Designing Participatory Budgeting Mechanisms Grounded in Judgment Aggregation[⋆]

Simon Rey, Ulle Endriss, Ronald de Haan

Abstract

We introduce a new approach for designing rules for participatory budgeting (PB), the problem of deciding on the use of public funds based directly on the views expressed by the citizens concerned. The core idea is to embed instances of the participatory budgeting problem into judgment aggregation, a powerful general-purpose framework for modelling collective decision making. Taking advantage of the possibilities offered by judgment aggregation, we enrich the familiar setting of participatory budgeting with additional constraints, namely dependencies between projects and quotas regarding different types of projects. We analyse the rules obtained in both algorithmic and axiomatic terms.

1 Introduction

Participatory budgeting (PB) is an instrument intended to improve the democratic process by allowing citizen to directly express their views regarding the use of public funds [6]. Since its first use for municipal budget, PB has now been adopted across the world [38]. PB proceeds in two stages. First, citizens submit project proposals, some of which are shortlisted. Then they vote on which of the shortlisted projects to fund, given the limitations set by the budget available. In this paper we introduce a new approach for designing voting rules for this second stage. The central idea is to embed PB into judgment aggregation (JA), a highly expressive framework for collective decision making that has been extensively studied in the field of (computational) social choice [13, 31].

Most existing formal work in social choice theory regarding PB views voting on projects as a generalisation of approval-based multiwinner voting [see, e.g., 2, 1, 39]. While this can provide useful intuitions regarding, for instance, the type of normative desiderata we may wish to postulate for PB, it does not allow for great flexibility when it comes to modelling expressive forms of PB [26, 27, 35] as it requires each time to introduce a new model and to re-prove previous results. This is why we take a complementary approach and study project selection as a special case of the more general problem of JA. This allows us to prove general results which will apply to any generalisation of PB that fits within our framework.

While JA is very expressive—e.g., it naturally generalises many forms of preference aggregation and voting [10, 29, 14]—computing outcomes for JA is typically computationally intractable [16]. The central challenge we address in this paper thus is to find ways of implementing PB via JA in an *efficient* manner. To do so, the core idea we explore is to look for tractable fragments of JA, by further developing the approach of De Haan [24] of modelling JA problems using Boolean circuits in decomposable negation normal form (DNNF). This allows us to model PB problems with multiple resources, dependencies between projects, and quotas of different types of projects.

Of course, an expressive framework for modelling PB scenarios and a set of algorithmically efficient PB rules alone are not sufficient. We also require a good understanding of whether

[⋆]This is an abridged version of a paper originally appearing in the proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning [37].

the rules we design are normatively adequate. We therefore provide an *axiomatic analysis* of the rules we propose, focusing on the notion of *exhaustiveness* (ruling out any under-use of the budget) and the *monotonicity axioms* proposed by Talmon and Faliszewski [39].

Related work. According to the terminology of Aziz and Shah [1], we focus on *combinatorial PB with binary projects and approval ballots.* For this framework, Aziz et al. [2] and Talmon and Faliszewski [39] analysed several rules in both axiomatic and algorithmic terms, proposing greedy algorithms and dynamic programming techniques. Using a different approach, Fain et al. [19] and Freeman et al. [21] instead studied PB solutions as market equilibria, in the spirit of the public decision making setting of Conitzer et al. [7]. Particularly relevant to our work, Fain et al. [20] considered a general setting of public decision making with matroid, matching, and packing constraints, allowing for great flexibility on what can be modelled. Further generalisations of PB have been introduced: Jain et al. [26] studied the computational complexity of PB when projects have types and utilities are defined over the types, Patel et al. [35] focused on fairness in a similar setting, Jain et al. [27] extended these analyses to a more general framework. Lu and Boutilier [32] considered yet another extension of PB, where the cost of a project might depend on the number of agents choosing it. Finally, De Haan [24] was the first to discuss the idea of embedding PB into JA.

Paper outline. We recall relevant definitions from PB and JA in Section 2 and then introduce our central definition of an *embedding* of PB into JA. Section 3 is devoted to the study of efficient embeddings for basic PB and the extensions we propose, Section 4 discusses exhaustiveness, and Section 5 contains the remainder of our axiomatic analysis. Missing proofs can be found in the full version of this paper [37].

2 Frameworks

In this section we recall basic definitions regarding the frameworks of participatory budgeting (PB) and judgment aggregation (JA). We also define the main concept of this paper, namely embeddings of PB instances into JA.

2.1 Participatory Budgeting

We mainly adopt the notation of Aziz and Shah [1]. PB is about selecting a set of projects to be funded, given a (possibly multi-dimensional) budget limit. The set of (binary) *projects* is denoted by $\mathcal{P} = \{p_1, \dots, p_m\}$. Let $\mathcal{R} = \{r_1, \dots, r_d\}$ be a set of *resources* and $\boldsymbol{b} = (b_1, \dots, b_d)$ a *budget limit vector*, with $b_i \in \mathbb{R}_{\geq 0}$ indicating the limit in terms of resource r_i. The costs of the projects are defined by a *cost function* $c : \mathcal{P} \times \mathcal{R} \to \mathbb{R}_{\geq 0}$, indicating for a given project the cost in terms of the given resource. Slightly overloading notation, we use $c(p) = (c(p, r_1), \dots, c(p, r_d))$ to denote the cost vector of project p. Moreover, for any subset $P \subseteq \mathcal{P}$, let $c(P, r) = \sum_{p \in P} c(p, r)$ and $c(P) = \sum_{p \in P} c(p)$. A *problem instance* $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ for PB consists of a set of resources, a budget limit vector, a set of projects, and a cost function. $\mathcal{I}$ is the set of all such instances.

A solution of a PB problem instance, called a *budget allocation*, is a subset of projects $A \subseteq \mathcal{P}$. A budget allocation A is said to be *feasible* if $c(A) \leq \boldsymbol{b}$. For a given $I \in \mathcal{I}$, the set of all feasible budget allocations is denoted by $\mathcal{A}(I)$.

Before deciding which budget allocation to recommend, we consult the *agents* belonging to a set $\mathcal{N} = \{1, \dots, n\}$. Each agent $i \in \mathcal{N}$ submits an *approval ballot* $A_i \subseteq \mathcal{P}$, giving rise to a *profile* $\boldsymbol{A} = (A_1, \dots, A_n)$. For any given project p, its *approval score* under profile $\boldsymbol{A}$ is $\sum_{i \in \mathcal{N}} \mathbb{1}_{p \in A_i}$, the number of agents approving of p. W.l.o.g., we assume that every project has an approval score of at least 1, as projects with approval score 0 can be removed in a pre-processing step. Finally, a *PB rule* is a function $F : \mathcal{I} \times (2^{\mathcal{P}})^n \to 2^{2^{\mathcal{P}}} \setminus \{\emptyset\}$ mapping any given instance I and profile $\boldsymbol{A}$ to a nonempty set $F(I, \boldsymbol{A}) \subseteq \mathcal{A}(I)$ of feasible budget allocations.[1] Returning a set allows us to model ties.

