Detecting and taking Project Interactions into account in Participatory Budgeting

Martin Durand and Fanny Pascual

Sorbonne Université, LIP6, CNRS, 5 Place Jussieu, Paris, 75005, France.

*Corresponding author(s). E-mail(s): martin.durand@lip6.fr;
Contributing authors: fanny.pascual@lip6.fr;

Abstract

The aim of this paper is to introduce models and algorithms for the Participatory Budgeting problem when projects can interact with each other. In this problem, the objective is to select a set of projects that fits in a given budget. Voters express their preferences over the projects and the goal is then to find a consensus set of projects that does not exceed the budget. Our goal is to detect such interactions thanks to the preferences expressed by the voters. Through the projects selected by the voters, we detect positive and negative interactions between the projects by identifying projects that are consistently chosen together. In presence of project interactions, it is preferable to select projects that interact positively rather than negatively, all other things being equal. We introduce desirable properties that utility functions should have in presence of project interactions and we build a utility function which fulfills the desirable properties introduced. We then give axiomatic properties of aggregation rules, and we study three classical aggregation rules: the maximization of the sum of the utilities, of the product of the utilities, or of the minimal utility. We show that in the three cases the problems solved by these rules are NP-hard, and we propose a branch and bound algorithm to solve them. We conclude the paper by experiments.

Keywords: Computational social choice, Participatory Budgeting

1 Introduction

Participatory budgeting is a democratic process in which community members decide how to spend part of a public budget. Started in Porto Alegre, Brazil, in 1989, this process has spread to over 7,000 cities around the world, and has been used to decide budgets from states, cities, housing authorities, universities, schools, and other institutions[1]. The principle is the following one: the authorities of a given community (e.g. a city, or a university) decide to dedicate a budget l between projects proposed by the community members. Some community members (e.g. citizens, or students) propose projects, and write a proposal presenting their project and estimating its cost. All the community members are then asked to vote on the projects. There are several ways to collect voters' preferences. Due to its simplicity, the most widely used method is *approval voting*, in which voters are asked to approve or not each of the proposed projects. A variant of this method, called *knapsack voting* (Goel, Krishnaswamy, Sakshuwong, & Aitamurto, 2019), and that we will consider in this paper, asks the voters to approve projects up to the budget limit l: with knapsack voting, each voter is encouraged to give the set of projects that he or she would like to be selected, given the budget allocated. We start by reviewing existing work on participatory budgeting. Once the preferences of the voters have been expressed, the authorities use an algorithm which aggregates them and returns a set of projects (a bundle) of total cost at most l. In practice, e.g. in Warsaw, the projects are usually selected by decreasing number of votes. We start by reviewing existing work on participatory budgeting.

1.1 Related work

Participatory budgeting is a very active field in computational social choice and numerous other algorithms have been proposed (Aziz, Lee, & Talmon, 2018; Aziz & Shah, 2021; Peters, Pierczyński, & Skowron, 2021; Talmon & Faliszewski, 2019). Several social welfare functions have been considered. The aim is usually either to maximize the minimal utility of a voter (Sreedurga, Ratan Bhardwaj, & Narahari, 2022); to guarantee proportional representation to groups of voters with common interest (Aziz et al., 2018; Freeman, Pennock, Peters, & Wortman Vaughan, 2021; Peters et al., 2021), both aiming to return "fair" solutions; to maximize the sum of the utilities of the voters (utilitarian welfare); or to maximize the products of these utilities (Nash product) (Aziz & Shah, 2021; Benade, Nath, Procaccia, & Shah, 2021; Goel et al., 2019). In this paper we are interested in optimizing three of the most classical criteria: the maximization of the sum of the utilities, of the product of the utilities, or of the minimal utility of the voters.

There are two main ways to define the utility of a voter. The first way defines the utility of a voter as the number of funded projects that he or she approves (Jain, Sornat, & Talmon, 2020; Peters et al., 2021). The second way defines the utility of a voter as the total amount of money allocated to projects approved by the voter (Freeman et al., 2021; Goel et al., 2019; Talmon & Faliszewski, 2019). This second way of measuring the satisfaction of a voter is particularly relevant in the case of knapsack voting, where each voter can only approve a total budget of l: if a voter chooses to approve a project with a large cost at the expense of projects with smaller costs, it means that he or she prefers the large project to the smaller ones. We will consider this way to measure utilities.

[1] https://www.participatorybudgeting.org/

Project interactions. Project interactions (also called synergies between projects) have been little explored so far. In almost all the papers, it is assumed that the utility of a bundle (a set of projects) for a given voter is the sum of the utilities of these projects (number of projects or total cost of these projects, depending on the model considered). In a recent paper, Fairstein Roy and Kobi (2023) do an empirical study of several voting formats, without considering synergies. However, they say in their conclusion that "real voter utilities likely exhibit complementarities and externalities — a far cry from our utility proxies". Indeed, in practice, positive and negative synergies do exist. For example, two projects which are facilities that are planned to be built in the same location, or two projects which are very similar (e.g., two projects of playgrounds, or two skateboard parks) will have negative synergies: for a given voter, the utility of such two projects A and B will be smaller than the sum of the utilities of A and B. On the contrary, some projects are complementary and therefore have positive synergies. This is for example the case when a project aims to build a bicycle garage and another project aims to build a meeting place nearby. For a given voter, the utility of two projects A and B with positive synergies will be larger than the sum of the utilities of A and B. For two projects A and B which are independent, i.e., do not have positive neither negative synergies, the utility of the two projects A and B will be as usual the sum of the utilities of A and B.

To the best of our knowledge, there are only two papers which deal with projects interactions (Rey & Maly, 2023). Jain et al. (2020) introduce a model in which they assume that the synergies between the projects are already known and are defined as a partition P over the projects. The projects which belong to a same set of the partition either have a substitution effect (i.e. a negative interaction) or a complimentary effect (a negative interaction). The authors define a utility function f such that $f(i)$ is the utility that a voter v gets from a set of the partition P if i projects from this set and approved by v are in the returned bundle. If f is concave (i.e. $f(i+1) - f(i) < f(i) - f(i-1)$) then projects in the same set of P have negative interaction; if f is convex (i.e. $f(i+1) - f(i) > f(i) - f(i-1)$) then projects in the same set of P have positive interaction. The utility of a voter is the sum of the utilities it has over the different sets of the partition. This model is the first one to consider project interactions. In a subsequent paper, Jain, Talmon, and Bulteau (2021), assuming such an existing partition of the projects to interaction structures, take voter preferences to find such interaction structures (in their model, voters submit interaction structures, and the goal is to find an aggregated structure). Fairstein, Meir, and Gal (2021) also consider an underlying partition structure and ask the voters to give a partition of projects into groups of substitutes projects: in this setting only negative interactions are considered.

These papers are the first ones to consider and model project interaction. However, by partitioning the projects, their model cannot represent situations in which a project A can be both in positive interaction with a project B and in negative interaction with a project C, situation that we wish to take into account in this paper. Furthermore, the authors of the previous mentioned papers assume that such a partition is either known (Jain et al., 2020), or computed thanks to the partitions of the projects asked to the voters (Fairstein et al., 2021; Jain et al., 2021), which can be a fastidious and complicated task for the voters.

1.2 Our approach to interaction detection

Our aim is not only to take into account interactions between projects into the utilities of voters, but also to detect the interactions through the preferences of the voters. Detecting such interactions through the votes is not possible if, as in (Jain et al., 2020), the voters use approval voting to give their opinions on the projects. Indeed, with approval voting, a voter tends to evaluate each project individually and to select the projects that he or she finds interesting according to his or her own criteria. Thus, it is likely that a voter who would like to see a playground built near his or her home will support all the playgrounds projects, even if such projects interact negatively. On the contrary, with knapsack voting, each voter is asked how he or she would spend the money if he or she had the opportunity to decide. In that context, it is unlikely that a voter selects projects that interact negatively, and on the contrary it is likely that projects that interact positively will be chosen. We think the best way to get reliable preferences (which express synergies) is to ask the following question to the voters: "How would you spend the budget if you could make the decision ?". Assuming most voters follow this recommendation, the synergies should be estimated quite accurately.

Detecting synergies can be done through the ballots approved by the voters, by looking at the frequencies of occurrence of groups of projects among the projects approved by the same voter, compared to the "expected" frequencies of this group of projects. If, for example, two projects A and B are selected together very often, we will deduce that they probably are in positive synergy. On the contrary, if two objects are never selected together, the synergy will be negative. Thus, by comparing the frequency of appearance of these projects A, B together with the product of the frequency of A and the frequency of B, we deduce synergies from the voters' choices.

Example 1. *Consider a budget* $l = 9$ *and 5 projects* $\{A, \ldots, E\}$ *of costs* $(2, 3, 3, 1, 1)$ *(i.e. project A has cost 2, while project E has cost 1). Consider the following votes of* 4 *voters:* $\{A, B, D, E\}$, $\{A, B, C\}$,$\{C, E\}$,$\{A, B, D\}$. *Each project has been selected 2 or 3 times but projects A and B are always selected together, and projects C and D are never selected in a same ballot: we will deduce that projects A and B have a positive synergy while projects C and D have a negative synergy. Hence, whereas both bundles* $\{A, B, C, E\}$ *and* $\{A, B, C, D\}$ *are optimal for the utilitarian welfare, bundle* $\{A, B, C, E\}$ *is preferable because C and D have a negative synergy while C and E do not.*

One could argue that two projects will not be chosen by the same voter because of the budget limit and not because they have a negative interaction. First, if the sum of the costs of these two projects is larger than l, then these two projects will anyway not be chosen in the returned bundle. Second, we examined the costs distribution of projects from the 247 real-world instances of knapsack voting from Pabulib (Stolicki, Szufa, & Talmon, 2020). These instances mainly have "small projects": the vector of costs of projects of these instances is in average: $(0.56, 0.18, 0.09, 0.06, 0.04, 0.02, 0.02, 0.01, 0.01,$

0.01) – which means than 56% of the projects have a cost between 0 and 10% of the budget, 18% of the projects have a cost between 10 and 20% of the budget, and so forth. Additionally, on the same instances, the average (resp. median) total cost of the projects selected by a voter represents 66% (resp. 75%) of the budget. This means that a majority of voters could have selected one more project, and this among most of the unapproved projects. Therefore, the overall low cost of the projects paired with the budget left unused in the votes suggests that if two projects are rarely selected together, it is usually not because of theirs costs.

