Appendix

A Missing Proofs of Section 4

Lemma 2 (★). *Let (G, c) be a delegation graph and let $(\mathcal{F}, E_y, y)$ be the output of Algorithm 1. Then:*

(i) For every (G, c), the output of the algorithm is unique, i.e., it does not depend on the choice of the strongly connected component in line 3.

(ii) $\mathcal{F}$ is laminar, i.e., for any $X, Y \in \mathcal{F}$ it holds that either $X \subseteq Y$, $Y \subseteq X$, or $X \cap Y = \emptyset$.

(iii) Branching B in (G, c) is min-cost iff (a) $B \subseteq E_y$, and (b) $|B \cap \delta^+(X)| = 1$ for all $X \in \mathcal{F}, X \subseteq N$.

(iv) For every $X \in \mathcal{F}$, an in-tree T in $G[X] = (X, E[X])$, where $E[X] = \{(u, v) \in E \mid u, v \in X\}$, is min-cost iff (a) $T \subseteq E_y$, and (b) $|T \cap \delta^+(Y)| = 1$ for all $Y \in \mathcal{F}$ such that $Y \subset X$.

Proof. Let (G, c) be a delegation graph and let $(\mathcal{F}, E_y, y)$ be the output of Algorithm 1.

We start by proving statement (ii). The sets in $\mathcal{F}$ correspond exactly to those sets with positive y-value. Assume for contradiction that there exist two sets $X, Y \in \mathcal{F}$ with $X \cap Y \neq \emptyset$ and none of the subsets is a subset of the other. Assume without loss of generality that X was selected before Y by the algorithm and let y_1 and y_2 be the status of the function y in each of the two situation. Then, by construction of the algorithm it holds that $G_1 = (N \cup S, E_{y_1})$ is a subgraph of $G_2 = (N \cup S, E_{y_2})$. This is because once an edge is added to the set of tight edges (denoted by E_y) it remains in this set. Since Y is a strongly connected component in G_2 without outgoing edge, it holds that for every $z \in X \setminus Y$ and $z' \in X \cap Y$, the node z' does not reach z in G_2. However, this is a contradiction to the fact that X is a strongly connected component in the graph G_1, which concludes the proof of statement (ii).

We now turn to prove statement (i) and already assume that $\mathcal{F}$ is laminar. We fix an order of the selected strongly connected components in line 3 of the algorithm. Then, suppose that for some other choices in line 3, the algorithm returns some other output $(\hat{\mathcal{F}}, E_{\hat{y}}, \hat{y})$. Note that $\hat{\mathcal{F}} \neq \mathcal{F}$ or $E_{\hat{y}} \neq E_y$ implies that $\hat{y} \neq y$. Thus, it suffices to assume for contradiction that $\hat{y} \neq y$. Then there must be a smallest set X, that has $y(X) \neq \hat{y}(X)$ (without loss of generality we assume $y(X) > \hat{y}(X)$). Let $\mathcal{X} = 2^X \setminus \{X\}$ be the set of all strict subsets of X. Since we defined X to be of minimal cardinality, we have $y[\mathcal{X}] = \hat{y}[\mathcal{X}]$, where $y[\mathcal{X}]$, and $\hat{y}[\mathcal{X}]$ denote the restriction of y and $\hat{y}$ to $\mathcal{X}$, respectively. Because $y(X) > 0$, all children of X are strongly connected by tight edges with respect to $y[\mathcal{X}]$ and have no tight edges pointing outside of X. Now, consider the iteration of the alternative run of the Algorithm 2, in which the algorithm added the last set in $\mathcal{X} \cup \{X\}$. Since $\hat{y}(X) < y(X)$, for every further iteration of the algorithm, a chosen set $X' \neq X$ cannot contain any node in X (because otherwise X' cannot form a strongly connected component without outgoing edge). However, since the nodes in X cannot reach a sink via tight edges, this is a contradiction to the termination of the algorithm.

We now prove statement (iii). The plan of attack is the following: First we define a linear program that captures the min-cost branchings in a delegation graph. Second, we dualize the linear program and show that y (more precisely a minor variant of y) is an optimal solution to the dual LP, and third, utilize complementary slackness to prove the claim. For a given delegation graph (G, c) with $V(G) = N \cup S$ we define the following linear program, also denoted by (LP):

$$
\begin{aligned}
\min \sum_{e \in E} c(e) x_e & \\
\sum_{e \in \delta^+(X)} x_e \geq 1 & \qquad \forall\, X \subseteq N \\
x_e \geq 0 & \qquad \forall e \in E
\end{aligned}
$$

We claim that every branching B in G induces a feasible solution to (LP). More precisely, given a branching B, let

$$x_e = \begin{cases} 1 & \text{if } e \in B \\ 0 & \text{if } e \notin B. \end{cases}$$

The last constraint is trivially satisfied. Now, assume for contradiction that there exists $X \subseteq N$ such that the corresponding constraint in (LP) is violated. In this case the nodes in X have no path towards some sink node in B, a contradiction to the fact that B is a (maximum cardinality) branching. In particular, this implies that the objective value of (LP) is at most the minimum cost of any branching in G (in fact the two values are equal, but we do not need to prove this at this point). We continue by deriving the dual of (LP), to which we refer to as (DLP):

$$\begin{aligned} \max \sum_{X \subseteq N} y_X & \\ \sum_{X \subseteq N \mid e \in \delta^+(X)} y_X \leq c(e) & \qquad \forall\, e \in E \\ y_X \geq 0 & \qquad \forall X \subseteq N \end{aligned}$$

Now, let y be the function returned by Algorithm 1. We define $\hat{y}$, which is intuitively y restricted to all subsets on N, more precisely, $\hat{y}(X) = y(X)$ for all $X \subseteq N$. We claim that $\hat{y}$ is a feasible solution to (DLP). This can be easily shown by induction. More precisely, we fix any $e \in E$ and show that the corresponding constraint in (DLP) is satisfied throughout the execution of the algorithm. At the beginning of the algorithm y (and hence $\hat{y}$) is clearly feasible for (DLP). Now, consider any step in the algorithm and let X be the selected strongly connected component. If $e \in \delta^+(X)$, then we know that the constraint corresponding to e is not tight (since X has no tight edge in its outgoing cut). Moreover, y is only increased up to the point that some edge in $\delta^+(X)$ becomes tight (and not higher than that). Hence, after this round, the constraint for e is still satisfied. If, on the other hand, $e \notin \delta^+(X)$, then the left-hand-side of e's constraint remains equal when $y(X)$ is increased. Hence, the constraint of e is still satisfied.

