Now We're Talking: Better Deliberation Groups through Submodular Optimization

Jake Barrett[1], Kobi Gal[1,2], Paul Gölz[3], Rose M. Hong[3], and Ariel D. Procaccia[3]

[1]University of Edinburgh,
[2]Ben-Gurion University of the Negev,
[3]Harvard University
j.t.l.barrett@sms.ed.ac.uk, kgal@exseed.ed.ac.uk, goelz@seas.harvard.edu, rosehong@college.harvard.edu, arielpro@seas.harvard.edu

Abstract

Citizens' assemblies are groups of randomly selected constituents who are tasked with providing recommendations on policy questions. Assembly members form their recommendations through a sequence of discussions in small groups (*deliberation*), in which group members exchange arguments and experiences. We seek to support this process through optimization, by studying how to assign participants to discussion groups over multiple sessions, in a way that maximizes interaction between participants and satisfies diversity constraints within each group. Since repeated meetings between a given pair of participants have diminishing marginal returns, we capture interaction through a submodular function, which is approximately optimized by a greedy algorithm making calls to an ILP solver. This framework supports different submodular objective functions, and we identify sensible options, but we also show it is not necessary to commit to a particular choice: Our main theoretical result is a (practically efficient) algorithm that simultaneously approximates every possible objective function of the form we are interested in. Experiments with data from real citizens' assemblies demonstrate that our approach substantially outperforms the heuristic algorithm currently used by practitioners.

1 Introduction

Can deliberation among groups of randomly selected people revitalize democracy? A growing number of political theorists, activists, and even politicians believe so (Fishkin 2018; Landemore 2020; Van Reybrouck 2016) and have been putting this idea into practice. In the last decade, hundreds of *citizens' assemblies* (also known as *deliberative polls* or *minipublics*) have been convened by civil society and by local and national governments (OECD 2020). Recently, these assemblies have become more numerous and higher profile: Citizens' assemblies established in 2016 and 2019 in Ireland, for example, have led to national referenda and, in turn, to major constitutional changes. As another example, France has embraced this paradigm at different levels of government (Landemore 2020); in particular, the recommendations of the Citizens Convention for Climate, established in 2019, have given rise to significant new legislation.

On a high level, the process of organizing a citizens' assembly consists of two phases. In the first phase, a pool of volunteers is put together, and the assembly is randomly selected from this pool. The assembly is required to be representative of the population in terms of features like gender, ethnicity, age, and education, but the pool of volunteers is typically unrepresentative of the population due to self-selection bias. In a series of papers, Flanigan et al. (2021a, 2020); Flanigan, Kehne, and Procaccia (2021) develop an algorithmic framework for randomly selecting citizens' assemblies in a way that is representative, fair to volunteers, and transparent.

In the second phase of the process, the assembly discusses the issues at hand and reaches conclusions that inform policy making. This discussion, known as *deliberation*, is what enables an assembly composed of laypeople to reach judicious recommendations on a complex issue. Though deliberation lies at the heart of a citizens' assembly's purpose, almost no work so far supports deliberation through algorithms, computation, or AI. Deliberation in a citizens' assembly takes place over a number of sessions, where in each session, participants are divided into discussion groups, which we refer to as *tables*. For example, the Citizens' Assembly of Scotland, which was convened by the Scottish Government in 2019–2020, ran over 16 sessions spread across 8 weekends; in each session, the 104 participants were divided across 12 tables.

This work arises out of a collaboration with the *Sortition Foundation* — a nonprofit organization which facilitates dozens of citizens' assemblies worldwide every year — on the design and implementation of algorithms for managing deliberation. One problem that our contacts brought up is scheduling the assignment of participants to tables, which we address in this paper. The practitioners' primary goal is to find a schedule that, over the course of the process, allows participants to exchange ideas with as many other participants as possible. In addition, tables must be demographically diverse; in the Citizens' Assembly of Scotland, they were diversified based on political view, age, and gender.

Currently, the Sortition Foundation as well as other nonprofits use a heuristic algorithm called GROUPSELECT (Verpoort 2020) to allocate tables, developed by the Sortition Foundation. Internally, this algorithm optimizes the objective of maximizing the number of pairs of participants who meet at least once, assigning no value to subsequent meetings. We see two shortcoming with this current approach: First, as we show in Section 6, GROUPSELECT performs quite poorly in terms of its chosen objective. Second, the objective itself

often fails to encourage good schedules. We elaborate on this problem in Section 3, but an example of such a problematic situation is when all participants have met each other. At that point, the objective is indifferent between all possible assignments, and thus even a schedule repeating the same table assignment across all remaining sessions would be optimal.

To overcome these shortcomings, we must address several challenges. On a conceptual level, we need a principled measure of interaction between participants, which we seek to maximize. If interaction is measured as a function of the number of times each pair of participants meets, how much value should the first meeting between Alice and Bob have relative to the second, third, or fourth? On a technical level, we aim to develop a theoretically sound and practical algorithmic framework for optimizing our measure of interaction, with an eye towards real-world deployment.

Our approach and results. It is intuitive that meetings between the same participants have diminishing marginal returns, e.g., the third meeting carries less value for deliberation than the second. We express this idea through what we call a *saturation function* f: for a monotone nondecreasing and concave function $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$, we model the goal of maximizing interaction between participants through the submodular objective $\hat{f} = \sum_{\{i,j\}} f(m_{i,j})$, where the sum ranges over all pairs of agents i, j and $m_{i,j}$ is the number of sessions in which i and j are assigned to the same table. For any choice of saturation function f, we obtain a practical algorithm that maximizes the corresponding objective $\hat{f}$ within an approximation factor of $1 - 1/e \approx 63\%$, building on a classical result in submodular maximization (Nemhauser, Wolsey, and Fisher 1978) and using an integer linear programming (ILP) solver as a subroutine.

