overline{i+1,n}$:

6.1. If $V_{k-1} - \lfloor V_{k-1} x_k / x_{k-1} \rfloor < 2$, then the solution doesn't exist. Stop.

6.2. $V_k = \lfloor V_{k-1} x_k / x_{k-1} \rfloor + 1 + g(V_{k-1} - \lfloor V_{k-1} x_k / x_{k-1} \rfloor - 2)$.

7. Checking the sizes V_j, $j = \overline{1,i}$ values compliance with requirement (3), taking into account the new values of sizes V_k, $k = \overline{i+1,n}$. They are compliant with requirement (6), that is $V_j \le V_{j\,\max}$, $j = \overline{1,i}$ occur.

7.1. Checking the V_1 value compliance with requirement (5), that is if the relation $V_1 \ge V_{1\,\min}$ occurs. Thus, for $j = \overline{2,i}$:

7.1.1. If $V_1 > V_j[c(x_1 - 1) + 1]/(cx_j + 1)$, then $j := j + 1$ and repeat Step 7.1.1.

7.1.2. The V_j value is too large. If $g = 0$, then the solution doesn't exist, because the V_j value is the minimal possible one. Stop.

7.1.3. Reducing the V_j value by assigning to it a maximal possible value with refer to V_1, that is $V_j = \lceil V_1(cx_j + 1)/[c(x_1 - 1) + 1] \rceil - 1$. If $V_j > V_{j-1} x_j / x_{j-1}$, then $j := j + 1$ and go to Step 7.1.1.

7.1.4. The V_{j-1} value is too large. If $j = 2$, then the solution doesn't exist, because $V_{j-1} = V_1$. Stop.

7.1.5. Reducing the V_{j-1} value (and may be those of the V_j, $j = \overline{2, i-2}$ ones), but assigning to it a maximal possible value with refer to V_j. Thus, for $k = \overline{j-1,2}$ in the decreasing order:

7.1.6. $V_k = \lceil V_{k+1} x_k / x_{k+1} \rceil - 1$. If $V_k > V_{k-1} x_k / x_{k-1}$, then $j := j + 1$ and go to Step 7.1.1.

7.1.6.1. The V_{k-1} value is too large. If $k > 2$, then $k = k - 1$ and go to Step 7.1.5.1.

7.1.6.2. If $V_2 \le V_1 x_2 / x_1$, then the solution doesn't exist because the V_1 value

can't be reduced. Stop.

7.2. Checking the V_j, $j = \overline{2,\iota}$ values compliance with requirement (5), that is if the relations $V_j \geq V_{j\,\min}$, $j = \overline{2,\iota}$ occur. Thus, for $j = \overline{2,\iota}$:

7.2.1. If $V_j > V_{j\,\min}$, where $V_{j\,\min}$ is determined according to (5), then $j := j + 1$ and repeat Step 7.2.1.

7.2.2. The V_{j+1} value is too large. If $g = 0$, then the solution doesn't exist, because the V_{j+1} value is the minimal possible one. Stop.

7.2.3. Reducing the V_{j+1} value, but assigning to it a maximal possible value with refer to V_j, that is $V_{j+1} = \lceil V_j\,(cx_{j+1} + 1)/[c(x_j - 1) + 1] \rceil - 1$. If $V_{j+1} > V_j\,x_{j+1}/x_j$, then $j := j + 1$ and go to Step 7.2.1.

7.2.4. The V_j value is too large. Reducing the V_j value (and may be those of the V_k, $k = \overline{2,\iota-2}$ ones), but assigning to it a maximal possible value with refer to V_{j+1}. Thus, for $\overline{k = j,\iota}$:

7.2.4.1. $V_k = \lceil V_{k+1}\,x_k/x_{k+1} \rceil - 1$. If $V_k > V_{k-1}\,x_k/x_{k-1}$, then $j := j + 1$ and go to Step 7.2.1.

