Sebastian Maier, Petur Zachariassen, Martin Zachariasen

Divisor-Based Biproportional Apportionment in Electoral Systems: A Real-Life Benchmark Study

Preprint Nr. 039/2007 — 23. November 2007

Institut für Mathematik, Universitätsstraße, D-86 135 Augsburg http://www.math.uni-augsburg.de/

Impressum:

Herausgeber:
Institut für Mathematik
Universität Augsburg
86135 Augsburg
`http://www.math.uni-augsburg.de/forschung/preprint/`

ViSdP:
Sebastian Maier
Institut für Mathematik
Universität Augsburg
86135 Augsburg

Divisor-Based Biproportional Apportionment in Electoral Systems: A Real-Life Benchmark Study

S. Maier[*] P. Zachariassen[†] M. Zachariasen[‡]

November 19, 2007

Abstract

Biproportional apportionment methods provide two-way proportionality in electoral systems where the electoral region is subdivided into electoral districts. The problem is to assign integral values to the elements of a matrix that are proportional to a given input matrix, and such that a set of row- and column-sum requirements are fulfilled. In a divisor-based method for biproportional apportionment the problem is solved by computing appropriate row- and column-divisors, and by rounding the quotients. We present a comprehensive experimental evaluation of divisor-based biproportional apportionment in an electoral system context. Firstly, we study the practical performance of a range of algorithms by performing experiments on real-life benchmark instances (election data with multi-member districts). Secondly, we evaluate the general quality of divisor-based apportionments with respect to, e.g., deviation from quota, reversal orderings and occurrences of ties.

1 Introduction

Methods of proportional apportionment translate electoral votes into parliamentary seats in such a way that the number of seats given to a party is in proportion to the vote counts of the party. In many electoral systems, the electoral region is subdivided into electoral districts. In these systems the distribution of seats should both be in proportion to party vote counts and to district magnitudes, i.e., the number of seats allocated to the electoral districts. District magnitudes are usually determined by population counts (as for the US House of Representatives [9, 16]) and/or by other measures such as geographical density (as for the Danish Folketing [15]).

Formally, we are given an $n \times m$ matrix $\mathbf{P} = (p_{ij})$ of nonnegative integers, where p_{ij} denotes the number of votes given to party i in district j. The total number of votes given to party i is $p_{i*} := \sum_j p_{ij}$, and the total number of votes cast in district j is $p_{*j} := \sum_i p_{ij}$. Based on the party (or row) sums $(p_{1*}, p_{2*}, \ldots, p_{n*})$ and the given house size h, the distribution of seats

[*]University of Augsburg, e-mail: Sebastian.Maier@Math.Uni-Augsburg.De.
[†]University of the Faroe Islands, e-mail: peturz@skulin.fo.
[‡]Department of Computer Science, University of Copenhagen, e-mail: martinz@diku.dk.

to parties $\mathbf{r} = (r_1, r_2, \ldots, r_n)$ is computed by using any of the classical vector apportionment methods (see [9] for an historic and mathematical treatment of these methods). Similarly, the district magnitudes $\mathbf{c} = (c_1, c_2, \ldots, c_m)$ are computed based on, e.g., district population counts. Note that we have $\sum_i r_i = \sum_j c_j = h$.

In *biproportional apportionment* the problem is to assign seat numbers to each party within each district such that the party (row-sum) requirements and district (column-sum) requirements are fulfilled. Formally, we should compute an $n \times m$ matrix $\mathbf{X} = (x_{ij})$ of nonnegative integers where $x_{i*} := \sum_j x_{ij} = r_i$ and $x_{*j} := \sum_i x_{ij} = c_j$ for all i, j. This should furthermore be done in such a way that the seat numbers x_{ij} are proportional to the vote counts p_{ij} for all i, j.

Variants and properties of biproportional apportionment were studied by Anthonisse [1], De Meur et al [31] and Gassner [22, 23]. Gassner showed that in the problem with pre-specified marginals, it is impossible to guarantee proportionality at the local level (e.g., within districts) unless there are at most two parties *and* at most two districts.

A biproportional apportionment can be computed using a divisor-based (or multiplier-based) method as suggested by Balinski and Demange [6]. This method works by computing row multipliers λ_i and column multipliers μ_j, such that

$$x_{ij} = [\lambda_i p_{ij} \mu_j], \ \ \forall (i, j)$$

and such that the row- and column-sum requirements are fulfilled; the function $[\cdot]$ rounds a fractional number q to either $\lfloor q \rfloor$ or $\lceil q \rceil$. Balinski and Demange [6] showed that divisor-based methods have several important and unique properties that make them well-suited for use in electoral systems.

The Zürich Canton Parliament recently chose to use divisor-based biproportional apportionment for the distribution of its seats; consequently, the same apportionment system is now used for the Zürich City Council [36]. In Table 1 the vote numbers and resulting seat numbers for the Zürich City Council election on February 12, 2006 are shown: This election was the first time ever that a divisor-based biproportional apportionment method was used for distributing parliament seats. Applications to other parliaments, including comparisons to alternative apportionment methods, are studied in [3, 4, 8, 11, 28, 42]. A software package named BAZI is available for computing vector and matrix apportionments using divisor-based methods [29, 34].

The purpose of our paper is to evaluate the applicability of divisor-based biproportional apportionment in electoral systems. Our benchmark study is based on simulation: Given a set of actual election results (vote counts), we randomly generate realistic (but artificial) election data, and use biproportional apportionment to assign the seats. This allows us to estimate *both* the expected behavior of the tested *algorithms*, and the average *quality* of seat distributions obtained from divisor-based biproportional apportionment. Furthermore, this makes it possible to estimate how often extreme situations appear — such as the occurrence of ties.

The use of extensive computer simulation for benchmarking electoral systems is still in its infancy. Benoit [10] examines some of the classical electoral formula using simulated election results generated from actual vote distributions. Christensen [12] describes a simulation approach for comparing six different single-winner voting procedures by using five different criteria. Fragnelli et al [17, 18] argue strongly for the use of simulation for the choice of an electoral system,

District Party 125	"1+2" 12	"3" 16	"4+5" 13	"6" 10	"7+8" 17	"9" 16	"10" 12	"11" 19	"12" 10	Divisor $1/\lambda_i$
SP 44	28518-4	45541-7	26673-5	24092-4	61738-5	42044-6	35259-4	56547-6	13215-3	1.006
SVP 24	15305-2	22060-3	8174-2	9676-1	27906-2	31559-4	19557-3	40144-4	10248-3	1.002
FDP 19	21833-3	10450-1	4536-1	10919-2	51252-5	12060-2	15267-2	19744-2	3066-1	1.010
Greens 14	12401-2	17319-3	10221-2	8420-1	25486-2	9154-1	9689-1	12559-1	2187-1	0.970
CVP 10	7318-1	8661-1	4099-1	4399-1	14223-1	11333-1	8347-1	14762-2	4941-1	1.000
EVP 6	2829-0	2816-0	1029-0	3422-1	10508-1	9841-1	4690-1	11998-2	0-0	0.880
AL 5	2413-0	7418-1	9086-2	2304-0	5483-1	2465-0	2539-0	3623-1	429-0	0.800
SD 3	1651-0	3173-0	1406-0	1106-0	2454-0	5333-1	1490-0	6226-1	2078-1	1.000
Divisor $1/\mu_j$	7000	6900	5000	6600	11200	7580	7800	9000	4000	

Table 1: Zürich City Council election on February 12, 2006. Table entries are of the form $p - x$, where p is the number of party votes in the district and x is the number of seats apportioned to that party's list in the district. Once the result is known, it can be verified by each voter in a quite simple way: The party ballot p is divided by the associated district and party divisors, and then rounded in the standard way to obtain x. The divisors are those that were published by the Zürich City administration (`http://daten.wahlen.stzh.ch/`). In district "1+2" the Greens had 12401 ballots and were awarded by 2 seats, since $12401/(7000 \times 0.97) \approx 1.83 \nearrow 2$.

and provide results for a case study where ten electoral systems are compared. Maier [28] compares a number of algorithmic variants for computing divisor-based biproportional apportionment, and Zachariasen and Zachariasen [42] compare eight different electoral systems for biproportional apportionment using computer simulations.

The first goal of our paper is to compare a range of algorithms for computing divisor-based biproportional apportionments. We present the algorithms and their asymptotic complexities in Section 2. The algorithms are tested on (randomly perturbed) election data from Denmark, Faroe Islands, Germany, Italy, Switzerland, and on difficult constructed problem instances that are described in Section 3. In this study we only consider problem instances where at least one district — and in most cases all districts — are *multi-member* districts. The application of divisor-based biproportional apportionment to electoral systems with single-member districts (such as UK or US) was recently suggested by Balinski [2]; however, single member-districts involve completely different aspects than problems with multi-member districts, so in this paper only problem instances with multi-member districts are considered.

As a second goal we would like to experimentally evaluate the *quality* of divisor-based biproportional apportionments (which are independent of the algorithm used to compute them). The quality is related to how close the seat numbers are to the ideal fair shares (or "quotas") as given by the vote counts. As there exists no single measure of quality, we present several meaningful measures in Section 4.

Computational results are presented in Section 5. We have chosen *not* to make comparisons with results obtained using quota-based methods such as controlled rounding [13]. The reason is that all quota-based methods have certain weaknesses that make them unsuitable for use in electoral systems [9]. Also, divisor-based methods have the advantage over controlled rounding that it is easy to *verify* a solution by paper and pencil once the divisors are known.

2 Algorithms for divisor-based biproportional apportionment

The problem we would like to solve is to compute a set of row multipliers λ_i, $i = 1, \ldots, n$, and column multipliers μ_j, $j = 1, \ldots, m$, such that if we set $x_{ij} = [\lambda_i p_{ij} \mu_j]$, then all row-sum requirements $x_{i*} = r_i$ and column-sum requirements $x_{*j} = c_j$ are fulfilled. A feasible set of multipliers exists if and only if the row- and column-sum requirements can be met by any matrix $\mathbf{X} = (x_{ij})$. Testing for feasibility can be performed in polynomial time by solving a maximum flow problem [5]. In the following we therefore assume that the problem is feasible. (This is a reasonable assumption, since all real-life problem instances considered in our simulation were in fact feasible.) Furthermore, as shown by Balinski and Demange [5], if a solution $\mathbf{X}$ given by some row- and column-multipliers exists, then this solution is *unique* — except when there is a tie in the rounding function $[\cdot]$, see also [41].

The general approach employed by all algorithms is iteratively to adjust the multipliers until the row- and column-sum requirements are fulfilled. For any set of multipliers and corresponding solution $\mathbf{X}$, the *error function*

$$\Delta(\mathbf{X}) = \frac{1}{2} \sum_i |x_{i*} - r_i| + \frac{1}{2} \sum_j |x_{*j} - c_j|$$

is iteratively minimized. When $\Delta(\mathbf{X}) = 0$, a valid set of multipliers has been found, and the algorithm terminates.

In this section we first introduce some technical notation concerning the rounding function (Section 2.1). Then we briefly discuss algorithms for solving the *vector* problem using a divisor-based method (Section 2.2); this problem appears as a subproblem in the algorithms for the biproportional problem. In Sections 2.3-2.6 we introduce the key algorithms and hybrids evaluated in this paper. Finally, in Section 2.7 we briefly present some recent theory and new algorithms for solving the divisor-based biproportional apportionment problem.

