Streamlining Equal Shares

Sonja Kraiczy, Isaac Robinson, Edith Elkind

Abstract

Participatory budgeting (PB) is a form of citizen participation that allows citizens to decide how public funds are spent. Through an election, citizens express their preferences on various projects (spending proposals). A voting mechanism then determines which projects will be approved. The Method of Equal Shares (MES) is the state of the art algorithm for a proportional, voting based approach to participatory budgeting and has been implemented in cities across Poland and Switzerland. A significant drawback of MES is that it is not *exhaustive* meaning that it often leaves a portion of the budget unspent that could be used to fund additional projects. To address this, in practice the algorithm is combined with a completion heuristic - most often the "add-one" heuristic which artificially increases the budget until a heuristically chosen threshold. This heuristic is computationally inefficient and will become computationally impractical if PB is employed on a larger scale. We propose the more efficient ADD-OPT heuristic for Exact Equal Shares (EES), a variation of MES that is known to retain many of its desirable properties. We solve the problem of identifying the next budget for which the outcome for EES changes in $O(mn)$ time for approval utilities and $O(m^2n)$ time for uniform utilities, where m is the number of projects and n is the number of voters. Our solution to this problem inspires the efficient ADD-OPT heuristic which bypasses the need to search through each intermediary budget. We perform comprehensive experiments on real-word PB instances from Pabulib and show that completed EES outcomes usually match the proportion of budget spent by completed MES outcomes. Furthermore, the ADD-OPT heuristic matches the proportion of budget spend by add-one for EES.

1 Introduction

Participatory budgeting is a democratic process that enables citizens to decide how public funds should be spent. A natural tool for this task is voting: the governing body (e.g., a city) runs an election to gather citizens' preferences on various projects and uses a voting mechanism to decide how to allocate the funds. The projects offered for a public vote are specific proposals that come with predetermined scope and cost, made publicly known before the election, e.g., "Build a cycle lane on Main Street for $10,000.". So, rather than electing representatives who make budgetary decisions, citizens vote directly on specific proposals. Today, participatory budgeting is used worldwide in hundreds of cities [4, 18] as well as in decentralized autonomous organizations (DAOs) [19, 3], which are blockchain-based self-governed communities. For example, in community funding, members can contribute funds to the DAO and then vote on which projects the funds should be granted to.

A commonly used, naive approach to determine which projects should be funded is *Greedy Approval (GrA)*: voters are asked to report which projects they approve, and then projects are selected in descending order of approval count, skipping those that exceed the remaining budget. However, GrA can be unfair: if 51% of voters support enough projects to exhaust the entire budget, they effectively control 100% of the budget, leaving the remaining 49% of voters unrepresented, whereas ideally, if 10% of citizens

vote for, say, cycling-related projects, approximately 10% of the budget should be allocated to the cycling infrastructure. This issue is addressed by the Method of Equal Shares (MES), which is a state-of-the-art voting rule for participatory budgeting this provides representation guarantees to groups of voters with shared preferences [12]. In a behavioral experiment by Yang et al. [20] voters consistently found the outcomes of MES to be fairer than those of GrA. MES has been successfully implemented in Świecie and Wieliczka in Poland, in Aarau and Winterthur in Switzerland, as well as Assen in the Netherlands [13].

To overcome the limitations of GrA, MES adopts a market-based approach: The budget is split evenly[1] among all voters, and, for a project to be selected, it must be funded by voters who support it, with all voters paying the same amount (with a caveat that if a voter cannot afford to pay their share, they can contribute their entire remaining budget instead). This limits how much of the budget any group of voters can control. Like GrA, MES selects the projects sequentially; however, in contrast to GrA, the projects are ranked based on the utility per unit of money paid by each fully contributing voter. This twist turns out to capture proportionality as defined by the axiom of Extended Justified Representation (EJR) [1, 15], which ensures that each group of voters with shared preferences holds voting power proportional to its size.

A shortcoming of MES is that it often terminates when the leftover budget is large enough to pay for projects that remained unfunded [13]. Consequently, MES is said to be *non-exhaustive*. This underspending is undesirable: indeed, citizens are unhappy when leftover funds could have been used to finance projects they voted for, and many governments have a "use it or lose it" policy, where underspending results in subsequent budgets being cut. This often leads to low-value projects being funded when excess budget is available [11]. To mitigate the issue of underspending, the base MES algorithm is supplemented with a heuristic, known as a *completion method*, to complete the MES outcome.

The strategy currently used in practice is to run MES with a virtual budget that is larger than the actual budget; the size of the virtual budget is selected so that the method spends a larger fraction of the actual budget (but does not exceed it). This strategy is appealing because it does not increase the conceptual complexity of the method: all voters still have equal voting power and, as before, the most cost-efficient projects are selected to be funded, until voters run out of money. In contrast, combining MES with a different algorithm (such as, e.g., GrA) would be harder to explain to the stakeholders, and therefore less appealing in practice.

The challenge, then, is how to determine the "correct" virtual budget. The commonly used ADD-ONE heuristic iteratively increases each voter's budget by one until either (1) the solution becomes exhaustive or (2) the true budget is exceeded, and then returns the last feasible solution. However, this approach is computationally expensive: it often produces identical outcomes across most budget increments [8], thereby wasting computational resources. This is especially relevant in the context of research that simulates the method on random instances, as achieving statistical significance requires many repetitions. Importantly, MES has only been employed in smaller communities so far; the ADD-ONE heuristic may be infeasible for large cities or DAOs. Another difficulty with the ADD-ONE heuristic is the non-monotonicity of MES: MES may overspend at a virtual budget of b, but then produce a feasible solution at $b' > b$. This means that the ADD-ONE method could terminate early and miss the virtual

[1] MES can also operate if starting with unequal budget distribution, which may be appropriate in some settings, such as DAOs.

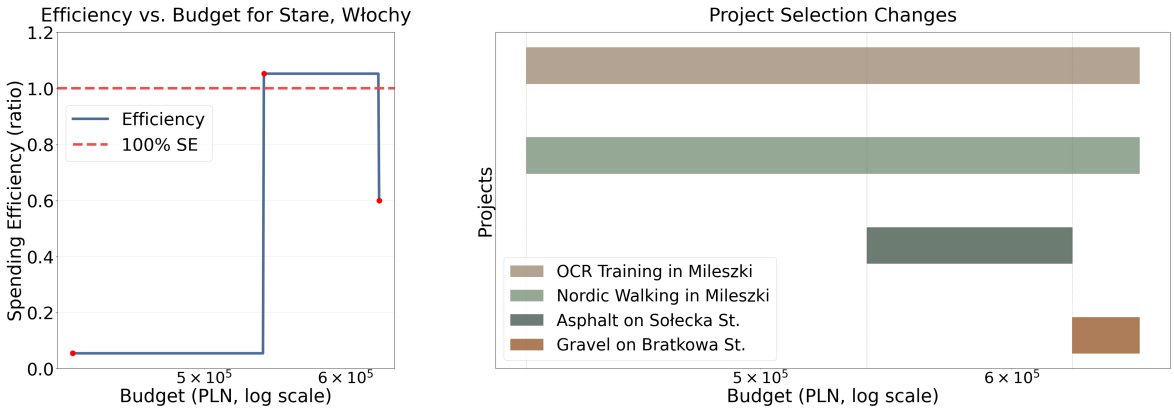


Figure 1: A real-world example where considering all budgets significantly increases spending efficiency, i.e., the fraction of the actual budget spent. The graph on the left shows the spending efficiency of the winning project set for a given virtual budget. The graph on the right shows which projects are selected for a given virtual budget. Here, the ADD-ONE heuristic stops as soon as the budget is exceeded, which happens when the Asphalt project is selected, resulting in less than 15% spending efficiency. A further increase in budget leads to the Asphalt project being dropped in favor of the Gravel project, as shown on the right, increasing the spending efficiency to 60%

budget that would spend the highest fraction of the budget. Figure 1 shows a real-life example of this phenomenon. Perhaps even more importantly, even if the costs of all projects are integer, the optimal virtual budget may be non-integer, so ADD-OPT may skip over the optimal virtual budget. Indeed, our analysis in Section 5 shows that for a non-trivial number of real-life instances there are outcomes that can only be achieved by a fractional budget.

Relatedly, the complexity of finding the optimal virtual budget under MES remains unknown. It is not even clear if the associated decision problem (given a value $b' \leq b$, determining if there is a virtual budget b^* such that MES with budget b^* spends at least b') is in NP: while b^* can be assumed to be rational: due to its sequential nature, MES may potentially produce numbers with super-polynomial bit complexity.

Contribution We consider a simplified variant of MES, which we call the *Exact Equal Shares (EES)* method. Under this rule, all voters who contribute to a project pay exactly the same amount (eliminating the caveat that the voters who are about to run out of money are allowed to contribute their entire budget); we will explain the differences between the two methods in Section 3. This rule was implicit in the work of Peters et al. [14] and Kraiczy and Elkind [8], whose results imply that it retains desirable proportionality properties of MES, but neither paper explicitly defines it.

The simplicity of EES enables us to propose a more principled and efficient approach to finding a good virtual budget. Our main theoretical contribution is a completion method ADD-OPT for EES, which finds the minimum per-voter budget increase that results in changing the set of selected projects or the set of voters paying for a project. The runtime of ADD-OPT is linear in the number of voters n. More specifically, for approval utilities (i.e., when we evaluate the projects assuming that each voter derives one unit of utility from each selected project they approve), its runtime is $O(mn)$, whereas for the more general model of uniform utilities defined in Section 3 (which subsumes, e.g., cost utilities [15]), its runtime is $O(m^2n)$.

By using ADD-OPT, we can iterate through all outcomes that can be accomplished by running EES with a virtual budget, and thereby ensure that we do not miss the optimal virtual budget; on the other hand, in contrast to ADD-ONE, ADD-OPT avoids redundant computation. Another advantage of ADD-OPT over ADD-ONE is that it is *currency agnostic*, i.e., the results remain consistent across currencies. In contrast, when using ADD-ONE, the practitioners need to decide what is an appropriate unit of currency: this choice is non-trivial as, e.g., one US dollar is approximately equal to 16,000 Indonesian rupiahs.

The ADD-OPT heuristic goes through all projects (including ones currently selected), and checks if, by increasing the virtual budget, it can increase the number of voters contributing to that project. While considering all projects is important for ensuring that the optimal virtual budget is not missed, we can achieve faster runtime by only considering projects that are not currently selected; we refer to the resulting heuristic as ADD-OPT-SKIP.

In order to evaluate the performance of EES with ADD-OPT and ADD-OPT-SKIP, we perform extensive experiments on real-life participatory budgeting instances. Our results indicate that, on average, EES with ADD-OPT-SKIP spends the same proportion of the budget as MES with ADD-ONE while enjoying far higher computational efficiency and eliminating counterintuitive phenomena such as the one illustrated in Figure 1.

Complete proofs of results marked by ♠ are deferred to the appendix.

Related Work Much of the progress in participatory budgeting builds on prior work in multiwinner voting, i.e., participatory budgeting with unit costs [10]. The Extended Justified Representation (EJR) axiom was first stated in this context by Aziz et al. [1].

Besides MES, Phragmén's method [16] and Proportional Approval Voting (PAV) [17] are well-established proportional rules in the multiwinner voting setting; Janson [7] provides an excellent overview. PAV satisfies EJR [1] and has optimal proportionality degree, but is NP-hard to compute. Its threshold-based local search variant is polynomial-time computable [2, 9], but may be hard to explain to voters. Phragmén's sequential method is also market-based. Unlike MES, it is exhaustive, but it does not satisfy EJR. Moreover, while Peters et al. [15] extended MES to participatory budgeting with general additive utilities, neither PAV nor Phragmén have been adapted to this general setting.

Exact Equal Shares for approval utilities was implicitly studied by Peters et al. [14] and Kraiczy and Elkind [8]. Peters et al. [14] introduce a stability notion for participatory budgeting that is satisfied by the outcome of Exact Equal Shares. Kraiczy and Elkind [8] propose an adaptive version of EES for approval utilities, which uses the outcome of EES for a smaller budget to compute the outcome of EES for a larger budget more efficiently, an alternative approach that complements our work. They also consider the problem of finding the minimum budget increment that changes the election outcome, and propose an $O(n^2m)$ algorithm for this problem in the context of approval utilities. However, in practice, both in cities and in DAOs, the number of voters (n) is usually large, while the number of projects (m) is relatively small. As EES itself has a linear dependency on n, a completion method with quadratic dependency on n is undesirable.

2 Preliminaries

For each $t \in \mathbb{N}$, we write $[t] = \{1, 2, \ldots, t\}$.

Participatory Budgeting (PB) We first introduce the model of participatory budgeting with approval ballots. An *election* is a tuple $E(b) = (N, P, \{A_i\}_{i \in N}, cost, b)$, where $b \in \mathbb{Q}_{\geq 0}$ is the available *budget*; $P = \{p_1, \ldots, p_m\}$ is the set of *projects* (for the purposes of tie-breaking, we fix a total order $\lhd$ on P, and write $p = \max Q$ for $Q \subseteq P$ whenever $p' \lhd p$ for all $p' \in Q \setminus \{p\}$); $N = [n]$ is the set of *voters*, and for each $i \in N$ the set $A_i \subseteq P$ is the *ballot* of voter i; and $cost : P \to \mathbb{Q}_{\geq 0}$ is a function that for each $p \in P$ indicates the cost of selecting p. For each $Q \subseteq P$, we denote the total cost of Q by $cost(Q) = \sum_{p \in Q} cost(p)$.

Given a project $p \in P$, we write $N_p = \{i \in N \mid p \in A_i\}$ for the set of voters who approve p. An *outcome* for an election $E(b)$ is a set of projects $W \subseteq P$ that is *feasible*, i.e., satisfies $cost(W) \leq b$. Our goal is to select an outcome based on voters' ballots. An *aggregation rule* (or, in short, a *rule*) is a function $\mathcal{R}$ that for each election E selects a feasible outcome $\mathcal{R}(E) = W$.

