Partition Aggregation for Participatory Budgeting

Pallavi Jain
Indian Institute of Technology Jodhpur
Jodhpur, India
pallavi@iitj.ac.in

Nimrod Talmon
Ben-Gurion University of the Negev
Be'er Sheva, Israel
talmonn@bgu.ac.il

Laurent Bulteau
LIGM, CNRS, Univ Gustave Eiffel
Marne-la-Vallée, France
laurent.bulteau@upem.fr

ABSTRACT

Recently, Jain et al. [IJCAI, 2019] studied the effect of project interactions in participatory budgeting (PB) by assuming an existing partition of the projects to interaction structures, namely a grouping of the projects into substitution and complementarity groups. Motivated by their study, here we take voter preferences to find such interaction structures. In our model, voters submit interaction structures, and the goal is to find an aggregated structure. Formally, given a set P of m projects, and n partitions of P, the task is to aggregate these n partitions into one aggregated partition. We consider this partition aggregation task both for substitution structures and for complementarity structures, studying several aggregation methods for each, including utility-based methods and Condorcet-based methods; we evaluate these methods by analyzing their computational complexity and their behavior with respect to certain relevant axiomatic properties.

KEYWORDS

Participatory Budgeting; Project Interactions; Computational Complexity; Axiomatic analysis

ACM Reference Format:
Pallavi Jain, Nimrod Talmon, and Laurent Bulteau. 2021. Partition Aggregation for Participatory Budgeting. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

In participatory budgeting (PB) [6] the task is to aggregate voter preferences over a set of projects, to decide upon a bundle of those projects to fund. It has received quite extensive recent attention from the research community, resulting in some aggregation methods to be used for such settings [2, 3, 5, 10, 11]. One aspect of PB which was neglected for long time, and recently studied by Jain et al. [14], is project interactions; in particular, two important ways by which projects can interact are by being substitutes or being complementary.

Example 1.1. As an example of a *substitution effect*, consider a toy PB instance consisting of 3 projects – one school, s, and two parks, p_1 and p_2; in many cases, it is natural to assume that, while certain voters might wish to have one park built in their city, not many voters would feel that funding two parks (especially if these parks are geographically close to each other) is a good use of public funds, as these two projects are *substitutes* of each other.

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

Example 1.2. As an example of a *complementarity effect*, consider a toy PB instance consisting of 3 projects – two schools, s_1 and s_2, and another project p that is a road leading from the city center to s_1; in many cases, it is natural to assume that many of the voters that wish to have the school s_1 being built would only wish so if also the road p leading to it would be funded; put differently, it is perhaps not a good use of public funds to fund only s_1 or only p, as these two projects are *complementary* to each other.

In this paper we are interested in figuring out the so-called *interaction structure* (we refer to interaction structure as both substitution structure and complementarity structure, whenever it is clear from the context) of the set of projects in a PB instance. E.g., in the toy examples described above, a natural corresponding substitution structure of the substitution example would be the partition $\{\{s\}, \{p_1, p_2\}\}$ – each part (i.e., $\{s\}$ and $\{p_1, p_2\}$) of this partition is referred to a *substitution class*; where a natural corresponding complementarity structure of the complementarity example would be the partition $\{\{s_1, p\}, \{s_2\}\}$ – each part (i.e., $\{s_1, p\}$ and $\{s_2\}$) of this partition is referred to a *complementarity class*.

Why are such interaction structures important? Here we take a utilitarian approach, by assuming that each voter has a utility for each set of projects selected for funding; say, the utility of voter u from a possible bundle P' is $u(P')$. Then, substitution effects might correspond to saying that the utility function u is submodular wrt. projects of the same part in the partition: e.g., in Example 1.1, $u(\{p_1, p_2\}) \leq u(\{p_1\}) + u(\{p_2\})$. Similarly, complementarity effects might correspond to supermodular utility functions. Thus, to be able to find a feasible bundle as the winning bundle of a given PB instance, it is useful to know the interaction structures of the instance. There are several ways by which an organizer of a PB instance might tackle this aspect of project interactions:

Dictatorial decision: One possibility would be for the organizer to decide herself upon the interaction structure – then, she might, e.g., ask voters to approve projects, but does not allow voters to approve more than one project in each substitution class, and require voters that wish to approve projects that are in a complementarity class together, to approve the whole class as well.

Explicit Exponential Elicitation: Another possibility would be for the organizer to allow each voter to explicitly specify her utility from each bundle of projects, however this would mean an exponential explosion and thus is usually not feasible.

Preliminary Election: A yet another possibility would be for the organizer to perform a preliminary election, in which she asks the voters – perhaps only a subset of the voters – to provide their interaction structures. Then, the organizer

can aggregate those partitions provided by the voters and use the aggregated partition as the global interaction structure (and then proceed, say, by similar ways as the dictator above uses – forbidding approving several projects from the same substitution class and enforcing approving whole complementarity classes together).

Here we choose the last option, thus concentrate on the subproblem of aggregating partitions. Here, one would be wondering why do we want to conduct election in two phases, that is, first asking for interaction structures from some voters, and then conduct standard election. The first reason is that elicitation cost is high. Since this is the preliminary step of participatory budgeting, it is natural that we only ask some people to provide us with their interaction structures, say only secretaries of different societies. Jain et al. [14] proposed the methods of using the aggregated partition in the subsequent participatory budgeting.

Here we do not study the problem of aggregating substitution classes and complementarity classes together, but we study two related, but formally different problems: first we consider the problem of aggregating substitution structures, and then, independently, we consider the problem of aggregating complementarity classes. Note that, indeed, these two problems regards aggregating partitions, however, we formulate the aggregation objective differently, as substitution effects differ greatly from complementarity effects.

