Participatory Budgeting with Project Groups

Pallavi Jain[1], **Krzysztof Sornat**[2], **Nimrod Talmon**[3] and **Meirav Zehavi**[3]
[1]Indian Institute of Technology Jodhpur, India
[2]MIT CSAIL, USA
[3]Ben-Gurion University, Israel
pallavi@iitj.ac.in, sornat@mit.edu, {talmonn, meiravze}@bgu.ac.il

Abstract

We study a generalization of the standard approval-based model of participatory budgeting (PB), in which voters are providing approval ballots over a set of predefined projects and—in addition to a global budget limit—there are several groupings of the projects, each group with its own budget limit. We study the computational complexity of identifying project bundles that maximize voter satisfaction while respecting all budget limits. We show that the problem is generally intractable and describe efficient exact algorithms for several special cases, including instances with only few groups and instances where the group structure is close to being hierarchical, as well as efficient approximation algorithms. Our results could allow, e.g., municipalities to hold richer PB processes that are thematically and geographically inclusive.

1 Introduction

In the standard approval-based model of participatory budgeting (PB) [Cabannes, 2004; Shah, 2007], specifically, the model of *Combinatorial PB* [Aziz and Shah, 2021], we are given a set of m projects, each with its cost, n approval votes (i.e., each voter provides a subset of projects she approves), and a budget limit. The task is to aggregate the votes to select a bundle (i.e., a subset) of projects that respects the budget limit. PB has caught quite considerable attention lately [Aziz *et al.*, 2018; Talmon and Faliszewski, 2019; Aziz and Shah, 2021] as it is being used around the world to decide upon the spending of public (mostly municipal) money. Indeed, while other ballot types are also natural, most real-world PB processes use approval ballots.

Here we consider a setting of participatory budgeting in which the projects are classified into groups that might be intersecting, and each group comes with its own budget constraint, so that the result of the PB—i.e., the aggregated bundle—shall respect not only the global budget limit, but also the limits of each of the groups. To make things concrete and formal, below is the decision version of our problem (in the optimization version of GROUP-PB, denoted MAX-GROUP-PB, the goal is to maximize the utility):

GROUP-PB
Input: A set P of projects with their cost function $c\colon P \to \mathbb{N}$, a set $\mathcal{V}$ of voters with their approval ballots $\mathcal{E} = \{P_v \subseteq P\colon v \in \mathcal{V}\}$, a family of groups of projects $\mathcal{F} \subseteq 2^P$ with their budget function $b\colon \mathcal{F} \to \mathbb{N}$, a global budget limit B, and a desired utility value u.
Question: Is there a set of projects $X \subseteq P$ such that $\sum_{v\in\mathcal{V}} |P_v \cap X| \geq u$, $\sum_{p\in X} c(p) \leq B$, and for every set $F \in \mathcal{F}$, $\sum_{p\in F\cap X} c(p) \leq b(F)$?

W.l.o.g., we assume $b(F) \leq B$ for every $F \in \mathcal{F}$, and that no two sets in $\mathcal{F}$ are identical, i.e., for all $S_1, S_2 \in \mathcal{F}, S_1 \neq S_2$. Also, w.l.o.g., we assume that every project in P is approved by at least one voter. (Note that voter utility equals the number of approved projects that are funded; this is the most popular definition of utility in PB however other definitions exist, some of which we discuss later.)

Example 1. Let $P = \{p_1, p_2, p_3, p_4\}$. Let the cost of projects be as follows: $c(p_1) = 2, c(p_2) = 1, c(p_3) = 3, c(p_4) = 1$. Suppose that we have only two voters, say v, v' and $P_v = \{p_1, p_2, p_3\}, P_{v'} = \{p_3, p_4\}$. Let $\mathcal{F} = \{F_1 = \{p_1, p_3\}, F_2 = \{p_2, p_4\}\}$. Let $b(F_1) = 3, b(F_2) = 2$. Note that we are allowed to take only one project from F_1. Let $B = 5$ and $u = 3$. Let $X = \{p_3, p_4\}$. Note that $|P_v \cap X| = 1$ and $|P_{v'} \cap X| = 2$. Thus, $\sum_{v\in\mathcal{V}} |P_v \cap X| = 3 = u$. Furthermore, $c(p_3) + c(p_4) = 4 \leq B$, and the costs of projects from sets F_1 and F_2 are 3 and 1, respectively, that is, $\sum_{p\in F_1} c(p) = 3 = b(F_1)$ and $\sum_{p\in F_2} c(p) = 1 \leq b(F_2)$.

Our work is motivated mainly by the following use-cases:

1. **Geographical budgeting:** Consider a city (say, Paris), consisting of several districts. To not spend all public funds on projects from, say, only one district, GROUP-PB is useful: group projects according to districts and select appropriate budget limits (making sure that, e.g., none of Paris's 20 districts would use more than 10% of the total budget). Indeed, for other cities the number of districts may be smaller (e.g., Jerusalem has four districts). A more fine-grained solution, incorporating neighborhoods, streets, etc., is also possible. Currently, such geographic inclusiveness is usually achieved ad hoc by holding separate per-district elections.
2. **Thematic budgeting:** Projects usually can be naturally grouped into types, e.g., educational projects, recre-

ational projects, and so on. Note that, in such cases it might be that groups do intersect: e.g., a recreational park might be of recreational purposes as well as for environmental purposes, thus contained in two sets of projects. GROUP-PB is useful here: group projects accordingly, making sure that not all the budget is being spent on, say, projects of only one type.