Note that taking account of synergies between the projects may be interesting even if all the projects have the same cost, as shown by the following example.

Example 2. *Let us consider a scenario with 12 voters, 8 projects of cost* 1 *and a budget of* 4. *Six voters select projects* 1 *and* 2 *plus a pair of projects in* $\{5, 6, 7, 8\}$, *different for each one. The six other voters select projects* 3 *and* 4 *plus a pair in* $\{5, 6, 7, 8\}$, *different for each one. Therefore, each project is selected exactly* 6 *times.*

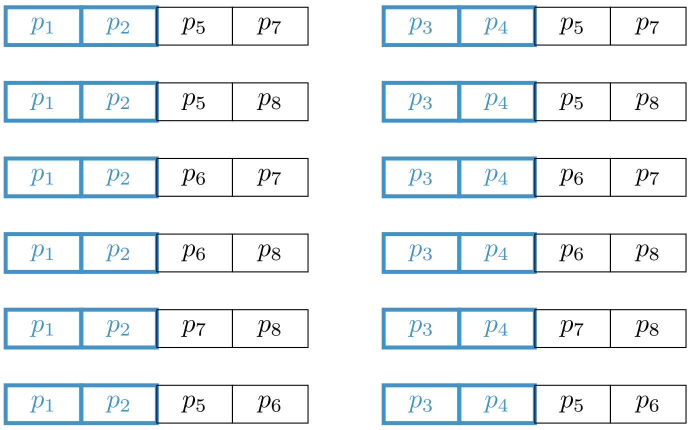

Fig. 1 Example with $l = 4$

Without synergies, each bundle of 4 *projects is optimal for the sum of the utilities. However, using synergies, we can detect that* $\{1, 2\}$ *and* $\{3, 4\}$ *are probably two strong pairs in comparison to the others. We can also see that the subset*$\{1, 2, 3, 4\}$ *is never chosen as a whole which may indicate an antisynergy of the complete subset.*

In the sequel, we will sometimes consider the k-additivity hypothesis, which means that there are synergies between groups of up to k projects. For example, with the 2-additivity hypothesis, we consider only interactions between pairs of projects, and not between more important groups of projects. In addition to the fact that it is realistic that synergies are important only for small values of k, considering this hypothesis will have repercussions on the complexity of our algorithms.

We conclude this introduction with an example showing that, in practice, positive (resp. negative) interactions may indeed be detected through the frequencies of co-occurrence of the projects in the same bundles.

Example 3. *By looking at real-world knapsack voting instances in the Pabulib library (Stolicki et al., 2020), and by considering that two projects interact positively (resp. negatively) when they are (resp. are not) chosen together, we identified several cases in which projects seem to interact positively or negatively*[2]*. For example, in Warsaw (poland_warszawa_2017_niskie-okecie.pb), two projects for the same neighbourhood, the first one being building a sport court and the second one building a playground, were chosen together less often than expected (given how often each one was individually approved). Our model says that they interact negatively, which makes sense, these projects being close to being susbtitutes. In another instance (poland_warszawa_2018_niskie-okecie.pb), two projects, the first one being building alleys in a park and the second one building public lightning in the same park, were consistently chosen together, which our model interpreted as a positive synergy. This also makes sense since these projects are clearly complementary.*

1.3 Overview of our results

We tackle the indivisible participatory budgeting problem, with knapsack voting, by considering that projects are not independent, but that there may have positive and negative synergies between them.

- In Section 3, we propose desirable properties for utility functions in presence of project interactions.
- In Section 4 we present a particular utility function, derived from Möbius transforms and denoted by u_M, that fulfills the axioms defined on the previous section.
- In Section 5, we study axiomatic properties of aggregation rules. We consider in particular three aggregation rules, which either maximize the sum of the utilities, the product of the utilities, or the minimal utility of the voters.
- In Section 6 we show that these rules solve NP-hard problems, and that synergies make the problem harder since it is NP-hard to maximize the sum of the utilities with unit size projects when there are synergies, whereas this problem can be solved easily without synergies. These results hold for utility function u_M but also for other very general synergy functions.
- In Section 7, we propose an exact branch and bound algorithm which can be used with any utility function, and we conclude with an experimental evaluation.

2 Preliminaries

We use the general framework for approval-based participatory budgeting proposed by Talmon and Faliszewski (2019). A *budgeting scenario* is a tuple $E = (A, V, c, l)$ where $A = \{a_1, \dots, a_n\}$ is a set of n *projects*, or items, and $c : A \to \mathbb{N}$ is a *cost function*: $c(a)$ is the cost of project $a \in A$ – abusing the notation, given a subset S, we denote by $c(S)$ the total cost of S: $c(S) = \sum_{a \in S} c(a)$. The budget limit is $l \in \mathbb{N}$. The set $V = \{v_1, \dots, v_v\}$ is a set of v voters. Each voter $v_i \in V$ gives a set of approved projects

[2]To be precise, to detect these interactions, we used the utility function u_M presented in Section 4, by considering 2-additivity hypothesis.

$A_i \subseteq A$, containing a set of projects that she approves of and such that $c(A_i) \leq l$. We denote by $\mathcal{E}^A$ the set of all possible budgeting scenarios having A as a set of projects.

A *budgeting method* r is a function taking a budgeting scenario $E = (A, V, c, l)$ and returning a *bundle* $B \subseteq A$ such that $c(B) \leq l$. We consider that a budgeting method always returns a unique bundle (we can use usual tie-breaking techniques to handle instances with several winning bundles). The winning bundle for a budgeting scenario E is denoted by $r(E)$. A project is *funded* if it is contained in the winning bundle B. Given a bundle B and a voter v_i with her approval set A_i, we denote by $B_i = A_i \cap B$ the set of projects common to A_i and B.

Utility functions. A *utility function* $u : 2^A \to \mathbb{R}_0^+$ is a set function which gives a value to each subset of items. A *linear utility function* is such that that the value of a bundle B is the sum of the utilities of its items: $u(B) = \sum_{a \in B} u(\{a\})$. The *overlap utility* function, introduced for the knapsack voting by Goel et al. (2019), considers that the utility of a bundle B is the sum of the costs of the projects in B: $f(A_i, B) = \sum_{a \in B_i} c(a)$

Satisfaction functions. A *satisfaction function* f is a function $f : 2^A \times 2^A \to \mathbb{R}$, which, given a voter $v_i \in V$ and a bundle $B \subseteq A$, returns the satisfaction that v_i gets from B. Given a selected bundle B and a utility function u, we will consider that the satisfaction of voter v_i from bundle B is the utility of $A_i \cap B$: $f(A_i, B) = uB_i$.

.

The utility function aims at associating to each possible bundle an evaluation of its quality. The satisfaction function indicates, given two sets of projects, the first one being the preference of a voter and the other being a potential solution, how satisfied the voter is given the solution.

In the sequel, we will consider generalizations of the overlap utility function that take into account potential projects interactions. Since these function may depend on the instance, we will denote the utility of the subset B_i as: $u(B_i, E)$.

Aggregating criterion. In order to obtain a solution satisfying the whole population, we study the three most classical aggregating methods: the sum ($\sum$), the product ($\prod$) and the minimum (min) of the satisfactions of the voters. We denote by $\alpha - r_u$ the budgeting method returning the α aggregation of the utility function u, where $\alpha \in \{\sum, \prod, \min\}$. This rule returns an optimal bundle of the associated maximization optimization problem, that we will call problem PB-Max–α–u (e.g. problem PB-Max–$\sum$–u consists in computing a bundle maximizing the sum of the utilities of the voters when the utility function used is u). These three aggregating concepts rely on different ideas of the collective satisfaction. The sum criterion maximizes the average satisfaction of a voter. The minimum tries to satisfy as much as possible the least satisfied voter – this is an egalitarian view. Finally the product stands in between the two previous criteria: the product is very penalized by the presence of very low utility values, however, it still takes into account the larger values. This last criterion has been the favourite of the voters in an experimental study (Rosenfeld & Talmon, 2021). These three criteria share several axiomatic and computational properties, as we will see in the following sections.

We will now discuss how to obtain satisfactory utility functions and how mathematical properties on such functions impact the budgeting methods.

3 Axioms for utility functions

In this section, we define desirable properties for utility functions in the presence of synergies.

The first property states that the utility of a single project should be proportional to its cost. This property is fulfilled by the *overlap utility* function (Goel et al., 2019). It is particularly meaningful in knapsack voting: since there is a budget constraint on the approval set of the voters, the approval of a project is done with full knowledge of its cost and the approval of a costly project is done at the expense of the budget for other projects.

Definition 1. *Given a budgeting scenario $E = (A, V, c, l)$, a utility function u^E : $2^A \times \mathcal{E}^A \to \mathbb{R}_0^+$ is* cost consistent *if there exists a constant k such that for each project a in A, we have $u(\{a\},) = k \cdot c(a)$.*

The factor k allows normalization. This property insures that the utility function follows the cost function for the sets containing only one project.

The following classical property ensures that the utility of a set does not decrease when the set grows. This ensures that we cannot decrease a voter satisfaction by adding a project that she selected.

