Participatory Budgeting with Project Interactions

Pallavi Jain, **Krzysztof Sornat** and **Nimrod Talmon**
Ben-Gurion University, Israel
{pallavi, sornat}@post.bgu.ac.il, talmonn@bgu.ac.il

Abstract

Participatory budgeting systems allow city residents to jointly decide on projects they wish to fund using public money, by letting residents vote on such projects. While participatory budgeting is gaining popularity, existing aggregation methods do not take into account the natural possibility of project interactions, such as substitution and complementarity effects. Here we take a step towards fixing this issue: First, we augment the standard model of participatory budgeting by introducing a partition over the projects and model the type and extent of project interactions within each part using certain functions. We study the computational complexity of finding bundles that maximize voter utility, as defined with respect to such functions. Motivated by the desire to incorporate project interactions in real-world participatory budgeting systems, we identify certain cases that admit efficient aggregation in the presence of such project interactions.

1 Introduction

Participatory budgeting (PB) [Cabannes, 2004] is a direct democracy approach for budgeting, most often used to decide a fraction of municipal budgets. Rooted in Brazil [Wampler, 2010], it keeps on gaining popularity as more and more cities are using it to decide on the distribution of increasing fractions of their mutual funds; in particular, it is used quite extensively in the United States [Russon Gilman, 2012], in Europe [Sintomer *et al.*, 2008], and elsewhere [Ganuza and Baiocchi, 2012].

The most popular ballot type for PB is approval (as it offers easiness of elicitation), thus our starting point is a standard model for approval-based participatory budgeting, in which we are given a set of projects, each with its cost; a collection of voters, each approving a subset of projects; and a budget limit. The aggregation task is to select a bundle of projects whose total cost does not exceed the budget limit, respecting voter preferences. There are quite a number of aggregation methods for approval-based participatory budgeting. In particular, Goel et al. [2019] study variants of the popular aggregation methods for this setting, jointly referred to as the *Greedy Approval* method (it is used, e.g., in Paris and Warsaw): Order the projects in decreasing order of their approval scores (the number of voters approving each of them); then, go over the list and fund items as long as the remaining budget limit suffices. Talmon and Faliszewski [2019] study a family of aggregation methods for this setting; and Aziz et al. [2018] consider proportional representation in this context.

The motivation for our work is the fact that, while many times there are certain interactions between projects, this important aspect of participatory budgeting is often ignored. Consider the following toy example of a participatory budgeting instance, containing the following set of proposed projects: Project p_1 - build a school in road A; project p_2 - build a school in road B; and project p_3 - improve the road from C to road A. Assume that road A and road B are quite close to each other, while C is quite far from them. Now, consider some voter v_1 living next to road A and road B, and wishing the city to build a school. In this case, it would be natural for voter v_1 to approve both p_1 and p_2. Note that, if there are many voters like v_1, then the greedy approval method might decide to fund both p_1 and p_2. However, this would not be a good use of public money, as road A and road B are very close to each other, thus perhaps having a school in either A or B would suffice. This is an extreme example of *project substitutions*, in which a set of projects (the set $\{p_1, p_2\}$ here) are of no use when selected together. To demonstrate a different kind of project interaction, consider some voter v_2 who lives close to road C, and assume that v_2 might be happy for a school to be built in road A, but only if the road from C to road A would be improved (as otherwise it would be hard for v_2 to attend school). In this case, it would be natural for voter v_2 to approve both p_1 and p_3. Note that, if there are many voters like v_2, but if, say, p_3 costs much more than p_1, then the greedy approval method might decide to fund p_1 but not fund p_3. Again, this would not be good use of public money as voters like v_2 perhaps could not attend the new school. Similarly, suppose school is at isolated place, then building only road would not be fruitful. This is an extreme example of *project complementarities*, in which a set of projects (the set $\{p_1, p_3\}$) are of no use when not selected together.

To the best of our knowledge, no system of PB used today deals with project interactions as demonstrated in the example above and there are no papers dealing with this.

There are, however, some works regarding multiwinner elections (MW) [Faliszewski *et al.*, 2017] (MW are PB elections with projects of unit cost) that consider some forms of interactions between committee members: Izsak et al. [2018] consider synergies between groups of candidates, albeit their model do not consider the interplay between candidate interactions and individual voter preferences. There are also works on diversity constraints in MW (see, e.g., [Izsak, 2017; Aziz, 2019]), however the interactions we consider here are different. (There's also some general relation to combinatorial auctions [De Vries and Vohra, 2003], in which there are some interactions, albeit for a different setting.)