3. **Non-budgeting use-cases:** GROUP-PB is useful in contexts other than PB: e.g., to decide which processes to run on a time-limited computing server, where available processes can be naturally grouped into types and it is not desired to use all computing power for, say, processes of only one type. (In this examples, voters may be various stakeholders.)

Remark 2. Note that in some settings the groups may or not intersect. Indeed, even if the groups do not intersect, we argue that it is better to hold the election as one global election, and consider the groupings of the projects as we do in our model; this is so, as it provides more flexibility of effectively dividing the global budget between the groups, based on voter preferences, and not based on some preliminary decision. On a related note, some projects indeed may be of use to, say, several districts (e.g., a park positioned in one specific district but enjoyed by residents from several districts): in such cases it might be better to, say, not include such a project in the group-wise budget of any district; or, consider a more fine-grained model in which such a project may be counted partially to each of the districts that are relevant to it—we do not consider such generalizations of our model here.

1.1 Our Contributions

We introduce and study GROUP-PB, first demonstrating its computational intractability even for some very restricted cases (Theorem 10, Theorem 14). Interestingly, GROUP-PB can be solved in polynomial-time if a project belongs to at most one group, but becomes NP-hard as soon as a project can belong to two groups (Theorem 14). Positively, we show that GROUP-PB can be solved in polynomial-time for a constant number of groups (Theorem 17) and for instances with hierarchical group structure (i.e., any pair of groups must be either non-intersecting or in containment relation; Lemma 11). Note that in some cases, e.g., when grouping projects by geographical regions/districts/neighborhoods, the group structure is indeed hierarchical.

We extend our study to parameterized algorithms. We consider the following parameters: the number of projects (m), the number of voters (n), the maximum size of an approval set (app), the budget (B), the maximum size of a group in the family $\mathcal{F}$ (s), the utility (u), the layerwidth (ℓ) (see Definition 4 for layerwidth), the size of the family $\mathcal{F}$ (g), and $b_{\max} = \max_{F \in \mathcal{F}} b(F)$; and obtain both tractability and intractability results. The motivation for considering these is the following: the number of voters, n, can be small in cases when we do PB by the council or in a small community; m is sometimes quite small (e.g., the PB instances of Stanford Participatory Budgeting Platform[1] usually consist of only 10–20 projects); ℓ and $b_{\max}$ can be set by designer—they are generally not small, yet we add them for completeness; B and u are also generally not small, but added for completeness; app, s and g can be set by the designer, and they are usually rather small, e.g. 5 to 10.

Since the problem can be solved in polynomial-time on hierarchical instances, we also consider two distance parameters to a hierarchical instance, D_p and D_g, the minimum number of projects (respectively, groups) whose deletion leads to a hierarchical instance; finding efficient algorithms for such distance parameters implies that not only hierarchical instances can be solved efficiently, but also instances that are close to being such. This is particularly useful in the presence of a few outliers (due to this, distance parameters are studied frequently in parameterized complexity, including in voting theory [Bredereck *et al.*, 2014; Gupta *et al.*, 2020a; Gupta *et al.*, 2020b]).

[1] https://pbstanford.org/

Result	Parameters	Reference
paraNP-hard	$b_{\max} + s + \mathsf{app} + \ell$	Theorem 10
paraNP-hard	$b_{\max} + s + n + \ell$	Theorem 10
W[1]-hard	$b_{\max} + s + \mathsf{app} + u$	Theorem 10
W[1]-hard	$b_{\max} + s + n + u$	Theorem 10
paraNP-hard	$\ell + D_g$	Theorem 14
FPT	D_p	Theorem 16
XP	g	Theorem 17
FPT	$g + u$	Corollary 19
FPT	$g + n$	Theorem 20
FPT	$g + B$	Theorem 21

Table 1: Parameterized complexity of GROUP-PB wrt. $b_{\max} = \max\{b(F) : F \in \mathcal{F}\}$, $s = \max\{|F| : F \in \mathcal{F}\}$, the maximum number app of projects a voter approves, the layerwidth ℓ (see Definition 4), the number n of voters, the required utility u, the number D_g (D_p) of groups (resp., projects) to delete to get a hierarchical structure, the number of groups $g = |\mathcal{F}|$, and the total budget B.

Upper-bound on g	Approximation ratio
$\mathcal{O}(1)$	P
$\log_4 \log(m^{\mathcal{O}(1)})$	PTAS
any g	$g + 2$

Table 2: Achieved approximation ratios (in polynomial-time) depending on the number g of groups. The smaller g compared to m, the better approximation for MAX-GROUP-PB we can achieve. The results are proved in the full version of the paper [Jain *et al.*, 2020b].

Lower-bound on g	Inapproximability
$\frac{3}{2}m$	no PTAS
m^2	no $g^{\frac{1}{2}-\epsilon}$-approximation algorithm
m^2	no $m^{1-\epsilon}$-approximation algorithm

Table 3: Achieved polynomial-time inapproximability results depending on the number g of groups. The larger g compared to m, the higher is the approximation ratio excluded. The results are proved in the full version of the paper [Jain *et al.*, 2020b].