Definition 2. *Given a budgeting scenario $E = (A, V, c, l)$, a utility function u^E is* super-set monotone *if for any subset X_{sub} and X such that $X_{sub} \subset X$, we have $u(X_{sub}, E) \leq u(X, E)$.*

Relaxing the *neutrality* principle (Brandt, Conitzer, Endriss, Lang, & Procaccia, 2016), the next property states that two similar projects should be treated equally. Given a set S, we denote by $S_{(a_i \leftrightarrow a_j)}$ the set obtained from S by swapping a_i and a_j: a_i (resp. a_j) belongs to $S_{(a_i \leftrightarrow a_j)}$ if and only a_j (resp. a_i) belongs to S, and each project $a_k \notin \{a_i, a_j\}$ belongs to $S_{(a_i \leftrightarrow a_j)}$ if and only if a_k belongs to S. We also denote by $E_{(a_i \leftrightarrow a_j)}$ the budgeting scenario obtained from E by swapping the approval of the projects a_i and a_j: a voter v_l approves a_i (resp. a_j) in $E^{(a_i \leftrightarrow a_j)}$ if and only if v_l approves a_j (resp. a_i) in E.

Definition 3. *Given a budgeting scenario $E = (A, V, c, l)$, and two projects a_i and a_j of A such that $c(a_i) = c(a_j)$, a utility function u^E is* cost-aware neutral *if $u(S, E) = u(S_{(a_i \leftrightarrow a_j)}, E_{(a_i \leftrightarrow a_j)})$.*

Note that this property is inspired by the *Processing Time Aware neutrality* property used in the collective schedules model (Durand & Pascual, 2022): this property ensures that two tasks of equal processing time are treated equally. We restrict our analysis to cost-aware neutral utility functions since no pair of projects with the same cost should be treated differently.

If a subset of item is consistently chosen as a whole, then the utility it brings should be higher than the sum of the utilities of the items. On the opposite side, if projects are never chosen together, then the utility of the whole subset should be lower than the sum of utilities of the items. The third axiom states that the more a subset appear together, the more its utility should increase, everything else being equal.

The next property, the *effect of positive synergies* ensures that the utility of subsets of projects that always appear together is larger than the sum of the utilities of its components.

Definition 4. *Given a budgeting scenario $E=(A,V,c,l)$, a utility function u^E fulfills the* effect of positive synergies *(resp.* strong effect of positive synergies*) property if, for each subset S in 2^A such that for each voter v_i we have either $S\subseteq A_i$ or $S\cap A_i=\emptyset$ and such that there exists $v_k\in V$ with $S\subseteq A_k$, then $u(S,E)\geq\sum_{a\in S}u(\{a\},E)$ (resp. $u(S,E)>\sum_{a\in S}u(\{a\},E)$).*

The next property ensures that the utility of subsets of projects that never appear together is smaller than (or equal to) the sum of the utilities of its components.

Definition 5. *Given a budgeting scenario $E=(A,V,c,l)$, a utility function u^E fulfills the* effect of negative synergies *(resp.* strong effect of negative synergies*) property if, for each subset S in 2^A such that for each voter $v_i\in V$ we have $|S\cap A_i|\leq 1$, then $u(S,E)\leq\sum_{a\in S}u(\{a\},E)$ (resp. $u(S,E)<\sum_{a\in S}u(\{a\},E)$).*

The next property states that the utility of a subset should increase with the number of appearances of the whole subset in the preferences of voters with respect to a solution in which the number of approvals of the items is the same but the items are not approved by the same voters.

Definition 6. *Let $E=(A,V,c,l)$ be a budgeting scenario, $S\subseteq A$ be a subset such that $c(S)\leq l$, and let v_i and v_j be two voters of V such that $S\subseteq(A_i\cup A_j)$, $A_i\cap A_j=\emptyset$, $S\not\subseteq A_i$, $S\not\subseteq A_j$, and $c(A_i\cup A_j\setminus S)\leq l$. Let $V_S=V\cup\{v_k,v_l\}\setminus\{v_i,v_j\}$, where v_k and v_l are two voters who are not in V and such that $A_k=S$ and $A_l=(A_i\cup A_j)\setminus S$. Let $E'=(A,V_S,c,l)$ be a budgeting scenario. A utility function u^E satisfies* regrouping monotonicity *if $u(S,E)<u(S,E')$.*

We can also imagine creating utility functions thanks to prior knowledge on the projects, however in such cases, it is possible that the last three properties are violated.

In the following section, we propose a utility function taking synergies into account, and that fulfills the properties that we have introduced in this section.

4 A utility function taking synergies into account

4.1 A function using Möbius transforms: u_M

Möbius transforms (Rota, 1964) are a classical tool for measuring synergies in sets of items. Given a utility function $u:2^A\rightarrow\mathbb{R}_0^+$, the Möbius transform of a subset S, denoted by $m(S)$, expresses the level of synergy between the items in S. For a set $S=\{a,b\}$ of two elements, and if $u(\emptyset)=0$, we have $m(S)=u(\{a,b\})-u(\{a\})-u(\{b\})$. More generally, the Möbius transform of a set S is calculated as follows:

$$m(S)=\sum_{C\subseteq S}(-1)^{|S\setminus\{C\}|}u(C)$$

The Möbius transform $m(S)$ expresses the level of synergy between the elements of the subset S. If it is negative, this indicates a negative interaction between the

C	$\emptyset$	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	$\{1,2,3\}$
$u(C)$	0	0.2	0.4	0.5	0.5	0.7	0.8	1

elements of S; if it is null, this indicates independence of the elements; and if it is positive, this indicates positive interaction between the elements.

Example 4. *Let us consider a utility function u over a set of items $\{1,2,3\}$. The utilities are as follows: Let us compute the Möbius transform of subset $\{1,2\}$*

$$\begin{aligned} m(\{1,2\}) &= (-1)^0 u(\{1,2\}) + (-1)u(\{1\}) + (-1)u(\{2\}) + (-1)^2 u(\emptyset) \\ m(\{1,2\}) &= u(\{1,2\}) - u(\{1\}) - u(\{2\}) + u(\emptyset) \\ m(\{1,2\}) &= -0.1 \end{aligned}$$

We find a negative Möbius transform, indicating a negative interaction between elements 1 *and* 2.

A utility function from Möbius transforms. It is not only possible to find the Möbius transforms from a utility function, it is also possible to build a utility function from the Möbius transforms thanks to the following expression (Rota, 1964):

$$u(S) = \sum_{C \subseteq S} m(C)$$

The utility of a subset S is then the sum of Möbius transforms of its elements – which is also the sum of their utilities – plus the Möbius transforms of the subsets included in S, representing their level of positive and negative synergies. Therefore, if we can measure the level of synergy of each subset, we can build a utility function.

We use a statistical approach in order to infer synergies from the preferences. Let $r(S,V)$ be the rate of occurrence of a subset S in the approval sets of voters in V (i.e. the ratio between the number of voters who selected all the projects of S, and the total number of voters). The expected rate of occurrence of a whole subset S if all of its elements were perfectly independent (ignoring possible cost constraints), would be $\prod_{a \in S} r(\{a\}, V)$, the product of the appearance rates of each the elements of S. We use $(r(S,V) - \prod_{a \in S} r(S,V))$ as a marker of synergy. If it is null, then the projects appear as independent in the preferences. If it is positive, then the subset appears more frequently than expected if the preferences were random, indicating a positive interaction. If it is negative, it indicates on the contrary a negative interaction.

We set $u(\emptyset) = m(\emptyset) = 0$ and, to insure cost consistency, we set $u(\{a\}) = m(\{a\}) = c(a)$. Since $(r(S,V) - \prod_{a \in S} r(S,V))$ has a range included in $[-1;1]$, we multiply this difference by the cost of the subset. We obtain: $m(S,E) = (r(S,V) - \prod_{a \in S} r(S,V)) \cdot c(S)$. We finally adapt this definition so that the utility function obtained from the

Möbius transforms fulfills super-set monotonicity:

$$m(S,E)=\begin{cases}0 & \text{if } S=\emptyset\\ c(a) & \text{if } S=\{a\}\\ \max\{\,(r(S,V)-\prod\limits_{a\in S} r(S,V))c(S), & \\ \qquad \max\limits_{a\in S}\left(-\sum\limits_{C\subset S, a\in C} m(C,E)\right)\} & \text{otherwise}\end{cases} \tag{1}$$

The intuition is the following one: $\sum\limits_{C\subset S, a\in C} m(C,E)$ is the sum of the Möbius transforms of subsets containing project a. By ensuring that the Möbius transform of S is larger than or equal to the opposite of this sum, we ensure that the utility of S is not smaller than the utility of $S\setminus\{a\}$.

Note that guaranteeing super-set monotonicity implies that we know the Möbius transforms value of smaller sets. The utility function u_M is as follows:

$$u_M(S,E)=\sum_{C\subseteq S} m(C,E) \tag{2}$$

4.2 Properties of u_M, and remarks on its computation

We use Equation 2 to determine the utility of a bundle with function u_M. Because of its recursive nature, we compute first, as a preprocessing step, the utility of singletons, then pairs, then triplets and so forth. Determining the utilities in this way costs up to 2^n (since there are 2^n subsets) times $v\times n$ operations (since determining the appearance rate of a subset costs $v\times n$ operations). This calculation is much faster with the k-additivity hypothesis, since the Möbius transform associated to any subset of size larger than k is then 0. Therefore, with such an hypothesis, we simply need to know the Möbius transforms of the subsets of size at most k: the preprocessing part is polynomial if k is a constant.

We now state that the utility function u_M fulfills all the desirable properties stated in Section 3. This is true even with the k-additivity assumption, for any value of k.