Next, we claim that there exists a branching B in G, such that for the resulting primal solution x, it holds that $\sum_{e \in E} c(e) x_e = \sum_{X \subseteq N} \hat{y}(X)$. The branching B will be constructed in a top-down fashion by moving along the laminar hierarchy of $\mathcal{F}$. To this end let G_X be the contracted graph as defined in Algorithm 2. We start by setting $X = N \cup S$. Since every node in N can reach some sink via tight edges, we also know that every node in G_X can reach some sink. Hence, a branching in G_X has exactly one edge per node in V_X that is not a sink. Let's pick such a branching B_X. We know that for every edge in $B_X = (Y, Z)$ there exists some edge in the original graph G that is also tight, i.e., $u \in Y$ and $v \in Z$ such that $(u, v) \in E_y$. For every edge in B_X pick an arbitrary such edge and add it to B. Now, pick an arbitrary node $Y \in V(G_X)$. By construction, we know that exactly one edge from B is included in $\delta^+(Y)$, call this edge (u, v). Then, within the graph G_Y, there exists exactly one node $Z \in V(G_Y)$, that contains u. We are going to search for a Z-tree within G_Y. We know that such a tree exists since G_Y is strongly connected by construction. We follow the pattern from before, i.e., finding a Z-tree, mapping the edges back to the original graph (arbitrarily), and then continuing recursively. For proving our claim, it remains to show that $\sum_{e \in E} c(e) x_e = \sum_{X \subseteq N} \hat{y}(X)$. The crucial observation is that, by construction, every set in $\hat{\mathcal{F}} = \mathcal{F} \setminus \{\{s\} \mid s \in S\}$ is left by exactly one edge in B. Hence, we can partition the set $\hat{\mathcal{F}}$ into sets $\bigcup_{e \in B} \hat{F}_e$, where $\hat{\mathcal{F}}_e = \{X \in \hat{\mathcal{F}} \mid e \in \delta^+(X)\}$. Moreover, observe that every edge in B is tight. As a result we get that

$$\sum_{e \in B} c(e) x_e = \sum_{e \in B} \sum_{X \in \hat{\mathcal{F}}_e} \hat{y}(X) = \sum_{X \subseteq N} \hat{y}(X),$$

proving the claim.

As a result, note that we found a primal solution B (precisely, the x induced by B), and a dual solution $\hat{y}$ having the same objective value. By weak duality, we can conclude that both solutions are in particular optimal. It only remains to apply complementary slackness to conclude the claim. To this end, let B be a min-cost branching and x be the induced primal solution. By the argument above we know that x is optimal. Now, for any $X \subseteq N$ for which $\hat{y}(X) > 0$ (hence $X \in \mathcal{F}$), complementary slackness prescribes that the corresponding primal constraint is tight, i.e., $\sum_{e \in \delta^+(X)} x_e = 1$. Hence, the branching corresponding to x leaves the set X exactly once, and part (b) of statement (iii) is satisfied. For statement (a) we apply complementary slackness in the other direction. That is, when $x_e > 0$, this implies that the corresponding dual constraint is tight, implying that e has to be tight with respect to $\hat{y}$ and therefore also with respect to y (recall that y and $\hat{y}$ only differ with respect to the sink nodes).

We now turn to proving statement (iv). This is done almost analogously to statement (iii). Fix $X \in \mathcal{F}$. In the following we argue about the min-cost in-trees in $G[X]$ and how to characterize these via a linear program. To this end, we add a dummy sink node r to the graph $G[X]$ and call the resulting graph $\hat{G}$. More precisely, $\hat{G} = (X \cup \{r\}, E[X] \cup \{(u,r) \mid u \in X\})$. The cost of any edge $(u,r), u \in X$ is set to $c^* := \max_{e \in E(G)} c(e) + 1$, where it is only important that this value is larger than any other cost in the graph. We define the following LP:

$$
\begin{aligned}
\min \sum_{e \in E(\hat{G})} c(e) x_e & & \\
\sum_{e \in \delta^+_{\hat{G}}(Z)} x_e \geq 1 & \qquad & \forall\, Z \subseteq X \\
x_e \geq 0 & \qquad & \forall e \in E(\hat{G})
\end{aligned}
$$

For every min-cost in-tree T in $G[X]$ we obtain a feasible solution to (LP). To this end, let $u \in X$ be the sink node of T and define $\hat{T} = T \cup \{(u,r)\}$. Then, translate $\hat{T}$ to its incidence vector x. Given this observation, we again derive the dual of (LP), to which we refer to as (DLP):

$$
\begin{aligned}
\max \sum_{Z \subseteq X} y_Z & & \\
\sum_{Z \subseteq X \mid e \in \delta^+_{\hat{G}}(Z)} y_Z \leq c(e) & \qquad & \forall\, e \in E(\hat{G}) \\
y_Z \geq 0 & \qquad & \forall Z \subseteq X
\end{aligned}
$$

Now, let y be the output of Algorithm 1 for the original graph G. We derive $\hat{y} : 2^X \to \mathbb{R}$ as follows:

$$
\hat{y}(Z) = \begin{cases} y(Z) & \text{if } Z \subset X \\ c^* - \max_{u \in X} \sum_{Z \subset X \mid e \in \delta^+_{\hat{G}}(Z)} y(Z) & \text{if } Z = X \end{cases}
$$

First, analogously to (iii), it can be verified that $\hat{y}$ is a feasible solution to (DLP). Moreover, again analogously to (iii), there exists some min-cost r-tree in $\hat{G}$ and a corresponding primal solution x, such that $\sum_{e \in E(\hat{G})} c(e) x_e = \sum_{Z \subseteq X} \hat{y}_X$. (This tree is derived by first chosing a tight edge towards the dummy root node r and then again recurse over the laminar family $\mathcal{F}$ restricted to X.) This implies by weak duality that $\hat{y}$ is an optimal solution to (DLP) and any min-cost r-tree in $\hat{G}$ is an optimal solution to (LP). As a result, we can again apply complementary slackness in both directions: Let T be a min-cost in-tree in $G[X]$ with sink node $u \in X$. Then let $\hat{T} = T \cup \{(u,r)\}$ be the corresponding min-cost r-tree in $\hat{G}$ and x be the corresponding incidence vector. Then, complementary slackness implies that for any $e \in E[X]$ for which $x_e > 0$ (and hence $e \in T$), it holds that the corresponding constraint in (DLP) is tight with respect to $\hat{y}$ (and also y). This implies that $e \in E_y$. On the other hand, for any $Z \subset X$, if $\hat{y}_Z > 0$, and hence $X \in \mathcal{F}$, complementary slackness prescribes that the corresponding primal constraint is tight, and hence $|T \cap \delta^+_{G[X]}(Z)| = 1$, concluding the proof. □

