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

Jake Barrett,[1] **Ya'akov (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

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 [8, 19, 32] 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 [25]. 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 [19]; in particular, the recommendations of the Citizens Convention for Climate, established in 2019 with the blessing of President Macron, 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. [10, 11, 13] 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 [33] 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 [23] 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 [30]. 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. This result applies not just to table allocation, but to maximum-coverage style problems in all other domains. Specifically, we show that the f-MAXCOVERAGE problem recently introduced by Barman, Fawzi, and Fermé [2] 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 [3, 6, 22, 28, 34] 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 [5, 7, 14, 26, 35] 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. [9] 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 [4, 21], sensors to be placed where they can collect complementary information [18], or papers to be assigned to reviewers with different expertise [1]. 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 [16] and coincides with the class of f-MAXCOVERAGE problems (for saturation functions f) defined by Barman et al. [2]. 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., 20, 29, 31]. 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

[1]In some assemblies in practice, partitions must additionally satisfy *clustering constraints*, which require that some participants

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 et al. [2], 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

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

For the saturation function $f^1(x) := \mathbb{1}\{x \geq 1\} = \min\{x, 1\}$, 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.

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 Appendix A.1, we show that the minimum number of repeated meetings is at least $k^2 \binom{x}{2} + x \cdot y$ in this case, where $x := \lfloor n/k^2 \rfloor$ and $y := n \bmod k^2$.

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 Appendix B, 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 partition without violating representativeness, (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, even if this means that the overwhelming majority of pairs meet either excessively often or just once. In such instances, it seems preferable to forgo some first meetings in P in order to allow a large number of pairs to meet a second time.

Motivated by the above limitations of $\hat{f}^1$, we generalize the optimization problem introduced by the Sortition Foundation by considering 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 sat-

(e.g. those unwilling to be photographed or those in need of translation services) be grouped together. Since our approach in Section 4 generalizes to clustering in a straight-forward way, we omit these constraints 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 $\sqsubseteq$.

uration function $f^r(x) := \min\{x, r\}$, 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 $g^\beta(x) := \sum_{i=1}^{x} \beta^i$. 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 $h(x) := \sum_{i=1}^{x} 1/i$. 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.[3]

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 (Appendix C). Thus, polynomial-time algorithms cannot produce partitions or schedules (unless $\mathsf{P} = \mathsf{NP}$), 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 [23] 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.[4] 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 $1 \leq \tau \leq k$:

$$\begin{aligned}
\text{maximize} \quad & \sum_{\substack{\{i,j\} \in \binom{N}{2} \\ 1 \leq \tau \leq k}} \big(f(m_{i,j} + 1) - f(m_{i,j})\big) \cdot y_{\{i,j\},\tau} \\
\text{subject to} \quad & \sum_{1 \leq \tau \leq k} x_{i,\tau} = 1 && \forall i \in N \\
& \lfloor n/k \rfloor \leq \sum_{i \in N} x_{i,\tau} \leq \lceil n/k \rceil && \forall 1 \leq \tau \leq k \\
& \ell_\varphi \leq \sum_{i \in A_\varphi} x_{i,\tau} \leq u_\varphi && \forall 1 \leq \tau \leq k, \varphi \in F \\
& \left.\begin{array}{l} y_{\{i,j\},\tau} \geq x_{i,\tau} + x_{j,\tau} - 1 \\ y_{\{i,j\},\tau} \leq x_{i,\tau} \\ y_{\{i,j\},\tau} \leq x_{j,\tau} \end{array}\right\} && \begin{array}{r} \forall \{i,j\} \in \binom{N}{2}, \\ 1 \leq \tau \leq k \end{array} \\
& x_{i,\tau} \in \{0,1\}, y_{\{i,j\},\tau} \in \{0,1\} && \forall i, j, \tau.
\end{aligned}$$

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.

[3]Consider a setting where we can achieve all pairs meeting exactly ten times. In an ideal setting, the problem would "reset" as if nobody had met: the tradeoff between participants meeting for the nth and $(n+1)$th times should be constant regardless of n. This property is satisfied by the geometric utility function, but for a harmonic utility function we will have a decreasing difference in utilities as n increases. For example, three third meetings have the same marginal utility as two second meetings ($\frac{3}{3} = \frac{2}{2}$), but once every pair has met ten times, three twelfth meetings have a higher marginal utility than two eleventh meetings ($\frac{3}{12} > \frac{2}{11}$).

[4]Technically, Nemhauser et al. [23] prove this for submodular *set* functions. Our setting differs slightly since we allow sets to 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 [17].

We can run the greedy maximization algorithm by iterating the following steps T times: solving the ILP, extracting the 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 et al. [23], and the greedy algorithm is known to outperform this approximation factor in many cases [27]. 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 [15].