Proposition 1. *The utility function u_M fulfills* cost consistency*,* super-set monotonicity*, the* effect of positive synergies *property, the* effect of negative synergies *property,* regrouping monotonicity *and* cost aware neutrality*. It also fulfills the strong effect of positive synergies property if for each project a, there is at least one voter who does not select a.*

Proof.
- **Cost consistency.** As defined in equation 1, the Möbius transform of a single project is its cost. Since the utility of a single project is its Möbius transform, assuming the Möbius transform and the utility of the empty set is 0, the utility $u_M(\{a\},E)=c(a)$ for any project a. Therefore u_M fulfills cost consistency.
- **Super-set monotonicity.** As detailed earlier, the super-set monotonicity of the function u_M is insured by the definition of the Möbius transform. As a reminder, to fulfill super-set monotonicity, the function u_M has to verify the following property: $u_M(S,E)\geq u_M(S\setminus\{a\},E)$ for all $S\in 2^A\setminus\emptyset$ and all $a\in S$. Since,

by definition, we have $m(S,E) \geq -\sum_{C \subset S, a \in C} m(C,E)$ for all $a \in S$, it means that $\sum_{C \subseteq S, a \in C} m(C,E) \geq 0$ for all $a \in S$ and therefore $\sum_{C \subseteq S} m(C,E) \geq \sum_{C' \subseteq S \setminus \{a\}} m(C',E)$ for all $a \in S$. By definition of u_M, this means $u_M(S,E) \geq u_M(S \setminus \{a\}, E)$ for all $a \in S$. The utility function u_M fulfills super-set monotonicity.

- **Effect of positive synergies.** Let S be a subset of projects such that for any $a \in S$ and any $v_i \in V$, $a \in A_i \implies S \subseteq A_i$ and such that $\exists v_k \in V$ with $a \in A_k$. In other words, if a voter approves of one of the elements of S, she approves of all projects in S and such a voter exists in V. For such a subset, the value $r(S,V) - \prod_{a \in S} r(\{a\},V)$ is equal to $r(S,V) - r(S,V)^{|S|}$. Since the $r(S,V)$ value is larger than 0 and smaller than or equal to 1, the difference $r(S,V) - r(S,V)^{|S|}$ is positive or null. This means that $(r(S,V) - \prod_{a \in S} r(\{a\},V))c(S)$ is positive or null, this means that the Möbius transform of S is positive or null, $m(S,E) \geq 0$. The same remark can be said about all subset $C \subseteq S$ since all the projects of S are only selected together. Therefore, we have $\sum_{C \subseteq S, |C| \geq 2} m(C) \geq 0$. By definition, the Möbius transforms of the single projects are their cost, we then have: $\sum_{C \subseteq S} m(C) \geq \sum_{a \in S} c(a)$, and consequently: $u_M(S,E) \geq 0$. The utility function u_M fulfills the effect of positive synergies property. If we suppose that for each project a, there is at least one voter who does not select a, then for each subset S, there is at least one voter who does not select S. Then $r(S,V)$ is smaller than 1 and the difference $r(S,V) - r(S,V)^{|S|}$ is strictly positive. In this case, u_M fulfills the strong effect of positive synergies property.
- **Effect of negative synergies.** Let S be a subset of projects such that for any $a \in S$ and any $v_i \in V$, $a \in A_i \implies S \cap A_i = \{a\}$. In other words, if a voter selects an element a of S, then it is the only element of S she selects. For such a subset, the value $r(S,V) - \prod_{a \in S} r(\{a\},V)$ is negative or null, since S never appears but the element of S can appear individually. This is true for any subset $C \subseteq S$ such that $|C| \geq 2$. When summing the Möbius transforms all these subsets included in S,we will have the Möbius transforms of singleton that are positive and equal to the cost the project and then null or negative values. This means that the overall utility of S cannot be greater than the sum of the utility of its components. Therefore, u_M fulfills the effect of positive synergies property.
- **Regrouping monotonicity.** Let $E = (A, V, c, l)$ be a budgeting scenario and let $S \in 2^A$ be a subset of projects with $c(S) \leq l$. Let v_i and v_j be two voters in V such that $A_i \cap A_j = \emptyset$, $S \subseteq A_i \cup A_j$ and $c(A_i \cup A_j \setminus S) \leq l$. We consider voters v_k and v_l with $A_k = S$ and $A_l = A_i \cup A_j \setminus S$, and a set of preferences $V_S = V \cup \{v_k, v_l\} \setminus \{v_i, v_j\}$. Let $E' = (A, V_S, c, l)$ be a budgeting scenario. In V_S any subset $C \subseteq S$ appears at least as often than in V and any project appears as much in V_S than in V, therefore for any $C \subseteq S$, $r(C, V_S) \geq r(C,V)$, consequently $m(C,E) \geq m(C,E)$ and $u_M(C,E') \geq u_M(C,E)$. Since $u_M(C,E') \geq u_M(C,E)$, for all $C \subseteq S$ and $r(S,V_S) > r(S,V)$ we see that $m(S,E') > m(S,E)$ and therefore $u_M(S,E',>)u_M(S,E)$ from equation 2. Thus, u_M fulfills regrouping monotonicity.
- **Cost aware neutrality.** Given a budgeting scenario $E = (A,V,c,l)$, let $E_{(a_i \leftrightarrow a_j)} = (A, V_{(a_i \leftrightarrow a_j)}, c, l)$ be the budgeting scenario obtained from E by swapping the approval of two projects a_i and a_j such that $c(a_i) = c(a_j)$. For a given subset S, let $S_{(a_i \leftrightarrow a_j)}$ be the subset obtained from S by swapping a_i and a_j, i.e.

$S_{(a_i \leftrightarrow a_j)}$ contains the same projects than S except for a_i and a_j, if S contains a_i, $S^{(a_i \leftrightarrow a_j)}$ contains a_j and if S contains a_j, $S^{(a_i \leftrightarrow a_j)}$ contains a_i. Since $c(a_i) = c(a_j)$, and since for any subset C, $r(C, V) = r(C_{(a_i \leftrightarrow a_j)}, V_{(a_i \leftrightarrow a_j)})$, we can see that $m(C, E) = m(C_{(a_i \leftrightarrow a_j)}, E_{(a_i \leftrightarrow a_j)})$ and, because of equation 2 that $u_M(C, E) = u_M(C_{(a_i \leftrightarrow a_j)}, E_{(a_i \leftrightarrow a_j)})$. The utility function u_M fulfills cost aware neutrality.

□

5 Axioms for budgeting methods

In this section we discuss some axiomatic properties of the different aggregation rules, relying on the properties of the utility function used. We try, when it is possible, to have general results relying on the properties introduced in section 3 instead of on specific utility functions. We start with the *inclusion maximality* axiom (Talmon & Faliszewski, 2019), also known as *exhaustiveness* (Aziz et al., 2018). This axiom states that if a bundle B is a winning bundle according to a budgeting method r, then it is either exhaustive, in the sense that it is impossible to add a project without exceeding the budget limit, or any of its feasible superset is also a winning bundle.

Definition 7. *A budgeting method $\mathcal{R}$ satisfies* inclusion maximality *if for any budgeting scenario $E = (A, V, c, l)$ and each pair of feasible bundles B and B' such that $B' \subset B$, it holds that $B' \in \mathcal{R}(E) \implies B \in \mathcal{R}(E)$.*

Proposition 2. *If a utility function u fulfills super-set monotonicity, then the budgeting method $\alpha - r_u$, for $\alpha \in \{\sum, \prod, \min\}$ fulfills inclusion maximality.*

Proof. Let u be a utility function satisfying super-set monotonicity and $\alpha - r_u$a budgeting method maximizing either the sum, the product or the minimum over all the voters utilities. For any voter v_i, and for any pair of feasible bundles B and B' such that $B' \subset B$, we call B_i and B'_i the common subsets between A_i and B and A_i and B' respectively. Since $B' \subset B$, we have $B'_i \subseteq B_i$. Since both the sum, product and the minimum utility of the voters are non decreasing with the utility of individual voters, if B' is optimal for any of these rules, then B is also optimal. The budgeting method $\alpha - \mathcal{R}_u$ thus satisfies inclusion maximality. □

Note that when a budgeting method is resolute, meaning that it returns only one winning bundle, this axioms requires that the only winning bundle is exhaustive. This means that if we use tie-breaking mechanism to choose a solution among several optimal ones, they should select an exhaustive solution. Note that it can be easily obtained by adding projects greedily from an optimal solution that is not exhaustive.

The next two axioms focus on robustness, especially when projects have a composite structure (i.e. a large project can be divided into several small projects, or small projects merged into one large project).

Definition 8. *A budgeting method r satisfies* splitting monotonicity *if for every budgeting scenario $E = (A, V, c, l)$, for each $a_x \in r(E)$ and each budgeting scenario E' which is formed from E by splitting a_x into a set of projects A' such that $c(A') = c(a_x)$,* *and such that the voters which approve a_x in E approve all items of A' in E' and no other voters approve items of A', it holds that $r(E') \cap A' \neq \emptyset$.*

Proposition 3. *For $\alpha \in \{\sum, \prod, \min\}$, the budgeting method $\alpha - r_{u_M}$ fulfills splitting monotonicity.*

Proof. Let $E = (A, V, c, l)$ be a budgeting scenario, and let B be the bundle returned by $\alpha - r_u$ for E. Let a_x be a project in the bundle $\alpha - r_u(E)$. Let $E' = (A', V', c', l)$ be the budgeting scenario formed from E in which a_x is divided into a set X' of projects such that $c(X') = c(a_x)$. Voters in V' are the same than in V except that any voter approving project a_x in V approves all the projects of X' in V'. The bundle B maximizes the objective of the rule $\alpha - r_u$. Note that the utility of any subset that does not contain a_x is identical for E and E', and brings the same satisfaction to each voter: it therefore has the same quality regarding the aggregating criterion of $\alpha - r_u$for E and E'. Let B' be the bundle B in which a_x is replaced by all the projects in X'. Bundle B' is a feasible solution for E'. Any voter v'_i in V' has a corresponding voter v_i in V. We recall that B_i denotes the set of projects that are common between a bundle B and the approval set of a voter v_i. There are two cases:

- $X' \cap B'_i = \emptyset$: in this case, $u_M(B'_i, E') = u_M(B_i, E)$
- $X' \subseteq B'_i$: we have $c(B'_i) = c(B_i)$ and $r(B'_i, V) = r(B_i, V)$. Additionally, we have $\prod_{b' \in B'_i} r(b', V') \leq \prod_{b \in B_i} r(b, V)$, since the rates do not change but the number of projects is larger in B'_i than in B_i. This is also true for any subset $C \subseteq B'_i$ such that $X' \subseteq C$. Therefore, because of the super-set monotonicity property, we have $u_M(B'_i, E') \geq u_M(B_i, E)$.