In particular, the main focus of our paper is on adding a group structure on top of a standard PB instance; from this point of view, PB election designers can choose how complex they want the group structure to be. Thus, studying the complexity of GROUP-PB wrt. our parameters—in particular, the parameter g and the distance parameters—sheds light on the effect of adding groups on the complexity of the problem (which is polynomial-time solvable when there are no groups). Following our parameterized tractability results, a PB election designer can practically use our group structures, albeit perhaps not with an arbitrary number of groups of unlimited structural complexity.

Table 1 lists most of our complexity results. Parameterized complexity wrt. g is open (Open Question 18), however we have an approximation scheme that is FPT wrt. g (Theorem 22). Tables 2 and 3 summarize our (in)approximability results that we present in the full version of the paper [Jain *et al.*, 2020b].

Remark 3. While some city planners may not care for complexity results, we are personally aware of some that are hesitant to use algorithmic methods that may not be efficient and thus may require extensive computational resources.[2] Thus, in addition to being of theoretical interest, our complexity analysis results have practical implications regarding the feasibility of adding group-wise budget upper bounds to PB. (At least, as much as theoretical results imply practical feasibility.)

Initial Observations. For completeness, we mention that GROUP-PB is trivially FPT wrt. m, by a brute-force algorithm in $\mathcal{O}^*(2^m)$ time[3] (and, as the Exponential Time Hypothesis implies a lower bound of $2^{o(|V|)}$ for INDEPENDENT SET, we conclude a lower bound of $2^{o(m)}$ following the reduction in the proof of Theorem 10). Furthermore, GROUP-PB is FPT wrt. $\mathtt{app} + n$ as every project is approved by at least one voter, implying $m \leq \mathtt{app} \cdot n$.

Road Map. In Section 2 we consider a structural property of family of sets, useful for obtaining a polynomial-time algorithm for hierarchical families and may also be of independent interest. Then, in Section 3, we present intractability results of GROUP-PB. Sections 4 and 5 are devoted to parameterized analysis of GROUP-PB. Due to space constraints, some results, proofs or proof details are deferred to the full version of the paper [Jain *et al.*, 2020b].

1.2 Related Work

The literature on PB is quite rich [Aziz *et al.*, 2018]; formally, we generalize the framework of Talmon and Faliszewski [2019] by adding group structures to approval-based PB. Jain et al. [2020a] and Patel et al. [2021] also consider—albeit significantly simpler—group structures (with layerwidth 1; see Definition 4). Fairness constraints are studied, e.g., in the contexts of influence maximization [Tsang *et al.*, 2019], clustering [Chierichetti *et al.*, 2017], and allocation problems [Benabbou *et al.*, 2018]. Our focus is on fairness in PB (e.g., not spending all funds on one district). Recently, Hershkowitz et al. [2021] introduced a district-fairness notion, allowing projects to have different utility for different districts. Researchers have also studied fairness and group structures for multiwinner elections [Izsak *et al.*, 2018; Celis *et al.*, 2018; Yang and Wang, 2018; Ianovski, 2019; Gupta *et al.*, 2020b], which is a special case of PB.

Technically, GROUP-PB is a special case of the d-DIMENSIONAL KNAPSACK problem (d-DK) [Kellerer *et al.*, 2004, Section 9]: given a set of items, each having a d-dimensional size-vector and its utility, a d-dimensional knapsack capacity vector β with an entry for each dimension, and required integer utility—with all input numbers being non-negative integers—the goal is to choose a subset of the items with at least the required total utility and such that the sum of the chosen items' sizes is bounded by the knapsack capacity, in each dimension. d-DK generalizes GROUP-PB: items in d-DK correspond to projects; fix an order on $\mathcal{F}$, i.e., $(F_1, F_2, \ldots, F_g)$, resulting in $d = g + 1$ many dimensions, a $(g+1)$-dimensional size vector γ for an item $p \in P$, defined by $\gamma_p(i) = c(p)$ if $p \in F_i$ and $\gamma_p(i) = 0$ otherwise, $\gamma_p(g+1) = c(p)$ for every $p \in P$, corresponding to a global budget, utility of an item $p \in P$ equals its approval score, required utility in d-DK equals u, and the $(g+1)$-dimensional bin β is defined via $\beta(i) = b(F_i)$ for $i \in \{1, 2, \ldots, g\}$, with $\beta(g+1) = B$. So, GROUP-PB is an instance of $(g+1)$-DK where each item $p \in P$ has only two possible sizes over dimensions, i.e., 0 and $c(p)$. Crucially, as our model is a special case we can use our special instance structure; hence, we treat results for d-DK as a good benchmark, in particular, the (in)approximability results for d-DK [Kellerer *et al.*, 2004].

2 Layer Decompositions

Definition 4. A *layer decomposition* of a family of sets $\mathcal{F}$ is a partition of the sets in $\mathcal{F}$ such that every two sets in a part are disjoint. Each part is a *layer*. The *width* of a layer decomposition is the number of layers in it. The *layerwidth* of a family of sets $\mathcal{F}$, denoted by $\ell(\mathcal{F})$ (or simply ℓ if $\mathcal{F}$ is clear from the context), is the minimum width among all possible layer decompositions of $\mathcal{F}$.