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     $p \leftarrow \lfloor (t/T) \cdot (1 + \log_2 T) \rfloor$
4     $Z \leftarrow Z + \left\{ \mathrm{argmax}_{S \in \mathcal{Z}} \, \hat{f}^{2^p}\left(Z + \{S\}\right) \right\}$
5 return $Z$
```

This algorithm and our analysis of simultaneous approximation apply not only to table allocation but to all f-MAX-COVERAGE 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 Appendix D.1.

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 Lemma D.4 in Appendix D.1, 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

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

In Appendix D.2, 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. We include our implementation in the supplementary material and will release it as open source. Currently, we are working with the Sortition Foundation to incorporate our algorithms into the tool that hosts GROUPSELECT [33], 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. [10]. 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 [12] to obtain an assembly that satisfies the actual representativeness constraints. In Appendix E.1, 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

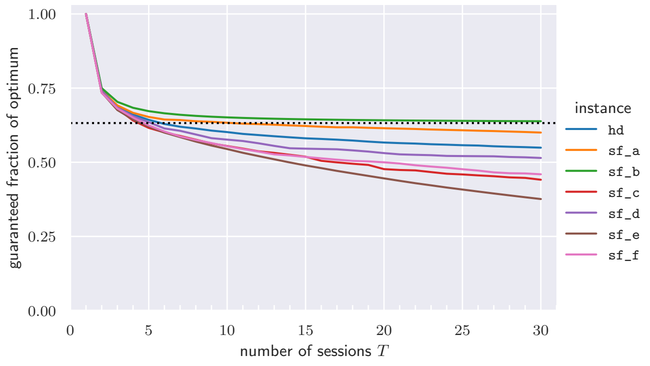


Figure 1: Approximation certificates for the greedy algorithm on $\hat{g}^{1/2}$, guaranteeing near-optimality. The dashed line marks $1 - 1/e$.

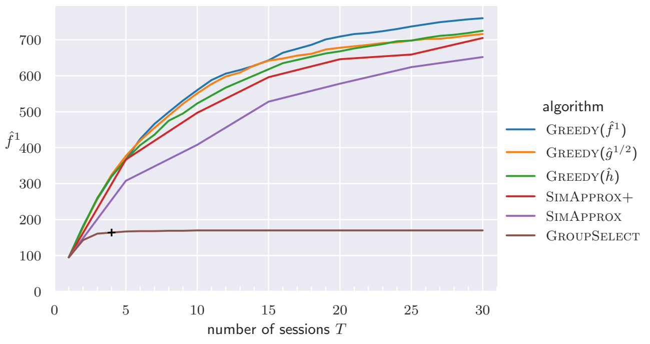


Figure 2: Performance of different algorithms on instance sf_f as measured by $\hat{f}^1$. The cross marks the historically chosen schedule.

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 Appendix E.3, we show that our algorithms continue to perform well for particularly tight representativeness constraints.)

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.[5] 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

[5]For the other 3 instances, the objective value is no better (within $\pm1\%$) than the greedy algorithm's result produced in 8 minutes.

optimize its objective well? In Appendix E.3, 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 optimization timeout to 120 seconds from here on, for the ILP calls both in the greedy algorithm and in SIMAPPROX.[6]

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.[7] 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.[8] 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 Appendix E.3. 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 Appendix E.2. 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" [24]. 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 [5, 7] can substantially support citizens' assemblies and, by extension, democratic innovation.

[6]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. [10] for large assemblies.

[7]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 et al. [23]. This ex post analysis combines the strengths of ILP and submodular maximization and is, to our knowledge, novel.

[8]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

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

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

[9] 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)*.

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

[11] 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)*.

[12] Flanigan, B.; Gölz, P.; Gupta, A.; Procaccia, A. D.; and Rusak, G. 2021. Panelot. [accessed: May 2022].

[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)*.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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).

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

[20] 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.

[21] 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.

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

[23] 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.

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

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

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

[27] 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.

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

[29] 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.

[30] 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.

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

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

[33] Verpoort, P. 2020. GroupSelect App.

[34] 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.

[35] 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.

Appendix

A Proofs for the Two Round Case

A.1 Minimum Number of Repeated Meetings Between Two Rounds

Definitions & Assumptions As defined in Section 2, suppose we have a set of n participants $[n] = \{0, \ldots, n-1\}$ and a set of k groups $[k] = \{0, \ldots, k-1\}$, and that participants are distributed among groups as evenly as possible, according to the size function Z:

$$Z(g) = \begin{cases} \lceil n/k \rceil & \text{if } g < (n \bmod k) \\ \lfloor n/k \rfloor & \text{otherwise} \end{cases} \tag{1}$$

We consider arbitrary back-to-back sessions $s, t \leq T$ where $t = s + 1$, so for convenience, let $s = 1$ and $t = 2$. Note that we only consider assemblies without clustered participants, and that the bound we derive does not take into account representation constraints (i.e., groups do not need to meet any specific representation quotas), so there is no need to associate attributes with each participant.

Now, for an arbitrary allocation A, let A be a function such that for a session $s \leq T$ and any participant $i \in [n]$, $A_s(i)$ returns the group that i has been assigned to for s. For each pair of participants (i, j) where $i, j \in [n]$ and $i < j$ and sessions $s, t \leq T$ where $s < t$, we say that (i, j) is a *repeated meeting* if the pair meets during both sessions; in other words, we can define an indicator $I_{s,t}(i, j)$ that corresponds to whether (i, j) is a repeated meeting as follows:

$$I_{s,t}(i,j) = \begin{cases} 1 & \text{if } (A_s(i) = A_s(j)) \wedge (A_t(i) = A_t(j)) \\ 0 & \text{otherwise} \end{cases}.$$

Objective For an arbitrary allocation A over the back-to-back sessions s and t, the number of repeated meetings generated between the two sessions can be expressed as

$$RM_{s,t}(A) = \sum_{i,j \in [n] \text{ s.t. } i<j} I_{s,t}(i,j).$$

We thus aim to minimize $RM_{s,t}(A)$ over all feasible allocations A.

Theorem (Minimum Repeated Meetings) The minimum number of repeated meetings $RM^*_{s,t}$ that can be achieved between back-to-back sessions in assemblies without clustered participants is

$$RM^*_{s,t} = \begin{cases} k^2 \binom{x}{2} + x \cdot y & \text{if } n > k^2 \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

where $x = \lfloor n/k^2 \rfloor$ and $y = n \bmod k^2$.

Proof For convenience, we will simply refer to participants assigned to a group $g \in [k]$ in session 1 as "g_1-participants." Consider an arbitrary allocation for session 1. We first prove the following lemma:

Lemma A.1. *For any $g \in [k]$, distributing the $Z(g)$ g_1-participants as evenly as possible among all k groups in session 2 minimizes the number of repeated meetings between g_1-participants.*

Proof. Fix an arbitrary group $g \in [k]$. For an arbitrary session-2 allocation A of the $Z(g)$ g_1-participants, we can adapt the notation from Section A.1 and let $A(i)$ denote the group that the g_1-participant i has been assigned to in A. Then, for any group $h \in [k]$, $A^{-1}(h)$ denotes the set of g_1-participants that were assigned to h in A.

First, consider any session-2 allocation A of the g_1-participants that contains two groups $\gamma, \gamma' \in [k]$ such that $|A^{-1}(\gamma)| \geq |A^{-1}(\gamma')| + 2$. We claim that A does not minimize the number of repeated meetings between g_1-participants; in particular, we can construct another allocation B from A that results in fewer such repeated meetings: Let B be identical to A, except with one of the $A^{-1}(\gamma)$ participants reassigned to γ'. Then, by definition, the number of repeated meetings between the g_1-participants in A is

$$RM_{1,2}(A) = \sum_{h \in [k]} \binom{|A^{-1}(h)|}{2},$$

while the number of repeated meetings between the g_1-participants in B is

$$\begin{aligned} RM_{1,2}(B) &= \sum_{h \in [k]} \binom{|B^{-1}(h)|}{2} \\ &= \binom{|A^{-1}(\gamma)| - 1}{2} + \binom{|A^{-1}(\gamma')| + 1}{2} + \sum_{h \in [k] \setminus \{\gamma, \gamma'\}} \binom{|A^{-1}(h)|}{2}. \end{aligned}$$

Rewriting $RM_{1,2}(A)$ as

$$\binom{|A^{-1}(\gamma)|}{2} + \binom{|A^{-1}(\gamma')|}{2} + \sum_{h \in [k] \setminus \{\gamma, \gamma'\}} \binom{|A^{-1}(h)|}{2},$$

we can then compute $RM_{1,2}(A) - RM_{1,2}(B)$ as follows:

$$\begin{aligned}
RM_{1,2}(A) - RM_{1,2}(B) &= \binom{|A^{-1}(\gamma)|}{2} + \binom{|A^{-1}(\gamma')|}{2} - \binom{|A^{-1}(\gamma)| - 1}{2} - \binom{|A^{-1}(\gamma')| + 1}{2} \\
&= \frac{|A^{-1}(\gamma)|(|A^{-1}(\gamma)| - 1)}{2} + \frac{|A^{-1}(\gamma')|(|A^{-1}(\gamma')| - 1)}{2} \\
&\quad - \frac{(|A^{-1}(\gamma)| - 1)(|A^{-1}(\gamma)| - 2)}{2} - \frac{|A^{-1}(\gamma')|(|A^{-1}(\gamma')| + 1)}{2} \\
&= |A^{-1}(\gamma)| - |A^{-1}(\gamma')| - 1 \\
&\geq (|A^{-1}(\gamma')| + 2) - |A^{-1}(\gamma')| - 1 = 1.
\end{aligned}$$

We have thus shown that $RM_{1,2}(A) - RM_{1,2}(B) > 0 \implies RM_{1,2}(B) < RM_{1,2}(A)$, and A does not minimize the number of repeated meetings between g_1-participants as desired.

Now, consider any session-2 allocation A of the g_1-participants that does not contain two groups $\gamma, \gamma' \in [k]$ such that $|A^{-1}(\gamma)| \geq |A^{-1}(\gamma')| + 2$, i.e., A such that for all $\gamma \in [k]$, the $|A^{-1}(\gamma)|$ differ by at most 1. We claim that all such allocations achieve the same number of repeated meetings between the g_1-participants: Let A be such an allocation, and for convenience, let $x_g = \lfloor Z(g)/k \rfloor$ and $y_g = Z(g) \bmod k$. Then, in order for the condition to hold, it is easy to verify that A must assign exactly $(x_g + 1)$ g_1-participants to y_g of the k groups and x_g g_1-participants to the remaining $k - y_g$ groups. The number of repeated meetings between g_1-participants in A is thus always

$$\begin{aligned}
RM_{1,2}(A) &= y_g \binom{x_g + 1}{2} + (k - y_g) \binom{x_g}{2} \\
&= y_g \frac{x_g(x_g + 1)}{2} + (k - y_g) \frac{x_g(x_g - 1)}{2} \\
&= k \binom{x_g}{2} + y_g x_g. \qquad (3)
\end{aligned}$$

Putting everything together, we can see that in order for a session-2 allocation to minimize the number of repeated meetings between g_1-participants, it must distribute the g_1-participants as evenly as possible among all k groups. But since all allocations that distribute the g_1-participants as evenly as possible achieve the same number of repeated meetings between g_1-participants, the allocations that minimize this number are exactly the allocations that distribute the g_1-participants as evenly as possible among all k groups. The statement of the lemma thus holds as desired. □

Corollary A.1.1. *For each $g \in [k]$, distributing the g_1-participants as evenly as possible among all k groups in session 2 minimizes the total number of repeated meetings between the two sessions.*

Proof. By Lemma A.1, it follows that for each $g \in [k]$, distributing the $Z(g)$ g_1-participants as evenly as possible among all k groups in session 2 minimizes the number of repeated meetings between g_1-participants. But because repeated meetings can only form between participants who were in the same group in session 1, distributing all groups in such a way in session 2 actually minimizes the total number of repeated meetings between the two sessions. We can thus compute $RM^*_{1,2}$ using Equation 3 as follows:

$$RM^*_{1,2} = \sum_{g \in [k]} k \binom{x_g}{2} + y_g x_g. \qquad (4)$$

Moreover, it is clear that this minimum is actually attainable: Algorithm 2 (p. 12) generates a session-2 allocation that for each $g \in [k]$ distributes the g_1-participants as evenly as possible among all k groups; it also guarantees that the right number of participants are assigned to each session-2 group (i.e., according to the size function Z defined in A.1), thus ensuring that the allocation is feasible. □

Returning to the proof of Theorem A.1, we now show how Equation 4 can be rewritten as Equation 2. We split the proof into three cases based on the relationship between n and k^2.

Case 1 ($n = k^2$):

Algorithm 2: Corollary A.1.1 – Sample Assignment Strategy