2.2 Judgment Aggregation

The JA framework we use is known as *binary aggregation with integrity constraints* [23].[2]

Let $\mathcal{L}_{\boldsymbol{P}}$ be the set of propositional formulas over a given set $\boldsymbol{P}$ of propositional atoms, using the usual connectives $\neg$, $\vee$, $\wedge$, $\rightarrow$, and logical constants $\bot$ and $\top$. Propositional atoms and their negations are called literals. For any $P \subseteq \boldsymbol{P}$, we write $\mathit{Lit}(P) = P \cup \{\neg x \mid x \in P\}$ for the set of literals corresponding to P. We often use x_i to denote atoms and ℓ_{x_i} to denote literals corresponding to x_i, i.e., $\ell_{x_i} \in \{x_i, \neg x_i\}$. We say that ℓ_{x_i} is positive if $\ell_{x_i} = x_i$ and negative if $\ell_{x_i} = \neg x_i$. A truth assignment $\alpha : \boldsymbol{P} \rightarrow \{0, 1\}$ is a mapping indicating for each atom its truth value. For $\ell_{x_i} \in \mathit{Lit}(\boldsymbol{P})$, set $\alpha(\ell_{x_i}) = \alpha(x_i)$ if ℓ_{x_i} is positive and $\alpha(\ell_{x_i}) = 1 - \alpha(x_i)$ otherwise. We write $\alpha \models \varphi$ whenever α is a model of φ.

In the context of JA, the atoms in $\boldsymbol{P}$ represent *propositions* an agent may either *accept* or *reject*. A *judgment* J is a set $J \subseteq \boldsymbol{P}$, indicating which propositions are accepted. Let $\mathit{ext}(J) = J \cup \{\neg x \mid x \in \boldsymbol{P} \setminus J\}$ for any given judgment J. Observe that a judgment J can be equivalently described as the truth assignment α such that $\alpha(x) = 1$ if and only if $x \in J$. In our examples, when we do not explicitly specify some propositions, it is assumed that we only consider judgments (and truth assignments) for which the unspecified propositions are rejected (mapped to 0).

An *integrity constraint* $\Gamma \in \mathcal{L}_{\boldsymbol{P}}$ is a formula used to constrain the range of admissible judgments. A judgment J *satisfies* Γ (written $J \models \Gamma$), if J, interpreted as a truth assignment, is a model of Γ. Let $\mathfrak{J}(\Gamma) = \{J \subseteq \boldsymbol{P} \mid J \models \Gamma\}$. A *problem instance* for JA is simply an integrity constraint Γ.

We again use $\mathcal{N} = \{1, \ldots, n\}$ to denote the set of *agents*. Each agent $i \in \mathcal{N}$ provides us with a judgment J_i, resulting in a *judgment profile* $\boldsymbol{J} = (J_1, \ldots, J_n)$. For a profile $\boldsymbol{J}$ and a literal $\ell \in \mathit{Lit}(\boldsymbol{P})$, we write $n_\ell^{\boldsymbol{J}} = \sum_{i \in \mathcal{N}} \mathbb{1}_{\ell \in \mathit{ext}(J_i)}$ for the number of supporters of ℓ. The *majoritarian outcome* for a profile, denoted by $m(\cdot)$, is the set of literals supported by a majority of agents: $m(\boldsymbol{J}) = \{\ell \in \mathit{Lit}(\boldsymbol{P}) \mid n_\ell^{\boldsymbol{J}} > \frac{n}{2}\}$.

A *JA rule* is a function $F : \mathcal{L}_{\boldsymbol{P}} \times (2^{\boldsymbol{P}})^n \rightarrow 2^{2^{\boldsymbol{P}}} \setminus \{\emptyset\}$ taking as input an integrity constraint Γ and a judgment profile $\boldsymbol{J}$ and returning a nonempty set $F(\Gamma, \boldsymbol{J}) \subseteq \mathfrak{J}(\Gamma)$ of admissible judgments. Observe that no assumption is made about the profile. In particular, we do not require $J_i \models \Gamma$ for any $i \in \mathcal{N}$. We refer the reader to the literature on JA [13] for a discussion on what makes some rules attractive or not.

Before reviewing a number of well-known concrete JA rules, let us first introduce a very general class of such rules.

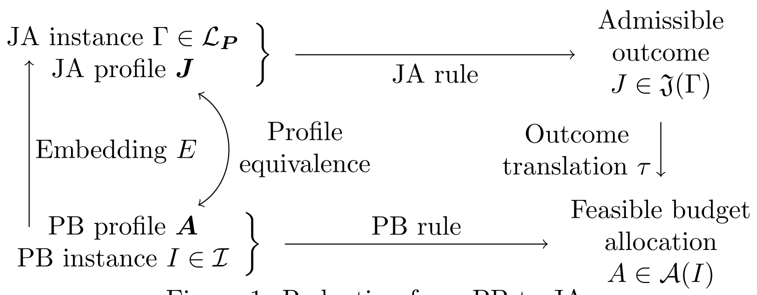
Definition 1 (Additive rules)**.** *A judgment aggregation rule F is an additive rule if there exists a function $f : \left(2^{\boldsymbol{P}}\right)^n \times \mathit{Lit}(\boldsymbol{P}) \rightarrow \mathbb{R}$ such that, for every integrity constraint Γ and every profile $\boldsymbol{J} \in \left(2^{\boldsymbol{P}}\right)^n$, we have:*

$$F(\Gamma, \boldsymbol{J}) \quad = \quad \operatorname*{argmax}_{J \in \mathfrak{J}(\Gamma)} \sum_{\ell \in \mathit{ext}(J)} f(\boldsymbol{J}, \ell).$$

This class generalises both the *scoring rules* of Dietrich [9] and the *additive majority rules* (AMRs) defined by Nehring and Pivato [34]. A scoring rule is associated with a scoring function $s : 2^{\boldsymbol{P}} \times \mathit{Lit}(\boldsymbol{P}) \rightarrow \mathbb{R}$ and corresponds to the additive rule with $f(\boldsymbol{J}, \ell) =$

[1] Observe that $\mathcal{A}(I)$ is never empty as the empty set is always feasible. This is not true for the extensions discussed in Section 3.

[2] While this framework is most convenient for our purposes, the original framework of List and Pettit [31] could be used as well, given that it is known that the former can be efficiently embedded into the latter [17].



JA instance $\Gamma \in \mathcal{L}_{\boldsymbol{P}}$
JA profile $\boldsymbol{J}$
JA rule
Admissible outcome $J \in \mathfrak{J}(\Gamma)$
Embedding E
Profile equivalence
Outcome translation τ
PB profile $\boldsymbol{A}$
PB instance $I \in \mathcal{I}$
PB rule
Feasible budget allocation $A \in \mathcal{A}(I)$

Figure 1: Reduction from PB to JA

$\sum_{i \in \mathcal{N}} s(J_i, \ell)$. An AMR is associated with a non-decreasing gain function $g : \{0, \ldots, n\} \to \mathbb{R}$ with $g(k) < g(k')$ for any $k < \frac{n}{2} \leq k'$ and is an additive rule with $f(\boldsymbol{J}, \ell) = g(n_\ell^{\boldsymbol{J}})$. We focus on three additive rules: the Kemeny and the Slater rules as they are prominent in the literature on JA and the leximax rule as it is similar to the greedy approval rule for PB.

- The *Slater rule* [33, 28] selects the admissible outcome closest to the majoritarian outcome in terms of the number of propositions they agree on. It is the AMR associated with the gain function g with $g(x) = 1$ if $x \geq \frac{n}{2}$ and $g(x) = 0$ otherwise.
- The *Kemeny rule* [36, 33] selects the feasible outcome that is the closest to the profile as a whole. It is both an AMR with $g(x) = x$ and a scoring rule with $s(J, \ell) = \mathbb{1}_{\ell \in ext(J)}$.
- The *leximax rule* [18, 34] favours the propositions supported by the largest majorities. It is the AMR defined by the gain function $g(x) = |\boldsymbol{P}|^x$.