Remark 1. *We acknowledge one natural criticism of our approach, namely that we assume that the interaction structures are "global", in the sense that it can be fixed to be the same for all voters. While, indeed, this might not always be the case, we believe that in most PB instances it is roughly global. Verifying this intuition and identifying cases in which this intuition is true and other cases in which it is violated is, again, an interesting avenue for future research.*

Remark 2. *In this paper we study how to aggregate partitions where the aggregated partitions are to be used for deciding on interaction structures for PB instances. Another PB scenario in which aggregating partitions might be useful is the following: A PB organizer shall distribute a pamphlet explaining the projects to the voters. On each page only a certain number of projects can be explained, thus in fact the pamphlet partitions the projects into pages. Using a partition aggregation method might help here, in particular, as the presentation affects the preferences.*

1.1 Related Work

A special case of the partition aggregation problem is *cluster ensembles* [16] which is also known as *cluster aggregation* [13] and *consensus clustering* [7]. In cluster aggregation, we are given a set of clusterings, and the goal is to find a clustering which agrees with the input clusterings as much as possible. Cluster aggregation is polynomial-time solvable when the input has two partitions, while it is APX hard when we have three partitions [4]. Strehl and Ghosh [16] proposed some techniques for cluster aggregation. In one of their approaches, given a set of clustering, they construct a hypergraph, and find a hyperedge separator that partitions the hypergraph into k unconnected components of approximately the same size.

Gionis et al. [13] gave some approximation algorithms. They considered cluster aggregation problem and correlation clustering. In cluster aggregation, they measure the dissimilarity between the clusterings. Let V be the given set of objects, and $C_1, \ldots, C_m$ be the set of clusterings. For two objects u and v in V, and two clusterings C_1, C_2, $d_{u,v}(C_1, C_2) = 1$, if u and v are in same part in C_1 and different parts in C_2, or vice-versa, otherwise 0. The dissimilarity between two clusterings C_1 and C_2 is defined as $d(C_1, C_2) = \sum_{u,v \in V} d_{u,v}(C_1, C_2)$. The goal is to find a clustering C such that the total dissimilarity, $\sum_{i=1}^{m} d(C, C_i)$, is minimized. This function is also known as total Mirkin distance. They also studied the maximization version of consensus clustering. Towards this, they defined similarity between two partitions C_1 and C_2, denoted by $s(C_1, C_2)$, as the number of objects which either belong to the same part in both the partitions or in different parts in both partitions. The goal is to find a partition C such that $\sum_{i=1}^{m} s(C_i, C)$ is maximised.

Dornfelder et al. [7] proved that cluster aggregation is NP-hard even when every partition contains at most two clusters. They proposed an FPT algorithm for cluster aggregation with respect to parameter average Mirkin distance. They also studied the local search variant of the problem, and showed that the problem is W[1]-hard when parameterized by radius of the Mirkin-distance neighborhood.

We also mention work on aggregating graphs [8], as aggregating partitions is equivalent to aggregating cluster graphs (graphs that are a collection of disjoint cliques).

1.2 Contributions

Our contributions are as follows:

- We describe a model of partition aggregation that is motivated by project interactions in PB instances, where the result of the aggregation process is an *interaction structure*, namely either a *substitution structure* or a *complementarity structure*.
- We study both substitution structures and complementarity structures. For each, we describe some axiomatic properties desired from aggregation methods for them. Importantly, as substitutions differ from complementarities, our axiomatic properties are different for the two cases.
- We offer some natural aggregation methods for substitution structures and complementarity structures, study their axiomatic properties, and investigate their computational complexity.

We summarise our results in Table 1.

2 PRELIMINARIES

We discuss PB, partitions, and our general utilitarian approach.

2.1 Projects and Partitions

Formally, we have a set, $P = \{p_1, \ldots, p_m\}$, of projects, (In PB, there is a cost $c(p)$ for each $p \in P$; we do not include these in our formal model as they do not affect the substitution structure) and a set, $V = \{v_1, \ldots, v_n\}$, of voters, where voter v_i corresponds to a partition P_{v_i} of P. A *partition* P_v is a disjoint set of *parts* whose union is P; i.e., $P_v = \{p^1, \ldots, p^z\}$, where for every $i, j \in [z]$, $p^i \cap p^j = \emptyset$ and $\cup_{j \in [z]} p^j = P$. Each p^j is referred to as a *part* of the partition P_v. A *partition aggregation method* is a function taking n partitions

of P and returning a partition S of P, referred to as the *aggregated partition*. (We ignore issues of tie-breaking as they clutter the presentation without adding significant insights.) We denote an instance of partition aggregation as (P, C), where P is the set of projects and C is the collection of n partitions of P.

2.2 Utilitiarianism

The main question we are studying here is how to define a "good" partition. Here we take a utilitiarian approach: We assume that each voter v, based on her partition P_v, would derive a certain utility from each possible aggregated partition S. While these utilities are unknown, they might be estimated (similarly to set extensions, which are used to estimate utilities over committees based on utilities over single candidates in multiwinner elections). Given a specific way of estimating such utilities over the set of possible partitions, a natural partition aggregation method would return, as the aggregated partition, a partition which maximizes the sum (or the minimum) – over the voters – of these utilities. Such a utilitarian approach has been applied successfully for many social choice settings, including multiwinner elections [9] and participatory budgeting [10].

Example 2.1. Consider the set of projects $P = \{p_1, p_2, p_3, p_4\}$ and a voter v with the partition $P_v = \{\{p_1, p_2\}, \{p_3, p_4\}\}$. It is natural to assume that the utility of voter v from the partition $S_1 = \{\{p_1, p_2\}, \{p_3\}, \{p_4\}\}$ would be fairly high, as P_v and S_1 are quite similar. Furthermore, the utility of v from $S_2 = \{\{p_1\}, \{p_2\}, \{p_3\}, \{p_4\}\}$ might be less than her utility from S_1, as S_2 seems to be less similar to P_v than S_1 is.

Our utilitarian approach is formally defined below.

Definition 2.2 (Utility function). Let P be a set of m projects and let $\mathcal{P}$ be the set of all partitions of P. A *utility function* is a function $f : \mathcal{P} \times \mathcal{P} \to \mathbb{N}$. For a voter v and a possible aggregated partition S, $f(v, S)$ is understood as the utility that voter v gets from S; the higher the better.

We will define different utility functions for when we study substitution structures and complementarity structures. In particular, in Section 3 we will define utility functions relevant for substitution structures, while in Section 4 we will define utility functions relevant for complementarity structures.

2.3 Aggregation Goals

Given a utility function f, we consider two partition aggregation goals: Maximizing the sum of utilities (Total) and maximizing the minimum utility (Egal).

*Definition 2.3 (***Total Utility (Total)***).* Given a set of m projects, a collection of n voters, V, along with their partitions, and a utility function f, the goal in *Total Utility* aggregation is to find a partition S such that $\sum_{v \in V} f(v, S)$ is maximized.