7.2.4.2. The V_{k-1} value is too large. If $k > 2$, then $k := k - 1$ and go to Step 7.2.4.1.

7.2.4.3. If $V_2 \leq V_1 x_2/x_1$, then the solution doesn't exist because the V_1 value can't be reduced. Stop.

8. If $i < n$, then $i := i + 1$ and go to Step 5.

9. The values of V_i, $i = \overline{1,n}$ were found. If $g = 0$, then $U_i = V_i$, $i = \overline{1,n}$; $g = 1$ and go to Step 2.

10. If $g = 1$, then $V_i := [(V_i + U_i)/2]$, $i = \overline{1,n}$. Stop.

The obtained values of V_i, $i = \overline{1,n}$ can be checked by applying the respective APP method.

Algorithm A2 was implemented in the computer application SIMAP. Examples 1 and 2 using SIMAP are described below.

Example 1 of generating of a d'Hondt method apportionment which fully favors large beneficiaries. Initial data: $c = 1$; $M = 279$; $n = 20$; $V_1 = 20000$; $g = 0$; the x_i, $i = \overline{1,n}$ values are specified in Table 1. Some results of calculations using SIMAP are systemized in Table 1.

Table 1 - Calculations for the apportionment to Example 1

i	V_i	x_i	$10^{-8}x_i/V_i$	i	V_i	x_i	$10^{-8}x_i/V_i$	i	V_i	x_i	$10^{-8}x_i/V_i$	i	V_i	x_i	$10^{-8}x_i/V_i$
1	20000	30	150000	6	13336	20	149970	11	8672	13	149908	16	4003	6	149888
2	18001	27	149992	7	12670	19	149961	12	7338	11	149905	17	2669	4	149869
3	16668	25	149988	8	12004	18	149950	13	6671	10	149903	18	2002	3	149850
4	15335	23	149984	9	10671	16	149939	14	5337	8	149897	19	1335	2	149813
5	14669	22	149976	10	9338	14	149925	15	4670	7	149893	20	668	1	149701

Data of Table 1 were checked – the apportionment is a d'Hondt method's one. At the same time it complies with requirements (1). Thus, it fully favors large beneficiaries. One has $V(g{=}0) = 186957$, $\Delta M(g{=}0) = 7$. Also, as expected, it takes place $x_i > a_i$ if and only if $i \leq K$, where $K = \max\{j = \overline{1,n} \mid x_j > a_j\}$. For Example 1 at $g = 0$, one has $K(g{=}0) = 7$, concretely: $x_i = a_i + 1$, $i = \overline{1,7}$ and $x_i = a_i$, $i = \overline{8,20}$. Thus, with refer to particular beneficiaries, seven beneficiaries ($i = \overline{1,7}$) are favored, and thirteen beneficiaries ($i = \overline{8,20}$) are disfavored.

It was identified that depending of the g value, the results can strongly differ. For example, the results of calculations for initial data of Example 1, but for the case of $g = 1$ (**Example 2**), using SIMAP are systemized in Table 2.

Table 2 - Calculations for the apportionment to Example 2

i	V_i	x_i	$10^{-7}x_i/V_i$	i	V_i	x_i	$10^{-7}x_i/V_i$	i	V_i	x_i	$10^{-7}x_i/V_i$	i	V_i	x_i	$10^{-7}x_i/V_i$
1	20000	30	1500	6	13999	20	1429	11	9333	13	1393	16	4666	6	1286
2	18666	27	1446	7	13333	19	1425	12	7999	11	1375	17	3333	4	1200
3	17333	25	1442	8	12666	18	1421	13	7333	10	1364	18	2666	3	1125
4	15999	23	1438	9	11333	16	1412	14	5999	8	1334	19	1999	2	1001
5	15333	22	1435	10	9999	14	1400	15	5333	7	1313	20	1333	1	0750