Overall, B' is at least as good as any solution containing no element of X', meaning that either B' maximizes the rule criterion or a solution containing at least one project in X' does. Therefore there is a a in X' such that a is in $\alpha - r_u(E')$: the $\alpha - r_u$ rule fulfills splitting monotonicity. □

Definition 9. *A budgeting method r satisfies* merging monotonicity *if for each budgeting scenario $E = (A, V, c, l)$, and for each $A' \subseteq r(E)$ such that for each $v_i \in V$ we either have $A_i \cap A' = \emptyset$ or $A' \subseteq A_i$ – i.e. a voter approves either all projects from A' or none – it holds that $a \in r(E')$ for $E' = (A \setminus \{A'\} \cup \{a\}, V', c', l)$, $c'(a) = c(A')$, and each voter $v_i \in V$ for which $A' \subseteq A_i$ in E approves a in E', and no other voter approves a.*

Proposition 4. *Let $\alpha \in \{\sum, \prod, \min\}$. If a utility function u fulfills the strong effect of positive synergy property and cost consistency, then the budgeting method $\alpha - r_u$ does not fulfill merging monotonicity.*

Proof. • Case where $\alpha = \Sigma$. Let T be an even positive integer. Let us consider a budgeting scenario $E = (A, V, c, l)$ with $A = x_1, x_2, y$, $c(x_1) = c(x_2) = T/2$, $c(y) = T$ and $l = T$. There are two types of voters in V. There are v_1 voters of the first type, and each one of them approves x_1 and x_2. There are v_2 voters of type 2, and they all approve y as shown in Figure 2. By cost consistency, we know that there exists a constant k such that $u(x_1, E) = u(x_2, E) = kT/2$ and $u(y, E) = kT$. By strong effect of positive synergies, we have $u(\{x_1, x_2\}, E) > u(\{x_1\}, E) + u(\{x_2\}, E)$ and consequently $u(\{x_1, x_2\}, E) > kT$. Let $\epsilon = u(\{x_1, x_2\}, E) - kT > 0$. The bundle $\{x_1, x_2\}$ has a total utility of $v_1(kT+\epsilon)$, the bundle $\{y\}$ has a utility of v_2kT. If $v_1kT - v_2kT + v_1\epsilon > 0$, then $\{x_1, x_2\}$ is the best bundle.

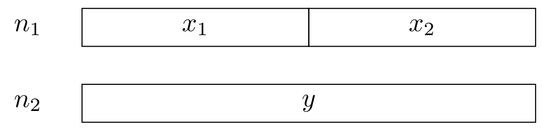

Fig. 2 First budgeting scenario E

We now consider $E' = (A', V', c', l)$ another budgeting scenario such that $A' = \{x, y\}$, $c'(x) = c(x_1) + c(x_2) = c'(y) = c(y) = T$. In V' we create v_1 voters approving x and v_2 voters approving y. Note that the budgeting scenario E' is similar to E except that the projects x_1 and x_2 have merged in a project of size T. Because of cost consistency, we have $u(\{x\}, E') = u(\{y\}, E') = kT$. Therefore the bundle $\{x\}$ has a total utility of v_1kT and the bundle $\{y\}$ still has a utility of v_2kT. If $v_1 < v_2$, $\{y\}$ is the winning bundle.

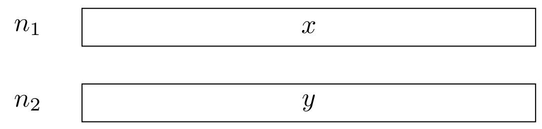

Fig. 3 Second budgeting scenario E'

By setting $v_1 = \lceil 2kT/\epsilon \rceil$ and $v_2 = v_1 + 1$, $\{x_1, x_2\}$ is the winning bundle for E and $\{y\}$ is the winning bundle for E', giving us an instance for which the $\sum$-r_u rule does not fulfill merging monotonicity.

• Case where $\alpha \in \{\prod, \min\}$. Let us consider a budgeting scenario $E = (A, V, c, l)$ with $A = \{x_1, x_2, x_3, x_4, y\}$, $c(x_1) = c(x_2) = c(x_3) = c(x_4) = (T-2)/4$, $c(y) = T/2 + 1$ with T an even integer and $l = T$. There are two voters in V: the first one approves of x_1, x_2, x_3 and x_4, the second one approves of y.

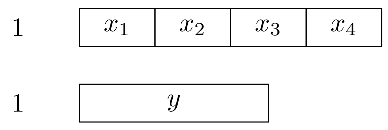

Fig. 4 First budgeting scenario E

When maximizing either the min utility or the product, for any utility function u fulfilling cost consistency and the strong effect of positive synergies, the winning bundle will be y plus two projects x_i and x_j. Let us assume, without loss of generality that the projects x_1 and x_2 are part of the winning bundle. Let $E' = (A', v', c', l)$ be a budgeting scenario formed from E in which projects x_1 and x_2 are merged into one project X of cost (and therefore utility) $T/2 - 1$.

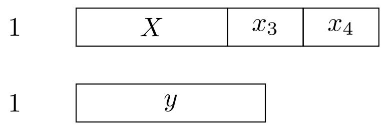

Fig. 5 Second budgeting scenario E'

The utilities of x_3 and x_4 are still $(T-2)/4$. By strong superadditivity of groups, the utility of $\{x_3, x_4\}$ is strictly larger than $2(T-2)/4$ and strictly larger than the utility of X consequently. Therefore the winning bundle for E' is $\{y, x_3, x_4\}$. Since X is not in this bundle, merging monotonicity is not fulfilled. □

The next axiom states that if the cost of a funded project decreases, it is still guaranteed to be funded. It is easy to see that this axiom is not compatible with the cost consistency property.

Definition 10. *A budgeting method r satisfies* discount monotonicity *if for each budgeting scenario $E=(A,V,c,l)$ and each item $b \in r(E)$, it holds that $b \in r(E')$ for $E'=(A,V,c',l)$ where for each item $a \neq b$, $c'(a)=c(a)$ and $c'(b)=c(b)-1$.*

Proposition 5. *Let $\alpha \in \{\sum, \prod, \min\}$. If a utility function u fulfills cost consistency, then the budgeting method $\alpha{-}r_u$ does not fulfill discount monotonicity.*

Proof. Let us consider a budgeting scenario $E=(A,V,c,l)$ with $A=x_1,x_2,y$, such that $c(x_1)=4, c(x_2)=3$, $c(y)=4$ and $l=8$. There are two voters in V: the first one approves x_1 and x_2, and the second one approves y. When maximizing either the $\sum$, the min or the $\prod$ of utilities, for any utility function u fulfilling cost consistency, the winning bundle will be y plus project x_1. Let $E'=(A,V,c',l)$ be a budgeting scenario formed from E in which project x_1 now has a cost of 2 instead of 4. The winning bundle is now $\{y, x_2\}$. The cost of project x_1 was reduced and it was removed from the winning bundle, therefore discount monotonicity is not fulfilled. □

This last axiom states that any funded project in a winning bundle is still funded when the budget limit increases.

Definition 11. *A budgeting method r fulfills* limit monotonicity *if for each pair of budgeting scenarios $E=(A,V,c,l)$ and $E'=(A,V,c,l+1)$ with no item which costs exactly $l+1$, it holds that $a \in r(E) \implies a \in r(E')$.*

Proposition 6. *Let $\alpha \in \{\sum, \prod, \min\}$. If a utility function u fulfills cost consistency, then the budgeting method $\alpha{-}r_u$does not fulfill limit monotonicity.*

Proof. • Case where $\alpha=\sum$: Let us consider a budgeting scenario $E=(A,V,c,l)$ with $A=x_1,x_2,x_3$, $c(x_1)=2, c(x_2)=5$, $c(x_3)=6$ and $l=6$. There are three voters in V: the first one approves of x_1, the second one approves of x_2 and the third one approves of x_3. When maximizing the sum of utilities, for any utility function u fulfilling cost consistency, $\{x_3\}$ will be the winning bundle. Let $E'=(A,V,c,l')$ be a budgeting scenario formed from E but such that the budget limit l' is now 7 instead of 6. The winning bundle is now $\{x_1, x_2\}$. The budget limit was increased and project x_3 was removed from the winning bundle, therefore limit monotonicity is not fulfilled.

• Case where $\alpha \in \{\min, \prod\}$: Let us consider a budgeting scenario $E = (A, V, c, l)$ with $A = x_1, x_2, x_3$, $c(x_1) = 1, c(x_2) = 2$, $c(x_3) = 3$ and $l = 4$. There are two voters in V: the first one approves of x_1 and x_2, the second one approves of x_3. When maximizing either the min or the product of utilities, for any utility function u fulfilling cost consistency, $\{x_1, x_3\}$ will be the winning bundle. Let $E' = (A, v, c, l')$ be a budgeting scenario formed from E but such that the budget limit l' is now 5 instead of 4. The winning bundle is now $\{x_2, x_3\}$. The budget limit was increased and project x_1 was removed from the winning bundle, therefore limit monotonicity is not fulfilled. □

From Proposition 1 and propositions from Section 5, we get the following corollary.

