A General Framework for Participatory Budgeting with Additional Constraints*

Simon Rey, Ulle Endriss and Ronald de Haan

Institute for Logic, Language and Computation (ILLC),
University of Amsterdam, Amsterdam, The Netherlands.

Contributing authors: s.j.rey@uva.nl; u.endriss@uva.nl;
r.dehaan@uva.nl;

Abstract

We introduce a new approach for designing rules for participatory budgeting, 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.

Keywords: Participatory Budgeting, Judgment Aggregation, Voting

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 (Cabannes, 2004). Since its first use for municipal budget allocation in Brazil over three decades ago, PB has been adopted across the world (Dias, 2018; Wampler et al., 2021). The typical PB process is divided

*This is an extended version of a paper originally presented at the 17th International Conference on Principles of Knowledge Representation and Reasoning (Rey et al., 2020), and also at the 8th International Workshop on Computational Social Choice.

into two stages (Shah, 2007; Wampler, 2012). In the first stage, citizens can submit project proposals they have for the city. Some of these proposals are then shortlisted and the citizens are asked to vote on which of the shortlisted projects they want to be implemented, given a budget constraint. While this describes the typical scenario of a PB process, each local implementation comes with its own specificities. In particular, it is quite common to encounter additional constraints on the range of projects that can be funded. In this paper, we develop a general framework that allows us to easily incorporate such additional constraints within the standard model for PB.

What types of additional constraints are used in real-life PB processes? By focusing on the case of the city of Paris (City of Paris, 2022) we can for instance learn that projects are sometimes grouped into categories and that special constraints apply to these categories. For instance, some projects are labelled "low income neighbuorhood" and a specific number of them has to be selected in the final bundle to be funded. Another category of projects are those designated for the whole city (and not a specific neighbourhood), and this category may also have a lower quota attached to it. Similar constraints can also be found in PB processes in Lisbon (Allegretti and Antunes, 2014) or in Amsterdam (City of Amsterdam, 2022). Other natural constraints that we can imagine include the existence of dependencies between projects, such as when some projects can only be realised if others are as well.

The challenge is then to incorporate such constraints into the formal study of PB. This is not a straightforward task. Indeed, most of the existing formal work in (computational) social choice regarding PB views voting on projects as a generalisation of multiwinner voting (see, e.g., Aziz et al., 2018; Talmon and Faliszewski, 2019; Peters et al., 2021). PB mechanisms that have been introduced in the literature are thus grounded in multiwinner voting, with an additional budget constraint. 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. In particular, it does not permit us to model the more expressive forms of PB we discussed earlier. Instead, by following this approach, one would need to redefine and and analyse from scratch everything for every specific variation of the base model motivated by real-life instances of PB. To address this problem, we develop in this paper a complementary approach, and study PB as a special case of judgment aggregation (JA), a highly expressive general-purpose aggregation framework (List and Puppe, 2009; Grossi and Pigozzi, 2014; Endriss, 2016).

JA is a framework in which opinions over binary issues are aggregated in a manner that ensures conformance to some feasibility constraint expressed in propositional logic. It is known to generalise many forms of preference aggregation and voting scenarios (Dietrich and List, 2007; Lang and Slavkovik, 2013; Endriss, 2018; Chingoma et al., 2022), including PB (De Haan, 2018). This high expressivity is particularly appealing for us as it allows us to easily encode additional constraints for PB. However, this comes at a computational cost that could be prohibitive if left unchecked. Indeed, the computational

complexity of computing outcomes of common JA rules is typically very high (Endriss et al., 2020), and hard even relative to complexity classes beyond NP. One of the main challenges we face in our endeavour thus is to find ways of implementing PB via JA in an efficient manner.

To cope with this algorithmic challenge, we explore the idea of looking for tractable fragments of JA. Of particular interest in this context is the prior work of De Haan (2018), who developed the idea of modelling JA problems using Boolean circuits in decomposable negation normal form (DNNF circuits) to save on the computational cost. We further develop this approach to express PB problems with additional constraints as DNNF circuits for JA, which can then be used in an efficient manner (from a computational perspective). More specifically, our contribution regarding the algorithmic part of this research agenda consists in two efficient embeddings of PB into JA for two kinds of additional constraints, namely dependencies between projects and quotas over categories 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 the rules we design are normatively adequate to be used in the context of PB. In the second part of the paper we therefore focus on this aspect and provide an axiomatic analysis of the rules we propose.

Among the requirements that have been introduced to study PB rules, the most fundamental one is probably exhaustiveness. It rules out any under-use of the budget by stating that it should not be possible for the leftover budget to be so large as to still allowing us to fund a further project. Interestingly, this requirement is incompatible with the usual view that JA rules take on logical negation. In JA, it is indeed standard to consider that when an agent says *no* to a given issue, selecting that issue would yield some dissatisfaction for the agent. When it comes to PB, the semantics of negation is usually 'weaker' in the following sense: An agent not approving of a given project would be neutral about whether the project will be selected. This motivates the introduction of asymmetrical counterparts of typical JA rules with this weaker take on negation. This approach allows us to define exhaustive rules within JA. We also provide an algorithmic solution to enforce exhaustiveness, but one that only works in limited cases (when the budget limit is uni-dimensional).

Incorporating exhaustiveness in JA is a mandatory step to make sure that the JA rules are potential candidates for PB rules, but it is not enough. To provide additional insight, we deepen our axiomatic analysis by studying how JA rules behave with respect to the monotonicity axioms for PB introduced by Talmon and Faliszewski (2019).

Thus, our contribution in this paper may be summarised as follows:

- We introduce a general framework for PB that makes it possible to easily incorporate additional constraints, and we exemplify this approach with two natural constraints on projects, namely dependencies and quotas.

- We demonstrate that, even though this more general framework is expressive enough to cover cases that require some extra computational cost, in many natural settings the overhead is manageable.
- We show that some of the typical JA rules are good candidates to be used in the context of PB, at least in view of monotonicity requirements.

Overall, our analysis demonstrates that this framework is suitable for the theoretical analysis of PB rules. Its main added value is the flexibility it offers to study of PB rules across different variants of the standard PB setting. Thanks to this framework, one can incorporate extra constraints for PB by "only" investigating the specific encoding of the constraint on the JA side, thereby immediately making available all other results previously established for the basic framework without those constraints. In particular, once a rule has been proven to satisfy a certain axiom, it will continue to do so, regardless of the extra constraints.[1] In this sense, this framework allows for an efficient study of a myriad of variations around the standard PB setting.

Related Work

According to the terminology of Aziz and Shah (2020), the basic model of PB we focus on in this paper is called *combinatorial PB with binary projects and approval ballots*. Several types of normative requirements have been studied for this framework. An important emphasis has been put on fairness and proportionality requirements (Fain et al., 2016; Aziz et al., 2018; Peters et al., 2021; Lackner et al., 2021; Los et al., 2022; Fairstein et al., 2022). Incentive compatibility is another important topic in the context of PB (Goel et al., 2019; Freeman et al., 2019; Rey et al., 2021). Other properties such as monotonicity requirements have also been studied (Talmon and Faliszewski, 2019). Much attention has been devoted to the algorithmic side of PB, too. Greedy and optimal welfare/fairness-maximising rules have been studied by Talmon and Faliszewski (2019), Fluschnik et al. (2019), Patel et al. (2021), and Sreedurga et al. (2022). Using a different approach, Fain et al. (2016) and Freeman et al. (2019) instead studied PB solutions as market equilibria, in the spirit of the public decision making setting of Conitzer et al. (2017).

While the review in the previous paragraph has focused on the standard model for PB, several extensions have also been explored. For instance, Lu and Boutilier (2011) considered an extension of PB where the cost of a project might depend on the number of agents choosing it; Benade et al. (2018) proposed different ballot formats for PB; Baumeister et al. (2020) focused on irresolute PB rules; Rey et al. (2021) investigated the interplay between the two stages we discussed in the introduction; Lackner et al. (2021) approached PB from a long-term perspective where several PB elections are held in

[1] To be precise, this is true for axioms that have a "universal flavour", i.e., for axioms that stipulate that certain conditions must be satisfied *for all* relevant situations (rather than requiring the *existence* of a situation exhibiting a specific property of interest). The monotonicity axioms of Talmon and Faliszewski (2019) are examples of such universal axioms.

sequence; Hershkowitz et al. (2021) looked into local versus global PB processes; Baumeister et al. (2022) studied PB settings with uncertainty on the cost and completion time of the projects. Particularly interesting for our concerns are the works of Jain et al. (2020) and Jain et al. (2021), who have investigated extensions of PB in which projects are grouped into categories, and where either the utility of each agent is a function of the number of selected projects from each category, or where there are specific budget limits for each category. Another exciting contribution is that of Fain et al. (2018) who followed a similar approach to ours. These authors considered a general setting of public decision making (introduced by Conitzer et al., 2017) and added different types of constraints to it (matroid, matching, and packing constraints), allowing for great flexibility on what can be modelled. Finally, Motamed et al. (2022) studied in more depth PB settings with multiple resources, an extension we also consider in this paper.

The study of how to embed preference aggregation problems into JA dates back to at least Dietrich and List (2007). The systematic study of how to embed voting rules into JA was then initiated by Lang and Slavkovik (2013) and later refined by Endriss (2018). Our work on the algorithmic aspects of such embeddings is based on the results of De Haan (2018), whose paper is also the first example for work investigating the embedding of PB into JA. Lately, a similar approach has been followed by Chingoma et al. (2022) to embed multiwinner voting 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. We draw our conclusions in Section 6.

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 (2020). 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 $\mathcal{P} = \{p_1, \ldots, p_m\}$. Let $\mathcal{R} = \{r_1, \ldots, r_d\}$ be a set of *resources* and $\boldsymbol{b} = (b_1, \ldots, b_d)$ 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 $c(p) = (c(p, r_1), \ldots, c(p, r_d))$ 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 $\mathcal{R}$, a budget limit vector $\boldsymbol{b}$, a set of projects $\mathcal{P}$, and a cost function c. We denote by $\mathcal{I}$ the set of all such instances.

A solution of a PB problem instance, called a *budget allocation*, is a subset of projects $\pi \subseteq \mathcal{P}$. A budget allocation π is said to be *feasible* if $c(\pi) \leq \boldsymbol{b}$. For a given $I \in \mathcal{I}$, the set of all feasible budget allocations is denoted by $\mathcal{A}(I)$. A budget allocation π is said to be *exhaustive* if there is no project $p \in \mathcal{P} \setminus \pi$ such that $c(\pi \cup \{p\}) \leq \boldsymbol{b}$. $\mathcal{A}_{EX}(I)$ is the set of all feasible and exhaustive budget allocation for an instance I.

Before deciding which budget allocation to recommend, we consult the *agents* belonging to a set $\mathcal{N} = \{1, \ldots, n\}$. Each agent $i \in \mathcal{N}$ submits an *approval ballot* $A_i \subseteq \mathcal{P}$, giving rise to a *profile* $\boldsymbol{A} = (A_1, \ldots, A_n)$. For any given project $p \in \mathcal{P}$, its *approval score* under profile $\boldsymbol{A}$ is defined as $\sum_{i\in\mathcal{N}} \mathbb{1}_{p\in A_i}$, the number of agents approving of p. Without loss of generality, 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.[2] Returning a set allows us to model possible ties in the outcome.

2.2 Judgment Aggregation

The specific JA framework we use is known as *binary aggregation with integrity constraints* (Grandi and Endriss, 2011).[3]

Let $\mathcal{L}_{\mathfrak{X}}$ be the set of propositional formulas over a given set $\mathfrak{X}$ 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 subset of atoms $X \subseteq \mathfrak{X}$, we write $\mathit{Lit}(X) = X \cup \{\neg x \mid x \in X\}$ 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 $\alpha : \mathfrak{X} \to \{0, 1\}$ is a mapping indicating for each atom its truth value. For $\ell_{x_i} \in \mathit{Lit}(\mathfrak{X})$, let $\alpha(\ell_{x_i}) = \alpha(x_i)$ if ℓ_{x_i} is positive and let $\alpha(\ell_{x_i}) = 1 - \alpha(x_i)$ otherwise. We write $\alpha \models \varphi$ whenever α is a model of φ according to the usual semantics of propositional logic (Van Dalen, 2013).

In the context of JA, the atoms in $\mathfrak{X}$ represent *propositions* an agent may either *accept* or *reject*. A *judgment* J is a set $J \subseteq \mathfrak{X}$, indicating which propositions are accepted. Let $\mathit{aug}(J) = J \cup \{\neg x \mid x \in \mathfrak{X} \setminus J\}$ be the judgment J augmented with the negative literals of the propositions not selected. 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

[2] Observe that $\mathcal{A}(I)$ is never empty. This is appropriate, given that the empty set of projects is always feasible. This, however, is not true for some of the extensions discussed in Section 3.