Data of Table 2 were checked – the apportionment is a d'Hondt method's one. At the same time, it complies with requirements (1). Thus, it fully favors large beneficiaries. One has $V(g=0) = 192351$, $\Delta M(g=1) = 10$. Also, as expected, it takes place $x_i > a_i$ if and only if $i \le K$, where $K = \max\{j = \overline{1,n} \mid x_j > a_j\}$. For Example 2, one has $K(g=1) = 9$, concretely: $x_1 = a_1 + 2$; $x_i = a_i + 1$, $i = \overline{2,9}$ and $x_i = a_i$, $i = \overline{10,20}$. Thus, with refer to particular beneficiaries, nine beneficiaries ($i = \overline{1,9}$) are favored, and eleven beneficiaries ($i = \overline{10,20}$) are disfavored. Moreover the first beneficiary is strongly favored because of $x_1 > a_1 + 1$.

Based on data in Tables 1 and 2 and on other results of calculations using SIMAP, may be concluded that for A2 algorithm the lower the g value, the lower the values of V, ΔM and of the number of beneficiaries that are favored (K). So, one has:

$V(g=0) = 186957 < V(g=0.5) = 192351 < V(g=1) = 198655$;

$\Delta M(g=0) = 7 < \Delta M(g=0.5) = 9 < \Delta M(g=1) = 10$;

$K(g=0) = 7 < K(g=0.5) = 9 = K(g=1) = 9$.

9.4. Generating apportionments that fully favor small beneficiaries

Similarly as in case of Section 3, initially in requirement (4) the factor $V_{i+1}x_i/x_{i+1}$ of the first inequality left side and the factor $\min\limits_{j=\overline{i+1,n}}\left[V_j \frac{cx_i+1}{c(x_j-1)+1}\right]$ of the second inequality right side are not known. Because of $x_i > x_{i+1}$, $i = \overline{1,n-1}$, one has $V_i > V_{i+1}$, $i = \overline{1,n-1}$ and basing on (2) to V_i, $i = \overline{2,n}$ can be preliminary assigned maximal possible values $V_i = \lceil V_{i-1}\, x_i/x_{i-1} \rceil - 1$, $i = \overline{2,n}$. If for at list one of the beneficiaries $i = \overline{2,n}$ occurs $V_i \ge V_{i-1}$ then the solution doesn't exist. Otherwise, taking into account requirement (4), the current V_i lower limit ($V_{i\ \min}$) can be determined as

$$V_{i\,\min} = \left\lfloor \max\left\{ V_{i+1}\frac{x_i}{x_{i+1}}; \max_{j=\overline{1,i-1}}\left[V_j \frac{c(x_i-1)+1}{cx_j+1}\right]\right\}\right\rfloor + 1 \qquad (7)$$

and the V_i high limit ($V_{i\,\max}$) can be determined as

$$V_{i\,\max} = \left\lfloor \min\left\{ V_{i-1}\frac{x_i}{x_{i-1}}; \min_{j=\overline{i+1,n}}\left[V_j \frac{cx_i+1}{c(x_j-1)+1}\right]\right\}\right\rfloor - 1. \qquad (8)$$

Similarly, instead of maximal possible values preliminary assigning to V_i, $i = \overline{2,n}$, one can use minimal possible values

$$V_i = V_{i\ \text{prmin}} = \left\lfloor \max_{j=\overline{1,i-1}} \left[V_j \frac{c(x_i - 1) + 1}{cx_j + 1} \right] \right\rfloor + 1 \qquad (9)$$

preliminary assigning to them. But thus determined minimal for assigning values must be lower than thus determined maximal for assigning values, otherwise the solution doesn't exist.

Thus, **algorithm A3** for determining **apportionments that fully favor small beneficiaries** by linear divisor methods consists in the following. Initially, the values of V_i, $i = \overline{2,n}$ are determined using $g = 0$ (minimal possible values), after they are determined using $g = 1$ (maximal possible values) and the final values, for stability, are determined as arithmetic mean of the first two values for each V_i, $i = \overline{2,n}$.