Which saturation function f should we use? Given that no objective seems universally better than the others, we pursue an approach of simultaneous approximation (Stein and Wein 1997). Specifically, we design an algorithm that produces schedules that $\Omega(1/\log T)$-approximate the objectives $\hat{f}$ for *all* saturation functions f at once, where T is the number of sessions. Beyond table allocation, this result applies to maximum-coverage style problems in all other domains. Specifically, we show that the f-MAXCOVERAGE problem of Barman, Fawzi, and Fermé (2021) can be $\Omega(1/\log T)$-approximated for all f at once, in polynomial time.

We then evaluate our optimization algorithms and GROUPSELECT on data from seven citizens' assemblies. We find that all our algorithms outperform GROUPSELECT by a wide margin, including when measured by its own objective. Two saturation functions, based on the harmonic and geometric series, seem promising for optimizing schedules in practice.

Related work. Along with those already mentioned, several works (Benadè, Gölz, and Procaccia 2019; Do et al. 2021; Meir, Sandomirskiy, and Tennenholtz 2021; Saran and Tumennasan 2013; Walsh and Xia 2012) in computational social choice have studied the random selection of citizens' assemblies, but none of them interface with the deliberation taking place once the assembly convenes. A second line of work (Chung and Duggan 2020; Fain et al. 2017; Goel and Lee 2016; Perote-Peña and Piggins 2015; Zvi, Leizerovich, and Talmon 2021) proposes and analyzes mathematical models for deliberation, which we see as complementary to our approach. Whereas these papers capture the dynamics of deliberation with more nuance than us, our paper approaches deliberation through the lens of a practical problem, table allocation, and its interaction with deliberation. Finally, Fishkin et al. (2019) develop a system that automatically manages speaking times and speaker order in online deliberation. This is the one example we know of that seeks to support deliberation in citizens' assemblies through a practical, computational approach, but it addresses a fundamentally different aspect of the deliberation process.

Our use of submodular objectives follows a long tradition in AI of maximizing submodular functions to obtain diverse solutions. For example, this methodology encourages different parts of a multi-document summary to refer to different sources (Carbonell and Goldstein 1998; Lin and Bilmes 2011), sensors to be placed where they can collect complementary information (Krause, Singh, and Guestrin 2008), or papers to be assigned to reviewers with different expertise (Ahmed, Dickerson, and Fuge 2017). In our application, the submodular objective encourages schedules to vary which pairs of assembly members meet across the sessions.

Our class of objective functions naturally generalizes the *maximum coverage* problem (Hochbaum and Pathria 1998) and coincides with the class of f-MAXCOVERAGE problems (for saturation functions f) defined by Barman, Fawzi, and Fermé (2021). Despite this connection, the result of Barman et al. (an algorithm that, for certain f, obtains a better approximation ratio than $1 - 1/e$) does not apply to table allocation: since the "sets" that can be chosen for coverage are implicitly represented in table allocation, their algorithm is neither polynomial-time nor practical to run in our setting.

Finally, our setup resembles the classic *social golfer* problem: *n golfers must be repeatedly partitioned into k groups, each of equal size s. Find a schedule of maximum length given that no two golfers may be placed in the same group twice.* Most work in this space analyzes the solutions for specific n and k, or optimizes Boolean satisfiability formulations to find long schedules (e.g., Lardeux et al. 2015; Schmand, Schröder, and Vargas Koch 2022; Triska and Musliu 2012). Our problem differs from the social golfer problem in two ways. First, the social golfer problem maximizes the number of sessions subject to a hard constraint on repeated meetings, whereas we minimize repeated meetings subject to a fixed schedule length. Second, whereas the social golfer problem allows to group any s golfers together, our representativeness constraints make the problem no longer symmetric and even less tractable than the social golfer problem.

2 Model

Table allocation problem. An instance of the table allocation problem is a tuple consisting of a set of *agents* $N = [n]$, a number of *tables* k, a number of *sessions* $T \geq 2$, and a set of *representativeness constraints*. These representativeness constraints are given as a set F of *features*, where each feature $\varphi \in F$ is defined by a set of agents $A_\varphi \subseteq N$ possessing this

feature, a lower quota ℓ_φ, and an upper quota u_φ such that $0 \leq \ell_\varphi \leq u_\varphi \leq \lceil n/k \rceil$.

A *partition* for this instance partitions the agents into k disjoint tables $N = \Delta_1 \dot\cup \cdots \dot\cup \Delta_k$, subject to two constraints: (1) each table Δ_i has size either $\lfloor n/k \rfloor$ or $\lceil n/k \rceil$, and (2) each table Δ_i satisfies all representativeness constraints, in the sense that $\ell_\varphi \leq |\Delta_i \cap A_\varphi| \leq u_\varphi$ for all features φ.[1] For ease of exposition, we assume that any given instance allows for at least one partition. Given a table allocation instance, our aim is to construct a *schedule*, which is a multiset Z over partitions containing T elements.[2]

The f-MAXCOVERAGE problem. The optimization objectives we propose for table allocation resemble maximum coverage, since we aim to cover pairs of agents by selecting T many partitions, each of which brings together ("covers") certain pairs of agents. Our objectives generalize maximum coverage in that they may reward not only the first meeting but also subsequent meetings to various degrees.

This generalization of maximum coverage coincides with the f-MAXCOVERAGE problem defined by Barman, Fawzi, and Fermé (2021), which is is parameterized by a *saturation function* $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ that is monotone nondecreasing, concave, and satisfies $f(0) = 0$. Each f-MAXCOVERAGE instance consists of a finite *ground set* G, a collection $\mathcal{Z}$ of sets $S \subseteq G$ of ground elements, and *target number* $T \geq 2$ of sets. For a given instance of f-MAXCOVERAGE, a *selection* is a multiset over $\mathcal{Z}$, and a *solution* is a selection of cardinality T. The goal of f-MAXCOVERAGE is to find a solution Z that maximizes the *objective* $\hat{f}$, which is the function mapping selections Z to $\mathbb{R}_{\geq 0}$ defined in terms of f such that

$$\begin{aligned}\hat{f}(Z) &\coloneqq \sum\nolimits_{g \in G} f(\text{number of sets in } Z \text{ that contain } g) \\ &= \sum\nolimits_{g \in G} f\left(\textstyle\sum_{S \in \mathcal{Z} : g \in S} Z(S)\right).\end{aligned}$$

For the saturation function $f^1(x) \coloneqq \mathbb{1}\{x \geq 1\} = \min\{x, 1\}$, f^1-MAXCOVERAGE coincides with classic maximum coverage. The saturation function f adds expressivity beyond maximizing coverage; e.g., the objective assigns value to second appearances of g if $f(2) > f(1)$. Generally, the concavity of f promotes schedules that contain ground elements similar numbers of times.