While there could be many, complex types of project interactions in PB instances, here we concentrate on substitution and complementarity effects for groups of projects. Our basic intuition is that a voter liking several projects, all of which are substitutions of each other, would have diminishing marginal utility as more projects from this group will be funded. In a case of extreme substitution it means that the voter would get the same utility as long as at least one project from this group is funded. We model such project interactions within groups of projects by augmenting the model of PB described above to contain—in addition to the set of projects with their costs, the set of voters, and the budget limit— a partition of the set of projects. We refer to this partition as the *interaction structure*, with the intent that projects in the same part of the interaction structure enjoy some substitution or complementarity effect. Importantly, here we do not tackle the problem of how to come up with such interaction structures, but rather assume that it is given: We mention that, e.g., it might be decided by a decree of the city mayor; voted upon by the city council; or, say, decided upon by a full-fledged election of its own. Here we are interested in finding good aggregation methods that find better solutions by taking such interaction structures into account. Our approach is utilitarian, in that we define certain voter utilities—in particular, the utility of a voter depends on her approval set and the additional interaction structure—and study aggregation methods that aim at maximizing the total voter utility. While, indeed, our model does not catch *all* possibilities of interactions between projects, it does catch a quite rich set of possibilities. In particular, we model not only extreme substitution/complementarity effects, but, by parameterizing our utility function, we in effect accommodate various degrees of such effects.

As the corresponding aggregation methods we consider are generally computationally intractable, our focus is on identifying special cases, including wrt. parameters that might be small in applications, and design efficient algorithms for these, more restricted settings. We believe that some of our algorithms can be used in practice, and view our results as a progress towards tackling the important problem of project interactions in PB.

2 Formal Model

Throughout the paper, we denote the number of projects by m and the number of voters by n.

In our setting we have the following ingredients: a set of projects, $P = \{p_1, \dots, p_m\}$; a cost function, $c \colon P \to \mathbb{R}^+$, with $\mathrm{cost}(B) := \sum_{p \in B} c(p)$ for a set of projects $B \subseteq P$, referred to as the cost of B; a set of voters, $V = \{v_1, \dots, v_n\}$, where every voter v_i specifies the set of projects $v_i \subseteq P$ approved by her—her *approval set*. Note that we use v_i both for a voter and for her approval set; moreover, we call V a profile when referring to a collection of approval sets of all the voters in V; a partition Z of the set of projects (i.e., $Z = \{z_1, \dots, z_{|Z|}\}$ with $\cup_{i \in [|Z|]} z_i = P$ and $z_i \cap z_j = \emptyset$, $i, j \in [|Z|]$), referred to as the *interaction structure*; and a budget limit $\ell \in \mathbb{R}$. An aggregation method for our setting takes an instance of participatory budgeting with interactions, $E = (P, c, Z, \ell, V)$, and returns a *bundle* $B \subseteq P$ with $\mathrm{cost}(B) \le \ell$.

A natural utilitarian approach is to define a *utility* for a voter from a possible bundle to be the number of projects in the bundle that are approved by the voter and to define an aggregation method that outputs a bundle that maximizes the sum of voter utilities. (Note that, as observed by Talmon and Faliszewski [2019], the Greedy Approval method is a greedy approximation algorithm for maximizing the total voter utility defined in this way.) Here we take into account also the interaction structure Z; to do so we define an *interaction function* $f \colon \mathbb{Z}^+ \to \mathbb{R}$, understanding $f(i)$ as the utility that a voter v gets from a part of Z in which there are exactly i projects approved by v and funded by the bundle at hand: i.e., the utility of some voter v from some bundle B wrt. interaction structure Z and interaction function f is: $\mathrm{util}(v, B) := \sum_{z \in Z} f(|z \cap v \cap B|)$. Each f induces an aggregation method that selects a bundle B maximizes the sum of voter utilities (i.e., the *total utility*), denoted by $\mathrm{util}(B) := \sum_{v \in V} \mathrm{util}(v, B)$. Note that a voter could apply different interaction functions in different parts of Z. What is more, each voter could have different set of interaction functions. Some of our algorithmic results hold in this general setting, in particular, those algorithms which are based on enumeration (e.g. Theorem 4). Nonetheless, for the sake of presentation, we decided to use a fixed f to all the voters and all the parts. We consider only nonzero interaction functions satisfying the following:

1. $f(0) = 0$: This means that if a voter v does not approve any project from a part $z_i \in Z$ that is funded, then the utility of this voter from part z_i is 0; this is a normalization condition.
2. $f(i) \ge f(j)$ for $i > j$: This means that the interaction function is non-decreasing; this is natural as the utility of a voter shall not decrease if more of her approved projects are being funded.
3. $f(i) = 1$ for the first i for which $f(i) > 0$; this is a normalization condition.