For the proof of the next theorem, we first explain how to compute the absorbing probabilities of an absorbing Markov chain (G, P) and show a related lemma that we need in Appendix C. W.l.o.g. we assume that the states $V(G)$ are ordered such that the non-absorbing states N come first and the absorbing states S last. We can then write the transition matrix as

$$
P = \begin{bmatrix} D & C \\ 0 & I_{|S|} \end{bmatrix} \quad ,
$$

where D is the $|N| \times |N|$ transition matrix from non-absorbing states to non-absorbing states and C is the $|N| \times |S|$ transition matrix from non-absorbing states to absorbing states. $I_{|S|}$ denotes the $|S| \times |S|$ identity matrix. The absorbing probability of an absorbing state $s \in S$, when starting a random walk in a state $v \in N$ is then given as the entry in the row corresponding to v and the column corresponding to s in the $|N| \times |S|$ matrix $(I_{|N|} - D)^{-1} C$ [Grinstead and Snell, 1997].

Lemma A.1. *Adding a self-loop to a non-absorbing state v with probability p and scaling all other transition probabilities from that state by $1-p$ does not change the absorbing probabilities of an absorbing Markov-chain (G, P).*

Proof. Let (D, C) and (D', C') be the transition matrices of the absorbing Markov chain before and after adding the self-loop. Let $\mathbf{d}_v, \mathbf{d}'_v, \mathbf{c}_v, \mathbf{c}'_v$ be the rows of D, D', C, C', corresponding to state v respectively. Then

$$\mathbf{d}'_v = (1-p)\mathbf{d}_v + p\mathbf{e}_v^\intercal \quad ,$$
$$\mathbf{c}'_v = (1-p)\mathbf{c}_v$$

and $\mathbf{d}'_u = \mathbf{d}_u$ and $\mathbf{c}'_u = \mathbf{c}_u$ for all $u \neq v$.

We want to show that $(I_{|N|} - D)^{-1}C = (I_{|N|} - D')^{-1}C'$. Let $Z = (I_{|N|} - D)^{-1}C$. Then $Z = (I_{|N|} - D')^{-1}C'$ if and only if $Z = D'Z + C'$. Notice, that only the row corresponding to v in D' and C' differ from D and C and therefore for all $u \neq v$

$$\mathbf{z}_u = \mathbf{d}_u Z + \mathbf{c}_u = \mathbf{d}'_u Z + \mathbf{c}'_u \quad ,$$

where $\mathbf{z}_u$ is the row of Z corresponding to u. The only thing left to show is $\mathbf{z}_v = \mathbf{d}'_v Z + \mathbf{c}'_v$. We have

$$\begin{aligned}
\mathbf{d}'_v Z + \mathbf{c}'_v &= ((1-p)\mathbf{d}_v + p\mathbf{e}_v^\intercal)Z + (1-p)\mathbf{c}_v \\
&= (1-p)\mathbf{d}_v Z + p\mathbf{e}_v^\intercal Z + (1-p)\mathbf{c}_v \\
&= (1-p)(\mathbf{d}_v Z + \mathbf{c}_v) + p\mathbf{e}_v^\intercal Z \\
&= (1-p)\mathbf{z}_v + p\mathbf{z}_v && \text{(since } DZ + C = Z) \\
&= \mathbf{z}_v \quad ,
\end{aligned}$$

which concludes the proof. □

Theorem 4 (★). *Algorithm 2 returns* MIXED BORDA BRANCHING *and runs in poly(n).*

Proof. We start by showing by induction that the given interpretation of the weight function on the nodes is correct, i.e., for any $v \in N$, $t_X(v)$ corresponds to the number of min-cost v-trees in the graph $G[X]$. The claim is clearly true for any singleton, since $t_{\{v\}}(v) = 1$ and the number of v-trees in $(\{v\}, \emptyset)$ is one, i.e., the empty set is the only v-trees. Now, we fix some $X \in \mathcal{F}'$ and assume that the claim is true for all children of X. In the following, we fix $v \in X$ and argue that the induction hypothesis implies that the claim holds for $t_X(v)$ as well.
For any node $u \in X$, let $Y_u \in \mathcal{F}$ be the child of X containing node u. Moreover, let $\mathcal{T}_v^*(G[X])$ (or short $\mathcal{T}_v^*$) be the set of min-cost v-trees in $G[X]$, and $\mathcal{T}_{Y_v}(G_X)$ (or short $\mathcal{T}_{Y_v}$) be the set of Y_v-trees in G_X. Lastly, for any $u \in X$, let $\mathcal{T}_u^*(G[Y_u])$ be the set of min-cost u-trees in $G[Y_u]$. In the following, we argue that there exists a many-to-one mapping from $\mathcal{T}_v^*$ to $\mathcal{T}_{Y_v}$. Note that, by statement (iv) in Lemma 2, every min-cost in-tree T in $G[X]$ (hence, $T \in \mathcal{T}_v^*$) leaves every child of X exactly once via a tight edge. Therefore, there exists a natural mapping to an element of $\mathcal{T}_{Y_v}$ by mapping every edge in T that connects two children of X to their corresponding edge in G_X. More precisely, $\hat{T} = \{(Y, Y') \in E_X \mid T \cap \delta^+(Y) \cap \delta^-(Y') \neq \emptyset\}$ is an Y_v-tree in G_X and hence an element of $\mathcal{T}_{Y_v}$.