Utility Models We assume that voters' utilities are induced by a *uniform utility function* $u : P \to \mathbb{Q}_{\geq 0}$ so that the utility of a voter $i \in N$ for a project $p \in P$ is given by $u_i(p) = u(p) \cdot \mathbb{I}[p \in A_i]$, where $\mathbb{I}$ is the indicator function. Important special cases are $u(p) \equiv 1$ (*approval utilities*) and $u(p) = cost(p)$ for all $p \in P$ (*cost utilities*). For each $T \subseteq P$ we write $u(T) = \sum_{p \in T} u(p)$ and $u_i(T) = \sum_{p \in T} u_i(p)$.

Price System A *price system* for an outcome W in an election $E(b) = (N, P, \{A_i\}_{i \in N}, cost, b)$ is a collection of nonnegative rational numbers $X = (x_{i,p})_{i \in N, p \in P}$, where $x_{i,p}$ is voter i's payment for project p, satisfying the following three conditions: (1) every voter spends at most her share of the budget: $\sum_{p \in W} x_{i,p} \leq \frac{b}{n}$; (2) for each project $p \in W$, the sum of payments towards p equals its cost: $\sum_{i \in N} x_{i,p} = cost(p)$ for all $p \in W$; (3) Voters can only pay for projects they approve: for each $i \in N$ if $x_{i,p} > 0$ then $p \in W \cap A_i$. Note that our definition of a price system is less demanding than that of Peters and Skowron [12], who additionally require that for each project in $P \setminus W$ its supporters do not have enough money left to pay for it.

If W is an outcome for $E(b)$ and X is a price system for W, we call the pair (W, X) a *solution* for $E(b)$. Let $N_p(X) = \{i \in N \mid x_{i,p} > 0\}$ be the set of voters who pay for p in X. We define $O_p(X) = N_p \setminus N_p(X)$ to be the subset of voters who approve p, but do not pay for it in X. Also, let $r_i = \frac{b}{n} - \sum_{p \in P} x_{i,p}$ denote voter i's *leftover budget*. We say that X is *equal-shares* if for each $p \in W$ and all $i, j \in N_p(X)$ we have $x_{i,p} = x_{j,p}$: that is, the voters who pay for p share the cost of p exactly equally. In this case, we also say that the solution (W, X) is *equal-shares*.

3 Exact Equal Shares Method

Peters et al. [14] and Kraiczy and Elkind [8] study a variant of MES, which we will call Exact Equal Shares (EES).[2] While they only define EES for approval utilities, we will now extend their definition to uniform utilities.

[2] In both of these papers, this rule and the analysis of its properties are a byproduct of the framework developed for other purposes, and neither paper coins a name for it.

Description Given an election $E(b)$, EES starts by allocating each voter a budget of $\frac{b}{n}$ and setting $W = \varnothing$. It then iteratively identifies the set of all projects $p \in P \setminus W$ such that a subset of voters $V \subseteq N_p$ have enough leftover budget to split the cost of p equally (i.e., by paying $cost(p)/|V|$ each), selects a project in this set with the maximum *bang per buck* $\frac{u(p)}{cost(p)/|V|}$, adds it to W and updates the budgets. We assume that ties are broken according to the order $\lhd$ on P (see Algorithm 1 for the pseudocode). We write $EES(E(b), u)$ for the solution (W, X) returned by EES when run on the election $E(b)$ with uniform utility function u. For approval utilities we will simply write $EES(E(b))$, and for cost utilities we will write $EES(E(b), cost)$.

The standard Method of Equal Shares (which we refer to as MES) operates similarly, with one exception: if a voter i approves a project, but cannot afford to pay as much as the other contributors, she is allowed to help by contributing her entire leftover budget r_i. The order in which projects are selected is nevertheless computed based on the contributions of fully paying voters. A detailed description of MES is provided by Peters et al. [15].

Algorithm 1: EES for uniform utilities

Input: $E(b) = (N, P, \{A_i\}_{i \in N}, cost, b)$, $u : P \to \mathbb{Q}_{\geq 0}$
Output: Solution (W, X)

$W = \varnothing$, $X = \mathbf{0}^{n \cdot m}$, $r_i = \frac{b}{n}$ for all $i \in N$;
while *true* **do**
  $\Phi = \left\{(p \in P \setminus W, V \subseteq N_p) \mid r_i \geq \frac{cost(p)}{|V|} \quad \forall i \in V\right\}$;
  if $\Phi = \varnothing$ **then**
    return (W, X)
  end
  else
    Let $\Phi^* = \arg\max_{(p,V) \in \Phi} \frac{|V| \cdot u(p)}{cost(p)}$;
    Choose (p^*, V^*) from Φ^* so that $p^* \lhd p$ for all $(p, V) \in \Phi^* \setminus \{(p^*, V^*)\}$;
    $W = W \cup \{p^*\}$;
    $x_{i,p^*} = \frac{cost(p^*)}{|V^*|}$, $r_i = r_i - \frac{cost(p^*)}{|V^*|} \quad \forall i \in V^*$;
  end
end

Proportionality Guarantees A key feature of MES is that for approval utilities it satisfies a strong proportionality axiom known as Extended Justified Representation (EJR). We will now formulate this axiom and its relaxation EJR1, and argue that EES is just as attractive as MES from this perspective.

Definition 3.1 (Extended Justified Representation)**.** Given an election $E(b) = (N, P, \{A_i\}_{i \in N}, cost, b)$ and a subset of projects $T \subseteq P$, we say that a group of voters V is *T-cohesive* if $T \subseteq \cap_{i \in V} A_i$ and $\frac{|V|}{n} \cdot b \geq cost(T)$.

An outcome W for $E(b)$ is said to provide *Extended Justified Representation (EJR)* (respectively, *Extended Justified Representation up to one project (EJR1)*) for uniform utilities u if for each $T \subseteq P$ and each T-cohesive group V of voters there exists a voter $i \in V$ such that $u_i(W) \geq u(T)$ (respectively, $u_i(W) \geq u(T)$ or $u_i(W \cup \{p\}) > u(T)$ for some $p \in P$).

A rule $\mathcal{R}$ satisfies EJR (respectively, EJR1) if for each election $E(b)$ the outcome $\mathcal{R}(E(b))$

provides EJR (respectively, EJR1).

Peters and Skowron [12] show that MES satisfies EJR for approval utilities, and the results of Peters et al. [14] and Kraiczy and Elkind [8] imply that the same is true for EES. In contrast, Peters et al. [15] show that finding an outcome that satisfies EJR is NP-hard for uniform utilities (they reduce from Knapsack, and construct an election with a single voter, so the utilities are clearly uniform), but MES satisfies EJR1 in an even more general model, which encompasses uniform utilities (namely, arbitrary additive utilities). For completeness, we give a simple proof that EES, too, satisfies EJR1 for uniform utilities.

Theorem 3.2 (♠). *EES satisfies EJR1 for uniform utilities.*

Thus, in our setting all proportionality guarantees that have been established for MES also hold for EES.

Fast Implementation Kraiczy and Elkind [8] show how to implement EES with approval utilities in time $O(m^2n)$. We will now argue that their approach can be used to guarantee the same runtime for EES with uniform utilities.

Let R_t, $t \geq 0$, be the list of pairs $(r_i^t, i)_{i \in N}$, where r_i^t is the leftover budget of voter i after t projects have been bought, sorted in non-decreasing order of first components. As every voter starts off with a budget of $\frac{b}{n}$, we can set $R_0 = \big((\frac{b}{n}, i)_{i \in N}\big)$.

To choose the $(t+1)$-st project, i.e., an as yet unselected project with the highest bang per buck, for each unselected project p we make a single pass through the sorted list R_t in order to identify the smallest index j such that the j-th entry (r, i) in $R_t[N_p]$ satisfies $r \geq \frac{cost(p)}{|N_p|-j+1}$. The value of j determines the bang per buck offered by p; we let p_{t+1} be the project with the highest bang per buck, and let V_{t+1} be the voters who pay for p_{t+1}.

We will now explain how to quickly compute R_{t+1} given R_t. Suppose each voter in V_{t+1} pays δ_{t+1}. Then, at step $t+1$ the budgets of voters in V_{t+1} are reduced by δ_{t+1}, while the budgets of voters in $N \setminus V_{t+1}$ remain unchanged. Given the list R_t and the set V_{t+1}, in a single pass over R_t we can create sorted lists $R_t[N \setminus V_{t+1}]$ and $R_t[V_{t+1}] - \delta_{t+1}$. These two sorted lists can then be merged into R_{t+1} in time $O(n)$. Since this step has to be done at most m times, it only contributes $O(mn)$ to the overall runtime of the algorithm.

Since the algorithm keeps track of the leftover budgets list, it can easily return this as auxiliary information; this will be relevant in Section 4.

4 Towards an Efficient Completion Method

Our new completion method for EES relies on solving the following computational problem, which we call add-opt: Given the outcome of EES for budget b, compute the minimum value of d such that if every voter gets additional budget d, EES returns a different outcome, in the sense that the set of selected projects changes or some project is paid for by more voters (or both).

A key to our approach is solving a subproblem concerning a notion of stability for equal-shares solutions, where stability is understood as resistance to deviations by groups of voters. Here we define stability for approval utilities in the spirit of Peters et al. [14] and Kraiczy and Elkind [8]; we give a generalization to uniform utilities towards the end of the section.

For approval utilities, the intuition is as follows. Given a solution (W, X), voter i can deviate from it by contributing her leftover budget r_i to support further projects in A_i. Moreover, even if i does not have enough budget left to contribute to new projects, she may still deviate by withdrawing her support from a project in W and reallocating it to a more cost-efficient project.

Formally, the *leximax payment* of a voter $i \in N$ in a solution (W, X) is the pair $c_i = (x_i, p_i)$, where $x_i = \max\{x_{i,p} \mid p \in P\}$ and $p_i = \max\{p \mid x_{i,p} = x_i\}$.We say that x_i and p_i are the *leximax budget* and the *leximax project* of voter i, respectively. Given two pairs $(x, p), (x', p') \in \mathbb{Q} \times P$, we write $(x, p) <_{lex} (x', p')$ if $x < x'$ or $x = x'$ and $p \lhd p'$. Given a solution (W, X), we say that voter i is *willing to contribute* x *to* p if $x \leq r_i$ or $(x, p) <_{lex} c_i$. Note that voter i's leftover budget r_i and her leximax budget x_i serve as two distinct sources of funds that i can use to deviate.

Definition 4.1. A pair (p, V) with $p \in P$, $V \subseteq N_p$ *certifies the instability* of an equal-shares solution (W, X) for an election $E(b)$ if $|V| > |N_p(X)|$ and each voter $i \in V \setminus N_p(X)$ is willing to contribute ${}^{cost(p)}/_{|V|}$ to p. A project $p \in P$ *certifies the instability* of an equal-shares solution (W, X) for election $E(b)$ if there exists a set of voters $V_p \subseteq N_p$ such that (p, V_p) certifies the instability of (W, X) for $E(b)$. An equal-shares solution (W, X) for $E(b)$ is *stable* if there is no project $p \in P$ that certifies the instability of (W, X).

This concept of stability captures the behavior of EES, as formalized by the following proposition.

Proposition 4.2 (♠). *EES returns a stable outcome.*

We are now ready to state our key subproblem, GreedyProjectChange: Given an election $E(b) = (N, P, \{A_i\}_{i \in N}, cost, b)$, a solution (W, X) for election $E(b)$ (along with some auxiliary information), and a project p, compute the minimum budget increase d such that project p certifies the instability of (W, X) for $E(b + dn)$.

4.1 GreedyProjectChange for Approval Utilities

A simple $O(n^2)$ solution to GreedyProjectChange [8] proceeds by iterating over all values $t = |N_p(X)| + 1, \ldots, |N_p|$ and, for each voter $i \in N_p$, calculating the additional budget d that would enable i to contribute $\frac{cost(p)}{t}$ towards project p. However, we aim for a solution that is linear in n, as in practice the number of voters is large, while the number of projects is small.

Intuition As a a warm-up, consider a set of customers $[n]$ with budgets $\beta_1 \leq \cdots \leq \beta_n$ interested in jointly purchasing a service that costs c; the costs of the service have to be shared equally by all participating customers. It is well-known how to identify the largest group of customers that can share the cost of the service: it suffices to find the smallest value of i such that $\beta_i \cdot (n - i + 1) \geq c$, so that each of $i, i + 1, \ldots, n$ can afford to pay $c/(n - i + 1)$ [6].

Now, suppose $\beta_i \cdot (n-i+1) < c$ for all $i \in [n]$, so no subset of $[n]$ can afford the service, but we can offer a subsidy of d to each customer; what is the smallest value of d such that some subset of the customers can purchase the service while sharing its cost equally? We can approach this question in a similar manner: if the service is to be shared by customers $i, \ldots, n$, it suffices to set $d = \max\{0, c/(n - i + 1) - \beta_i\}$. Thus, we can compute the minimum subsidy in linear time by setting $d = \max\{0, \min_{i \in [n]}(c/(n - i + 1) - \beta_i)\}$.

GreedyProjectChange can be seen as a variant of this problem, with leftover budgets r_i playing the role of β_i and $c = cost(p)$. However, it has two additional features: first, there may be some voters who are already paying for p (i.e., $N_p(X) \neq \varnothing$), and second, the voters may choose to fund p from their leximax budgets rather than their leftover budgets. We will now argue that, despite these complications, GreedyProjectChange admits a linear-time algorithm.

Description of the algorithm Our linear-time solution to GreedyProjectChange, Algorithm 2, uses two pointers, i and j, to iterate over the two lists containing the two different sources of money voters in $O_p(X) = N_p \setminus N_p(X)$ can use to pay for p: their leftover budgets in (W, X) and their leximax payments in (W, X). Both lists are sorted in non-decreasing order. We will refer to these lists as *leftover budgets list* and *leximax payments list*, respectively.