```
Input  : a partition $\Delta_0^1 \dot\cup \Delta_1^1 \dot\cup \cdots \dot\cup \Delta_{k-1}^1$ for the first session
Output : a partition $\Delta_0^2 \dot\cup \Delta_1^2 \dot\cup \cdots \dot\cup \Delta_{k-1}^2$ for the second session
1 $c \leftarrow 0$
2 $\Delta_0^2 \leftarrow \emptyset; \Delta_1^2 \leftarrow \emptyset; \ldots; \Delta_{k-1}^2 \leftarrow \emptyset$
3 for $\tau = 0, 1, \ldots, k-1$ do
4     for $i \in \Delta_\tau^1$ do
5         $\Delta_{c \bmod k}^2 \leftarrow \Delta_{c \bmod k}^2 \cup \{i\}$
6         $c \leftarrow c + 1$
7 return the partition $\Delta_0^2 \dot\cup \Delta_1^2 \dot\cup \cdots \dot\cup \Delta_{k-1}^2$
```

We claim that $RM_{1,2}^* = 0$: Since $n = k^2 \equiv 0 \bmod k$, $Z(g) = \lfloor k^2/k \rfloor = k$ for all $g \in [k]$. But then $x_g = \lfloor k/k \rfloor = 1$ and $k \equiv 0 \bmod k \implies y_g = 0$ for all $g \in [k]$, so it follows from Equation 4 that

$$\begin{aligned} RM_{1,2}^* &= \sum_{g \in [k]} k\binom{1}{2} + 0 \cdot 1 \\ &= \sum_{g \in [k]} 0 \\ &= 0. \end{aligned}$$

Intuitively, this makes sense: Suppose we fix an arbitrary group $g \in [k]$. In order to achieve 0 repeated meetings between g_1-participants, the k g_1-participants must be assigned to distinct groups (which may include g) in session 2; otherwise, at least one pair among the g_1-participants will be meeting for the second time. But this is clearly possible, as there are k total groups, and we can simply assign one of the g_1-participants to each group. Moreover, since $Z(g) = k$ for all $g \in [k]$, it follows by symmetry that all k^2 participants can be reallocated such that there are 0 repeated meetings between sessions 1 and 2; one such allocation that attains this minimum, for example, is generated by Algorithm 2.

Case 2 ($n < k^2$):

We again claim that $RM_{1,2}^* = 0$: By definition, $Z(g) \le k$ for all $g \in [k]$. It thus follows that $x_g \le \lfloor k/k \rfloor = 1 \implies \binom{x_g}{2} = 0$ for all $g \in [k]$. Now, consider an arbitrary group $g \in [k]$. If $x_g = 0$, it immediately follows that $y_g x_g = 0$. Otherwise, if $x_g = 1$, we must have that $Z(g) = k$. Thus, $k \equiv 0 \bmod k \implies y_g = 0$, and we again have that $y_g x_g = 0$. Putting everything together, it follows from Equation 4 that

$$\begin{aligned} RM_{1,2}^* &= \sum_{g \in [k]} k\binom{x_g}{2} + y_g x_g \\ &= \sum_{g \in [k]} 0 \\ &= 0. \end{aligned}$$

Moreover, as in Case 1, one such allocation that attains this minimum is generated by Algorithm 2.

Case 3 ($n > k^2$):

Finally, in the case that $n > k^2$, we use Algorithm 2 to show how Equation 4 can be rewritten as Equation 2. In particular, we already know from Corollary A.1.1 that Algorithm 2 generates a session-2 allocation A that minimizes the total number of repeated meetings between sessions 1 and 2, so we can simply compute $RM_{1,2}^*$ by counting the total number of repeated meetings between the session-1 allocation and A. We split Case 3 into two subcases.

Subcase 3-1: This subcase will provide intuition for the more general subcase to follow. Consider when $n = xk^2$ for $x \in \mathbb{N}$ such that $x > 1$. By definition, $Z(g) = xk$ for all $g \in [k]$. Now, suppose without loss of generality that for each $g \in [k]$, the set of g_1-participants is labeled $g_1 = \{0, 1, \ldots, Z(g) - 1\}$. In order to count the number of repeated meetings between the session-1 allocation and A, we can first imagine dividing the n participants into $n/k^2 = x$ "blocks" of size k^2 such that block 1 contains the set $[k] \subset g_1$ (i.e., the first k g_1-participants) for all $g \in [k]$, block 2 contains the set $[2k] \backslash [k] \subset g_1$ (i.e., the next k g_1-participants) for all $g \in [k]$, and so on. More explicitly, the x blocks $b_1, b_2, \ldots, b_x$ are

$$b_1 : \{\underbrace{0, 1, \ldots, k-1}_{0_1\text{-participants}}, \underbrace{0, 1, \ldots, k-1}_{1_1\text{-participants}}, \ldots, \underbrace{0, 1, \ldots, k-1}_{(k-1)_1\text{-participants}}\},$$

$$b_2 : \{\underbrace{k, k+1, \ldots, 2k-1}_{0_1\text{-participants}}, \underbrace{k, k+1, \ldots, 2k-1}_{1_1\text{-participants}}, \ldots, \underbrace{k, k+1, \ldots, 2k-1}_{(k-1)_1\text{-participants}}\},$$

$$\vdots$$

$$b_x : \{\underbrace{(x-1)k, (x-1)k+1, \ldots, xk-1}_{0_1\text{-participants}}, \underbrace{(x-1)k, (x-1)k+1, \ldots, xk-1}_{1_1\text{-participants}}, \ldots,$$
$$\underbrace{(x-1)k, (x-1)k+1, \ldots, xk-1}_{(k-1)_1\text{-participants}}, \}.$$

This formulation simplifies our counting process. In particular, we can consider how Algorithm 2 assigns the participants in each block to groups as follows: Fix an arbitrary group $g \in [k]$. For each block, it is clear that Algorithm 2 assigns exactly one of the k g_1-participants to each of the k groups in A. Then, since there are x total blocks, Algorithm 2 must assign exactly x g_1-participants to each of the k groups in A. We can thus compute $RM^*_{1,2}$ as follows: Because any pair of these x g_1-participants (which we henceforth refer to as a "g_1-clique") has already met in session 1, they generate $\binom{x}{2}$ repeated meetings between the two sessions. Then, since there are k of these g_1-cliques (where g_1 is fixed), and k possible values for g_1, it follows that

$$RM^*_{1,2} = k^2 \binom{x}{2}.$$

Moreover, since $xk^2 \equiv 0 \bmod xk^2 \implies y = 0 \implies x \cdot y = 0$, this result matches the one given in Equation 2.

Subcase 3-2: Now, consider when $n = xk^2 + y$ for $x, y \in \mathbb{N}$ such that $x > 0$ and $0 < y < k^2$. In order to count the number of repeated meetings between the session-1 allocation and A, we can again imagine dividing the n participants into blocks. In particular, we already know from Subcase 3-1 that the first $\lfloor n/k^2 \rfloor = x$ blocks of size k^2 contribute $k^2\binom{x}{2}$ repeated meetings to the total count. In this subcase, however, we also have an additional "remainder" block of size $y = n \bmod k^2$. We can again consider how Algorithm 2 assigns the participants in this remainder block to groups as follows: First, because participants are distributed among groups as evenly as possible in all sessions (and, in particular, in session 1), the remainder block consists of at most k g_1-participants for all $g \in [k]$. It is thus clear that for each $g \in [k]$, Algorithm 2 assigns at most one of the g_1-participants to each of the k groups in A. Then, since there are already x g_1-participants in each group, assigning any participant in the remainder block to a group increases the number of repeated meetings by x. The remainder block thus contributes $x \cdot y$ repeated meetings to the total count, and it follows that