Next, we want to understand how many elements of $\mathcal{T}_v^*$ map to the same element in $\mathcal{T}_{Y_v}$. Fix any $\hat{T} \in \mathcal{T}_{Y_v}$. We can construct elements of $\mathcal{T}_v^*$ by combining (an extended version of) $\hat{T}$ with min-cost in-trees within the children of X, i.e., with elements of the sets $\mathcal{T}_u^*(G[Y_u])$, $u \in X$. More precisely, for any edge $(Y, Y') \in \hat{T}$, we can independently chose any of the edges in $(u, u') \in E_y \cap (Y \times Y')$ and combine it with any min-cost u-tree in the graph $G[Y]$. This leads to

$$\Big(\prod_{(Y,Y')\in\hat{T}} \sum_{(u,u')\in E_y\cap(Y\times Y')} t_Y(u)\Big) t_{Y_v}(v) = \Big(\prod_{(Y,Y')\in\hat{T}} w_X(Y, Y')\Big) \cdot t_{Y_v}(v)$$

many different elements from $\mathcal{T}_v^*$ that map to $\hat{T} \in \mathcal{T}_{Y_v}$. Hence,

$$|\mathcal{T}_v^*| = \sum_{\hat{T}\in\mathcal{T}_{Y_v}} \prod_{(Y,Y')\in\hat{T}} w_Y(Y, Y') \cdot t_{Y_v}(v) = w_X(\mathcal{T}_{Y_v}) \cdot t_{Y_v}(v) = t_X(v),$$

where the last inequality follows from the definition of $t_X(v)$ in the algorithm. This proves the induction step, i.e., $t_X(v)$ corresponds to the number of min-cost v-trees in the graph $G[X]$.

Now, let $X = N \cup S$, i.e., we are in the last iteration of the algorithm. Due to an analogous reasoning as before, there is a many-to-one mapping from the min-cost branchings in G to branchings in G_X. More precisely, for every branching $B \in \mathcal{B}_{Y,\{s\}}(G_X)$, there exist

$$\prod_{(Y,Y')\in B} w_X(Y,Y') = w_X(B)$$

branchings in G that map to B. Hence, by the Markov chain tree theorem (Lemma 3), we get

$$A_{v,s} = Q_{v,s} = \frac{\sum_{B\in\mathcal{B}_{Y_v,\{s\}}(G_X)} w_X(B)}{\sum_{B\in\mathcal{B}(G_X)} w_X(B)} = \frac{\sum_{B\in\mathcal{B}^*_{v,s}(G)} 1}{\sum_{B\in\mathcal{B}^*(G)} 1} \quad ,$$

where (G'_X, P) is the Markov chain corresponding to G_X and $Q = \lim_{\tau\to\infty} \frac{1}{\tau} \sum_{i=0}^{\tau} P^\tau$. This equals the definition of MIXED BORDA BRANCHING.

Lastly, we argue about the running time of the algorithm. For a given delegation graph (G, c), let $n = V(G)$, i.e., the number of voters. Algorithm 1 can be implemented in $\mathcal{O}(n^3)$. That is because, the while loop runs for $\mathcal{O}(n)$ iterations (the laminar set family $\mathcal{F}$ can have at most $2n - 1$ elements), and finding all strongly connected components in a graph can be done in $\mathcal{O}(n^2)$ (e.g., with Kosaraju's algorithm [Hopcroft et al., 1983]). Coming back to the running time of Algorithm 2, we note that the do-while loop runs for $\mathcal{O}(n)$ iterations, again, due to the size of $\mathcal{F}'$. In line 7, the algorithm computes $\mathcal{O}(n)$ times the number of weighted spanning trees with the help of Lemma 1 (Tutte [1948]). Hence, the task is reduced to calculating the determinant of a submatrix of the laplacian matrix. Computing an integer determinant can be done in polynomial time in n and $\log(M)$, if M is an upper bound of all absolute values of the matrix[9]. Note, that all values in every Laplacian (the out-degrees on the diagonals and the multiplicities in the other entries) as well as the results of the computation are upper-bounded by the total number of branchings in the original graph G (this follows from our argumentation about the interpretation of $t_X(v)$ in the proof of Theorem 4), hence in particular by n^n. Therefore, the running time of each iteration of the do-while loop is polynomial in n. In the final step we compute the absorbing probabilities of the (scaled down version) of the weighted graph (G_X, w_X) (where $X = N \cup S$). For that, we need to compute the inverse of a $\mathcal{O}(n) \times \mathcal{O}(n)$ matrix, which can be done using the determinant and the adjugate of the matrix. Computing these comes down to computing $\mathcal{O}(n^2)$ determinants, for which we argued before that it is possible in polynomial time[10]. Summarizing, this gives us a running time of Algorithm 1 in $\mathcal{O}((n^7 \log(n) + n^4 \log(n \log(n))) * (\log^2 n + (\log(n \log(n)))^2))$. $\square$

B Further Remarks on Section 5

Alternative Interpretation of Algorithm 2 We stated Algorithm 2 in terms of counting min-cost branchings. There exists a second natural interpretation that is closer to the definition of the RANDOM WALK RULE, in which we want to compute the limit of the absorbing probabilities of a parametric Markov chain. We give some intuition on this reinterpretation of the algorithm with the example in Figure 2, and later extend this interpretation to a larger class of parametric Markov chains.

Intuitively speaking, every set $X \in \mathcal{F}$ in the Markov chain $(G, P^{(\varepsilon)})$ corresponding to the delegation graph G is a strongly connected component whose outgoing edges have an infinitely lower probability than the edges inside of X as ε approaches zero. We are therefore interested in the behavior of an infinite random walk in $G[X]$. While in the branching interpretation, the node weight $t_X(v)$ can be interpreted as the number of min-cost v-arborescences in $G[X]$, in the Markov chain interpretation we think of $t_X(v)$ as an indicator of the relative time an infinite random walk spends in v (or the relative number of times v is visited) in the Markov chain given by the strongly connected graph $G[X]$. Consider the example iteration depicted in Figure 2a, where we are given an unprocessed $X \in \mathcal{F}$ whose children Y_1, Y_2 are all processed. When contracting Y_1 and Y_2 the weights on the edges should encode how likely a transition is from one set to another, which is achieved by summing over the relative time spent in each node with a corresponding edge. We then translate the resulting

[9] More precisely, it can be computed in $\mathcal{O}((n^4 \log(nM) + n^3 \log^2(nM)) * (\log^2 n + (\log\log M)^2))$ [Gathen and Gerhard, 2013]

[10] We argued this only for integer matrices, but we can transform the rational matrix into an integer one by scaling it up by a factor which is bounded by n^n.

graph (Figure 2b) into a Markov chain and again compute the relative time spend in each node. This computation is equivalent to calculating the sum of weights of all in-trees (up to a scaling factor, see Theorem 3). Indeed, we get a ratio of one to three for the time spend in Y_1 and Y_2. To compute $t_X(v)$ we multiply the known weight $t_{Y_v}(v)$ by the newly calculated weight of Y_v. In the example this means that since we know, we spend three times as much time in Y_2 as in Y_1 all weights of nodes in Y_2 should be multiplied by three (see Figure 2c).