Note that the three rules presented above are all *majority-consistent*, meaning that whenever the majoritarian outcome is admissible, it is the unique judgement returned by the rules.

2.3 Embedding PB into JA

The aim of this paper is to design rules to solve PB problems. To this end, we want to embed PB into JA and then use JA rules to compute budget allocations (see also Figure 1).

For a given PB instance, we introduce one proposition for each project to obtain $\boldsymbol{P}$. So we have a direct correspondence between budget allocations $A \subseteq \mathcal{P}$ and judgments $J \subseteq \boldsymbol{P}$, and thus also between PB profiles and JA profiles. Similarly, any JA outcome can be translated back into the PB setting.

Definition 2 (Outcome translation). *Let $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ be a PB instance and let $\Gamma \in \mathcal{L}_{\boldsymbol{P}}$ be an integrity constraint expressed over the atoms $\boldsymbol{P} = \{x_p \mid p \in \mathcal{P}\}$. The outcome translation $\tau : 2^{\boldsymbol{P}} \to 2^{\mathcal{P}}$ maps any judgment $J \in 2^{\boldsymbol{P}}$ to a budget allocation $A = \tau(J) = \{p \in \mathcal{P} \mid x_p \in J\}$.*

We moreover extend the outcome translation to sets $\mathcal{J} \subseteq 2^{\boldsymbol{P}}$ of judgments by stipulating that $\tau(\mathcal{J}) = \{\tau(J) \mid J \in \mathcal{J}\}$.

An *embedding* is a function $E : \mathcal{I} \to \mathcal{L}_{\boldsymbol{P}}$ that takes a PB instance as input and returns an integrity constraint (i.e., a JA instance). Given an embedding, we can now translate any input of a PB rule into an input for a JA rule, apply the JA rule, and finally translate the result obtained into a set of budget allocations (see Figure 1). However, to be meaningful, the integrity constraint should express the budget constraint of the PB instance. This is captured by the notion of correctness.

Definition 3 (Correct embedding). *An embedding $E : \mathcal{I} \to \mathcal{L}_{\boldsymbol{P}}$ is said to be correct if, for every PB instance $I \in \mathcal{I}$, we have $\tau(\mathfrak{J}(E(I))) = \mathcal{A}(I)$.*

3 Efficient Embeddings

In this section we present specific embeddings of enriched PB instances into JA. Given that the problem of computing outcomes for the JA rules defined in Section 2.2 is known to be highly intractable [30, 15], we need to ensure that PB instances are mapped into JA instances that permit efficient outcome determination. To this end, we first present a class of Boolean functions (to encode integrity constraints) for which the outcome determination can be solved efficiently.

3.1 Tractable Languages for JA

As shown by De Haan [24], computing outcomes under Kemeny and Slater can be done efficiently when the integrity constraint is a Boolean circuit in decomposable negation normal form (DNNF) [8]. We are going to extend this result to all additive rules.

Definition 4 (DNNF circuits)**.** *A circuit in negation normal form (NNF) is a rooted directed acyclic graph whose leaves are labelled with $\top, \bot, x$ or $\neg x$, for $x \in \boldsymbol{P}$ and whose internal nodes are labelled with $\wedge$ or $\vee$. A DNNF circuit C is an NNF circuit that is decomposable: for every conjunction in C, no two conjuncts share a common propositional variable.*

For a given JA rule F, we define the outcome determination problem $\textsc{Outcome}(F)$ as follows: given an integrity constraint Γ, a judgment profile $\boldsymbol{J}$, and a subset of literals $L \subseteq \mathit{Lit}(\boldsymbol{P})$; is there a $J \in F(\Gamma, \boldsymbol{J})$ such that $L \subseteq \mathit{ext}(J)$? We can show that for any additive JA rule F we can solve $\textsc{Outcome}(F)$ efficiently when Γ is a DNNF circuit.

Theorem 1. *Let F be an additive JA rule defined w.r.t. some polynomial-time computable function f. Then $\textsc{Outcome}(F)$ is polynomial-time solvable if the integrity constraint Γ in the input is represented as a DNNF circuit.*

This general result immediately implies tractability of outcome determination for the rules we are interested in here and will allow us to use these rules to compute budget allocations for PB instances embedded into JA.

Corollary 2. *When the integrity constraint is represented as a DNNF circuit, then the problem $\textsc{Outcome}(F)$ can be solved in polynomial time when F is either the Kemeny, the Slater, or the leximax rule.*

3.2 DNNF Circuit Embeddings

We now move on to the description of embeddings returning integrity constraints represented as DNNF circuits. In doing so, we follow De Haan [24] but use a slight generalisation of his approach, allowing us to deal with PB instances with multiple resources. The basic idea is that every $\vee$-node in the DNNF circuit will represent the choice of selecting or not a given project. At each of these nodes, we need to keep track of the amount of resources already used to determine whether a project can be selected without exceeding the budget.

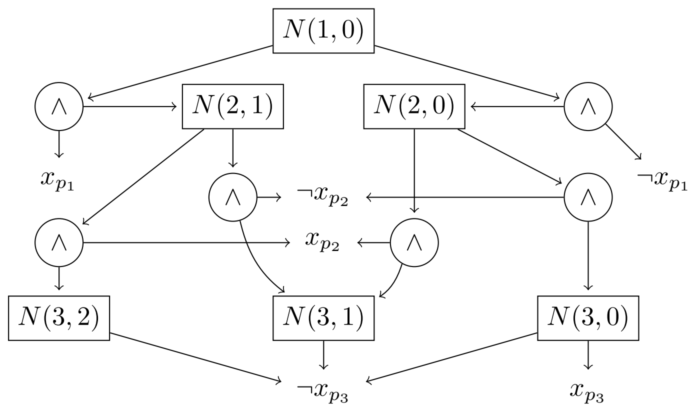
For a project index j and a vector of used quantities per resources $\boldsymbol{v} \in \mathbb{R}^d_{\geq 0}$, we introduce the $\vee$-node $N(j, \boldsymbol{v})$, corresponding to the situation where we previously made a choice on projects with indices 1 to $j-1$, and where for these choices we used resources according to $\boldsymbol{v}$. These nodes $N(j, \boldsymbol{v})$ are defined as follows.

$$\begin{cases} \top & \text{if } j = m+1 \\ \vee \begin{array}{l} \big(x_{p_j} \wedge N(j+1, \boldsymbol{v} + c(p_j))\big) \\ \big(\neg x_{p_j} \wedge N(j+1, \boldsymbol{v})\big) \end{array} & \text{if } \boldsymbol{v} + c(p_j) \leq \boldsymbol{b} \\ \big(\neg x_{p_j} \wedge N(j+1, \boldsymbol{v})\big) \vee (x_{p_j} \wedge \bot) & \text{otherwise} \end{cases}$$

For a PB instance $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c\rangle$, the *tractable embedding* $TE(I)$ returns the integrity constraint defined by $N(1, \mathbf{0}_d)$, where $\mathbf{0}_d$ denotes the vector of length d whose components are all equal to 0.

Let us illustrate this embedding on a simple example.