That is, in Total Utility aggregation we look for a partition that maximizes the total utility (i.e., $\sum_{v \in V} f(v, S)$). In the decision version of the problem, given an integer k additionally, we look for a partition that has total utility at least k. We use the same problem name for optimisation as well as decision version of the problem as it will be clear from the context.

We also consider egalitarian aggregation methods, in which we care for the least satisfied voter (similarly in spirit to egalitarian committee scoring rules [1]).

*Definition 2.4 (***Egalitarian Utility (Egal)***).* Given a set of m projects, a collection of n partitions, $P_1, \ldots, P_n$, and a utility function f, in *Egalitarian Utility* aggregation the goal is to find a partition S such that $\min_{v \in V} f(v, S)$ is maximized.

In the decision version of the problem, given an integer k additionally, we look for a partition that has Egalitarian Utility at least k.

Given a specific utility function f, by Total-f (Egal-f) we refer to the aggregation method aiming at maximizing the sum (minimum) of voter utility, where voter utility is defined via f.

3 SUBSTITUTION STRUCTURES

Below we study aggregating partitions for substitution structures. We define two utility functions that are relevant for substitution structures; we then study the computational complexity of the aggregation methods corresponding to these utility functions. Lastly, we define several axiomatic properties relevant for substitution structures and study them wrt. the aggregation methods.

3.1 Two Utility Functions

We consider two utility functions. The PA utility function is perhaps the first utility function one might think of: The utility of a voter equals the number of pairs for which her vote agrees with the aggregated vote. The PAM utility function is slightly less natural as the utility is the number of pairs for which the voter agrees with the aggregated vote, minus the number of pairs not in the same partition in the voter's partition but in the same partition in the aggregated vote. Thus, in essence, we define a larger fine for merging a pair of projects unnecessarily (a fine of -1), than for splitting a pair of projects unnecessarily (a fine of 0). The reason for this asymmetry is our motivation from substitution structures: We believe that an aggregated partition that defines a set of projects as substitutions to each other, while most voters prefer not to have them defined as such is more harmful than an aggregated partition that simply fails to define a substitution in cases in which it should.

Number of pairwise agreements (PA): Let p_1, p_2 be two projects in P. For a voter v, and an aggregated partition S, let

$$\delta^{\mathsf{PA}}_{v,S}(p_1, p_2) = \begin{cases} 1 & \text{if } p_1, p_2 \text{ are in same (different) part(s)} \\ & \text{in both } P_v \text{ and } S\ ; \\ 0 & \text{otherwise}\ . \end{cases}$$

We define the *Number of pairwise agreements (PA)* utility function for voter v from S to be as follows:

$$f^{\mathsf{PA}}(v, S) = \sum_{p_1, p_2 \in P} \delta^{\mathsf{PA}}_{v,S}(p_1, p_2)\ .$$

Number of pairwise agreements minus number of mergings (PAM): Let p_1, p_2 be two projects in P. For a voter v, and aggregated

partition S, let

$$\delta^{\mathsf{PAM}}_{v,S}(p_1,p_2) = \begin{cases} 1 & \text{if } p_1, p_2 \text{ are in same(different) part(s)} \\ & \text{in both } P_v \text{ and } S\ ; \\ 0 & \text{if } p_1 \text{ and } p_2 \text{ are in same part in } P_v \\ & \text{but in different parts in } S\ ; \\ -1 & \text{if } p_1 \text{ and } p_2 \text{ are in different parts in } P_v \\ & \text{but in same part in } S\ . \end{cases}$$

We define the *Number of pairwise agreements minus number of mergings (PAM)* utility function for voter v from S to be:

$$f^{\mathsf{PAM}}(v,S) = \sum_{p_1,p_2 \in P} \delta^{\mathsf{PAM}}_{v,S}(p_1,p_2)\ .$$

Example 3.1. Our two utility functions and two aggregation types define four aggregation methods: Total-PA, Egal-PA, Total-PAM, and Egal-PAM. Consider the following set of partitions to illustrate the difference between these methods. Let $P_{v_1} = \{\{p_1,p_2,p_3\}\}$, $P_{v_2} = \{\{p_1\},\{p_2,p_3\}\}$, and $P_{v_3} = \{\{p_1\},\{p_2\},\{p_3\}\}$.

Let $S = \{\{p_1,p_2\},\{p_3\}\}$ be an aggregated partition. We first describe Total-PA. For the pair of projects p_1,p_2, $\delta^{\mathsf{PA}}_{v_1,S}(p_1,p_2) = 1$ as p_1 and p_2 are in same part in both the partitions; however $\delta^{\mathsf{PA}}_{v_1,S}(p_1,p_3) = 0$ as p_1 and p_3 are in same part for v_1 and in different parts in S. Similarly, $\delta^{\mathsf{PA}}_{v_1,S}(p_2,p_3) = 0$. Hence, the PA utility for voter v_1 from S, $f^{\mathsf{PA}}(v_1,S)$, is 1. Similarly, the PA utility for voter v_2 and v_3 from S are 1 and 2, respectively. The Total-PA utility from S is 4. The Egal-PA utility is 1. Next, we describe Total-PAM utility. For the pair of projects a,b, $\delta^{\mathsf{PAM}}_{S,v_1}(p_1,p_2) = 1$; however $\delta^{\mathsf{PAM}}_{S,v_1}(p_1,p_3) = 0$ and $\delta^{\mathsf{PAM}}_{S,v_1}(p_2,p_3) = 0$. Hence, the PAM utility for v_1 from S, $f^{\mathsf{PAM}}(v_1,S)$, is 1. Furthermore, $\delta^{\mathsf{PAM}}_{S,v_2}(p_1,p_2) = -1$ as p_1 and p_2 are in different parts for v_2, while they are in the same part in S; $\delta^{\mathsf{PAM}}_{S,v_2}(p_1,p_3) = 1$ and $\delta^{\mathsf{PAM}}_{S,v_2}(p_2,p_3) = 0$. Hence, the PAM utility for v_2 from S is 0. Similarly, the PAM utility for v_3 from S is 1. Therefore, the Total-PAM utility from S is 2, and the Egal-PAM utility is 0.

3.2 Computational Complexity

Recall that the problem of finding an aggregated partition with maximum Total-PA is equivalent to the Consensus Clustering problem. Since Consensus Clustering is known to be NP-hard even for three partitions [7], we have the following result.