We furthermore distinguish between two types of interaction functions: Concave (resp., convex) functions, in which $f(i+1) - f(i) \le f(i) - f(i-1)$ (resp., $f(i+1) - f(i) \ge f(i) - f(i-1)$), correspond to substitution effects (resp., complementarity effects) as concavity (resp. convexity) corresponds to diminishing (resp., increasing) marginal utilities. As such, these are referred to as *substitution functions* (resp., *complementarity functions*). When we deal with the decision version of our problem, in which we decide existence of a bundle with at least some given total utility, then we refer to f-PB (if f is an interaction function), f-PBSUB (if f is a substitution function), or f-PBCOMP (if f is a complementarity function). We refer to the corresponding optimization problems (in which we wish to identify a bundle maximizing total utility) as MAX-f-PB, MAX-f-PBSUB, or MAX-f-PBCOMP. Note that, in all problems, f is part of the problem name, and thus fixed.

Example 1. Let $P = \{a, b, c, d, e, f\}$ be a set of projects and $Z = \{\{a, b, c\}, \{d, e\}, \{f\}\}$ be a partition of P. Let $V = \{v_1, v_2\}$ with $v_1 = \{a, b, c\}$ and $v_2 = \{a, d, f\}$. Consider the substitution function $f(i) = \sum_{j=1}^{i} 1/j$. For the bundle $B = \{a, b, d\}$, the total utility is 3.5, as $\text{util}(v_1, B) = 1.5$ and $\text{util}(v_2, B) = 2$. If all projects cost 1 and the budget limit is 3, then MAX-f-PBSUB would select the bundle $\{a, d, f\}$, as it maximizes the total utility with, achieving total utility 4. Consider the complementarity function $f(i) = i^2$. For the bundle $B = \{a, b, d\}$, the total utility is 6, as $\text{util}(v_1, B) = 4$ and $\text{util}(v_2, B) = 2$. Here, if all projects cost 1 and the budget limit is 3, then MAX-f-PBCOMP would select the bundle $\{a, b, c\}$, as it maximizes the total utility, achieving total utility 10.

Paper structure. We study the possibility of identifying bundles maximizing total utility for various interaction functions. In Section 3, we show general intractability. Then, to better understand the combinatorics of our problems and to identify special cases admitting efficient algorithms, we proceed in 2 directions: In Section 4 we study the parameterized complexity of our problems wrt. problem parameters that might be small in real-world instances; and in Section 5 we study the complexity of our problems for instances of two domain restrictions suited for our setting. Section 6 concludes.

3 General Computational Complexity

f-PBSUB and f-PBCOMP generalize the standard model of approval-based PB by introducing an interaction function f. Talmon and Faliszewski [2019] study a utilitarian setting of approval-based PB without project interactions; in particular, they define a voting rule, denoted by $\mathcal{R}^{m}_{|B_v|}$, which maximizes voter utility, where the utility of a voter v from some bundle B is the number of funded projects approved by her (i.e., $|v \cap B|$). They show that $\mathcal{R}^{m}_{|B_v|}$ can be solved in polynomial time via a standard dynamic programming for Unary Knapsack [Talmon and Faliszewski, 2019, Proposition 1]. Next we identify two cases for which MAX-f-PB is equivalent to the no-interaction setting of $\mathcal{R}^{m}_{|B_v|}$; as such these are polynomial-time solvable. (Intuitively, the next result follows as $|Z| = m$ corresponds to the all-singletons interaction partition; and linear interaction functions nullify the interaction partition.)

Corollary 2. *If $|Z| = m$ or if f is the linear function, where m is the number of projects, then* MAX-f-PB *is polynomial-time solvable.*

However, as we show next, intractability appears even for a small deviation of f from linearity or of Z from the all-singletons partition.

Theorem 3. f-PB *is* NP*-complete and* W[1]*-hard wrt. the budget limit ℓ for every $|Z| \in \{1, 2, \ldots, \lfloor \epsilon m \rfloor\}$ even for constant app, where m is the number of projects and app is the maximum size of an approval set, for any $\epsilon \in (0, 1)$ and any non-linear interaction function f.*[1]

The proof of Theorem 3 follows by a reduction from the INDEPENDENT SET or CLIQUE problems on r-regular graphs, depending on how f deviates from being linear, with required total utility u different for the two cases. Edges correspond to voters, vertices correspond to projects. All projects are of unit cost. Each vertex-project is approved by exactly r edge-voters. All vertex-projects are in one part. By choosing vertex-projects we get utility depending on the number of satisfied voters (covered edges). If $f(1) = 1$ and $f(2) < 2$, then f deviates from linearity in a downward way and we get more utility if we maximize the number of edges covered once; hence, we are using reduction from INDEPENDENT SET). However, if $f(1) = 1$ and $f(2) > 2$ (or $f(1) = 0$ and $f(2) = 1$), then f deviates from linearity in an upward way and we get more utility if we maximize the number of edges covered twice (hence we are using reduction from CLIQUE). If the function is linear (or equal to 0) up to some point i (i is a constant as f is fixed), and then f deviates from linearity, then we add $i - 1$ dummy wonderful-projects that are approved by every voter and live in the same part as the vertex-projects and we increase the budget by $i-1$. We set the total utility in such a way that all dummy wonderful-projects are in every feasible solution, and the effect of f being linear (equal to 0) up to i is canceled out as every voter is satisfied by $i - 1$ dummy wonderful-projects. To provide hardness for the case with many parts we add dummy bad-projects that are not approved by any voter. Every bad-project belongs to a singleton part; then, there is no incentive to choose these projects to the solution. As i, as well as the number of vertex-projects, are fixed for a fixed graph and fixed function f, the number of bad-projects can be arbitrarily close to number of projects.