Example 1. Consider an instance I with just one resource r and projects p_1, p_2, and p_3. The cost of the projects in r is $c(p_1) = c(p_2) = 1$ and $c(p_3) = 2$ and the budget limit is $b = 2$. Call x_{p_1}, x_{p_2}, and x_{p_3} the propositional atoms corresponding to p_1, p_2, and p_3, respectively. *TE* on I would construct the following DNNF circuit (which we simplified a bit):



△

We can show that the tractable embedding does encode PB instances correctly.

Proposition 3. *The tractable embedding TE is correct, and for any given PB instance $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c\rangle$ returns an integrity constraint $TE(I)$ represented as a DNNF circuit of size in $\mathcal{O}(m \times |\{c(A) \mid A \subseteq \mathcal{A}(I)\}|)$.*

At this point, it should be noted that the exponential factor in the size of the embedding, namely $|\{c(A) \mid A \subseteq \mathcal{A}(I)\}|$, is bounded from above by the sum of the budget limits for each resource. Hence, the corresponding DNNF circuit is of size in $\mathcal{O}(m \times \sum_{r\in\mathcal{R}} b_r)$, making it pseudo-polynomial in the size of the PB instance.

In the remainder of this section we investigate to what extent this approach allows us to introduce additional distributional constraints for PB.

3.3 Dependencies between Projects

We now consider the situation where some projects can only be achieved if some others are also achieved.

Take a PB instance $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c\rangle$. We introduce a set of implications, $Imp \subseteq \boldsymbol{\mathcal{L}_P}$, linking projects together. A set of implications is a set of propositional formulas of the form $\ell_{x_p} \to \ell_{x_{p'}}$ for p and p' two projects in $\mathcal{P}$ with ℓ_{x_p} and $\ell_{x_{p'}}$ being the corresponding literals. Note that this corresponds to 2-CNF formulas. Such an implication indicates that if ℓ_{x_p} is positive (resp. negative), p can be selected (resp. not selected) only if p' is selected (resp. not selected) if $\ell_{x_{p'}}$ is positive (resp. negative). A budget allocation A satisfies the set of implications *Imp* if and only if the previously described semantics is satisfied.

First of all, we can show in the extended setting, finding a feasible budget allocation is a NP-complete problem by a reduction from 2-CNF Minimal Model [4].

Proposition 4. *Let $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c\rangle$ be a PB instance and Imp a set of implications over I. Deciding whether there exists a feasible budget allocation for I satisfying Imp is NP-complete, and NP-hardness holds even for a single resource.*

Based on this result, we cannot hope to find an embedding into a DNNF ciruit of polynomial size. However, we can still define an interesting parameterized embedding, in the spirit of parameterized complexity [11]. To that end we introduce the interconnection graph $G = \langle \mathcal{P}, E \rangle$ of a set of implications *Imp* where there is an edge $(p_i, p_j) \in E$ between p_i and p_j if and only if there exists an implication in *Imp* linking the two projects.

Theorem 5. *Let $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ be a participatory budgeting instance and Imp a set of implications over I. There exists a correct embedding from I and Imp to an integrity constraint expressed as a DNNF circuit Γ whose size is in $\mathcal{O}\left(m \times |\{c(A) \mid A \subseteq \mathcal{A}(I)\}| \times 2^k\right)$, where k is the pathwidth [5] of the interconnection graph of Imp.*

Let us briefly present the embedding for Theorem 5. We order the projects according to the order in which they are introduced in an optimal path decomposition of the interconnection graph. We introduce $\vee$-nodes $N(j, \boldsymbol{v}, L)$ where j is a project index, $\boldsymbol{v} \in \mathbb{R}^d_{\geq 0}$ a vector of used quantities per resource and $L \subseteq \mathit{Lit}(\boldsymbol{P})$ a subset of literals. Intuitively, the set L specifies the literals that we selected and that we should remember. If $j = m+1$, then $N(j, \boldsymbol{v}, L) = \top$. If the positive literal x_{p_j} is implied by some literal in L w.r.t. *Imp*, then $N(j, \boldsymbol{v}, L) = N(j+1, \boldsymbol{v} + c(p_j), L \cup \{x_{p_j}\})$. Similarly, if the negative literal $\neg x_{p_j}$ is implied by some literal in L w.r.t. *Imp*, then $N(j, \boldsymbol{v}, L) = N(j+1, \boldsymbol{v}, L \cup \{\neg x_{p_j}\})$. Otherwise, if $\boldsymbol{v} + c(p_j) \leq \boldsymbol{b}$, then $N(j, \boldsymbol{v}, L) = (x_{p_j} \wedge N[j+1, \boldsymbol{v} + c(p_j), L \cup \{x_{p_j}\}]) \vee (\neg x_{p_j} \wedge N[j+1, \boldsymbol{v}, L \cup \{\neg x_{p_j}\}])$, and otherwise, $N(j, \boldsymbol{v}, L) = (x_{p_j} \wedge \bot) \vee (\neg x_{p_j} \wedge N[j+1, \boldsymbol{v}, L \cup \{\neg x_{p_j}\}])$.

We conclude by giving an example the embedding described in the proof above.

Example 2. Consider the instance described in Example 1. Assume that, if project p_1 is selected, then also project p_2 should be selected. The optimal path decomposition of the interconnection graph is thus $(\{p_1, p_2\}, \{p_2\})$. So we will consider the projects in the following order: p_1, p_2, p_3. The embedding described above would return the following DNNF circuit (which again has been simplified a bit):

$$\begin{tikzcd}
 & & \boxed{N(1,0,\emptyset)} \arrow[dll] \arrow[drrr] & & & \\
\bigcirc\!\!\!\!\wedge \arrow[r] \arrow[d] & \boxed{N(2,1,\{x_{p_1}\})} \arrow[d] & & \boxed{N(2,0,\{\neg x_{p_1}\})} \arrow[d] \arrow[dr] & & \bigcirc\!\!\!\!\wedge \arrow[ll] \arrow[d] \\
x_{p_1} & \bigcirc\!\!\!\!\wedge \arrow[r] \arrow[d] & x_{p_2} & \bigcirc\!\!\!\!\wedge \arrow[l] \arrow[dl] & \bigcirc\!\!\!\!\wedge \arrow[dl] \arrow[dr] & \neg x_{p_1} \\
 & \boxed{N(3,2,\emptyset)} \arrow[dr] & \boxed{N(3,1,\emptyset)} \arrow[d] & \boxed{N(3,0,\emptyset)} \arrow[dl] \arrow[d] & & \neg x_{p_2} \\
 & & \neg x_{p_3} & x_{p_3} & &
\end{tikzcd}$$

$\triangle$

3.4 Quotas on Types of Projects

Another very natural constraint is to consider types and quotas over the projects. The idea is that the projects belong to various types (health, education, environment to name a few) and that some quotas over these types are to be respected by the final budget allocation (at least two health-related projects for instance). We model this idea by defining a type system.

Formally, for a given PB instance $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$, a type system is a tuple $\langle \mathcal{T}, \mathcal{Q}, q, f \rangle$ where $\mathcal{T} \in 2^{2^{\mathcal{P}}}$ is a set of types, each type being a subset of projects; $\mathcal{Q} = \langle Q, +, 0, \leq_Q \rangle$ is an ordered group over which the quotas are expressed; $q : \mathcal{T} \to Q^2$ is a quota function such that for any type $t \in \mathcal{T}$, $q(t) = (a, b) \in Q^2$ with $a \leq_Q b$ and $f : \mathcal{T} \times \mathcal{A}(I) \to Q$ is a type aggregator. For $t \in \mathcal{T}$ such that $q(t) = (a, b)$, we write $q(t)^- = a$ and $q(t)^+ = b$, which indicate the lower and upper quota for type t respectively. A budget allocation A is feasible if the quotas are respected—that is, if for every $t \in \mathcal{T}$, we have $q(t)^- \leq_Q f(t, A) \leq_Q q(t)^+$.