The key local variables in our algorithm are the *per-voter price* (*PvP*), the set of *liquid-voters LQ*, and the set of *solvent* voters *SL*. The liquid voters are expected to pay for p by using their leftover budgets, while the solvent voters are expected to pay for p by deviating from another project. Algorithm 2 starts by placing all voters in *LQ*, i.e., it sets $LQ = O_p(X)$. Subsequently, a voter may be moved from *LQ* to *SL* or discarded altogether.

The set of *buyers* B consists of the voters in $N_p(X)$ (who already pay for p in (W, X)), the liquid voters, and the solvent voters. Throughout the algorithm, we maintain the property that each buyer in B is willing to pay the *per-voter price* $PvP = {}^{cost(p)}/_{|B|}$ towards p. If at some iteration B contains a voter who cannot afford this payment, they are removed, thereby increasing *PvP* in the next iteration. That is, we iterate through the values $PvP = {}^{cost(p)}/_{|N_p|}, \ldots, {}^{cost(p)}/_{(|N_p(X)|+1)}$, and update the value of d whenever it holds that every voter in B is willing to contribute ${}^{cost(p)}/_{|B|}$ towards p.

We will now explain how the sets *LQ* and *SL* evolve during the execution of the algorithm. Initially, all voters in $O_p(X) = N_p \setminus N_p(X)$, i.e., all voters who approve p, but do not contribute towards it in (W, X), are placed in *LQ* and no voter is placed in *SL*. The algorithm has three means to update these sets of voters: (1) A voter who is liquid may be relabelled as solvent (Line 12), or a voter may be removed from the set of buyers either by ceasing to be solvent (Line 8) or by ceasing to be liquid (Line 16).

It each iteration, Algorithm 2 recomputes the per-voter price by sharing the cost of p among all voters in B. Pointer j keeps track of the solvent voter with the smallest leximax payment; if this voter cannot afford the current price from her leximax budget, she is removed and j is increased by 1.

Then Algorithm 2 uses a pointer i to the leftover budgets list to identify a voter $v_i \in LQ$ with the smallest leftover budget. It checks if this voter can afford the current per-voter price from her leximax budget, i.e., if she is willing to deviate from her leximax project to p; if yes, she is moved to *SL*. Thus, we maintain the property that the leftover budgets of solvent voters do not exceed those of the liquid voters.

On the other hand, if v_i is not willing to deviate from her leximax project, then, for her to contribute *PvP* towards p, her leftover budget should be increased by at least $PvP - r_{v_i}$. In this case, we update d as $d = \min\{d, PvP - r_{v_i}\}$. We then increase the pointer i by one and remove v_i from *LQ*, and thereby from the set of buyers, as including v_i cannot reduce d below its current value. Thus, each iteration weakly decreases d.

We provide an example illustrating the executing of Algorithm 2 in the appendix.

Algorithm 2: GREEDYPROJECTCHANGE (GPC)

Input: $E(b) = (N, P, \{A_i\}_{i \in N}, cost, b)$, stable equal-shares solution (W, X), project p;
leftover budgets of voters $O_p(X)$: $r_{v_1}, \ldots, r_{v_k}$ with $k = |O_p(X)|$, $r_{v_i} \leq r_{v_{i+1}}$ for all $i \in [k-1]$;
leximax payments of voters $O_p(X)$ in (W, X): $c_{w_1}, \ldots, c_{w_k}$ where $c_{w_j} \leq_{lex} c_{w_{j+1}}$ for $j \in [k-1]$;
Output: Minimum value $d > 0$ such that p certifies the instability of (W, X) for $E(b + nd)$

$i, j \leftarrow 1, 1$;
$SL \leftarrow \varnothing$; } $B := N_p(X) \cup liquid \cup solvent$
$LQ \leftarrow O_p(X)$; }
$d \leftarrow +\infty$;
while $LQ \cup SL \neq \varnothing$ **do**
  $PvP \leftarrow \frac{cost(p)}{|N_p(X) \cup LQ \cup SL|}$;
  if $j \leq |O_p(X)|$ ***and*** $c_{w_j} <_{lex} (PvP, p)$ **then**
    $SL \leftarrow SL \setminus \{w_j\}$;
    $j \leftarrow j + 1$;
  else if $c_{v_i} >_{lex} (PvP, p)$ **then**
    $LQ \leftarrow LQ \setminus \{v_i\}$;
    $SL \leftarrow SL \cup \{v_i\}$;
    $i \leftarrow i + 1$;
  else
    $d \leftarrow \min\{d, PvP - r_{v_i}\}$;
    $LQ \leftarrow LQ \setminus \{v_i\}$;
    $i \leftarrow i + 1$;
  end
end
return d

We can characterize the running time of Algorithm 2 and the value it computes as follows.

Theorem 4.3 (♠). *Algorithm 2 runs in time $O(n)$. Moreover, given an election $E(b)$, a stable equal-shares outcome (W, X) of $E(b)$, and a project p, let d^* be the smallest value such that there exists a $B^* \subseteq N_p$ with the property that (p, B^*) certifies the instability of (W, X) in $E(b + nd^*)$. Then Algorithm 2 returns d^*.*

Our ADD-OPT algorithm for approval utilities (Algorithm 3) iterates over all projects, runs GREEDYPROJECTCHANGE for each project, and returns the minimum value of d over all such runs.

Theorem 4.4 (♠). *Let $(W, X) = EES(E(b))$, where $E(b) = (N, P, \{A_i\}_{i \in N}, cost, b)$. Given (W, X) and $E(b)$, ADD-OPT computes the minimum value d^* such that $d^* > 0$ and $EES(E(b + nd^*)) \neq (W, X)$, and runs in time $O(mn)$.*

In the appendix, we extend this result to uniform utilities, at the expense of increasing the running time to $O(m^2 n)$.

5 Empirical Evaluation

The goal of this section is to compare MES and EES (with and without suitable completion heuristics) on real-life data. To this end, we execute both of these methods

Algorithm 3: ADD-OPT

Input: $E(b) = (N, P, \{A_i\}_{i \in N}, cost, b)$, a stable equal-shares solution (W, X) for $E(b)$;
$A = [r_{v_1}, \ldots, r_{v_n}]$ where $\{v_1, \ldots, v_n\} = [n]$ and $r_{v_i} \leq r_{v_{i+1}}$ for $i \in [n-1]$;
$B = [c_{w_1}, \ldots, c_{w_n}]$ where $\{w_1, \ldots, w_n\} = [n]$ and $c_{w_j} \leq_{lex} c_{w_{j+1}}$ for $j \in [n-1]$;
Output: Minimum $d > 0$ such that (W, X) is unstable for $E(b + nd)$

```
d = +∞;
for p ∈ P do
    A' ← subarray of A restricted to voters O_p(X);
    B' ← subarray of B restricted to voters O_p(X);
    d = min{d, GreedyProjectChange(E, (W, X), p, A', B')};
end
return d
```

on over 250 real-world participatory budgeting instances, and analyze both the number of iterations and the ability of each method to find a good virtual budget. We postpone the description of the datasets and implementation to the appendix.

The key measure that we use to evaluate the performance of aggregation rules for participatory budgeting elections is their *spending efficiency*, i.e., the proportion of the budget they utilize.

Definition 5.1. Given an election $E(b)$ and an outcome W, the *spending efficiency* of W is defined as $\frac{1}{b} \cdot \sum_{p \in W} cost(p)$. The *spending efficiency* of an aggregation rule $\mathcal{R}$ on an election $E(b)$ is the spending efficiency of $\mathcal{R}(E(b))$.

Although it is possible to construct examples where EES uses a larger proportion of the actual budget, it is natural to expect that, in the absence of completion heuristics, on most instances MES has a higher spending efficiency than EES: enforcing exact equal sharing (and not using agents' leftover budgets) is likely to result in a smaller set of projects. Our experiments (see Figure 3 and Figure 4 in the appendix) confirm that this is indeed the case.

However, it is less clear what happens if one extends both of these methods with a completion heuristic. As a baseline, we execute both MES and EES with the standard ADD-ONE completion heuristic. This heuristic executes the underlying rule with budgets $b, b + n, b + 2n, \ldots$ until either all projects are selected or the next increase would result in overspending.

Our experiments on 250 Pabulib instances (Figure 9) paint a positive picture for EES: with the ADD-ONE completion heuristic in over 77% of cases for cost utilities and in over 85% of cases for approval utilities the spending efficiency of EES is at least as high as that of MES. Moreover, both for approval and for cost utilities, EES has a higher spending efficiency than MES on more than 10% of the instances.

Heuristics Our primary motivation for introducing EES is that it admits a more sophisticated completion heuristic, namely, ADD-OPT. Recall that, given a solution (W, X) for $E(b)$, ADD-OPT identifies the smallest value of d such that $EES(E(b + nd)) \neq (W, X)$. Crucially, this heuristic is based on reinterpreting the EES outcomes as outcomes that are stable in the sense of Definition 4.1; it is not clear if MES outcomes can be interpreted in this way, and, as a consequence, we cannot use ADD-OPT with MES. Indeed, for MES it is not known if the problem of finding the smallest budget increase that changes the outcome admits a polynomial-time (let alone a linear-time) algorithm.

When using EES with ADD-OPT, we start by setting $b^{(1)} = b$, and compute $(W^{(1)}, X^{(1)}) = EES(E(b^{(1)}))$. Then in each iteration i we compute $d^{(i)}$ by running ADD-OPT on $E(b^{(i)})$ and $(W^{(i)}, X^{(i)})$, and set $b^{(i+1)} = b^{(i)} + n \cdot d^{(i)}$, $(W^{(i+1)}, X^{(i+1)}) = EES(E(b^{(i+1)}))$. Just like with ADD-ONE, we repeat this procedure until the actual budget is exhausted or the next budget increment results in overspending. We also consider a complete version of this method EES + ADD-OPT (C), where we increase the budget using ADD-OPT until all projects are selected, i.e., $W^{(i)} = P$; then, among the outcomes $W^{(1)}, \ldots, W^{(i)}$ we select one that has the highest spending efficiency among all outcomes that are feasible for the original election $E(b)$. We define a complete version of MES with ADD-ONE (denoted by MES+ADD-ONE (C)) in a similar way.

Further, leveraging ADD-OPT, we define a new completion method for EES, which we call ADD-OPT-SKIP. This method modifies the ADD-OPT heuristic in two key ways. First, given an outcome of EES, we invoke GREEDYPROJECTCHANGE *only for projects not currently included* in the outcome. Second, this process is repeated until *all projects have been selected in a run of the algorithm* for the first time. It then returns the feasible outcome with the highest spending efficiency found.

We evaluate all completion methods based on two criteria: (1) spending efficiency and (2) the number of calls to the computationally expensive base method (EES or MES) required by each completion method. Our results for approval utilities are summarized in Table 1; for cost utilities see Table 2, and Figure 5b in the appendix. The first three columns refer to the number of iterations, and the last three columns refer to the spending efficiency.

For ADD-OPT, the mean per-voter budget increment size across our dataset is 37.3 units for cost utilities and 35 for approval utilities. The median of these budget increments is 6.4 for cost utilities and 4.6 for approval utilities. These values are greater than 1, which means that typically ADD-OPT considers substantially fewer budgets than ADD-ONE, while also guaranteeing the identification of a budget that maximizes the spending efficiency within the tested range.

Table 1: Comparison results: approval utilities. 'Ex' refers to the number of executions, 'Eff' refers to budget efficiency.

TMethod	Avg Ex.	Med Ex.	Std Ex.	Avg Eff.	Med Eff.	Std Eff.
MES + ADD-ONE	535.4	393.0	433.0	0.855	0.890	0.124
MES + ADD-ONE (C)	2888.7	1996.0	3132.4	0.862	0.896	0.123
EES + ADD-OPT	279.6	100.0	356.3	0.848	0.888	0.130
EES + ADD-OPT (C)	625.8	237.0	794.9	0.854	0.892	0.131
EES + ADD-OPT-SKIP	27.9	17.0	27.3	0.853	0.890	0.130
MAX	563.3	423.0	425.5	0.871	0.906	0.119

Interestingly, we observe that in some iterations ADD-OPT returns a per-voter increase of less than 1. This means that ADD-ONE may skip possible allocations, and thus is not guaranteed to find the budget that results in the most spending-efficient outcome, even if that budget lies within the tested range. Indeed, in our dataset we find over 10 such instances, demonstrating that this is not only theoretically possible, but something that occurs in realistic PB elections. In contrast, using ADD-OPT enables us to consider *every* distinct allocation within our tested range.

We provide a detailed discussion of our experimental findings in the appendix. In summary, EES with ADD-OPT-SKIP offers a viable alternative to the state-of-the-art, i.e., MES with ADD-ONE.

References

[1] Haris Aziz, Markus Brill, Vincent Conitzer, Edith Elkind, Rupert Freeman, and Toby Walsh. Justified representation in approval-based committee voting. *Social Choice and Welfare*, 48(2):461–485, 2017.

[2] Haris Aziz, Edith Elkind, Shenwei Huang, Martin Lackner, Luis Sánchez-Fernández, and Piotr Skowron. On the complexity of extended and proportional justified representation. In *AAAI'18*, 2018.

[3] Usman W Chohan. The decentralized autonomous organization and governance issues. In *Decentralized Autonomous Organizations*, pages 139–149. Routledge, 2017.

[4] Michiel S De Vries, Juraj Nemec, and David Špaček. International trends in participatory budgeting. *Cham: Palgrave Macmillan*, 2022.

[5] Piotr Faliszewski, Jarosław Flis, Dominik Peters, Grzegorz Pierczyński, Piotr Skowron, Dariusz Stolicki, Stanisław Szufa, and Nimrod Talmon. Participatory budgeting: Data, tools, and analysis, 2023. URL https://arxiv.org/abs/2305.11035.

[6] Kamal Jain and Mohammad Mahdian. Cost sharing. In *Algorithmic game theory*, chapter 15, pages 385–410. Cambridge University Press, 2007.

[7] Svante Janson. Phragmén's and Thiele's election methods. *arXiv preprint arXiv:1611.08826*, 2016.

[8] Sonja Kraiczy and Edith Elkind. An adaptive and verifiably proportional method for participatory budgeting. In *WINE'23*, pages 438–455. Springer, 2023.