$$RM^*_{1,2} = k^2 \binom{x}{2} + x \cdot y.$$

The statement of the theorem thus holds as desired. □

Corollary (Maximum Links) Finally, the bound on the minimum number of repeated meetings that can be achieved between back-to-back sessions in assemblies without clustered participants also yields a bound on the maximum number of links $L^*_{s,t}$ that can be achieved between such sessions. In particular, the naive maximum (i.e., all possible meetings that can generated between two sessions) is $m = 2\sum_{g\in[k]} \binom{Z(g)}{2}$, so adjusting for the minimum repeated meetings yields

$$L^*_{s,t} = \begin{cases} m - \left(k^2\binom{x}{2} + x \cdot y\right) & \text{if } n > k^2 \\ m & \text{otherwise} \end{cases}, \tag{5}$$

where $x = \lfloor n/k^2 \rfloor$ and $y = n \bmod k^2$.

B Sub-Optimal Results from Prioritizing First Meetings

To demonstrate the limitations of optimizing unique meetings, we construct a family of table allocation instances in which this objective leads to a controversial schedule.

Each such instance is parameterized by a prime p. For such a p, we construct an instance with $n := p\,(p+2)$ agents, $k := p$ tables, and $T := \binom{p}{2}$ sessions. We will refer to p agents as "heavies", to p other agents as "mediums", and to the remaining p^2 agents as "lightweights". We will further label each lightweight with a number: p many lightweights are labelled 1, two lightweights are labelled 2, and (for all $t = 3, 4, \ldots, k$) $p+1$ many lightweights are labeled t.

The key part of the instance are the representativeness constraints. For ease of exposition, we will not give them explicitly, but define them implicitly using the following lemma:

Lemma B.1. *Fix a set of agents $N = [n]$ and a number of tables k such that k divides n.*[9] *Furthermore, fix an arbitrary set $\mathcal{P} \subseteq \binom{N}{k}$, i.e., any set $\mathcal{P}$ whose elmeents are sets of k many agents each. Then, there exists a set of representativeness constraints such that a set Δ of k agents is a representative panel iff $\Delta \in \mathcal{P}$.*

[9]This last assumption is made for convenience, not because it is fundamental to the lemma.

Proof. Create one feature f_Δ for each $\Delta \in \binom{N}{k} \setminus \mathcal{P}$, and set $A_{f_\Delta} := \Delta$, $\ell_{f_\Delta} = 0$, and $u_{f_\Delta} = k-1$, i.e., the constraint for feature f_Δ rules out that a table might be exactly Δ. One easily verifies that any $\Delta \in \mathcal{P}$ satisfies all these constraint, and that no $\Delta \in \binom{N}{k} \setminus \mathcal{P}$ does. □

Recall that, in the instance we are constructing, each table must contain $p+2$ agents. Using the above lemma, we define representativeness constraints such that a table satisfies representativeness iff one of the following conditions applies

Type "HM": the table contains one heavy, one medium, and any p lightweights,

Type "HL": the table contains one heavy and all $p+1$ lightweights of a single label $3 \leq t \leq k$,

Type "HHL": the table contains two "heavies" and all p lightweights of label 1, or

Type "ML": the table contains all k "mediums", and both lightweights with label 2.

Lemma B.2. *In $3p$ sessions, it is possible to make all pairs of agents meet at least once, except for heavy–heavy pairs.*

Proof. First, we will show that all lightweight-lightweight pairs can meet within $p+1$ sessions: If we keep one heavy and one medium fixed at each table, the lightweights can be split across the tables without restriction (type "HM"), where each table will contain exactly p lightweights. Khare and Federer (1979)[10] show that the lightweights can be scheduled in such a way across $p+1$ rounds that each lightweight–lightweight pair meets exactly once. Second, we will show that all heavy–medium and heavy–lightweight pairs that haven't met yet can meet in $p-1$ additional sessions. For this, simply start from any of the previous table allocations, and cyclically permute the heavies $p-1$ times. Third, we can cyclically permute the mediums for $p-1$ more sessions to ensure that all medium–lightweight pairs have also met. Finally, all medium–medium pairs can meet in one more session: allocate all of them in a table of type "ML", and group all remaining agents in tables of types "HHL" and "HL". Thus all pairs other than heavy–heavy pairs have met in $(p+1) + 2(p-1) + 1 = 3p$ sessions, as claimed. □

Lemma B.3. *Each heavy–heavy pair can meet, but at most one heavy–heavy pair can meet at a time, and, whenever a heavy–heavy pair meets, lightweights of different label do not meet.*

Proof. Fix any two heavies. These two heavies can be placed together with the lightweights of label 1 to form a table of type "HHL". Next, form a table of type "ML" containing all the mediums and the lightweights of label 2. Then, the remaining $p-2$ heavies can be seated on tables of type "HL", where each table contains a heavy and then lightweights of homogeneous label. This shows that heavy–heavy pairs can meet, and one verifies that, in this specific partition, only one pair of heavies meets and no lightweight with different labels meet.

To show the lemma, observe that the above construction is the only possible one: two heavies can only sit on a table of type "HHL"; since one table will not have a heavy by the pigeon-hole principle, it must have type "ML"; and since no mediums remain for the other tables, all of them must have type "HL". □

By the above observations, the schedule optimizing $\hat{f}^1$ over $T = \binom{p}{2}$ sessions must make at least

$$\left(\binom{n}{2} - \binom{p}{2}\right) + \left(\binom{p}{2} - 3\,p\right) = \binom{n}{2} - 3\,p$$