Proposition 3.2. *Total-PA is NP-hard even for three voters.*

Consensus Clustering is also known to be NP-hard when every input partition has at most two parts [7]. Hence, we have following result.

Proposition 3.3. *Total-PA is NP-hard even when every voter has at most two parts.*

We next present our intractability result for Total-PAM.

Theorem 3.4. *Total-PAM is NP-hard even when each partition contains at most two parts.*

Proof. We describe a polynomial-time reduction from the known NP-hard problem Cluster Deletion (Given a graph G, and an integer k; we shall decide the existence of at most k-sized set of edges whose deletion from G results into a cluster graph (i.e., a disjoint union of cliques) [15]. Let (G,k) be an instance of the Cluster Deletion problem. Let $|V(G)| = n, |E(G)| = m$. Without loss of generality, assume that $n-2 = 2^\ell$, for some positive integer ℓ. We first construct the set of projects P. For each vertex $u \in V(G)$, we add a project u in the set P. Now, we construct a collection of partitions, C, of projects. For every pair of vertices $u,v \in V(G)$, we create a collection of partitions C_{uv} as follows. If $uv \in E(G)$, then $|C_{uv}| = 3 \cdot 2^{2\ell-1}$, otherwise $|C_{uv}| = 3 \cdot 2^{4\ell-1}$. If $uv \in E(G)$, then in every partition in C_{uv}, u and v are in same part; otherwise u and v are in different parts in every partition in C_{uv}. For every pair of vertices $x,y \in V(G) \setminus \{u,v\}$, there are $|C_{uv}|/3$ partitions in C_{uv} in which x and y are in different parts, and $2\cdot|C_{uv}|/3$ partitions in which x and y are in same part. This collection of partitions C can be construed in polynomial time, however we skip the justification due to space constraint. The intuitive idea for such a collection of partitions is that for a pair of project x,y, the total PAM utility due to partitions in C_{uv}, where u,v are distinct from x,y, is $2^{2\ell-1}$ for any aggregated partition, as if x,y are in different parts in the aggregated partition, then the total PAM utility due to these partitions is $2\cdot|C_{uv}|/3 - |C_{uv}|/3 = |C_{uv}|/3$, otherwise $|C_{uv}|/3$. In essence, the utility of pair of project for $C_{u,v}$ does not depend on the parts to which x,y belongs in the aggregated partition. However, for project u,v, total PAM utility for $C_{u,v}$ depends on their parts in the aggregated partition. The set of partitions in our instance is $C = \cup_{u,v\in V(G)} C_{uv}$. We set total PAM utility as

$$\begin{aligned} k' =& 2^{4\ell-1}\Big(\frac{n(n-1)}{2} - m\Big)\Big(\frac{n(n-1)}{2} + 2\Big) \\ &+ 2^{2\ell-1}\Big(m\Big(\frac{n(n-1)}{2} - m + n + 2\Big) - 3k\Big)\ . \end{aligned}$$

□

We proceed to the egalitarian rules.

Theorem 3.5. *Egalitarian-{PA, PAM} are NP-hard.*

Proof. Due to space constraints, we describe a sketch of the reduction. We reduce from Unary Bin Packing, seen as a partition problem (one seeks a partition of $[kB]$ into k parts of size B where items, represented as disjoint subsets, must be included in single parts). We use $[kB]$ as the set of projects, and first build two voters: $R := \{[kB]\}$ and $S := \{\{1\},\ldots,\{kB\}\}$. We can enforce that the solution must have utility at least $t_R := k\binom{B}{2}$ with R and $t_S := \binom{kB}{2} - t_R$ with S ($t_S := \binom{kB}{2} - 2t_R$ in the PAM model). These two thresholds yield a partition with k blocks of size B. Then, for each pair of elements $e = (x,y)$ from the same item, we enforce that both elements are in the same part using a partition $Q_e = \{\{x\},\{y\},[kB]\setminus\{x,y\}\}$ with utility threshold $t_Q := t_R + 2(m-2B) + 3$ (or $t_Q := t_R + 4(m-2B) + 6$ in the PAM model). Together, these constraints ensure that the target solution is a valid bin packing of the original instance (we achieve distinct utility thresholds with additional gadgets appended to each partition). □

3.3 Axiomatic Properties

Here we consider various axiomatic properties that are relevant for substitution aggregation methods, and test our aggregation methods against them. While it is possible to define many axiomatic

Method	Complexity	Unanimity	Majority-based	IIP
Total-PA	NP-h even for $n = 3$ (Prop. 3.2) or $\ell \leq 2$ (Prop. 3.3)	Yes	Yes	No
Egal-PA	NP-h (Thm. 3.5)	No	No	No
Total-PAM	NP-h even for $\ell \leq 2$ (Thm. 3.4)	Yes	No	**Open**
Egal-PAM	NP-h (Thm. 3.5)	**Open**	No	**Open**
Total-COMP	NP-h (Thm. 4.2)	Yes	No	N/A
Egal-COMP	NP-h (Thm. 4.3)	Yes	No	N/A

Table 1: Summary of our results. We denote the maximum number of parts in any partition by ℓ.

properties, we chose axioms that seem especially relevant, with the application of aggregating substitution structures in mind.

The axiom of *Unanimity*, defined next, says that if all voters agree on whether some two projects shall be in the same part or in different parts (i.e., all voters are unanimous wrt. to this pair of projects), then the aggregated partition shall also agree with the voters regarding these two projects.

*Definition 3.6 (***Unanimity)**. A partition aggregation method $\mathcal{R}$ satisfies *Unanimity* if the following hold: Let $P_1, \ldots, P_n$ be the set of partitions of projects $P = \{p_1, \ldots, p_m\}$. If two projects p_i and p_j, $i, j \in [m]$, belong to the same part in P_i, for all $i \in [n]$, then p_i, p_j belong to the same part in the aggregated partition. Similarly, if p_i, p_j belong to different parts in P_i, for all $i \in [n]$, then p_i, p_j belong to different parts in the aggregated partition.

While Unanimity requires that the aggregated partition agrees with the voters on those pairs of projects for which all voters are in complete agreement, majority-based aggregation considers pairs of projects with majority agreement among the voters. As we show, such aggregated partitions need not exist; thus, we require an aggregation method to output such partitions only when they exist.