[3] While this framework is most convenient for our purposes, the original framework of List and Pettit (2002) could be used as well, given that it is known that the former can be efficiently embedded into the latter (Endriss et al., 2016).

explicitly specify the status of some of the 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* $\Gamma \in \mathcal{L}_{\mathfrak{X}}$ 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. Such a judgment J is then said to be *admissible*. Let $\mathfrak{J}(\Gamma) = \{J \subseteq \mathfrak{X} \mid J \models \Gamma\}$ be the set of all admissible judgments for any $\Gamma \in \mathcal{L}_{\mathfrak{X}}$. 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 $\ell \in \mathit{Lit}(\mathfrak{X})$, we write $n_\ell^{\boldsymbol{J}} = \sum_{i \in \mathcal{N}} \mathbb{1}_{\ell \in \mathit{aug}(J_i)}$ 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 (strict) majority of agents:

$$m(\boldsymbol{J}) = \{\ell \in \mathit{Lit}(\mathfrak{X}) \mid n_\ell^{\boldsymbol{J}} > n/2\}.$$

A *JA rule* is a function $F : \mathcal{L}_{\mathfrak{X}} \times (2^{\mathfrak{X}})^n \to 2^{2^{\mathfrak{X}}} \setminus \{\emptyset\}$ 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 of the agents $i \in \mathcal{N}$.

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 JA rule F is an additive rule if there exists a function $f : (2^{\mathfrak{X}})^n \times \mathit{Lit}(\mathfrak{X}) \to \mathbb{R}$ mapping pairs of profiles and literals to real values, such that, for every integrity constraint $\Gamma \in \mathcal{L}_{\mathfrak{X}}$ and every profile $\boldsymbol{J} \in (2^{\mathfrak{X}})^n$, we have:

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

This class generalises both the *scoring rules* of Dietrich (2014) and the *additive majority rules* (AMRs) defined by Nehring and Pivato (2019). More specifically, a scoring rule is associated with a scoring function $s : 2^{\mathfrak{X}} \times \mathit{Lit}(\mathfrak{X}) \to \mathbb{R}$ mapping judgments and literals to scores, and corresponds to the additive rule defined with respect to the function f such that:

$$f(\boldsymbol{J}, \ell) = \sum_{i \in \mathcal{N}} s(J_i, \ell).$$

An AMR is associated with a non-decreasing gain function $g : \{0, \ldots, n\} \to \mathbb{R}$ with $g(k) < g(k')$ for any $k < \frac{n}{2} \leq k'$ that maps the support of a literal to a

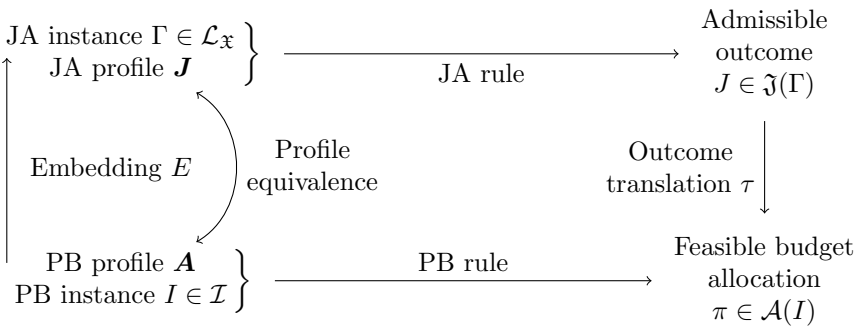

Fig. 1 Full process to use JA rules for PB instances.

score, and is an additive rule defined with respect to the function f such that:

$$f(\boldsymbol{J}, \ell) = g(n_\ell^{\boldsymbol{J}}).$$

Three additive rules are of particular importance for our purposes:

- The *Slater rule* (Miller and Osherson, 2009; Lang et al., 2011) 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 following gain function g:

$$g(x) = \begin{cases} 0 & \text{if } 0 \leq x < \frac{n}{2}, \\ 1 & \text{if } \frac{n}{2} \leq x \leq n. \end{cases}$$

- The *Kemeny rule* (Pigozzi, 2006; Miller and Osherson, 2009) selects the feasible outcome that is the closest to the profile as a whole. It is both an AMR with the gain function $g(x) = x$, and a scoring rule with the scoring function $s(J, \ell) = \mathbb{1}_{\ell \in aug(J)}$.
- The *leximax rule* (Everaere et al., 2014; Nehring and Pivato, 2019) favours the propositions supported by the largest majorities. It is the AMR defined by the gain function $g(x) = |\mathfrak{X}|^x$.

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 decide on outcomes for PB problems. To this end, we want to embed PB into JA and then use JA rules to compute budget allocations. A full schematic representation of the process is presented in Figure 1.

For a given PB instance, we introduce one proposition for each project to obtain $\mathfrak{X}$. So we have a direct correspondence between budget allocations

$\pi \subseteq \mathcal{P}$ and judgments $J \subseteq \mathfrak{X}$, 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 $\Gamma \in \mathcal{L}_{\mathfrak{X}}$ be an integrity constraint expressed over the atoms $\mathfrak{X} = \{x_p \mid p \in \mathcal{P}\}$. The outcome translation $\tau : 2^{\mathfrak{X}} \to 2^{\mathcal{P}}$ maps any judgment $J \in 2^{\mathfrak{X}}$ to a budget allocation $\pi = \tau(J)$ such that:

$$\tau(J) = \{p \in \mathcal{P} \mid x_p \in J\}.$$

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

We now define one of the fundamental elements of our approach: *embeddings*. An embedding is a function $E : \mathcal{I} \to \mathcal{L}_{\mathfrak{X}}$ that takes a PB instance as input and returns an integrity constraint (i.e., a JA instance). Given an embedding, we can 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 that states that the outcome translation τ defines a bijection between the set of budget allocations on the PB side and the set of admissible judgments on the JA side.

Definition 3 (Correct embedding) An embedding $E : \mathcal{I} \to \mathcal{L}_{\mathfrak{X}}$ is said to be correct if, for every PB instance $I \in \mathcal{I}$, we have:

$$\tau\left(\mathfrak{J}(E(I))\right) = \mathcal{A}(I).$$

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 in general (Endriss et al., 2020), if we nevertheless want to design PB rules that are tractable, 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 Language for Judgment Aggregation

As shown by De Haan (2018), computing outcomes under Kemeny and Slater can be done efficiently when the integrity constraint is a Boolean circuit in decomposable negation normal form (DNNF). We are going to extend this result to all additive rules. But let us first recall the definition of a DNNF circuit (Darwiche and Marquis, 2002).

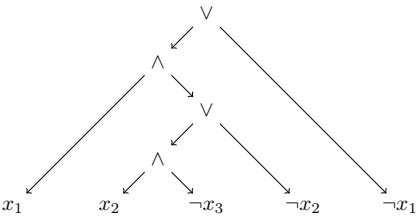

Fig. 2 Example of a decomposable negation normal form (DNNF) circuit.

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 $x \in \mathfrak{X}$ and whose internal nodes are labelled with $\wedge$ or $\vee$. A DNNF circuit C is an NNF circuit that is decomposable in the sense that, for every conjunction in C, no two conjuncts share a common propositional atom.

Figure 2 shows an example for a DNNF circuit. It can easily be checked that no two conjuncts share a common propositional atom.

For a given JA rule F, we define the outcome determination problem as the following decision problem:

Outcome(F)	
Input:	An integrity constraint Γ, a judgment profile $\boldsymbol{J}$, and a subset of literals $L \subseteq Lit(\mathfrak{X})$.
Question:	Is there a $J \in F(\Gamma, \boldsymbol{J})$ such that $L \subseteq aug(J)$?

We now show that for most additive JA rules F we can solve Outcome(F) efficiently when Γ is given as a DNNF circuit. The result does not apply to *all* additive rules, but only to those for which the associated function f does not require expensive computation. This is captured by the notion of *polynomial-time computable* functions, i.e., functions for which there exists an algorithm computing the outcome of the function and running in time that is polynomial in the size of the input.

Theorem 1 *Let F be an additive JA rule defined with respect to some polynomial-time computable function f. Then* Outcome(F) *is polynomial-time solvable when the integrity constraint Γ in the input is represented as a DNNF circuit.*

Proof We show that when Γ is a DNNF circuit, we can use the Algebraic Model Counting (AMC) problem to solve Outcome(F). Given a propositional formula $\varphi \in \mathcal{L}_{\mathfrak{X}}$, a commutative semi-ring $\langle A, \oplus, \otimes, e^{\oplus}, e^{\otimes} \rangle$,[4] and a labeling function λ :

[4]A semi-ring $\langle A, \oplus, \otimes, e^{\oplus}, e^{\otimes} \rangle$ is an algebraic structure such that $\oplus$ and $\otimes$ are associative binary operations over A; $\oplus$ is commutative; $e^{\oplus}$ is the identity element of $\oplus$ and $e^{\otimes}$ that of $\otimes$; $\otimes$ is left and right distributive over $\oplus$; and finally $e^{\oplus} \otimes a = a \otimes e^{\oplus} = e^{\oplus}$ for any $a \in A$. A semi-ring is commutative if $\otimes$ is commutative too.

$Lit(\mathfrak{X}) \to A$, the AMC problem is to compute:

$$\mathrm{AMC}(\varphi) = \bigoplus_{\substack{\alpha\colon \mathfrak{X}\to\{0,1\}\\ \alpha\models\varphi}} \bigotimes_{\substack{\ell\in Lit(\mathfrak{X})\\ \alpha(\ell)=1}} \lambda(\ell).$$

The pair $\langle \oplus, \lambda\rangle$ is called *neutral* if and only for every propositional atom $x \in \mathfrak{X}$, $\lambda(x) \oplus \lambda(\neg x) = e^{\otimes}$. Kimmig et al. (2017) proved that when φ is a DNNF circuit, $\oplus$ is idempotent,[5] and $\langle \oplus, \lambda\rangle$ is neutral, then the AMC problem can be solved in polynomial time.

We now show that Outcome(F) can be solved using the AMC problem when F is an additive rule. Consider the max-plus algebra—a commutative and idempotent semi-ring (Akian et al., 2006)—defined by $A = \mathbb{R}\cup\{-\infty, \infty\}$, $e^{\oplus} = -\infty$, and $e^{\otimes} = 0$, where $\oplus$ and $\otimes$ are the usual max and + operators over $\mathbb{R} \cup \{-\infty, \infty\}$.

Consider now an additive JA rule F associated with f. For a profile $\boldsymbol{J}$ we introduce a labelling function $\lambda_{\boldsymbol{J}}(\cdot)$ defined as follows for every literal $\ell_x \in Lit(\mathfrak{X})$ of the atom $x \in \mathfrak{X}$:

$$\lambda_{\boldsymbol{J}}(\ell_x) = f(\boldsymbol{J}, \ell_x) - \max\left(f(\boldsymbol{J}, x), f(\boldsymbol{J}, \neg x)\right),$$

where f is the function with respect to which F is an additive rule.

Since we have $\max(\lambda_{\boldsymbol{J}}(x), \lambda_{\boldsymbol{J}}(\neg x)) = 0$ for every $x \in \mathfrak{X}$, it is easy to see that the pair $\langle \lambda_{\boldsymbol{J}}, \oplus\rangle = \langle \lambda_{\boldsymbol{J}}, \max\rangle$ is neutral.

For every such labelling function, we then have:

$$\begin{aligned}
&\operatorname*{argmax}_{J\in\mathfrak{J}(\Gamma)} \sum_{\ell_x\in aug(J)} \lambda_{\boldsymbol{J}}(\ell_x)\\
&= \operatorname*{argmax}_{J\in\mathfrak{J}(\Gamma)} \left[\sum_{\ell_x\in aug(J)} \Big(f(\boldsymbol{J}, \ell_x) - \max\left(f(\boldsymbol{J}, x), f(\boldsymbol{J}, \neg x)\right)\Big)\right] \qquad (1)\\
&= \operatorname*{argmax}_{J\in\mathfrak{J}(\Gamma)} \left[\sum_{\ell_x\in aug(J)} f(\boldsymbol{J}, \ell_x) - 2\times\sum_{x\in\mathfrak{X}} \max\left(f(\boldsymbol{J}, x), f(\boldsymbol{J}, \neg x)\right)\right] \qquad (2)\\
&= \operatorname*{argmax}_{J\in\mathfrak{J}(\Gamma)} \sum_{\ell\in aug(J)} f(\boldsymbol{J}, \ell)) = F(\Gamma, \boldsymbol{J})
\end{aligned}$$

Let us briefly explain the computations above. The transition between lines (1) and (2) comes from the fact that $aug(J)$ include exactly one literal for each propositional atom in $\mathfrak{X}$. Observe then that $2\times\sum_{x\in\mathfrak{X}} \max\left(f(\boldsymbol{J}, x), f(\boldsymbol{J}, \neg x)\right)$ does not depend on the J over which the argmax loops, and can thus be dropped.