Corollary 1. *The rules $\alpha - r_u$ for $\alpha \in \{\sum, \min, \prod\}$ and $u = u_M$ fulfill* inclusion maximality *and* splitting monotonicity. *They do not fulfill* merging monotonicity, discount monotonicity *and* limit monotonicity.

6 Complexity

We show in this section that, for each $\alpha \in \{\sum, \prod, \min\}$, problem PB-MAX–$\alpha$–$u$ is NP-hard when there are synergies, and this even if all the projects have unitary cost and for a very general class of utility functions. This shows that synergies add complexity, since problem PB-MAX–$\sum$–u is polynomially solvable when projects have the same cost and without synergies (i.e. when the function u is linear). Indeed, without synergies and with unitary size projects, selecting the projects by decreasing number of votes maximizes the sum of the utilities of the voters. Let us now show that, with synergies, this problem is NP-hard even with very general utility functions. We start by proving a preliminary result for the CLIQUE problem.

Lemma 1. *The* CLIQUE *problem is strongly NP-complete even if it is restricted to graphs G in which $d_{\max} < \sqrt{m}$, where m is the number of edges and $d_{\max}$ is the maximum degree of a vertex of G.*

Proof. The CLIQUE problem is the following one. We are given an undirected graph $G = (V, E)$, with V the set of n vertices and E the set of m edges. We denote by d_i the degree of a vertex i, and by $d_{\max}$ the maximum degree of any vertex in V. We are also given an integer K. The question is: does there exist a clique of size K in G?

This problem is known to be strongly NP-complete (Garey & Johnson, 1979), and we now show that it is still strongly NP-complete when the graph G is such that $\sqrt{m} > d_{\max}$.

We reduce the CLIQUE problem in any graph into the CLIQUE problem in a graph where $\sqrt{m} > d_{\max}$. Let G and K be an instance of the CLIQUE problem without any constraint on m and $d_{\max}$. We first transform graph G into a graph G', as follows. Graph G' is built from graph G by "copying" G m times, obtaining m connected components: for any vertex v_i in V, we create $m + 1$ vertices $\{v_{i,0}, v_{i,1} \cdots v_{i,m}\}$ in V', and for each edge (v_i, v_j) in E, we create $m+1$ edges $\{(v_{i,0}, v_{j,0}) \cdots (v_{i,m}, v_{j,m})\}$ in E'. We denote by $d^G_{\max}$ (resp. $d^{G'}_{\max}$) the maximum degree of a vertex of G (resp. G'), and by m (resp. m') the number of edges in G (resp. G'). We have $d^{G'}_{\max} = d^G_{\max}$ and $m' = (m+1)m$. Since $m \geq d^G_{\max}$ and $d^{G'}_{\max} = d^G_{\max}$, we have: $m' = m(m+1) \geq d^G_{\max}(d^G_{\max} + 1)$.

Therefore, $\sqrt{m'} > d^{G'}_{\max}$. We now show that there is a clique of size K in G' if and only if there is a clique of size K in G.

- Let us first assume that there is a clique $C = \{v_1, v_2 \cdots v_K\}$ of size K in G. In that case, the set $C' = \{v_{1,0}, v_{2,0} \cdots v_{K,0}\}$ is clique of size K in G' since for any edge connecting two vertices v_i and v_j in G we created an edge connecting $v_{i,0}$ and $v_{j,0}$ in E'.
- Let us now assume that there exists a clique of size K in G'. Such a clique can only be formed by a set of vertices $\{v_{1,i}, v_{2,i} \cdots v_{K,i}\}$ with a fixed i since no edges in E' connect two vertices $v_{k,i}$ and $v_{l,j}$ with $i \neq j$ by construction. If such a clique exists, then the subset $C = \{v_1, v_2 \cdots v_K\}$ in G is a clique as well since if an edge $(v_{k,i}, v_{l,i})$ exists in E', an edge (v_k, v_l) exists in E. Therefore C is a clique of size K in G and the answer to the Clique problem is yes.

Since our problem is in NP, and that there exists a polynomial time reduction of the strongly NP-complete Clique problem into the Clique problem when $\sqrt{m} > d_{\max}$, we conclude that the Clique problem is strongly NP-complete even when $\sqrt{m} > d_{\max}$. □

Proposition 7. *Problem PB-*Max*–*$\sum$*–*u *is strongly NP-hard, even if all the projects have unit costs. This is true if* $u = u_M$, *as well as for any utility function* u *such that the utility of two projects that have been selected together by at least one voter is strictly larger than the utility of two projects approved by the same number of voters but that have never been selected together by a same voter.*

Proof. The decision version of our problem, that we will denote PB-Max–$\sum$–u-dec, is the following one. We are given a number $R \in \mathbb{Z}$ and a budgeting scenario $E = (A, V, c, l)$ with c a cost function such that the cost of each project of A is exactly one. We consider that the utility function u is such that the utility of a pair of projects selected at least once together is strictly larger than the utility of any other pair of projects that have been selected the same number of times but that have never been selected together. The set A is a set of v voters $\{v_1, \ldots, v_v\}$, having each one approved up to l projects of A. The question is: does there exist a set $B \subset A$ of up to l projects such that the utility of B, $\sum_{v_i \in V} u(B_i, E)$, is at least R ?

We reduce the strongly NP-complete problem Clique to this problem. We will assume that the instance of the Clique problem is a graph such that $\sqrt{m} > d_{\max}$ (the Clique problem is still NP-complete in this case, as shown by Lemma 1). The Clique problem is as follows: given an graph $G = (V, E)$, such that $\sqrt{m} > d_{\max}$, and an integer K, the question is: does there exist a clique of size K?

Given an instance (G, K) of the Clique problem, we create an instance of PB-Max–$\sum$–u-dec as follows. We first transform graph G into a graph G', as follows. We start by setting $G' = G$, and we assume that the $|V|$ vertices of G' are labelled $\{1, \ldots, |V|\}$. For each vertex i of degree $d_i < d_{\max}$, we add $(d_{\max} - d_i)$ new neighbor vertices, denoted by $Dummy(i, 1), \ldots Dummy(i, d_{\max} - d_i)$. By doing this, the vertices of $\{1, \ldots, |V|\}$ are all of degree $d_{\max}$. Let $G' = (V', E')$ be the graph obtained. Each newly added vertex $Dummy(i, j)$ is of degree 1 in G'. Therefore, the number of newly added vertices in G' is $n_{dummy} = \sum_{i=1}^{|V|} d_{\max} - d_i = |V| d_{\max} - 2|E|$, and the number of newly added edges is the same value. We label the newly added vertices (if any) as $\{|V|+1, \ldots, |V|+n_{dummy}\}$.

We now create from G' a set of projects A as follows. To each vertex $i \in \{1, \ldots, |V'|\}$ we create a corresponding project P_i of cost 1: there are thus $|V|$ projects corresponding each one to one vertex of V, and n_{dummy} projects corresponding each one to one dummy vertex. We create a set $\mathcal{V}$ of $m_V = |E'| + (d_{\max}-1)n_{dummy}$ voters. To each edge $\{x, y\} \in E'$, we create a voter which approves exactly two projects: projects P_x and P_y, corresponding to vertices x and y. For each dummy vertex, we create $(d_{\max}-1)$ voters that approve only the project corresponding to the dummy vertex.

We fix the maximum budget to $l = K$ (since all the projects have a unitary cost, this means that up to K projects can be selected). The value of R, the target utility, depends of the synergy function. We observe that in our instance of PB-MAX–$\sum$–u-dec each project is chosen by the same number of voters $(d_{\max}-1)$. Let $u_{together}$ be the utility that a voter obtains for a set of two projects which have both been chosen by the voter. The sequel of the proof works for all utility function such that $u_{together} > 2$. This is in particular true for u_M, as shown by the following fact.

Fact 1: If the utility function is u_M, then $u_{together} > 2$.
Proof of the fact: Let us show that the utility function u_M count positive interactions for pairs of projects corresponding to vertices connected by an edge in G'. For function u_M, we have:

$$m(\{x, y\}, \mathcal{V}) \geq r(\{x, y\}, \mathcal{V}) - r(\{x\}, \mathcal{V})r(\{y\}, \mathcal{V})$$
$$m(\{x, y\}, \mathcal{V}) \geq \frac{1}{m_V} - \frac{(d_{\max})^2}{(m_V)^2} = \frac{1}{m_V}\left(1 - \frac{(d_{\max})^2}{m_V}\right)$$

Furthermore:

$$m_V = |E| + \sum_{i=1}^{|V|} d_{\max}(d_{\max} - d_i) = |E| + |V|(d_{\max})^2 - \sum_{i=1}^{|V|} d_{\max} d_i$$

Since $d_i d_{\max} \leq (d_{\max})^2$, we get $\sum_{i=1}^{|V|} d_i d_{\max} \leq |V|(d_{max})^2$, and thus $m_V \geq |E|$. Since $\sqrt{|E|} > d_{max}$, $m_V > (d_{\max})^2$ and the Möbius transform of the pair is larger than 0, meaning that the utility of the subset $\{x, y\}$ is larger than the sum of their costs: $u_{together} > 2$.

Let us now show that it possible to select a set of at most K projects of total utility larger than or equal to $R = Kd_{max} + \frac{K(K-1)}{2}(u_{together} - 2)$ if and only if there is a clique of size K in G.