1. Initial data are: c, n, V_1, $x_i > x_{i+1}$, $i = \overline{1,n-1}$. $g = 0$.
2. Determining the preliminary values of sizes V_i, $i = \overline{2,n}$ basing on requirement (2) and relation (9). Thus, for $i = \overline{2,n}$ assigning to V_i a value in the range [$V_{i\ \text{prmin}}$; $V_{i\ \text{prmax}}$], where $V_{i\ \text{prmin}}$ is determining according to (9) and $V_{i\ \text{prmax}} = \lceil V_{i-1}\, x_i/x_{i-1} \rceil - 1$:
   2.1. If $V_{i\ \text{prmin}} > V_{i\ \text{prmax}}$, then the solution doesn't exist. Stop.
   2.2. $V_i = V_{i\ \text{prmin}} + g(V_{i\ \text{prmax}} - V_{i\ \text{prmin}})$.
3. Concretizing the preliminary values of sizes V_i, $i = \overline{2,n}$ by taking into account the value of V_1 and the requirements of the used apportionment method. Because of Step 2.2, the V_1 value is compliant with requirement (7), that is the relation $V_1 \geq V_{1\ \text{min}}$ occurs. Checking the compliance of V_1 value with requirement (8), that is if the relation $V_1 \leq V_{1\ \text{max}}$ occurs. Thus, for $i = \overline{2,n}$:
   3.1. If $V_1 < V_i\,(cx_1 + 1)/[c(x_i - 1) + 1]$, then V_1 and V_i are compliant, $i := i + 1$ and repeat Step 3.1.
   3.2. The V_i value is too small. If $g = 1$, then the solution doesn't exist, because the V_i preliminary value is the maximal possible one. Stop.
   3.3. Increasing the V_i value, but assigning to it a minimal possible value with refer to V_1: $V_i = \lfloor V_1[(cx_i - 1) + 1]/(cx_1 + 1) \rfloor + 1$. If $V_i < V_{i-1}\, x_i/x_{i-1}$, then $i := i + 1$ and go to Step 3.1.
   3.4. The V_{i-1} value is too small. If $i = 2$, then the solution doesn't exist, because $V_{i-1} = V_1$. Stop.
   3.5. Increasing the V_{i-1} value (and may be those of the V_j, $j = \overline{2,i-2}$ ones), but assigning to it a minimal possible value with refer to V_i. Thus, for $j = \overline{i-1,2}$ in decreasing order: $V_j = \lfloor V_{j+1}\, x_j/x_{j+1} \rfloor + 1$. If $V_j < V_{j-1}\, x_j/x_{j-1}$, then $i := i + 1$ and go to Step 3.1.
   3.6. If $V_2 \geq V_1 x_2/x_1$, then the solution doesn't exist because $V_{i-1} = V_1$. Stop.
4. The preliminary values of sizes V_i, $i = \overline{2,n}$ were determined, being the minimal possible one for the used g value; concretizing them (Steps 5-8). $i = 2$. $V_{n+1} = 0$.
5. Determining $V_{i\ \text{min}}$ according to (7) and $V_{i\ \text{max}}$ according to (8). Assigning to V_i a value in the range [$V_{i\ \text{min}}$, $V_{i\ \text{max}}$], the $V_{i\ \text{min}}$ value being the minimal possible one for the used g value.
   5.1. If $V_{i\ \text{min}} \leq V_{i\ \text{max}}$, then go to Step 5.4.
   5.2. The $V_{i\ \text{max}}$ value is too small. If g = 1, then the solution doesn't exist, because the $V_{i\ \text{max}}$ value can't be increased. Stop.
   5.3. Increasing the V_{i+1} value, but assigning to it a minimal possible value with refer

to V_i, that is $V_{i+1} = \lfloor V_i[c(x_{i+1} - 1) + 1]/(cx_i + 1) \rfloor + 1$. If $V_{i+1} \geq V_i x_{i+1}/x_i$, then the V_{i+1} value is too large. The solution doesn't exist, because V_{i+1} can't be reduced. Stop.