We can then solve Outcome(F) by using the AMC problem. For Γ, $\boldsymbol{J}$ and $L \subseteq Lit(\mathfrak{X})$ given as inputs of the Outcome(F) problem, we will solve the AMC problem twice: first for $\varphi = \Gamma$ and then for $\varphi = \Gamma'$, where Γ' is obtained from Γ by fixing the value of the atoms as in L. If the solution of the AMC problem is the same in both cases, we answer the Outcome(F) problem by the positive.

To conclude the proof, observe that the pair $\langle \max, \lambda\rangle$ is neutral. Thus, the AMC problem can be solved in polynomial time when φ is a DNNF circuit (Kimmig et al., 2017). Hence, the Outcome(F) problem can also be solved in polynomial time when Γ is a DNNF circuit. □

[5] A binary operator $\oplus$ over A is idempotent if for every $a \in A$, we have: $a \oplus a = a$.

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* 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

At this point we know that we can efficiently compute the outcome of JA rules when the integrity constraint is represented as a DNNF circuit, but we still need to demonstrate that it in fact is possible to encode PB problems as integrity constraints of this kind. So we move on to the description of embeddings of PB into JA returning integrity constraints represented as DNNF circuits. In doing so, we follow De Haan (2018) 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. To know whether it is possible to select a given project, we keep track of the amount of resources that has been used so far. Selecting a project can thus only be done if it would not lead to a violation of the budget constraint.

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:

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

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 DNNF circuit presented in Figure 3. Note that we simplified the DNNF circuit a bit to improve its readability. △

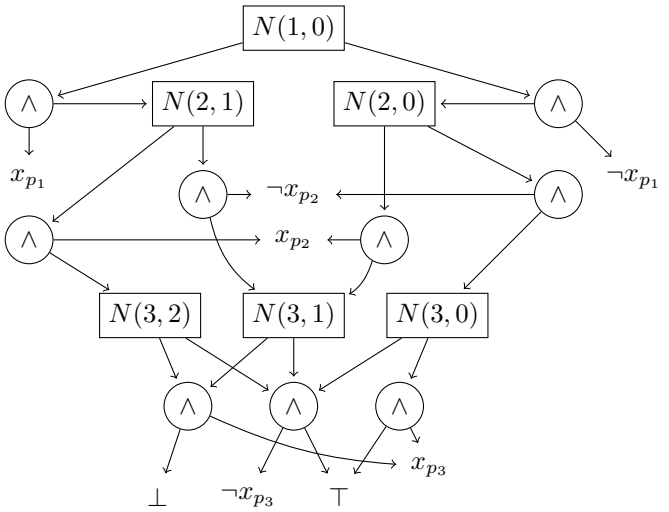

Fig. 3 (Simplified) DNNF circuit produced by *TE* on the instance of Example 1.

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* $\mathcal{O}(m \times |\{c(\pi) \mid \pi \subseteq \mathcal{A}(I)\}|)$.

Proof Let $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ be a PB instance, and Γ an integrity constraint such that $\Gamma = TE(I)$.

We first show that Γ is represented as DNNF circuit. First, observe that Γ is a Boolean circuit rooted in $N(0, \mathbf{0}_d)$. Next, observe that every $\vee$-node is of the form $(x \wedge \beta_1) \vee (\neg x \wedge \beta_2)$, where $x \in \mathfrak{X}$ is a propositional atom and β_1, β_2 are either $\vee$-nodes, $\bot$, or $\top$. This implies that Γ is represented as an NNF circuit. Because each project is only considered once, the propositional atom corresponding to the project cannot appear in two distinct conjuncts. Hence, Γ is a DNNF circuit.

Observe that there are at most $m \times |\{c(\pi) \mid \pi \subseteq \mathcal{A}(I)\}|$ $\vee$-nodes in Γ—one for each $N(j, \boldsymbol{v})$ for which the budget is not exceeded—all of them having at most two child $\wedge$-nodes. There are moreover $2m + 2$ leaves, one per literal and two for $\bot$ and $\top$, hence the size of the DNNF circuit.

We now show that the tractable embedding is correct. Observe that a branch leading to the $\bot$-leaf is chosen if and only if one would violate the budget limit by selecting a project p_j. Hence, finding an assignment that does not lead to a $\bot$ leaf in Γ can only be done by selecting feasible projects. The set of such assignments defines the set of outcomes satisfying Γ, so $\tau(E(I)) \subseteq \mathcal{A}(I)$. Now, consider $\pi \in \mathcal{A}(I)$. Since π is feasible, it is clear that there exists a branch in the DNNF circuit Γ along which the selected projects correspond exactly to those that are in π. We thus have $\tau(E(I)) = \mathcal{A}(I)$. □

At this point, it should be noted that the exponential factor in the size of the embedding, namely $|\{c(\pi) \mid \pi \subseteq \mathcal{A}(I)\}|$, is bounded from above by the product of the budget limits for each resource. Hence, the corresponding DNNF circuit is of size in $\mathcal{O}(m \times \prod_{r \in \mathcal{R}} b_r)$. This is pseudo-polynomial in the size of the PB instance when the number of resources is fixed. The next natural question then is whether we can do better. For instance, is it possible to reduce the size to something pseudo-polynomial in the size of the PB instance regardless of the number of resources, i.e., in $\mathcal{O}(m \times \sum_{r \in \mathcal{R}} b_r)$? The following result answers this question in the negative.

Proposition 4 *There exists no embedding into a DNNF circuit that can be computed in time polynomial in $m + \sum_{r \in \mathcal{R}} b_r$ for any instance $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$, unless* P = NP.

Proof We first show that the following problem is strongly NP-complete.

Maximal Exhaustive Allocation	
Input:	A PB instance I and a natural number $k \in \mathbb{N}$
Question:	Is there a $\pi \in \mathcal{A}_{EX}(I)$ such that $\lvert\pi\rvert \geq k$?

First, note that Maximal Exhaustive Allocation obviously is in NP, given that checking that a budget allocation is exhaustive and selects at least k projects is just a matter of looping through all the projects of the instance.

We now show that Maximal Exhaustive Allocation is strongly NP-hard. To do so we reduce from the 3-dimensional Matching problem, which was shown to be NP-complete by Karp (1972). Note that, since its input does not involve numbers, the 3-dimensional Matching problem is immediately also strongly NP-complete.

3-dimensional Matching	
Input:	A finite set T and a set $X \subseteq T \times T \times T$
Question:	Is there a set $M \subseteq X$ such that $\lvert M\rvert = \lvert X\rvert$, and for all (x_1, x_2, x_3) and (y_1, y_2, y_3) in M, we have $x_1 \neq y_1$ and $x_2 \neq y_2$ and $x_3 \neq y_3$?

Consider, without loss of generality, an instance of the 3-dimensional Matching problem $\langle T, X \rangle$ such that $T = \{1, \ldots, t\}$ and $X = \{x_1, \ldots, x_{|X|}\}$. The corresponding PB instance is $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ where the set of resources is $\mathcal{R} = \{r_i^j \mid i \in T, j \in \{1, 2, 3\}\}$ and for every resource $r \in \mathcal{R}$, we have $b_r = 1$. The set of projects is $\mathcal{P} = \{p_1, \ldots, p_{|X|}\}$. Consider project $p_i \in \mathcal{P}$ corresponding to $x_i = (x_i^1, x_i^2, x_i^3) \in X$, its cost is 1 for the three resources $r^1_{x_i^1}$, $r^2_{x_i^2}$ and $r^3_{x_i^3}$ and 0 for any other resource. Moreover we set $k = |X|$. We claim that the answer for the 3-dimensional Matching problem on $\langle T, X \rangle$ is yes if and only if the answer for the Maximal Exhaustive Allocation problem on $\langle I, k \rangle$ is yes too.

To a matching $M \subseteq X$ corresponds the budget allocation $\pi = \{p_i \mid x_i \in M\}$. A matching $M \subseteq X$ is a solution of the 3-dimensional Matching problem if and only if no triplet in M share a coordinate. Because of the budget limit, this is possible if and only if the corresponding budget allocation π is feasible. Moreover $|M| = |X|$

if and only if $|\pi| = |X| = k$. Note that in this case π would be exhaustive, which proves the claim. The reduction is clearly done in polynomial time which shows that the Maximal Exhaustive Allocation problem is strongly NP-complete.

To conclude the proof, we now show that, if there exists an embedding of PB into a DNNF circuit of size polynomial in $m + \sum_{r \in \mathcal{R}} b_r$, then we would be able to solve the Maximal Exhaustive Allocation problem in pseudo-polynomial time, which would imply that P = NP.

Let us prove that last claim. Let Γ be the integrity constraint returned by a suitable exhaustive embedding on an arbitrary instance I. Note that the answer to Maximal Exhaustive Allocation problem is yes if and only if the outcome of the Algebraic Model Counting (AMC) problem is at least $k - m$ when run on Γ with the max-plus algebra (see the proof of Theorem 1 for the definitions) and the labeling function λ such that $\lambda(x) = 0$ and $\lambda(\neg x) = -1$ for all $x \in \mathfrak{X}$. Since Γ is a DNNF circuit and the pair $\langle \lambda, \oplus \rangle$ is neutral, we can compute the outcome of the AMC problem in polynomial, proving the claim. □

Even though there is no hope to find pseudo-polynomial embeddings when the number of resources is unbounded, we still argue that the embedding is efficient for realistic scenarios. First in most typical PB processes, the number of resources would likely be small. Indeed the difficulty of assessing the different costs and of the deliberation and voting processes increases significantly with the number of dimensions. It thus seems particularly unlikely that the cost will be expressed in more than, say, five dimensions. Moreover, if the number of resources is fixed, or at least bounded, the size of the DNNF circuit will not really present a serious limitation with state-of-the-art solvers.

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 the completion of some projects is directly dependent on the completion of some others.

Take a PB instance $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$. We introduce a set of implications, $Imp \subseteq \mathcal{L}_{\mathfrak{X}}$, 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' that are 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. In the case that ℓ_{x_p} is positive (resp. negative), such an implication indicates that p can be selected (resp. cannot be selected) only if p' is selected, when $\ell_{x_{p'}}$ is positive, or not selected, when $\ell_{x_{p'}}$ is negative. A budget allocation π satisfies the set of implications Imp if and only if the previously described semantics is satisfied. Moreover, we will write $\ell_{x_p} \to^* \ell_{x_{p'}}$ if there is a chain of implication in Imp linking ℓ_{x_p} to $\ell_{x_{p'}}$.

In terms of applications, this approach to model dependencies is quite flexible. For instance, it allows us to model the fact that project p_1 can only be implemented if projects p_2 and p_3 would also be implemented. This would be encoded as $Imp = \{x_{p_1} \to x_{p_2}, x_{p_1} \to x_{p_3}\}$. At the same time, we can also model "negative" dependencies, or "incompatibilities", where p_1 can only

be implemented if p_2 is not: $Imp = \{x_{p_1} \to \neg x_{p_2}\}$. We are not aware of any previous work that would allow for such constraints to be expressed within the PB framework. Note though that one can implement dependencies of the type $x_p \to \neg x_{p'}$ by having specific budget constraints over categories over project as done by Jain et al. (2021).

To start our analysis, we show that finding a feasible budget allocation when there are implications between project is an NP-complete problem.

Proposition 5 *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 the case of a single resource.*

Proof The problem of finding a feasible budget allocation satisfying *Imp* is clearly in NP. Indeed, checking that the budget limit is not exceeded can be done by summing the costs of the selected projects. Moreover, verifying that the set of implications is satisfied simply amounts at checking the truth value of each implication in *Imp*. Both of these problems can be solved in polynomial time.

To show that the problem is NP-hard, we reduce from the NP-complete problem 2-CNF Minimal Model[6] (Ben-Eliyahu and Dechter, 1996).

2-CNF Minimal Model	
Input:	A formula $\varphi \in \mathcal{L}_{\mathfrak{X}}$ in conjunctive normal form with exactly two literals per clauses and $k \in \mathbb{N}$.
Question:	Is there a model α such that $\alpha \models \varphi$ and $\lvert\{p \in \mathfrak{X} \mid \alpha(p) = 1\}\rvert \leq k$?

Take an instance $\langle \varphi, k \rangle$ of the 2-CNF Minimal Model problem. We construct the following participatory budgeting instance I. The set of resources is $\mathcal{R} = \{r\}$ with budget limit $b_r = k$. There is one project per propositional atom in φ, $\mathcal{P} = \{p_x \mid x \in P\}$, and $c(p) = 1$ for every $p \in \mathcal{P}$. Finally, the set of implications *Imp* is the set of clauses in φ.