A function s that maps selections to $\mathbb{R}_{\geq 0}$ satisfies *diminishing returns* if, for any two selections $Z_1 \sqsubseteq Z_2$ and for any $S \in \mathcal{Z}$, $s(Z_1 + \{S\}) - s(Z_1) \geq s(Z_2 + \{S\}) - s(Z_2)$. We call s *monotone* if, for all selections $Z_1 \sqsubseteq Z_2$, $s(Z_1) \leq s(Z_2)$. One easily verifies that all objectives $\hat{f}$ have diminishing returns and are monotone. Finally, for some $\alpha \in (0, 1)$, a solution Z α-approximates an objective $\hat{f}$ if $\hat{f}(Z) \geq \alpha \cdot \max_{\text{solution } Z'} \hat{f}(Z')$. A solution Z is a *simultaneous α-approximation* if it α-approximates the objectives $\hat{f}$ for all saturation functions f at once.

[1] In some assemblies, partitions must additionally satisfy *clustering constraints*, requiring that some participants (e.g. those unwilling to be photographed or in need of translation services) be grouped together. Since our approach in Section 4 generalizes to clustering in a straight-forward way, we omit it for ease of exposition.

[2] A multiset over a finite support X is a function $ms : X \to \mathbb{N}$, where $ms(x)$ indicates how many copies of $x \in X$ are contained in the multiset. We write $|ms| = \sum_{x \in X} ms(x)$ for the cardinality of a multiset and denote multiset addition by $+$, multiset difference by $-$, and multiset inclusion by $\sqsubseteq$.

3 Table Allocation as f-MAXCOVERAGE

Looking at the table allocation problem, it is not obvious what makes one schedule more conducive to deliberation than another, other than a vague intuition that discussion groups should be "mixed up" between sessions. The Sortition Foundation's work on GROUPSELECT makes an important contribution by declaring a mathematically precise objective: maximizing how many pairs of assembly members meet at least once. This objective is certainly an incomplete perspective on what makes a schedule conducive to deliberation, but it is rooted in the Sortition Foundation's extensive experience in organizing citizens' assemblies.

We can express the objective optimized by GROUPSELECT by casting a given table allocation instance as a f^1-MAXCOVERAGE problem: Let the ground set G be the set $\binom{N}{2}$ of all unordered pairs of agents, and let the collection $\mathcal{Z}$ contain, for each partition $S = \Delta_1 \dot\cup \cdots \dot\cup \Delta_k$ of the table allocation instance, the set $\bigcup_{1 \leq i \leq k} \binom{\Delta_i}{2}$ of all pairs sitting at the same table in S. GROUPSELECT's objective reduces to f^1-MAXCOVERAGE since, for the saturation function $f^1(x) = \min\{x, 1\}$, each pair i, j that does not meet contributes 0 to the objective $\hat{f}^1$ and all other pairs contribute 1. Throughout this paper, we will use the same reduction to optimize the objectives $\hat{f}$ for other saturation functions f over schedules.

As mentioned in the introduction, GROUPSELECT's objective $\hat{f}^1$ seems inappropriate in many cases. One obvious concern is that $\hat{f}^1$ does not express any preference between schedules in which all pairs meet, which, our empirical evaluation shows, is not just a hypothetical, but relevant on real data. Since *some* repeated meetings are typically unavoidable, an objective should arguably express a preference over how these repeated meetings are distributed over pairs. Even for just two sessions and without representativeness constraints, repeated meetings are inevitable whenever $n > k^2$. In the full version of this paper, we show that the minimum number of repeated meetings is at least $k^2\binom{x}{2} + x \cdot y$ in this case, where $x \coloneqq \lfloor n/k^2 \rfloor$ and $y \coloneqq n \bmod k^2$.

A more subtle issue with optimizing $\hat{f}^1$ is that prioritizing first meetings might not be worth an arbitrarily high cost in terms of which other pairs meet. In the full version, we present a table allocation problem in which it is difficult to arrange meetings for a subset P of the pairs, in the sense that (1) at most one pair in P can meet per (representative) partition, (2) whenever a pair in P meets, representativeness implies that the other pairs meeting each other are essentially always the same ones, and (3) if no pair in P meets, there is a lot of freedom in who meets whom. In this instance, an algorithm optimizing $\hat{f}^1$ will expend most sessions to make pairs in P meet one by one, and will make the overwhelming majority of pairs meet either very often or just once. In such instances, it seems preferable to forgo some first meetings in

P to allow a large number of pairs to meet a second time.

Motivated by the above limitations of $\hat{f}^1$, we generalize the Sortition Foundation's optimization problem to saturation functions other than f^1. Each saturation function has its distinct advantages and disadvantages, which might matter to different degrees depending on the instance. For example, for any $r \geq 2$, consider the saturation function $f^r(x) \coloneqq \min\{x, r\}$, for which each pair's contribution to $\hat{f}^r$ increases by 1 per meeting up to the r'th meeting, and does not increase beyond that. On the upside, the objective $\hat{f}^r$ pushes the schedule towards an ideal point in which each pair meets r times with maximum vigor. On the downside, if the representativeness constraints force some pairs to meet fewer than r times (or more than r times), $\hat{f}^r$ is indifferent between how equally the number of meetings below r (or above r, respectively) are spread.