In the following, we provide two type aggregators that are very natural.

- **Cardinality-type aggregator**: the quotas express lower and upper bounds on the number of projects selected for each type. We have $Q = \mathbb{N}$, $\leq_Q$ is the usual order on $\mathbb{N}$, and the type aggregator is $f^{\text{card}}(t, A) = |A \cap t|$.
- **Cost-type aggregator**: the quotas define lower bound and upper bound on the total cost of the selected projects for each type. Here $Q = \mathbb{R}^d_{\geq 0}$, $\leq_Q$ is the component-wise order defined in the preliminaries, and the type aggregator is $f^{\text{cost}}(t, A) = \sum_{p \in A \cap t} c(p)$.

Using Set Splitting [22], we can easily show that deciding whether there is a feasible budget allocation with a given type system is NP-complete for both of the type aggregators.

Proposition 6. *Let $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ be a PB instance and $\langle \mathcal{T}, \mathcal{Q}, q, f \rangle$ a type system over I. Deciding whether there exists a feasible budget allocation A is NP-complete when f is either the cardinality or the cost-type aggregator, and NP-hardness holds even for a single resource.*

We can still define a parameterized embedding for PB with types and quotas. It works for any *additive* type aggregator $f : \mathcal{T} \times \mathcal{A}(I) \to Q$, that is, any type aggregator f for which there exists a *score type function* s that takes as input a project $p \in \mathcal{P}$ and returns an element in Q such that for every type $t \in \mathcal{T}$ and every allocation $A \in \mathcal{A}(I)$, $f(t, A) = \sum_{p \in A} s(p)$. The two type aggregators described above are both additive, with $s^{\text{card}}(p) = 1$ and $s^{\text{cost}}(p) = c(p)$.

Let I be an instance and $\langle \mathcal{T}, \mathcal{Q}, q, f \rangle$ a type system over I, the overlap graph of the type system is the graph $G = \langle \mathcal{T}, E \rangle$, where there is an edge $\{t, t'\}$ in E if and only $t \cap t' \neq \emptyset$.

Theorem 7. *Let $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ be a PB instance and $\langle \mathcal{T}, \mathcal{Q}, q, f \rangle$ a type system where f is an additive type aggregator defined w.r.t. the score type function s. There exists a correct embedding for I and $\langle \mathcal{T}, \mathcal{Q}, q, f \rangle$ that returns an integrity constraint represented as a DNNF circuit whose size is in $\mathcal{O}\left(m \times |\{c(A) \mid A \subseteq \mathcal{A}(I)\}| \times k^*\right)$ where $k^* = \max_{t \in \mathcal{T}}(|\{f(t, A) \mid A \in \mathcal{A}(I)\}|)^{k+1}$ and where k is the pathwidth of the overlap graph of $\langle \mathcal{T}, \mathcal{Q}, q, f \rangle$.*

Note that for the cardinality and the cost type aggregators, $\max_{t \in \mathcal{T}}(|\{f(t, A) \mid A \in \mathcal{A}(I)\}|)^{k+1}$ is upper-bounded by $|\mathcal{P}|$ and $\prod_{r \in \mathcal{R}} \boldsymbol{b}_r$ respectively.

Interestingly, the embedding is efficient in the natural case of non-overlapping types (the overlap graph is empty). Indeed, in that case the pathwidth of the overlap graph is 0.

Corollary 8. *For an additive type aggregator and non-overlapping types, the size of the DNNF circuit returned by the previous embedding is in $\mathcal{O}\left(m \times |\{c(A) \mid A \subseteq \mathcal{A}(I)\}|\right)$.*

4 Enforcing Exhaustiveness

Amongst the very basic requirements of a budget allocation is that of *exhaustiveness* [2], or *inclusion maximality* [39]. It requires that the budget be used as much as possible.

Definition 5 (Exhaustiveness)**.** *Given a PB problem instance $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$, a budget allocation $A \in \mathcal{A}(I)$ is said to be exhaustive if, for every project $p \in \mathcal{P} \setminus A$, there exists at least one resource $r \in \mathcal{R}$ such that $c(A \cup \{p\}, r) > \boldsymbol{b}_r$.*

For a given instance I, we denote by $\mathcal{A}_{EX}(I)$ the set of feasible and exhaustive budget allocations. An embedding $E : \mathcal{I} \to \mathcal{L}_{\boldsymbol{P}}$ is said to be exhaustive if for every $I \in \mathcal{I}$, we have $\tau(\mathfrak{J}(E(I))) \subseteq \mathcal{A}_{EX}(I)$. A JA rule F is said to be exhaustive if for every correct embedding E, every instance $I \in \mathcal{I}$ and every profile $\boldsymbol{A}$ it is the case that $\tau(F(E(I), \boldsymbol{A})) \subseteq \mathcal{A}_{EX}(I)$.

Similarly, a PB rule is said to be exhaustive if it only returns exhaustive budget allocations. Finally, an exhaustive embedding E is correct if $\mathcal{A}_{EX}(I) = \tau(\mathfrak{J}(E(I)))$, for every instance I.

Because the scenarios typically modelled using JA are rather different from PB, the exhaustiveness axiom is not satisfied by the main JA rules. This has to do with the semantics of rejection (of a proposition) in the context of JA.

Proposition 9. *No majority-consistent JA rule is exhaustive.*

Proof. Consider a correct but not exhaustive embedding E (for instance TE). As E is not exhaustive, there exists a PB instance I such that there is at least one admissible JA outcome $J \in \mathfrak{J}(E(I))$ with $\tau(J) \notin \mathcal{A}_{EX}(I)$. Now consider a profile $\boldsymbol{A}$ with n agents in which $\lceil n/2 \rceil + 1$ agents only approve of the projects in $\tau(J)$; the other agents are not constrained. On the JA side, the majoritarian outcome will be J. Since the majoritarian outcome is admissible, any majority-consistent rule F must return $\{J\}$ on $E(I)$ and $\boldsymbol{A}$, which is not exhaustive. □

This result is far-reaching as most JA rules have been designed to be majority-consistent.

A first approach to circumvent this problem—that we only sketch here—could be to define exhaustive embeddings in which exhaustiveness is logically encoded in the integrity constraint. The tractable embedding could be extended that way by remembering the cost of the smallest non-selected project to check for every leaf whether the budget allocation is exhaustive or not. However, this approach can only be efficient for instances with a single resource. In the multi-resources case, there could be exponentially many "cheapest projects".

We will now discuss another approach: introducing new JA rules which are exhaustive. In the context of PB, when an agent does not include a project in her approval ballot, this does not imply that she does not want to see the project being funded, but rather that it is not one of her top projects. Therefore, to implement PB via JA we need to adapt the JA rules so that not selecting a project (i.e., not accepting a proposition) is not interpreted as a rejection. To this end we introduce the new family of *asymmetric* JA rules. They avoid the symmetric treatment of acceptance and rejection common in most standard JA rules.