Given a family $\mathcal{F}$ of sets and an integer ℓ, the decision problem LAYER DECOMPOSITION asks for the existence of a layer decomposition with width at most ℓ. The following hardness result follows a reduction from EDGE COLORING.

Theorem 5. LAYER DECOMPOSITION *is* NP*-hard even when* $\ell = 3$ *and* $s = 2$.

A reduction to 2-GRAPH COLORING gives a polynomial-time algorithm for layerwidth 2.

Theorem 6. *There exists a polynomial-time algorithm that finds a layer decomposition of layerwidth two, if it exists.*

We discuss hierarchical families of sets (also known as *laminar families*).

Definition 7. A family of sets $\mathcal{F}$ is called *hierarchical*, if every two sets F_1 and F_2 in the family $\mathcal{F}$ are either disjoint or $F_1 \subset F_2$ or $F_2 \subset F_1$.

[2] In particular, one of the authors, while trying to convince a deputy mayor of a medium-sized city to implement a participatory budgeting process, faced significant criticism regarding the worry of the need of using extensive computational resources.

[3] $\mathcal{O}^*$ hides factors that are polynomial in the input size.

Theorem 8. *There exists a polynomial-time algorithm that solves a given instance $(\mathcal{F}, \ell)$ of* LAYER DECOMPOSITION *when $\mathcal{F}$ is a hierarchical family.*

Remark 9. The general idea of the algorithm described in the proof of Theorem 8 is to build a graph with one vertex for each group and edges corresponding to group intersections, followed by traversing the graph in topological order and constructing the corresponding hierarchical tree. Note that, conveniently, the algorithm can be modified to construct an ordered layer decomposition such that every set in the i-th layer is a subset of a set in the $(i-1)$-th layer.

3 Intractability of General Instances

Next we prove intractability, showing that GROUP-PB is NP-hard even when some of the input parameters are constant. Note, importantly, that we can solve the standard PB problem—without project groups—in polynomial-time (as it can be solved using dynamic programming via equivalence to UNARY KNAPSACK [Talmon and Faliszewski, 2019]).

The following result is obtained via reductions from the INDEPENDENT SET (IS) problem on 3-regular 3-edge colorable graphs.

Theorem 10. GROUP-PB *is* NP*-complete even when $b_{\max} = 1$, $s = 2$, $\mathtt{app} = 1$, and $\ell = 3$; and even when $b_{\max} = 1$, $s = 2$, $n = 1$, and $\ell = 3$. Furthermore,* GROUP-PB *is* W[1]*-hard wrt. u even when $b_{\max} = 1$, $s = 2$, and $\mathtt{app} = 1$; and even when $b_{\max} = 1$, $s = 2$, and $n = 1$.*

4 Tractability of Hierarchical Instances

We start our quest for tractability by considering GROUP-PB instances whose group structure is hierarchical; when $\mathcal{F}$ is hierarchical, we refer to the GROUP-PB problem as HIERARCHICAL-PB. Fortunately, such instances can be solved in polynomial-time. Practically, hierarchical instances are appropriate, e.g., when considering disjoint geographical districts of a city.

Lemma 11. *There exists a polynomial-time algorithm that solves* HIERARCHICAL-PB.

Proof. Let $(\mathcal{V}, P, \mathcal{E}, \mathcal{F}, c, B, b, u)$ be a given instance of HIERARCHICAL-PB. W.l.o.g., assume that P is a set in the family $\mathcal{F}$ (otherwise, add it to $\mathcal{F}$ and set $b(P) = B$). Using Remark 9, let $\mathcal{L}$ be an ordered layer decomposition of $\mathcal{F}$ such that every set is a subset of some set in the preceding layer, and note that the first layer is $\{P\}$. Let S be a set in some layer, say L_i, such that $|S| > 1$. Suppose that there exists a project $p \in S$ such that p is not in any set of L_{i+1}. We add the set $\{p\}$ to $\mathcal{F}$ and L_{i+1}, and set $b(\{p\}) = c(p)$ (if the layer L_{i+1} does not exist, then we add this new layer). Note that we might increase the number of layers by 1. Let ℓ be the number of layers.

Now, we solve the problem using dynamic programming. For a set $S \in \mathcal{F}$ in the i-th layer, where $i \in [\ell - 1]$, such that $|S| > 1$, let $\mathtt{Part}_S$ denote the partition of a set S such that every part in $\mathtt{Part}_S$ is a set in the $(i+1)$-th layer. For a singleton set S, $\mathtt{Part}_S = \{S\}$. For every set $S \in \mathcal{F}$, we order the parts in $\mathtt{Part}_S$ arbitrarily. Let $|\mathtt{Part}_S|$ denote the number of parts in $\mathtt{Part}_S$ and let $\mathtt{Part}_S(i)$ denote the i-th part in $\mathtt{Part}_S$. Our DP table entries are defined as follows: For every set $S \in \mathcal{F}$, $j \in |\mathtt{Part}_S|$, and utility $z \in [u]$,

$$T[S, j, z] = \text{minimum cost of a subset of projects in first } j \text{ parts in } \mathtt{Part}_S \text{ that has utility } z.$$

For a project p, let $a(p)$ denote the number of voters who approve the project p (approval score). For a set S, let $a(S) = \sum_{p \in S} a(p)$. We compute the table entries level-wise in bottom-up order, that is, we first compute the value corresponding to sets at lower levels.