5.4. $V_i = V_{i\,\min} + g(V_{i\,\max} - V_{i\,\min})$.

6. If $i < n$, then actualizing V_k, $k = \overline{i+1,n}$ by assigning to them values in the range $[V_{k\ \text{amin}};\ V_{k\ \text{amax}}]$, where $V_{k\ \text{amin}}$ is determining according to (9) and $V_{k\ \text{amax}} = \lceil V_{k-1}\, x_k/x_{k-1} \rceil - 1$. Thus, for $k = \overline{i+1,n}$:

6.1. If $V_{k\ \text{amin}} > V_{k\ \text{amax}}$, then the solution doesn't exist. Stop.

6.2. $V_k = V_{k\ \text{amin}} + g(V_{k\ \text{amax}} - V_{k\ \text{amin}})$.

7. Checking the sizes V_j, $j = \overline{1,i}$ values compliance with requirement (4), taking into account the new values of sizes V_k, $k = \overline{i+1,n}$. Because of Step 6.3, they are compliant with requirement (7), that is $V_j \geq V_{j\,\min}$, $j = \overline{1,i}$ occur.

7.1. Checking the V_1 value compliance with requirement (8), that is if the relation $V_1 \leq V_{1\,\max}$ occurs. Thus, for $j = \overline{2,i}$:

7.1.1. If $V_1 < V_j(cx_1 + 1)/[c(x_j - 1) + 1]$, then $j := j + 1$ and repeat Step 7.1.1.

7.1.2. The V_j value is too small. If $g = 1$, then the solution doesn't exist, because the V_j value is the maximal possible one. Stop.

7.1.3. Increasing the V_j value, but assigning to it a minimal possible value with refer to V_1, that is $V_j = \lfloor V_1[c(x_j - 1) + 1]/(cx_1 + 1) \rfloor + 1$. If $V_j < V_{j-1}\, x_j/x_{j-1}$, then $j := j + 1$ and go to Step 7.1.1.

7.1.4. The V_{j-1} value is too small. If $j = 2$, then the solution doesn't exist, because $V_{j-1} = V_1$. Stop.

7.1.5. Increasing the V_{j-1} value (and may be those of the V_j, $j = \overline{2,i-2}$ ones), but assigning to it a minimal possible value with refer to V_j. Thus, for $k = \overline{j-1,2}$ in the decreasing order:

7.1.5.1. $V_k = \lfloor V_{k+1}\, x_k/x_{k+1} \rfloor + 1$. If $V_k < V_{k-1}\, x_k/x_{k-1}$, then $j := j + 1$ and go to Step 7.1.1.

7.1.5.2. The V_{k-1} value is too small. If $k > 2$, then $k = k - 1$ and go to Step 7.1.5.1.

7.1.5.3. If $V_2 \geq V_1 x_2/x_1$, then the solution doesn't exist because $V_{i-1} = V_1$. Stop.

7.2. Checking the V_j, $j = \overline{2,i}$ values compliance with requirement (6), that is if the relations $V_j \leq V_{j\,\max}$, $j = \overline{2,i}$ occur. Thus, for $j = \overline{2,i}$:

7.2.1. If $V_j < V_{j\,\max}$, where $V_{j\,\max}$ is determined according to (8), then $j := j + 1$ and repeat Step 7.2.1.

7.2.2. The V_{j+1} value is too small. If $g = 1$, then the solution doesn't exist, because the V_{j+1} value is the maximal possible one. Stop.

7.2.3. Increasing the V_{j+1} value, but assigning to it a minimal possible value with refer to V_j, that is $V_{j+1} = \lfloor V_j\ [c(x_{j+1} - 1) + 1]/(cx_j + 1) \rfloor + 1$. If $V_{j+1} < V_j\, x_{j+1}/x_j$, then then $j := j + 1$ and go to Step 7.2.1.