2.1 Rounding function

The rounding function $[\cdot]$ is based on a *signpost sequence* $d(z)$ that maps an integer z to a fractional number in the interval $[z, z+1]$. For a fractional number q, we compute $[q]$ as follows. If $q < d(\lfloor q \rfloor)$ then $[q] = \lfloor q \rfloor$, and if $q > d(\lfloor q \rfloor)$ then $[q] = \lceil q \rceil$. When $q = d(\lfloor q \rfloor)$, then $[q]$ can either be $\lfloor q \rfloor$ or $\lceil q \rceil$. Note that we have $d(x-1) \leq q \leq d(x)$ when $x = [q]$. Choosing $d(z) = z + 0.5$ results in standard rounding of fractions, where numbers with fractional parts greater than 0.5 are rounded up while numbers with fractional parts less than 0.5 are rounded down. In addition to the above, the signpost sequence should fulfill some technical conditions [7, 9].

In the following we present all algorithmic details for a general rounding function (i.e., signpost sequence). However, in our experiments we will only use *standard rounding*. For vector apportionment standard rounding is associated with the names Webster and Sainte-Laguë.

2.2 Vector apportionment

As a subproblem in our algorithms to compute matrix apportionments, we need to compute vector apportionments: Given an n-vector $\mathbf{p} = (p_i)$ of non-negative integers and a positive integer h, find a multiplier $\lambda > 0$ such $\sum_i x_i = h$, where $x_i = [\lambda p_i]$ for $i = 1, \ldots, n$. For illustrative purposes, we will refer to p_i (resp. x_i) as the number of votes (resp. seats) given to party i.

This problem is typically solved in one of two ways. In the first method, we guess an appropriate multiplier $\lambda > 0$, and set $x_i = [\lambda p_i]$ for all parties i. If $\sum_i x_i = h$, then a valid multiplier has been found, and the algorithm terminates. If $\sum_i x_i < h$, then λ is iteratively increased until the sum of seats becomes h. Similarly, if $\sum_i x_i > h$, then λ is iteratively decreased until the correct number of seats is obtained.

The initial error of the so-called canonical multiplier $\lambda = h / \sum_i p_i$ was studied by Happacher and Pukelsheim [24, 25]; the canonical multiplier has zero expected error if standard rounding is used. (Happacher and Pukelsheim also gave a canonical multiplier for every stationary rounding method, i.e., rounding methods with signpost sequences $z + q$, $q \in [0, 1]$, and $z = 0, 1, \ldots$) Dorfleitner and Klein [14] showed that the total running time is $O(n^2)$ if the canonical multiplier is used as the initial guess; by using a better implementation, a running time of $O(n \log n)$ can be achieved [41].

Another approach for computing a vector apportionment — which is very common when solving the problem by hand (see also [9]) — is to list the fractions

$$\begin{array}{l} p_1/d(0),\ p_1/d(1),\ p_1/d(2), \ldots \\ p_2/d(0),\ p_2/d(1),\ p_2/d(2), \ldots \\ \vdots \\ p_n/d(0),\ p_n/d(1),\ p_n/d(2), \ldots \end{array}$$

(Note that the fractions in each row are strictly decreasing.) The vector apportionment problem can now be solved by picking the h largest fractions in these n rows. This procedure corresponds to starting with $\lambda = 0$ and iteratively increasing λ until the correct number of seats is allocated. A trivial implementation in which we iteratively pick the largest fraction, and remove this fraction from the corresponding row, requires $O(hn)$ time; by picking the largest fraction more efficiently, we can reduce the running time to $O(n + h \log n)$.

A hybrid method was suggested by Rote and Vogel [37]. In this method λ is first chosen such that $h - n \leq \sum_i x_i \leq h$, where $x_i = [\lambda p_i]$ for $i = 1, \ldots, n$. Then the fractions

$$\begin{array}{l} p_1/d(x_1),\ p_1/d(x_1 + 1),\ p_1/d(x_1 + 2), \ldots \\ p_2/d(x_2),\ p_2/d(x_2 + 1),\ p_2/d(x_2 + 2), \ldots \\ \vdots \\ p_n/d(x_n),\ p_n/d(x_n + 1),\ p_n/d(x_n + 2), \ldots \end{array}$$

are used to allocate the (up to) n remaining seats. Picking the largest (up to) n elements in n sorted rows can be solved in $O(n)$ time by using an algorithm by Frederickson and Johnson [19]. Hence the complete vector apportionment problem can be solved in $O(n)$ time.

A solution to the vector apportionment problem can be characterized in an alternative way using a so-called *max-min-inequality* [9]. An assignment of seat numbers $\mathbf{x} = (x_i)$ is a valid vector apportionment if and only if $\sum_i x_i = h$ and

$$\max_{p_i>0} \frac{d(x_i - 1)}{p_i} \leq \min_{p_i>0} \frac{d(x_i)}{p_i}$$

In fact, this inequality induces a feasible interval of valid multipliers: Any multiplier λ in the interval

$$\left[\max_{p_i>0} \frac{d(x_i - 1)}{p_i},\ \min_{p_i>0} \frac{d(x_i)}{p_i}\right]$$

results in a feasible vector apportionment.

2.3 Iterative proportional fitting (IPF)

The *matrix scaling* problem is the following: Given an $n \times m$ matrix $\mathbf{P} = (p_{ij})$, row-sum requirements r_i and column-sum requirements c_j, compute row multipliers λ_i and column multipliers μ_j, such that the matrix $\mathbf{X} = (x_{ij})$ given by $x_{ij} = \lambda_i p_{ij} \mu_j$ fulfills the given row- and column-sum requirements. Therefore, in matrix scaling no rounding is performed of the values $\lambda_i p_{ij} \mu_j$, and thus the problem may be seen as a continuous version of the (discrete) divisor-based biproportional apportionment problem.

Matrix scaling has numerous applications in statistics, numerical analysis and operations research (see, e.g., [3, 27] and references herein). The problem is usually solved using the *iterative proportional fitting (*IPF*)* algorithm, which alternatively scales the rows and columns to meet the desired row- and column-sums: In even iterations we choose λ_i such that $\sum_j \lambda_i p_{ij} \mu_j = \lambda_i(\sum_j p_{ij} \mu_j)$ is equal to r_i for each row i. Similarly, in odd iterations we choose μ_j such that $\sum_i \lambda_i p_{ij} \mu_j = \mu_j(\sum_i \lambda_i p_{ij})$ is equal to c_j for each column j. Each iteration clearly takes $O(nm)$ time, and as shown by Sinkhorn [39], this algorithm converges ($\Delta(\mathbf{X}) \to 0$). Convergence can, however, be slow if the numbers in $\mathbf{P}$ are of very different magnitude. If $L(\mathbf{P})$ denotes the logarithm of the ratio of the largest to the smallest nonzero entry in $\mathbf{P}$, then there exist examples where IPF needs $\Omega(L(\mathbf{P})/\epsilon)$ iterations to achieve an accuracy of $\Delta(\mathbf{X}) \leq \epsilon$ for $\epsilon > 0$. Nevertheless, IPF typically converges much faster, and is therefore the method of choice for practical applications. It should be noted that theoretically faster algorithms for solving the matrix scaling algorithm exist [27, 38], but these are significantly more complex than IPF, and in most cases not faster than IPF in practice.

In this paper we will use IPF as a "preprocessing" step in some of our algorithms to achieve good *guesses* on the multipliers before starting the algorithms for the discrete biproportional apportionment problem. (A good guess for the multipliers is one for which $\Delta(\mathbf{X})$ is small.)

2.4 Alternating scaling (AS)

The discrete version of the IPF algorithm is named *alternating scaling (*AS*)* [3]. In alternating scaling, we iteratively solve vector apportionment problems for the rows and columns:

- In even iterations a vector apportionment problem with weights $(p_{i1}\mu_1, p_{i2}\mu_2, \ldots, p_{im}\mu_m)$ and house size r_i is solved for each row i. This gives a set of updated row-multipliers λ_i.
- In odd iterations a vector apportionment problem with weights $(\lambda_1 p_{1j}, \lambda_2 p_{2j}, \ldots, \lambda_n p_{nj})$ and house size c_j is solved for each column j. This gives a set of updated column-multipliers μ_j.

Clearly, the vector apportionment problem for row i is only solved if the current seat numbers $(x_{i1}, x_{i2}, \ldots, x_{im})$ do not have the correct sum r_i — and similarly for the columns.

The running time per iteration is $O(nm)$, since we can solve the vector apportionment problem in linear time (see Section 2.2). It is easy to see that the error $\Delta(\mathbf{X})$ cannot increase from one iteration to the next, but $\Delta(\mathbf{X})$ may remain *constant* for a number of iterations. In fact, there exist instances where AS has cyclic behavior and therefore does *not* terminate. Extensive simulations indicate that cycling only occurs if there are ties in the rounding function, that is, when multiple solutions exist for the biproportional apportionment problem [3].

The max-min-inequality for vector apportionment induces an interval of valid multipliers in each iteration. Hence, besides the case of a tie, a valid multiplier could be chosen from the interval $[\lambda_i^{min}, \lambda_i^{max}]$ for rows and $[\mu_j^{min}, \mu_j^{max}]$ for columns. In the sequel three strategies are investigated:

1. *Midpoint multiplier* (ASMDPT): Choose a multiplier from the middle of the interval.
2. *Extreme multiplier* (ASEXTR): Choose λ_i^{min} for overrepresented rows, λ_i^{max} for underrepresented rows, μ_j^{min} for overrepresented columns, and μ_j^{max} for underrepresented columns.
3. *Random multiplier* (ASRAND): Choose a divisor randomly from $[\lambda_i^{min}, \lambda_i^{max}]$ for rows, and from $[\mu_j^{min}, \mu_j^{max}]$ for columns.

The extreme divisor is proposed to remove as much as possible from the underlying continuous weights of an overrepresented row or column and to give as much as possible to the underlying continuous weights of an underrepresented row or column.

2.5 Tie-and-transfer (TT)

The so-called tie-and-transfer (TT) algorithm for computing divisor-based biproportional apportionments was proposed by Balinski, Demange and Rachev [5, 7]. In each iteration of this algorithm, the error $\Delta(\mathbf{X})$ is decreased by exactly 1, so in contrast to the AS algorithm, the TT algorithm is guaranteed to converge. In the following description we assume that the columns of $\mathbf{X}$ continuously add up correctly, so that the error solely comes from over- and underrepresented rows. One way to achieve this is to set $\lambda_i = 1$ for all rows, and solve the vector apportionment problem for each column. This takes $O(nm)$ time and guarantees that the initial error is at most h.

The idea is now to search for path in the row/column graph $G = (I \cup J, E)$, which is a bipartite directed graph with a vertex set consisting of the rows I and columns J of $\mathbf{X}$. The set of edges E is the union of:

1) For each $i \in I$ and $j \in J$ where $p_{ij} > 0$ and $\lambda_i p_{ij} \mu_j = d(x_{ij})$ we have an edge $(i, j) \in E$. This means that x_{ij} may be *rounded up* without changing the multipliers. (Recall that we have $d(x_{ij} - 1) \leq \lambda_i p_{ij} \mu_j \leq d(x_{ij})$ for all i, j.)

2) For each $i \in I$ and $j \in J$ where $p_{ij} > 0$ and $\lambda_i p_{ij} \mu_j = d(x_{ij} - 1)$ we have an edge $(j, i) \in E$. This means that x_{ij} may be *rounded down* without changing the multipliers.

Informally, we have an edge (i, j) out of row i for each column j where x_{ij} can be rounded up. Similarly, we have an edge (j, i) out of column j for each row i where x_{ij} can be rounded down. Note that for any pair of vertices i, j, there is at most one directed edge that connects them.