Base case: For every set S, where $S = \emptyset$ or $S \in \mathcal{F}$ and $0 \leq z \leq u$

$$T[S, 0, z] = \begin{cases} 0 & \text{if } z = 0, \\ \infty & \text{otherwise.} \end{cases}$$

Recursive Step: For every set $S \in \mathcal{F}$, $j \in [|\mathtt{Part}_S|]$, and $0 \leq z \leq u$, we compute as follows:

$$T[S, j, z] = \min_{0 \leq z' \leq z} \{T[S, j-1, z - z'] + T[\mathtt{Part}_S(j), |\mathtt{Part}_S(j)|, z']\}.$$

Correctness proof can be found in [Jain *et al.*, 2020b]. □

Some instances might not be hierarchical but only close to being such, thus we study two distance parameters, namely, the minimum number D_g of groups and the minimum number D_p of projects, respectively, whose deletion results in a hierarchical instance. Indeed, having efficient algorithms for such instances means that even more instances can be efficiently solved (e.g., instances with group structures corresponding to thematic districts in which some projects fit several groups).

We have the following lemmas—used later in the proofs for the parameterized complexity of GROUP-PB wrt. $D_g + s$ and D_p—which are concerned with computing the set of groups/projects whose deletion leads to hierarchical instance; their proofs follow branching arguments, as, for D_g, at least one set from each pair of conflicting groups shall be deleted, and, for D_p, for a pair of conflicting groups G_1, G_2, either $G_1 \setminus G_2$ or $G_2 \setminus G_1$ or $G_1 \cap G_2$ shall be removed.

Lemma 12. *There exists an algorithm, running in $\mathcal{O}^*(2^{D_g})$ time, that finds a minimum-sized set of groups whose deletion results in a hierarchical instance.*

Lemma 13. *There exists an algorithm, running in $\mathcal{O}^*(3^{D_p})$ time, that finds a minimum-sized set of projects whose deletion results in a hierarchical instance.*

Unfortunately, we have the following intractability result.

Theorem 14. GROUP-PB *is* NP*-hard even when $D_g = 2$ and $\ell = 2$.*

Nevertheless, combining with s helps:

Theorem 15. GROUP-PB *is* FPT *wrt. $D_g + s$.*

In contrast to Theorem 14, parametrization by the delete-project-distance of an instance to be hierarchical is tractable.

Theorem 16. GROUP-PB *is* FPT *wrt. D_p.*

5 Tractability with Few Groups

Next we concentrate on the number g of groups as a parameter as, indeed, the groups are the new ingredient we bring to the standard model of PB. Practically, the number g of groups may be set by the entity organizing the PB process, thus can be as small as the organizer wishes. First, we have the following result for the parameter g. The proof follows an algorithm that considers 2^g "types" of projects, guesses the joint utility achieved from projects of each type, and finds corresponding bundles using dynamic programming.

Theorem 17. GROUP-PB *is* XP *wrt.* g.

Unfortunately, we do not know whether GROUP-PB is FPT wrt. g; indeed, this is the main question left open.

Open Question 18. Is GROUP-PB FPT wrt. g?

Note, however, that in [Jain *et al.*, 2020b] we provide a proof of W[1]-hardness wrt. g, albeit for a slightly more general problem, in which we are also given utility requirements for each group. Next we consider combined parameters. The next result follows the proof of Theorem 17.

Corollary 19. GROUP-PB *is* FPT *wrt.* $g + u$.

Careful Mixed Integer Linear Programming (MILP) formulation implies the following.

Theorem 20. GROUP-PB *is* FPT *wrt.* $g + n$.

Proof. Recall that w.l.o.g. we assume that every project is approved by at least one voter. We construct a MILP by defining a type of a project by a pair (R, w), where $R \subseteq \mathcal{F}$ and $w \in [n]$. A project of type (R, w) belongs to all the groups in R (and to none of the groups in $\mathcal{F} \setminus R$) and it is approved by exactly w voters. Note that we have $n \cdot 2^g$ types of projects. We define an integer variable $x_{R,w}$ for all $R \subseteq \mathcal{F}$ and $w \in [n]$, meaning how many projects of type (R, w) are in a solution. Let $|(R, w)|$ be the number of projects of type (R, w).