pairs meet. By Lemma B.3, this implies that at least $\binom{p}{2} - 3p$ out of the $\binom{p}{2}$ sessions made heavy–heavy pairs meet and therefore did not let any lightweight–lightweight pairs interact across labels. However, lightweight–lightweight pairs with different labels make up $p^4/2 - \mathcal{O}(p^3)$ of the $p^4/2 + \mathcal{O}(p^3)$ total pairs. If our objective was, say, $\hat{f}^{3\,p}$, the above schedule gives at most $3\,p$ points for each of the $\Omega(p^3)$ meeting that is not between lightweights of different label plus at most $\mathcal{O}(p^2)$ points for lightweight–lightweight meetings across $3\,p$ meetings in which those appear, for a total objective value in $\mathcal{O}(p^4)$. By contrast, if we had simply repeated the first p steps of Lemma B.1 $\Omega(p)$ times, the interactions between lightweights alone would have contributed $\Omega(p) \cdot \binom{p^2}{2} \in \Omega(p^5)$ points to the objective. If p is large enough, this difference becomes substantial. This is one way of capturing the intuition that, in this setup, spending $\Omega(p^2)$ rounds on heavy–heavy meetings that keep the partition mostly unchanged might be less desireable than mixing the lightweight–lightweigh interactions well.

C NP-hardness of Partition Feasability

Proposition C.1. *Deciding whether some partition is possible for the given representativeness constraints is NP-hard.*

Proof. By reduction from equitable graph k-coloring. For each node in the original graph, create an agent, and set the number of tables to k. (For equitable graph coloring, we may assume that k divides the number of nodes.) For each edge in the graph, create a new feature f, where A_f contains the two endpoints of the edge, where $\ell_f = 0$ and $u_f = 1$. Now, there exists a partition iff the graph had an equitable coloring. Indeed, this is easy to see if we identify tables with colors: That table sizes must be equal to n/k directly corresponds to the equitability constraint in the coloring problem; and the representativeness constraints express exactly that no two neighboring nodes may possess the same color. □

[10] M. Khare and W. T. Federer. 1979. A Simple Construction Procedure for Resolvable Incomplete Block Designs for Any Number of Treatments.

D Simultaneous Approximation

D.1 Deferred Proofs

Step 1: Approximation of $\hat{f}^{2^p}$

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. Fix some $0 \leq p \leq \log_2 T$. Line 3 of the algorithm will select this p whenever $p \leq (t/T) \cdot (1+\log_2 T) < p+1$, or equivalently whenever $\frac{p\,T}{1+\log_2 T} \leq t < \frac{(p+1)\,T}{1+\log_2 T}$. This range must contain at least $\lfloor \frac{T}{1+\log_2 T} \rfloor$ integer values of t, and all these integers correspond to iterations of the algorithm since $t \geq \frac{p\,T}{1+\log_2 T} \geq 0$ and $t < \frac{(p+1)\,T}{1+\log_2 T} \leq \frac{((\log_2 T)+1)\,T}{1+\log_2 T} = T$.

Let Z_{t_0} denote the value of Z when p first gets selected in Line 3, and let Z_{t_1} denote the value of Z after completing the subsequent $\lfloor \frac{T}{1+\log_2 T} \rfloor$ steps of optimizing $\hat{f}^{2^p}$. The steps in between can be seen as greedily optimizing a function g mapping selections to real numbers where $g(Z') \coloneqq \hat{f}^{2^p}(Z' + Z_{t_0}) - \hat{f}^{2^p}(Z_{t_0})$. A function of this form is called a *contraction* of $\hat{f}^{2^p}$, which implies that g inherits diminishing returns and monotonicity from $\hat{f}^{2^p}$. By Nemhauser, Wolsey, and Fisher [23],

$$g(Z_{t_1} - Z_{t_0}) \geq (1-1/e) \cdot \max_{\substack{\text{selection } Z' \\ |Z'| = \lfloor \frac{T}{1+\log_2 T} \rfloor}} g(Z'),$$

which implies that

$$\begin{aligned}
\hat{f}^{2^p}(Z) &\geq \hat{f}^{2^p}(Z_{t_1}) = \hat{f}^{2^p}(Z_{t_0}) + g(Z_{t_1} - Z_{t_0}) \\
&\geq \hat{f}^{2^p}(Z_{t_0}) + (1-1/e) \cdot \max_{\substack{\text{selection } Z' \\ |Z'| = \lfloor \frac{T}{1+\log_2 T} \rfloor}} \left(\hat{f}^{2^p}(Z' + Z_{t_0}) - \hat{f}^{2^p}(Z_{t_0}) \right) \\
&= \frac{\hat{f}^{2^p}(Z_{t_0})}{e} + (1-1/e) \cdot \max_{\substack{\text{selection } Z' \\ |Z'| = \lfloor \frac{T}{1+\log_2 T} \rfloor}} \hat{f}^{2^p}(Z' + Z_{t_0}) \\
&\geq (1-1/e) \cdot \max_{\substack{\text{selection } Z' \\ |Z'| = \lfloor \frac{T}{1+\log_2 T} \rfloor}} \hat{f}^{2^p}(Z')
\end{aligned}$$

Finally, it is well known that, for any submodular function $\hat{f}$, the optimal value among sets of cardinality T_1 is at least a T_1/T_2 fraction of the optimal value among sets of cardinality $T_2 \geq T_1$. As claimed, it follows that

$$\hat{f}^{2^p}(Z) \geq \left(1 - \tfrac{1}{e}\right) \frac{\lfloor \frac{T}{1+\log_2 T} \rfloor}{T} \cdot \max_{\text{solution } Z'} \hat{f}(Z') \geq \left(1 - \tfrac{1}{e}\right) \left(\frac{1}{1+\log_2 T} - \frac{1}{T} \right) \cdot \max_{\text{solution } Z'} \hat{f}(Z'). \qquad \square$$

Step 2: Approximation of $\hat{f}^r$

Lemma D.1. *For any $i \geq j$ and any solution Z, $\hat{f}^i(Z) \geq \hat{f}^j(Z)$.*

Proof. Since $f^i(x) \geq f^j(x)$ for all $x \in \mathbb{N}$,

$$\hat{f}^i(Z) = \sum_{g \in G} f^i(\text{\# of sets in } Z \text{ that contain } g) \geq \sum_{g \in G} f^j(\text{\# of sets in } Z \text{ that contain } g) = \hat{f}^j(Z). \qquad \square$$

Lemma D.2. *For any $i \leq j$ and any schedule Z, $\hat{f}^i(Z) \geq \frac{i}{j} \hat{f}^j(Z)$.*

Proof. Since, for all $x \in \mathbb{N}$,

$$f^i(x) = \min(x, i) = \frac{i}{j} \min\left(\frac{j}{i} \cdot x, \frac{j}{i} \cdot i \right) = \frac{i}{j} \min\left(\frac{j}{i} \cdot x, j \right) \geq \frac{i}{j} \min(x, j) = \frac{i}{j} f^j(x),$$

it follows that

$$\begin{aligned}
\hat{f}^i(Z) &= \sum_{g \in G} f^i(\text{\# of sets in } Z \text{ that contain } g) \\
&\geq \sum_{g \in G} \frac{i}{j} \cdot f^j(\text{\# of sets in } Z \text{ that contain } g) = \frac{i}{j} \cdot \hat{f}^j(Z). \qquad \square
\end{aligned}$$

Lemma D.3. *For any α, $i \geq j$, and schedule Z, if Z α-approximates $\hat{f}^j$, then Z must $\alpha \cdot \frac{j}{i}$-approximate $\hat{f}^i$.*

Proof.

$$\begin{aligned}
\hat{f}^i(Z) &\geq \hat{f}^j(Z) && \text{(by Lemma D.1)} \\
&\geq \alpha \cdot \max_{\text{solution } Z'} \hat{f}^j(Z') && \text{(by assumption)} \\
&\geq \alpha \cdot \hat{f}^j(\operatorname*{argmax}_{\text{solution } Z'} \hat{f}^i(Z')) \\
&\geq \alpha \cdot \frac{j}{i} \cdot \max_{\text{solution } Z'} \hat{f}^i(Z'). && \text{(by Lemma D.2)}
\end{aligned}$$ □