Assume that G contains a simple path $p = \{(i_1, j_1), (j_1, i_2), (i_2, j_2) \ldots, (j_{k-1}, i_k)\}$. The path p defines a ***transfer*** by rounding up $x_{i_1 j_1}$, rounding down $x_{i_2 j_1}$, rounding up $x_{i_2 j_2}$ etc. and finally rounding down $x_{i_k j_{k-1}}$. The net effect of this transfer is that it increases the i_1'th row sum by 1 and decreases the i_k'th row sum by 1. No other row sums and no column sums are affected by the transfer.

Let I^- be the set of underrepresented rows (where $x_{i*} < r_i$), and let I^+ be the set of overrepresented rows (where $x_{i*} > r_i$). The main idea of the TT algorithm is to search for a path in G from a vertex in I^- to a vertex in I^+. By rounding the elements x_{ij} corresponding to the edges of such a path up and down, the error function $\Delta(\mathbf{X})$ is decreased by exactly 1. If no such path exists, the multipliers — and hence the row/column graph — are updated in a so-called *tie update* step. After at most $n + m$ tie update steps, a path from a vertex in I^- to a vertex in I^+ is found. The running time complexity of each main iteration of the TT algorithm (in which one transfer is made) is $O(nm(n + m))$ [41]. Rote and Zachariasen [38] showed that the running time of an iteration of the TT algorithm can be reduced to $O(nm)$, making TT asymptotically as fast per iteration as IPF and AS.

The main weakness of the original TT algorithm is that it always requires $\Delta(\mathbf{X_0})$ iterations to converge, where $\mathbf{X_0}$ is the initial seat distribution. For large house sizes h, this can be very time-consuming. Zachariasen [41] showed how to reduce the linear dependence of h to a logarithmic dependence of h.

2.6 Hybrid algorithms

The hybrid algorithms combine two of the above mentioned algorithms to combine the advantages of the single ones. AS is usually fast in the beginning, but the convergence may be very slow towards the end. The original TT algorithm has to reduce the error one unit at a time, even if the error is very large. Balinski and Demange [5] suggest to use the solution to the continuous problem (IPF) as starting point for TT. Experiments [28] indicate that the combinations AS–TT and IPF–TT perform best. The challenge is to implement an appropriate switching condition.

For the combination AS–TT we check at the end of every column step for a switch. If the error function stays the same for two iterations (after fitting the columns), we continue with TT. It would be possible to choose more adaptive switching rules, but this has little influence in practice. AS is used in the versions ASMDPT, ASEXTR and ASRAND. For the combination IPF–TT, we switch when the error drops below 5; after fitting the columns, we continue with TT. The switching point of 5 for IPF–TT was chosen to stay away from or not to run into possible slow convergence for IPF.

2.7 Recent developments

Anthonisse [1] pointed out that biproportional apportionment may be solved using network flows, if the problem can be stated as an optimization problem with an objective function that measures the "difference" between $\mathbf{P}$ and $\mathbf{X}$. Gaffke and Pukelsheim [20] essentially presented such an optimization framework, and finally Rote and Zachariasen [38] showed that divisor-based biproportional apportionment can be solved by classical minimum-cost flow. This opens up a range of new algorithms that may be applied to the problem in the future.

3 Benchmark instances

The set of benchmark instances consists mainly of election data from recent European elections. The house sizes range from 32 to 617 members, and the average district sizes range from 4.6 to 25.7. An overview of short-names (including the year of the considered election) and parameters for the benchmark instances is given in Table 2. The electoral systems are ordered by their number of districts (smallest first), and secondly, by their house size (smallest first). By making experiments on instances with a wide range of house/district sizes, we expect to be able to see how the quality of the apportionments depends on these parameters. In order to study the algorithmic performance in more detail, we added a set of so-called "Marathon instances" which are designed to require many iterations by some of the algorithms.

Since only two of the electoral systems considered in this study actually use divisor-based biproportional apportionment today, below we briefly describe how we made it feasible to use biproportional apportionment in the remaining electoral systems studied (Section 3.1). We generated simulated election data by randomly perturbing the real vote numbers (Section 3.2); this generated data formed the input to the investigated apportionment algorithms.

3.1 Detailed description of benchmarks

Faroese Parliament (Fo04)

The electoral system of the parliament of the Faroe Islands, the *Løgting*, can be classified as a List PR system with two-tier districting. At the lower-level 27 members are elected in 7 multi-member districts using the d'Hondt (Jefferson) apportionment method. District magnitudes are 2,2,2,4,4,5,8. Then, at the national level, up to 5 adjustment seats are distributed among parties and districts based on the LR Hare (Hamilton) method. With this two-tier method, the total number of seats can vary from 27 to 32. The electoral threshold is 1/27 for receiving an adjustment seat, while district seats are distributed with the threshold inherent to the d'Hondt method.

In our simulation we fixed the parliament size to 32, and set the district sizes using the Sainte-Laguë (or Webster) method based on the vote counts for the districts from the 2004 election.

Electoral system	Short name	House size	Number of districts	District sizes Avg	Min–Max	Number of participating parties	Number of parties that received seats
Faroese Parliament	**Fo04**	32	7	4.6	$1-12$	7	7
Bavarian Parliament	**Bav03**	180	7	25.7	$17-57$	15	3
Zürich City Council	**ZhCC06**	125	9	13.9	$10-19$	14	8
Danish Folketing	**Dk05**	175	17	10.3	$2-22$	9	8
Zürich Canton Parliament	**ZhCP07**	180	18	10.0	$4-16$	11	9
Swiss National Assembly	**Ch03**	200	26	7.7	$1-34$	43	15
Italian National Assembly	**It06**	617	26	23.7	$3-44$	13	13
Marathon instances	**MOB3T**	$3 \cdot 10^3$	3	10^3	10^3-10^3	3	3
	MOC3T	$3 \cdot 10^3$	3	10^3	10^3-10^3	3	3
	MOD3T	$3 \cdot 10^3$	3	10^3	10^3-10^3	3	3
	MOB3M	$3 \cdot 10^6$	3	10^6	10^6-10^6	3	3
	MOC3M	$3 \cdot 10^6$	3	10^6	10^6-10^6	3	3
	MOD3M	$3 \cdot 10^6$	3	10^6	10^6-10^6	3	3

Table 2: Overview of benchmark instances. For each instance, we give the short name used to identify the instance (which includes the election year), the house size, the number of districts, the distribution of district sizes, the number of parties participating in the election and the number of parties receiving a seat using the biproportional apportionment algorithm on the original election data. (For **It06** the reported number of participating parties is actually the number of parties that received a seat in the election. For **Fo04** only 6 parties got a seat using the current electoral system, and for **Ch03** only 14 parties got a seat using the current electoral system.)

Bavarian Parliament (Bav03)

The Free State of Bavaria, a part of Germany, is divided into 7 administrative districts. Among them the 180 seats of the Bavarian parliament, the *Landtag*, are allocated according to population counts. This leads to 17 seats for the smallest and 57 for the largest district. A five percent threshold in the whole electoral region is required to be passed by the participating parties [26]. In the 2003 elections 3 out of 15 parties passed the threshold. Although the Bavarian system employs a district-wise apportionment, the configuration also can be used for a biproportional apportionment.

Zürich City Council (ZhCC06)

The world debut for a biproportional system was the Zürich City Council election on February 12, 2006. We used the data of this premiere election. The 14 parties are running for the 125 seats at stake in 9 districts; the district magnitudes differ from 10 to 19. Parties participate in the apportionment if they receive at least five percent of the votes in at least one district. We refer to this kind of threshold as a district-wise threshold. In the 2006 election, 8 parties received a seat [36].

Danish Folketing (Dk05)

The unicameral Danish parliament, the *Folketing*, has 179 members, 175 from Denmark proper and two each from the Faroe Islands and Greenland. The members from these two parts of the realm are elected according to separate rules. The following deals with the election of the 175 members from Denmark.

Basically, the electoral system of the Folketing is a so-called "list PR two-tier system" with 135 seats allocated in 17 multi-member districts in the lower tier, and 40 adjustment seats allocated in the upper nationwide tier. Every five years, the 135 district seats are distributed proportionally to the multi-member districts on the basis of the sum of three separate numbers: (1) population, (2) number of registered voters in the latest general election, and (3) area in square kilometers multiplied by 20 (as a measure of population density). In a similar way the 40 adjustment seats are distributed proportionally to three national regions, i.e., groupings of the 17 districts.

In the lower tier the district seats are allocated to parties within each district separately by using the modified Sainte-Laguë formula (divisors $1.4, 3, 5, \ldots$). The upper tier allocation is carried out using a fairly complicated method (see [15] for details). There are three different thresholds: (1) winning a seat directly in any of the 17 multi-member districts, (2) obtaining in two of the three regions a number of votes corresponding to the regional votes/seat ratio (excluding adjustment seats), and (3) obtaining two percent of the valid, national vote. In our simulation we only used the third of these thresholds (an overall 2% threshold).

Because of the way the adjustment seats are allocated within in a region it cannot be foreseen precisely before an election to the Folketing, how the seats will distribute on the 17 districts. Therefore, we used the district sizes given by a fair Sainte-Laguë district magnitude (based on the vote counts from the 2005 election). The differences between a fair Sainte-Laguë district magnitude and the actual district seat distribution in the 2005 election are moderate: for 7 districts there is no difference, for 8 districts there is a one seat difference, and for 2 districts there is a two seat difference.

Zürich Canton Parliament (ZhCP07)

The rules for the Zürich Canton Parliament elections are the same as for the Zürich City Council elections. The 180 seats are allocated to 18 districts of whom the smallest one has 4 (Andelfingen) and the largest one (Bülach) 17 seats. For the last election in 2007 there were 11 competing parties. There is a district-wise five percent threshold [36].

Swiss National Assembly (Ch03)

For the Swiss National Assembly, Bochsler [11] has done some model calculations using divisor-based biproportional apportionment (using Zürich's new electoral system). We used his data from the 2003 elections. Each canton is a district, 26 in total. The smallest have only 1 seat, the largest has 34 seats. There are 43 competing parties. The seats are allocated to the districts in proportion to population counts, such that each canton gets at least 1 seat. In our simulation we used a district-wise five percent threshold as in the Zürich system.

Italian National Assembly (It06)

The new electoral system for Italy is also somehow a biproportional system, although it suffers from some unproportional elements like a majority reward for the winning coalition and some bugs in the implemented seat allocation [32, 33]. We decided to incorporate the data of the 2006 Italian election to see how a correct biproportional seat allocation would work. There are 26 districts, and we used the 13 parties which got seats in the parliament. The district magnitudes differ from 3 (Molise) to 44 (Puglia), and there are 617 seats at stake; the remaining 13 seats in the national assembly are for Italians living abroad.

Marathon instances from BAZI (MO)

These are 3×3 weight matrices that are motivated by the literature on the continuous iterative proportional fitting algorithm [30]. These matrices have large "house sizes" of 3000 (for each district 1000), or 3000000 (for each district 1000000). They are designed to require many iterations.

3.2 Random perturbation of election data

We created 10000 uniform random perturbations in a range $\pm 20\%$ of each weight in the vote matrix for the underlying electoral result for the real-life benchmark instances, and 1000 for the Marathon benchmark instances. Parties which did not pass the threshold in the simulated data were automatically removed. The free statistical package `R` [40] was used for the simulation. Therefore an R-package named RBazi, which is an interface from `R` to the BAZI-program[1] was created. This

[1] `www.uni-augsburg.de/bazi`

package provides full access to all apportionment methods of BAZI and allows to use R for further analysis and the simulation of data.

The simulated data is used to compute the party marginal first, while the district magnitudes are fixed for the entire simulation. With this input and the vote matrix we compute the biproportional apportionment.