In search of saturation functions whose marginal returns diminish more smoothly, two kinds of saturation functions strike us as promising. The first are the *geometric* saturation functions g^β (for some $0 < \beta < 1$), where $g^\beta(x) \coloneqq \sum_{i=1}^{x} \beta^i$. Given that the marginals $\beta^{m_{i,j}}$ decay exponentially in the number $m_{i,j}$ of previous meetings of the pair, the geometric objectives $\hat{g}^\beta$ should still put much weight on the first meeting. Geometric objectives possess the intuitively appealing "self-similarity" property that, if we fix a partial schedule in which all pairs appear equally often, the problem of optimizing the remaining partitions looks just like optimizing a shorter schedule, with the objective multiplied by a constant. A final example is the *harmonic* saturation function $h(x) \coloneqq \sum_{i=1}^{x} 1/i$. Since this function's marginals decrease more slowly, we would expect the objective $\hat{h}$ to prioritize earlier meetings less radically. Note that the "self-similarity" property is not satisfied by this objective (see full version).

4 Optimizing a Specific Saturation Function

Having built intuition about the preference over allocation tradeoffs expressed by a saturation function, we investigate how a given objective $\hat{f}$ can be approximately optimized.

One immediate obstacle is that already the problem of deciding whether any partition exists for the given representativeness constraints is NP-hard (see full version). Thus, polynomial-time algorithms cannot produce partitions or schedules (unless P = NP), which is why we will search for algorithms that (though not theoretically polynomial-time) run sufficiently fast on practical inputs. Fortunately, state-of-the-art solvers for Integer Linear Programming (ILP) can reliably find a representative partition in little time. Though ILP solvers are powerful, formulating the entire optimization over schedules as an ILP is intractable as we show in Section 6. Therefore, our algorithmic approach will use ILP as a powerful subroutine for finding partitions, but our approach will handle in outside logic how the contributions of different partitions interact in the objective.

What will enable us to break down the optimization into generating partitions one at a time are the properties of the objectives $\hat{f}$ we consider, namely diminishing returns, monotonicity, and that $\hat{f}(\emptyset) = 0$. These properties are useful since, for any multiset function over $\mathcal{Z}$ satisfying them, Nemhauser, Wolsey, and Fisher (1978) showed that a simple greedy algorithm returns a multiset of cardinality T whose objective value is at least a $1 - 1/e$ fraction of the optimal objective value across all multisets of size T.[3] This greedy algorithm iteratively constructs a multiset Z by starting from the empty multiset and T times adding the set $S \in \mathcal{Z}$ with largest marginal increase $\hat{f}(Z + \{S\}) - \hat{f}(Z)$. In most cases where this greedy algorithm is run, the collection of sets $\mathcal{Z}$ is not too large and explicitly given, which allows to identify S by enumerating $\mathcal{Z}$. By contrast, the set of all partitions might be exponentially large, so enumerating them is not an option.

Thus, we instead implement each step of the greedy algorithm by solving an ILP that will yield the partition with largest marginal increase. This ILP formulation makes use of the specific shape of our objectives, which decompose into a sum over pairs of agents, and which have the property that any partition's marginal contribution to a pair $\{i, j\}$'s summand is either zero (if i and j do not meet) or a constant value $f(m_{i,j} + 1) - f(m_{i,j})$ (if i and j meet), where $m_{i,j}$ denotes the number of times i and j have met before. Below we describe the ILP, whose variables are $x_{i,\tau}$ ("agent i is allocated to table τ") and $y_{\{i,j\},\tau}$ ("agents i and j are both allocated to table τ"), for all $i \neq j \in N$ and $1 \leq \tau \leq k$:

$$\text{maximize} \sum_{\substack{\{i,j\} \in \binom{N}{2} \\ 1 \leq \tau \leq k}} \big(f(m_{i,j}+1) - f(m_{i,j})\big) \cdot y_{\{i,j\},\tau}$$

$$\text{subject to} \sum_{1 \leq \tau \leq k} x_{i,\tau} = 1 \qquad \forall i \in N$$

$$\lfloor n/k \rfloor \leq \sum_{i \in N} x_{i,\tau} \leq \lceil n/k \rceil \qquad \forall 1 \leq \tau \leq k$$

$$\ell_\varphi \leq \sum_{i \in A_\varphi} x_{i,\tau} \leq u_\varphi \qquad \forall 1 \leq \tau \leq k, \varphi \in F$$

$$\left.\begin{array}{l} y_{\{i,j\},\tau} \geq x_{i,\tau} + x_{j,\tau} - 1 \\ y_{\{i,j\},\tau} \leq x_{i,\tau} \\ y_{\{i,j\},\tau} \leq x_{j,\tau} \end{array}\right\} \quad \forall \{i,j\} \in \tbinom{N}{2},\ 1 \leq \tau \leq k$$

$$x_{i,\tau} \in \{0,1\}, y_{\{i,j\},\tau} \in \{0,1\} \qquad \forall i, j, \tau.$$

Observe that, for each pair $\{i, j\}$, the 4th, 5th, and 6th constraints constrain $y_{\{i,j\},\tau}$ to equal one iff $x_{i,\tau}$ and $x_{j,\tau}$ are one. As a result, at most one variable $y_{\{i,j\},\tau}$ can be nonzero for each $\{i, j\}$, and thus each pair contributes either $f(m_{i,j} + 1) - f(m_{i,j})$ or nothing to the objective, as intended. Due to the quadratically many $y_{\{i,j\},\tau}$ variables and the constraints tying them to the $x_{i,\tau}$, this ILP is substantially more difficult to solve than just finding a valid partition, but we will show in Section 6 that a state-of-the-art ILP solver can optimize these programs to sufficient accuracy.

We can run the greedy maximization algorithm by iterating the following steps T times: solving the ILP, extracting the

[3] Technically, Nemhauser, Wolsey, and Fisher (1978) prove this for submodular *set* functions. In our setting, sets may be selected multiple times, but the claimed result for multiset functions follows directly by duplicating all sets T times. Whereas diminishing returns and submodularity are equivalent for set functions, they differ for multiset functions (Kapralov, Post, and Vondrák 2013).

new partition from the $x_{i,\tau}$, adding the new partition to Z, and updating the $m_{i,j}$. If the ILP solver optimizes all subproblems to optimality, the resulting schedule will $(1-1/e)$-approximate the objective $\hat{f}$ as proved by Nemhauser, Wolsey, and Fisher (1978), and the greedy algorithm is known to outperform this approximation factor in many cases (Pokutta, Singh, and Torrico 2020). Even if we should be forced to terminate some ILP calls before reaching optimality, our guarantees degrade smoothly: If all ILPs return a partition whose marginal increase is at least an $\alpha > 0$ fraction of the optimal marginal increase, the resulting schedule is still at least a $(1-1/e^{\alpha})$-approximation (Goundan and Schulz 2007).