4 Fixed-Parameter Algorithms

Theorem 3 above shows that MAX-f-PB is generally NP-hard, and also implies hardness wrt. the budget limit ℓ and wrt. the maximum approval set size app. In this section we consider further input parameters, concentrating on these parameters: The number of projects: m; the number of voters: n; the number of parts in Z: $|Z|$; and the maximum projects in a part of Z: $s := \max_{z \in Z} |z|$.

First, observe that MAX-f-PB is FPT wrt. the number m of projects as we can try all possible bundles. Indeed, in some real-world PB instances m is rather small (e.g., most PB instances powered by Stanford Participatory Budgeting Platform[2] contain less than 20 projects). Note that assuming ETH, CLIQUE and INDEPENDENT SET do not admit $2^{o(\tilde{n}+\tilde{m})}$

[1]W[2]-hardness of f-PB wrt. ℓ, for $f(i) = \min(1, i)$ can follow by a reduction from Dominating Set [Bonnet *et al.*, 2016]. Hardness of f-PBSUB can also be derived by reducing from winner determination for non-decreasing non-constant OWA vectors [Skowron *et al.*, 2016, Theorem 5].

[2]https://pbstanford.org/

algorithm, where $\tilde{n}$ and $\tilde{m}$ are number of vertices and edges in the graph, respectively [Impagliazzo *et al.*, 2001]. Since in both the reductions (reduction from CLIQUE and INDEPENDENT SET), for $|Z| = 1$, number of vertices and edges in the graph are mapped to the number of projects (with some additional constant many projects) and the number of voters, assuming ETH, there is no $2^{o(n+m)}$ algorithm for MAX-f-PB, for any non-linear function f.

Regarding the parameter *app*, it is generally decided by the election organizer, and is indeed rather small (around 5-10 in most instances, e.g., Paris and Warsaw; and even 1 or 2, e.g., in Wrocław). Observe that MAX-f-PB is polynomial-time solvable if $app = 1$, however, there are some functions f for which MAX-f-PB is NP-hard already for $app = 2$ (in fact, any function f deviating from linearity for some value i is NP-hard as soon as $app = i + 1$ using Theorem 3), and remains polynomial time, otherwise.

Below we describe efficient algorithms for MAX-f-PB wrt. s, $m - |Z|$, and n.

4.1 Parameterization by s

In many cases, the parameter s—the maximum size of a part in the "interactions partition" Z—could be rather small; in particular, several projects of the same kind and in close geographical position might be in the same part of an interaction partition, however projects that are far away from each might probably should not be in the same part, so, if the projects are across the country, usually s is small. Here, we give an FPT algorithm wrt. s for MAX-f-PB, where $f : \mathbb{Z}^+ \to \mathbb{Z}^+$ and $f(h) \geq 1$, where h is the smallest positive integer for which $f(h)$ is non-zero. We call such function as *integer-valued function*. We restrict to those functions here as it is needed technically for the dynamic programming (DP) algorithm we use to prove the claim; note that such functions can nevertheless still encompass complicated substitution/complementarity effects, however to a lesser precision.

Theorem 4. MAX-f-PB *is* FPT *wrt. s, where s is the maximum number of projects in a part and f is an integer-valued function.*

Proof. Let (P, c, Z, ℓ, V) be the given instance of MAX-f-PB. Let $Z = \{z_1, \dots, z_q\}$. We define the dynamic programming table as follows: for $i \in [q]$, $j \in [nqf(s)] \cup \{0\}$, we define $T[i, j]$ as the minimum cost of a bundle that belongs to first i parts of Z and whose total utility is j. If no such bundle exists, then $T[i, j] = \infty$. Note that we have only $\mathcal{O}(nq^2 f(s))$ table entries. We compute our table entries as follows.

$$T[1,j] = \begin{cases} \text{cost}(B) & B \subseteq z_1 \text{ is a cheapest bundle} \\ & \text{with total utility } j, \text{ if exists} \\ \infty & \text{otherwise} \end{cases} \tag{1}$$

$T[1, j] = 0$, for all $j < h$, as the empty bundle is the only least cost bundle of total utility 0. Note that $T[1, j]$ is correct. Time taken to compute $T[1, j]$ is $2^{|z_1|}$ as we can check the cost of all subsets of z_1. For $i > 1$, we compute the table entry recursively as follows. For $S \subseteq P$, let u_S denote the total utility of S.