7.2.4. The V_j value is too small. Increasing the V_j value (and may be those of V_k, $k = \overline{2,i-2}$ ones), but assigning to it a minimal possible value with refer to V_{j+1}. Thus, for $k = \overline{j,i}$:

7.2.4.1. $V_k = \lfloor V_{k+1}\, x_k/x_{k+1} \rfloor + 1$. If $V_k < V_{k-1}\, x_k/x_{k-1}$, then $j := j + 1$ and go to Step 7.2.1.

7.2.4.2. The V_{k-1} value is too small. If $k > 2$, then $k = k - 1$ and go to Step 7.2.4.2.

7.2.4.3. If $V_2 \geq V_1 x_2/x_1$, then the solution doesn't exist because $V_{j-1} = V_1$. Stop.

8. If $i < n$, then $i := i + 1$ and go to Step 5.

9. The values of V_i, $i = \overline{1,n}$ were found. If $g = 0$, then $U_i = V_i$, $i = \overline{1,n}$; $g = 1$ and go to Step 2.

10. If $g = 1$, then $V_i := [(V_i + U_i)/2]$, $i = \overline{1,n}$. Stop.

The obtained values of V_i, $i = \overline{1,n}$ can be checked by applying the respective APP method.

Algorithm A3 was implemented in the computer application SIMAP. Examples 3 and 4 using SIMAP are described below.

Example 3 of generating of a Sainte-Laguë method apportionment which fully favors small beneficiaries. Initial data: $c = 2$; $M = 279$; $n = 20$; $V_1 = 20000$; $g = 0$; the x_i, $i = \overline{1,n}$ values are the same as for Example 1. Some results of calculations using SIMAP are systemized in Table 3.

Table 3 - Calculations for the apportionment to Example 3

i	V_i	x_i	$10^{-7}x_i/V_i$	i	V_i	x_i	$10^{-7}x_i/V_i$	i	V_i	x_i	$10^{-7}x_i/V_i$	i	V_i	x_i	$10^{-7}x_i/V_i$
1	20000	30	15000	6	12787	20	15641	11	8197	13	15859	16	3607	6	16634
2	17378	27	15537	7	12132	19	15661	12	6886	11	15974	17	2296	4	17422
3	16066	25	15561	8	11476	18	15685	13	6230	10	16051	18	1640	3	18293
4	14755	23	15588	9	10164	16	15742	14	4919	8	16263	19	984	2	20325
5	14099	22	15604	10	8853	14	15814	15	4263	7	16420	20	328	1	30488

Data of Table 3 were checked – the apportionment is a Sainte-Laguë method's one. At the same time it complies with requirements (2). Thus, it fully favors small beneficiaries. One has $V(g=0) = 177060$, $\Delta M(g=0) = 10$. Also, as expected, it takes place $x_i > a_i$ if and only if $i \geq K$, where $K = \min\{j = \overline{1,n} \mid x_j > a_j\}$. For Example 2 at $g = 0$, one has $K(g=0) = 10$. Also, it takes place $a_1 = 31 > x_1 = 30$. So, one has: $x_1 = a_1 - 1$; $x_i = a_i$, $i = \overline{2,9}$ and $x_i = a_i + 1$, $i = \overline{10,20}$. With refer to particular beneficiaries, eleven beneficiaries ($i = \overline{10,20}$) are favored, and nine beneficiaries ($i = \overline{1,9}$) are disfavored. Moreover the first beneficiary is strongly disfavored because of $x_1 < a_1$.

It was identified that depending of the g value, the results can strongly differ. For example, the results of calculations for initial data of Example 3, but for the case of $g = 1$ (**Example 4**), using SIMAP are systemized in Table 4.