[9] Sonja Kraiczy and Edith Elkind. A lower bound for local search proportional approval voting. In *ESA'24*, pages 82:1–82:14, 2024.

[10] Martin Lackner and Piotr Skowron. Approval-based committee voting. In *Multi-Winner Voting with Approval Preferences*, pages 1–7. Springer, 2023.

[11] Jeffrey B Liebman and Neale Mahoney. Do expiring budgets lead to wasteful year-end spending? evidence from federal procurement. *American Economic Review*, 107(11):3510–3549, 2017.

[12] Dominik Peters and Piotr Skowron. Proportionality and the limits of welfarism. In *ACM EC'20*, pages 793–794, 2020.

[13] Dominik Peters and Piotr Skowron. Completion of the Method of Equal Shares . https://equalshares.net, 2023. [Online; accessed 14-May-2023].

[14] Dominik Peters, Grzegorz Pierczyński, Nisarg Shah, and Piotr Skowron. Market-based explanations of collective decisions. In *AAAI'21*, pages 5656–5663, 2021.

[15] Dominik Peters, Grzegorz Pierczyński, and Piotr Skowron. Proportional participatory budgeting with additive utilities. In *NeurIPS'21*, pages 12726–12737, 2021.

[16] Edvard Phragmén. Sur une méthode nouvelle pour réaliser, dans les élections, la représentation proportionelle des partis. *Öfversigt af Kongliga Vetenskaps-Akademiens Förhandlingar*, 51(3):133–137, 1894.

[17] Thorvald Nicolai Thiele. Om flerfoldsvalg. *Oversigt over det Kongelige Danske Videnskabernes Selskabs Forhandlinger*, (2):415–441, 1895.

[18] Brian Wampler, Stephanie McNulty, and Michael Touchton. *Participatory budgeting in global perspective*. Oxford University Press, 2021.

[19] Shuai Wang, Wenwen Ding, Juanjuan Li, Yong Yuan, Liwei Ouyang, and Fei-Yue Wang. Decentralized autonomous organizations: Concept, model, and applications. *IEEE Transactions on Computational Social Systems*, 6(5):870–878, 2019.

[20] Joshua C Yang, Carina I Hausladen, Dominik Peters, Evangelos Pournaras, Regula Hänggli Fricker, and Dirk Helbing. Designing digital voting systems for citizens:

Achieving fairness and legitimacy in participatory budgeting. *Digital Government: Research and Practice*, 2024.

A Appendix

A.1 Omitted Proofs and Examples

Theorem 3.2 (♠). *EES satisfies EJR1 for uniform utilities.*

Proof. Let W be the outcome selected by Exact Equal Shares on instance $(N, P, (A_i)_{i\in N}, b, cost)$. Let S be a T-cohesive group for $T \subseteq P$. Let $Y \subseteq T$ be the set of projects in Y paid for by less than $|S|$ voters (this includes being paid by no voters). If the set Y is empty, we are done since every voter $i \in S$ has utility $u_i(W)$ for the outcome satisfying $u_i(W) \geq u(T)$. So suppose set Y is non-empty. Let $y^* \in \arg\max_{y\in Y} \frac{u(y)}{cost(y)}$. There must be some voter $i \in S$ such that the budget z_i not being used to pay for projects at bang per buck at least $\frac{u(y^*)|S|}{cost(y^*)}$ (this includes leftover budget) satisfies the inequality $z_i < \frac{cost(y^*)}{|S|}$, as otherwise the voters in S could jointly pay for a project from set Y. Now voter i may spend some of her money on projects $T \setminus Y$, each such project P it pays for at most $\frac{cost(p)}{|S|}$. So on projects in $W \setminus T$ with bang per buck at least $\frac{u(y^*)|S|}{cost(y^*)}$, i spends at least $\frac{b}{n} - \sum_{p\in T\setminus Y} \frac{cost(p)}{|S|} - z_i$. So her utility for the set $W \setminus T$ can be lower bounded as follows

$$\begin{aligned}
u_i(W \setminus T) &\geq \frac{u(y^*)|S|}{cost(y^*)} \left(\frac{b}{N} - \sum_{p\in T\setminus Y} \frac{cost(p)}{|S|} - z_i \right) \\
&\geq \frac{u(y^*)}{cost(y*)} \cdot \left(cost(T) - \sum_{p\in T\setminus Y} cost(p) \right) - \frac{|S|z_i \cdot u(y^*)}{cost(y*)} \\
&> \frac{u(y^*)}{cost(y*)} \cdot cost(Y) - |S| \frac{cost(y^*)}{|S|} \frac{u(y^*)}{cost(y*)} \\
&\geq \sum_{y\in Y} cost(y) \frac{u(y^*)}{cost(y^*)} - u(y^*) \\
&\geq \sum_{y\in Y} cost(y) \frac{u(y)}{cost(y)} - u(y^*) \qquad (2) \\
&= \sum_{y\in Y} u(y) - u(y^*),
\end{aligned}$$

where the line 2 follows since y^* gives the largest value of $\frac{u(y)}{cost(y)}$ among all projects $y \in Y$. Overall, voter i has utility at least

$$\begin{aligned}
u_i(W \setminus T) + u_i(T \setminus Y) > \\
u_i(Y) + u_i(T \setminus Y) - u(y^*) = u(T) - u(y^*)
\end{aligned}$$

for projects in W that are paid for by at least $|S|$ people, implying that after including y^* we get

$$u(W \cup \{y^*\}) \geq u(W \setminus T) + u(T \setminus Y) + u(y^*) > u(T),$$

as desired. □

Example A.1. To illustrate Algorithm 2, we consider an instance with five voters v_1, v_2, v_3, v_4, v_5 and three projects p_1, p_2 and p_3. The project costs are $cost(p_1) = 2$, $cost(p_2) = 3.2$ and $cost(p_3) = 6$, and the total budget is $b = 10$. The project approvals are given by $N_{p_1} = \{v_1, v_2\}$, $N_{p_2} = \{v_3, v_4\}$, $N_{p_3} = \{v_2, v_3, v_4, v_5\}$. It is easy to check that EES selects $W = \{p_1, p_2\}$ on this instance, with voters in N_{p_1} sharing the cost of p_1 and voters in N_{p_2} sharing the cost of p_2. We will now execute Algorithm 2 on (W, X) with $p = p_3$. Note that $|N_{p_3}| = 4$ and so *PvP* $= \frac{6}{4} = 1.5$. All voters in N_{p_3} are initially placed in *LQ*. The first two entries of the leximax payments list correspond to voters v_5 and v_2, whose leximax budgets are 0 and 1, respectively. Hence, in the first two iterations of Algorithm 2, pointer j is increased to 3. Note that v_5 and v_2 will never be placed in *SL*, because *PvP* will never drop below 1.5. For voters v_3 and v_4 their leximax budgets are $1.6 >$ *PvP*, so they remain on the list (but they are not placed in *SL* at that point). In the third iteration we consider v_3 at position $i = 1$ in the leftover budgets list. Since v_3 qualifies as solvent (her leximax budget is 1.6), we move her from *LQ* to *SL*. Similarly, in the fourth iteration we consider v_4 at position $i = 2$ in the leftover budgets list and move her from *LQ* to *SL*. In the fifth iteration, we consider v_2 at position $i = 3$ in the leftover budgets list. Since v_2 does not qualify as solvent, we update $d = \min\{+\infty, \textit{PvP} - r_{v_2}\} = 1.5 - 1 = 0.5$. We then remove v_2 from the set of buyers (in the pseudocode this is done by removing her from *LQ*) and increase i to 4.

As a result, *PvP* is increased to 2. In the next two iterations v_3 and v_4 are removed from *SL* since *PvP* > 1.6, and j is increased to 5. The last remaining buyer v_5 is liquid (but does not qualify as sD tolvent, as we know from the first iteration) and requires a subsidy of 4 to pay for p_3 on her own; since this is more than 0.5, the algorithm terminates returning 0.5. Indeed, *EES* with budget $10 + 0.5 \cdot 5 = 12.5$ will select p_1 and p_3.

To prove Theorem 4.3, we start by making an observation about the set of solvent voters.

Lemma A.2. *If Algorithm 2 attempts to remove v from SL in some iteration, then v will never be placed in SL in subsequent iterations.*

Proof. If the algorithm attempts to remove v from *SL*, this means that v satisfies the condition of the **if** in Line 7, i.e., $c_v <_{\textit{lex}} (PvP, p)$. In order for v to be added to *SL*, Line 10 must be executed, i.e. it must hold that $c_v >_{\textit{lex}} (PvP, p)$. But this is impossible since *PvP* is nondecreasing. □

We are now ready to establish that GreedyProjectChange runs in linear time.

Proposition A.3. *Algorithm 2 runs in time $O(n)$.*

Proof. Each iteration of the **while** loop takes a constant time (we can represent the sets *LQ* and *SL* as k-bit arrays). In each iteration, we increase either i or j by 1. More precisely, if the **if** condition in Line 7 is satisfied then j will be incremented, while if the **else if** condition in Line 10 or the **else** condition in Line 14 is satisfied, then i will be incremented. Further, by design, j can not exceed $|O_p(X)| + 1$. To complete the proof, we will now argue that i cannot exceed $|O_p(X)| + 1$ either, and hence the **while** loop terminates after at most $2n$ iterations.

Suppose the algorithm sets $i = |O_p(X)| + 1$ while $j \leq |O_p(X)|$. Then it has iterated through the entire leftover budgets list, so the set *LQ* must be empty at this point. For the next iteration of the **while** loop to proceed, it must be the case that *SL* $\neq \varnothing$. We

claim that in this iteration, and in all subsequent iterations, the condition in Line 7 is satisfied and hence the size of *SL* is reduced by 1. Indeed, suppose the condition in Line 7 is not satisfied. Line 7 is only executed when $LQ \cup SL \neq \varnothing$, so we have $SL \neq \varnothing$ at this point. But then $(p, N_p(X) \cup SL)$ witnesses the instability of (W, X), a contradiction with (W, X) being stable. Thus, each subsequent iteration increments j and hence the total number of iterations does not exceed $2 \cdot |O_p(X)|$.

On the other hand, suppose $j = |O_p(X)| + 1$ occurs first. Then the algorithm has iterated through the entire leximax payments list, and by Lemma A.2 the set *SL* will remain empty throughout the remainder of the algorithm. Therefore, every subsequent iteration will execute Line 16 and Line 17 and remove a voter from *LQ*. Again, we terminate after at most $2 \cdot |O_p(X)| \leq 2n$ iterations. □

Proposition 4.2 (♠). *EES returns a stable outcome.*

Proof. The proposition and its proof is subsumed by Proposition A.8. □

Theorem 4.3 (♠). *Algorithm 2 runs in time $O(n)$. Moreover, given an election $E(b)$, a stable equal-shares outcome (W, X) of $E(b)$, and a project p, let d^* be the smallest value such that there exists a $B^* \subseteq N_p$ with the property that (p, B^*) certifies the instability of (W, X) in $E(b + nd^*)$. Then Algorithm 2 returns d^*.*

Proof. Our first claim follows from Proposition A.3. We will now prove our second claim.

Let d denote the value returned by Algorithm 2. First, we will argue that $d^* \leq d$.

Lemma A.4. *There exists a $B \subseteq N_p$ such that (p, B) certifies the instability of (W, X) for $E(b + nd)$.*

Proof. Consider the iteration of the **while** loop in which d is set to its final value. Let $B = N_p(X) \cup LQ \cup SL$ be the set of buyers at this point, let $\pi = {}^{cost(p)}/_{|B|}$ be the per-voter price, and let i^* and j^* be the values of i and j, respectively. Then $d = \pi - v_{i^*}$. We will show that (p, B) certifies the instability of (W, X) for budget $b + nd$. In particular, we will show that (1) for each $v \in LQ$ we have $r_v + d \geq \pi$, (2) for each $v \in SL$ we have $c_v >_{lex} (\pi, p)$, and (3) for each $v \in N_p(x)$ we have $x_{v,p} > \pi$.

For (1), recall that Algorithm 2 increments i right after removing v_i from *LQ*, so when d is set to $PvP - r_{v_{i^*}}$, it holds that $LQ = \{v_{i^*}, \ldots, v_k\}$. Since the leftover budgets list is sorted in non-decreasing order, this implies $r_v \geq r_{v_{i^*}}$ and hence $r_v + d \geq PvP = \pi$ for each $v \in LQ$.

For (2), if $j^* > |O_p(X)|$ we have $SL = \varnothing$, so the claim trivially holds. Now, suppose that $j^* \leq |O_p(X)|$. As j^* did not trigger the condition in Line 7, and it can not be the case that $c_{w_{j^*}} = (\pi, p)$ (the voter w_{j^*} is in $O_p(X)$, so her leximax project is not p), it follows that $c_{w_{j^*}} >_{lex} (\pi, p)$. Further, whenever j is incremented, voter w_j is removed from *SL*. By Lemma A.2 it follows that every voter in *SL* must appear at index j^* or greater in the leximax payments list. Since the leximax payments list is sorted in nondecreasing order, we have $c_w \geq_{lex} c_{w^*} >_{lex} (\pi, p)$ for each $w \in SL$.

For (3), the condition of the **while** loop implies $LQ \cup SL \neq \varnothing$ and hence $|B| > |N_p(X)|$. Thus, the leximax payment of each $v \in N_p(X)$ satisfies $x_v = {}^{cost(p)}/_{|N_p(X)|} > \pi$. This completes the proof. □

Our second lemma establishes that $d^* \geq d$.

Lemma A.5. *Suppose Algorithm 2 sets d to PvP $- r_{v_i}$ for some $v_i \in O_p(X)$. Then $d^* \geq$ PvP $- r_{v_i}$.*

Proof. Note that we can assume that $N_p(X) \subseteq B^*$: otherwise, sharing the cost of p among the voters in $B^{**} = B^* \cup N_p(X)$ would lower the per-voter price and hence (p, B^{**}) would also certify the instability of (W, X) with budget $b + nd^*$. On the other hand, Algorithm 2 always includes $N_p(X)$ in the set of buyers B.