We claim that there exists a model of φ setting no more than k variables to true if and only if there exists a feasible budget allocation for I that satisfies the set of implications *Imp*. Indeed, to a given truth assignment α, corresponds the budget allocation $\pi = \{p_x \mid \alpha(x) = 1\}$. Observe first that there exists $\alpha \models \varphi$ if and only if π satisfies *Imp*. Moreover, since every project is of cost 1, the cost of π is exactly the number of project that are selected. As the set of projects is the set of variables in φ, the cost of π is also equal to the number of variables set to true in α. Because the budget limit for resource r is k, the budget allocation π satisfies the budget limit if and only no more than k propositional atoms are set to true in α.

Observing that this reduction clearly can be done in polynomial time concludes the proof. □

[6] A propositional logic formula $\varphi \in \mathcal{L}_{\mathfrak{X}}$ is in Conjunctive Normal Form (CNF) if it is expressed as a conjunction of clauses, where a clause is a disjunction of literals. It is moreover a 2-CNF formula if all clauses are of size 2, *i.e.*, if φ is a conjunction of disjunctions over two literals.

Based on this result, we cannot hope to find an embedding into a DNNF circuit of polynomial size. However, we can still define an interesting embedding the size of which is parameterized by some parameter on the structure of the implication set (in the spirit of parameterized complexity (Downey and Fellows, 2013)). The parameter in question is the pathwidth (Robertson and Seymour, 1983; Bodlaender, 1998) of the interconnection graph. Let us define these two terms.

First, we introduce the *interconnection graph* G of a set of implications *Imp*. It is the graph $G = \langle \mathcal{P}, E \rangle$ where there is an edge $\{p, p'\} \in E$ between projects p and p' if and only if there exists an implication in *Imp* linking the two projects, i.e., *Imp* includes at least one implication of the form $\ell_{x_p} \to \ell_{x_{p'}}$ for some $\ell_{x_p} \in \{x_p, \neg x_p\}$, $\ell_{x_{p'}} \in \{x_{p'}, \neg x_{p'}\}$.

Second, let us discuss the pathwidth of a graph $G = \langle V, E \rangle$. A path-decomposition of a graph G is a vector of subset of vertices $(V_1, \ldots, V_q)$, called bags, such that: (i) for every edge $(v_1, v_2) \in E$, there is a bag V_i such that v_1 and v_2 are in V_i, and (ii) for every $i \leq j \leq k$ we have $V_i \cap V_k \subseteq V_j$. The second property should be understood as saying that the set of bags in which a given vertex appears is contiguous. The width of a tree decomposition is the size of the largest bag minus one. The pathwidth of a graph G is the minimum width of any of its path-decomposition.

We are now equipped with all the definitions we need to formulate our embedding for dependencies, denoted by TE_{dep}.

Theorem 6 *Let $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ be a PB instance and Imp a set of implications over I. Then there exists a correct embedding from I and Imp to an integrity constraint expressed as a DNNF circuit Γ with size in $\mathcal{O}\left(m \times |\{c(\pi) \mid \pi \subseteq \mathcal{A}(I)\}| \times 2^k\right)$, where k is the pathwidth of the interconnection graph of Imp.*

Proof The proof will be presented in several steps. We will first present our embedding TE_{dep}, then investigate the size of the integrity constraint returned and finally show that the embedding is correct. In the following we assume that $\mathfrak{X}$ is exactly the set $\{x_p \mid p \in \mathcal{P}\}$.

Let $G = \langle \mathcal{P}, E \rangle$ be the interconnection graph of *Imp*. We order the projects in the same order in which they are introduced in an optimal path-decomposition of G. Then, following the idea developed for *TE*, 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 $L \subseteq \mathit{Lit}(\mathfrak{X})$ a subset of literals. Intuitively, the set L specifies the literals that we selected and that we should remember because they might trigger implications later on. The nodes $N(j, \boldsymbol{v}, L)$ are then defined such that:

- If $j = m + 1$, then $N(j, \boldsymbol{v}, L) = \top$;
- If both the positive literal x_{p_j} and the negative literal $\neg x_{p_j}$ are implied by some literal in L according to *Imp*, then $N(j, \boldsymbol{v}, L) = \bot$;
- If the positive literal x_{p_j} is implied by some literal in L according to *Imp*, and $\boldsymbol{v} + c(p_j) \leq \boldsymbol{b}$, then $N(j, \boldsymbol{v}, L) = N(j + 1, \boldsymbol{v} + c(p_j), L \cup \{x_{p_j}\})$;

- If the positive literal x_{p_j} is implied by some literal in L according to *Imp*, but there exists a resource $r_q \in \mathcal{R}$ such that $v_q + c(p_j, r_q) > b_q$, then $N(j, \boldsymbol{v}, L) = \bot$;
- If the negative literal $\neg x_{p_j}$ is implied by some literal in L according to *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:

$$N(j, \boldsymbol{v}, L) = (x_{p_j} \wedge N(j+1, \boldsymbol{v} + c(p_j), L \cup \{x_{p_j}\})) \vee (\neg x_{p_j} \wedge N(j+1, \boldsymbol{v}, L \cup \{\neg x_{p_j}\}));$$

- Otherwise, $N(j, \boldsymbol{v}, L) = (x_{p_j} \wedge \bot) \vee (\neg x_{p_j} \wedge N(j+1, \boldsymbol{v}, L \cup \{\neg x_{p_j}\}))$.

The tractable embedding with dependencies, written TE_{dep}, refers to the integrity constraint defined by $N(j, \mathbf{0}_m, \emptyset)$.

It is clear that the integrity constraint returned by the embedding described above is represented as a DNNF circuit. The proof is very similar to the one for the tractable embedding (Proposition 3).

We now investigate the maximum size of the DNNF circuit. We need to count the maximum number of $\vee$-nodes, that is the number of $N(j, \boldsymbol{v}, L)$. At a first glance, the number of possible $L \subseteq Lit(\mathfrak{X})$ is upper-bounded by $2^{|\mathcal{P}|}$. However, we can have a more fine-grained analysis of this last term. The set L is used to keep track of the projects for which a truth value has already been assigned and that could imply the truth value of some other project appearing later in the ordering. However, since the projects are considered following an optimal path-decomposition of the interconnection graph, we know that we never need to remember the truth value of more than $k+1$ projects, where k is the pathwidth of the interconnection graph. Indeed, by definition of a path-decomposition, for any project p_j, whenever we consider another project $p_{j'}$ such that $p_{j'}$ never appears in a bag together with p_j, we no longer need to keep track of the truth value associated with x_{p_j} as p_j will never be involved in implications with subsequent projects. Overall, the size of the DNNF circuit is in $\mathcal{O}\left(m \times |\{c(P) \mid P \subseteq \mathcal{P}\}| \times 2^k\right)$.

Finally, we show that TE_{dep} is correct. Remember that this is the case if for every instance I:

$$\{\tau(J) \mid J \in \mathfrak{J}(TE_{dep}(I, Imp))\} = \mathcal{A}(I, Imp).$$

Consider a JA outcome $J \in \mathfrak{J}(TE_{dep}(I, Imp))$. It is clear that $\tau(J)$ does not exceed the budget limit as every were selecting a project could lead to a too high cost, the branch in the DNNF circuit ends up in the $\bot$ leaf. We then need to prove that $\tau(J)$ satisfies *Imp*. Observe that every time a literal is implied by a literal that has been previously given a truth value, we follow the implication. Hence it can never be the case that the premise of an implication is set to true but not the conclusion. Moreover, every time triggering implications would lead to an inconsistent outcome (a project being both selected and not selected), the branch in the DNNF circuit also leads to the $\bot$ leaf. Overall $\tau(J)$ does satisfy *Imp*. We have thus proved that $\{\tau(J) \mid J \in \mathfrak{J}(TE_{dep}(I, Imp))\} \subseteq \mathcal{A}(I, Imp)$. The proof for the reversed inclusion is exactly as that presented in Proposition 3. □

The size of the DNNF circuit produced by the embedding includes a factor 2^k, where k is the pathwidth of the interconnection graph of *Imp*. The value of k is in general upper-bounded by the number of projects, for instance in the case where the interconnection graph is complete (when there are dependencies between any two projects). Once again, it seems fair to assume projects not to be very interconnected, leading to small values of k in practice.

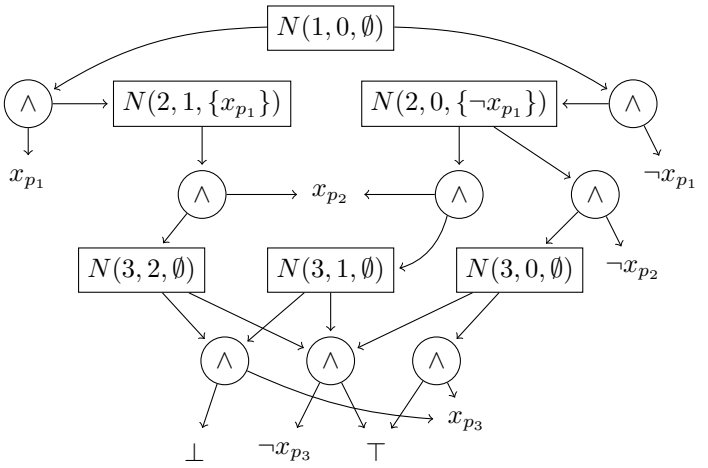

Fig. 4 (Simplified) DNNF circuit produce by TE_{dep} on the instance of Example 2.

Let us illustrate the embedding used in the proof on an example.

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. In our model, this means that $Imp = \{x_{p_1} \to x_{p_2}\}$. An optimal path-decomposition of the interconnection graph is thus $(\{p_1, p_2\}, \{p_3\})$. So we will consider the projects in the following order: p_1, p_2, p_3. The embedding TE_{dep} would return the DNNF circuit presented in Figure 4. It is important to understand how we can "forget" about the truth values of x_{p_1} and x_{p_2} once we consider project p_3. △

We conclude this section with a small detour into the field of Knowledge Compilation (see, e.g., Marquis, 2015). Embedding PB problems with dependencies into a DNNF circuit is actually equivalent to asking whether the conjunction of a 2-CNF formula with a DNNF circuit can be efficiently encoded as a DNNF circuit of polynomial size. It turns out to be impossible to do so unless the polynomial hierarchy collapses.

Proposition 7 *A 2-CNF formula $\varphi \in \mathcal{L}_{\mathfrak{X}}$ cannot be compiled into a DNNF circuit of size polynomial in the size of φ unless the polynomial hierarchy collapses to the second level.*

Proof We will prove that if 2-CNF formulas can be compiled into polynomial-size DNNF circuits, then the NP-complete problem Clique would be in P/poly, using similar techniques as Cadoli et al. (2002). Because of the Karp-Lipton theorem (Karp and Lipton, 1980), this would immediately entail the Polynomial Hierarchy to collapse at the second level.

Let us first introduce the problem Clique shown to be NP-complete by Karp (1972). Given an undirected graph $G = \langle V, E \rangle$, we say that a subset of vertices

$V' \subseteq V$ forms a clique in G if for every $v_1, v_2 \in V'$, we have $\{v_1, v_2\} \in E$. The decision problem Clique is then defined as follows.

Clique	
Input:	An undirected graph $G = \langle V, E \rangle$ and an integer $k \in \mathbb{N}$.
Question:	Is there a clique $V' \subseteq V$ such that $\lvert V' \rvert = k$ in G?

For any two integers $n, k \in \mathbb{N}$, we introduce a 2-CNF formula $\varphi(n, k)$ such that the answer of Clique on a graph $G = \langle V, E \rangle$ with $|V| = n$ and k can be deduced by means of queries of the following problem the we call Max SAT Extension: Given a partial truth assignment α of φ, what is the maximum number of variables that can be set to true in any extension of α in such a way that α satisfies $\varphi(n, k)$. Importantly, the formula $\varphi(n, k)$ is the same for all graphs with n vertices and inquired clique size k. In the following we will assume that the vertices in V are named $v_1, \ldots, v_n$.