*Definition 3.7 (***Majority based aggregation)**. An aggregated partition $\tilde{P}$ is majority-based if any two projects p_1, p_2 are in the same part in $\tilde{P}$ if and only if p_1 and p_2 are placed in the same part for more than half of the voters. A partition aggregation method $\mathcal{R}$ satisfies *Majority-based aggregation* if it always outputs majority-based aggregated partitions, whenever such exists.

Remark 3. *One might consider a continuum of axioms between Majority-based aggregation and Unanimity, by employing supermajorities: An aggregated partition p would place a pair of projects in the same partition iff at least a δ-Supermajority among the voters does so.*

We also consider an adaptation of the fundamental axiom of Independent of Irrelevant Alternatives to our setting of aggregating partitions: We refer to our adaptation as *Independent of Irrelevant Projects*. In essence, it means that if the restriction of some two profiles to a pair of projects is the same, then the restriction to this pair of projects of both aggregated partition shall be the same.

*Definition 3.8 (***Independent of Irrelevant Projects)**. A partition aggregation method $\mathcal{R}$ satisfies *Independent of Irrelevant Projects* if the following holds: Let P and P' be two partition profiles and let S and S' be their aggregated partitions according to $\mathcal{R}$. Let p_1 and p_2 be two projects such that the number of voters that places them in the same part in P and in P' are the same. Then, S and S' shall either both place p_1 and p_2 in the same part or in different parts.

3.4 Rules vs Axioms

Next, we compare our four aggregation methods wrt. to the axiom of Unanimity: We wish to identify which of our methods are Unanimous and which are not. First, we note that a unanimous partition always exists, and can be found in polynomial time:

Observation 1. *A unanimous partition always exists, and can be found in polynomial time.*

The proof follows from the fact that we can output any partition that is provided by some voter. Note that it is unanimous.

Next we show that Total-PA is also Unanimous. This is intuitively appealing, as, to maximize the total utility, it seems natural for the aggregated partition to agree with the voters at least on those pairs of projects for which the voters are in total agreement among themselves.

Theorem 3.9. *Total-PA is unanimous.*

In contrast, Egal-PA is not Unanimous. This is also intuitively appealing, as egalitarian methods care for the least satisfied voter.

Proposition 3.10. *Egalitarian-PA is not unanimous.*

Proof. We show this by an example. Suppose that we have 2 voters and 5 projects, p_1, p_2, p_3, p_4, p_5. Let the partition for first voter be $\{\{p_1, p_2, p_3, p_4, p_5\}\}$, and for second voter it is $\{\{p_1, p_2\}, \{p_3, p_4, p_5\}\}$. Let us consider some possible aggregated partition. In particular, we consider all partitions in which p_1 and p_2 are in the same part, and one partition in which p_1 and p_2 are in different parts to show that any partition containing p_1 and p_2 in same part can not be optimal solution of Egal-PA. Note that partitions $\{\{p_1, p_2, p_3, p_4\}, \{p_5\}\}$, $\{\{p_1, p_2, p_3, p_4\}, \{p_4\}\}$, and $\{\{p_1, p_2, p_4, p_5\}, \{p_3\}\}$ have same PA utility for both the voters; therefore in Table 2, we only mention one of these. Similarly, we omit some partitions in the table that contains p_1, p_2 in same part but have same utilities as some partition in the table. □

Next we consider Total-PAM. While we show that, similarly to Total-PA, Total-PAM also satisfies Unanimity, we conjecture that Egal-PAM, similarly to Egal-PA, does not satisfy Unanimity.

Theorem 3.11. *Total-PAM is unanimous.*

Next we consider Majority-based partitions. First, we show that, contrary to Unanimous partitions, Majority-based partitions do not always exist.

Aggregated Partition	PA utility	
	1st voter	2nd voter
$S = \{\{p_1, p_2, p_3, p_4, p_5\}\}$	10	4
$S = \{\{p_1, p_2, p_3, p_4\}, \{p_5\}\}$	6	4
$S = \{\{p_1, p_2, p_3\}, \{p_4\}, \{p_5\}\}$	3	5
$S = \{\{p_1, p_2, p_3\}, \{p_4, p_5\}\}$	4	6
$S = \{\{p_1, p_2\}, \{p_3\}, \{p_4, p_5\}\}$	2	7
$S = \{\{p_1, p_2\}, \{p_3\}, \{p_4\}, \{p_5\}\}$	1	7
$S = \{\{p_1, p_2\}, \{p_3, p_4, p_5\}\}$	4	10
$S = \{\{p_1, p_3, p_4, p_5\}, \{p_2\}\}$	6	6

Table 2: Example for non-unanimity of Egalitarian PA (Proposition 3.10).

Proposition 3.12. *A majority based aggregated partition need not exist.*

Proof. Consider three voters, v_1, v_2, v_3, and three projects p_1, p_2, p_3. Let the partitions corresponding to v_1, v_2, and v_3 be $P_{v_1} = \{\{p_1, p_2\}, \{p_3\}\}$, $P_{v_2} = \{\{p_1, p_2, p_3\}\}$, and $P_{v_3} = \{\{p_1, p_3\}, \{p_2\}\}$. According to the property of Majority-based aggregation, we shall have the following: (1) p_1, p_3 shall be in the same part; (2) p_1, p_2 shall be in the same part; and (3) p_2, p_3 shall be in different parts. As the three constraints above cannot be simultaneously satisfied, we conclude that there is no Majority-based partition for this profile. □

In a sense, the counterexample in the above proof is similar to the canonical counterexample showing the existence of Condorcet-cycles in single-winner elections (i.e., voters $a > b > c$, $b > c > a$, and $c > a > b$, requiring the contradicting requirements of $a > b$, $b > c$, and $c > a$ as the canonical example of Condorcet paradox.

Nevertheless, Majority-based aggregation exists for some profiles; for these profiles, it can be found in polynomial time.

Theorem 3.13. *A majority based aggregated partition can be found in polynomial time, if it exists.*

Proof. Let P be the set of projects. Let V be the set of voters. We construct an aggregated partition S iteratively as follows. Initially, let $S = \emptyset$. In each iteration i, we add a part S_i in S as follows: consider a project $x \in P \setminus (\cup_{j=1}^{i-1} S_j)$ (that is a project that has not been added in any part in previous iterations); add it in the part S_i, and add all the projects $x' \in P \setminus (\cup_{j=1}^{i-1} S_j \cup \{x\})$, such that x and x' are in same part for more than half of the voters, in S_i. Now, we check whether every two projects in S_i are in same part for more than half of the voters. If not, then the algorithm returns No. We also check whether there exists projects $y \in S_i$ and $y' \in S_j$, where $i \neq j$, such that y and y' are in different parts for at least half of the voters. If not, then the algorithm returns No. We repeat until there is a project which is not placed in any part in S; and return set S. Clearly, the algorithm runs in polynomial time.