At the start of Algorithm 2 we have *LQ* $= O_p(X)$ and hence $B = N_p(X) \cup$ *LQ* $\cup$ *SL* $= N_p$. In every iteration Algorithm 2 eliminates at most one buyer, and, when it terminates (which it does by Lemma A.3), we have $B = N_p(X)$. Hence, throughout the execution *PvP* ranges over $\frac{cost(p)}{|N_p|}, \frac{cost(p)}{|N_p|-1}, \ldots, \frac{cost(p)}{|N_p(X)|+1}$. Since $|N_p| \geq |B^*| > |N_p(X)|$, there exists a last iteration q for which *PvP* $= \frac{cost(p)}{|B^*|}$. At the beginning of this iteration we have $|B| = |B^*|$.

Let i^* be the index in the leftover budgets list of the voter $v_{i^*} \in B^*$ with the lowest remaining budget whose leximax payment satisfies $c_{v_{i^*}} \leq_{lex} (cost(p)/|B^*|, p)$. We then have $d^* = cost(p)/|B^*| - r_{v_{i^*}}$.

Suppose the value of the pointer i during iteration q satisfies $i > i^*$. If v_{i^*} was removed from the set of buyers by ceasing to be liquid in iteration $\ell < q$, then the value of *PvP* was strictly smaller before the removal than in iteration q. Hence, the set of buyers B during iteration ℓ was larger than $|B^*|$, implying that after execution of Line 15 it was the case that $d = \frac{cost(p)}{|B|} - r_{v_{i^*}} < \frac{cost(p)}{|B^*|} - r_{v_{i^*}} = d^*$, a contradiction. Otherwise, v_{i^*} was moved from *LQ* to *SL* in an iteration $\ell < q$. In iteration ℓ voter v_{i^*} satisfied $c_{v_{i^*}} >_{lex} (PvP, p)$, while in iteration q it holds that $c_{v_{i^*}} \leq_{lex} (PvP, p)$. This increase in per-voter price implies that in the meantime, i.e., in some iteration ℓ_2 with $\ell < \ell_2 < q$ a voter distinct from v_{i^*} was removed from B. For a voter in *SL* to be removed from B after iteration ℓ, *PvP* must increase first, as otherwise a voter in *SL* would have been removed in iteration ℓ already after Line 7 was executed. But *PvP* only changes when we remove voters from B. Thus, at the first iteration ℓ_1 during which a voter v is removed from B, she is removed from the set *LQ*, and so $v = v_{i'}$ for some $i' > i^*$. Since $r_{v_{i'}} \geq r_{v_{i^*}}$ and the set of buyers before $v_{i'}$'s removal is strictly larger than B^*, we obtain a contradiction, as $d = \frac{cost(p)}{|B|} - r_{v_{i'}} \leq \frac{cost(p)}{|B|} - r_{v_{i^*}} < \frac{cost(p)}{|B^*|} - r_{v_{i^*}} = d^*$.

Now suppose $i \leq i^*$. Every voter $v \in B^* \setminus N_p(X)$ who has leximax payment $c_v >_{lex} (PvP, p)$ is still in *SL* or *LQ* by the end of iteration q. Every voter $v \in B^* \setminus N_p(X)$ who has $c_v <_{lex}$ *PvP* must have leftover budget $r_v \geq r_{v_{i^*}}$ by the definition of i^* and so has index $i' \geq i^* \geq i$ in the leftover budgets list. This implies that at the beginning of iteration q, $v = v_{i'}$ is in *LQ*. In other words, B^* is a subset of the set of buyers B at the beginning of iteration q. But then $|B| = |B^*|$ implies $B^* = B$. Since we chose q to be the last iteration in which *PvP* $= \frac{cost(p)}{|B^*|}$, a voter is removed from set B in this iteration, and by the previous argument no voter $v \in B^* = B$ is removed from *SL*. We conclude that in iteration q Algorithm 2 executes Line 15, and so $d = \frac{cost(p)}{|B|} - r_{v_i} \leq \frac{cost(p)}{|B^*|} - r_{v_{i^*}} = d^*$, completing the proof. □

By combining Lemmas A.4 and A.5, we obtain the desired result. □

Theorem 4.4 (♠). *Let $(W, X) = EES(E(b))$, where $E(b) = (N, P, \{A_i\}_{i \in N}, cost, b)$. Given (W, X) and $E(b)$,* ADD-OPT *computes the minimum value d^* such that $d^* > 0$ and $EES(E(b + nd^*)) \neq (W, X)$, and runs in time $O(mn)$.*

Proof. By construction, ADD-OPT returns a value d such that (W, X) is unstable for $E(b+nd)$. By Proposition 4.2, the outcomes of EES are stable, so $(W, X) \neq EES(E(b+nd))$. Hence, $d \geq d^*$.

We will now prove that $d \leq d^*$. Let $(W^*, X^*) = EES(E(b + nd^*))$. Compare the execution of EES on $E(b+nd^*)$ and $E(b)$, and let ℓ be the first iteration in which the two executions differ (they may differ by selecting different projects, or they may select the same project, but have it funded by different groups of voters; it may also be the case that $EES(E(b))$ terminates after $\ell - 1$ iterations, while $EES(E(b + nd^*))$ does not, but not the other way around). Let p be the project selected by EES on $E(b+nd^*)$ in iteration ℓ, let $V_p = N_p(X^*)$, and set $\pi = cost(p)/|V_p|$. We will show that (p, V_p) certifies the instability of (W, X) for budget $b + nd^*$; this implies $d \leq d^*$.

First, we will show that if $EES(E(b))$ selects p in some iteration $\ell' > \ell$ then the set of voters V'_p who share the cost of p in $EES(E(b))$ is a strict subset of V_p. Indeed, suppose that $V'_p \setminus V_p \neq \varnothing$. Each voter in $V'_p \setminus V_p$ can afford to pay $^{cost(p)}/_{|V'_p|}$ in iteration ℓ' in $EES(E(b))$, so each of them can afford to pay $^{cost(p)}/_{|V_p \cup V'_p|}$ in iteration ℓ in $EES(E(b+nd^*))$, a contradiction with the choice of V_p. Thus, $V'_p \subseteq V_p$. We will now argue that $V_p \setminus V'_p \neq \varnothing$. Let p' be the project chosen by $EES(E(b))$ in iteration ℓ. Since $EES(E(b + nd^*))$ favors p over p' in iteration ℓ, while $EES(E(b))$ makes the opposite choice, and the two executions are identical up to that point, it has to be the case that in $E(b)$ some voters in V_p cannot afford to pay π in iteration ℓ; this will still be the case in iteration $\ell' > \ell$. Thus, $V_p \setminus V'_p \neq \varnothing$. This means that each $v \in V'_p$ contributes more than π towards p in $EES(E(b))$.

Now, consider a voter $v \in V_p$ who does not pay for p in $EES(E(b))$ (either because p is not selected in $EES(E(b))$ or because p is selected, but $v \notin V'_p$). Let r_v be her leftover budget after EES has been executed on $E(b)$. We know that after iteration $\ell - 1$ in the execution of $EES(E(b + nd^*))$ this voter was able to pay π for p, so her remaining budget at that point in $E(b+nd^*)$ was at least π. Consequently, her remaining budget in $E(b)$ after $\ell - 1$ iterations was at least $\pi - d^*$. If she did not contribute to any projects after the first $\ell - 1$ iterations of $EES(E(b))$, we have $r_v \geq \pi - d^*$. Otherwise, she contributed to some project p' in a subsequent iteration. If voters in $E(b)$ could afford p' in iteration $\ell + 1$ or later, the voters in $E(b+nd^*)$ could afford p' in iteration ℓ. Since $EES(E(b+nd^*))$ chose p over p' in iteration ℓ, every supporter of p' at that point (in both executions) would have to contribute at least π towards p', and that would also be the case in all subsequent iterations. Thus, $X_{v,p'} \geq \pi$, and if $X_{v,p'} = \pi$, then $p \lhd p'$ (because $EES(E(b + nd^*))$ chose p over p'). Thus, we conclude that in this case $c_v >_{lex} (\pi, p)$.

Therefore, for each $v \in V_p$ with $X_{v,p} = 0$ we have $r_v \geq \pi - d^*$ or $c_v >_{lex} (\pi, p)$, and for each $v \in V_p$ with $X_{v,p} > 0$ we have $X_{v,p} > \pi$. Hence, (p, V_p) witnesses the instability of (W, X) for budget $b + nd^*$. This concludes the proof. □

Remark A.6. Algorithm 2 does not necessarily return the minimum value of d such that if each voter were given additional budget d, project p would be included in the outcome selected by EES. Indeed, if there is another project $p' \neq p$ such that Algorithm 2 returns $d' < d$ on p', then if the budget is increased to $b + nd'$, EES may select p', and this will enable it to select p at a later step.

Concretely, consider four projects p_1, p_2, p_3 and p_4 with costs $cost(p_1) = 2$, $cost(p_2) = 98$, $cost(p_3) = 100$, $cost(p_4) = 51$ and budget $b = 150$. The set of voters is $\{1, 2, 3\}$, where $A_1 = \{p_1, p_2\}$, $A_2 = \{p_2, p_3\}$, and $A_3 = \{p_3, p_4\}$. EES selects $\{p_1, p_3\}$. For project p_4, Algorithm 2 returns $d = cost(p_4) = 51$. However, project p_2 certifies the instability of $\{p_1, p_3\}$ for budget $b' = b + 3d'$, where $d' = 1 < d$, and the outcome selected by EES with budget 153 is $\{p_1, p_2, p_4\}$.

A.2 Uniform Utilities

In practice, MES is typically used under the assumption of cost utilities. In this section, we extend our approach to handle uniform utilities, which encompass approval utilities and cost utilities as special cases. The distinctive feature of approval utilities is the inverse relationship of the bang per buck of a project p for a voter i and the voter's payment $x_{i,p}$. Specifically, if one project can be bought at a higher bang per buck than another, that precisely means it is cheaper for the voter. Consequently, if the voter is willing to contribute some amount to p, she can do so by deviating from *at most one* project. In contrast, for more general utilities it is no longer sufficient to simply reallocate support from a single project with a lower bang per buck. Instead, it may require withdrawing support from *multiple* projects, thereby significantly increasing the combinatorial complexity of the problem. This suggests that extending the algorithm to handle general uniform utilities may be less efficient.

EES can be implemented so that it returns auxiliary information, such as voters' leftover budgets in non-decreasing order, without increasing its runtime of $O(m^2n)$. Similarly, we can trivially modify EES to return the selected projects in W in the order they were selected, i.e. in order of non-increasing bang per buck (with lexicographic tie-breaking). We will denote this sequence as $p_1, p_2, \ldots, p_w$ where $w = |W|$. Using this auxiliary input, GreedyProjectChange for uniform utilities can be implemented in time $O(m+n)$.The overall solution can be computed in time $O(m^2n)$. To achieve this, we generalize our definition of stability. For this section we define the relation $<_t$ for $(x,p) >_t (y,p')$, $x, y \geq 0$ and $p, p' \in P$ to mean $x > y$ or $x = y$ and $p \lhd p'$[3]. Consider again the approval utility case where $u(p) = 1$ for every project $p \in P$.

Lemma A.7. *Given outcome (W, X) and $u(p) = 1$ for every $p \in P$, voter v is willing to contribute $\frac{cost(p)}{t}$ if and only if the sum of her leftover budget and the total amount she spends on less preferred projects p' in the set $\{p' \mid (BpB(p'), p') <_t (\frac{u(p)t}{cost(p)}, p)\}$ exceeds $\frac{cost(p)}{t}$.*

Proof. Let $v \in V \setminus N_p(X)$ has $r_v \geq \frac{cost(p)}{|V|}$ or $(\frac{cost(p)}{|V|}, p) <_{lex} c_v$. In the latter case, the voter spends at least $\frac{cost(p)}{|V|}$ on a less preferred project p'. So the sum of her leftover budget and the budget she spends on less preferred projects is at least $\frac{cost(p)}{|V|}$, as desired.

For the other direction of the claim, suppose now voter v has $t_v \geq \frac{cost(p)}{|V|}$ where t_v is the combined total of r_v and the money spent on projects p' in $\{p' \mid (BpB(p'), p') <_t (\frac{u(p)t}{cost(p)}, p)\}$. If the latter set is empty, then $r_v \geq \frac{cost(p)}{|V|}$. If it is non-empty, then such a project p' has

$$BpB(p') \leq \frac{u(p)|V|}{cost(p)} \iff \frac{u(p')|N_{p'}(X)|}{cost(p')} \leq \frac{u(p)|V|}{cost(p)} \iff$$
$$\frac{|N_{p'}(X)|}{cost(p')} \leq \frac{|V|}{cost(p)} \iff \frac{cost(p)}{|V|} \leq \frac{cost(p)}{|N_{p'}(X)|}.$$

So it follows that $(\frac{cost(p)}{|V|}, p) <_{lex} (\frac{cost(p)}{|N_{p'}(X)|}, p')$ implying in particular that $(\frac{cost(p)}{|V|}, p) <_{lex} c_v$. So $r_v \geq \frac{cost(p)}{|V|}$ or $(\frac{cost(p)}{|V|}, p) <_{lex} c_v$ hold, implying that v is willing to contribute $\frac{cost(p)}{|V|}$ to p. □

[3] x and y represent potential values of bang per buck (*BpB*), where larger values are preferred, unlike for *PvP* in Section 4, and so we cannot use $(x,p) >_{lex} (y,p')$.

With this result in hand, we can now overload the definition of willingness to contribute in the case of uniform utilities. Given a pair (W, X), we now say that voter i is *willing to contribute* $\frac{cost(p)}{t}$ to p if the second condition in Lemma A.7 holds. With this updated definition, Definition 4.1 of what it means for (p, V) to certify the instability of (W, X) applies to uniform utilities. Note that this definition is equivalent to Definition 4.1 if $u(p) = 1$ by Lemma A.7.