Extension of Algorithm 2 In addition, we remark that our algorithm could be extended to a larger class of parametric Markov chains, namely, to all Markov chains $(G, P^{(\varepsilon)})$, where G is a graph in which every node has a path to some sink node, and, for every $e \in E(G)$, $P_e^{(\varepsilon)}$ is some rational fraction in ε, i.e., $\frac{f_e(\varepsilon)}{g_e(\varepsilon)}$, where both f_e and g_e are polynomials in ε with positive coefficients.[11] Now, we can construct a cost function c on G, by setting $c(e) = x_e - z_e + 1$, where x_e is the smallest exponent in $f_e(\varepsilon)$ and z_e is the smallest exponent in $g_e(\varepsilon)$. Note that, if $c(e) < 1$, then the Markov chain cannot be well defined for all $\varepsilon \in (0, 1]$. Now, we run Algorithm 2 for the delegation graph (G, c) with the only one difference, i.e., the weight function w_X also has to incorporate the coefficients of the polynomials $f_e(\varepsilon)$ and $g_e(\varepsilon)$. More precisely, we define for every $e \in E$, the number q_e as the ratio between the coefficient corresponding to the smallest exponent in f_e and the coefficient corresponding to the smallest exponent in g_e. Then, we redefine line 4 in the algorithm to be

$$w_X(Y, Y') \leftarrow \sum_{(u,v) \in E_y \cap (Y \times Y')} t_Y(u) \cdot q_{(u,v)}.$$

One can then verify with the same techniques as in Section 4 and Section 5, that this algorithm returns the outcome of the above defined class of Markov chains.

C Missing Proofs and Further Results of Section 6

Theorem 6 (★). *The* RANDOM WALK RULE *satisfies anonymity.*

Proof. Since σ is a graph automorphism, we know that for all $v \in V(G)$ it holds that $|\delta^+(v)| = |\delta^+(\sigma(v))|$ and $c((v, w)) = c((\sigma(v), \sigma(w)))$ for any edge $(v, w) \in \delta^+(v)$. In the corresponding Markov chain M_ε we therefore get $P^{(\varepsilon)}_{(v,w)} = P^{(\varepsilon)}_{(\sigma(v),\sigma(w))}$ (see Equation 1). Since through the bijection between the edges of the graph, we also get a bijection between all walks in the graph $\mathcal{W}$ and for every $s \in S$ and walk in $\mathcal{W}[s, v]$ there is a corresponding walk in $\mathcal{W}[\sigma(v), \sigma(s)]$ of the same probability. Therefore we have

$$A_{v,s} = \lim_{\varepsilon \to 0} \sum_{W \in \mathcal{W}[v,s]} \prod_{e \in W} P_e^{(\varepsilon)} = \lim_{\varepsilon \to 0} \sum_{W \in \mathcal{W}[\sigma(v),\sigma(s)]} \prod_{e \in W} P_e^{(\varepsilon)} = A_{\sigma(v),\sigma(s)} \quad ,$$

which concludes the proof. □

Theorem 7 (★). *The* RANDOM WALK RULE *satisfies copy-robustness.*

Proof. Let (G, c), v, $(\hat{G}, c)$, A, $\hat{A}$ and S_v be defined as in the definition of copy-robustness. Let $(\mathcal{F}, y)$ and $(\hat{\mathcal{F}}, \hat{y})$ be the set families and functions returned by Algorithm 1 for G and $\hat{G}$, respectively. In this proof, we restrict our view to the subgraphs of only tight edges, denoted by $G_y = (N \cup S, E_y)$ and $\hat{G}_{\hat{y}} = (N \setminus \{v\} \cup V \cup \{v\}, E_{\hat{y}})$, respectively. Note, that this does not change the result of the RANDOM WALK RULE, since it is shown to be equal to MIXED BORDA BRANCHING, which only considers tight edges (in the contracted graph) itself.

First, we observe that the set S_v is exactly the subset of S reachable by v in G_y. This is because the assignment A returned by the RANDOM WALK RULE is given as the absorbing probability of a Markov chain on the graph (G_X, w_X) with $X = N \cup S$, computed by Algorithm 2. The graph is constructed from G_y by a number of contractions, which do not alter reachability, i.e. for $s \in S$ the node $\{s\}$ is reachable from the node Y_v containing v in G_X exactly if s is reachable from v in G_y. Since all edge weights w_X are strictly positive, in the corresponding Markov chain all transition probabilities on the edges of G_X are strictly positive as well. This gives $\{s\}$ a strictly positive absorbing probability when starting a random walk in Y_v exactly if s is reachable from v in G_y.

[11] This class is reminiscent of a class of parametric Markov chains studied by Hahn et al. [2011].

Our next observation is that $\hat{\mathcal{F}} = \mathcal{F} \setminus \{Y \in \mathcal{F} \mid v \in Y\} \cup \{\{v\}\}$, $\hat{y}(\{v\}) = 1$ and $y(Y) = \hat{y}(Y)$ for all $Y \in \hat{\mathcal{F}} \setminus \{\{v\}\}$. Consider the computation of $\mathcal{F}$ in Algorithm 1. Since the output is unique (see Lemma 2 statement (i)), we can assume without loss of generality that after initializing $\mathcal{F}$, all sets in $\{Y \in \mathcal{F} \mid v \notin Y\}$ are added to $\mathcal{F}$ first and then the remaining sets $\{Y \in \mathcal{F} \mid v \in Y\}$. In $\hat{G}$, the only edges missing are the outgoing edges from v, therefore, when applying Algorithm 1 to $\hat{G}$ all sets in $\{Y \in \mathcal{F} \mid v \notin Y\}$ can be added to $\hat{\mathcal{F}}$ first (with $\hat{y}(Y) = y(Y)$). Note, that the set $\{v\}$ with $y(\{v\}) = 1$ was added to $\hat{\mathcal{F}}$ in the initialization. We claim, that the algorithm terminates at that point. Suppose not, then there must be another strongly connected component $X \subseteq N$ with $\delta^+(X) \cap E_{\hat{y}} = \emptyset$. If $v \in X$ then since v has no outgoing edges $X = \{v\}$, which is already in $\mathcal{F}$. If $v \notin X$ then X would have already been added.