$$T[i,j] = \min_{\substack{S \subseteq z_i: \\ u_S \leq j}} \{T[i-1, j - u_S] + \text{cost}(S)\} \tag{2}$$

The correctness of Equation 2 follows from the fact that if $B \subseteq z_1 \cup \ldots \cup z_i$ is a least cost bundle with total utility j, then $B \setminus z_i$ is a potential candidate for $T[i-1, j - u_{B \cap z_i}]$. Similarly, for $S \subseteq z_i$, if $B' \subseteq z_1, \dots, z_{i-1}$ is a bundle with total utility $j - u_S$, then $B' \cup S$ is a potential candidate for $T[i, j]$. The maximum total utility that can be achieved with budget ℓ is equal to $\max\{j \in [nqf(s)] : T[q, j] \leq \ell\}$. ☐

4.2 Parameterization by $m - |Z|$

As argued above, there are scenarios in which the size of every part in Z is small, thus $|Z|$ is large, and $m - |Z|$ is small. Here, we describe an FPT algorithm wrt. $m - |Z|$. Recall that using Corollary 2, when $|Z| = m$ (that is, there are no interactions at all), MAX-f-PB can be solved in polynomial time. Our algorithm works as follows. Let (P, c, Z, ℓ, V) be the given instance of MAX-f-PB. First, note that there are at most $m - |Z|$ parts in Z that contain more than one project each, and, thus, at most $m - |Z|$ projects which belongs to such parts (part that contain more than one project). Therefore, we can guess the projects from these parts in the solution bundle ($2^{m-|Z|}$ guesses, in total), delete these parts from Z (this means that we also delete projects in these parts from P and every approval set) and solve the reduced instance using Corollary 2. Thus, we have the following:

Theorem 5. MAX-f-PB *is* FPT *wrt. $m - |Z|$, where m is the number of projects.*

4.3 Parameterization by n

We furthermore consider the number n of voters as it is a basic parameter of our instances, and since PB is useful not only on a city-level (in which usually all residents are eligible voters, thus n is rather large), but also on, say, an apartment-level (consider PB used by a house/building committee), and on a city-level in which, say, only members of the city council are eligible for voting. We describe efficient algorithms wrt. n for interaction functions that eventually become constant. Here, also we consider integer-valued functions as we will use Theorem 4 as subroutine.

Definition 6 (Eventual-constant interaction functions)**.** An interaction function f is *eventual-constant* if there exists some $i \in \mathbb{N}$ such that $f(j) = f(i)$ for each $j > i$.

First, we apply the following reduction rule, exhaustively; i.e., we *preprocess* the given instance and bound the number of projects in every part of Z by $\mathcal{O}(2^n)$.

Reduction Rule 7. *Let (P, c, Z, ℓ, V) be the given instance of* MAX-f-PB*. Suppose that there exists $i \in \mathbb{N}$ such that for all $j > i$, $f(j) = f(i)$. If there exists more than i projects, say $p_1, \dots, p_{i+1}$ in a part of Z such that they are approved by the same set of voters and $c(p_1) \leq \ldots \leq c(p_{i+1})$, then delete p_{i+1} from P, Z and from every approval set. Let the reduced instance be (P', c, V', Z', ℓ).*

The correctness of Reduction Rule 7 follows from the fact that there exists an optimal solution B to (P, c, Z, ℓ, V) that does not contain p_{i+1} as $f(i) = f(i + 1)$. Thus, B is also an optimal solution to (P', c, V', Z', ℓ). Similarly, we can argue the other direction. Note that, after the exhaustive application of Reduction Rule 7, for every subset of voters, there is at most i projects in a part of Z which is approved by only these voters. Moreover, since f is fixed, every part of Z contains at most $\mathcal{O}(2^n)$ projects. Thus, due to Theorem 4, we have following:

Theorem 8. *For any eventual-constant function f,* MAX-f-PB *is* FPT *wrt. n, where n is the number of voters and f is an integer-valued function.*

FPT wrt. n holds also for (not necessarily eventual-constant) concave functions. The proof involves a DP on top of the MILP technique of Bredereck et al. [2020].

Theorem 9. MAX-f-PBSUB *is* FPT *wrt. n, where n is the number of voters and f is an integer-valued function.*

Proof sketch. The overall algorithm is a DP similar to the one used in the proof of Theorem 4 with the crucial difference that, instead of checking all subsets of a part in a brute-force way, here we solve a Mixed Integer Linear Program (MILP). As the number of integer variables in the MILP is upper-bounded by a function depending only on the parameter n, FPT follows by the result of Lenstra [1983] (as discussed and used by Bredereck et al. [2020]). In the MILP we partition P into $2^n - 1$ subsets, each indexed by a voter subset. Then, we have integer variables corresponding to number of projects from each such part taken to the solution. We then need to encode (1) cost constraint; (2) utility constraint. As the cost constraint can be encoded as minimizing a convex function and the utility as maximizing a concave function, we can use an encoding following the technique of Bredereck et al. [2020] to use only additional real-valued variables.