Definition 6 (Asymmetric Additive Rules)**.** *Let F be an additive JA rule associated with $f : (2^{\boldsymbol{P}})^n \times Lit(\boldsymbol{P}) \to \mathbb{R}_{\geq 0}$. Then its asymmetric counterpart F_{asy} is the rule for which, for every integrity constraint Γ and every profile $\boldsymbol{J}$, we have:*

$$F_{asy}(\Gamma, \boldsymbol{J}) = \underset{J \in \mathfrak{J}(\Gamma)}{\operatorname{argmax}} \sum_{\substack{\ell \in ext(J) \\ \ell \text{ is positive}}} f(\boldsymbol{J}, \ell) + \epsilon,$$

where $\epsilon \leq \frac{1}{|\boldsymbol{P}|} \times \min\{f(\boldsymbol{J}, \ell) \neq 0 \mid \boldsymbol{J} \in (2^{\boldsymbol{P}})^n, \ell \in ext(J), \ell \text{ is positive}\}$.

Importantly, this definition applies only if f is $\mathbb{R}_{\geq 0}$-valued. The use of ϵ guarantees that accepting positive literals will always be more appealing than accepting negative ones, while being small enough so as to not impact the relative values of positive literals. Note that $\epsilon = \frac{1}{|\boldsymbol{P}|+1}$ is a suitable choice for the three rules defined near the end of Section 2.2.

Proposition 10. *Let F be an additive JA rule associated with an $\mathbb{R}_{\geq 0}$-valued function f. Then the asymmetric counterpart of F satisfies exhaustiveness.*

Proof. Executing F_{asy} involves computing a score for every admissible candidate outcome J. By definition, no negative literal in J can contribute to its score, while every positive literal makes a strictly positive contribution of at least ϵ. Thus, flipping a negative literal increases the score. So F_{asy} only returns admissible judgments for which flipping any negative literal would violate the integrity constraint. This corresponds to exhaustiveness. □

Observe that the asymmetric counterpart of any additive rule is additive itself (and similarly for scoring rules, albeit not for AMRs). Finally, it is interesting to note that the asymmetric variant of the leximax rule is very similar to the well-known *greedy approval rule* for PB [1].

	Kemeny rule		Slater rule		Leximax rule	
	usual	asymmetric	usual	asymmetric	usual	asymmetric
Exhaustiveness	✗	✓	✗	✓	✗	✓
Limit Monotonicity	✗	✗	✗	✗	✗	✗
Discount Monotonicity	✓	✓	✓	✓	✓	✓
Splitting Monotonicity	✗	✓	✗	✓	✗	✓
Merging Monotonicity	✗	✗	✗	✗	✗	✗

Table 1: Summary of the axiomatic results.

5 Axiomatic Analysis

Axioms, such as exhaustiveness, are means for encoding formal properties related to the normative adequacy of mechanisms for collective decision-making [40]. In this section we investigate to what extent axioms proposed for PB are satisfied by JA rules. We consider exhaustiveness to be a basic requirement in all PB processes, so in the following we will only focus on settings where it is enforced—either via an exhaustive embedding or by the use of asymmetric rules. The results of this section are summarised in Table 1.

The literature on axioms for PB is still sparse. We focus on the monotonicity axioms introduced by Talmon and Faliszewski [39], generalising their definitions to allow for multiple resources and irresolute rules. Formally, for a given PB axiom $\mathfrak{X}$, we say that the JA rule F *satisfies* $\mathfrak{X}$ *w.r.t. embedding* E if, for every PB instance I, the PB rule mapping $\boldsymbol{A}$ to $\tau(F(E(I), \boldsymbol{A}))$ for any given profile $\boldsymbol{A}$ satisfies $\mathfrak{X}$.

Moreover, for a resolute rule F, these axioms are usually stated as "when one moves from $(I, \boldsymbol{A})$ to another pair $(I', \boldsymbol{A}')$, then if $F(I, \boldsymbol{A})$ satisfies a certain property, $F(I', \boldsymbol{A}')$ should satisfy a corresponding property." We generalise these axioms to the irresolute case by requiring that, if *every* budget allocation returned by our rule for $(I, \boldsymbol{A})$ satisfies the property in question, then *every* budget allocation for $(I', \boldsymbol{A}')$ should satisfy the corresponding property. Note that "existential" generalisation have also been studied [3].

The first axiom is called *limit monotonicity.* It states that after any increase in the budget limit that is not so substantial as to make some previously unaffordable project affordable, any funded project should continue to get funded. This axiom is closely related to that of committee monotonicity for multiwinner voting rules [12]. It is easy to see that none of the JA rules of interest will satisfy it, even when $|\mathcal{R}| = 1$.

Definition 7 (Limit monotonicity)**.** *A PB rule F is said to be limit-monotonic if, for any two PB instances $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c\rangle$ and $I' = \langle \mathcal{R}, \boldsymbol{b}', \mathcal{P}, c\rangle$ with $\boldsymbol{b} \leq \boldsymbol{b}'$ and $c(p) \leq \boldsymbol{b}$ for all projects $p \in \mathcal{P}$, it is the case that $\bigcap F(I, \boldsymbol{A}) \subseteq \bigcap F(I', \boldsymbol{A})$ for all profiles $\boldsymbol{A}$.*

We move on to *discount monotonicity*, an axiom stating that, if the cost of a selected project is reduced, then that project should continue to be selected.

Definition 8 (Discount monotonicity)**.** *A PB rule F is said to be discount-monotonic if, for any two PB instances $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c\rangle$ and $I' = \langle \mathcal{R}, \boldsymbol{b}', \mathcal{P}, c\rangle$ with $c(p) \geq c'(p)$ and $c(p') = c'(p')$ for all $p' \in \mathcal{P} \setminus \{p\}$ for some distinguished project $p \in \mathcal{P}$, it is the case that $p \in \bigcap F(I, \boldsymbol{A})$ implies $p \in \bigcap F(I', \boldsymbol{A})$ for all profiles $\boldsymbol{A}$.*

To study how JA rules deal with discount monotonicity, we introduce a new axiom for JA. This axiom is relevant for us, since it is a sufficient condition for discount monotonicity.

Definition 9 (Constraint monotonicity)**.** *A JA rule F is said to be constraint-monotonic if, for any two integrity constraints $\Gamma, \Gamma' \in \mathcal{L}_{\boldsymbol{P}}$ with $\mathfrak{J}(\Gamma) \subseteq \mathfrak{J}(\Gamma')$ and any profile $\boldsymbol{J}$, it is the case that $F(\Gamma', \boldsymbol{J}) \setminus F(\Gamma, \boldsymbol{J}) \subseteq \mathfrak{J}(\Gamma') \setminus \mathfrak{J}(\Gamma)$.*

Lemma 11. *Every constraint-monotonic JA rule is discount-monotonic with respect to any correct embedding.*

Our axiom turns out to be satisfied by many JA rules.