• Let us first assume that there is a clique C of size K in G. Let S^{clique} be the set of the K projects which correspond to the K vertices of C. Note that for each couple of projects x and y of S^{clique}, exactly one voter has approved both x and y. The utility of S^{clique} is thus $u_{together}$ for each of these $K(K-1)/2$ voters. Note also that each project has been selected by exactly d_{max} voters. Therefore, for the $Kd_{max} - 2 \times K(K-1)/2$ voters who approve exactly one project of S^{clique}, the utility of S^{clique} is 1. The other voters do not approve any project of S^{clique} and have a utility of 0. The total utility of S^{clique} is thus $1 \times (Kd_{max} - 2 \times \frac{K(K-1)}{2}) + u_{together}\frac{K(K-1)}{2} = Kd_{max} + \frac{K(K-1)}{2}(u_{together} - 2) = R$. The answer to our problem is thus 'yes'.

• Let us now assume that there is a set C of at most K projects of total utility at least $R = Kd_{max} + \frac{K(K-1)}{2}(u_{together} - 2)$. Note that each project is approved by exactly d_{max} voters. The utility of C for a given voter is 0 if the voter does not select any project of C, 1 if it selects exactly one project, and $u_{together} > 2$ if it approves exactly two projects (recall that a voter approves at most 2 projects). The utility of C is thus equal to n_1, the number of voters who approve exactly one project of C, plus $n_2 \times u_{together}$, where n_2 is the number of voters who approve exactly two projects of C. We have $R = Kd_{max} - 2\frac{K(K-1)}{2} + \frac{K(K-1)}{2}u_{together} \leq n_1 + n_2 u_{together}$, and $n_1 + 2n_2 \leq Kd_{max}$ (since $n_1 + 2n_2$ is equal to the total number of votes for projects of C and C is of size at most K). Therefore, $n_1 = Kd_{max} - 2\frac{K(K-1)}{2}$, and $n_2 = \frac{K(K-1)}{2}$. This means that there are exactly K projects in C and that for each couple of projects of C, there is a voter who approves both projects (recall that there is exactly one voter by edge in G'). Therefore, there exists a clique of size K in G', and thus a clique of size K in G. There exists a polynomial time reduction of the strongly NP-complete CLIQUE problem into the decision version of our problem: PB-MAX–$\sum$–u-dec is thus strongly NP-hard. □

The next result extends the result from Sreedurga et al. (2022), which proves that the maxmin participatory budgeting problem is strongly NP-hard for approval voting when the utility function is the sum of the costs of the funded approved projects. We generalize this result by proving that this is true for both the maxmin and the product of utilities and we show that we only need a very weak condition on the utility function for this to be true. Additionally, it holds for knapsack voting, which is more specific that approval voting. We also show that the problem is hard to approximate. We first prove the following lemma.

Lemma 2. *The* SET COVER *problem is strongly NP-complete even when restricted to instances in which the number of subsets containing the same element is bounded by K, the size of a feasible solution.*

Proof. The SET COVER problem is the following one: we are given a set $\mathcal{U}$ of n elements, called the universe, and a collection S of m sets whose union is $\mathcal{U}$. Given an integer $K < m$, the question is: does there exist a set $\mathcal{S}$ of elements in S, such that $\cup_{s \in \mathcal{S}} = \mathcal{U}$ and $|\mathcal{S}| \leq K$?

From an instance $\mathcal{U}$,S,K, we create a new instance $\mathcal{U}'$,S',K'. In this new instance, we create m dummy elements $\{x^1_{dummy} \cdots x^m_{dummy}\}$ and m dummy sets $\{s^1_{dummy} \cdots s^m_{dummy}\}$ such that s^i_{dummy} contains x^i_{dummy}. We then have $n' = n + m$ and $\mathcal{U}' = \mathcal{U} \cup \{x^1_{dummy}, \cdots, x^m_{dummy}\}$, $m' = 2m$ and $S' = S \cup \{s^1_{dummy} \cdots s^m_{dummy}\}$ and $K' = K + m$. In this new instance, it is easy to see that each element is contained by at most $m < K'$ sets.

We now prove that there exists a set cover of $\mathcal{U}'$ with subsets of S' and of size K' at most if and only if there exists a set cover of $\mathcal{U}$ with subsets of S and of size K at most.

- Let us first suppose that there exist a cover C' of $\mathcal{U}'$ with subsets of S' and of size K' at most. This cover necessarily contains the m dummy sets since these sets are the only one containing the m dummy vertex. The $K' - m < K$ other sets of the cover form a feasible cover of the n elements of $\mathcal{U}$ and all of these sets are in S.
- Now, we suppose that that there exist a cover C of $\mathcal{U}$ with subsets of S and of size K at most. The elements of $\mathcal{U}'$ that are not covered by C are the dummy elements. By adding the m dummy sets of S' to C, we obtain a cover C' covering all the elements from $\mathcal{U}$ plus the m dummy elements and of size of $|C|+m$. Since $|C| \leq K$, we have $|C|+m \leq K+m = K'$, we therefore have a feasible cover of $\mathcal{U}'$.

There exist a polynomial time reduction between any instance of Set cover to a version of the Set cover problem in which the number of sets containing the same element is bounded by K. Therefore the Set cover is still strongly NP-complete in that case. □

Proposition 8. *Problems PB-*Max*–*min*–u and PB-*Max*–$\prod$–u are strongly NP-hard for any utility function u such that $u(\emptyset, E) = 0$ and $u(S,E) > 0$ for each $S \neq \emptyset$. For any $\delta > 1$, there is no polynomial time δ-approximate algorithm if $P \neq NP$.*

Proof. The decision version of our problem is the following one. We are given a real number R and a budgeting scenario $E = (A, V, c, l)$ with A a set of n projects and c a cost function such that the cost of each project is exactly one. We consider a utility function u such that $u(S,E) > 0$ if $S \neq \emptyset$. The set V is a set of v voters $\{v_1, \ldots, v_v\}$, having each one approved up to l projects of A. The question is: does there exist a set $B \subset A$ of up to l projects such that the utility of B, $\prod_{v_i \in V} u(B_i, E)$ (or $\min_{v \in V} u(B_i, E)$), is at least R ?

We will reduce the strongly NP-complete problem Set cover (Garey & Johnson, 1979) to this problem. The Set cover problem is the following one: we are given a set $\mathcal{U}$ of n elements, called the universe, and a collection S of m sets whose union equals the universe. Given an integer K, the question is: does there exist a set $\mathcal{S}$ of sets in S, such that $\cup_{s \in \mathcal{S}} = \mathcal{U}$ and $|\mathcal{S}| \leq K$? We suppose that the number of subsets containing the same element is bounded by K – as shown by Lemma 2, the problem is still strongly NP-complete in that case.

Let $\mathcal{U}$, S and K be an instance of the Set cover problem. Let us create an instance of our problem.

For each element s in $\mathcal{U}$, we create a voter v_e. For every subset s in S, we create a project a_s of cost 1. This project is approved by any voter v_e such that $e \in s$. Note that, since the number of sets containing the same element is smaller than or equal to K, the number of projects approved by a voter is smaller than or equal to K. We set $l = K$ and $R = \epsilon$ with $\epsilon > 0$. The question is now: does there exist a bundle B of projects such that the product (or minimum) of the voters' utilities for bundle B is greater than or equal to ϵ ? Since ϵ can be as small as we want, we can simply look for a solution with value strictly larger than 0.

We show that there is a positive answer to this question if and only if there exists a cover of size K in S.

- Let us first assume that there is a cover C of size K in S. Let B^{cover} be the set of the K projects which correspond to the K sets of S. All voters have at least one of their approved projects in the bundle B^{cover}, since the projects corresponding to the sets have been chosen by the voters matching with the elements. Therefore, if a voter did not have at least one approved project in B^{cover}, then the cover C would not cover the element corresponding to the voter. The answer to our problem is thus 'yes'.
- Let us now assume that it possible to select at most K projects such that the total utility is at least $R = \epsilon$. Since we use the product or the min, this means that every voter has at least one of her approved projects in the funded bundle B. We know that for each $v_e \in V$, there is one project of A_e in B. If we consider the cover C^B formed by the sets corresponding to the projects in B, this means that for every element e, there is a subset s in C^B such that $e \in s$. Since the size of B is at most K, the size of C^B is at most K, which means that C^B is a feasible cover for the Set cover problem. The answer is thus 'yes'.

There exists a polynomial time reduction of the strongly NP-complete Set cover problem into our problem: our problem is strongly NP-hard. Furthermore, a δ-approximate algorithm, with δ, would allow to detect whether there exist a solution with a product (or minimum) of utilities strictly larger than 0, and thus would allow to solve the Set cover problem. Therefore, for any $\delta > 0$, there does not exist polynomial time δ-approximate solution for our problem, unless $P = NP$. $\square$

7 A branch and bound algorithm

In this section, we propose an exact branch and bound algorithm for $\alpha - r_u$for $\alpha \in \{\sum, \prod, \min\}$ since, as shown in the previous section, this is NP-hard. We also run experiments on real-life instances.

7.1 Description of the algorithm

Let us now present a branch and bound algorithm which solves PB-Max–α–u exactly, for $\alpha \in \{\sum, \min, \prod\}$. Each level of the decision tree corresponds to a project: we either add it to the funded projects – if it fits in the remaining budget, or we ban it for the current node and all of its sons. In such a decision tree, each leaf corresponds to a feasible bundle. Since every decision is binary and there are n consecutive decisions, corresponding to the n projects, there are 2^n leaves corresponding to the 2^n possible subsets. Since the cost of an optimal bundle is at most l, at a current node, we add a project only if its cost is at most l minus the cost of the currently funded projects – this allows us to prune the tree. Moreover, at each node, we compute a feasible solution, and an upper bound of the value of the quality (w.r.t. the objective function of PB-Max–$\alpha - u$) of a bundle that is reachable from this node. If the upper bound of the value of a reachable bundle is smaller than the value of a feasible solution we already know, then exploring the node's sons is useless, and we prune the tree.