Next, we split $x_{R,w}$ into a sum of $|(R, w)|$ real variables:

$$x_{R,w} = \sum_{i \in [|(R,w)|]} y_{R,w,i},$$

where $y_{R,w,i} \in [0, 1]$ is a continuous extension of a binary variable that indicates whether we take the i-th cheapest projects of type (R, w) to a solution. From equation (5) we get also $x_{R,w} \in \{0, 1, \ldots, |(R, w)|\}$. Note that we have $\sum_{R \subseteq \mathcal{F}} \sum_{w \in [n]} |(R, w)| = m$ real variables $y_{R,w,i}$ because each project has exactly one type. We implement the budget function by writing a constraint for each group $F \in \mathcal{F}$:

$$\sum_{R: F \in R \subseteq \mathcal{F}} \sum_{w \in [n]} \sum_{i \in [|(R,w)|]} y_{R,w,i} \cdot c(R, w, i) \leq b(F),$$

where $c(R, w, i)$ is the cost of the i-th cheapest project of type (R, w). We add a global budget limit constraint as follows:

$$\sum_{R \subseteq \mathcal{F}} \sum_{w \in [n]} \sum_{i \in [|(R,w)|]} y_{R,w,i} \cdot c(R, w, i) \leq B.$$

The objective function is:

$$\max \sum_{R \subseteq \mathcal{F}} \sum_{w \in [n]} w \cdot x_{R,w}.$$

Any optimal solution (x^*, y^*) of the MILP can be transformed into an optimal solution (x^*, y^{int}) consisting of integer variables only: define $y^{\text{int}}_{R,w,i} = 1$ for $i \in \{1, \ldots, x^*_{R,w}\}$ and $y^{\text{int}}_{R,w,i} = 0$ for $i \in \{x^*_{R,w} + 1, \ldots, |(R, w)|\}$.

A proof that (x^*, y^{int}) is a feasible and an optimal solution can be found in [Jain *et al.*, 2020b]. Our MILP can be solved in $\mathcal{O}^*(2^{n \log(n) \cdot 2^{\mathcal{O}(g)}})$ time [Lenstra, 1983; Bredereck *et al.*, 2020]. □

Also combining g with the budget B helps, as we can use the DP for $(g + 1)$-DK, which runs in time upper-bounded by $\mathcal{O}^*(n \cdot (b_{\max} + 1)^g (B + 1)) \leq \mathcal{O}^*(n \cdot (B + 1)^{(g+1)})$ [Kellerer *et al.*, 2004, Section 9.3.2].

Theorem 21. GROUP-PB *is* FPT *wrt.* $g + B$.

5.1 FPT Approximation Scheme for g

Recall that GROUP-PB is XP wrt. g (Theorem 17) and recall our open question regarding whether GROUP-PB is FPT wrt. g (Open Question 18). Next we show an approximation scheme for MAX-GROUP-PB that is FPT wrt. g (compare this result also to that described in [Jain *et al.*, 2020b], showing that there does not exist a constant-factor approximation algorithm unless P = NP, even if g is as small as m^2).

In particular, our approximation notion is the following: an algorithm has an approximation factor $\alpha \geq 1$ if it always outputs a solution that has at most α factor less utility than the optimal solution.

Theorem 22. *There exists an algorithm that for any fixed* $\epsilon > 0$ *finds a* $(1 + \epsilon)$*-approximate solution to* MAX-GROUP-PB *in* FPT *time wrt.* g.

Proof. The idea of the algorithm is as follows. First, we reduce the given instance of GROUP-PB to an instance of GROUP-PB with an additional feasibility restriction, in particular, such that a feasible solution has to contain exactly one project from each project type, where a type of a project is uniquely defined by the family of groups to which the project belongs. The reduction, shown below, takes FPT time wrt. g. In the second step we will round down the approval score of each project to the closest multiplicity of $(1 + \epsilon)$, in effect bounding the number of different approval scores of a project to the logarithmic function of the input size. Then we will apply a brute-force enumeration that runs in FPT time wrt. g.

More formally, let us fix $\epsilon > 0$ and an instance $\mathcal{I} = (\mathcal{V}, P, \mathcal{E}, \mathcal{F}, c, B, b)$ of MAX-GROUP-PB. Recall that w.l.o.g. we assumed that each project is approved by at least one voter. Let $a \colon P \to \{1, 2, \ldots, |\mathcal{V}|\}$ be an approval score function, i.e., $a(p) = |\{v \in \mathcal{V} : p \in P_v\}|$. Notice that the approval score function $a(\cdot)$ can be encoded in unary (instead of having voters explicitly). Let A be the total approval score of all the projects, e.g., $A = \sum_{p \in P} a(p)$. Let $u^*(\mathcal{I})$ be the value (total utility) of an optimal solution to $\mathcal{I}$. To avoid triviality, w.l.o.g. we assume $u^*(\mathcal{I}) > 0$.

Now, given a subfamily $R \subseteq \mathcal{F}$, we say that a project p is of type R if it belongs to all the groups in R and to none of the groups in $\mathcal{F} \setminus R$ (so every project has an unique type). We have at most 2^g types of projects. First, we fix an optimal solution X^* and we do the following preprocessing on the

instance $\mathcal{I}$. For every project type, we guess whether at least one project of the type is contained in X^*, and if none, we delete all projects of that type. We can do this preprocessing in $\mathcal{O}^*(2^{2^g})$ time. Note that, after this step, the number of project types cannot increase because the number of projects cannot increase. (As this is just a preprocessing, in our next steps we override the notation and use P for the set of projects after the preprocessing.) We define the number of project types after the preprocessing as t, with $t \leq 2^g$.