4 Quality measures

The quality of an apportionment can be measured in several ways. One approach is to compare the seat numbers with the fair share (or quota). However, for biproportional apportionment the definition of quota is ambiguous, and we describe several definitions and quality measures based on quota in Section 4.1. Another important issue in biproportional apportionment is the existence of so-called *discordant* seat allocations (or seat reversals). The problem is that within districts or within a party, the seat numbers are not necessarily proportional to the votes. Quality measures related to discordant seat allocations are described in Section 4.2.

4.1 Deviation from quota

One important attribute which is investigated in vector problems is the quota. This is the fair share of seats a party should get — or the amount of seats if also fractions of parliamentary seats would be allocated. Let p_i be the number of votes given to party i, and let p_* be the total number of votes given. The quota q_i for a party i is defined as

$$q_i := \frac{p_i}{p_*} \cdot h.$$

Obviously, the quotas sum to the given house size h.

At first glance it is not clear how to extend the concept of a quota to matrix problems. Let the vote count matrix be $\mathbf{P} = (p_{ij})$. We consider four possible quota definitions:

1. The *overall (general) quota* is simply applying the formula for the quota to the whole matrix:

   $$q_{ij}^{(o)} := \frac{p_{ij}}{p_{**}} \cdot h$$

   where $p_{**} = \sum_i \sum_j p_{ij}$. Gassner [23] notes that this general quota is not appropriate if pre-specified row and column marginals have to be fulfilled.

2. In some electoral systems the seats are allocated to the parties separately within each district. In this case the *district quota* is the appropriate measure for the question of whether the apportionment within a district is fair. It is defined as

   $$q_{ij}^{(d)} := \frac{p_{ij}}{p_{*j}} \cdot c_j$$

3. For German federal elections the opposite way is chosen. The seats of each party are allotted to the state lists in proportion to the votes of the lists. Then, the measure for fairness is the *party quota:*
$$q_{ij}^{(p)} := \frac{p_{ij}}{p_{i*}} \cdot r_i$$

4. One of the features of the quota for vector apportionment is that it sums to the given house size. Therefore, another natural approach is to ask for a quota that sums to the given row and column marginals. A candidate for this *fair share quota* is the solution to the continuous matrix scaling problem:
$$q_{ij}^{(f)} := \lambda_i^{(f)} p_{ij} \mu_j^{(f)}$$
for some row and column multipliers $\lambda_i^{(f)}$ and $\mu_j^{(f)}$ such that $\forall i = 1, \ldots, n : \sum_j q_{ij}^{(f)} = r_i$ and $\forall j = 1, \ldots, n : \sum_i q_{ij}^{(f)} = c_j$. Informally, the fair share quota can be obtained by applying the quota formula for vector apportionments iteratively on rows and columns until the given row and column requirements are fulfilled (see Section 2.3).

Note that all quotas sum to the given house size, but in general the overall quota does not sum to the given row and column marginals. The district quota may fail to achieve the pre-specified party-sums, and the party quota fails for the district-sums. Only the fair share quota guarantees that the district and party marginals are fulfilled, but on the downside, it is less intuitive.

In the sequel we focus our attention on the *district* quota and the *fair share* quota. We have chosen the first because the relevant electoral systems either had district-wise apportionment earlier (Canton Zürich), or are systems with district-wise apportionment where it has been considered to introduce a biproportional electoral system (Faroes, Cantons Aargau and Schaffhausen). Henceforth, as the district quota is the yardstick now, it is a good measure for comparison. The fair share quota is chosen for its obvious property of being a real biproportional quota.

Let $\mathbf{Q} = (q_{ij})$ be the quota matrix (defined using one of the definitions above). We count the number of quota breaches as

$$N_{\text{breach}}(\mathbf{Q}, \mathbf{X}) := \sum_i \sum_j \mathbf{1}(\{|x_{ij} - q_{ij}| > 1\}),$$

where $\mathbf{1}(E)$ returns 1 if E is true and 0 otherwise.

Another attribute of interest is the distance between quota and apportionment. A common measure is the L_2-distance, in the context of elections better known as the Gallagher index [21]. First, quota and apportionment is scaled to a total sum of 100, i.e. $\tilde{q}_{ij} = 100 \cdot \frac{q_{ij}}{h}$, and $\tilde{x}_{ij} = 100 \cdot \frac{x_{ij}}{h}$. The Gallagher index is defined as

$$GI(\tilde{\mathbf{Q}}, \tilde{\mathbf{X}}) := \sqrt{\frac{1}{2} \sum_i \sum_j (\tilde{q}_{ij} - \tilde{x}_{ij})^2}$$

For instances with thresholds, we *exclude* from the computation of the Gallagher index all parties which do not pass the threshold. Usually parties with zero seats are included when calculating the Gallagher index, but this would lead to a fair share or party quota of zero for each entry of the quota matrix for parties getting no seats. Our approach ensures the same basis for all quotas.

4.2 Discordant seat allocations

Discordant seat allocations are inevitable for biproportional apportionment [23]. Two measures of discordant seat allocations were considered:

1. Number of districts with discordant seat allocations:

$$ND_{\text{dis}}(\mathbf{P}, \mathbf{X}) := |\{j : \exists i, k \; : \; p_{ij} > p_{kj} \Rightarrow x_{ij} < x_{kj}\}|$$

2. Number of districts having a party getting more seats than the strongest party in this district:

$$ND_{\text{spdis}}(\mathbf{P}, \mathbf{X}) := |\{j : \exists i : p_{ij} < p_{ik} \Rightarrow x_{ij} > x_{ik}, k = \arg\max_{l} p_{lj}\}|$$

We also considered reporting the number of pairwise discordant seat allocations, but this measure would be highly correlated with district sizes. We believe that the important measure is the *number of districts* with discordant seat allocations.

5 Results

The results of our extensive computer simulation are presented in this section. In Section 5.1 we begin by describing a variant of the pure biproportional problem — the so-called strongest party constrained (SPC) rule. For some of the benchmarks considered, adding this constraint has a positive effect on the quality of the computed apportionments. In Section 5.2 we briefly discuss the problem of ties in biproportional apportionment, and basically argue that these appear very seldomly. Results on the performance of the algorithmic variants considered are presented in Section 5.3. Finally, in Section 5.4 we quantify the quality of the computed biproportional apportionments. Detailed results for each benchmark instance can be found in the appendix.

In this section *super*-apportionment refers to the apportionment of the total number of seats to the parties (i.e., computing the row marginals based on total vote counts for parties), while *sub*-apportionment refers to the actual biproportional apportionment. Recall that district sizes (i.e., column marginals) are predefined in all computations.

5.1 Pure and constrained biproportional apportionment

While biproportional electoral systems ensure proportionality for all parties over the whole electoral region, there is also a price to pay for this. Within the districts it might happen that one party receives more seats than another party while having fewer votes. Moreover, such discordant seat allocations are inevitable [23].

A special subproblem of discordant seat allocations arises in districts with few seats: the strongest party gets no seat while other parties get seats. To avoid this case, we added as a constraint to the matrix apportionment that the strongest party in a district should always get at least one seat in this district. Note that in general for our benchmark instances single-member districts are an exception (see Table 2). In the case of single-member districts biproportional apportionment is called fair majority voting [2]. We call the unconstrained version the *pure biproportional rule* and the constrained version the *strongest party constrained biproportional rule (SPC)*.

More formally, for the SPC rule we set $a_{ij} = 1$, if party i is the strongest party in district j, and $a_{ij} = 0$ otherwise. If there is more than one strongest party, each such party gets a guaranteed seat (if there are enough seats in the district — otherwise lots are drawn). Then the row multipliers λ_i and the column multipliers μ_j are chosen such that

$$x_{ij} = \max(a_{ij}, [\lambda_i p_{ij} \mu_j])$$

and such that the row- and column-sum requirements are fulfilled. This does not affect the asymptotic running time per iteration of the investigated algorithms.

A necessary condition for the feasibility of the sub-apportionment for the SPC rule is to include the aggregated constraints for all parties as a restriction for the super-apportionment. It turns out that the constraints for the SPC rule are so weak that the super-apportionments for the pure and the SPC rule were the same throughout our simulation. This seems to indicate that the SPC rule is able to avoid that the strongest party stays unrepresented in a district, while not disturbing the overall proportionality for all parties.

Effects of the SPC rule for the sub-apportionment can be observed for **Fo04** and **Ch03**. Both of these instances have average district sizes below 10 and several small districts. When applying the SPC rule for these two instances, they are denoted **Fo04SPC** and **Ch03SPC**.

5.2 Ties

Two kinds of ties may occur in biproportional electoral systems. The first are ties in the super-apportionment, when multiple apportionments arise during the allocation of seats to the parties. We resolved these ties and computed a sub-apportionment for each of the super-apportionments. The second type of ties are multiple matrix apportionments. It turned out that in our simulation we *never* encountered a tie in the sub-apportionment.

Except for **ZhCC06**, **ZhCP07**, and **Fo04** there were no ties in the super-apportionment. The number of ties for these instances are summarized in Table 3. To give a more intuitive impression on how rare ties actually are, we computed the average number of elections between the occurrence of ties and the average waiting time for such an unusual event assuming a four year electoral period.

Zürich's new electoral system also includes a rounding step when computing the vote counts for the super-apportionment [35, 36]. This might cause ties in the super-apportionment when this special variant of biproportional system is used. When compared to the other problem instances, the vote counts for **Fo04** are somewhat smaller. Combined with our specific simulation design, this also causes more ties.

5.3 Algorithmic performance

In this section we present results on the performance of the biproportional apportionment algorithms that have been implemented and tested (see Section 2). We focus on reporting the number of *iterations* performed by each of the algorithms: For alternating scaling (and IPF), solving the

Instance	Ties	Approximately	
		every ...th election	once in ... years
ZhCC06	1	10000	40000
ZhCP07	3	3333	13300
Fo04	14	714	2860

Table 3: Number of ties and the average time of years for their occurrence assuming a four year electoral period for the benchmark instances with occurrence of ties in the super-apportionment.

vector apportionment problem for each row is one iteration, and solving the vector apportionment problem for each column is another iteration. For tie-and-transfer an iteration is the same as performing a single transfer. Note that for both alternating scaling and tie-and-transfer is it possible to perform an iteration in $O(nm)$ time, i.e., in linear time in the size of the matrix.

The advantage of focusing on the number of iterations is that this number is independent on the computer used for the simulation. Furthermore, the number of iterations is somewhat independent on the size of the matrix; it rather depends on the structure of the matrix and on the house size. However, in order to quantify the actual running times of the tested algorithms, in the next section we start by summarizing the running times *per iteration*.

Running time per iteration

From Table 4 we observe that all algorithms use within the same order of magnitude of CPU-time per iteration — but with tie-and-transfer (TT) being the slowest. Among the alternating scaling variants there is no clear difference, although ASEXTR is little slower on most instances. But we may conclude that all algorithms are pretty fast, and most of them can perform more than 100 iterations per second on the real-life benchmark instances.

In Table 4 we have also attempted to illustrate the dependence of the running time per iteration on the size of the matrix. This dependence is, however, not entirely obvious. For TT the running time per iteration also depends on how "ugly" the multipliers are. Since the vote counts for **Fo04** are small, the final solution is "close" to being a tie, and therefore fractional numbers with large numerators and denominators are often needed for the multipliers in order to distinguish between solutions. In **Ch03** more than half of the elements in the vote matrix are zero, and this is a definite advantage for TT. Otherwise, we must conclude that the size range of the benchmark instances is too small to clearly see the effect on the running time per iteration.

The running times presented here are for our *integer* version of the TT algorithm. This means that all computations are done using exact arithmetic, i.e., by representing the multipliers as exact rational numbers (both the numerator and denominator are represented using arbitrarily large integers). We also implemented a *floating point* version of TT, where the multipliers were represented using 64-bit floating point numbers. The floating point version was on average twice as fast on the real-life instances, but did occasionally fail due to numerical problems.