Next we prove the correctness of the algorithm. Indeed, if the algorithm returns a partition S, then it is a majority based aggregated partition, as at every step the algorithm checks this property. Now, we show that if the algorithm returns No, then there is no majority based aggregated partition for V. Towards the contradiction, let S' be a majority based aggregation for V. Note that the algorithm returns No in two cases: (1) there are two projects, say y, z, in a part, say S_i, in S that are not in same part for more than half of the voters, or (2) there are two projects, say y, z, in different parts, say S_i, S_j, respectively, of S that are in same part for more than half of the voters. Consider the case (1). Clearly, y and z are in different parts in S'. Let the algorithm first added project x to S_i. Thus, x, y and x, z are in same part for more than half of the voters, as the algorithm added y, z in S_i, a contradiction that S' is majority based aggregation. Next, consider case (2). Clearly, y, z are in same part in S'. Let the algorithm first added project x to S_i. Then, algorithm added y to S_i but not z. This implies that x, y are in same part for more thant half of the voters but not x, z. Since the part containing y, z in S' does not contain x as x, z are in different parts for at least half of the voters, a contradiction that S' is a majority based aggregation, as x, y are in different parts in S'. □

Corollary 3.14. *The existence of a Majority-based partition for a given profile can be decided in polynomial time.*

Moreover, note that, if it exists, then a Majority-based aggregated partition is unique. Next, we check whether our partition aggregation methods output the Majority-based partition whenever it exists.

Theorem 3.15. *If a majority based aggregated partition exists, then it maximizes Total-PA.*

Proof. Let V be a set of partitions corresponding to all the voters. Let S be the majority based aggregated partition for V. Let $S' \neq S$ be an optimal solution of Total-PA for V. Since majority based aggregated partition is unique, S' is not a majority based aggregated partition. Thus, either (1) there exists a part S_i in S' containing two projects that are not in same part for more than half of the voters, or (2) there exists two parts S_i, S_j in S' such that there is a project in S_i and a project in S_j that are in same part for more than half of the voters.

Consider case (1). Let x and y be two projects in S_i that are in different parts for at least half of the voters. Let M_x be the set of projects that are in same part with x for more than half of the voters, and N_x be the set of projects that are not in same part with x for more than half of the voters. Note that a pair of projects a, b, where $a \in M_x$ and $b \in N_x$ are not in same part for more than half of the voters, otherwise majority based aggregated partition does not exist. Also, note that every two projects in M_x are in same part for more than half of the voters, otherwise majority based aggregated partition does not exist. Now, create a new partition by deleting N_x from S_i, and add a new set of projects N_x to S. Since every pair of project a, b, where $a \in \{x\} \cup M_x$ and $b \in N_x$, are in different parts for at least half of the voter; either the total PA utility of new partition is same as that of S' or larger than that of S. It can not be larger as S' is an optimal solution of Total-PA. We apply this step for all the parts in S_i that contains at least two projects that are not in same part for more than half of the voters. Note that the total PA utility for new partition is same as that of S'. Let us denote this new partition by S''.

Next, we consider case (2). Note that if there exists two parts S_i, S_j in S' such that there is a project in S_i and a project in S_j that are in same part for more than half of the voters, then such parts also exist in S'' (because we have not merged any two parts). Let

S_p and S_q be two parts in S'' that contains x and y, respectively, such that x and y are in same part for more than half of the voters. We merge parts S_p and S_q in the partition S''. Since S'' does not contain any part that violates majority based aggregation property due to case (1), all the pair of projects in S_p(or S_q) are in same part for more than half of the voters. Note that every pair of projects a, b such that $a \in S_p$ and $b \in S_q$ are in same part for more than half of the voters, otherwise majority based aggregated partition does not exist. Therefore, merging S_p and S_q increases the total PA utility, a contradiction to the optimality of S. Hence, S' does not contain parts that satisfies condition in case (2). □

Using the same example as in Proposition 3.10, we have following result.

Proposition 3.16. *Egalitarian-PA does not satisfy Majority-based aggregation.*

We go on to consider Total-PAM and Egal-PAM.

Proposition 3.17. *Total-PAM does not satisfy Majority-based aggregation.*

Proof. We offer a counterexample: Consider a set of 2 projects, $\{p_1, p_2\}$; and a set of 5 voters. Let the partition for 3 voters be $\{\{p_1, p_2\}\}$, and for 2 voters, the partition is $\{\{p_1\}, \{p_2\}\}$. Note that $\{\{p_1, p_2\}\}$ is a majority based aggregated partition; its total PAM utility is 1. However, the total PAM utility for $\{\{p_1\}, \{p_2\}\}$ is 2. □

The example in Lemma 3.17 implies the following as well.

Proposition 3.18. *Egalitarian-PAM does not satisfy the property of majority based aggregation.*

Next we consider our adaptation of the axiom of Independence of Irrelevant Alternatives to our setting.

Proposition 3.19. *Total-PA does not satisfy Independence of Irrelevant Projects.*

Observe that, if an aggregation rule is not unanimous, then it can not satisfy independent of irrelevant projects properties as we can have one profile in which all the projects are in the same part (or, equivalently, in different parts) for all voters, and another profile which is an instance that violates the Unanimity axiom. Hence, due to Proposition 3.10, we have following result.

Corollary 3.20. *Egalitarian-PA does not satisfy Independence of Irrelevant Projects.*

4 COMPLEMENTARITY STRUCTURES

Below we consider complementarity structures.

4.1 One Utility Function

Here we have one utility function. Intuitively, we take an extreme point of view regarding complementarity effects, assuming that projects in a complementarity part are "useless" unless the whole complementarity part that they are contained in is funded as a whole. Thus, we define the utility of a voter to be number of parts in her suggested partition that are – exactly as she suggested – in the aggregation partition.

Formally, we have the following:

Number of Correct Parts (COMP): Let v be a voter with the partition P_v and S be an aggregated partition. Then, $f^{COMP}(v, S) = |\{q \in P_v : q \in S\}|$. That is, the utility of voter v is the number of parts in her partition that are present in the aggregated partition.