Case where $\alpha = \sum$. We compute a new feasible solution using a greedy rule, called $\mathcal{R}^g_{|B_v|}$ by Talmon and Faliszewski (2019), and which simply selects the projects by decreasing number of selections. At each node we consider the non yet considered projects by decreasing number of selections, and we add a project if it fits in the remaining budget. As we will see in Section 7.2, using this algorithm at the root of the tree can also be used as a good and fast heuristic.

The upper bound follows the same principle than the classic upper bound for the KNAPSACK problem, it is a linear relaxation. In order to compute our upper bound, we need an upper bound on the utility that each project can give to a voter. By multiplying it by the number of voters who selected this project, we obtain an upper bound of the utility that a project can bring to the whole set of voters.

Before starting the exploration of the decision tree, for each project a, we compute the sum of the Möbius transforms of each feasible subset in which a appears, divided by the size of this subset. This is an upper bound of how much utility a project can provide to one voter, we multiply it by the number of voters who selected this project, and obtain an upper bound of how much utility the project can bring to the whole set of voters. Note that this can be applied to other utility functions since the Möbius transforms can be computed for any utility function.

At each node, we then run the greedy algorithm selecting the (non yet selected nor eliminated) projects by decreasing upper bounds and we relax the integrity constraint, obtaining a fractional solution. This gives us an upper bound of the best solution that can be obtained at the current node. Note that the k-additivity assumption is particularly useful here since the maximum utility a project can give decreases when k decreases, since all the Möbius transforms of subsets of size strictly greater than k are null.

Case where $\alpha \in \{\min, \prod\}$. We compute a feasible solution as follows: we look for the set of least satisfied voters. We choose the most frequently selected project by these voters, among projects that fits into the remaining budget. We repeat this process until there is no budget left.

For the upper bound: at the root of the decision tree, we assume that each voter gets her favorite set of projects. At each node, we consider that each voter gets the projects that she voted for among the already selected projects, plus all the projects that she selected among projects that still fits in the budget and which have not been considered yet. For example, if the selected projects cost half the budget, then any project costing more than half the budget could not be chosen and is therefore banned. If a project is banned, then we simply add it to the ban list. Then, we remove all newly banned project from the best reachable subsets of the voters. This gives us an upper bound of the value of any reachable solution.

Computing the utilities. The utility provided by a given solution B to a voter v_i is the utility of B_i. Determining B_i and computing its utility can be done in polynomial time if we know the utility function. Therefore, for each node of the decision tree, computing solutions and determining their value as upper and lower bounds can be done in polynomial time.

To determine the utility of a bundle with function u_M, we use Equation 2. Because of its recursive nature, we compute first, as a preprocessing step, the utility of singletons, then pairs, then triplets and so forth. Determining the utilities in this way cost up to 2^n (since there are 2^n subsets) times nv operations (since determining the appearance rate of a subset costs nv operations). This calculation is much faster with

Function	$n=5$	$n=8$	$n=10$	$n=12$	$n=15$
1-additive	0.013	0.057	0.10	0.26	0.77
2-additive	0.015	0.076	0.15	0.60	3.60
3-additive	0.016	0.11	0.25	1.43	9.85

Table 1 Completion time (s) of the branch and bound algorithm.

the k-additivity hypothesis, stating, as seen earlier, that we can consider interactions only in subsets of projects of size at most k.

With the k-additivity hypothesis, it is possible to know the utility of a subset of size j in $O(j^k)$ operations, since its utility is the sum of all the Möbius transforms of its parts, and there are at most j^k parts with a non null Möbius transform. This hypothesis has great implications on the computational side.

7.2 Experiments

We use real instances from the Pabulib (Stolicki et al., 2020) library with a budget limit on the approbation sets of the voters. Experiments are run on an Intel Core i5-8250U processor with 8GB of RAM. We study the completion time of our algorithm and the impact of the synergies on the returned solutions. We consider that $\alpha = \sum$ for the experiments since the sum is the most common aggregator.

Quality of the heuristic.

On average, the solution returned by the exact (branch and bound) algorithm has an overall utility 0.28% higher than the utility of the solution returned by the heuristic $\mathcal{R}^g_{|B_v|}$ for the u_M function: the heuristic returns, on the instances of Pabulib, very good solutions with regards to our optimization criterion.

Impact of the k-additivity assumption.

The k-additivity assumption allows to decrease the calculation time significantly – the lower k is, the fastest is the algorithm. Table 1 indicates the computation times obtained when $k=1$ (no synergy), and when $k=2$ and $k=3$ with utility function u_M.

Impact of considering synergies.

We compare the optimal solution for the overlap utility function (1 additive) and the u_M function with no k-additivity assumption. The optimal solutions are different in 35% of the instances, and the amount of money spent differently on average for all the instances is of 28.5%. Therefore, taking synergies into account impacts the returned bundle in a little bit more than a third of the instances, and this impact may be important since the returned bundle considering synergies then differs significantly from an optimal bundle ignoring synergies.

8 Conclusion and future works

This paper represents a first step towards taking project interactions into account in participatory budgeting problems. We introduced a utility function u_M based on the frequency of selection of groups of projects by the voters, and we showed that it fulfills desirable axioms. We furthermore showed that taking into account synergies is NP-hard with the main aggregation criterion, and this for very general utility functions. We designed an exact algorithm that we implemented with u_M but which can also be used with others utility functions.

Whereas, for very costly projects, decision makers will probably identify synergies "by hand", when there are numerous small projects, the authorities will likely be unable or unwilling to identify the synergies. In such settings, identifying the synergies thanks to the preferences of the voters, is promising. We could also imagine settings where a community decides to use a participatory budgeting approach to set a program of a maximum fixed total duration l among various events (presentations, courses, documentaries, etc), each event having a duration (considered as a cost). Members of the community could be asked to select the events they prefer, using knapsack voting: this situation is a participatory budgeting problem for which it would be particularly interesting to take into account synergies between the events.

There are numerous future work directions. For example, it would be useful to design utility functions that fit as much as possible to the reality experienced by the users. Another challenging direction would be to design algorithms that take into account synergies while ensuring proportional representation of groups of voters.

References

Aziz, H., Lee, B.E., Talmon, N. (2018). Proportionally representative participatory budgeting: Axioms and algorithms. E. André, S. Koenig, M. Dastani, & G. Sukthankar (Eds.), *Proceedings of the 17th international conference on autonomous agents and multiagent systems, AAMAS 2018* (pp. 23–31). International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM.

Aziz, H., & Shah, N. (2021). Participatory budgeting: Models and approaches. *Pathways between social science and computational social science* (pp. 215–236). Springer.

Benade, G., Nath, S., Procaccia, A.D., Shah, N. (2021). Preference elicitation for participatory budgeting. *Management Science*, *67*(5), 2813–2827,

Brandt, F., Conitzer, V., Endriss, U., Lang, J., Procaccia, A.D. (2016). *Handbook of computational social choice.* Cambridge University Press.

Durand, M., & Pascual, F. (2022). Collective schedules: Axioms and algorithms. *Symposium on algorithmic game theory-sagt 2022* (Vol. 13584, pp. 454–471).

Fairstein, R., Meir, R., Gal, K. (2021). Proportional participatory budgeting with substitute projects. *CoRR*, *abs/2106.05360*, , Retrieved from https://arxiv.org/abs/2106.05360 2106.05360

Fairstein Roy, B.G., & Kobi, G. (2023). Participatory budgeting design for the real world. *Proceedings of the aaai conference on artificial intelligence.*

Freeman, R., Pennock, D.M., Peters, D., Wortman Vaughan, J. (2021). Truthful aggregation of budget proposals. *Journal of Economic Theory*, *193*, 105234, https://doi.org/https://doi.org/10.1016/j.jet.2021.105234 Retrieved from https://www.sciencedirect.com/science/article/pii/S002205312100051X

Garey, M.R., & Johnson, D.S. (1979). *Computers and intractability* (Vol. 174). San Francisco.

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

Jain, P., Sornat, K., Talmon, N. (2020). Participatory budgeting with project interactions. *Ijcai* (pp. 386–392).

Jain, P., Talmon, N., Bulteau, L. (2021). Partition aggregation for participatory budgeting. *Proceedings of the 20th international conference on autonomous agents and multiagent systems* (pp. 665–673).

Peters, D., Pierczyński, G., Skowron, P. (2021). Proportional participatory budgeting with additive utilities. *Advances in Neural Information Processing Systems*, *34*, 12726–12737,

Rey, S., & Maly, J. (2023). *The (computational) social choice take on indivisible participatory budgeting.*

Rosenfeld, A., & Talmon, N. (2021). What should we optimize in participatory budgeting? an experimental study. *CoRR*, *abs/2111.07308*, , Retrieved from https://arxiv.org/abs/2111.07308 2111.07308

Rota, G.-C. (1964). On the foundations of combinatorial theory i. theory of möbius functions. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, *2*(4), 340–368,

Sreedurga, G., Ratan Bhardwaj, M., Narahari, Y. (2022, 7). Maxmin participatory budgeting. L.D. Raedt (Ed.), *Proceedings of the thirty-first international joint conference on artificial intelligence, IJCAI-22* (pp. 489–495). International Joint Conferences on Artificial Intelligence Organization. Retrieved from https://doi.org/10.24963/ijcai.2022/70 (Main Track)

Stolicki, D., Szufa, S., Talmon, N. (2020). Pabulib: A participatory budgeting library. *CoRR*, *abs/2012.06539*, , Retrieved from https://arxiv.org/abs/2012.06539 2012.06539

Talmon, N., & Faliszewski, P. (2019). A framework for approval-based budgeting methods. *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 2181–2188).