5 Simultaneously Optimizing All Saturation Functions

Even though we have found a way to optimize the objective for any given saturation function f, such an approach remains not entirely satisfying given that we chose the saturation function somewhat arbitrarily. As we discussed in Section 3, how much the saturation function should encourage pairs to meet for the i'th time across the different i seems to depend on which distribution of meeting numbers are possible, which is hard to predict for a given instance.

This challenge of settling on a single saturation function raises the question of whether it is possible to produce schedules that perform well relative to the objectives belonging to *all* saturation functions simultaneously. Since we have seen in Section 3 that different objective can lead to starkly different schedules, and since there is an infinite variety of saturation functions, it would be natural if simultaneous α-approximations would in general only exist for extremely low α. Instead, Algorithm 1 below (SIMAPPROX) provides a simultaneous $\Omega(1/\log T)$-approximation to all objectives.

Algorithm 1: SIMAPPROX

1 $Z \leftarrow \emptyset$
2 **for** $t = 0, 1, \ldots, T-1$ **do**
3 $\quad p \leftarrow \lfloor (t/T) \cdot (1 + \log_2 T) \rfloor$
4 $\quad Z \leftarrow Z + \left\{ \operatorname{argmax}_{S \in \mathcal{Z}} \hat{f}^{2^p}\left(Z + \{S\}\right) \right\}$
5 **return** Z

This algorithm and our analysis of simultaneous approximation apply not only to table allocation but to all f-MAXCOVERAGE problems, which have many further applications. For these problems, SIMAPPROX even runs in polynomial time, since the description of an f-MAXCOVERAGE instance includes $\mathcal{Z}$. For table allocation instances, the implicitly defined $\mathcal{Z}$ might be exponentially large, but the ILP from Section 4 implements Line 4 in practically efficient running time.

The structure of SIMAPPROX closely resembles that of greedy maximization in that (using the terminology of table allocation) it constructs a schedule Z, partition by partition, greedily adds partitions whose marginal increase relative to some objective $\hat{f}$ is largest, and uses the same ILP formulation to identify these partitions. The big difference between both algorithms is that SIMAPPROX does not optimize marginals of the same objective in each iteration. Instead, it first optimizes marginals for $\hat{f}^{2^0}$ for some number of steps, then marginals for $\hat{f}^{2^1}$, then for $\hat{f}^{2^2}$, through the powers of two up to around $\hat{f}^T$, each for a roughly equal number of steps. In particular, SIMAPPROX is computationally no more complex than the greedy maximization algorithm.

The key insight of this algorithm is that α-approximating the logarithmically many objectives of the form $\hat{f}^{2^p}$ (for some p) suffices to approximate *all* objectives $\hat{f}$ within a constant factor of α. Thus, our proof that SIMAPPROX is a simultaneous $\Omega(1/\log T)$-approximation proceeds in three steps: First, we show that the schedule returned by the algorithm $\Omega(1/\log T)$-approximates all $\hat{f}^r$ where r is a power of two (Lemma 5.1). Second, we show that the solution approximates the objectives $\hat{f}^r$ for all r (Lemma 5.2). Finally, we prove that this implies simultaneous approximation for all objectives $\hat{f}$ (Theorem 5.3). We sketch these arguments below and defer the formal proofs to the full version.

Lemma 5.1. *For each $0 \leq p \leq \log_2 T$, the solution Z returned by* SIMAPPROX *approximates $\hat{f}^{2^p}$ within a factor of $(1-1/e) \cdot (\frac{1}{1+\log_2 T} - \frac{1}{T})$.*

Proof sketch. Since $\hat{f}^{2^p}$ is greedily optimized in roughly $T/\log T$ of the steps, the objective value is at least a $(1-1/e)$ fraction of the optimal objective value obtained by any schedule of length $T/\log T$, and this holds despite the steps optimizing other objectives coming before and after. Since $\hat{f}^{2^p}$ has diminishing returns, the optimal objective value for a schedule of length $T/\log T$ is at least a $1/\log T$ fraction of the optimal objective value for a schedule of length T. □

Lemma 5.2. *For each $1 \leq r \leq T$, the solution Z returned by* SIMAPPROX *approximates $\hat{f}^r$ within a factor of $\frac{1-1/e}{2} \cdot (\frac{1}{1+\log_2 T} - \frac{1}{T})$.*

Proof sketch. For two values $r_1 \approx r_2$, the objectives $\hat{f}^{r_1}$ and $\hat{f}^{r_2}$ are similar to the point that, if $r_1 \leq r_2$, any schedule that α-approximates $\hat{f}^{r_1}$ at least $\alpha \cdot \frac{r_1}{r_2}$-approximates $\hat{f}^{r_2}$. For a given r, let 2^p denote its next-lower power of two. By Lemma 5.1, Z $\Omega(1/\log T)$-approximates $\hat{f}^{2^p}$; hence, Z must $\frac{2^p}{r} \cdot \Omega(1/\log T) \geq \frac{1}{2} \cdot \Omega(1/\log T)$-approximate $\hat{f}^r$. □

Theorem 5.3. SIMAPPROX *produces solutions that simultaneously α-approximate f-*MAXCOVERAGE *for all f, in polynomial time, for $\alpha = \frac{1-1/e}{2} \cdot (\frac{1}{1+\log_2 T} - \frac{1}{T}) \in \Omega(1/\log T)$.*