Example 4.1. Consider a voter v with the partition $P_v = \{\{a, b\}, \{c, d, e\}, \{f\}\}$ and an aggregated partition $S = \{a\}, \{b, f\}, \{c, d, e\}\}$. Then, the utility of v wrt. S is 1, as only the part $\{c, d, e\}$ is present both in P_v and in S.

The corresponding aggregation methods are thus referred to as Total-COMP and Egal-COMP: In Total-COMP we wish to maximize the sum of voter utilities according to the utility function defined above; while in Egal-COMP we wish to maximize the minimum voter utility according to the utility function defined above.

4.2 Computational Complexity

Both Total-COMP and Egal-COMP are NP-hard.

Theorem 4.2. *Total-COMP is NP-hard.*

Proof. We reduce from the Set Packing problem, in which given sets $\mathcal{S} = \{S_1, \ldots, S_m\}$ over universe $X = \{x_1, \ldots, x_n\}$, we shall decide the existence of at least k disjoint sets of the set $\mathcal{S}$. Without loss of generality, we assume that $k > 2$, all the sets in $\mathcal{S}$ are distinct. We create a project x corresponding to each element in $x \in X$, and another dummy project d_p. For each set $S \in \mathcal{S}$, we add a voter v_S with its partition: one part is S, and the other part consists of all the remaining projects and dummy project. We also add a dummy voter v_d with its partition: one part is $\{d_p\}$, and the other part consists of all the remaining projects. We set the utility $u = k + 1$. Next, we prove equivalence between the instance $(X, \mathcal{S})$ of Set Packing and (P, C) of Total-COMP.

The idea of the proof is that the sets in the solution to (P, C) along with a set of all the elements which are not in the solution and the set $\{d_p\}$, will lead to a solution of $(X, \mathcal{S})$. In the backward direction, we observe that the utility of every voter is at most 1, otherwise we cannot achieve the desired utility. Further, we can pick at most one "big" part containing for any voter as every such part contains dummy project. So, there are at least k voters corresponding to whom we pick a part that corresponds to k disjoint set in $\mathcal{S}$. □

Theorem 4.3. *Egal-COMP is NP-hard.*

Proof. We provide a reduction from the Hitting String problem [12] in which we are given a set of n' strings $s_1, \ldots, s_{n'}$ over alphabet $\{0, 1, *\}$ of length m' each and shall decide whether there is a string q over alphabet $\{0, 1\}$ such that for each string s_i there is at least one $j \in [m']$ such that the jth character of s_i is equivalent to the jth character of q. (Note that q is over $\{0, 1\}$ while all $s_{n'}$'s are over $\{0, 1, *\}$.)

Given such an instance of Hitting String we construct an instance for Egal-COMP, as follows. For each $j \in [m']$ we construct projects $id_j, zero_j, one_j$ (so we have $3m'$ projects). Furthermore, we add one dummy project d_p. For each string s_i we have a voter s_i with the following partition: for each $j \in [m]$ for which the jth character of s_i is 0 we have a part $\{id_j, zero_j\}$; for each $j \in [m]$ for which the jth character of s_i is 1 we have a part $\{id_j, one_j\}$; and we have another part containing all other projects (we call this part of the partition

as the "*big*" part). Furthermore, we add one dummy voter v_d with the following partition: one part is $\{d\}$ and another part containing all other projects. We ask whether there is a solution achieving egalitarian utility at least 1. This finishes the construction.

For the forward direction, given a solution q to the Hitting String instance, we construct a solution for the Egal-COMP instance, as follows: for each $j \in [m]$ for which the jth character of q is 0 we have a part $\{id_j, zero_j\}$; for each $j \in [m]$ for which the jth character of q is 1 we have a part $\{id_j, one_j\}$; a part $\{d\}$, and one more part containing all other projects. Then, the utility of each voter is at least 1. For the backward direction, let $\mathcal{P}$ be a an aggregated partition with with egalitarian utility 1. We first note that for the dummy voter, the utility is due to the part $\{d\}$ as if $P \setminus \{d\}$ is in $\mathcal{P}$, the utility of other voters is 0. Since $\{d\}$ is in $\mathcal{P}$, for any voter, the "**big**" part is not in the aggregated partition as these part contains d. Thus, for every voter the utility is due to $\{id_j, zero_j\}$ or $\{id_j, one_j\}$ for some $j \in [m]$. We construct a string q as follows: if $\{id_j, zero_j\}$ is in $\mathcal{P}$, then jth character of q is 0, otherwise 1. Clearly, it is a hitting string for $s_1, \ldots, s_{n'}$. □

4.3 Axiomatic Properties

Here we consider axiomatic properties that are relevant for complementarity aggregation methods, similar to the one defined for substitution structure, and test our aggregation methods against them.

Unanimity requires that, if all voters agree on a part, then the aggregated partition shall contain that part.

*Definition 4.4 (***Unanimity)**. A partition aggregation method $\mathcal{R}$ satisfies *Unanimity* if the following holds: Let $P_1, \ldots, P_n$ be the set of partitions of projects $P = \{p_1, \ldots, p_m\}$. If there is a part $X \subseteq P$ that belongs to every P_i, where $i \in [n]$, then X also belongs to the aggregated partition.

Majority-based aggregation requires that, if a majority of the voters agree on a part, then this shall be in the aggregated partition.

*Definition 4.5 (***Majority based aggregation)**. A partition aggregation method $\mathcal{R}$ satisfies *Majority based aggregation* if the following holds: Let $P_1, \ldots, P_n$ be the set of partitions of projects $P = \{p_1, \ldots, p_m\}$. If there is a part $X \subseteq P$ that belongs to more than half of the partitions, then X also belongs to the aggregated partition.