For every project type $R \subseteq \mathcal{F}$ we run a dynamic programming procedure that outputs the following: For every value $v \in \{1, 2, \ldots, A\}$, we compute $\mathrm{cost}(R, v)$, which is the minimum cost of a subset of projects of type R whose total value is exactly v (it is equal to ∞ if there is no such subset). Also we store a bundle of projects, $\mathrm{bundle}(R, v)$, that realizes the minimum cost $\mathrm{cost}(R, v)$. We can compute $\mathrm{cost}(R, v)$ together with $\mathrm{bundle}(R, v)$ in time upper-bounded by $\mathcal{O}^*(t \cdot A \cdot |P|) = \mathcal{O}^*(2^g)$.

Now, we create a new instance $\mathcal{I}' = (\mathcal{V}', P', \mathcal{E}', \mathcal{F}', c', B', b')$ of GROUP-PB as follows. For every project type $R \subseteq \mathcal{F}$ and every value $v \in \{1, 2, \ldots, A\}$ such that $\mathrm{cost}(R, v)$ is not ∞, we define a project $\mathrm{proj}(R, v) \in P'$ of cost $c'(\mathrm{proj}(R, v)) = \mathrm{cost}(R, v)$ and approval score $a'(\mathrm{proj}(R, v))$ that is equal to v (equivalently, we can define A many voters in $\mathcal{V}'$, where the i-th voter approves all the projects $\mathrm{proj}(R, v)$ such that $v \geq i$). We define the type of all the projects $\mathrm{proj}(R, v)$, $v \in \{1, 2, \ldots, A\}$, as R. Note that a project $\mathrm{proj}(R, v)$ corresponds to a bundle of projects of type R from the original instance.

We keep the same global budget limit, i.e., $B' = B$. For every group $F \in \mathcal{F}$, we define a group $T_F \in \mathcal{F}'$ that contains all projects $\mathrm{proj}(R, v)$ whose type R contains the group F, i.e., $T_F = \{\mathrm{proj}(R, v) : F \in R\}$. We define $b'(T_F) = b(F)$.

We show correspondence of feasible solutions in both instances. Let $\mathcal{I}'_1$ be the instance $\mathcal{I}'$ of GROUP-PB restricted to solutions containing exactly one project of each type.

Lemma 23. *Every feasible solution to $\mathcal{I}'_1$ can be transformed into a feasible solution to $\mathcal{I}$ with the same utility in polynomial-time.*

Lemma 24. *We have $u^*(\mathcal{I}) \leq u^*(\mathcal{I}'_1)$.*

Note that from Lemma 23 and Lemma 24 we could derive $u^*(\mathcal{I}) = u^*(\mathcal{I}'_1)$. Although the crucial ingredient of Lemma 23 is that we can transform a feasible solution to $\mathcal{I}'_1$ into a feasible solution to $\mathcal{I}$ (keeping the value of the solution) in polynomial-time. In the second step (called *bucketing*) we round down the approval score of each project to the closest multiple of $(1 + \epsilon)$. Let $\mathcal{I}''_1$ be an instance after the bucketing procedure (note that we do not change costs and budget limits when bucketing the approval scores).

Lemma 25. *We have $u^*(\mathcal{I}''_1) \geq \frac{u^*(\mathcal{I}'_1)}{1+\epsilon}$.*

Because of bucketing, it is possible that two projects of the same type but with different approval scores are rounded down to the same value. Hence we keep the project of minimum cost. So, overall, after bucketing we have at most $\log_{1+\epsilon}(A)$ projects of each type. For each project type, we branch on which project of that type is selected and we store the best solution X'' (it has utility $u^*(\mathcal{I}''_1)$). This takes $\mathcal{O}^*((\log_{1+\epsilon}(A))^t)$ time, which can be bounded by $\mathcal{O}^*(2^{\frac{1}{2} \cdot 4^g})$, i.e., the algorithm runs in FPT time wrt. g. Note that ϵ (which is a fixed constant) is present only in the polynomial on the input size term.

The solution X'' is feasible to $\mathcal{I}'_1$, hence using Lemma 23 we can construct a feasible solution to $\mathcal{I}$ with utility equal to

$$u^*(\mathcal{I}''_1) \geq \frac{u^*(\mathcal{I}'_1)}{1+\epsilon} \geq \frac{u^*(\mathcal{I})}{1+\epsilon},$$

where the first inequality follows from Lemma 25 and the second inequality follows from Lemma 24. □

6 Outlook

Motivated by PB scenarios in which it is useful to consider geographic constraints and thematic constraints, we enriched the standard approval-based model of PB by introducing a group structure over the projects and requiring group-specific budget limits for each group. We have showed that, while being computationally intractable in general, the enriching PB instances with such group structure and its corresponding budget constraints comes at essentially no computational cost if there are not so many such groups or if the structure of these groups is hierarchical or close to being such; we complemented our analysis with lower bounds and approximation algorithms. Practically, while our focus is on a theoretical understanding of the combinatorial structure of our problems, some of our results are already showing efficient complexity and thus can be used practically as they are. Other results giving an evidence of being in FPT or XP, while having rather high complexity (e.g., double exponential dependency on the relevant parameter), show nevertheless that efficient algorithms may exist, thus more research might be instructive in finding algorithms with even better running time.

Further research may concentrate on: solving our open question regarding the complexity of GROUP-PB wrt. g; considering other relevant parameters; study group-wise lower bounds (instead of upper bounds, as we do here), and enriching GROUP-PB by considering project interactions (e.g., in the spirit of Jain et al. [2020a]).