Proof sketch. As we show in the full version of the paper, the f^r form a sort of "basis" of the space of saturation functions in the sense that, for any saturation function f and any T, there exist nonnegative weights $\{w_i\}_{1 \leq i \leq T}$ such that $f(x) = \sum_{i=1}^{T} w_i \cdot f^i(x)$ for all $0 \leq x \leq T$. Note that it must then also hold that $\hat{f} = \sum_{i=1}^{T} w_i \cdot \hat{f}^i$. For any saturation function f, by Lemma 5.2, it holds that

$$\hat{f}(Z) = \sum\nolimits_{i=1}^{T} w_i \cdot \hat{f}^i(Z) \geq \sum\nolimits_{i=1}^{T} w_i \cdot \alpha \cdot \max_{\text{solution } Z'} \hat{f}^i(Z')$$

$$= \alpha \cdot \sum_{i=1}^{T} \max_{\text{solution } Z'} w_i \cdot \hat{f}^i(Z')$$

$$\geq \alpha \cdot \max_{\text{solution } Z'} \sum_{i=1}^{T} w_i \cdot \hat{f}^i(Z') = \alpha \cdot \max_{\text{solution } Z'} \hat{f}(Z'). \square$$

In the full version, we show that SIMAPPROX's simultaneous approximation ratio of $\Omega(1/\log T)$ for f-MAXCOVERAGE is optimal up to a log log factor:

Theorem 5.4. *There exists a family of maximum coverage instances such that no solution has a simultaneous approximation ratio larger than $\mathcal{O}(\log\log T/\log T)$. This holds even if all sets $S \in \mathcal{Z}$ have equal cardinality (like in the table allocation problem when k divides n).*

In these instances, the ground elements are partitioned into multiple blocks, and each block represents a different tradeoff between (a) how many ground elements of the block are included in a set in $\mathcal{Z}$ and (b) how many ground elements are in the block overall. For large values of r, $\hat{f}^r$ is maximized by choosing sets from blocks scoring high on (a) because they cover many ground elements per set. For small r, by contrast, blocks scoring high on (b) allow to avoid selecting ground elements more than r times, which would not help $\hat{f}^r$. Since scoring high on different objectives $\hat{f}^r$ requires selecting disjoint sets, no solution can simultaneously approximate them within a high factor. We conjecture that Theorem 5.4's impossibility on simultaneous approximation extends to the table allocation problem; however, the symmetry between tables and the transitivity of which pairs can simultaneously meet make analogous instances highly cumbersome to construct.

6 Implementation and Empirical Results

We have implemented all algorithms in this work in Python, using Gurobi as our ILP solver. Our implementation is open source at https://github.com/rosemhong/tables. Currently, we are working with the Sortition Foundation to incorporate our algorithms into the tool that hosts GROUPSELECT (Verpoort 2020), which will allow users to switch to our improved algorithms with little effort.

We perform our experiments on seven datasets, each based on data from a real citizens' assembly. Two of these datasets, `sf_e` and `sf_f`, exactly describe assemblies coorganized by the Sortition Foundation. The other five datasets, `sf_a` through `sf_d` and `hd` are derived from assembly-selection data used by Flanigan et al. (2021a). For these latter datasets, we do not have access to the members who ended up being drawn for the citizens' assembly, but we can "re-run" the lottery process using the selection software Panelot (Flanigan et al. 2021b) to obtain an assembly that satisfies the actual representativeness constraints. In the full version, we describe the processing of the datasets and the experimental setup in more detail. To compute experiments in parallel, we run them on an AWS EC2 C5 instance with a 3.6 GHz processor, 16 threads, and 32 GB of RAM. Given that we limit each experiment to a single thread, individual running times of our algorithms are comparable to consumer hardware.

GROUPSELECT cannot satisfy a given list of representativeness constraints but only makes a best effort at proportionally representing the diversified features. To compare it to our algorithms on equal terms, we run our algorithms with representativeness constraints derived from GROUPSELECT's output on the given instance. (In the full version, we show that our algorithms continue to perform well for particularly tight representativeness constraints.)

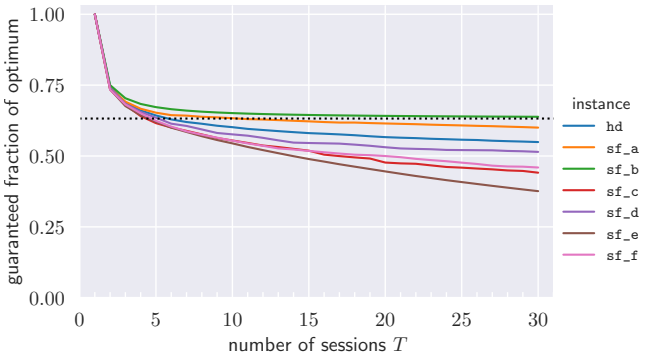

Figure 1: Approximation certificates for the greedy algorithm on $\hat{g}^{1/2}$, guaranteeing near-optimality. The dashed line marks $1 - 1/e$.

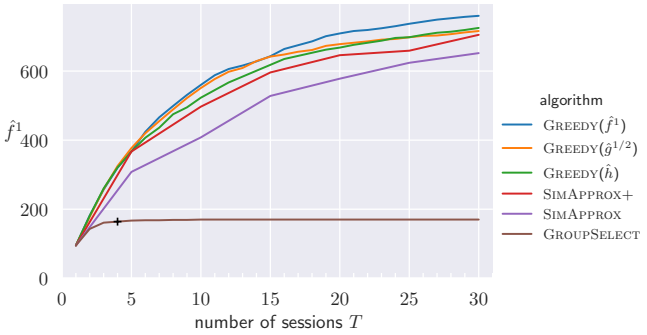

Figure 2: Performance of different algorithms on instance `sf_f` as measured by $\hat{f}^1$. The cross marks the historically chosen schedule.

6.1 How Well Does the Greedy Algorithm Optimize Its Objective?

We begin by verifying that optimizing schedules in a one-shot ILP is not tractable, which justifies our greedy approach. Indeed, when optimizing $\hat{g}^{1/2}$ for a mere 4 sessions, in 4 hours running time, the ILP solver did not find *any* feasible schedules from the one-shot ILP in 4 out of the 7 instances.[4] We conclude that the runtime of this one-shot approach scales prohibitively in the number of sessions to be useful.