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. Fix some r, and let $p := \lfloor \log_2(r) \rfloor$, which means that $2^p \leq r < 2^{p+1}$. By Lemma 5.1, Z approximate $\hat{f}^{2^p}$ within a factor of $(1 - 1/e) \cdot (\frac{1}{1+\log_2 T} - \frac{1}{T})$. By Lemma D.3, it follows that Z approximates $\hat{f}^r$ by a factor of

$$\frac{2^p}{r} \cdot (1 - 1/e) \cdot (\frac{1}{1 + \log_2 T} - \frac{1}{T}) > \frac{1}{2} \cdot (1 - 1/e) \cdot (\frac{1}{1 + \log_2 T} - \frac{1}{T}).$$ □

Step 3: Approximation of Any $\hat{f}$

Lemma D.4. *Fix some saturation function f and a number T of sessions. There exist nonnegative weights $\{w_i\}_{1 \leq i \leq T}$ such that, for all $1 \leq x \leq T$, $f(x) = \sum_{i=1}^{T} w_i \cdot f^i(x)$.*

Proof. Indeed, for any $0 \leq x \leq T$,

$$\begin{aligned}
f(x) &= f(x) - f(0) \\
&= \sum_{i=1}^{x} (f(x) - f(x-1)) \\
&= \sum_{i=1}^{T} (f(x) - f(x-1)) \cdot \mathbb{1}\{x \geq i\} \\
&= \sum_{i=1}^{T} (f(x) - f(x-1)) \cdot \big(f^i(x) - f^{i-1}(x)\big) \\
&= \sum_{i=1}^{T-1} \underbrace{\Big(\big(f(i) - f(i-1)\big) - \big(f(i+1) - f(i)\big)\Big)}_{\geq\, 0 \text{ by concavity}} \cdot f^i(x) \\
&\qquad + \underbrace{\big(f(T) - f(T-1)\big)}_{\geq\, 0 \text{ by monotonicity}} \cdot f^T(x) - \big(f(1) - f(0)\big) \cdot \underbrace{f^0(x)}_{=0}
\end{aligned}$$ □

D.2 Lower Bound

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).*

Proof. We begin by describing the f-MaxCoverage instances (for an unspecified saturation function f) and will then analyze them. For $k = 1, 2, \ldots$, set $b := k + 2$ and create an instance as follows: Define the ground set as the disjoint union

$$G = B_0 \,\dot\cup\, B_1 \,\dot\cup \cdots \dot\cup\, B_k \,\dot\cup\, B_{k+1}$$

of $k + 2$ many *blocks*. For $i = 0, 1, \ldots, k + 1$, block B_i contains b^{2k+1-i} many elements. We will treat all ground elements within the same block as identical.

Next, we will define the collection $\mathcal{Z}$, which is partitioned into $k + 1$ many *types*:

$$\mathcal{Z} = T_0 \,\dot\cup\, T_1 \,\dot\cup \cdots \dot\cup\, T_k.$$

Each type T_i consists of all sets $S \subseteq B_i \cup B_{k+1}$ such that $|S| = b^k$ and $|S \cap B_i| = b^i$, that is, b^i elements can be freely chosen from block B_i and the remaining $b^k - b^i$ elements freely from B_{k+1}. Finally, we set $T := b^{2k+1}$, and which concludes the definition of the maximum coverage instance.

Next, we will establish lower bounds on which objective values can be achieved for certain objective functions $\hat{f}^r$. Consider some $i = 0, 1, \dots, k$, and consider a set Z that only contains sets of type T_i, in such a way that each ground element in block B_i is selected exactly $T\, b^i / b^{2k+1-i} = b^{2i}$ times.[11] For such a Z, it clearly holds that $\hat{f}^{b^{2i}}(Z) \geq T\, b^i = b^{2k+i+1}$.

Now, consider any solution Z. By the pigeon-hole principle, we can fix a $0 \leq i \leq k$ such that no more than $\frac{T}{k+1}$ elements of Z are of type T_i. We will show that $\hat{f}^{b^{2i}}(Z)$ must be substantially lower than the value computed in the last paragraph. We will separately bound the terms on the right-hand side of the following decomposition:

$$\hat{f}^{b^{2i}}(Z) = \sum_{g \in B_i} f^{b^{2i}}\left(\textstyle\sum_{S \in \mathcal{Z}: g \in S} Z(S)\right) + \sum_{g \in \bigcup_{0 \leq j < i} B_j} f^{b^{2i}}\left(\textstyle\sum_{S \in \mathcal{Z}: g \in S} Z(S)\right) + \sum_{g \in \bigcup_{i < j \leq k+1} B_j} f^{b^{2i}}\left(\textstyle\sum_{S \in \mathcal{Z}: g \in S} Z(S)\right)$$

Term "$g \in B_i$". Since, by assumption, at most $T/(k+1)$ elements of $\mathcal{Z}$ have type T_i, since each $S \in T_i$ contains b^i elements of B_i, and since no other type contains elements of B_i, this term is at most $\frac{T}{k+1}\, b^i = \frac{b^{2k+i+1}}{k+1}$.

Term "$g \in \bigcup_{0 \leq j < i} B_j$". Any $S \in \mathcal{Z}$ contains at most b^{i-1} elements in $\bigcup_{0 \leq j < i} B_j$. Hence, this term is at most $T\, b^{i-1} = \frac{b^{2k+i+1}}{b}$.

Term "$g \in \bigcup_{i < j \leq k+1} B_j$". Compared to B_i, these blocks contain relatively few elements:

$$\left| \bigcup_{i<j\leq k+1} B_j \right| = b^{2k-i} + b^{2k-i-1} + \dots + b^k = b^{2k+1-i} \cdot \left(\frac{1}{b} + \frac{1}{b^2} + \dots + \frac{1}{b^{k+1-i}} \right)$$
$$\leq b^{2k+1-i} \cdot \sum_{j=1}^{\infty} \frac{1}{b^j} = \frac{b^{2k+1-i}}{b-1}.$$

Since each ground element contributes at most b^{2i} to the objective, the third term is at most $b^{2i} \cdot \frac{b^{2k+1-i}}{b-1} = \frac{b^{2k+i+1}}{b-1}$.

Putting the above bounds together, it follows that

$$\hat{f}^{b^{2i}}(Z) \leq \left(\frac{1}{k+1} + \frac{1}{b} + \frac{1}{b-1} \right) \cdot b^{2k+i+1} \leq \frac{3}{k+1} \cdot b^{2k+i+1} \leq \frac{3}{k+1} \cdot \max_{Z'} \hat{f}^{b^{2i}}(Z').$$

We have thus established that no Z can simultaneously approximate all f^r with an approximation ratio better than $\frac{3}{k+1}$. The statement follows by observing that

$$\frac{3}{k+1} = \frac{6}{2k+2} < \frac{6}{2k+1} = \frac{6}{2k+1} \cdot \frac{\ln(k+2)}{\ln(k+2)} < \frac{6}{2k+1} \cdot \frac{\ln\left((2k+1)\ln(k+2)\right)}{\ln(k+2)}$$
$$= 6\, \frac{\ln\ln\left((k+2)^{2k+1}\right)}{\ln\left((k+2)^{2k+1}\right)} = 6 \frac{\ln\ln T}{\ln T}. \qquad \square$$

E Empirical Results

E.1 Data Preprocessing and Experimental Setup

As mentioned in the main text, our empirical analyses are based on seven datasets: `sf_a`, `sf_b`, `sf_c`, `sf_d`, `sf_e`, `sf_f`, and `hd`. All datasets whose name begins with "sf" are based on citizens' assemblies coorganized by the Sortition Foundation; the dataset `hd` to an assembly organized by the nonprofit Healthy Democracy. We use the Sortition Foundation datasets with the organization's permission; use of `hd` did not require permission, as noted by Flanigan et al. [10]. None of our datasets contain information that would allow to identify individuals; individuals are only labeled with their values for attributes (e.g., gender, age category, education level, ...). Though the datasets are not personally identifiable, our nonprofit partners have asked us not to identify the specific citizens' assemblies belonging to the datasets, nor to reveal which features were used, and we cannot publish our datasets. We are not aware of any alternative datasets on citizens' assemblies that would be more openly available. As we describe in the body, the assemblies for datasets `sf_a` through `sf_d` and `hd` were selected from the original lottery requirements using Panelot.org, for which the following random seeds were generated on that website and used: `sf_a`: 125131; `sf_b`: 228155; `sf_c`: 402199; `sf_d`: 722865; `hd`: 283988.