Proposition A.8. *EES returns a stable (for uniform utilities) outcome.*

Proof. Let E be an election and let $(W, X) = EES(E)$. We can trivially modify EES to return the selected projects in W in the order they were selected, i.e. in order of non-increasing bang per buck (with lexicographic tie-breaking). We will denote this sequence as $p_1, p_2, \ldots, p_w$ where $w = |W|$. Suppose for the sake of contradiction that (W, X) is unstable, as certified by a pair (p, V). Then for every $v \in V$ we have $r_i(v) \geq {cost(p)}/{|V|}$ where i is the smallest index for which $(BpB(p_i), p_i) <_t (\frac{u(p)|V|}{cost(p)}, p)$. Since (W, X) is unstable, we have that i is well-defined. Now consider the project selected in the ith iteration of EES in which p_i is selected. By the definition of EES the project p is affordable by voters V in this round since by the choice of i, $r_i(v) \geq {cost(p)}/{|V|}$. Furthermore, since $(BpB(p_i), p_i) <_t (\frac{u(p)|V|}{cost(p)}, p)$ holds, project p has higher priority than p_i, and so would be selected by EES instead. This contradicts that EES returns (W, X) and the project ordering projects $p_1, \ldots, p_w$. We conclude that (W, X) is stable. □

We now show how to compute $L_1, \ldots, L_w$ used in Algorithm 4 and computed in Algorithm 5 given L_{w+1} and $p_1, \ldots, p_w$ using dynamic programming.

Lemma A.9. *Suppose we have W given in order $p_1, \ldots, p_w$ such that $(BpB(p_i), p_i) >_t (BpB(p_{i+1}), p_{i+1})$ as well as L_{w+1}. Then we can compute $L_1, \ldots, L_w$ in time $O(mn)$.*

Proof. L_i will contain the values $r_i(v)$ sorted in non-decreasing order. We note that either a voter contributes to project p_i or she does not, and in the former case, every such voter contributes an equal amount by the definition of exact equal shares. So to obtain L_i from L_{i+1} it suffices to merge the sorted lists $L_{i+1}[N_{p_i}(X)] + \frac{cost(p_i)}{|N_{p_i}(X)|}$ (voters who pay for p_i) and $L_{i+1}[O_{p_i}(X)]$ (voters who do not pay for p_i) in time $O(n)$. Since we create $w = O(m)$ lists this way, the overall runtime is $O(mn)$. □

For readability, from now on we will refer to outcomes that are stable for uniform utilities as simply stable.

A.3 Time Complexity and Correctness

Algorithm 4, GreedyProjectChange for uniform utilities, solves the problem of finding the minimum per voter budget increase d such that project p certifies the instability for $E(b + nd)$ for uniform utilities. The key insight is that, under uniform utilities, project costs being shared exactly equally combined with uniform utilities give rise to the uniform bang per buck, given by $BpB(p) = \frac{u(p)N_p(X)}{cost(p)}$, for each contributing voter in $N_p(X)$. Similarly to Algorithm 2, for given $p \in P$ we compute the minimum budget increase so that p will certify the instability of (W, X): For each project p we calculate the additional budget per voter required so exactly t voters can afford to pay for p for the first time for each $t = |N_p|, \ldots, |N_p(X)| + 1$. Leveraging the fact that we only need to consider the $t - |N_p(X)|$ richest voters in $O_p(X)$ as measured by how much they are

willing to contribute with t voters then yields a computationally efficient solution for uniform utilities.

Key input data to Algorithm 4 consists of the $w+1$ lists $L_1, \ldots, L_{w+1}$, where each list L_i, $i \leq w$ corresponds to a distinct project $p_i \in W$ and L_{w+1} contains voters' leftover budgets sorted in nondecreasing order. These lists can be computed in time $O(mn)$ as a preprocessing step in ADD-OPT for uniform utilities (Algorithm 5). Define $r_{w+1}(v) = r_v$ and for $i \in [w]$ $r_i(v) = r_{i+1}(v) + x_{v,p_i}$, representing the total amount v spends on projects $p_i, \ldots, p_w$ and her leftover budget. For $i \geq 1$, the i-th list L_i contains, in non-decreasing order, for each voter the total budget $r_i(v)$ that each voter v contributes to projects "no better than" project p_i combined with the voter's leftover budget. Specifically, this includes precisely those projects $p \in W$ with $(\mathit{BpB}(p), p) <_t (\mathit{BpB}(p_i), p_i)$. To identify voters willing to pay, the lists $L_1, \ldots, L_{w+1}$ are particularly useful due to the following simple observation.

Lemma A.10. *Voter v is willing to contribute $\frac{cost(p)}{t}$ to project p if and only if for some $i \geq 1$ satisfying $(\frac{u(p)t}{cost(p)}, p) >_t (\mathit{BpB}(p_i), p_i)$ it holds that $r_i(v) \geq \frac{cost(p)}{t}$ or else $r_v \geq \frac{cost(p)}{t}$.*

Algorithm 4: GREEDYPROJECTCHANGE (GPC) for uniform utilities

Input: $E = (N, P, \{A_i\}_{i \in N}, b, cost)$, equal shares solution (W, X), project p; lists $L_1, \ldots, L_w, L_{w+1}$

Output: Minimum $d > 0$ such p certifies the instability of (W, X) for $E(b + dn)$

```
d ← ∞;
ℓ ← 1;
i ← w + 1;
while ℓ ≤ |O_p(X)| do
    i ← min{i | (u(p) / (cost(p)/(ℓ+|N_p(X)|)), p)) >_t (BpB(p_i), p_i)} ∪ {w + 1};
    d ← min{d, cost(p)/(ℓ+|N_p(X)|) − L_i[|O_p(X)| − ℓ]};
    ℓ ← ℓ + 1;
end
return d;
```

With this auxiliary information in hand, for each project p and each $t = |N_p|$ to $t = |N_p(X)| + 1$ Algorithm 4 computes the budget increase d needed so that at least t voters would deviate and collectively pay for p.

Lemma A.11. *Algorithm 4 returns the minimum amount d^* such that there exists a set of voters V such that (V, p) certifies the instability of (W, X) for $E(b + d^*n)$.*

Proof. For $\ell = 1, \ldots, |O_p(X)|$ let d_ℓ be the minimum amount such that there exists a set of voters V_ℓ of size $|N_p(X)|+\ell$ such that (V_ℓ, p) certifies the instability of (W, X) for $E(b+nd_\ell)$. Clearly $d^* = \min_{\ell \in [|O_p(X)|]} d_\ell$, so it suffices to prove that (1) $d_\ell = \frac{cost(p)}{\ell+|N_p(X)|} - L_i[|O_p(X)| - \ell]$ where i is the index at the end of the ℓth iteration of Algorithm 4, and (2) we have identified a corresponding set V'_ℓ that certifies the instability of (W, X) for $E(b + nd_\ell)$. Let V'_ℓ be the set of voters corresponding to the last ℓ entries of list L_i where i is defined as in Line 5 of Algorithm 4 for our value of ℓ. After an increase in budget by an amount of $\frac{cost(p)}{\ell+|N_p(X)|} - L_i[|O_p(X)| - \ell]$ every voter in V'_ℓ is willing to contribute an amount $\frac{cost(p)}{\ell|N_p(X)|}$ to p. Thus, $d_\ell \leq \frac{cost(p)}{\ell+|N_p(X)|} - L_i[|O_p(X)| - \ell]$. Observe that every voter is willing to use the money corresponding to their entry in in L_i to contribute to p with $\ell + |N_p(X)|$ or more

voters by the definition of i and similarly, by the definition of i no voter is willing to give up support for a project p_j with $j < i$. So any other set V_ℓ'' of size $|N_p(X)| + \ell$ satisfied $\min_{v \in V_\ell''} r_i(v) \leq \min_{v \in V_\ell'}$. This implies voters in V_ℓ'' need at least as much additional budget as voter V_ℓ', implying that $d_\ell \geq \frac{cost(p)}{\ell + |N_p(X)|} - L_i[|O_p(X)| - \ell]$. This completes the proof. □

Lemma A.12. *Algorithm 4 can be implemented with runtime $O(m+n)$.*

Proof. Since ℓ increases by 1 in every round, the while loop terminates in $O(n)$ rounds. Thus, we only need to justify that Line 5 can be implemented efficiently, so that the overall runtime does not exceed $O(m+n)$. The key observation is that the values of i are non-increasing. Suppose that ℓ increases to $\ell_2 > \ell$ so $t = |N_p(X)| + \ell$ increases to $t_2 = |N_p(X)| + \ell_2$. If $(\frac{u(p)}{\frac{cost(p)}{t}}, p) >_t (BpB(p_i), p_i)\}$ then also $(\frac{u(p)}{\frac{cost(p)}{t_2}}, p) >_t (BpB(p_i), p_i)\}$.

So since $(BpB(p_i), p_i) >_t BpB(p_{i+1}, p_{i+1})$ for all $i = 1, \dots w-1$, it follows that

$$\min\left\{i \mid \left(\frac{u(p)}{cost(p)/t}, p\right) >_t (BpB(p_i), p_i)\right\} \cup \{w+1\} \geq \min\left\{i \mid \left(\frac{u(p)}{cost(p)/t_2}, p\right) >_t (BpB(p_i), p_i)\right\} \cup \{w+1\}.$$

It follows that as ℓ increases, i does not increase. So it suffices to simply decrease i until the condition $(\frac{u(p)}{\frac{cost(p)}{\ell + |N_p(X)|}}, p) >_t (BpB(p_i), p_i)$ is satisfied. Since i is initialized to $w+1 = O(m)$, we conclude that Algorithm 4 runs in time $O(m+n)$. □

Analogously to Section 4, we define ADD-OPT for uniform utilities (Algorithm 5) and show that it returns the minimum budget increase resulting in instability and runs in time $O(m^2 n)$.

Algorithm 5: ADD-OPT for uniform utilities

Input: $E = (N, P, \{A_i\}_{i \in N}, cost, b)$, equal shares solution (W, X) for E; $p_1, \dots, p_w$ where $(BpB(p_i), p_i) >_t (BpB(p_{i+1},)p_{i+1})$, $L_{w+1} = [r_{v_1}, \dots, r_{v_n}]$ where $r_{v_i} \geq r_{v_{i+1}}$, $v_i \neq v_j, i < j$ and r_v is v's leftover budget in (W, X);
Output: Minimum $d > 0$ such that (W, X) is unstable for $E(b + dn)$

1 $d = +\infty$;
2 $L_\ell = [r_\ell(v_{\ell,1}), \dots, r_\ell(v_{\ell,n})]$ where $r_\ell(v_{\ell,i}) \geq r_\ell(v_{\ell,i+1})$, $v_{\ell,i} \neq v_{\ell,j}, i < j$ for each $p_\ell \in W$;
`// Implementation discussed in Lemma A.9`
3 **for** $p \in P$ **do**
4 $\quad d = \min(d, \text{GPC}(E, (W, X), p, L_1[O_p(X)], \dots, L_{w+1}[O_p(X)])$;
5 **end**
6 **return** d;

Theorem A.13. *Algorithm 5 can be implemented in time $O(m^2 n)$.*

Proof. By Lemma A.9 lists $L_1, \dots L_w$ can be computed in time $O(mn)$ from the ordering $p_1, \dots, p_w$ and L_{w+1}. For the remainder of Algorithm 5, we execute Algorithm 4 m times which by Lemma A.12 can be implemented in time $O(m+n)$. However, when calling GreedyProjectChange in Line 5, we partially copy the lists $L_1, \dots, L_{w+1}$ to obtain the sublists $L_1(O_p(x)) \dots L_{w+1}(O_p(X))$ which takes time $O(mn)$, resulting in an overall runtime of $O(m^2 n)$. This completes the proof. □

Theorem A.14. *Algorithm 5 returns the minimum budget $b^* > b$ such that $EES(E^*) \neq (W, X)$ where $E^* = (N, P, \{A_i\}_{i \in N}, b^*, cost)$*

Proof. The proof proceeds in analogy to the proof of Theorem 4.4 with minor differences. Let $(W^*, X^*) = EES(E(b + nd^*))$. Since $(W, X) \neq (W^*, X^*)$ we show that there exists (p, V_p) that certifies the instability of (W, X) for budget b^*, i.e. after an increase of $\frac{b^*-b}{n}$ per voter from budget b; indeed, there exists a first iteration, say iteration q, in which EES selects a project p paid for by voters V_p for $E(b+nd^*)$, such that in the qth iteration for $E(b)$ for budget b the same does not happen (more precisely, the algorithm may terminate before iteration q or it may not select p in iteration q, or it may be pair for by a different set of voters). For budget b, either p is never selected or it is eventually selected. In the first case this means that for budget b^* at the end of iteration $q-1$, every voter $v \in V_p$ has $r_i(v) \geq \frac{cost(p)}{|V_p|}$ for some i with $BpB(p_i), p_i) <_t (\frac{u(p)|V_p|}{cost(p)}, p)$ which by Lemma A.10 implies that (V_p, p) certifies the instability of (W, X) for budget b^*, as claimed. If instead p is (eventually) selected for budget b, it will be paid for by a strict subset of the voters V_p; indeed if there was be a voter $v \notin V_p$ paying for p, then this voter would also pay for p in EES run on $E(b^*)$. Clearly p cannot be paid for by all of V_p, as then EES would select p paid for by V_p in iteration q for both elections E and E^*, contrary to our assumption. However, as before, every voter $v \in V_p$ has $r_i(v) \geq \frac{cost(p)}{|V_p|}$ for some i with $BpB(p_i), p_i) <_t (\frac{u(p)|V_p|}{cost(p)}, p)$ which by Lemma A.10 implies that (V_p, p) certifies the instability of (W, X) for budget b^*, as claimed. So since an increase of $\frac{b^*-b}{n}$ per voter results in p certifying the instability of E, by Theorem 4.3, Algorithm 2 for p will return an amount $d \leq \frac{b^*-b}{n}$, so that the output b' of Algorithm 3 is at most b^*. Furthermore, for $b' = b + dn$ where d is computed by Algorithm 3, there exists (p', V'_p) that certifies instability of (W, X). Since EES returns stable outcomes, this implies that for budget b', EES does not return (W, X) i.e. $EES(E(b')) \neq (W, X)$ and so by our definition of b^* it follows that $b^* \leq b'$. This concludes the proof. □

A.4 Lower bound on the number of distinct outcomes of EES

Since the algorithms in the previous section aim to find the next budget at which EES produces a different outcome, and given that for a sufficiently large budget all projects will be selected[4], a natural question arises: How many distinct outcomes are there? In other words, how large can the set

$$\{EES(E(b), u) \mid b > 0\}$$

be as a function of the instance size for uniform utilities u?