Proposition 12. *Every additive JA rule is constraint-monotonic.*

Proof. Consider any additive rule F. Suppose, that F is *not* constraint-monotonic. Then there exist two integrity constraints Γ and Γ' with $\mathfrak{J}(\Gamma) \subseteq \mathfrak{J}(\Gamma')$ and a profile $\boldsymbol{J}$ for which there exists a $J \in F(\Gamma', \boldsymbol{J}) \setminus F(\Gamma, \boldsymbol{J})$ with $J \in \mathfrak{J}(\Gamma) \setminus \mathfrak{J}(\Gamma')$. As $J \notin F(\Gamma, \boldsymbol{J})$, there exists some $J' \in \mathfrak{J}(\Gamma)$ with a higher total score than that of J. Moreover, since $\mathfrak{J}(\Gamma) \subseteq \mathfrak{J}(\Gamma')$, this same J' would outperform J also under Γ'. This implies that $J \notin F(\Gamma', \boldsymbol{J})$, which is a contradiction, so we are done. $\square$

Corollary 13. *The Kemeny, Slater, and leximax rules as well as their asymmetric counterparts are all discount-monotonic w.r.t. any correct embedding.*

The last two axioms we consider deal with situations where projects are split into subprojects (and the dual operation of merging projects). First, *splitting monotonicity* states that, if a selected project is split into a set of projects approved by the same agents, then some of these new projects should still be selected. The axiom of *merging monotonicity* expresses a similar condition when merging projects.

Given a PB instance $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ and a profile $\boldsymbol{A}$, we say that $I' = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}', c' \rangle$ and $\boldsymbol{A}'$ are *the result of splitting project $p \in \mathcal{P}$ into P* (with $P \cap \mathcal{P} = \emptyset$), if $\mathcal{P}' = (\mathcal{P} \setminus \{p\}) \cup P$, for all $p' \in P$, $c(p) \neq \mathbf{0}_d$, $c'(P) = c(p)$, $c'(p') = c(p')$ for all $p' \in \mathcal{P}' \setminus P$, $A'_i = A_i$ for all $i \in \mathcal{N}$ with $p \notin A_i$, and $A'_i = (A_i \setminus \{p\}) \cup P$ for all other $i \in \mathcal{N}$. We also say that I and $\boldsymbol{A}$ are *the result of merging P into p* given I' and $\boldsymbol{A}'$.

Definition 10 (Splitting monotonicity)**.** *A PB rule F is said to be splitting-monotonic if, for any two PB instances $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ and $I' = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}', c' \rangle$ with corresponding profiles $\boldsymbol{A}$ and $\boldsymbol{A}'$ such that I' and $\boldsymbol{A}'$ are the result of splitting project p into P given I and $\boldsymbol{A}$, it is the case that if $p \in \bigcap F(I', \boldsymbol{A})$ then $A' \cap P \neq \emptyset$ for all $A' \in F(I', \boldsymbol{A})$.*

Definition 11 (Merging monotonicity)**.** *A PB rule F is said to be merging-monotonic if, for any two PB instances $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ and $I' = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}', c' \rangle$ with corresponding profiles $\boldsymbol{A}$ and $\boldsymbol{A}'$ such that I' and $\boldsymbol{A}'$ are the result of merging project set P into project p given I and $\boldsymbol{A}$, it is the case that $P \subseteq \bigcap F(I, \boldsymbol{A})$ implies $p \in \bigcap F(I', \boldsymbol{A})$.*

We first show that splitting monotonicity is satisfied by the asymmetric counterpart of any AMR.

Proposition 14. *Every asymmetric counterpart of an AMR is splitting-monotonic.*

Proof. Let F be the asymmetric counterpart of an AMR and let E be a correct exhaustive embedding. Consider a PB instance $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ and a profile $\boldsymbol{A}$. Let $I' = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}', c' \rangle$ and $\boldsymbol{A}'$ be the instance and profile resulting from splitting $p \in \bigcap \tau(F(E(I), \boldsymbol{A}))$ into P.

Consider any outcome $J_1 \in F(E(I), \boldsymbol{A})$. Note that for all $J \in \mathfrak{J}(E(I)) \cap \mathfrak{J}(E(I'))$, we have $p \notin \tau(J)$ and $\tau(J) \cap P = \emptyset$. Because $p \in \bigcap \tau(F(E(I), \boldsymbol{A}))$, this implies that J_1 has a higher total score than any $J_2 \in \mathfrak{J}(E(I)) \cap \mathfrak{J}(E(I'))$.

Consider now a possible outcome $J'_1 = (J_1 \setminus \{p\}) \cup \{p'\}$ for some $p' \in P$. Based on the definition of $\boldsymbol{A}'$, it is clear that $n_x^{\boldsymbol{A}} = n_x^{\boldsymbol{A}'}$ for every $x \in J_1 \setminus \{x_p\}$ and that $n_{x_p}^{\boldsymbol{A}} = n_{x_{p'}}^{\boldsymbol{A}'}$. Hence, because F is the asymmetric counterpart of an AMR, J_1 and J_2 have the same total score. Indeed, the total score for an AMR only depends on the approval score of each literal and because F is asymmetric we only consider positive literals. This implies that J'_1 has a higher total score than any $J_2 \in \mathfrak{J}(E(I)) \cap \mathfrak{J}(E(I'))$. Thus, $\mathfrak{J}(E(I)) \cap F(E(I'), \boldsymbol{A}') = \emptyset$. As for every $J' \in \mathfrak{J}(E(I')) \setminus \mathfrak{J}(E(I))$ we have $P \cap \tau(J') \neq \emptyset$, every outcome returned by F would have a nonempty intersection with P. $\square$

While this last result is promising, it unfortunately does not extend to symmetric rules.

Example 3. Consider the following pairs of instances and three-agent profiles: I and $\boldsymbol{A}$ on the left and I' and $\boldsymbol{A}'$ on the right. Both involve just one resource, with $b_1 = 4$.

Project	p_1	p_2	p_3	p_4	p_1	$\{p_2^1, p_2^2, p_2^3, p_2^4\}$	p_3	p_4
Cost	2	2	1	1	2	0.5	1	1
Agents 1 and 2	✓	✗	✗	✗	✓	✗	✗	✗
Agent 3	✗	✓	✓	✓	✗	✓	✓	✓

Note that I' and $\boldsymbol{A}'$ are the result of splitting p_2 into $\{p_2^1, p_2^2, p_2^3, p_2^4\}$, given I and $\boldsymbol{A}$. One can check that the Kemeny, the Slater, and the leximax rules would all return $\{\{p_1, p_2\}\}$ for $(I, \boldsymbol{A})$ when used with a correct exhaustive embedding. However, they would return $\{\{p_1, p_3, p_4\}\}$ for $(I', \boldsymbol{A}')$, hence, violating splitting monotonicity. △

We finally investigate merging monotonicity. It turns out that none of the rules we are considering in this paper satisfy it. A simple counterexample is described in the following. Consider an instance with one resource, a budget of 4, and six projects: four of cost 1 and two of cost 2. Consider now a profile with a single agent approving of every project. Then all of our rules (Kemeny, Slater, and leximax) would select the four projects of cost 1. Now, if the four projects of cost 1 are merged into one project of cost 4, all our rules would select the two projects of cost 2. This is a violation of merging monotonicity as the newly created project is not selected. By construction, the same also holds for the asymmetric counterparts.

To conclude, we shortly discuss the overall axiomatic picture for JA rules. The most striking results are that none of our rules satisfy limit and merging monotonicity. For limit monotonicity, it should be noted that no PB rule we know of satisfies it [39]. It seems to be too strong a requirement. For merging monotonicity, the situation is less clear-cut: Some PB rules satisfy it but none that are widely used. Other axiomatic results are in line with those of Talmon and Faliszewski [39]. Overall, JA rules perform similarly to other PB rules in normative terms.