[11] One could choose such an Z by going over the elements of B_i in a round robin manner.

For each dataset with n participants, we chose $k = \lceil \sqrt{n} \rceil$ for all experiments. This definition was intuited from the number of groups (namely, $k = 7$) practitioners chose for the `sf_f` dataset, a 40-participant assembly coorganized by the Sortition Foundation, as well as from the typical size of tables. Our definition of k results in groups of size at most k, which appears to be reasonable for all seven datasets provided to us. A k value near to $\sqrt{n}$ also indicates there is a low minimum number of repeated links between two rounds, as per Appendix A.1.

Code for the baseline algorithm GROUPSELECT is available at https://github.com/sortitionfoundation/groupselect-app, and it is released under a GNU General Public License v3.0.

One important detail is that GROUPSELECT is not able to directly take in representativeness constraints as our algorithm. As we describe in E.2, it instead takes in a priority ordering over feature categories and then makes a best effort at representing these features similarly across tables. Here, we describe how we selected the feature ordering for our runs of GROUPSELECT: For the `sf_e` dataset coorganized by the Sortition Foundation, practitioners suggested the (anonymized) ordering $[e, g, b, f, a]$. The ordering for datasets `sf_a` through `sf_d` was then extrapolated to preserve the suggested ordering as much as possible (i.e., by considering the actual mapping between pre- and post-anonymized features of the `sf_e` dataset). Moreover, in instances where features were not identical across datasets, we substituted features that are often correlated with the feature present in `sf_e`; for example, the feature "socioeconomic status" might be substituted for "education." For datasets `sf_f` and `hd`, on the other hand, we did not have access to pre-anonymized features; in these cases, we simply ordered the features in the order they were given.

Finally, we define representativeness constraints for our algorithm based on solutions produced by GROUPSELECT. As described in E.2, GROUPSELECT produces solutions in which (for a given set of inputs) features are distributed across tables identically in all allocations. Thus, for each experiment, we only need to observe one of these allocations to determine the lower and upper quotas that GROUPSELECT satisfies for each feature in F. These quotas were then provided as input our algorithm to ensure that our solutions (at worst) match the representativeness constraints met by GROUPSELECT.

Across different plots, each data point in the plot is based on a single run of the algorithm. This makes sense for our algorithms (which, other than the not entirely predictable timing of the ILP solver) are essentially deterministic. GROUPSELECT is random in principle (we use the seed 1 throughout), but this randomness has very limited effect, as can be seen from the stability of GROUPSELECT's objective value across varying numbers of sessions (see Fig. 2 and Appendix E.3), and as we explain further below.

We run our experiments on Amazon AWS EC2, in a single "C5.4xlarge" instance running Ubuntu with a 3.6 GHz processor, 16 virtual CPUs, and 32 GB of RAM. We ran up to 15 experiments in parallel, limiting each to use Gurobi (version 9.5) in a single thread. The total running time for all our experiments on this machine was less than 24 hours.

E.2 Description of Baseline Algorithm (GROUPSELECT)

Inputs. For each assembly, GROUPSELECT takes as input a *participants' attributes table* like the one given below, where each row represents a participant, and each column represents an attribute such as "Label," "Gender," or "Geography." That is, GROUPSELECT assumes that the relevant features can be grouped into multiple partitions (by value of the attribute) of the panel.

Table 1: List of participant attributes in example instance

Label	Gender	Geography
$WR1$	Female	Rural
$WR2$	Female	Rural
$WR3$	Female	Rural
WC	Female	City
MR	Male	Rural
MC	Male	City

A *mode* can then be chosen for each attribute, where the modes are defined as follows:

- *Diversify:* For this attribute, the algorithm roughly tries to distribute its values evenly across all tables. The algorithm requires that at least one attribute be *diversified.*
- *Cluster:* For this attribute, the algorithm roughly tries to group participants with the same attribute values together. For example, if Gender is put in Cluster mode (i.e., Gender is *clustered*), then the algorithm tries to group all women together on the same tables, and all men together on the same tables. Furthermore, as will become important later, GROUPSELECT clusters a binary attribute by distributing all of the clustered participants among as few groups as possible.

Across all clustered and diversified attributes, GROUPSELECT requires an *attribute ordering* to be specified. This feature roughly allows organizers to rank the attributes in Cluster and Diversify mode, respectively, by importance so that higher-priority attributes are more likely to reflect the desired properties of the chosen mode. The algorithm also allows for *manual allocation* so that organizers have the option to assign participants to groups themselves.

Finally, GROUPSELECT takes as input the number of tables desired, the number of sessions, and a random seed.

Output. For each session, GROUPSELECT outputs the participants that have been assigned to each group. The output for two sessions of the assembly given in Table 1, for example, could look like the following:

Session	Table 1	Table 2
1	$WR1$, $WR2$, $WR3$	WC, MR, MC
2	$WR1$, $WR2$, MC	$WR3$, WC, MR

Algorithm Description. The following description simplifies GROUPSELECT by ignoring manual overrides and clustering constraints:

Algorithm 3: GROUPSELECT