Let us describe how to construct $\varphi(n, k)$. For every $i, i' \in \{1, \ldots, n\}$ and each $1 \leq j < j' \leq k$, we introduce the variable $x_{j,j',i,i'}$. The idea of these variables is to indicate which vertex was selected at a given position in an arbitrary ordering of the clique. So having $x_{j,j',i,i'}$ set to true means that in the ordering of the clique in which vertex v_i is at position j, then vertex $v_{i'}$ is at position j'. Now for this to be correct, we need to enforce that no two vertices can be selected at the same position. For every $j_1, j_2, j_3, j_4 \in \{1, \ldots, k\}$ such that $j_1 < j_2$, $j_1 < j_3$ and $j_4 < j_2$, and for every $i_1, i_2, i_3, i_4 \in \{1, \ldots, n\}$, we thus add the following two clauses to the formula $\varphi(n, k)$:

$$(\neg x_{j_1,j_2,i_1,i_2} \vee \neg x_{j_1,j_3,i_3,i_4})$$
$$(\neg x_{j_1,j_2,i_1,i_2} \vee \neg x_{j_4,j_2,i_3,i_4})$$

Now, we want to check whether any given graph $G = \langle V, E \rangle$ with $|V| = n$ has a clique of size k. Consider the formula $\varphi(n, k)$. We first create a partial truth assignment α that sets $x_{j,j',i,i'}$ and $x_{j,j',i',i}$ to false for all $1 \leq j < j' \leq k$ and all $i, i' \in \{1, \ldots, n\}$ such that $\{v_i, v_{i'}\} \notin E$. Now it is easy to see that G has a clique of size k if and only if we can extend this α to a (non-partial) truth assignment that satisfies $\varphi(n, k)$ and that sets at least $k(k-1)/2$ variables to true.

At this point, it is important to observe that the problem Max SAT Extension can be solved in polynomial time if the formula provided is a DNNF circuit. It is a simple variant of the Maximum Model problem that we can solve via dynamic programming on DNNF circuits.

Now, suppose we can compile 2-CNF formulas into DNNF circuits in polynomial-space, then we compile $\varphi(n, k)$ into a DNNF for each n and each k. Call this DNNF circuit $D(n, k)$. From the above, it should be clear now that with $D(n, k)$ we can solve the Clique problem on a graph G with n nodes and an inquired clique size of k in polynomial time.

To conclude the proof we show that this would then all mean that Clique is in P/poly. We first informally define P/poly, we refer the reader to Chapter 6 in the book by Arora and Barak (2009) for a formal definition. P/poly is a complexity class that contains problems that can be solved in polynomial time by a deterministic Turing machine that has access to one advice string per size of the input. In our case, the advice string for an input of size n will be the sequence $(D(n, 1), \ldots, D(n, n))$. Note that the size of the advice string is polynomial in n as each DNNF circuit in it is. For any input of Clique, we can then find the corresponding DNNF circuit $D(n, k)$ to

solve the problem in polynomial time. This proves that CLIQUE would be in P/poly, and so that NP ⊆ P/poly (since CLIQUE is NP-complete). The Karp-Lipton Theorem (Karp and Lipton, 1980) then immediately implies that the polynomial hierarchy would collapse to the second level. □

We already knew that there was little hope for a DNNF circuit encoding of PB instances with dependencies in polynomial-time (unless P = NP). Under a stronger computational complexity assumption, we now have gone further by showing that it is also unlikely to be able to do this in polynomial space.

3.4 Quotas on Types of Projects

The second extension of PB we consider is when projects are grouped into categories, or types, that are constrained by quotas.[7] The idea is that the projects belong to various types (health, education, environment to name a few) and that certain quotas over these types are to be respected by the final budget allocation (at least two health-related projects must be funded, for instance). We model this idea by defining a type system.

Let us now define what a type system is. 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;
- $\mathcal{Q} = \langle Q, \oplus, e^{\oplus}, \leq_Q \rangle$ is an ordered group[8] 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 $a \leq_Q b$;
- $f : \mathcal{T} \times \mathcal{A}(I) \to Q$ is a type aggregator.

For any $T \in \mathcal{T}$ 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 π is feasible if the quotas are respected—that is, if and only if for every type $T \in \mathcal{T}$, we have $q(T)^- \leq_Q f(T, \pi) \leq_Q q(T)^+$.

The type aggregator $f(\cdot)$ can be defined in several different ways. 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 $Q = \mathbb{N}$, $\oplus$ is the usual addition operator on numbers with identity element 0, and $\leq_Q$ is the usual linear order on numbers. Now for every type $T \in \mathcal{T}$ and budget

[7] After our initial work on this topic (Rey et al., 2020), Jain et al. (2021) studied a specific subcase of our model, namely PB instances in which projects are grouped into categories and quotas regarding the total cost of projects from within each category need to be satisfied. There is a small overlap between the two papers: Proposition 8 in this section is implied by Theorem 10 of Jain et al. (2021). Other results are incomparable and the two papers nicely complement one another.

[8] A group $\langle Q, \oplus, e^{\oplus} \rangle$ is an algebraic structure equipped with a binary operation $\oplus$ over Q that is associative, that has an identity element $e^{\oplus}$, and such that for every $a \in Q$, there exists a unique $b \in Q$ such that $a \oplus b = e^{\oplus}$ and $b \oplus a = e^{\oplus}$. An ordered group $\langle Q, \oplus, e^{\oplus}, \leq_Q \rangle$ is a group $\langle Q, \oplus, e^{\oplus} \rangle$ equipped with a total order $\leq_Q$ over Q.

allocation π, the cardinality-type aggregator is defined as follows:

$$f^{\text{card}}(T, \pi) = |\pi \cap T|.$$

- **Cost-type aggregator.** The quotas define lower and upper bounds on the total cost of the projects selected for each type. Here $Q = \mathbb{R}^d_{\geq 0}$, $\oplus$ is the component-wise addition over vectors with identity element $\mathbf{0}_d$, and $\leq_Q$ is the component-wise order over vectors.[9] Now for every type $T \in \mathcal{T}$ and budget allocation π, the cost-type aggregator is defined as follows:

$$f^{\text{cost}}(T, \pi) = \sum_{p \in \pi \cap T} c(p).$$

We first show that deciding whether there is a feasible budget allocation with a given type system is NP-complete, for both the cardinality- and the cost-type aggregator.

Proposition 8 *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 π is* NP*-complete when f is either the cardinality or the cost-type aggregator, and* NP*-hardness holds even for a single resource.*

Proof The problem is clearly in NP. Indeed, verifying that the budget allocation π does not exceed the budget limit can be done easily by summing the costs of the selected projects. Note that both the cardinality- and the cost-type aggregators can be computed in polynomial time. Then, verifying that every quota is respected is just a matter of scanning π for each quota.

Let us show now that the problem is NP-hard. To do so we reduce from the NP-hard Set Splitting problem (Garey and Johnson, 1979) described below.

Set Splitting	
Input:	A collection C of subsets of a given set S.
Question:	Are there two sets S_1 and S_2 partitioning S such that $\forall c \in C, c \nsubseteq S_1$ and $c \nsubseteq S_2$?

Let $\langle C, S \rangle$ be an instance of Set Splitting. We construct a participatory budgeting instance $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ such that $\mathcal{R} = \{r\}$, and $b_r = |S|$. There is one project per element in S, $\mathcal{P} = \{p_s \mid s \in S\}$, and $c(p_s) = 1$ for every $s \in S$. Thus, the budget limit can never be exceeded. The corresponding set of types is $\mathcal{T} = \{\{p_s \mid s \in c\} \mid c \in C\}$ and for a given $T \in \mathcal{T}$, the quota is $q(T) = (1, |T| - 1)$. With one resource and projects whose costs are in $\{0, 1\}$, the cardinality-type aggregator and the cost-type aggregator coincide.

We claim that $\langle C, S \rangle$ is a yes-instance of Set Splitting if and only if there exists a feasible budget allocation in the instance I with the previous type system. For a

[9] For two vectors $\boldsymbol{v}$ and $\boldsymbol{v'}$ of the same length we write $\boldsymbol{v} \leq \boldsymbol{v'}$ whenever each component of $\boldsymbol{v}$ is no larger than the corresponding component in $\boldsymbol{v'}$.

given partition of S, (S_1, S_2), a suitable corresponding budget allocation is $\pi = S_1$ (or equivalently $\pi = S_2$).

A partition (S_1, S_2) is a solution of the Set Splitting problem if and only if for every $c \in C$, at least one element of c is in S_1 and at least one element of c is not in S_1 (and is then in S_2). Based on the type system we defined, this is equivalent to π satisfying the quota associated to c. Moreover, observe that every budget allocation respects the budget limit. Hence (S_1, S_2) is a solution of the Set Splitting problem if and only if π is a feasible budget allocation.

The reduction is clearly done in polynomial time, hence the problem of finding whether a feasible budget allocation exists is NP-complete when using both the cost and the cardinality type aggregator. □

Once again, this implies that no efficient embedding can be defined for this extension. In what follows we present a parameterized embedding for PB with types and quotas.

The embedding 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 $\pi \in \mathcal{A}(I)$:

$$f(T, \pi) = \sum_{p \in \pi} s(p).$$

Note that both the cardinality- and the cost-type aggregators are additive, with $s^{\text{card}}(p) = 1$ and $s^{\text{cost}}(p) = c(p)$ respectively.

As for the dependencies case, the size of our embedding will be parameterized by the pathwidth of a graph. This time, it will be the *overlap graph* of a type system. 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 $\{T, T'\} \in E$ between types T and T' if and only $T \cap T' \neq \emptyset$, *i.e.*, T and T' do not overlap.

We are now ready to present the embedding.

Theorem 9 *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 with respect to the score type function s. Then 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 $\mathcal{O}\left(m \times |\{c(\pi) \mid \pi \subseteq \mathcal{A}(I)\}| \times k^\star\right)$ where $k^\star = \max_{T \in \mathcal{T}}(|\{f(T, \pi) \mid \pi \in \mathcal{A}(I)\}|)^{k+1}$ and where k is the pathwidth of the overlap graph of $\langle \mathcal{T}, \mathcal{Q}, q, f \rangle$.*

Proof We use a similar strategy as for Theorem 6. The general idea is that because the type aggregator is additive, we can keep track of the current value of the quotas, and then, when deciding whether a project can be selected or not, we can check the current quota value before making our choice.

Let $G = \langle \mathcal{P}, E \rangle$ be the overlap graph of $\langle \mathcal{T}, \mathcal{Q}, q, f \rangle$. We order the projects in the same order in which they are introduced in an optimal path-decomposition of

G. As usual, we will then define the $\vee$-nodes $N(j, \boldsymbol{v}, \boldsymbol{q})$, where j is a project index, $\boldsymbol{v} \in \mathbb{R}^d_{\geq 0}$ a vector of used resources and $\boldsymbol{q} \in Q^{|\mathcal{T}|}$ is a vector of current quota value. We introduce extra notation before defining them. Let $\mathcal{T}_{p_j} = \{T \in \mathcal{T} \mid p_j \in T\}$ be the set of types containing project p_j. For a given $\boldsymbol{q} \in Q^{|\mathcal{T}|}$, define $\boldsymbol{q}^{p_j}$ such that $\boldsymbol{q}^{p_j}_T = \boldsymbol{q}_T$ for every $T \notin \mathcal{T}_{p_j}$, and such that $\boldsymbol{q}^{p_j}_T = \boldsymbol{q}_T \oplus s(p_j)$ for every $T \in \mathcal{T}_{p_j}$. The $\vee$-nodes $N(j, \boldsymbol{v}, \boldsymbol{q})$ are then defined as follows:

- If $j = m + 1$ we have $N(j, \boldsymbol{v}, \boldsymbol{q}) = \top$;
- If there are two types $T_1, T_2 \in \mathcal{T}_{p_j}$ such that p_j is the last project in T_2 to be considered, and the quotas satisfy $\boldsymbol{q}^{p_j}_{T_1} >_Q q(T_1)^+$ but $\boldsymbol{q}^{p_j}_{T_2} <_Q q(T_2)^-$, then $N(j, \boldsymbol{v}, \boldsymbol{q}) = \bot$;
- If there is a type $T \in \mathcal{T}_{p_j}$ such that $\boldsymbol{q}^{p_j}_T >_Q q(T)^+$, then:
$$N(j, \boldsymbol{v}, \boldsymbol{q}) = (x_{p_j} \wedge \bot) \vee (\neg x_{p_j} \wedge N(j + 1, \boldsymbol{v}, \boldsymbol{q}));$$
- If there is a type $T \in \mathcal{T}_{p_j}$ such that p_j is the last project from T to be considered and the quota over T satisfies $\boldsymbol{q}_T <_Q q(T)^-$ and $\boldsymbol{q}^{p_j}_T <_Q q(T)^-$, then:
$$N(j, \boldsymbol{v}, \boldsymbol{q}) = \begin{cases} (x_{p_j} \wedge N(j + 1, \boldsymbol{v} + c(p_j), \boldsymbol{q}^{p_j})) \vee (\neg x_{p_j} \wedge \bot) & \text{if } \boldsymbol{v} + c(p_j) \leq \boldsymbol{b}, \\ \bot & \text{otherwise;} \end{cases}$$
- If $\boldsymbol{v} + c(p_j) \leq \boldsymbol{b}$, then:
$$N(j, \boldsymbol{v}, \boldsymbol{q}) = (x_{p_j} \wedge N(j + 1, \boldsymbol{v} + c(p_j), \boldsymbol{q}^{p_j})) \vee \left(\neg x_{p_j} \wedge N(j + 1, \boldsymbol{v}, \boldsymbol{q})\right);$$
- If there exists $r \in \mathcal{R}$ such that $\boldsymbol{v} + c(p_j, r) > b_r$, then:
$$N(j, \boldsymbol{v}, \boldsymbol{q}) = (x_{p_j} \wedge \bot) \vee (\neg x_{p_j} \wedge N(j + 1, \boldsymbol{v}, \boldsymbol{q})).$$

The tractable embedding for types and quotas, written TE_{quo}, refers to the integrity constraint defined by $N(1, \mathbf{0}_m, \mathbf{0}_{|\mathcal{T}|})$.