In the overall DP, we define $T[i,j]$ in the same way as in Theorem 4, i.e., $T[i,j]$ is the minimum cost of a bundle that belong to first i parts of Z and whose total utility is at least j. If no such bundle exists, then $T[i,j] = \infty$. Note that f is concave hence $f(m) \leq mf(1) = \mathcal{O}(m)$ and we have $\mathcal{O}(n|Z|^2 f(m)) \subseteq \mathcal{O}(nm^3)$ table entries in total. We compute our table entries as follows.

$$T[1,j] = \text{MILP}(z_1, j)$$
$$T[i,j] = \min_{u \in \{0,1,\ldots,j\}} \{T[i-1, j-u] + \text{MILP}(z_i, u)\}$$

Hence, the maximum total utility that can be achieved with budget ℓ is equal to $\max\{j \in [nm^2] : T[|Z|, j] \leq \ell\}$.

We are filling $\mathcal{O}(nm^3)$ entries in the table, each solving $\mathcal{O}(nm^2)$ many MILPs, each in time $\mathcal{O}^\star(2^{2^{\mathcal{O}(n)}})$. Hence the total time is $\mathcal{O}^\star(2^{2^{\mathcal{O}(n)}})$, thus FPT wrt. n. □

5 Restricted Domains

In our quest for identifying tractable cases for MAX-f-PB, we extend our study by considering the computational complexity of MAX-f-PB for various domain restrictions that model extreme cases of situations that might be natural to occur in participatory budgeting with interactions. Here also we restrict our study to integer-valued functions to identify tractability. In particular, we consider two domain restrictions for PB.

5.1 Restricting to r-DISTRICT

The first domain restriction we consider, r-DISTRICT, models a setting in which each voter approves only projects from her district (more generally, it models a situation in which voters do not care for projects that are not directly relevant to them); note that, in its formal definition below, we consider an additional partition of the projects that does not have to be equal to Z.

Definition 10 (r-DISTRICT). A profile V satisfies r-DISTRICT if the set of projects P can be partitioned into subsets $P_1, \ldots, P_k$ such that $|P_i| \leq r$, $i \in [k]$, and each approval ballot $v \in V$ is a subset of exactly one P_i, $i \in [k]$.

Assuming a fixed number r of proposed projects from a district, a solution can be found in polynomial time.

Theorem 11. MAX-f-PB *under* r-DISTRICT *can be solved in polynomial time, where f is an integer-valued function.*

Our algorithm is similar in spirit to the dynamic programming algorithm in Theorem 4, as here also we can do dynamic programming over the subsets of $P_1, \ldots, P_k$. Here, the intuition is that since approval set of a voter is a subset of P_i, where $i \in [k]$, we can guess the projects funded from each P_i in polynomial time (as r is fixed), and do the similar dynamic programming as in Theorem 4.

5.2 Restricting to PARTITION

The second domain restriction we consider, PARTITION, is an extreme case of r-DISTRICT in which every voter approves all the projects in her district. Here we do not care for the maximum size of a district. One motivation for this domain restriction comes from political parties: If only parties suggest projects, and voters follow party lines by supporting the projects suggested by their party, then the profile would satisfy PARTITION. (We mention that this domain restriction is studied by Elkind and Lackner [2015], albeit not for participatory budgeting.)

Definition 12 (PARTITION). We say that a profile $\mathcal{D}$ satisfies PARTITION if the projects in P can be partitioned into $P_1, \ldots, P_k$ such that each voter approves one of the sets $P_1, \ldots, P_k$.

Next we describe a polynomial-time algorithm for MAX-f-PBSUB, for PARTITION instances, where f is an integer-valued function. Our algorithm is, in essence, a nested dynamic programming: The main DP is used to find the cheapest bundle that is a subset of P_i, $i \in [k]$, for all possible utilities (Lemma 13); this DP uses a "nested" DP that is used to find a bundle that maximises the utility using the solution of sub-problems computed above.

Lemma 13. *Let* (P, c, Z, ℓ, V, u)[3] *be an instance of* f-PBSUB*, where f is an integer-valued function, such that every voter approves all projects. Then, one can find a least cost bundle with utility u in polynomial time, if exists.*

Proof. We describe a dynamic programming algorithm. Let $Z = \{z_1, \ldots, z_q\}$. In the table, T, each entry $T[i,j]$, where

[3]We slightly abuse notation and use f-PBSUB to *find* a solution with utility u.