4.4 Rules vs Axioms

Next, we compare our two complementarity aggregation methods wrt. to the axiom of Unanimity:

Theorem 4.6. *Total-COMP is unanimous.*

Proof. Let $\mathcal{P}$ be an optimal aggregated partition for a set of voters $\{v_1, \ldots, v_n\}$ that maximizes Total-COMP. Let $X \subseteq P$ be a part that is in the partitions of all the voters. We claim that X is in $\mathcal{P}$. Suppose not, then we create an aggregated partition $\mathcal{P}'$ as follows: initially, $\mathcal{P}' = \mathcal{P}$; add X to $\mathcal{P}'$, delete projects in X from other parts in $\mathcal{P}'$. We claim that the utility of $\mathcal{P}'$ is more than that of $\mathcal{P}$. Let $X_1, \ldots, X_\ell$ be parts in $\mathcal{P}$ that intersect with X. No voter gets utility from these parts as every voter has part X. All the parts in $\mathcal{P} \setminus \{X_1, \ldots, X_\ell\}$ are also present in $\mathcal{P}'$. $\mathcal{P}'$ also contains X. Thus, the utility for $\mathcal{P}'$ is at least one more than the utility for $\mathcal{P}$, a contradiction to the optimality of $\mathcal{P}$. □

Using similar arguments, we show that Egal-COMP also satisfies this axiom.

Theorem 4.7. *Egal-COMP is unanimous.*

Proof. Let $\mathcal{P}$ be an optimal aggregated partition for a set of voters, $\{v_1, \ldots, v_n\}$, that maximizes Total-COMP. Let $X \subseteq P$ be a part that is in the partitions of all the voters. We claim that X is in $\mathcal{P}$. Suppose not, then we create an aggregated partition $\mathcal{P}'$ as follows: initially, $\mathcal{P}' = \mathcal{P}$, add X to $\mathcal{P}'$, delete projects in X from other parts in $\mathcal{P}'$. We claim that the egailtarian utility of $\mathcal{P}'$ is more than that of $\mathcal{P}$. Let $X_1, \ldots, X_\ell$ be parts in $\mathcal{P}$ that intersects with X. Clearly, no voter gets utility from these parts as every voter has part X. All the parts in $\mathcal{P} \setminus \{X_1, \ldots, X_\ell\}$ are also present in $\mathcal{P}'$. $\mathcal{P}'$ also contains X. Thus, the satisfaction of every voter increases by at least 1 for $\mathcal{P}'$, a contradiction to the optimality of $\mathcal{P}$. □

Next we consider Majority-based aggregation. We show that, contrary to Unanimity, Total-COMP and Egal-COMP does not satisfy this axiom.

Theorem 4.8. *Total-COMP and Egal-COMP do not satisfy majority-based aggregation.*

Proof. We first argue for Total-COMP using the following example. Let $P_{v_1} = P_{v_2} = \{p_1, p_2, p_3\}$ and $P_{v_3} = \{\{p_1\}, \{p_2\}, \{p_3\}\}$. Note that the utility for the partition $\{\{p_1\}, \{p_2\}, \{p_3\}\}$ is 3, while it is 2 for the majority based aggregated partition $\{p_1, p_2, p_3\}$.

We argue for Egal-COMP using the following example. Let $P_{v_1} = P_{v_2} = \{p_1, p_2\}, \{p_3\}\}$ and $P_{v_3} = \{\{p_1\}, \{p_2, p_3\}\}$. Note that the egalitarian utility for the partition $\{\{p_1\}, \{p_2\}, \{p_3\}\}$ is 1, while it is 0 for the majority based aggregated partition $\{\{p_1, p_2\}, \{p_3\}\}$. □

5 OUTLOOK

We have studied several aggregation methods for aggregating partition, both for identifying good substitution structures and for identifying good complementarity structures. Beside studying further utility functions and aggregation goals and performing computer-based simulations on a carefully designed heuristic methods to solve the combinatorial problems we discuss here, the most pressing avenue for future research is to consider substitution structures and complementarity structures simultaneously: concretely, one might study how to aggregation "signed partitions", in which each voter partitions the set of projects and declare, for each part, whether it is a substitution part or a complementarity part.

ACKNOWLEDGEMENT

Nimrod Talmon was supported by the Israel Science Foundation (ISF; Grant No. 630/19).

REFERENCES

[1] Haris Aziz, Piotr Faliszewski, Bernard Grofman, Arkadii Slinko, and Nimrod Talmon. 2018. Egalitarian Committee Scoring Rules. In *Proceedings of IJCAI '18.* 56–62.

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

[3] Gerdus Benade, Swaprava Nath, Ariel D Procaccia, and Nisarg Shah. 2017. Preference elicitation for participatory budgeting. In *Thirty-First AAAI Conference on Artificial Intelligence*.
[4] Paola Bonizzoni, Gianluca Della Vedova, Riccardo Dondi, and Tao Jiang. 2008. On the approximation of correlation clustering and consensus clustering. *J. Comput. System Sci.* 74, 5 (2008), 671–696.
[5] Florian Brandl, Felix Brandt, Dominik Peters, Christian Stricker, and Warut Suksompong. 2019. Donor Coordination: Collective Distribution of Individual Contributions.
[6] Yves Cabannes. 2004. Participatory budgeting: a significant contribution to participatory democracy. *Environment and Urbanization* 16, 1 (2004), 27–46.
[7] Martin Dörnfelder, Jiong Guo, Christian Komusiewicz, and Mathias Weller. 2014. On the parameterized complexity of consensus clustering. *Theoretical Computer Science* 542 (2014), 71–82.
[8] Ulle Endriss and Umberto Grandi. 2017. Graph aggregation. *Artificial Intelligence* 245 (2017), 86–114.
[9] Piotr Faliszewski, Piotr Skowron, Arkadii Slinko, and Nimrod Talmon. 2016. Committee Scoring Rules: Axiomatic Classification and Hierarchy. 250–256.
[10] Piotr Faliszewski and Nimrod Talmon. 2019. A Framework for Approval-based Budgeting Methods. In *AAAI '19*.
[11] Rupert Freeman, David M Pennock, Dominik Peters, and Jennifer Wortman Vaughan. 2019. Truthful Aggregation of Budget Proposals.
[12] Michael R Garey and David S Johnson. 1979. *Computers and intractability*. Vol. 174. freeman San Francisco.
[13] Aristides Gionis, Heikki Mannila, and Panayiotis Tsaparas. 2007. Clustering aggregation. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1 (2007), 4.
[14] Pallavi Jain, Krzysztof Sornat, and Nimrod Talmon. 2020. Participatory Budgeting with Project Interactions. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, Christian Bessiere (Ed.). ijcai.org, 386–392.
[15] Ron Shamir, Roded Sharan, and Dekel Tsur. 2004. Cluster graph modification problems. *Discrete Applied Mathematics* 144, 1-2 (2004), 173–182.
[16] Alexander Strehl and Joydeep Ghosh. 2002. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research* 3 (2002), 583–617.