6 Conclusion

We have proposed an efficient way of solving PB problems by means of JA rules. The richness of the JA framework allowed us to develop embeddings for generalised forms of PB. While the resulting problems are computationally hard in general, we nevertheless were able to present useful parameterized embeddings for them. Regarding the axiomatic properties, we observed that a naïve way of embedding PB into JA leads to rules that violate the crucial exhaustiveness requirement of PB. We suggested to use asymmetric JA rules to enforce it. We also analysed some common JA rules and their asymmetric counterparts in view of monotonicity axioms for PB.

In terms of future work, it would be interesting to study more PB axioms, to allow us to better differentiate between different JA rules. An exciting direction is to investigate proportionality axioms such as those introduced by Aziz et al. [2] and Haret et al. [25]. Whether the model of Jain et al. [26] fits in our framework is another one. Finally, it seems worth exploring whether some structural assumptions on the PB side could be translated into meaningful properties on the JA side.

Beyond its immediate significance to the theory and practice of PB, we believe our work also highlights some important aspects of working with different frameworks for collective decision making. The high expressive power of JA permits us to encode many problems of practical interest as well. Finding efficient ways of solving decision problems embedded into JA can be hard, but once identified, these methods allow for great flexibility.

References

[1] Haris Aziz and Nisarg Shah. Participatory budgeting: Models and approaches. In *Pathways between Social Science and Computational Social Science: Theories, Methods and Interpretations*. Springer, 2020.

[2] Haris Aziz, Barton E. Lee, and Nimrod Talmon. Proportionally representative participatory budgeting: Axioms and algorithms. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 23–31, 2018.

[3] Dorothea Baumeister, Linus Boes, and Tessa Seeger. Irresolute approval-based budgeting. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1774–1776, 2020.

[4] Rachel Ben-Eliyahu and Rina Dechter. On computing minimal models. *Annals of Mathematics and Artificial Intelligence*, 18(1):3–27, 1996.

[5] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1-2):1–45, 1998.

[6] Yves Cabannes. Participatory budgeting: A significant contribution to participatory democracy. *Environment and Urbanization*, 16(1):27–46, 2004.

[7] Vincent Conitzer, Rupert Freeman, and Nisarg Shah. Fair public decision making. In *Proceedings of the 18th ACM Conference on Economics and Computation (EC)*, pages 629–646, 2017.

[8] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[9] Franz Dietrich. Scoring rules for judgment aggregation. *Social Choice and Welfare*, 42 (4):873–911, 2014.

[10] Franz Dietrich and Christian List. Arrow's Theorem in judgment aggregation. *Social Choice and Welfare*, 29(1):19–33, 2007.

[11] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.

[12] Edith Elkind, Piotr Faliszewski, Piotr Skowron, and Arkadii Slinko. Properties of multiwinner voting rules. *Social Choice and Welfare*, 48(3):599–632, 2017.

[13] Ulle Endriss. Judgment aggregation. In Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia, editors, *Handbook of Computational Social Choice*, chapter 17. Cambridge University Press, 2016.

[14] Ulle Endriss. Judgment aggregation with rationality and feasibility constraints. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 946–954, 2018.

[15] Ulle Endriss and Ronald De Haan. Complexity of the winner determination problem in judgment aggregation: Kemeny, Slater, Tideman, Young. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2015.

[16] Ulle Endriss, Umberto Grandi, and Daniele Porello. Complexity of judgment aggregation. *Journal of Artificial Intelligence Research*, 45:481–514, 2012.

[17] Ulle Endriss, Umberto Grandi, Ronald De Haan, and Jérôme Lang. Succinctness of languages for judgment aggregation. In *Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2016.

[18] Patricia Everaere, Sébastien Konieczny, and Pierre Marquis. Counting votes for aggregating judgments. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1177–1184, 2014.

[19] B. Fain, A. Goel, and K. Munagala. The core of the participatory budgeting problem. In *Proceedings of the 12th International Workshop on Internet and Network Economics (WINE)*, pages 384–399, 2016.

[20] Brandon Fain, Kamesh Munagala, and Nisarg Shah. Fair allocation of indivisible public goods. In *Proceedings of the 19th ACM Conference on Economics and Computation (EC)*, pages 575–592, 2018.

[21] Rupert Freeman, David M. Pennock, Dominik Peters, and Jennifer Wortman Vaughan. Truthful aggregation of budget proposals. In *Proceedings of the 20th ACM Conference on Economics and Computation (EC)*, pages 751–752, 2019.

[22] Michael R. Garey and David S. Johnson. *Computers and Intractability*, volume 29. W.H. Freeman, 1979.

[23] Umberto Grandi and Ulle Endriss. Binary aggregation with integrity constraints. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 204–209, 2011.

[24] Ronald De Haan. Hunting for tractable languages for judgment aggregation. In *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 194–203, 2018.

[25] Adrian Haret, Martin Lackner, Andreas Pfandler, and Johannes P. Wallner. Proportional belief merging. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, pages 2822–2829, 2020.

[26] Pallavi Jain, Krzysztof Sornat, and Nimrod Talmon. Participatory budgeting with project interactions. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[27] Pallavi Jain, Krzysztof Sornat, Nimrod Talmon, and Meirav Zehavi. Participatory budgeting with project groups. *arXiv preprint arXiv:2012.05213*, 2020.

[28] Jérôme Lang, Gabriella Pigozzi, Marija Slavkovik, and Leendert Van der Torre. Judgment aggregation rules based on minimization. In *Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, pages 238–246, 2011.

[29] Jérôme Lang and Marija Slavkovik. Judgment aggregation rules and voting rules. In *Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT)*, pages 230–243, 2013.

[30] Jérôme Lang and Marija Slavkovik. How hard is it to compute majority-preserving judgment aggregation rules? In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI)*, pages 501–506, 2014.

[31] Christian List and Philip Pettit. Aggregating sets of judgments: An impossibility result. *Economics & Philosophy*, 18(1):89–110, 2002.

[32] Tyler Lu and Craig Boutilier. Budgeted social choice: From consensus to personalized decision making. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 280–286, 2011.

[33] Michael K. Miller and Daniel Osherson. Methods for distance-based judgment aggregation. *Social Choice and Welfare*, 32(4):575–601, 2009.

[34] Klaus Nehring and Marcus Pivato. Majority rule in the absence of a majority. *Journal of Economic Theory*, pages 213–257, 2019.

[35] Deval Patel, Arindam Khan, and Anand Louis. Group fairness for knapsack problems. In *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2021.

[36] Gabriella Pigozzi. Belief merging and the discursive dilemma: An argument-based account to paradoxes of judgment aggregation. *Synthese*, 152(2):285–298, 2006.

[37] Simon Rey, Ulle Endriss, and Ronald de Haan. Designing participatory budgeting mechanisms grounded in judgment aggregation. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2020.

[38] Anwar Shah, editor. *Participatory budgeting.* Public Sector Governance and Accountability Series. The World Bank, Washington, DC, 2007.

[39] Nimrod Talmon and Piotr Faliszewski. A framework for approval-based budgeting methods. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*, pages 2181–2188, 2019.

[40] William Thomson. On the axiomatic method and its recent applications to game theory and resource allocation. *Social Choice and Welfare*, 18(2):327–386, 2001.

Simon Rey
ILLC, University of Amsterdam
Amsterdam, The Netherlands
Email: `s.j.rey@uva.nl`

Ulle Endriss
ILLC, University of Amsterdam
Amsterdam, The Netherlands
Email: `u.endriss@uva.nl`

Ronald de Haan
ILLC, University of Amsterdam
Amsterdam, The Netherlands
Email: `r.dehaan@uva.nl`