$i \in [q], j \in [u] \cup \{0\}$ is the minimum cost of a bundle which is a subset of $z_1, \ldots, z_i$ with total utility u. We compute the table entries as follows. Let $B_{i,j}$ be a set of x least cost projects in z_i such that $|V| \cdot f(x) = j$, if exists.

$$T[1, j] = \begin{cases} \mathrm{cost}(B_{1,j}) & \text{if } B_{1,j} \text{ exists} \\ \infty & \text{otherwise} \end{cases}$$

For $i > 1$, we compute the table entries as follows.

$$T[i, j] = \min_{\substack{1 \le h \le j: \\ B_{i,(j-h)} \text{ exists}}} \{T[i-1, h] + \mathrm{cost}(B_{i,(j-h)})\} \quad (3)$$

Correctness follows in a similar way as in Theorem 4. □

We use Lemma 13 for designing a polynomial-time algorithm for MAX-f-PBSUB for PARTITION instances, where f is an integer-valued function. Let $(\mathcal{P} = \{P_1, \ldots, P_k\}, c, Z, \ell, V)$ be the given instance of MAX-f-PBSUB under PARTITION. Let V_{P_i} be the set of voters who approve the projects in P_i and $Z|_{P_i}$ be the partition obtained after removing all the projects from parts in Z that are not in P_i. Intuitively, if B is a solution to $(\mathcal{P} = \{P_1, \ldots, P_k\}, c, V, Z, \ell)$ such that the utility of every voter in V_{P_i} from B is u_i, then the total utility for B is $u_1 + \ldots + u_k$ as no voter approves projects from different parts in $\mathcal{P}$, and we can find a least cost bundle for the instance $(P_i, c, V_{P_i}, Z|_{P_i}, u)$ of f-PBSUB using Lemma 13.

Theorem 14. MAX-f-PBSUB *under* PARTITION *can be solved in polynomial time, where f is an integer-valued function.*

Proof. Let $(\mathcal{P} = \{P_1, \ldots, P_k\}, c, V, Z, \ell)$ be the given instance of MAX-f-PBSUB under PARTITION. Here, we do dynamic programming over subsets of $P_1, \ldots, P_k$. For each P_i and j, where $i \in [k]$, $0 \le j \le n|Z|f(m)$, $T[i, j]$ is the least cost of a bundle which is a subset of P_i with total utility j. Note that $T[i, j]$ can be computed using algorithm in Lemma 13 with instance $(P_i, c, V_{P_i}, Z|_{P_i}, \ell, u_j)$. Next, we create a table $\tilde{T}$, where each entry $\tilde{T}[i, j]$ is the least cost of a bundle which is a subset of $P_1, \ldots, P_i$ with total utility j as follows. For all $0 \le j \le n|Z|f(m)$, $\tilde{T}[1, j] = T[i, j]$. For $i > 1$, we compute the table entries as follows.

$$\tilde{T}[i, j] = \min_{0 \le j' \le j} \{\tilde{T}[i-1, j'] + T[i, j-j']\} \quad (4)$$

The correctness follows from the disjointness of parts in $\mathcal{P}$. Since both the tables T and $\tilde{T}$ have at most $nk|Z|f(m)$ entries, hence polynomial size as f is concave (similarly as argued in Theorem 9). Since each entry can be computed in polynomial time, the algorithm runs in polynomial time. □

6 Outlook

We proposed a model that incorporates certain kinds of interactions between projects in participatory budgeting. We believe that this aspect is of great importance and, when taken into account, can improve the quality of the selected bundles, thus increasing the social welfare (i.e., the total utility). Acknowledging that there might be many forms of project interactions, we tackled a certain kind of project interactions that corresponds to various degrees of substitutions and complementarities between groups of projects.

While we showed that finding optimal bundles is generally intractable—in particular, the problem becomes hard as soon as one deviates from the singletons-only interaction partition and from the linear interaction function—we identified several well-motivated special cases that admit efficient algorithms: We showed that our problems are FPT wrt. either $m - |Z|$ or s; FPT wrt. n if interactions are limited to eventual-constant concave functions; and polynomial-time solvable for profiles satisfying r-District or Partition.

Approximability. Another natural way at coping with the computational hardness of our problems is by approximation algorithms. First, as one can reduce an instance with $|Z|$ parts and n voters to an instance with 1 part and $n|Z| \le nm$ projects (by splitting each voter to $|Z|$-many voters), $|Z|$ does not affect approximability in time polynomial on nm; hence, below we consider only $|Z| = 1$.[4] Approximability of MAX-f-PB was only considered (under different names) for specific functions out of which we imply that (1) MAX-f-PBSUB admits a $(1-1/e)$-approximation due to submodularity of concave function f and the result of Sviridenko [2004]; (2) Assuming Gap-ETH, for any $\epsilon > 0$ and any function T there is no $T(k)(n+m)^{o(k)}$-time algorithm for MAX-f-PBSUB to within a factor of $(1 - 1/e + \epsilon)$—already for $f(i) = \min(i, 1)$ and unit-cost projects [Manurangsi, 2020]; (3) Assuming Gap-ETH, there is no polynomial time approximation algorithm for MAX-f-PB—for every function f with $f(1) = 0$—with approximation guarantee better than $n^{o(1)}$. This follows from the reduction in Theorem 3 and hardness of approximation of Densest k-Subgraph [Manurangsi, 2017] similarly as it was done in [Skowron *et al.*, 2016]. Good approximation for convex functions with $f(1) = 1$, or better ratios for specific concave functions are open. Note that recently, for the case of unit-cost projects, a tight approximation for $f(q) = \sum_{i=1}^{q} 1/q$ (and any *geometrically dominant* function) was shown [Dudycz *et al.*, 2020].