We are particularly interested in determining whether this size can be bounded by a polynomial in the size of the instance. For approval utilities, we leave this question as an open problem. However, for cost utilities, we answer this question in the negative by presenting an instance with exponentially many different outcomes relative to the size of the instance.

Theorem A.15. *There exists an instance E of size $O(m^3)$ and budgets $b_1 < b_2 < \ldots < b_{2^m}$ such that for $(W_i, X_i) = EES(E(b_i), cost)$ it holds that $W_i \neq W_j$ for any $i \neq j$, $1 \leq i, j \leq 2^m$.*

Proof. We consider the budgets $b_i = \sum_{j=1}^{m} 2^{j-1} d_{i,j}$ where $d_{i,m} \ldots d_{i,2} d_{i,1}$ is the binary expansion of i for $i = 1, \ldots, 2^m$. We construct $E(b) = (N, P, \{A_i\}_{i \in N}, cost, b)$ where

[4] without loss of generality, we assume that every project is approved by at least one voter

$N = V \cup \cup_{i=1}^{m} D_i$ is a set of $n = 2m^2 + m + m^3$ voters and the set of projects is $P = \{p_1, \ldots, p_m, a_1, \ldots, a_m\}$. We set $cost(p_i) = 2^i \left(\frac{2m^2+i}{n}\right) < 2^{i+1}$ for $i \in [m]$. We set the price of a_i to $cost(a_i) = 2^i \frac{m-i+m^2}{n}$ to $i \in [m]$. The set of voters D_i for each $i \in [m]$ has size m^2 and each voter in D_i approves only a_i. The set of voters V has size $2m^2 + m$. For each $i \in [m]$, the project p_i is approved by exactly $2m^2 + i$ voters among V, and the approvals are distributed in such a way that every voter in V does not approve at most one project p_i. This is possible because each project p_i is not approved by $2m^2 + m - (2m^2 + i) = m - i$ voters from V which amounts to a total of $\sum_{i=0}^{m-1} i = \frac{(m-1)m}{2}$ pairs (v, p_i) such that $v \in V$ does not approve $p_i, i \in [m]$. So we can make sure that less than m^2 agents among V do not approve one project p_i, $i \in [m]$. To complete the approval sets, a voter in V who does not approve p_i, does approve a_i.

We claim that $(W_i, X_i) = EES(E(b_i))$ satisfies $W_i \cap P = \{p_j \mid d_{i,j} = 1\}$. Note that if all its $2m^2 + j$ supporters contributed equally, the *PvP* of p_j is $\frac{cost(p_j)}{2m^2+j} = \frac{2^j}{n}$ and its bang per buck is $2m^2 + i$. Suppose $d_{i,j_1}, \ldots, d_{i,j_k}$, where $j_\ell > j_{\ell+1}$, are all equal to 1 and $d_{i,j}$ for $j \notin \{j_1, \ldots, j_k\}$ is equal to 0. We claim EES selects $p_{j_1}, \ldots, p_{j_k}$ in this order, shared exactly by all the respective projects supporters and then proceeds to select the projects $a_{j_1}, \ldots, a_{j_k}$ in some order among the set of projects $a_1, \ldots, a_m$, resulting in all voters in V having run out of money. it potentially selects further projects among $a_1, \ldots, a_m$ and terminates.

We prove the claim by induction. Consider project p_m and suppose first that $d_{i,m} = 1$. We claim that p_m is the first project to be selected and is fully paid by all of its supporters. First of all p_m can be afforded by its supporters since $b_i \geq 2^m$ and $\frac{cost(p_m)}{2m^2+m} = \frac{2^m}{n}$ and each voter has budget $\frac{b_i}{n} \geq \frac{2^m}{n}$. Indeed, each project a_i is supported by m^2 voters and so can achieve a *BpB* of at most $m^2 < 2m^2 + m$, where $2m^2 + m$ is the *BpB* if p_m is paid for by all of its supporters. Similarly every project p_j, $j < m$ has smaller *BpB* (namely $2m^2 + j$) even if every agents contributes. Now suppose $d_{i,m} = 0$. In this case $b_i \leq 2^m - 1$ and so $\frac{cost(p_m)}{2m^2+m} = \frac{2^m}{n} > \frac{b_i}{n}$ and so p_m cannot be afforded (even if all of its supporters contributed).

For the inductive step, consider project $p_j \notin W$, $j < m$, and assume that for all ℓ with $m \geq \ell > j$ it holds that

1. if $d_{i,\ell} = 1$, p_ℓ has been selected by EES and is paid for by all its supporters,
2. if $d_{i,\ell} = 0$, p_ℓ has not been selected by EES.

Furthermore, we assume no project p_ℓ with $\ell \leq j$ has been selected and all projects $a_1, \ldots, a_m$ are either not affordable or affordable at a bang per buck at most $m^2+m \leq 2m^2$. First suppose that $d_{i,j} = 1$, we will show that in this case p_j can be paid for equally by all its $2m^2 + j$ supporters, and since it has the largest bang per buck among all the affordable projects, is the next in line to be selected. The supporters of p_j have each spent at most $\frac{\frac{2^m d_m + \ldots + 2^{j+1}}{2m^2} d_{j+1}}{n}$ on projects $p_{j+1}, \ldots, p_m$ and in particular have at least $\frac{2^j}{n}$ leftover budget per voter. This is precisely the price per voter if all the $2m^2 + i$ supporters of p_i pay for p_i together as $\frac{cost(p_j)}{2m^2+j} = \frac{2^j}{n}$.
Now suppose $d_{i,j} = 0$. All except less than m^2 voters from V have spent exactly $\frac{\frac{2^m d_{i,m} + \ldots + 2^{j+1}}{2m^2} d_{i,j+1}}{n}$. These voters therefore have a leftover budget of less than $\frac{2^j}{n}$, implying that even if every voter contributed towards p_j, they would not have enough leftover budget. So the largest bang per buck for p_j we can obtain is less than m^2. Since

$d_{i,m}, \dots d_{i,2}d_{i,1}$ is the binary expansion of i it holds that $d_j = 1$ for some $j \in [m]$. The corresponding project a_j is affordable at a bang per buck $m - j + m^2 > m^2$ and so would be selected before p_j. Furthermore, any for any j with $1 \leq \ell < j$ if $d_\ell = 1$, then the corresponding bang per buck is $2m^2 + \ell > 2m^2 > m^2 + m \geq m^2 + m - j$, such a project is selected before p_i and before any a_ℓ, $\ell \in [m]$. This shows that indeed the first projects to be selected are exactly $p_{j_1}, \dots, p_{j_k}$. It remains to show that no p_j, $j \notin \{j_1, \dots, j_k\}$ is selected subsequently. Suppose $d_{i,j} = 1$ and so p_j was selected. There are $m - j$ voters in V who did not pay for p_j and have *exactly* $\frac{2^j}{n}$ leftover budget (since by construction every voter in V does not approve at most one project in $\{p_1, \dots, p_m\}$. These voters all approve a_j and a_j can be bought at a bang per buck of $m^2 + m - j$ at a per voter cost of exactly $\frac{cost(p_j)}{m^2+m-j} = \frac{2^j}{n}$ since every supporter of a_j has leftover budget at least $\frac{2^j}{n}$. Any project p_ℓ with $d_\ell = 0$ we previously argued has a bang per buck of less than m^2, so all affordable projects a_j will be prioritized over affordable projects p_ℓ. It follows that EES selects each a_j with $j \in \{j_1, \dots, j_k\}$. After this, no voter V has a leftover budget as either they approve all projects $p_{j_1}, \dots p_{j_k}$ and spent exactly $\frac{b_i}{n} = \frac{d_{i,j_1}2^{j_1}+\dots+d_{i,j_k}2^{j_k}}{n}$ on them or they approve all but one project $p_{j\ell}$ and spent exactly $\frac{b_i}{n} - \frac{2^{j_\ell}}{n} = \frac{d_{i,j_1}2^{j_1}+\dots+d_{i,j_k}2^{j_k}}{n} - \frac{2^{j_\ell}}{n}$ on projects $p_{j_1}, \dots p_{j_k}$ and the remaining $\frac{2^{j_\ell}}{n}$ budget on a_{j_ℓ}. So none of the supporters of p_ℓ for $d_\ell = 0$ has any leftover money. This completes the proof. □

A.5 Experiments: Further Details

Datasets All our experiments were conducted on real-world data from `Pabulib`, the Participatory Budgeting Library [5]. `Pabulib` contains detailed information on over 300 participatory budgeting elections that took place between 2017 and 2023, of which we analyze 250; this selection was made due to time limit of 24 hours to complete our most computationally expensive experiments.

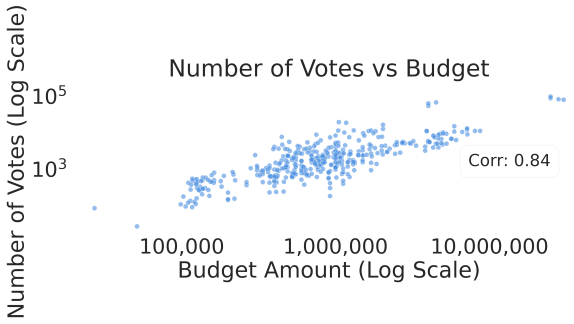
Figure 2 presents key characteristics of our dataset, including distributions of voters, projects, and budgets.

Implementation We use the `pabutools` Python library [5] to calculate MES outcomes. To monitor the number of calls to MES, we implement custom versions of the completion methods for MES. Similarly, we implement custom Python code for EES and all completion heuristics defined in this section. The source code for our implementation is available at https://github.com/psherman2023/Scalable_Proportional_PB/tree/master.

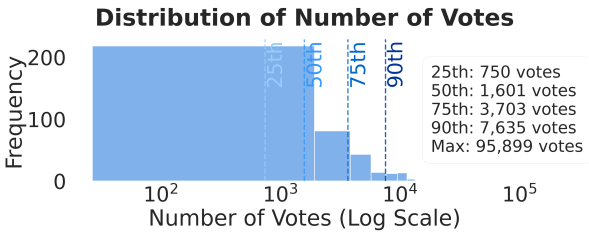
Table 2: Comparison results: cost utilities.

Method	Avg Ex.	Med Ex.	Std Ex.	Avg Eff.	Med Eff.	Std Eff.
MES + ADD-ONE	465.6	346.0	431.4	0.900	0.944	0.110
MES + ADD-ONE (C)	2894.9	2033.0	3125.8	0.902	0.945	0.109
EES + ADD-OPT	432.7	106.0	751.2	0.881	0.944	0.140
EES + ADD-OPT (C)	1263.6	360.0	1812.9	0.882	0.945	0.140
EES + ADD-OPT-SKIP	12.4	10.0	7.3	0.855	0.903	0.138
MAX	478.0	357.0	428.9	0.909	0.950	0.103

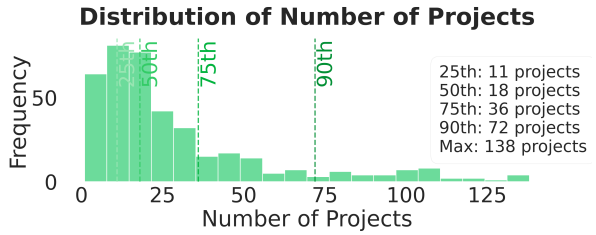
Results and Discussion



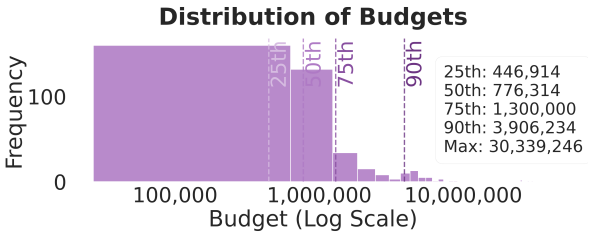
(a) Number of voters versus budget amount in the selected Pabulib instances.



(b) Histogram of the number of voters across all selected Pabulib instances.



(c) Histogram of the number of projects across all selected Pabulib instances.



(d) Histogram of the budget size across all selected Pabulib instances.

Figure 2: Dataset characteristics showing the distribution of voters, projects, and budgets across all analyzed Pabulib instances.

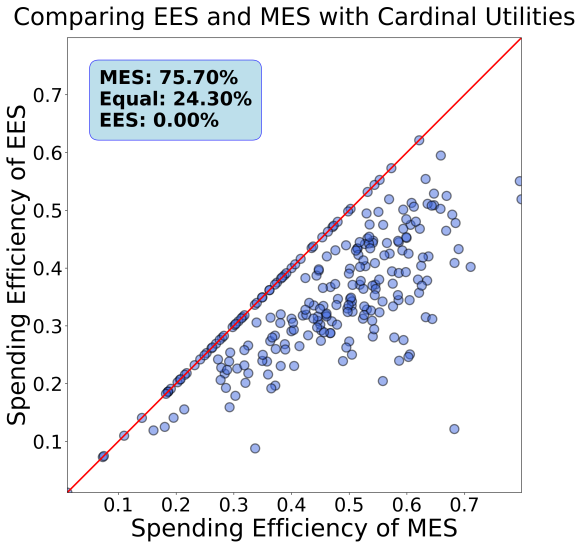


Figure 3: A comparison of the spending efficiency of MES vs. EES without a completion method: approval utilities.