With these two observations, we can show the following claim: For every casting voter $s \in S \setminus S_v$ the voting weight remains equal, when v turns into a casting voter, i.e., $\pi_s(A) = \pi_s(\hat{A})$. Fix $s \in S \setminus S_v$ and let $U \subset N$ be the set of nodes not reachable from v in G_y. We know that $\hat{\mathcal{F}} = \mathcal{F} \setminus \{Y \in \mathcal{F} \mid v \in Y\} \cup \{v\}$, which implies that for every node $u \in U$ the sets containing u are equal in $\mathcal{F}$ and $\hat{\mathcal{F}}$, i.e., $\{Y \in \mathcal{F} \mid u \in Y\} = \{Y \in \hat{\mathcal{F}} \mid u \in Y\}$. Therefore, the outgoing edges from any $u \in U$ are equal in G_y and $\hat{G}_{\hat{y}}$. Since $\hat{\mathcal{F}} \subseteq \mathcal{F}$, the edges in $\hat{G}_{\hat{y}}$ are a subset of the edges in G_y and therefore the set U is not reachable from v in $\hat{G}_{\hat{y}}$. When translating $\hat{G}_{\hat{y}}$ into the Markov chain $(\hat{G}_{\hat{y}}, \hat{P}^{(\varepsilon)})$ (see Equation 1), we get for the probability of any tight out-edge e of u and any $\varepsilon > 0$, that $P^{(\varepsilon)}_e = \hat{P}^{(\varepsilon)}_e$, where $P^{(\varepsilon)}$ is the transition matrix induced by the original graph G_y. In the following we argue about the set of walks in G_y and $G_{\hat{y}}$. To this end we define for every $u \in N$, the set $\mathcal{W}[u, s]$ ($\hat{\mathcal{W}}[u, s]$, respectively) as the set of walks in G_y (in $G_{\hat{y}}$, respectively) that start in u and end in sink s. Since all walks from any $u \in U$ to s contain only outgoing edges from nodes in U, we have $\hat{\mathcal{W}}[u, s] = \mathcal{W}[u, s]$. For any other voter $w \in N \setminus U$ we have $\hat{\mathcal{W}}[w, s] = \mathcal{W}[w, s] = \emptyset$ and therefore

$$\pi_s(\hat{A}) = 1 + \sum_{u \in U} \lim_{\varepsilon \to 0} \sum_{\hat{W} \in \hat{\mathcal{W}}[u,s]} \prod_{e \in \hat{W}} P^{(\varepsilon)}_e = 1 + \sum_{u \in U} \lim_{\varepsilon \to 0} \sum_{W \in \mathcal{W}[u,s]} \prod_{e \in W} P^{(\varepsilon)}_e = \pi_s(A) \quad ,$$

which concludes the proof of the claim.

Summarizing, we know that that for any casting voter $s \in S \setminus S_v$ we have $\pi_s(A) = \pi_s(\hat{A})$, which directly implies that $\sum_{s \in S_v} \pi_s(A) = \pi_v(\hat{A}) + \sum_{s \in S_v} \pi_s(\hat{A})$. □

Theorem 8 (★). *The* RANDOM WALK RULE *satisfies confluence.*

Proof. Before proving the claim, we introduce notation. For any walk W in some graph G, and some node $v \in V(G)$, we define $W[v]$ to be the subwalk of W that starts at the first occasion of v in W. For two nodes $u, v \in V(G)$, we define $W[u, v]$ to be the subwalk of W that starts at the first occasion of u and ends at the first occasion of v. (Note that $W[v]$ and $W[u, v]$ might be empty.) Now, for a set of walks $\mathcal{W}$ and $u, v, s \in V(G)$, we define $\mathcal{W}[v] = \{W[v] \mid W \in \mathcal{W}\}$ and $\mathcal{W}[u, v] = \{W[u, v] \mid W \in \mathcal{W}\}$. Lastly, we define $\mathcal{W}[u, v, s] = \{W \in W[u, s] \mid v \in W[u, s]\}$. We usually interpret a walk W as a sequence of nodes. In order to facilitate notation, we abuse notation and write $v \in W$ for some node $v \in V(G)$ in order to indicate that v appears in W, and for an edge $e \in E(G)$, we write $e \in W$ to indicate that tail and head of e appear consecutively in W.

For the remainder of the proof we fix $\mathcal{W}$ to be the set of walks in the input delegation graph G starting in some node from N and ending in some sink node S. Moreover, let G_X be the graph at the end of Algorithm 2, i.e., G_X for $X = N \cup S$. We fix $\hat{\mathcal{W}}$ to be the set of walks which start in some node of G_X and end in some sink node of G_X (which are exactly the nodes in $\{\{s\} \mid s \in S\}$).

In the following, we define for every $v \in N$ a probability distribution $f_v : \mathcal{W}[v] \to [0, 1]$, such that it witnesses the fact that the RANDOM WALK RULE is confluent. To this end, we define a mapping $\gamma_v : \hat{\mathcal{W}}[Y_v] \to \mathcal{W}[v]$, where Y_v is the node in G_X that contains v. Given a walk $\hat{W} \in \hat{\mathcal{W}}[Y_v]$, we construct $\gamma_v(\hat{W}) \in \mathcal{W}[v]$ as follows: Let $\hat{W} = Y^{(1)}, \dots Y^{(k)}$. By construction of G_X we know that for every $i \in \{1, \dots, k\}$, the fact that $(Y^{(i)}, Y^{(i+1)}) \in E_X$ implies that there exists $(b^{(i)}, a^{(i+1)}) \in E$ with $b^{(i)} \in Y^{(i)}$ and $a^{(i+1)} \in Y^{(i+1)}$. Moreover, we define $a^{(1)} = v$ and $b^{(n)} = s$, where $\{s\} = Y^{(k)}$. Under this construction it holds that $a^{(i)}, b^{(i)} \in Y^{(i)}$ for all $i \in \{1, \dots, k\}$, but the two nodes may differ. Therefore, we insert subwalks $W^{(i)}$ connecting $a^{(i)}$ to $b^{(i)}$ by using only nodes in $Y^{(i)}$ and visiting each of these nodes at least once. The final walk $\gamma_v(\hat{W})$ is then defined by $(a^{(1)}, W^{(1)}, b^{(1)}, \dots, a^{(n)}, W^{(n)}, b^{(n)})$. We remark that this mapping is injective, and it holds that $\hat{W}$ visits some node $Y \in V(G_X)$ if and only if $\gamma_v(\hat{W})$ visits all nodes in Y.