Algorithm Instance	(Size)	TT	ASMDPT	ASEXTR	ASRAND
Fo04	(49)	10.60	3.63	3.62	3.56
Bav03	(21)	3.60	7.08	6.84	6.83
ZhCC06	(72)	5.28	3.43	3.71	3.54
Dk05	(119)	11.15	3.16	3.84	3.55
ZhCP07	(162)	8.55	3.76	4.70	4.38
Ch03	(390)	9.61	6.67	7.77	7.30
It06	(338)	22.74	5.77	6.76	6.58

Table 4: Running times in *milliseconds per iteration* for the studied algorithms. The size of an instance is the number of elements in the vote matrix, where parties that do not pass the threshold in the biproportional apportionment algorithm on the original election data have been eliminated (see Table 2). Running times are averages over 250 runs using R-system-time profiling. All experiments were made on an Intel P4 (3.0GHz) with 1024 MB of RAM running Suse Linux 9.3.

Alternating scaling

The number of iterations used by ASEXTR and ASRAND on the real-life instances is shown as a series of boxplots in Figure 1 (the results for ASMDPT were somewhere between these two). The boxplots for the extreme multiplier method (ASEXTR) reveil extremely skewed distributions. While most of the the observations are in a range up to 20 iterations, there are many outliers, and some problem instances required almost 5000 iterations. The distributions resulting from the random multiplier method (ASRAND) are skewed, too, but no instances required more than 100 iterations.

An example of the behavior of ASEXTR can be seen in Figure 2 (left), where the error count (as defined in Section 2) is given as a function of the number of iterations. This is the 772nd simulated instance of **ZhCC06**, which requires 1777 iterations to finish. The error count starts at 3 in the first iteration, then stays at 2 for another two iterations. But then there are 1773 iterations needed to reduce the error count from 1 to 0. For the same problem instance, ASMDPT needs only 14 iterations in total.

In Figure 2 (right) the error count decrease for the original data of **MOD3M** using ASMDPT is shown. In total 9473 iterations are required, but most of the time is needed towards the end. It takes 369 iterations to reduce the error count from 5 to 4, 39 iterations from 4 to 3, 617 iterations from 3 to 2, 77 iterations from 2 to 1, and finally, 1883 iterations to reach the solution.

Despite the skewed distributions for ASEXTR, this algorithmic variant has the smallest overall median. For almost 70% of the simulated problem instances ASEXTR either used fewer or equally many iterations as each of the two other variants.

Tie-and-transfer

The number of iterations performed by the tie-and-transfer algorithm is presented in Figure 3 (left). Recall that the number of iterations is bounded by the house size of a problem instance, so the distributions have very small ranges. (The number of iterations is identical to the initial error

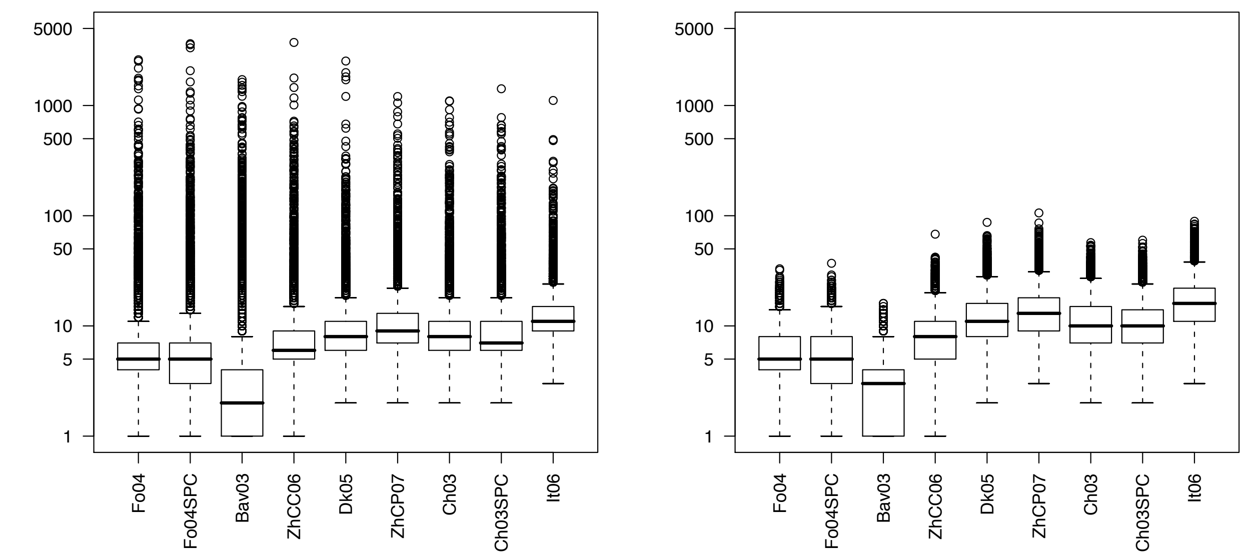


Figure 1: Number of iterations performed by alternating scaling using variants ASEXTR (left) and ASRAND (right). A boxplot is based on the order statistics. The box is constructed around the median, its borders are the lower and upper hinges (similar to the quartiles). Around the box there is a range of 1.5 times the boxlength. Outside this area an observation is an outlier, i.e. an extreme observation.

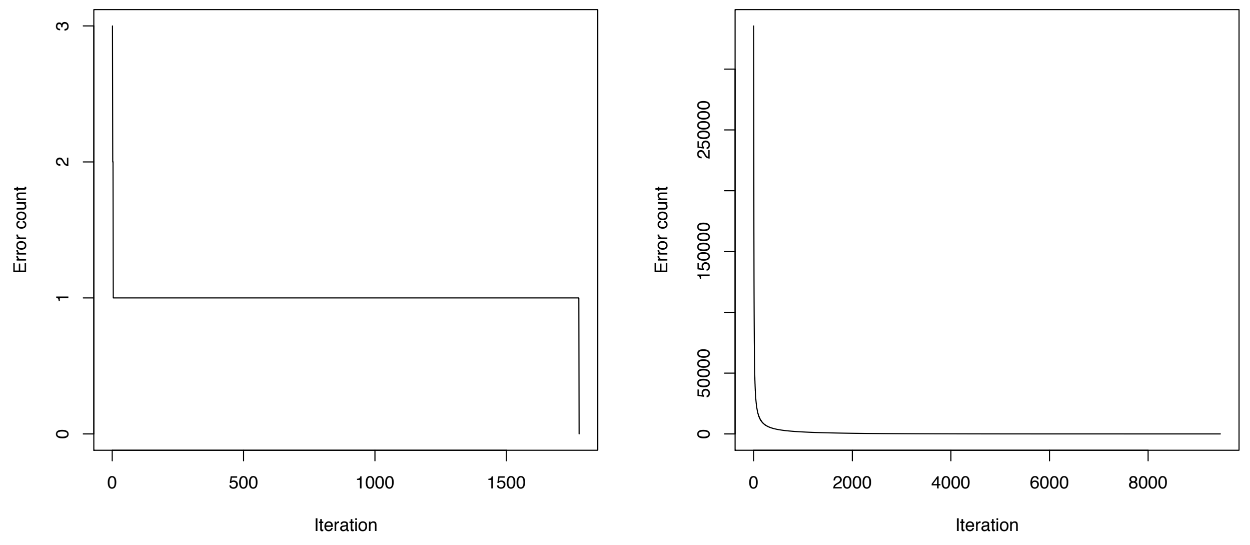
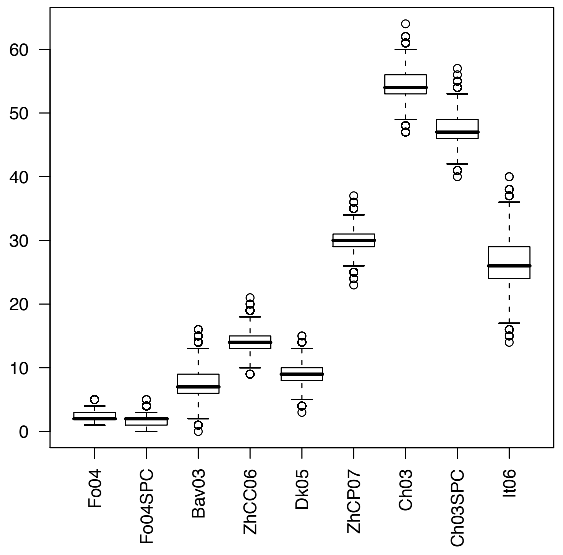


Figure 2: Decrease in error as a function of number of iterations for an instance of **ZhCC06** (left) and **MOD3M** (right). The algorithms ASEXTR and ASMDPT were used, respectively.



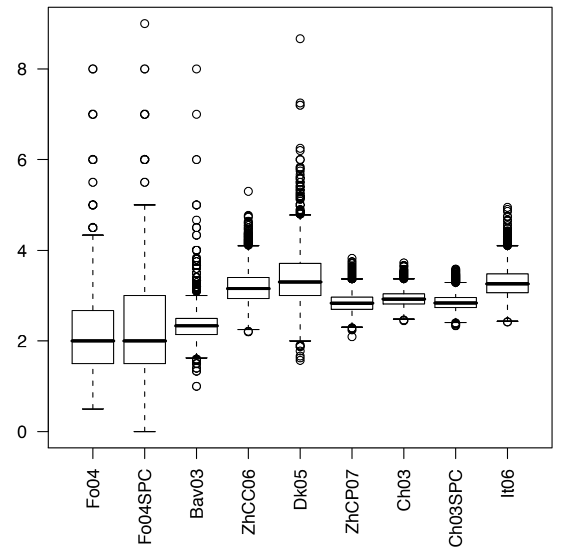


Figure 3: Number of iterations performed by tie-and-transfer algorithm (left), and tie update to transfers ratio (right).

count.) For **Ch03** the relatively high number of iterations (or initial error count) appears to be due to a large variation in district sizes and many small districts (6 of size 1 and 20 of size 10 or less); furthermore, **Ch03** has many small parties.

In Figure 3 (right) we present the tie update to transfer ratios. Recall that the number of tie updates per transfer is bounded by $n + m$. In practice the ratio is much smaller: It is below 4 on average for all problem instances. For **ZhCP07**, **Ch03** and **Ch03SPC** the tie update to transfer ratio has the smallest variation with an average below 3. Based on this data we may conclude that the worst-case number of tie updates never occurs in practice.

Hybrid algorithms

In Figure 4 the results for two of the hybrid variants are presented, namely IPF–TT and ASEXTR–TT. If we compare the iteration counts with those for pure alternating scaling (Figure 1) and tie-and-transfer (Figure 3), we note that the average number of iterations is actually smaller for the hybrid algorithms — in particular for the more difficult instances. So the hybrid algorithms are indeed very competitive and recommendable in practice for real-life problems.

For the IPF–TT algorithm, the tie update to transfer ratio for TT is lower than for pure TT, while the opposite is the case for ASEXTR–TT. So the input multipliers from the front-end algorithm do have an effect on the behavior of the TT algorithm.