Similarly to the proof of Theorem 6, we can order the projects according to the ordering of types in a suitable path-decomposition of the overlap graph. By doing so, in each node $N(j, \boldsymbol{v}, \boldsymbol{q})$, we can "forget" all types in $\boldsymbol{q}$ for which we already considered all projects—thereby reducing the number of nodes $N(j, \boldsymbol{v}, \boldsymbol{q})$ for each j and $\boldsymbol{v}$.

It is clear that TE_{quo} returns an integrity constraint represented as a DNNF circuit. We now show that the size of DNNF circuit is in:

$$\mathcal{O}\left(m \times |\{c(\pi) \mid \pi \subseteq \mathcal{A}(I)\}| \times \max_{T \in \mathcal{T}}(|\{f(T, \pi) \mid \pi \in \mathcal{A}(I)\}|)^{k+1}\right),$$

where k is the path-width of the overlapping graph.

Let us begin by observing that the first two term of the product above are the same as for the other tractable embeddings presented before, we do not expend on them. Then, observe that a quota can take at most $\max_{T \in \mathcal{T}}(|\{f(T, \pi) \mid \pi \in \mathcal{A}(I)\}|)$ different values. Hence, each component of $\boldsymbol{q}$ can only have $\max_{T \in \mathcal{T}}(|\{f(T, \pi) \mid \pi \in \mathcal{A}(I)\}|)$ distinct values. Now, how do we get it to the power k and not $|\mathcal{T}|$ in the size of the DNNF circuit? The idea is similar to that of the proof of Theorem 6: Whenever all projects of a given type have been considered, the value of the corresponding quota will no longer change and we can thus "forget" about that type. Hence, the maximum number of types we need to keep track of their quota is upper bounded by $k+1$. That proves the claim about the size of the DNNF circuit. Once again, remember that we can use known FPT algorithm to find an optimal path-decomposition (Bodlaender and Kloks, 1996).

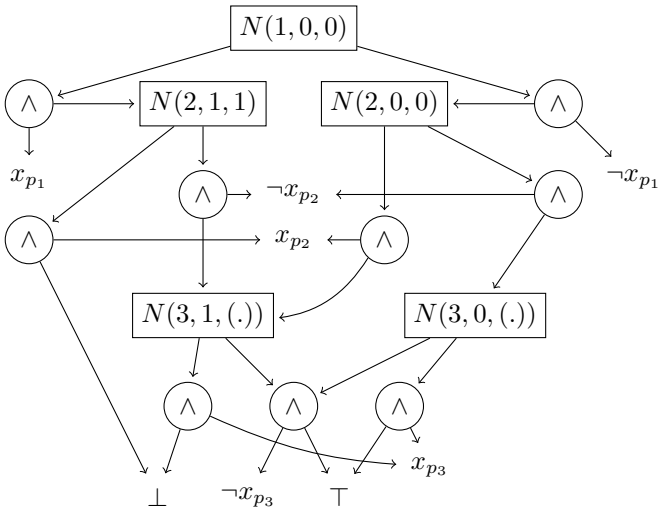

Fig. 5 (Simplified) DNNF circuit produce by TE_{dep} on the instance of Example 3 where (.) indicates the empty vector.

We now show that the embedding is correct. Consider a JA outcome $J \in \mathfrak{J}(TE_{quo}(I, Imp))$. It is clear from the two previous embeddings that $\tau(J)$ satisfies the budget constraint. Moreover, in TE_{quo}, before considering any project, if the current value of a quota violate the quota constraint associated with it, the branch in the DNNF circuit ends up on the $\bot$ leaf. Because the type aggregator is additive, we know that the quota values in $\boldsymbol{v}$ are the quota values of the corresponding budget allocation. Hence we have $\tau(j) \in \mathcal{A}(I)$. We have thus proved that $\{\tau(J) \mid J \in \mathfrak{J}(TE_{dep}(I, Imp))\} \subseteq \mathcal{A}(I, Imp)$. The proof for the reversed inclusion is exactly as the one presented for the proof of Proposition 3. □

Before discussing the size of the DNNF circuit produced by the embedding, we will present TE_{quo} on an example.

Example 3 Consider the instance described in Example 1. Assume that, projects p_1 and p_2 are health-related projects and that no more than one should be selected. In terms of our model, this means that we are considering the type system $\left\langle \{T\}, \langle \mathbb{N}, +, 0, \leq \rangle, q, f^{\text{card}} \right\rangle$, where $T = \{p_1, p_2\}$ and $q(T) = (0, 1)$. An optimal path-decomposition of the overlap graph is thus $(\{p_1, p_2\}, \{p_3\})$. So we will consider the projects in the following order: p_1, p_2, p_3. The embedding TE_{quo} would return the DNNF circuit presented in Figure 5. It is important to understand how we can "forget" about the truth values of x_{p_1} and x_{p_2} once we consider project p_3. △

Regarding the size of the DNNF circuit, it should be noted that the factor $\max_{T \in \mathcal{T}}(|\{f(T, \pi) \mid \pi \in \mathcal{A}(I)\}|)^{k+1}$ can be very high. However, for the

cardinality and the cost-type aggregators, we can derive the following bounds:

$$\max_{T\in\mathcal{T}} |\{f^{\text{card}}(T,\pi) \mid \pi \in \mathcal{A}(I)\}| = \max_{T\in\mathcal{T}} q(T)^+ \leq |\mathcal{P}|,$$
$$\max_{T\in\mathcal{T}} |\{f^{\text{cost}}(T,\pi) \mid \pi \in \mathcal{A}(I)\}| = \max_{T\in\mathcal{T}} q(T)^+ \leq \prod_{r\in\mathcal{R}} b_r.$$

This in particular implies that the integrity constraint for these quota aggregators would be reasonable of reasonable size, as long as k, the pathwidth of the overlap graph, is small.

The question is then how small or large k can be. Of course, in principle, it can be as large as the number of projects. That is the case if all projects appear in all types. On the other hand, the pathwidth of the overlapping graph would be 0 in the case where the types are non-overlapping. This special case is actually very natural. For instance, if one takes types to represent distinct areas of a city that are to be developed, then no project concerning one area will also concern another area. We get the following result for non-overlapping types.

Corollary 10 *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 over the score type function s. If the types are not overlapping, that is, if the overlapping graph of $\langle \mathcal{T}, \mathcal{Q}, q, f\rangle$ is the empty graph, the size of the integrity constraint returned by TE_{quo} is in $\mathcal{O}\left(m \times |\{c(P) \mid P \subseteq \mathcal{P}\}| \times \max_{T\in\mathcal{T}}(|\{f(T,\pi) \mid \pi \in \mathcal{A}(I)\}|)\right)$.*

4 Enforcing Exhaustiveness

Amongst the most basic requirements of a budget allocation is that of *exhaustiveness* (Aziz et al., 2018), sometimes also referred to as *budget monotonicity* (Talmon and Faliszewski, 2019). It states that the budget should be used as much as possible. This property is crucial for PB, given that it is usually assumed that achieving a project is always better than not achieving it. We first recall the definition.

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

Let us extend this definition to rules and embeddings. Remember that $\mathcal{A}_{EX}(I)$ is the set of feasible and exhaustive budget allocations for an instance I. An embedding $E : \mathcal{I} \to \mathcal{L}_{\mathfrak{X}}$ is said to be *exhaustive* if, for every instance $I \in \mathcal{I}$, we have $\tau(\mathfrak{J}(E(I))) \subseteq \mathcal{A}_{EX}(I)$. On the other hand, an exhaustive embedding E is correct if $\mathcal{A}_{EX}(I) = \tau(\mathfrak{J}(E(I)))$ for every instance I. Finally, 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.

Because the scenarios typically modelled using JA are rather different from PB, the exhaustiveness axiom is not satisfied by common JA rules. Indeed, JA rules are usually designed to be majority-consistent—they would always return the majoritarian outcome if it is admissible—which is incompatible with exhaustiveness. This has to do with the semantics of rejection (of a proposition) in the context of JA.

Proposition 11 *No majority-consistent JA rule is exhaustive.*

Proof Consider a correct but not exhaustive embedding E (for instance TE as defined above). 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 being unconstrained. 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 does not correspond to an exhaustive budget allocation. □

This result is far-reaching because exhaustiveness really clashes with basic properties of JA.[10] To circumvent this problem and to enforce exhaustiveness, we will investigate two approaches: either encoding exhaustiveness in the integrity constraint or designing new JA rules.

4.1 Exhaustive Embeddings

We introduce the *exhaustive tractable embedding*, which is an adaptation of the tractable embedding to maintain exhaustiveness when there is exactly one resource.

Consider a PB instance $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ with $\mathcal{R} = \{r\}$. Similarly to the previous embeddings, we introduce the $\vee$-nodes of the integrity constraint as $N(j, v, c^*)$, where j is a project index, v is the budget used in terms of resource r, and c^* is the cost of the cheapest non-selected project. They are defined as follows:

- If $j = m + 1$, we have:

$$N(j, v, c^*) = \begin{cases} \top & \text{if } c^* > b - v \\ \bot & \text{otherwise.} \end{cases}$$

- If $v + c(p_j) \leq b$, let $c^{*\prime} = \min(c^*, c(p_j))$, then we have:

[10] Were it not for our assumption that every project must have at least one supporter (which rules out certain profiles), Proposition 11 could be strengthened to say that no unanimous JA rule is exhaustive (F is *unanimous* if $F(J, \ldots, J) = \{J\}$ for all judgments J).

$$N(j, v, c^*) = \left(x_{p_j} \wedge N(j+1, v + c(p_j), c^*)\right) \vee \left(\neg x_{p_j} \wedge N(j+1, v, c^{*\prime})\right).$$

- Otherwise, $N(j, v, c^*) = \left(\neg x_{p_j} \wedge N(j+1, \boldsymbol{v}, c^*)\right) \vee (x_{p_j} \wedge \bot)$.

The exhaustive tractable embedding *ETE* returns the integrity constraint defined by $N(1, 0, \max_{p \in \mathcal{P}} c(p))$. We can prove that the embedding *ETE* behaves as it is expected to.

Proposition 12 *The exhaustive tractable embedding is correct and exhaustive, and returns an integrity constraint* Γ *represented as a DNNF circuit of size* $\mathcal{O}(m^2 \times |\{c(\pi) \mid \pi \subseteq \mathcal{A}(I)\}|)$*, for any* $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ *where* $|\mathcal{R}| = 1$.

Proof The structure of the embedding is very similar to that of the tractable embedding *TE* presented above. We only prove that the embedding is exhaustive. Let $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ be a PB instance. Consider an outcome $J \in \mathfrak{J}(ETE(I))$ and the budget allocation π such that $\pi = \tau(J)$. Note that the budget allocation π is exhaustive if and only if the cheapest not selected project does not fit in it. Observe that in the exhaustive tractable embedding we are keeping track of the cheapest project that has not been selected so that whenever all the projects have been considered two cases can be are left. If the cheapest project fits in the budget allocation then the latter is not exhaustive and we link the branch to the $\bot$ leaf. Otherwise the budget allocation is exhaustive and the $\top$ leaf can be linked to it. This proves that the embedding is exhaustive. The fact that the constructed integrity constraint is a DNNF circuit of the suitable size is almost immediate given all the proofs we have already seen on that topic. □

This embedding is only defined for instances with a single resource and, unfortunately, the idea does not generalise (under some widely accepted computation complexity assumptions). The reason is that, when there are several resources, then there could be exponentially many "cheapest projects".

In the following we turn to another way to enforce exhaustiveness: asymmetric JA rules.

4.2 Asymmetric Judgment Aggregation Rules

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 familiar 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, if not all, established JA rules.