Can the ILP solver solve the ILPs from Section 4, and how much time does the solver need for the greedy algorithm to optimize its objective well? In the full version of the paper, we show that increasing the ILP solver's timeout generally increases objective value attained by the greedy algorithm, but that these increases level off after around 60 seconds. To accommodate one outlier and to be safe, we set the optimiza-

[4] For the other 3 instances, the objective value is no better (within $\pm 1\%$) than the greedy algorithm's result produced in 8 minutes.

tion timeout to 120 seconds from here on, for the ILP calls both in the greedy algorithm and in SIMAPPROX.[5]

Ideally, we want to know how close to optimal the schedules produced by the greedy algorithm are, but this is impossible to exactly evaluate because we see no way of finding the optimal schedules for nontrivial instances. We can, however, modify the greedy algorithms to produce, in addition to a schedule, what we call a *certificate of approximation*, which is a fraction α such that the produced schedule is guaranteed to be at least an α-approximation of the optimal schedule.[6] As shown in Fig. 1, for example, greedily optimizing the objective $\hat{g}^{1/2}$ produces schedules that are a 0.45-approximation or better across all instances and numbers of sessions we study. We stress that these certificates are lower bounds, and that the schedules are likely to be much closer to optimal than is guaranteed by the certificates. For example, the perfect greedy algorithm (i.e., with perfectly optimal ILP solutions) would have a certificate of $1 - (1 - \frac{1}{T})^T \approx 0.63$, but typically performs much closer to optimal.[7] The proximity of the certificates to this number suggests that terminating the ILP solver yields schedules that are nearly as good as those of the perfect greedy algorithm and not far from the optimal objective value.

6.2 Comparison across Table-Allocation Algorithms

After having measured the greedy algorithm in terms of the objective it specifically aims to optimize, we now compare the performance of different algorithms on a given instance and according to the same metric. In Fig. 2, we show such results for `sf_f` and $\hat{f}^1$; experiments for other instances and objectives can be found in the full version. This instance–objective combination is particularly relevant to investigate, since the Sortition Foundation did, in fact, maximize $\hat{f}^1$ using GROUPSELECT for this assembly, and since we know the table allocation (for $T = 4$ sessions) determined at the time. As the figure shows quite dramatically, GROUPSELECT cannot compete with our other algorithms. Indeed, the Sortition Foundation chose a schedule with 164 distinct meetings for four sessions. By contrast, greedily maximizing $\hat{f}^1$ yields an objective value of nearly twice that, at 320 distinct meetings. Across our datasets and objective functions, GROUPSELECT leads to objective values that stagnate at a much lower level than what our algorithms can achieve, for reasons which we explain in the full version of this paper. This observation is a powerful argument for practitioners to move away from GROUPSELECT.

As is not very surprising, greedily optimizing $\hat{f}^1$ produces schedules with many unique meetings. Given that `sf_f` has $\binom{40}{2} = 780$ pairs of agents, around 90% of pairs meet at least once within the first 20 sessions. More surprisingly, greedily optimizing a geometric objective or the harmonic objective leads to numbers of distinct meetings that are nearly as high, across all numbers of sessions T we study. Indeed, throughout our experiments, we see that greedily optimizing $\hat{g}^{1/2}$ or $\hat{h}$ leads to "well-rounded" schedules in the sense that they perform well according to other objective metrics, which makes either algorithm an attractive option for adoption in practice. Optimizing $\hat{f}^1$ tends to perform very well on other objectives when T is small but falls behind for larger T, when encouraging, say, second meetings becomes an important aspect of what makes a partition contribute to the objective.

A straight-forward implementation of SIMAPPROX does not perform as well as the above-mentioned algorithms, even if still much better than GROUPSELECT. A possible explanation is that SIMAPPROX spends much of its time optimizing objectives $\hat{f}^r$ for fairly large r. If most pairs have met fewer than r times at that point, the ILP might have a large number of optimal solutions, between which the ILP has no preference. To mitigate this problem, we test a variant of SIMAPPROX called SIMAPPROX+, which spends an extra 30 seconds after each ILP call to break ties in favor of partitions with the more well-rounded objective $\hat{g}^{1/2}$. As shown in the figure, SIMAPPROX+ gets substantially closer to the performance of the best greedy algorithms. While such variants of the simultaneous-approximation algorithm might have value for highly constrained table allocation problems or for large numbers of sessions, greedily optimizing $\hat{g}^{1/2}$ or $\hat{h}$ seems more worthwhile on the practical instances we study.

7 Discussion

As the last section shows, our algorithms produce schedules that excel in terms of the objective chosen by the practitioners, as well as in terms of the generalized objectives we introduced. The fundamental research problem, however — optimizing the group assignment in a way that increases the quality of deliberation — remains wide open and will require a multi-faceted approach. According to a handbook for assembly organizers, mixing groups up has a whole range of benefits: it helps assembly members "find common ground across the *whole* diverse group" (emphasis added), avoids situations where they "form cliques," breaks up unproductive group dynamics, and overall "keeps things energised" (newDemocracy Foundation and United Nations Democracy Fund 2018). Not only might each of these benefits suggest a different schedule, but predicting how well a schedule promotes each of these effects is also an open question. We believe that an approach combining optimization, behavioral research, and dynamic models of deliberation (Chung and Duggan 2020; Fain et al. 2017) can substantially support citizens' assemblies and, by extension, democratic innovation.

[5] For an assembly with many sessions (30), the total optimization runs in around one hour, which is the runtime of the assembly selection algorithm by Flanigan et al. (2021a) for large assemblies.

[6] Calculating these certificates is possible since (1) the ILP solver returns, in every step, not only a new partition but also an upper bound on the largest possible marginal increase, and since (2) these bounds naturally fit into the approximation bound by Nemhauser, Wolsey, and Fisher (1978). This ex post analysis combines the strengths of ILP and submodular maximization and is, to our knowledge, novel.