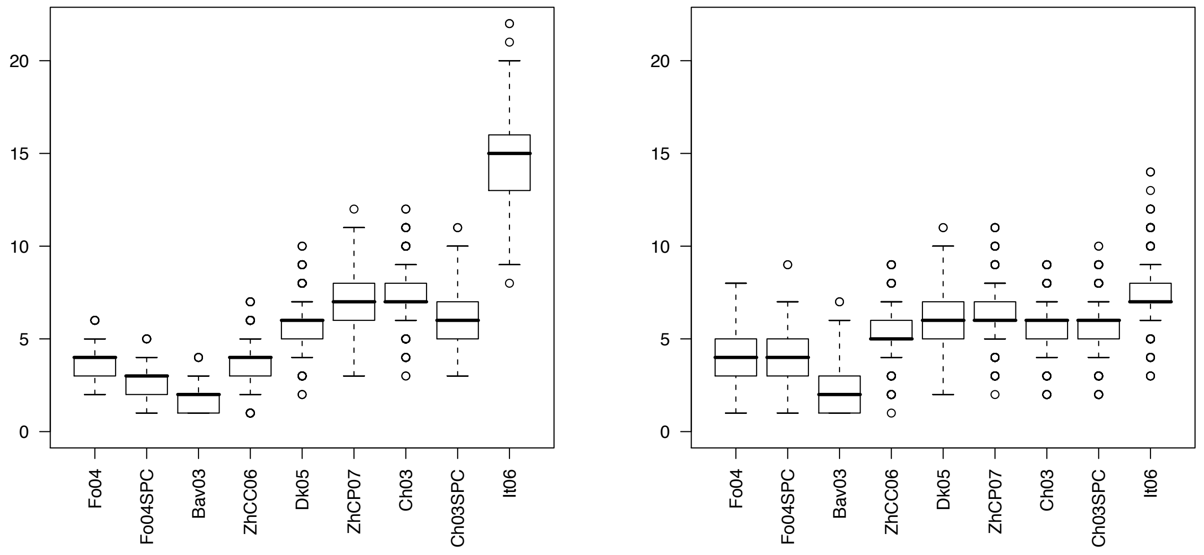


Figure 4: Number of iterations performed by hybrid algorithms. Iterative proportional fitting (IPF) followed by tie-and-transfer (left). Alternating scaling (extreme multiplier) followed by tie-and transfer (right).

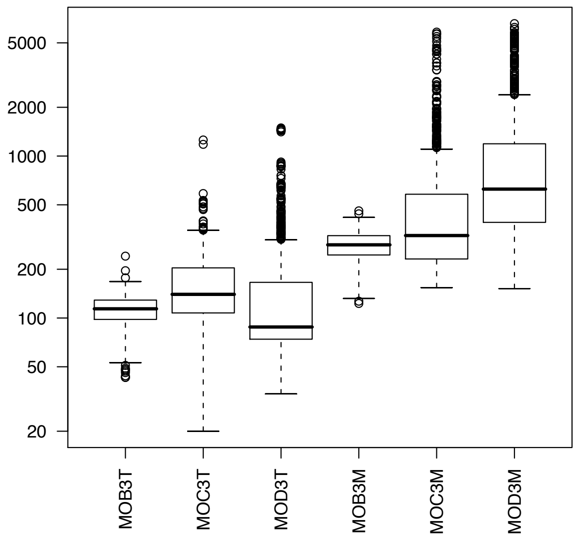
Marathon instances

We conclude the study of the algorithmic performance by considering the results for the Marathon instances (Figure 5). The Marathon instances are constructed to require many iterations by IPF and alternating scaling, and our experiments did confirm that. ASEXTR used significantly more iterations on the 3×3 Marathon instances than on real-life instances of comparable size (Figure 5, left). The random process generated only a small number of instances having a higher number of iterations than the original problem instances.

The results for the TT algorithm are presented for the variant where the running time has a logarithmic dependence on the house size, and we see that the number of iterations only increases by a factor of 2.5 when going from a house size of 10^3 to a house size of 10^6 (Figure 5, right). If we had used the original TT algorithm, this dependence would have been linear — and indeed extremely time-consuming for the Marathon instances.

5.4 Quality of apportionments

Our final results section is on the quality of divisor-based biproportional apportionments. Recall that the apportionment itself is independent on the algorithm used to produce it [5, 7]. Here we present results for the quality measures defined in Section 4.



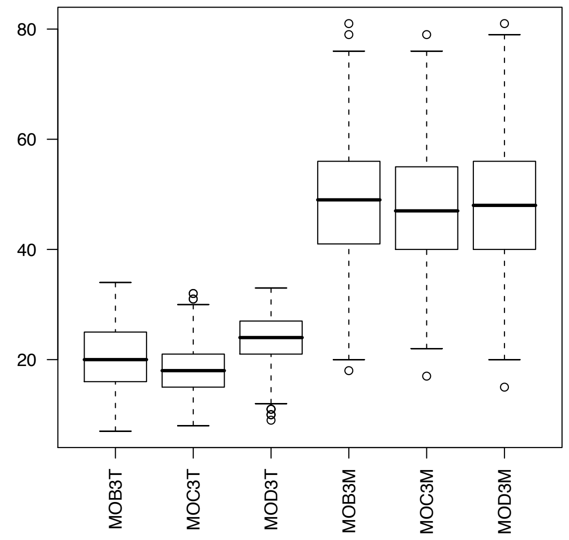


Figure 5: Number of iterations performed by ASEXTR (left), and TT (right) on Marathon instances.

Gallagher index

The distribution of the Gallagher indices — based on district quota and fair share quota — for the real-life problem instances are presented in Figure 6. The difference between the Gallagher index based on district quota and fair share quota, respectively, is hardly visible. Both Gallagher indices are on average below 2 for all problem instances, except for **Fo04** and **Fo04SPC**. In the electoral systems literature, a Gallagher index below 2 is considered to be excellent [21].

The reason why **Fo04** and **Fo04SPC** have a significantly greater Gallagher index is the small house size (32 seats), which gives a more rough discretisation than in the other cases. **Bav03** and **It06** are on the opposite end of the scale having a Gallagher index below 1; for **Bav03** this is due to the small number of parties and districts (which results in more seats per party per district), and for **It06** this appears to be due to the large house size.

Quota breaches

Quota breaches are not particularly frequent for the problem instances considered, and this is independent on whether quota breaches are measured relative to district quota or fair share quota (Figure 7). The only real exception is **It06**, where the relativly high number of quota breaches can be explained by big differences in party rankings within districts; the slightly higher number of quota breaches for **Ch03** and **Ch03SPC** can also be explained by significant party ranking differences.

The SPC rule increases the number of quota breaches slightly. While additional constraints are usually introduced for some good reason, they do interfere with the idea of proportionality, and our

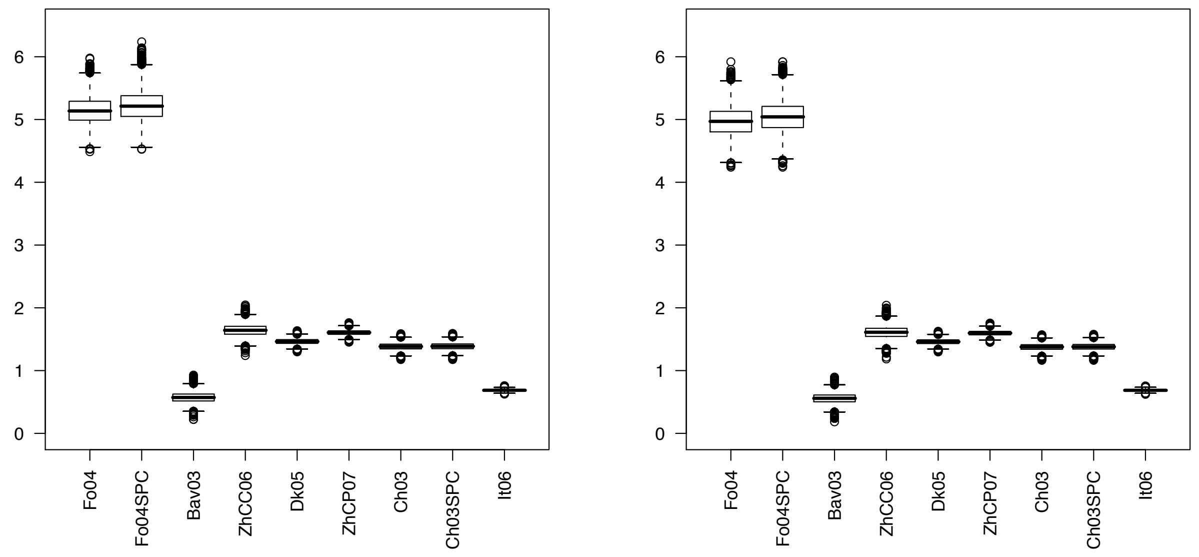


Figure 6: Gallagher index based on district quota (left) and fair share quota (right).

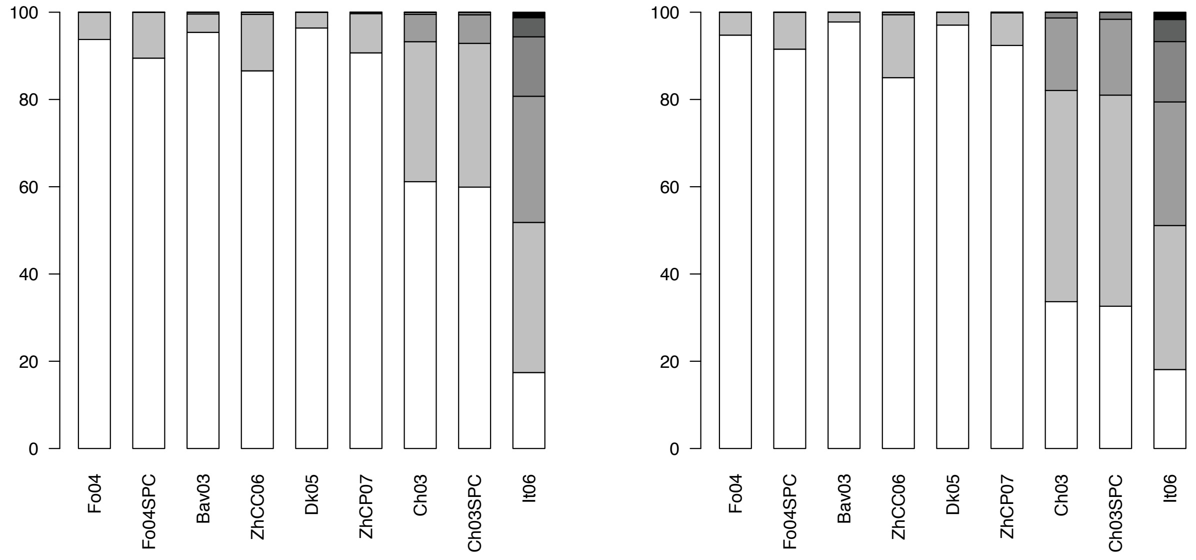


Figure 7: Bar-plot showing the fraction of instances with various levels of quota breaches based on district quota (left) and fair share quota (right). White bars indicate no quota breaches, and increasingly shaded bars indicate 1, 2, 3, 4 and more than 4 quota breaches (black shading).