Definition 6 (Asymmetric Additive Rules) Let F be an additive JA rule associated with $f : (2^{\mathfrak{X}})^n \times Lit(\mathfrak{X}) \to \mathbb{R}_{\geq 0}$. Then its asymmetric counterpart F_{asy} is the rule

where for every integrity constraint Γ and every profile $\boldsymbol{J}$, we have:

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

Here ϵ is a small positive constant such that:

$$\epsilon < \frac{1}{|\mathfrak{X}|} \times \min \left\{ f(\boldsymbol{J}, \ell) \neq 0 \mid \boldsymbol{J} \in (2^{\mathfrak{X}})^n, \ell \in aug(J),\ \ell \text{ is positive} \right\}.$$

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 $\epsilon = \frac{1}{|\mathfrak{X}|+1}$ is a suitable choice for the three rules defined near the end of Section 2.2.

The class of asymmetrical additive rules is particularly interesting for us, as we can show that every rule in this class satisfies exhaustiveness.

Proposition 13 *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 always results in an increased 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 itself additive (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 (Aziz and Shah, 2020).[11]

5 Axiomatic Analysis

Axioms are means for encoding formal properties related to the normative adequacy of mechanisms for collective decision making (Thomson, 2001). Exhaustiveness is an example for such an axiom. In this section we investigate to what extent other important axioms proposed in the literature on PB are satisfied by JA rules when used for the purpose of PB. 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 (2019), generalising their definitions to allow for multiple resources and irresolute rules. Formally, for

[11] This rule selects projects based on their approval score in a greedy fashion, until the budget limit is reached. It actually corresponds to the asymmetric variant of the *ranked-agenda rule* (Lang et al., 2011).

	KEM		*SLAT*		*LEXIMAX*	
	sym	asym	sym	asym	sym	asym
Exhaustiveness	✗	✓	✗	✓	✗	✓
Limit Monotonicity	✗	✗	✗	✗	✗	✗
Discount Monotonicity	✓	✓	✓	✓	✓	✓
Splitting Monotonicity	✗	✓	✗	✓	✗	✓
Merging Monotonicity	✗	✗	✗	✗	✗	✗

Table 1 Axiomatic results: "sym" denotes the usual rule and "asym" its asymmetric counterpart.

a given PB axiom $\mathfrak{A}$, we say that the JA rule F *satisfies* $\mathfrak{A}$ *with respect to embedding* E if, for every PB instance I and profile $\boldsymbol{A}$, the PB rule mapping I and $\boldsymbol{A}$ to $\tau(F(E(I), \boldsymbol{A}))$ satisfies $\mathfrak{A}$.

Moreover, for a resolute rule F, the axioms of Talmon and Faliszewski (2019) are usually stated as "when one moves from an instance/profile pair $(I, \boldsymbol{A})$ 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, still in the context of PB, Baumeister et al. (2020) chose a different generalisation, involving existential rather than universal quantification.

The first axiom we study 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 (Elkind et al., 2017).

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 $\boldsymbol{A}$.

In the following, we show that this axiom is not satisfied by any of the JA rules of interest, even when there is only one resource.

Proposition 14 *None of the Kemeny, the Slater, or the leximax rules, or their asymmetric counterparts satisfy limit monotonicity with respect to any correct embedding.*

Proof Assume the embedding is correct. Consider two instances $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ and $I' = \langle \mathcal{R}, \boldsymbol{b}', \mathcal{P}, c \rangle$ such that $|\mathcal{R}| = 1$. There are three projects p_1, p_2, and p_3, and the

budget limits are $\boldsymbol{b} = (4)$ and $\boldsymbol{b}' = (5)$. The profile of interest $\boldsymbol{A}$ together with the instances I and I' are presented below, I on the left and I' on the right.

	p_1	p_2	p_3
Cost	3	2	1
A_1	✓	✓	✓
A_2	✓	✓	✓
A_3	✓	✓	✓
A_4	✓	✓	✗
A_5	✓	✗	✗
$\boldsymbol{b} = (4)$			

	p_1	p_2	p_3
Cost	3	2	1
A_1	✓	✓	✓
A_2	✓	✓	✓
A_3	✓	✓	✓
A_4	✓	✓	✗
A_5	✓	✗	✗
$b = (5)$			

We claim that on I, the Kemeny, the leximax rules, and their asymmetric counterpart would all return $\{\{p_1, p_3\}\}$. However, they would all return $\{\{p_1, p_2\}\}$ on I'. Project p_3 is thus a witness of the violation of limit monotonicity.

For the Slater rule and its asymmetric counterpart, consider the situation depicted below with two instances $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c\rangle$ and $I' = \langle \mathcal{R}, \boldsymbol{b}', \mathcal{P}, c\rangle$ involving three projects and a single resource, *i.e.*, $|\mathcal{R}| = 1$. The budget limits are $\boldsymbol{b} = (4)$ and $\boldsymbol{b}' = (6)$. We denote by $\boldsymbol{A}$ (on the left) and $\boldsymbol{A}'$ (on the right) the corresponding profiles, presented below.

	p_1	p_2	p_3
Cost	1	2	4
A_1	✓	✓	✓
$\boldsymbol{b} = (4)$			

	p_1	p_2	p_3
Cost	1	2	4
A_1	✓	✓	✓
$\boldsymbol{b}' = (6)$			

On I and $\boldsymbol{A}$, both the Slater rule and its asymmetric counterpart would return $\{\{p_1, p_2\}\}$. On the other hand, on I' and $\boldsymbol{A}$, both rules would return $\{\{p_1, p_2\}, \{p_1, p_3\}, \{p_2, p_3\}\}$. Since $\bigcap\{\{p_1, p_2\}, \{p_1, p_3\}, \{p_2, p_3\}\} = \emptyset$, both projects p_1 and p_2 are witnesses of the violation of limit monotonicity. □

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 $I' = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c'\rangle$ such that for some distinguished project $p^\star \in \mathcal{P}$, we have $c(p^\star) \geq c'(p^\star)$, and $c(p) = c'(p)$ for all $p \in \mathcal{P} \setminus \{p^\star\}$, it is the case that $p^\star \in \bigcap F(I, \boldsymbol{A})$ implies $p^\star \in \bigcap F(I', \boldsymbol{A})$ 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 $\Gamma, \Gamma' \in \mathcal{L}_{\mathfrak{X}}$ with $\mathfrak{J}(\Gamma) \subseteq \mathfrak{J}(\Gamma')$ and any profile $\boldsymbol{J}$, it is the case that $F(\Gamma', \boldsymbol{J}) \setminus F(\Gamma, \boldsymbol{J}) \subseteq \mathfrak{J}(\Gamma') \setminus \mathfrak{J}(\Gamma)$.

We obtain the following formal connection between the two axioms.

Lemma 15 *Every constraint-monotonic JA rule is discount-monotonic with respect to any correct embedding.*

Proof Let F be a JA rule that is constraint-monotonic. Let E be a correct embedding. Consider the instances $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ and $I' = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c' \rangle$, where a project $p \in \mathcal{P}$ became cheaper from I to I' as in Definition 8.

Let $\boldsymbol{A}$ be an arbitrary profile with $p \in \bigcap \tau(F(E(I), \boldsymbol{A}))$. We need to show that $p \in \bigcap \tau(F(E(I'), \boldsymbol{A}))$. Observe that $\mathcal{A}(I) \subseteq \mathcal{A}(I')$. Because E is correct, we also have $\mathfrak{J}(E(I)) \subseteq \mathfrak{J}(E(I'))$. Moreover, for every $\pi \in \mathcal{A}(I') \setminus \mathcal{A}(I)$, we have $p \in \pi$ as only $c'(p)$ changed in I'. Hence, for every outcome $J \in \mathfrak{J}(E(I')) \setminus \mathfrak{J}(E(I))$, we have $p \in \tau(J)$. From constraint-monotonicity, we have that for every profile $\boldsymbol{A}$, $F(E(I'), \boldsymbol{A}) \subseteq F(E(I), \boldsymbol{A}) \cup \mathfrak{J}(E(I')) \setminus \mathfrak{J}(E(I))$. Hence, for every $J \in F(E(I'), \boldsymbol{A})$, we have $p \in \tau(J)$. □

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

Proposition 16 *Every additive 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. □

Recall that several well-known JA rules are additive and thus subject to this result.

Corollary 17 *The Kemeny, Slater, and leximax rules as well as their asymmetric counterparts are all discount-monotonic with respect to 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 the following conditions are satisfied:

- The project p is replaced by P in the set of projects: $\mathcal{P}' = (\mathcal{P} \setminus \{p\}) \cup P$.
- The total cost of P is that of p: for all $p' \in P$, it is the case that $c'(p') \neq \mathbf{0}_d$ and $c'(P) = c(p)$.
- The cost of every other project is as in I: $c'(p') = c(p')$ for all projects $p' \in \mathcal{P}' \setminus P$.
- The project p is replaced by P in the ballots containing it: for every $i \in \mathcal{N}$ with $p \notin A_i$, we have $A'_i = A_i$; 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 AMRs and their asymmetric counterparts.

Proposition 18 *Every AMR as well as the asymmetric counterpart of every AMR are splitting-monotonic with respect to any correct embedding.*

Proof Let F be either an AMR or the asymmetric counterpart of an AMR, and let E be a correct and exhaustive embedding. Consider a PB instance $I = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}, c \rangle$ and a profile $\boldsymbol{A}$. Let $\boldsymbol{J}$ be the JA profile corresponding to $\boldsymbol{A}$. Let $I' = \langle \mathcal{R}, \boldsymbol{b}, \mathcal{P}', c' \rangle$ and $\boldsymbol{A}'$ be the instance and profile resulting from splitting a given project $p^\star \in \bigcap \tau(F(E(I), \boldsymbol{J}))$ into the set of projects $P^\star$. Let $\boldsymbol{J}'$ be the JA profiles corresponding to $\boldsymbol{A}'$.

Consider an outcome $J_1 \in F(E(I), \boldsymbol{J})$. Note that any outcome J that is admissible before and after the splitting, *i.e.*, any $J \in \mathfrak{J}(E(I)) \cap \mathfrak{J}(E(I'))$, cannot include either $p^\star$ or any project from $P^\star$. Since $p^\star \in \bigcap \tau(F(E(I), \boldsymbol{J}))$, this implies that J_1 has a higher total score than any $J \in \mathfrak{J}(E(I)) \cap \mathfrak{J}(E(I'))$.

Consider now an outcome $J'_1 = (J_1 \setminus \{p^\star\}) \cup \{p\}$ for some newly created project $p \in P^\star$. By definition of the new cost function, and since E is a correct embedding, we have $J'_1 \models E(I')$. J'_1 thus determines an admissible outcome for the constraint corresponding to I'. Based on the definition of $\boldsymbol{J}'$, it is clear that $n_\ell^{\boldsymbol{J}} = n_\ell^{\boldsymbol{J}'}$ for

every $\ell \in aug(J_1 \setminus \{x_{p^\star}\})$ and that $n^{\boldsymbol{J}}_{x^\star_p} = n^{\boldsymbol{J}'}_{x_p}$. Hence, because the internal score used by F only depends on the number of supporters, we know that J_1 and J'_1 have the same total score. This implies that J'_1 has a higher total score than any $J \in \mathfrak{J}(E(I)) \cap \mathfrak{J}(E(I'))$. Thus, $\mathfrak{J}(E(I)) \cap F(E(I'), \boldsymbol{J}') = \emptyset$. As for every newly admissible judgement $J' \in \mathfrak{J}(E(I')) \setminus \mathfrak{J}(E(I))$ we have $P^\star \cap \tau(J') \neq \emptyset$, every outcome returned by F would have a non-empty intersection with $P^\star$. □

Interestingly, this result provides a sufficient condition for PB rules to satisfy splitting monotonicity: any rule behaving as an AMR (with a suitable definition of AMR for PB) will satisfy it.

For the specific set of rules we study, we obtain the following corollary.

Corollary 19 *The Kemeny, Slater, and leximax rules as well as their asymmetric counterparts are all splitting-monotonic with respect to any correct embedding.*

Interestingly, when enforcing exhaustiveness through the embedding, this last result is no longer valid for symmetric rules.

Proposition 20 *None of the Kemeny, the Slater, or the leximax rules satisfy splitting monotonicity with respect to any correct and exhaustive embedding.*

Proof 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.

	p_1	p_2	p_3	p_4
Cost	2	2	1	1
A_1	✓	✗	✗	✗
A_2	✓	✗	✗	✗
A_3	✗	✓	✓	✓
$\boldsymbol{b} = (4)$				

	p_1	p_2^1	p_2^2	p_2^3	p_2^4	p_3	p_4
Cost	2	1/2	1/2	1/2	1/2	1	1
A_1	✓	✗	✗	✗	✗	✗	✗
A_2	✓	✗	✗	✗	✗	✗	✗
A_3	✗	✓	✓	✓	✓	✓	✓
$\boldsymbol{b} = (4)$							

Observe 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 $\boldsymbol{A}$. We leave the relevant computations to the reader, but the Kemeny, the Slater, and the leximax rules would all return $\{\{p_1, p_2\}\}$ on $(I, \boldsymbol{A})$ when used with a correct and exhaustive embedding. However, they would return $\{\{p_1, p_3, p_4\}\}$ on $(I', \boldsymbol{A}')$. Hence, p_2 is a witness of a violation of 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 proving this claim is provided next.