[7] The certificates are also conservative in that the ILP solver often struggles with tightening the upper bounds. Thus, each partition's marginals are probably closer to optimal than our bounds suggest.

References

Ahmed, F.; Dickerson, J. P.; and Fuge, M. 2017. Diverse Weighted Biparite b-Matching. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 35–41.

Barman, S.; Fawzi, O.; and Fermé, P. 2021. Tight Approximation Guarantees for Concave Coverage Problems. In *Proceedings of the International Symposium on Theoretical Aspects of Computer Science (STACS)*, 9:1–9:17.

Benadè, G.; Gölz, P.; and Procaccia, A. D. 2019. No Stratification Without Representation. In *Proceedings of the 20th ACM Conference on Economics and Computation (EC)*, 281–314.

Carbonell, J.; and Goldstein, J. 1998. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 335–336.

Chung, H.; and Duggan, J. 2020. A Formal Theory of Democratic Deliberation. *American Political Science Review*, 114(1): 14–35.

Do, V.; Atif, J.; Lang, J.; and Usunier, N. 2021. Online Selection of Diverse Committees. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI)*, 154–160.

Fain, B.; Goel, A.; Munagala, K.; and Sakshuwong, S. 2017. Sequential Deliberation for Social Choice. In *Proceedings of the 13th Conference on Web and Internet Economics (WINE)*, 177–190.

Fishkin, J. 2018. *Democracy When the People Are Thinking: Revitalizing Our Politics Through Public Deliberation*. Oxford University Press.

Fishkin, J.; Garg, N.; Gelauff, L.; Goel, A.; Munagala, K.; Sakshuwong, S.; Siu, A.; and Yandamuri, S. 2019. Deliberative Democracy with the Online Deliberation Platform. In *Proceedings of the 7th AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*.

Flanigan, B.; Gölz, P.; Gupta, A.; Hennig, B.; and Procaccia, A. D. 2021a. Fair Algorithms for Selecting Citizens' Assemblies. *Nature*, 596: 548–552.

Flanigan, B.; Gölz, P.; Gupta, A.; and Procaccia, A. D. 2020. Neutralizing Self-Selection Bias in Sampling for Sortition. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Flanigan, B.; Gölz, P.; Gupta, A.; Procaccia, A. D.; and Rusak, G. 2021b. Panelot. https://www.panelot.org/. Accessed: 2023-04-13.

Flanigan, B.; Kehne, G.; and Procaccia, A. D. 2021. Fair Sortition Made Transparent. In *Proceedings of the 35th Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Goel, A.; and Lee, D. T. 2016. Towards Large-Scale Deliberative Decision-Making: Small Groups and the Importance of Triads. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)*, 287–303.

Goundan, P. R.; and Schulz, A. S. 2007. Revisiting the Greedy Approach to Submodular Set Function Maximization. Working paper.

Hochbaum, D. S.; and Pathria, A. 1998. Analysis of the Greedy Approach in Problems of Maximum k-Coverage. *Naval Research Logistics*, 45(6): 615–627.

Kapralov, M.; Post, I.; and Vondrák, J. 2013. Online Submodular Welfare Maximization: Greedy Is Optimal. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1216–1225.

Krause, A.; Singh, A.; and Guestrin, C. 2008. Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *Journal of Machine Learning Research*, 9(2).

Landemore, H. 2020. *Open Democracy: Reinventing Popular Rule for the Twenty-First Century*. Princeton University Press.

Lardeux, F.; Monfroy, E.; Crawford, B.; and Soto, R. 2015. Set Constraint Model and Automated Encoding into SAT: Application to the Social Golfer Problem. *Annals of Operations Research*, 235(1): 423–452.

Lin, H.; and Bilmes, J. 2011. A Class of Submodular Functions for Document Summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT)*, 510–520.

Meir, R.; Sandomirskiy, F.; and Tennenholtz, M. 2021. Representative Committees of Peers. *Journal of Artificial Intelligence Research*, 71: 401–429.

Nemhauser, G. L.; Wolsey, L. A.; and Fisher, M. L. 1978. An Analysis of Approximations for Maximizing Submodular Set Functions – I. *Mathematical Programming*, 14(1): 265–294.

newDemocracy Foundation; and United Nations Democracy Fund. 2018. Enabling National Initiatives to Take Democracy Beyond Elections. Technical report.

OECD. 2020. *Innovative Citizen Participation and New Democratic Institutions: Catching the Deliberative Wave*. OECD.

Perote-Peña, J.; and Piggins, A. 2015. A Model of Deliberative and Aggregative Democracy. *Economics & Philosophy*, 31(1): 93–121.

Pokutta, S.; Singh, M.; and Torrico, A. 2020. On the Unreasonable Effectiveness of the Greedy Algorithm: Greedy Adapts to Sharpness. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 7772–7782.

Saran, R.; and Tumennasan, N. 2013. Whose Opinion Counts? Implementation by Sortition. *Games and Economic Behavior*, 78: 72–84.

Schmand, D.; Schröder, M.; and Vargas Koch, L. 2022. A Greedy Algorithm for the Social Golfer and the Oberwolfach Problem. *European Journal of Operational Research*, 300(1): 310–319.

Stein, C.; and Wein, J. 1997. On the Existence of Schedules That Are Near-Optimal for Both Makespan and Total Weighted Completion Time. *Operations Research Letters*, 21(3): 115–122.

Triska, M.; and Musliu, N. 2012. An Effective Greedy Heuristic for the Social Golfer Problem. *Annals of Operations Research*, 194(1): 413–425.

Van Reybrouck, D. 2016. *Against Elections: The Case for Democracy*. Random House.

Verpoort, P. 2020. GroupSelect App. https://github.com/sortitionfoundation/groupselect-app/tree/8fe508f. Accessed: 2023-04-13.

Walsh, T.; and Xia, L. 2012. Lot-Based Voting Rules. In *Proceedings of the 11th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 603–610.

Zvi, G. B.; Leizerovich, E.; and Talmon, N. 2021. Iterative Deliberation via Metric Aggregation. In *Proceedings of the 7th International Conference on Algorithmic Decision Theory (ADT)*, 162–176.