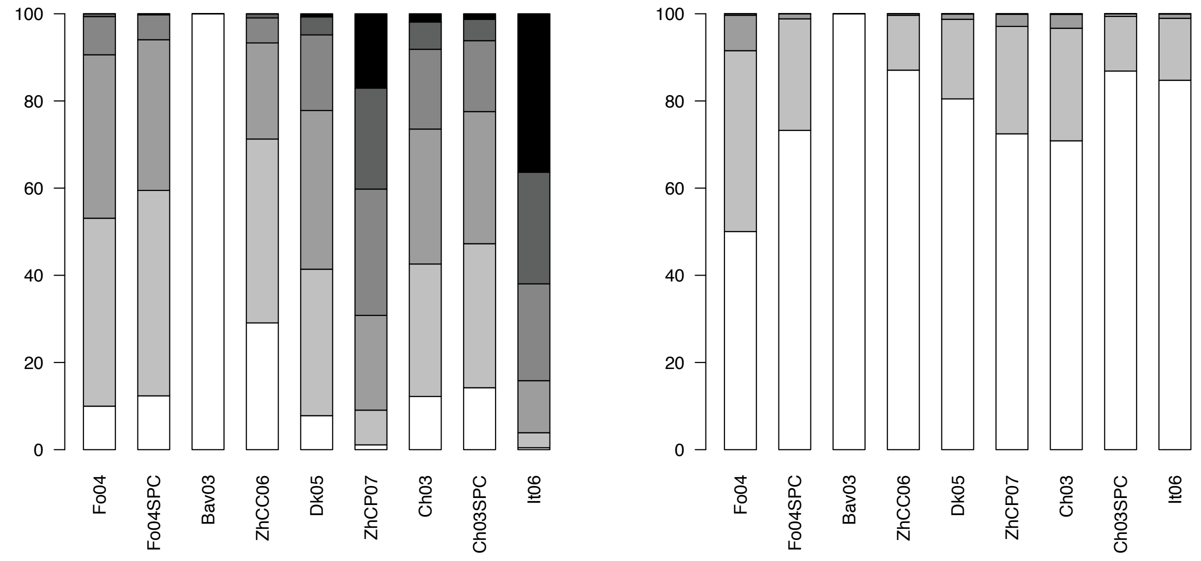


Figure 8: Bar-plot showing fraction of instances with various levels of discordant seat allocations: Number of districts with discordant seat allocations (left) and number of districts that have a party that receives more seats than the strongest party in the district (right). White bars indicate no discordant seat allocations, and increasingly shaded bars indicate 1, 2, 3, 4 and more than 4 districts with discordant seat allocations (black shading).

results confirm that.

For **Ch03** and **Ch03SPC** the number of quota breaches is somewhat higher for the fair share quota than for the district quota. This appears to be due to the higher number of zeros in the vote matrix: Parties running in only one district are a little overrepresented in the fair share quota, and this appears to create more quota breaches.

Discordant seat allocations

In Figure 8 (left), the distribution of discordant seat allocations is shown. We see that discordant seat allocations are almost inevitable, except for the very homogeneous **Bav03** problem instance. For most problem instances there are typically one or more districts with discordant seat allocations. Problem instances **ZhCP07** and **It06** have particularly many discordant seat allocations. This is partly due to a relativly large number of districts and parties — combined with major differences in party rankings within districts.

If we look at the number of districts that have a party that receives more seats than the strongest party in the district (Figure 8, right), we see that this does not happen very often. Here **Fo04** shows the worst behavior, which can be explained by the the small house size, and thus very small seat numbers within districts.

The SPC rule significantly decreases the number of discordant seat allocations. So for this particular measure of (local) proportionality, adding a constraint to the biproportional apportionment problem has a positive influence.

In many cases discordant seat allocations can be explained by the relative strength of the district vote counts within the same party. On other words, if a party i is part of a discordant seat allocation within district j, then in most cases the reason is the quest for proportionality *within* party i across the districts.

6 Conclusions

Our real-life benchmark study of divisor-based biproportional apportionment clearly shows that for real-life instances all considered algorithms deliver the apportionment almost instantly (i.e., within a fraction of a second). Furthermore, the quality of the obtained apportionments is excellent both wrt. the Gallagher index and quota breaches. One seemingly unavoidable weakness is the occurrence of discordant seat allocations (or seat reversals) within districts — but this is the price one has to pay to achieve proportionality over the whole electoral region. The SPC rule corrects the most counter-intuitive effects of the pure biproportional rule, and often it also reduces the number of discordant seat allocations in settings with small districts.

Among the alternating scaling algorithms, ASMDPT is particularly recommendable, although ASEXTR on average is faster (but may result in extreme iteration counts occasionally). If there is no need to have a completely reproducible result, ASRAND is a good and fast alternative.

The tie-and-transfer algorithm has the advantage that there exists a mathematical proof of its convergence, but the algorithm is difficult to explain and must be implemented using integer arithmetic in order to avoid numerical problems completely. For large house sizes, a hybrid algorithm or the fast tie-and-transfer algorithm would be the best choice.

If biproportional apportionment is used in an electoral system, it would be reasonable to explain the electoral system on basis of alternating scaling because of its simplicity — but not forgetting to mention that there exist other bulletproof ways to find the unique solution. Our results show that the results are worth the price of complexity one has to pay for this electoral system.

References

[1] J. M. Anthonisse. Proportional Representation in a Regional Council. *Centrum voor Wiskunde en Informatica (CWI) Newsletter*, December:22–29, 1984.

[2] M. Balinski. Fair Majority Voting (or How to Eliminate Gerrymandering). *The American Mathematical Montly*, February, 2008.

[3] M. Balinski and F. Pukelsheim. Matrices and Politics. In *Festschrift for Tarmo Pukkila on His 60th Birthday*, pages 233–242, Department of Mathematics, Statistics, and Philosophy, University of Tampere, 2006.

[4] M. L. Balinski. Wahlen in Mexico - Verhältniswahlrecht häppchenweise. *Spektrum der Wissenschaft*, Oktober:72–74, 2002.

[5] M. L. Balinski and G. Demange. Algorithms for Proportional Matrices in Reals and Integers. *Mathematical Programming*, 45:193–210, 1989.

[6] M. L. Balinski and G. Demange. An Axiomatic Approach to Proportionality Between Matrices. *Mathematics of Operations Research*, 14:700–719, 1989.

[7] M. L. Balinski and S. T. Rachev. Rounding Proportions: Methods of Rounding. *Mathematical Scientist*, 22:1–26, 1997.

[8] M. L. Balinski and V. Ramírez. Mexico's 1997 Apportionment Defies its Electoral Law. *Electoral Studies*, 18:117–124, 1999.

[9] M. L. Balinski and H. P. Young. *Fair Representation — Meeting the Ideal of One Man, One Vote*. Brookings Institution Press, Washington, D.C., second edition, 2001.

[10] K. Benoit. Which Electoral Formula is the Most Proportional? A New Look with New Evidence. *Political Analysis*, 8:381–388, 2000.

[11] D. Bochsler. Biproportionale Wahlverfahren für den Schweizer Nationalrat. `www.opus-bayern.de/uni-augsburg/volltexte/2005/160`, 2005.

[12] N. B. Christensen. Evaluating Voting Procedures using Different Criteria and Computer Simulations. In *European Public Choice Society Annual Meeting*, 2003.

[13] L. H. Cox and L. R. Ernst. Controlled Rounding. *INFOR*, 20:423–432, 1982.

[14] G. Dorfleitner and T. Klein. Rounding with Multiplier Methods: An Efficient Algorithm and Applications in Statistics. *Statistical Papers*, 40:143–158, 1999.

[15] J. Elklit. Denmark: Simplicity Embedded in Complexity (or is it the Other Way Round?). In M. Gallagher and P. Mitchell, editors, *The Politics of Electoral Systems*, pages 453–471. Oxford University Press, 2005.

[16] L. R. Ernst. Apportionment Methods for the House of Representatives and the Court Challenges. *Management Science*, 40:1207–27, 1994.

[17] V. Fragnelli, G. Monella, and G. Ortona. A Simulative Approach for Evaluating Electoral Systems. *Homo Oeconomicus*, 22:525–549, 2005.

[18] V. Fragnelli and G Ortona. Comparison of Electoral Systems: Simulative and Game Theoretic Approaches. In *Mathematics and Democracy. Recent Advances in Voting Systems and Collective Choice*, pages 65–81, New York, 2006. Springer.

[19] G. N. Frederickson and D. B. Johnson. Complexity of Selection and Ranking in X+Y and Matrices with Sorted Columns. *Journal of Computer and System Sciences*, 24:197–208, 1982.

[20] N. Gaffke and F. Pukelsheim. Divisor Methods for Proportional Representation Systems: An Optimization Approach to Vector and Matrix Problems. `http://www.opus-bayern.de/uni-augsburg/volltexte/2007/625/`, 2006.

[21] M. Gallagher. Proportionality, Disproportionality and Electoral Systems. *Electoral Studies*, 10:33–51, 1991.

[22] M. Gassner. Two-Dimensional Rounding for a Quasi-Proportional Representation. *European Journal of Political Economy*, 4:529–538, 1988.

[23] M. Gassner. Biproportional Delegations: A Solution for Two-Dimensional Proportional Representation. *Journal of Theoretical Politics*, 3:321–342, 1991.

[24] M. Happacher. The Discrepancy Distribution of Stationary Multiplier Rules for Rounding Probabilities. *Metrika*, 53:171–181, 2001.

[25] M. Happacher and F. Pukelsheim. Rounding Probabilities: Maximum Probability and Minimum Complexity Multipliers. *Journal of Statistical Planning and Inference*, 85:145–158, 2000.

[26] P. James. The Free State of Bavaria: A special case. *Representation*, 33:17–20, 1995.

[27] N. Linial, A. Samorodnitsky, and A. Wigderson. A Deterministic Strongly Polynomial Algorithm for Matrix Scaling and Approximate Permanents. *Combinatorica*, 20:545–568, 2000.

[28] S. Maier. Algorithms for Biproportional Apportionment Methods. In *Mathematics and Democracy. Recent Advances in Voting Systems and Collective Choice*, pages 105–116, New York, 2006.

[29] S. Maier and F. Pukelsheim. Bazi: A Free Computer Program for Proportional Representation Apportionment. Manuscript, 2007.

[30] A. W. Marshall and I. Olkin. Scaling of Matrices to Achieve Specified Row and Column Sums. *Numerische Mathematik*, 12:83–90, 1968.

[31] G. De Meur, M. Gassner, and X. Hubaut. A Mathematical Model for Political Bipolarization. *European Journal of Political Research*, 13:409–420, 1985.

[32] A. Pennisi. The Italian Bug: A Flawed Procedure for Bi-Proportional Seat Allocation. In *Mathematics and Democracy. Recent Advances in Voting Systems and Collective Choice*, pages 151–164, New York, 2006. Springer.

[33] A. Pennisi, F. Ricca, and B. Simeone. Malfunzionamenti dell'allocazione biproporzionale di seggi nella riforma elettorale italiana. Technical Report 21, Dimartimento de Statistica, Propabilità e Statistiche Applicate, Università degli studi di Roma, 2005.

[34] F. Pukelsheim. BAZI – A Java Program for Proportional Representation. In *Oberwolfach Reports*, volume 1, pages 735–737, 2004.

[35] F. Pukelsheim. Current Issues of Apportionment Methods. In *Mathematics and Democracy. Recent Advances in Voting Systems and Collective Choice*, pages 168–176, New York, 2006.

[36] F. Pukelsheim and C. Schuhmacher. Das neue Zürcher Zuteilungsverfahren für Parlamentswahlen. *Aktuelle Juristische Praxis - Pratique Juridique Actuelle*, 5:505–522, 2004.

[37] G. Rote and A. Vogel. A Heuristic for Decomposing Traffic Matrices in TDMA Satellite Communication. *ZOR - Methods and Models of Operations Research*, 38:281–307, 1993.

[38] G. Rote and M. Zachariasen. Matrix Scaling by Network Flow. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 848–854, 2007.

[39] R. Sinkhorn. A Relationship between Arbitrary Positive Matrices and Doubly Stochastic Matrices. *Ann. Math. Statist.*, 35:876–879, 1964.

[40] R Development Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0, 2006.

[41] M. Zachariasen. Algorithmic Aspects of Divisor-Based Biproportional Rounding. Technical Report 06-05, DIKU, Department of Computer Science, University of Copenhagen, 2006.