Proposition 21 *None of the Kemeny, the Slater, or the leximax rules, or their asymmetric counterparts satisfy merging monotonicity with respect to any correct embedding.*

Proof Consider the following pairs of instances and one-agent profiles: I and $\boldsymbol{A}$ on the left and I' and $\boldsymbol{A}'$ on the right. Both involve just one resource.

	p_1	p_2	p_3	p_4	p_5	p_6
Cost	2	2	1	1	1	1
A_1	✓	✓	✓	✓	✓	✓
$\boldsymbol{b} = (4)$						

	p_1	p_2	p'_3
Cost	2	2	4
A_1	✓	✓	✓
$\boldsymbol{b} = (4)$			

Observe that I' and $\boldsymbol{A}'$ are the result of merging $\{p_3, p_4, p_5, p_6\}$ into project p'_3, given I and $\boldsymbol{A}$. It is easy to see that the Kemeny, the Slater, and the leximax rules would all return $\{\{p_3, p_4, p_5, p_6\}\}$ on $(I, \boldsymbol{A})$ when used with a correct embedding. However, they would return $\{\{p_1, p_2\}\}$ on $(I', \boldsymbol{A}')$. Hence, p'_3 is a witness of a violation of merging monotonicity. Moreover, since the only agent approves of every projects, the same hold for the asymmetric counterpart of the rules. □

To conclude this section, 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 (Talmon and Faliszewski, 2019). 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. Our other axiomatic results are in line with those of Talmon and Faliszewski (2019). Overall, relative to the range of axioms we have considered here, we may summarise the situation by saying that JA rules perform similarly to other PB rules in normative terms.

6 Conclusion

We have proposed an efficient way of determining the outcome for 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 of JA rules for PB, we observed that a naïve way of embedding PB into JA leads to rules that violate the crucial exhaustiveness requirement of PB. We then suggested two ways of enforcing exhaustiveness: either through exhaustive embeddings or by using asymmetric JA rules. We also analysed several common JA rules and their asymmetric counterparts in view of basic monotonicity axioms for PB and found that the asymmetric rules fare better than the original rules.

In terms of future work, it would be interesting to study a wider range of PB axioms, to allow us to better differentiate between different JA rules. Indeed, for now, the Kemeny, Slater, and leximax rule cannot be distinguished based on the PB-specific axioms we studied (although they of course can be distinguished by means of axioms studied in the JA literature). A particularly exciting direction would be to investigate proportionality axioms such as those introduced by Aziz et al. (2018), Haret et al. (2020), and Chingoma et al. (2022). Interestingly, Chingoma et al. (2022) introduced proportionality requirements in the context of JA. However the rules satisfying those requirements are not additive rules and thus do not fit within our framework. It would therefore be interesting to investigate in more depth whether we can find additive rules that satisfy a relevant notion of proportionality. Another important direction would be to establish direct links between certain JA rules and their counterparts for PB. We, for instance, observed that the leximax rule is a refinement of the greedy approval rule—the most widely used PB rule in practice. Drawing similar parallels between other JA and PB rules is a natural next step for the research agenda initiated in this paper.

Beyond its immediate significance to the theory and practice of PB, we believe our work also, more generally, 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 as properties and constraints. Finding efficient ways of solving decision problems embedded into JA can be hard, but once identified, these methods allow for great flexibility.

Acknowledgements

We would like to thank the anonymous reviewers of the KR-2020 conference, the COMSOC-2021 workshop, and *Social Choice and Welfare* for their valuable feedback. We are also grateful for the fruitful discussions we had with Jan Maly when he visited us in Amsterdam in 2019.

References

Akian, M., R. Bapat, and S. Gaubert. 2006. Max-plus algebra, In *Handbook of Linear Algebra*, ed. Hogben, L., Chapter 35. London: Chapman and Hall.

Allegretti, G. and S. Antunes. 2014. The Lisbon participatory budget: Results and perspectives on an experience in slow but continuous transformation. *Field Actions Science Reports* .

Arora, S. and B. Barak. 2009. *Computational Complexity: A Modern Approach*. Cambridge: Cambridge University Press.

Aziz, H., B.E. Lee, and N. Talmon 2018. Proportionally representative participatory budgeting: Axioms and algorithms. In *Proceedings of the 17th*

International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 23–31.

Aziz, H. and N. Shah. 2020. Participatory budgeting: Models and approaches, *Pathways between Social Science and Computational Social Science: Theories, Methods and Interpretations*. Switzerland: Springer Nature.

Baumeister, D., L. Boes, and C. Laußmann 2022. Time-constrained participatory budgeting under uncertain project costs. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*.

Baumeister, D., L. Boes, and T. Seeger 2020. Irresolute approval-based budgeting. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1774–1776.

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

Benade, G., N. Itzhak, N. Shah, A.D. Procaccia, and Y.K. Gal. 2018. Efficiency and usability of participatory budgeting methods. Unpublished manuscript.

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

Bodlaender, H.L. and T. Kloks. 1996. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms 21*(2): 358–402 .

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

Cadoli, M., F.M. Donini, P. Liberatore, and M. Schaerf. 2002. Preprocessing of intractable problems. *Information and Computation 176*(2): 89–120 .

Chingoma, J., U. Endriss, and R. de Haan 2022. Simulating multiwinner voting rules in judgment aggregation. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

City of Amsterdam. 2022. Oost begroot. https://www.amsterdam.nl/stadsdelen/oost/oost-begroot/. Last accessed on 29 April 2022.

City of Paris. 2022. Paris Budget Participatif. https://budgetparticipatif.paris.fr/. Last accessed on 29 April 2022.

Conitzer, V., R. Freeman, and N. Shah 2017. Fair public decision making. In *Proceedings of the 18th ACM Conference on Economics and Computation (ACM-EC)*, pp. 629–646.

van Dalen, D. 2013. *Logic and Structure* (5th ed.). Springer.

Darwiche, A. and P. Marquis. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research* 17: 229–264 .

Dias, N. ed. 2018. *Hope for Democracy: 30 Years of Participatory Budgeting.* Vila Ruiva and Faro: Epopeia Records and Oficina.

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

Dietrich, F. and C. List. 2007. Arrow's Theorem in judgment aggregation. *Social Choice and Welfare 29*(1): 19–33 .

Downey, R.G. and M.R. Fellows. 2013. *Fundamentals of Parameterized Complexity.* London: Springer.

Elkind, E., P. Faliszewski, P. Skowron, and A. Slinko. 2017. Properties of multiwinner voting rules. *Social Choice and Welfare 48*(3): 599–632 .

Endriss, U. 2016. Judgment aggregation, In *Handbook of Computational Social Choice*, eds. Brandt, F., V. Conitzer, U. Endriss, J. Lang, and A.D. Procaccia, Chapter 17, 399–426. New York: Cambridge University Press.

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

Endriss, U., U. Grandi, R. de Haan, and J. Lang 2016. Succinctness of languages for judgment aggregation. In *Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR).*

Endriss, U., R. de Haan, J. Lang, and M. Slavkovik. 2020. The complexity landscape of outcome determination in judgment aggregation. *Journal of Artificial Intelligence Research* 69: 687–731 .

Everaere, P., S. Konieczny, and P. Marquis 2014. Counting votes for aggregating judgments. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1177–1184.

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

Fain, B., K. Munagala, and N. Shah 2018. Fair allocation of indivisible public goods. In *Proceedings of the 19th ACM Conference on Economics and Computation (ACM-EC)*, pp. 575–592.

Fairstein, R., R. Meir, D. Vilenchik, and K. Gal 2022. Welfare vs. representation in participatory budgeting. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Fluschnik, T., P. Skowron, M. Triphaus, and K. Wilker 2019. Fair knapsack. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*, Volume 33, pp. 1941–1948.

Freeman, R., D.M. Pennock, D. Peters, and J. Wortman Vaughan 2019. Truthful aggregation of budget proposals. In *Proceedings of the 20th ACM Conference on Economics and Computation (ACM-EC)*, pp. 751–752.

Garey, M.R. and D.S. Johnson. 1979. *Computers and Intractability*. New York: W.H. Freeman.

Goel, A., A.K. Krishnaswamy, S. Sakshuwong, and T. Aitamurto. 2019. Knapsack voting: Voting mechanisms for participatory budgeting. *ACM Transaction on Economics and Computation* 7(2): 8:1–8:27 .

Grandi, U. and U. Endriss 2011. Binary aggregation with integrity constraints. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 204–209.

Grossi, D. and G. Pigozzi. 2014. *Judgment Aggregation: A Primer*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

de Haan, R. 2018. Hunting for tractable languages for judgment aggregation. In *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 194–203.

Haret, A., M. Lackner, A. Pfandler, and J.P. Wallner 2020. Proportional belief merging. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 2822–2829.

Hershkowitz, D.E., A. Kahng, D. Peters, and A.D. Procaccia 2021. District-fair participatory budgeting. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*.

Jain, P., K. Sornat, and N. Talmon 2020. Participatory budgeting with project interactions. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*.

Jain, P., K. Sornat, N. Talmon, and M. Zehavi 2021. Participatory budgeting with project groups. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*.

Karp, R.M. 1972. Reducibility among combinatorial problems, In *Complexity of Computer Computations*, eds. Miller, R.E., J.W. Thatcher, and J.D. Bohlinger, 85–103. Boston: Springer.

Karp, R.M. and R.J. Lipton 1980. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 302–309.

Kimmig, A., G. van den Broeck, and L. de Raedt. 2017. Algebraic model counting. *Journal of Applied Logic* 22: 46–62 .

Lackner, M., J. Maly, and S. Rey 2021. Fairness in long-term participatory budgeting. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 299–305.

Lang, J., G. Pigozzi, M. Slavkovik, and L. van der Torre 2011. Judgment aggregation rules based on minimization. In *Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, pp. 238–246.

Lang, J. and M. Slavkovik 2013. Judgment aggregation rules and voting rules. In *Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT)*, pp. 230–243.

List, C. and P. Pettit. 2002. Aggregating sets of judgments: An impossibility result. *Economics & Philosophy 18*(1): 89–110 .

List, C. and C. Puppe. 2009. Judgment aggregation: A survey, In *Handbook of Rational and Social Choice*, eds. Anand, P., P. Pattanaik, and C. Puppe. Oxford University Press.

Los, M., Z. Christoff, and D. Grossi 2022. Proportional budget allocations: Towards a systematization. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*.

Lu, T. and C. Boutilier 2011. Budgeted social choice: From consensus to personalized decision making. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 280–286.

Marquis, P. 2015. Compile! In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 4112–4118.

Miller, M.K. and D. Osherson. 2009. Methods for distance-based judgment aggregation. *Social Choice and Welfare 32*(4): 575–601 .

Motamed, N., A. Soeteman, S. Rey, and U. Endriss 2022. Participatory budgeting with multiple resources. In *Proceedings of the 19th European Conference on Multiagent Systems (EUMAS)*.

Nehring, K. and M. Pivato. 2019. Majority rule in the absence of a majority. *Journal of Economic Theory* 183: 213–257 .

Patel, D., A. Khan, and A. Louis 2021. Group fairness for knapsack problems. In *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Peters, D., G. Pierczynski, and P. Skowron 2021. Proportional participatory budgeting with additive utilities. In *Proceedings of the 35th Annual Conference on Neural Information Processing Systems (NeurIPS)*.

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

Rey, S., U. Endriss, and R. de Haan 2021. Shortlisting rules and incentives in an end-to-end model for participatory budgeting. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 370–376.

Rey, S., U. Endriss, and R. de Haan 2020. Designing participatory budgeting mechanisms grounded in judgment aggregation. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR)*.

Robertson, N. and P.D. Seymour. 1983. Graph minors. i. excluding a forest. *Journal of Combinatorial Theory, Series B 35*(1): 39–61 .

Shah, A. ed. 2007. *Participatory Budgeting*. Washington: The World Bank.

Sreedurga, G., M.R. Bhardwaj, and Y. Narahari 2022. Maxmin participatory budgeting. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*.

Talmon, N. and P. Faliszewski 2019. A framework for approval-based budgeting methods. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*, pp. 2181–2188.

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

Wampler, B. 2012. Participatory budgeting: Core principles and key impacts. *Journal of Public Deliberation 8*(2): 12 .

Wampler, B., S. McNulty, and M. Touchton. 2021. *Participatory Budgeting in Global Perspective*. Oxford: Oxford University Press.