Recall that the assignment A of the RANDOM WALK RULE can be computed via a Markov chain (G'_X, P) derived from the contracted graph (G_X, w_X) (see Section 4 and Section 5), where G'_X is derived from G_X by adding self-loops. In Lemma A.1 we show that introducing (and thus removing) self-loops to states in an absorbing Markov chain does not change its absorbing probabilities. We retrieve the Markov chain $(G_X, \hat{P})$ by removing all self loops of all voters in N and rescaling the other probabilities accordingly. We then make use of this Markov chain in order to define f_v over $\mathcal{W}[v]$. That is, for any $W \in \mathcal{W}[v]$ let

$$f_v(W) = \begin{cases} \prod_{e \in \hat{W}} \hat{P}_e & \text{if there exists } \hat{W} \in \hat{\mathcal{W}}[Y_v] \text{ such that } \gamma_v(\hat{W}) = W \\ 0 & \text{else.} \end{cases}$$

Note that, the above expression is well defined since γ_v is injective.

In the remainder of the proof, we show that f_v witnesses the confluence of the RANDOM WALK RULE. First, we show that f_v is indeed consistent with the assignment A returned by RANDOM WALK RULE. That is, for any $v \in N$ and $s \in S$ it holds that

$$\mathbb{P}_{W \sim f_v}[s \in W] = \sum_{W \in \mathcal{W}[v,s]} f_v(W) = \sum_{\hat{W} \in \hat{\mathcal{W}}[Y_v, \{s\}]} \prod_{e \in \hat{W}} \hat{P}_e = A_{v,s} \quad .$$

The second equality comes from the fact that γ_v is injective and exactly those walks in $\hat{\mathcal{W}}[Y_v, \{s\}]$ are mapped by γ_v to walks in $\mathcal{W}[v, s]$. Moreover, all walks in $\mathcal{W}[v, s]$ that have no preimage in $\hat{\mathcal{W}}[Y_v, \{s\}]$ are zero-valued by f_v. The last equality comes from the fact that $A_{v,s}$ equals the probability that the Markov chain (G'_X, P) (equivalently, $(G_X, \hat{P})$) reaches $\{s\}$ if started in Y_v (see Section 4 and Section 5).

We now turn to the second condition on the family of probability distributions $f_v, v \in N$. That is, for every $u, v \in N, s \in S$ it holds that

$$\begin{aligned}
\mathbb{P}_{W \sim f_u}[v \in W \wedge s \in W] &= \sum_{W \in \mathcal{W}[u,v,s]} f_u(W) = \sum_{\hat{W} \in \hat{\mathcal{W}}[Y_u, Y_v, \{s\}]} \prod_{e \in \hat{W}} \hat{P}_e \\
&= \sum_{\hat{W} \in \hat{\mathcal{W}}[Y_u, Y_v, \{s\}]} \Big(\prod_{e \in \hat{W}[Y_u, Y_v]} \hat{P}_e \Big) \Big(\prod_{e \in \hat{W}[Y_v, \{s\}]} \hat{P}_e \Big) \\
&= \Big(\sum_{\hat{W} \in \hat{\mathcal{W}}[Y_u, Y_v]} \prod_{e \in \hat{W}} \hat{P}_e \Big) \cdot \Big(\sum_{\hat{W} \in \hat{\mathcal{W}}[Y_v, \{s\}]} \prod_{e \in \hat{W}} \hat{P}_e \Big) \\
&= \Big(\sum_{s' \in S} \sum_{\hat{W} \in \hat{\mathcal{W}}[Y_u, Y_v, \{s'\}]} \prod_{e \in \hat{W}} \hat{P}_e \Big) \cdot \Big(\sum_{\hat{W} \in \hat{\mathcal{W}}[Y_v, \{s\}]} \prod_{e \in \hat{W}} \hat{P}_e \Big) \\
&= \Big(\sum_{s' \in S} \sum_{W \in \mathcal{W}[u,v,s']} f_u(W) \cdot \Big(\sum_{W \in \mathcal{W}[v,s]} f_v(W) \Big) \\
&= \mathbb{P}_{W \sim f_u}[v \in W] \cdot \mathbb{P}_{W \sim f_v}[s \in W].
\end{aligned}$$

The second equality follows from the same reason as above, i.e., γ_v is injective, exactly those walks in $\hat{\mathcal{W}}[Y_u, Y_v, \{s\}]$ are mapped by γ_v to walks in $\mathcal{W}[u, v, s]$, and all walks in $\mathcal{W}[u, v, s]$ that have no preimage in $\hat{\mathcal{W}}[Y_u, Y_v, \{s\}]$ are zero-valued by f_v. The third inequality holds by the fact that every walk that is considered in the sum can be partitioned into $\hat{W}[Y_u, Y_v]$ and $\hat{W}[Y_v, \{s\}]$. The fourth equality follows from factoring out by the subwalks. The fifth equality follows from the fact that every walk in $\hat{\mathcal{W}}$ reaches some sink node eventually, and therefore, the additional factor in the first bracket sums up to one. Lastly, the sixth equality follows from the very same argument as before.

From the above equation we get in particular that for every $u, v \in N, s \in S$ it holds that

$$\mathbb{P}_{W \sim f_u}[s \in W \mid v \in W] = \frac{\mathbb{P}_{W \sim f_u}[s \in W \wedge v \in W]}{\mathbb{P}_{W \sim f_u}[v \in W]} = \mathbb{P}_{W \sim f_v}[s \in W].$$

This concludes the proof. $\square$

The next axiom was in its essence first introduced by Behrens and Swierczek [2015] and first given the name *guru-participation* in Kotsialou and Riley [2020]. The idea is that a representative (the *guru*) of a voter, should not be worse off if said voter abstains from the vote. Brill et al. [2022] define this property for non-fractional ranked delegations by requiring that any casting voter that was not a representative of the newly abstaining voter should not loose voting weight. This definition translates well into the setting of fractional delegations where we can have multiple representatives per voter. For simplicity, we made a slight modification to the definition[12], resulting in a slightly stronger axiom.