[42] P. Zachariasen and M. Zachariasen. A Comparison of Electoral Formulae for the Faroese Parliament (The Løgting). In *Mathematics and Democracy. Recent Advances in Voting Systems and Collective Choice*, pages 235–251, New York, 2006. Springer.

Appendix

Table 5: Descriptive Statistics Faroese Parliament 2004 (**Fo04**)

	Min	Mean	Median	Maximum
Quota Breaches				
District	0	0.063	0	2
Fairshare	0	0.053	0	2
Galagher Index				
District	4.487	5.143	5.136	5.978
Fairshare	4.241	4.969	4.970	5.920
Discordant seat allocations				
Number of districts	0	1.470	1	4
Number of districts strongest party	0	0.588	0	3
Iterations				
ASMDPT	1	6.751	5	79
ASEXTR	1	10.576	5	2601
ASRAND	1	6.338	5	33
TT	1	2.448	2	5
ASMDPT+ TT	1	4.099	4	7
ASEXTR+ TT	1	4.161	4	8
ASRAND+ TT	1	4.219	4	7
IPF+ TT	2	3.594	4	6
Tie update to transfer ratio				
TT	0.500	2.196	2.000	8.000

Table 6: Descriptive Statistics Faroese Parliament 2004 (SPC) (**Fo04SPC**)

	Min	Mean	Median	Maximum
Quota Breaches				
District	0	0.106	0	2
Fairshare	0	0.085	0	2
Galagher Index				
District	4.526	5.219	5.213	6.238
Fairshare	4.241	5.042	5.042	5.920
Discordant seat allocations				
Number of districts	0	1.344	1	4
Number of districts strongest party	0	0.279	0	2
Iterations				
ASMDPT	1	6.348	5	63
ASEXTR	0	10.902	5	3641
ASRAND	1	5.997	5	37
TT	0	1.758	2	5
ASMDPT+ TT	1	4.004	4	7
ASEXTR+ TT	1	4.133	4	9
ASRAND+ TT	1	4.123	4	7
IPF+ TT	1	2.647	3	5
Tie update to transfer ratio				
TT	0.000	2.231	2.000	9.000

Table 7: Descriptive Statistics Bavarian State Parliament 2003 (**Bav03**)

	Min	Mean	Median	Maximum
Quota Breaches				
District	0	0.050	0	2
Fairshare	0	0.023	0	2
Galagher Index				
District	0.221	0.572	0.571	0.928
Fairshare	0.184	0.556	0.558	0.896
Discordant seat allocations				
Number of districts	0	0.000	0	0
Number of districts strongest party	0	0.000	0	0
Iterations				
ASMDPT	1	2.910	2	21
ASEXTR	1	8.873	2	1718
ASRAND	1	3.194	3	16
TT	0	7.330	7	16
ASMDPT+ TT	1	2.338	2	7
ASEXTR+ TT	1	2.358	2	7
ASRAND+ TT	1	2.431	3	7
IPF+ TT	1	1.794	2	4
Tie update to transfer ratio				
TT	1.000	2.372	2.333	8.000

Table 8: Descriptive Statistics Zurich City Council 2006 (**ZhCC06**)

	Min	Mean	Median	Maximum
Quota Breaches				
District	0	0.140	0	3
Fairshare	0	0.156	0	3
Galagher Index				
District	1.240	1.644	1.642	2.045
Fairshare	1.186	1.609	1.610	2.040
Discordant seat allocations				
Number of districts	0	1.074	1	5
Number of districts strongest party	0	0.133	0	2
Iterations				
ASMDPT	1	11.728	8	168
ASEXTR	1	11.283	6	3734
ASRAND	1	8.954	8	68
TT	9	14.353	14	21
ASMDPT+ TT	1	5.109	5	9
ASEXTR+ TT	1	5.103	5	9
ASRAND+ TT	1	5.194	5	10
IPF+ TT	1	3.657	4	7
Tie update to transfer ratio				
TT	2.200	3.185	3.154	5.300

Table 9: Descriptive Statistics Danish Parliament 2005 (**Dk05**)

	Min	Mean	Median	Maximum
Quota Breaches				
District	0	0.037	0	2
Fairshare	0	0.030	0	2
Galagher Index				
District	1.299	1.464	1.463	1.635
Fairshare	1.298	1.460	1.460	1.625
Discordant seat allocations				
Number of districts	0	1.786	2	6
Number of districts strongest party	0	0.209	0	4
Iterations				
ASMDPT	2	21.522	14	508
ASEXTR	2	12.165	8	2532
ASRAND	2	12.883	11	87
TT	3	8.681	9	15
ASMDPT+ TT	2	6.007	6	11
ASEXTR+ TT	2	5.806	6	11
ASRAND+ TT	2	6.067	6	11
IPF+ TT	2	5.738	6	10
Tie update to transfer ratio				
TT	1.571	3.369	3.300	8.667

Table 10: Descriptive Statistics Zurich Canton Parliament 2007 (**ZhCP07**)

	Min	Mean	Median	Maximum
Quota Breaches				
District	0	0.096	0	3
Fairshare	0	0.078	0	2
Galagher Index				
District	1.453	1.606	1.606	1.763
Fairshare	1.452	1.597	1.597	1.757
Discordant seat allocations				
Number of districts	0	3.230	3	9
Number of districts strongest party	0	0.306	0	3
Iterations				
ASMDPT	2	27.059	16	1062
ASEXTR	2	12.813	9	1207
ASRAND	3	14.668	13	106
TT	23	30.054	30	37
ASMDPT+ TT	2	6.677	7	11
ASEXTR+ TT	2	6.450	6	11
ASRAND+ TT	2	6.873	7	11
IPF+ TT	3	6.870	7	12
Tie update to transfer ratio				
TT	2.094	2.843	2.833	3.821

Table 11: Descriptive Statistics Swiss National Assembly 2003 (**Ch03**)

	Min	Mean	Median	Maximum
Quota Breaches				
District	0	0.461	0	4
Fairshare	0	0.856	1	4
Galagher Index				
District	1.176	1.384	1.384	1.587
Fairshare	1.166	1.376	1.377	1.571
Discordant seat allocations				
Number of districts	0	1.821	2	7
Number of districts strongest party	0	0.326	0	3
Iterations				
ASMDPT	2	16.816	11	319
ASEXTR	2	11.298	8	1102
ASRAND	2	11.721	10	57
TT	47	54.341	54	64
ASMDPT+ TT	2	5.781	6	11
ASEXTR+ TT	2	5.627	6	9
ASRAND+ TT	2	5.916	6	11
IPF+ TT	3	7.409	7	12
Tie update to transfer ratio				
TT	2.446	2.930	2.923	3.722

Table 12: Descriptive Statistics Swiss National Assembly 2003 (SPC) (**Ch03SPC**)

	Min	Mean	Median	Maximum
Quota Breaches				
District	0	0.478	0	4
Fairshare	0	0.880	1	4
Galagher Index				
District	1.176	1.387	1.387	1.591
Fairshare	1.166	1.379	1.380	1.578
Discordant seat allocations				
Number of districts	0	1.687	2	7
Number of districts strongest party	0	0.138	0	3
Iterations				
ASMDPT	2	16.056	11	337
ASEXTR	2	10.960	7	1419
ASRAND	2	11.344	10	60
TT	40	47.513	47	57
ASMDPT+ TT	2	5.736	6	11
ASEXTR+ TT	2	5.591	6	10
ASRAND+ TT	2	5.876	6	11
IPF+ TT	3	6.263	6	11
Tie update to transfer ratio				
TT	2.333	2.849	2.837	3.587

Table 13: Descriptive Statistics Italian National Assembly 2006 (**It06**)

	Min	Mean	Median	Maximum
Quota Breaches				
District	0	1.571	1	7
Fairshare	0	1.601	1	7
Galagher Index				
District	0.626	0.687	0.686	0.759
Fairshare	0.623	0.686	0.686	0.755
Discordant seat allocations				
Number of districts	0	4.012	4	10
Number of districts strongest party	0	0.164	0	3
Iterations				
ASMDPT	3	38.886	22	982
ASEXTR	3	13.979	11	1111
ASRAND	3	17.900	16	89
TT	14	26.424	26	40
ASMDPT+ TT	3	7.765	8	14
ASEXTR+ TT	3	7.496	7	14
ASRAND+ TT	3	7.957	8	14
IPF+ TT	8	14.527	15	22
Tie update to transfer ratio				
TT	2.417	3.282	3.258	4.947

Table 14: Descriptive Statistics **MOB3T**

	Min	Mean	Median	Maximum
Iterations				
ASmdpt	55	138.908	141	219
ASextr	43	111.748	114	241
ASrand	56	137.462	139	212
TT	7	20.464	20	34
ASmdpt+ TT	30	72.001	74	112
ASextr+ TT	31	70.824	73	100
ASrand+ TT	30	72.016	74	118
IPF+ TT	38	93.244	97	130
Tie update to transfer ratio				
TT	1.857	2.487	2.421	3.650

Table 15: Descriptive Statistics **MOC3T**

	Min	Mean	Median	Maximum
Iterations				
ASmdpt	25	227.898	172	2445
ASextr	20	166.275	140	1257
ASrand	25	222.565	170	2322
TT	8	18.097	18	32
ASmdpt+ TT	20	64.751	69	136
ASextr+ TT	19	63.860	69	136
ASrand+ TT	20	64.781	69	137
IPF+ TT	21	120.363	116	231
Tie update to transfer ratio				
TT	1.727	2.317	2.250	3.600

Table 16: Descriptive Statistics **MOD3T**

	Min	Mean	Median	Maximum
Iterations				
ASMDPT	41	249.246	121	2686
ASEXTR	34	164.461	88	1492
ASRAND	41	242.851	119	2583
TT	9	23.601	24	33
ASMDPT+ TT	26	46.917	46	74
ASEXTR+ TT	25	46.204	45	72
ASRAND+ TT	26	46.841	46	74
IPF+ TT	26	86.309	71	303
Tie update to transfer ratio				
TT	1.889	2.689	2.750	3.462

Table 17: Descriptive Statistics **MOB3M**

	Min	Mean	Median	Maximum
Iterations				
ASMDPT	136	308.987	310	488
ASEXTR	123	281.821	283	459
ASRAND	134	306.846	308	473
TT	18	48.845	49	81
ASMDPT+ TT	113	240.762	243	348
ASEXTR+ TT	112	239.280	242	346
ASRAND+ TT	113	240.682	243	348
IPF+ TT	148	288.908	291	421
Tie update to transfer ratio				
TT	1.891	2.505	2.456	3.525

Table 18: Descriptive Statistics **MOC3M**

	Min	Mean	Median	Maximum
Iterations				
ASMDPT	166	640.199	349	6885
ASEXTR	154	577.130	323	5825
ASRAND	164	636.913	348	6717
TT	17	46.948	47	79
ASMDPT+ TT	89	424.759	292	6122
ASEXTR+ TT	89	426.105	290	6120
ASRAND+ TT	89	426.109	293	6121
IPF+ TT	165	550.924	331	4957
Tie update to transfer ratio				
TT	1.875	2.436	2.394	3.533

Table 19: Descriptive Statistics **MOD3M**

	Min	Mean	Median	Maximum
Iterations				
ASmdpt	162	1141.594	680	7817
ASextr	152	1032.584	625	6575
ASrand	164	1133.989	677	7639
TT	15	47.598	48	81
ASmdpt+ TT	94	851.376	499	16797
ASextr+ TT	91	826.274	498	16797
ASrand+ TT	95	822.530	497	16797
IPF+ TT	142	958.450	606	5401
Tie update to transfer ratio				
TT	1.972	2.602	2.545	3.679