Acknowledgements

K. Sornat was partially supported by the Foundation for Polish Science (FNP) within the START programme, the National Science Centre, Poland (NCN; Grant No. 2018/28/T/ST6/00366) and the Israel Science Foundation (ISF; Grant No. 630/19). N. Talmon was supported by the Israel Science Foundation (ISF; Grant No. 630/19). M. Zehavi was supported by the Israel Science Foundation (ISF; Grant No. 1176/18) and the United States-Israel Binational Science Foundation (BSF; Grant No. 2018302). This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (ERC; Grant No. 101002854).

References

[Aziz and Shah, 2021] Haris Aziz and Nisarg Shah. Participatory budgeting: Models and approaches. In Tamás Rudas and Gábor Péli, editors, *Pathways Between Social Science and Computational Social Science: Theories, Methods, and Interpretations*, pages 215–236. Springer, 2021.

[Aziz *et al.*, 2018] Haris Aziz, Barton E. Lee, and Nimrod Talmon. Proportionally representative participatory budgeting: Axioms and algorithms. In *Proceedings of AAMAS'18*, pages 23–31, 2018.

[Benabbou *et al.*, 2018] Nawal Benabbou, Mithun Chakraborty, Xuan-Vinh Ho, Jakub Sliwinski, and Yair Zick. Diversity constraints in public housing allocation. In *Proceedings of AAMAS'18*, pages 973–981, 2018.

[Bredereck *et al.*, 2014] Robert Bredereck, Jiehua Chen, Piotr Faliszewski, Jiong Guo, Rolf Niedermeier, and Gerhard J. Woeginger. Parameterized algorithmics for computational social choice: Nine research challenges. *Tsinghua Science and Technology*, 19(4):358–373, 2014.

[Bredereck *et al.*, 2020] Robert Bredereck, Piotr Faliszewski, Rolf Niedermeier, Piotr Skowron, and Nimrod Talmon. Mixed integer programming with convex/concave constraints: Fixed-parameter tractability and applications to multicovering and voting. *Theor. Comput. Sci.*, 814:86–105, 2020.

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

[Celis *et al.*, 2018] L. Elisa Celis, Lingxiao Huang, and Nisheeth K. Vishnoi. Multiwinner voting with fairness constraints. In *Proceedings of IJCAI'18*, pages 144–151, 2018.

[Chierichetti *et al.*, 2017] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. In *Proceedings of NIPS'17*, pages 5029–5037, 2017.

[Gupta *et al.*, 2020a] Sushmita Gupta, Pallavi Jain, Sanjukta Roy, Saket Saurabh, and Meirav Zehavi. Gehrlein stability in committee selection: Parameterized hardness and algorithms. *Auton. Agents Multi Agent Syst.*, 34(1):27, 2020.

[Gupta *et al.*, 2020b] Sushmita Gupta, Pallavi Jain, and Saket Saurabh. Well-structured committees. In *Proceedings of IJCAI'20*, pages 189–195, 2020.

[Hershkowitz *et al.*, 2021] D. Ellis Hershkowitz, Anson Kahng, Dominik Peters, and Ariel D. Procaccia. District-fair participatory budgeting. In *Proceedings of AAAI'21*, 2021.

[Ianovski, 2019] Egor Ianovski. Electing a committee with constraints. *CoRR*, abs/1902.05909, 2019.

[Izsak *et al.*, 2018] Rani Izsak, Nimrod Talmon, and Gerhard J. Woeginger. Committee selection with intraclass and interclass synergies. In *Proceedings of AAAI'18*, pages 1071–1078, 2018.

[Jain *et al.*, 2020a] Pallavi Jain, Krzysztof Sornat, and Nimrod Talmon. Participatory budgeting with project interactions. In *Proceedings of IJCAI'20*, pages 386–392, 2020.

[Jain *et al.*, 2020b] Pallavi Jain, Krzysztof Sornat, Nimrod Talmon, and Meirav Zehavi. Participatory budgeting with project groups. *CoRR*, abs/2012.05213, 2020.

[Kellerer *et al.*, 2004] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*. Springer, 2004.

[Lenstra, 1983] Hendrik W. Lenstra. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983.

[Patel *et al.*, 2021] Deval Patel, Arindam Khan, and Anand Louis. Group fairness for knapsack problems. In *Proceedings of AAMAS'21*, pages 1001–1009, 2021.

[Shah, 2007] Anwar Shah, editor. *Participatory Budgeting. Public Sector Governance and Accountability*. World Bank, 2007.

[Talmon and Faliszewski, 2019] Nimrod Talmon and Piotr Faliszewski. A framework for approval-based budgeting methods. In *Proceedings of AAAI'19*, pages 2181–2188, 2019.

[Tsang *et al.*, 2019] Alan Tsang, Bryan Wilder, Eric Rice, Milind Tambe, and Yair Zick. Group-fairness in influence maximization. In *Proceedings of IJCAI'19*, pages 5997–6005, 2019.

[Yang and Wang, 2018] Yongjie Yang and Jianxin Wang. Multiwinner voting with restricted admissible sets: Complexity and strategyproofness. In *Proceedings of IJCAI'18*, pages 576–582, 2018.