Previously, we stated the general assumption that every delegating voter in a delegation graph (G, c) has a path to some casting voter in G. In this section we modify given delegation graphs by removing nodes or edges, which may result in an invalid delegation graph not satisfying this assumption. To prevent this, we implicitly assume that after modifying a delegation graph, all nodes in N not connected to any sink in S (we call them *isolated*) are removed from the graph.

Guru Participation: A delegation rule satisfies *guru-participation* if the following holds for every instance (G, c): Let $(\hat{G}, c)$ be the instance derived from (G, c) by removing a node $v \in N$ (and all newly isolated nodes), let $S_v = \{s \in S \mid A_{v,s} > 0\}$ be the set of representatives of v and let A and $\hat{A}$ be the assignments returned by the delegation rule for (G, c) and $(\hat{G}, c)$, respectively. Then

$$\pi_s(\hat{A}) \geq \pi_s(A) \quad \forall s \in S \setminus S_v \quad .$$

In particular, this implies

$$\sum_{s \in S_v} \pi_s(\hat{A}) + 1 \leq \sum_{s \in S_v} \pi_s(A) \quad .$$

In order to prove that the Random Walk Rule satisfies guru-participation we first show the following lemma, saying that the voting weight of no casting voter decreases, when the in-edges of another casting voter are removed from the graph.

Lemma C.1. *For the* Random Walk Rule*, removing the incoming edges of some casting voter $s \in S$ (and all newly isolated voters) does not decrease the absolute voting weight of any casting voter $s' \in S \setminus \{s\}$.*

Proof. Let (G, c) be a delegation graph and $s \in S$ a sink. Let $(\hat{G}, c)$ be the delegation graph, where the in-edges of s and all voters disconnected from casting voters are removed. Let $P^{(\varepsilon)}$ and $\hat{P}^{(\varepsilon)}$ be the transition matrices of the corresponding Markov chains M_ε and $\hat{M}_\varepsilon$. Then, for any $\varepsilon > 0$ and edge e in $\hat{G}$ we have $P_e^{(\varepsilon)} \leq \hat{P}_e^{(\varepsilon)}$. Since no edge on a path from any $v \in N$ to any $s' \in S \setminus \{s\}$ was removed, we have $\hat{\mathcal{W}}[v, s'] = \mathcal{W}[v, s']$ and $\hat{P}_e^{(\varepsilon)} \geq P_e^{(\varepsilon)}$ for every edge e in $\hat{G}$ and $\varepsilon > 0$. Therefore, for the absolute voting weight of any $s' \in S \setminus \{s\}$ in $\hat{G}$ we get

$$\pi_{s'}(\hat{A}) = 1 + \sum_{v \in N} \lim_{\varepsilon \to 0} \sum_{\hat{W} \in \hat{\mathcal{W}}[v,s']} \prod_{e \in \hat{W}} P_e^{(\varepsilon)} \geq 1 + \sum_{v \in N} \lim_{\varepsilon \to 0} \sum_{W \in \mathcal{W}[v,s']} \prod_{e \in W} P_e^{(\varepsilon)} = \pi_{s'}(A) \quad ,$$

which concludes the proof. $\square$

Using Lemma C.1 and the proof of Theorem 7, we can show that guru-participation is satisfied by the Random Walk Rule by removing a delegating voter step by step.

Theorem C.2. *The* Random Walk Rule *satisfies guru participation.*

Proof. Let (G, c) be a delegation graph and $v \in N$ a delegating voter. We remove v from G in three steps. First, we remove all out-edges of v, making v a casting voter and call the new delegation graph $(\hat{G}_1, c)$. Then we remove the in-edges of v (and all newly isolated voters) and get $(\hat{G}_2, c)$. Finally, we remove v itself to retrieve $(\hat{G}, c)$ as in the definition of guru-participation. Let A, $\hat{A}_1$, $\hat{A}_2$ and $\hat{A}$ be the assignments returned by the Random Walk Rule for (g, c), $(\hat{G}_1, c)$, $(\hat{G}_2, c)$ and $(\hat{G}, c)$, respectively. From the proof of Theorem 7 we know that for every casting voter $s \in S \setminus S_v$ the voting weight in the instances (G, c) and $(\hat{G}_1, c)$ is equal, i.e., $\pi_s(\hat{A}_1) = \pi_s(A)$. From Lemma

[12] More specifically, Brill et al. [2022] use the notion of *relative* voting weight between the casting voters in the definition of the axiom, which follows from our version of the axiom using absolute voting weight.

C.1 it follows that the voting weight of these voters can only increase if also the in-edges of v are removed, i.e., $\pi_s(\hat{A}_2) \geq \pi_s(\hat{A}_1)$. Finally, removing the now completely isolated (now casting) voter v does not change the absolute voting weight of any other voter and therefore $\pi_s(\hat{A}) \geq \pi_s(A)$. □

Top-rank priority: For any delegation graph and output of the delegation rule A, if voter $v \in N$ has exactly one outgoing edge of cost 1 and that edge ends in a casting voter $s \in S$, then $A_{v,s} = 1$.

Theorem C.3. MIXED BORDA BRANCHING *satisfies top-rank priority.*

Proof. Let G, v and s be defined as above. We show that for the assignment returned by MIXED BORDA BRANCHING $A_{v,s} = 1$ by showing that every Borda branching contains the edge (v, s). Suppose there is a Borda branching B' with $(v, s) \notin B$, then we construct a new branching $\hat{B}$ by removing the out-edge of v from B' and adding (v, s) instead. $\hat{B}$ is a branching, since no cycles can be introduced by adding an edge to a sink and $|\hat{B}| = |B'|$. Since v has only one outgoing edge of cost one, $\hat{B}$ has lower total cost that B', contradicting the assumption that B' is a Borda branching. □

D Broader Impact

We are aware of the fact that any delegation rule, and in particular the one suggested in this paper, may be implemented in a liquid democracy system and could thereby have real world impact. In this paper, we chose the axiomatic method in order to evaluate the suggested rule in a principled way. While, with respect to the axioms considered in the literature so far, our delegation rule performs very well, we want to point out that this is the very first paper introducing fractional delegation rules for ranked delegations. In particular, there is a risk of some unforeseen disadvantages of the rule that could possibly be used for manipulations or lead to other negative societal effects. Therefore, we think that further theoretical and also empirical research is necessary before recommending our suggested delegation rule for (high-stake) real-world decision making.