Future research. We believe that our paper opens the way for other approaches to incorporate project interactions to participatory budgeting. In particular, the main concrete avenues for immediate future research are exploring other cases in which PB with interactions can be solved efficiently (perhaps assuming that all approval sets individually satisfy the budget constraint); considering other utility functions; and, generally, considering other (perhaps non utilitarian) ways to incorporate project interactions to the PB setting.

Acknowledgements

The research was supported by the Israel Science Foundation (grant no. 630/19). K. Sornat was partially supported by the Foundation for Polish Science (FNP) within the START programme and the National Science Centre, Poland (grant no. 2018/28/T/ST6/00366). We would like to thank the anonymous reviewers for their helpful comments.

[4]Note that this reduction is not useful when parameterizing by the number n of voters.

References

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

[Aziz, 2019] Haris Aziz. A rule for committee selection with soft diversity constraints. *Group Decis. Negot.*, 28(6):1193–1200, 2019.

[Bonnet *et al.*, 2016] Édouard Bonnet, Vangelis Th. Paschos, and Florian Sikora. Parameterized exact and approximation algorithms for Maximum k-Set Cover and related satisfiability problems. *RAIRO - Theor. Inf. and Applic.*, 50(3):227–240, 2016.

[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. *Environ. Urban.*, 16(1):27–46, 2004.

[De Vries and Vohra, 2003] Sven De Vries and Rakesh V Vohra. Combinatorial auctions: A survey. *INFORMS J. Comput.*, 15(3):284–309, 2003.

[Dudycz *et al.*, 2020] Szymon Dudycz, Pasin Manurangsi, Jan Marcinkowski, and Krzysztof Sornat. Tight approximation for Proportional Approval Voting. In *IJCAI*, 2020.

[Elkind and Lackner, 2015] Edith Elkind and Martin Lackner. Structure in dichotomous preferences. In *IJCAI*, pages 2019–2025, 2015.

[Faliszewski *et al.*, 2017] Piotr Faliszewski, Piotr Skowron, Arkadii Slinko, and Nimrod Talmon. Multiwinner voting: A new challenge for social choice theory. In U. Endriss, editor, *Trends in Computational Social Choice*. AI Access Foundation, 2017.

[Ganuza and Baiocchi, 2012] Ernesto Ganuza and Gianpaolo Baiocchi. The power of ambiguity: How participatory budgeting travels the globe. *J. Public Deliberation*, 8(2):8:1–8:12, 2012.

[Goel *et al.*, 2019] Ashish Goel, Anilesh K. Krishnaswamy, Sukolsak Sakshuwong, and Tanja Aitamurto. Knapsack voting for participatory budgeting. *ACM Trans. Economics and Comput.*, 7(2):8:1–8:27, 2019.

[Impagliazzo *et al.*, 2001] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

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

[Izsak, 2017] Rani Izsak. Working together: Committee selection and the supermodular degree. In *AAMAS*, pages 103–115, 2017.

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

[Manurangsi, 2017] Pasin Manurangsi. Almost-polynomial ratio ETH-hardness of approximating Densest k-Subgraph. In *STOC*, pages 954–961, 2017.

[Manurangsi, 2020] Pasin Manurangsi. Tight running time lower bounds for strong inapproximability of Maximum k-Coverage, Unique Set Cover and related problems (via t-Wise Agreement Testing Theorem). In *SODA*, pages 62–81, 2020.

[Russon Gilman, 2012] Hollie Russon Gilman. Transformative deliberations: Participatory budgeting in the United States. *J. Public Deliberation*, 8(2):11:1–11:20, 2012.

[Sintomer *et al.*, 2008] Yves Sintomer, Carsten Herzberg, and Anja Röcke. Participatory budgeting in Europe: Potentials and challenges. *Int. J. Urban Reg. Res.*, 32(1):164–178, 2008.

[Skowron *et al.*, 2016] Piotr Skowron, Piotr Faliszewski, and Jérôme Lang. Finding a collective set of items: From proportional multirepresentation to group recommendation. *Artif. Intell.*, 241:191–216, 2016.

[Sviridenko, 2004] Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.*, 32(1):41–43, 2004.

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

[Wampler, 2010] Brian Wampler. *Participatory Budgeting in Brazil: Contestation, Cooperation, and Accountability*. Penn State Press, 2010.