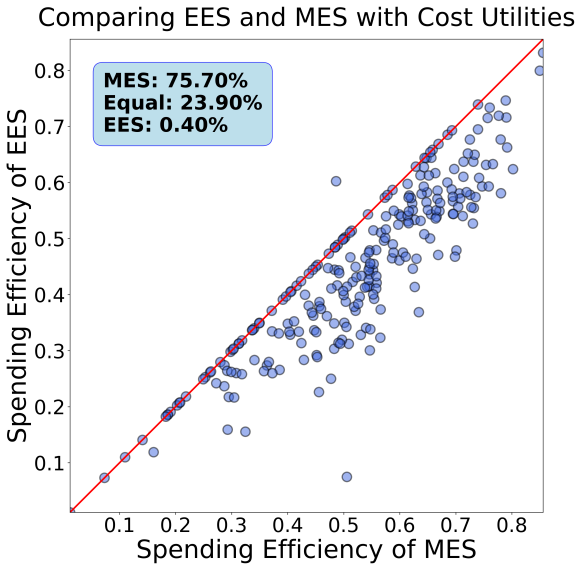
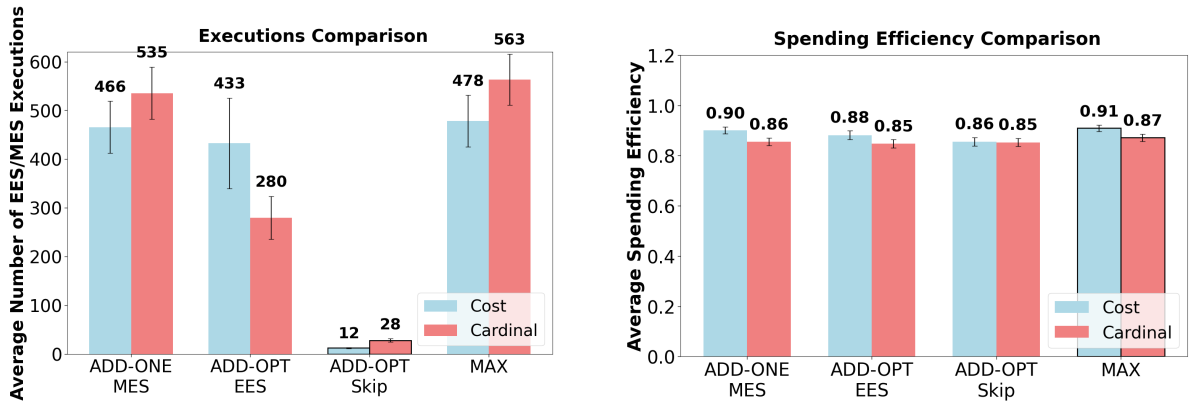


Figure 4: A comparison of the spending efficiency of MES vs. EES without a completion method: cost utilities.

Findings Our experiments highlight the advantages of ADD-OPT-SKIP. Below are our key findings:

1. **EES with ADD-OPT-SKIP requires an order of magnitude fewer calls to the base**



(a) EES vs. MES number of iterations

(b) EES vs. MES spending efficiency

Figure 5: MAX is the result of running MES + ADD-ONE and EES + ADD-OPT-SKIP and taking the result with the higher spending efficiency.

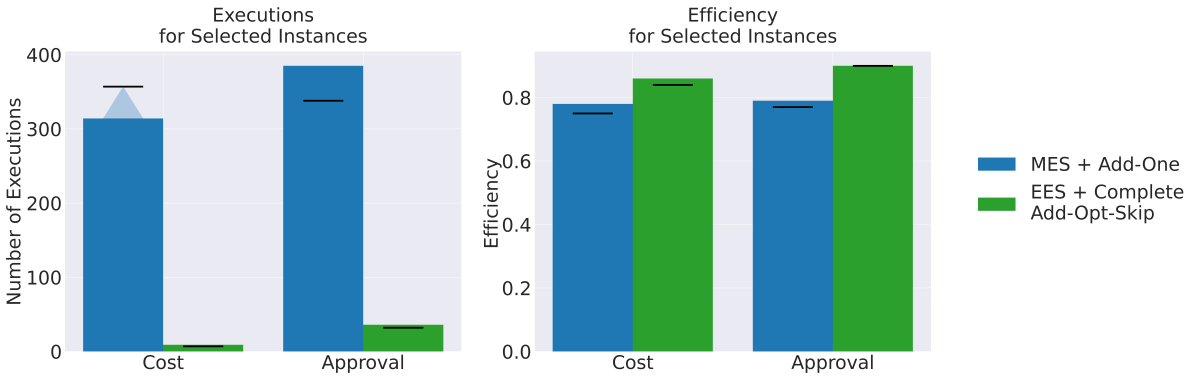


Figure 6: Graph showing executions and spending efficiency for identified cases where the optimal virtual budget occurs after the point at which the true budget is first overspent. In these cases, EES with ADD-OPT-SKIP achieves much higher spending efficiency than MES with ADD-ONE, increasing from an average of 78% to 86% for the 3 identified instances with cost utilities and from 78% to 90% for the 14 identified instances with approval utilities.

method than MES with ADD-ONE:

- For approval utilities, the average number of calls drops from 535 to just 28.
- For cost utilities, the average number of calls decreases from 466 to only 12.
- Despite this, EES with ADD-OPT-SKIP provides comparable spending efficiency: 0.85 for approval utilities (vs. 0.86 for MES with ADD-ONE) and 0.86 for cost utilities (vs. 0.9 for MES with ADD-ONE).

2. **EES with ADD-OPT-SKIP often outperforms MES in spending efficiency:**

   - In 85% of datasets, EES with ADD-OPT-SKIP achieves spending efficiency that is at least as high as that of MES with ADD-ONE, with strictly higher efficiency in 16% of cases for approval utilities. For cost utilities, its spending efficiency is at least as high as that of MES with ADD-ONE on 55% of the datasets and strictly higher on 8% of the datasets.

3. **High spending efficiency on non-monotone instances:**

   - In some real-world instances such as the one in Figure 1, the optimal virtual budget (in terms of spending efficiency) is larger than the smallest virtual

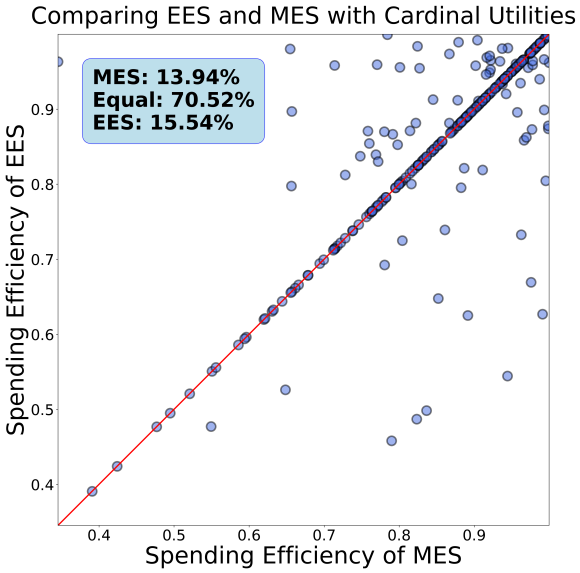


Figure 7: Spending efficiency of EES with ADD-OPT-SKIP vs. MES with ADD-ONE: approval utilities.

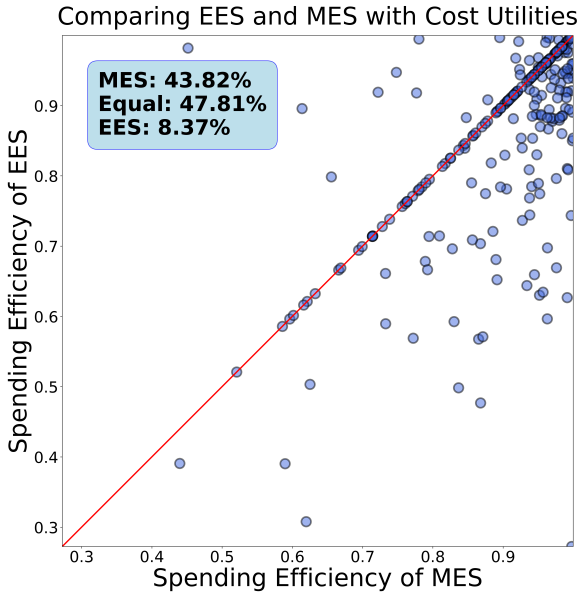


Figure 8: Spending efficiency of EES with ADD-OPT-SKIP vs. MES with ADD-ONE: cost utilities.

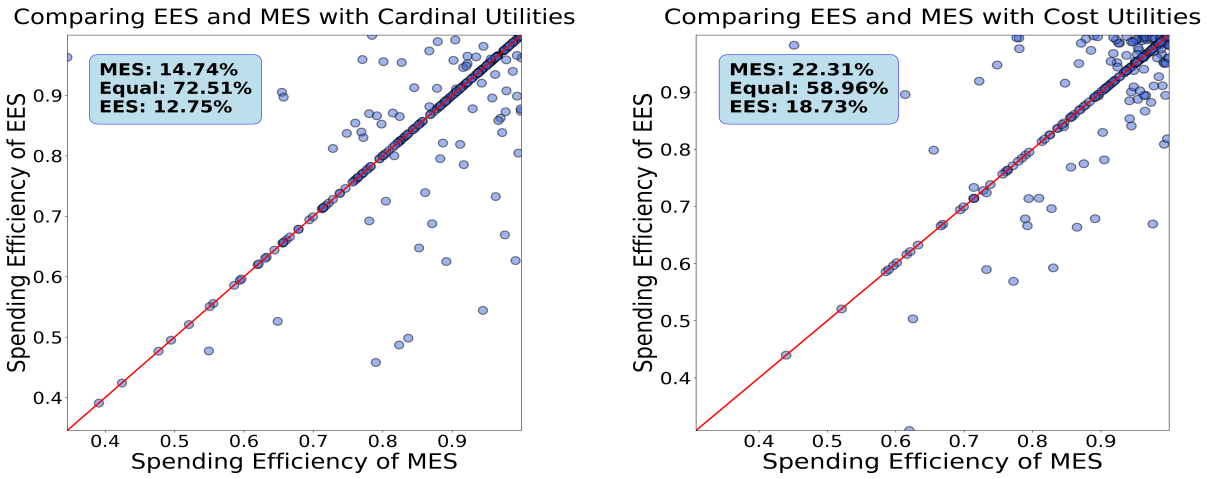
budget that causes overspending. We identify 14 such instances for approval utilities and 3 for cost utilities. Heuristics that terminate as soon as overspending occurs perform poorly on such instances. ADD-OPT-SKIP, on the other hand, is able to explore the space of virtual budgets in a more comprehensive fashion, avoiding these worst-case scenarios, and demonstrates on average 10% higher spending efficiency in these cases (see Figure 6).

For ADD-OPT, the observed benefits are less pronounced. While it reduces the number of calls to EES compared to ADD-ONE, one needs to execute Algorithm 5 (which has a runtime comparable to that of EES) for every EES run, leading to minimal computational savings.

Recommendations Our experimental results suggest that EES+ADD-OPT-SKIP achieves comparable spending efficiency to MES+ADD-ONE while (1) using orders of magnitude fewer calls to EES and Algorithm 5, and (2) avoiding worst-case scenarios, such as the one illustrated in Figure 1. These advantages make EES+ADD-OPT-SKIP particularly suitable for real-world use in cities, as well as in computational experiments on synthetic data, where many repetitions are necessary for statistical significance. Alternatively, one can explore a hybrid approach, which runs both MES+ADD-ONE and EES+ADD-OPT-SKIP, as it incurs a negligible computational overhead relative to MES+ADD-ONE (see Figure 5b).

Case Study: Stare Implementation Figure 10 examines a specific implementation from Stare, Poland, demonstrating how project selection changes with budget allocation.

Sonja Kraiczy
University of Oxford
Oxford, United Kingdom
Email: sonja.kraiczy@cs.ox.ac.uk



(a) Approval utilities.

(b) Cost utilities.

Figure 9: Spending efficiency of MES and EES with ADD-ONE heuristic. Each point in the scatter point represents a `Pabulib` data set.

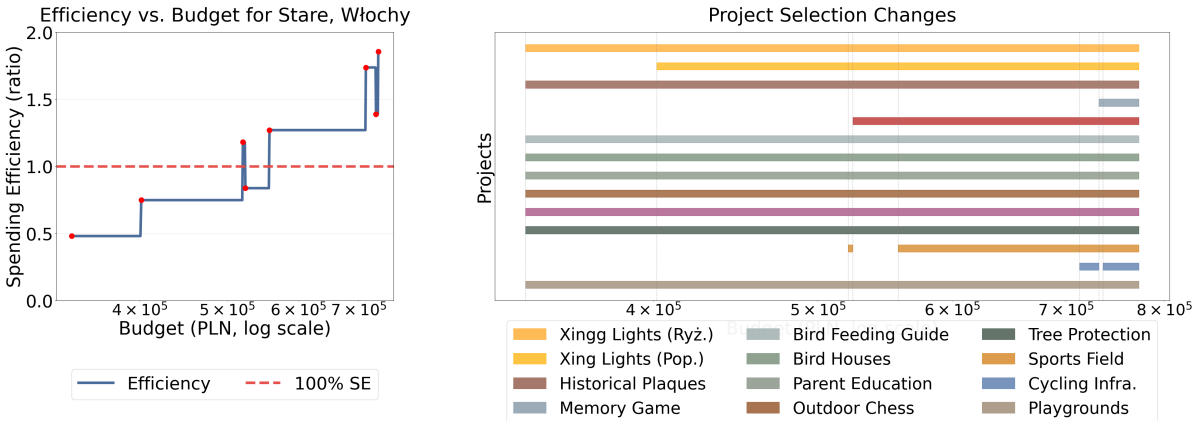


Figure 10: Graph showing which projects are implemented for a given budget for an instance from Stare, Poland using ADD-ONE until all projects are selected for the first time. The left shows the spending efficiency for the given virtual budget, the right shows the specific projects implemented for that budget.

Isaac Robinson
University of Oxford
Oxford, United Kingdom
Email: isaac.robinson@cs.ox.ac.uk

Edith Elkind
Northwestern University
Evanston, USA
Email: edith.elkind@northwestern.edu