Table 4 - Calculations for the apportionment to Example 4

i	V_i	x_i	$10^{-7}x_i/V_i$	i	V_i	x_i	$10^{-7}x_i/V_i$	i	V_i	x_i	$10^{-7}x_i/V_i$	i	V_i	x_i	$10^{-7}x_i/V_i$
1	20000	30	150000	6	13330	20	150038	11	8663	13	150063	16	3996	6	150150
2	17999	27	150008	7	12663	19	150043	12	7330	11	150068	17	2663	4	150207
3	16665	25	150015	8	11996	18	150050	13	6663	10	150083	18	1997	3	150225
4	15331	23	150023	9	10663	16	150052	14	5330	8	150094	19	1331	2	150263
5	14664	22	150027	10	9330	14	150054	15	4663	7	150118	20	665	1	150376

Data of Table 4 were checked – the apportionment is a Sainte-Laguë method's one. At the same time it complies with requirements (2). Thus, it fully favors small beneficiaries. One has $V(g=1) = 185942$, $\Delta M(g=1) = 13$. Also, as expected, it takes place $x_i > a_i$ if and only if $i \geq K$, where $K = \min\{j = \overline{1,n} \mid x_j > a_j\}$. For Example 4, one has $K(g=1) = 8$. So, one has: $x_i = a_i$, $i = \overline{1,7}$ and $x_i = a_i + 1$, $i = \overline{8,20}$. Thus, with refer to particular beneficiaries, thirteen beneficiaries ($i = \overline{8,20}$) are favored, and seven

beneficiaries ($i = \overline{1,7}$) are disfavored.

Based on data in Tables 3 and 4 and on other results of calculations using SIMAP, may be concluded that for A3 algorithm, as for the A2 algorithm, the lower the *g* value, the lower the values of *V*, ΔM and of the number of beneficiaries that are favored ($M - K + 1$). So, one has:

$V(g=0) = 177060 < V(g=0.5) = 181494 < V(g=1) = 185942$;

$\Delta M(g=0) = 11 = \Delta M(g=0.5) = 11 < \Delta M(g=1) = 13$;

$K(g=0) = 10 = K(g=0.5) = 10 < K(g=1) = 8$.

Conclusions

In order to determine linear divisor methods' apportionments which fully favor beneficiaries, the algorithms A2 and A3 were elaborated. They guarantee the solution (if it exists), regardless of the value of *n*. The A2 algorithm is for the generation of apportionments which fully favor large beneficiaries and A3 algorithm is for the generation of apportionments which fully favor small beneficiaries. These two algorithms are implemented in the computer application SIMAP. Four examples of calculations at $n = 20$ using SIMAP are described:

- Example 1 of a d'Hondt method's apportionment which fully favors large beneficiaries – the V_i, $i = \overline{1,n}$ values were determined at $g = 0$;
- Example 2 of a d'Hondt method's apportionment which fully favors large beneficiaries – the V_i, $i = \overline{1,n}$ values were determined at $g = 1$;
- Example 3 of a Sainte-Laguë method's apportionment which fully favors small beneficiaries – the V_i, $i = \overline{1,n}$ values were determined at $g = 0$;
- Example 4 of a Sainte-Laguë method's apportionment which fully favors small beneficiaries – the V_i, $i = \overline{1,n}$ values were determined at $g = 1$.

All four obtained apportionments fully favor beneficiaries even if the *n* value is relatively large ($n = 20$). In this context, it should be noted that in all 25 million variants of initial data with $n = 20$, for which the V_i, $i = \overline{1,n}$ values were generated stochastically at uniform distribution, none of the apportionments obtained using the SIMAP application [15] does not fully favor the beneficiaries.

At the same time, it was identified that the results of calculations depends considerably not only on the initial data V_1 and x_i, $i = \overline{1,n}$, but also on the parameter *g* value of algorithms A2 and A3. For both algorithms, the lower the *g* value ($0 \leq g \leq 1$), the lower the values of variable *V* and of the number of beneficiaries that are favored.