```
1  function GROUPSELECT(N, num_allocs)
2      $partitions \leftarrow []$
3      repeat 100 times
4          append the result of GETRANDOMPARTITION() to $partitions$
5      $schedules \leftarrow \emptyset$
6      repeat 100 times
7          uniformly draw $T$ partitions from $partitions$, add result to $schedules$
8      return $\text{argmax}_{Z \in schedules}\, \hat{f}^1(Z)$
9  function GETRANDOMPARTITION()
10     create a list $ordered\_agents$ of the agents, sorted lexicographically by the attribute ordering, and ordering agents with
         identical features uniformly at random                                      ▷ See details.
11     $\Delta_1 \leftarrow \emptyset; \Delta_2 \leftarrow \emptyset; \ldots; \Delta_k \leftarrow \emptyset$
12     for $i$ in $ordered\_agents$ do
13         $\tau \leftarrow$ table where $i$'s features are least frequent              ▷ See details.
14         $\Delta_\tau \leftarrow \Delta_\tau \cup \{i\}$
15     return partition $\Delta_1 \dot\cup \Delta_2 \dot\cup \cdots \dot\cup \Delta_k$
```

Below are a few additional details, labeled by line number:

Line 10: To illustrate this lexicographic ordering, consider the assembly from Table 1 for the attribute ordering [Gender, Geography]. In this case, the lexicographic ordering of the possible combinations of attributes is

$$\big[[\text{Female}, \text{Rural}], [\text{Female}, \text{City}], [\text{Male}, \text{Rural}], [\text{Male}, \text{City}]\big],$$

and thus one possible orderings of agents that $ordered\_agents$ might take on is

$$[WR1, WR2, WR3, WC, MR, MC].$$

Note that the only degree of freedom in $ordered\_agents$ is the relative order in which agents with exactly the same combination of features appear (in this case, $WR1$, $WR2$, $WR3$). This degree of freedom is determined uniformly at random. Note that, if there are many different features, there are exponentially many combinations, which means that the randomness of GETRANDOMPARTITION might have little to no effect.

Line 13: Fixing agent i, the algorithm considers the first attribute a_1 in the attribute ordering. Agent i will be placed at a table τ where i's value for a_1 is so far represented the least frequently. For example, if a_2 equals Gender, and i is male, then i will be placed among the tables that so far have been allocated the lowest number of male agents.

In many cases, this criterion will not determine a unique table; in the example, multiple tables might have the same minimum number of male agents. In this case, ties are broken in favor of tables where i's value for the second attribute a_2 in the ordering is rarest; if ties remain, the tables are successively refined by the third, fourth, etc. attribute of the ordering.

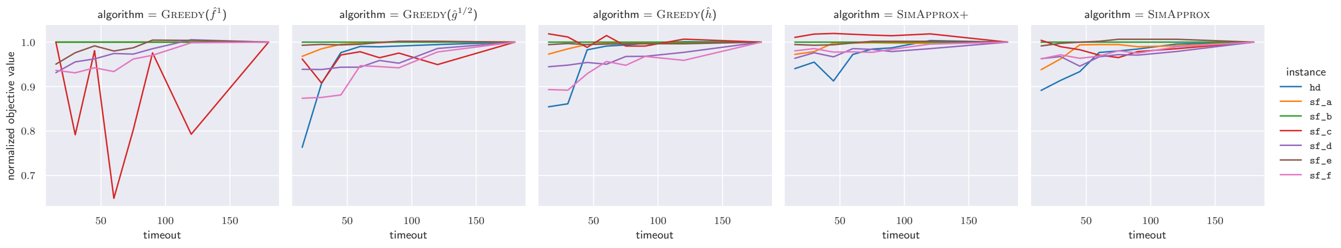
Note that, regardless of the randomness, the partition produced by GETRANDOMPARTITION is deterministic up to permutations between agents with the exact same combination of features.

This also explains why GROUPSELECT is so far from optimal in optimizing its objective $\hat{f}^1$. To summarize Algorithm 3, GROUPSELECT algorithm proceeds in two phases: First, it creates a pool of partitions. Second, it repeatedly draws random schedules from this pool (by uniformly sampling T partitions without replacement), computes their objective value, and returns the best schedule across these samples.

GROUPSELECT's main problem is that the pool generated in step 1 lacks diversity. This is understandable given that practitioners need to generate many partitions that all satisfy the same representativeness constraints; since finding a partition for given representativeness constraints is NP-hard (Appendix C), no simple and efficient algorithms for this problem are available. The practitioners' approach to this problem is equivalent to generating a single partition with (heuristically) good representativeness, and then permuting "clones", i.e., panel members that have *exactly* the same vector of features. While this approach makes sense for few numbers of features, it severely restricts the possible partitions for the numbers of features that practitioners typically use the algorithm for. For example, if two agents do not have "clones" on the same table in the original single partition, they can never meet. In typical use cases, "clones" seem to have been frequent enough that the Sortition Foundation did not realize this severe limitation. But it clearly shows in comparison with the much larger degree of meetings enabled by our algorithms, which is possible since integer linear programming allows to efficiently optimize over the full space of representative partitions.

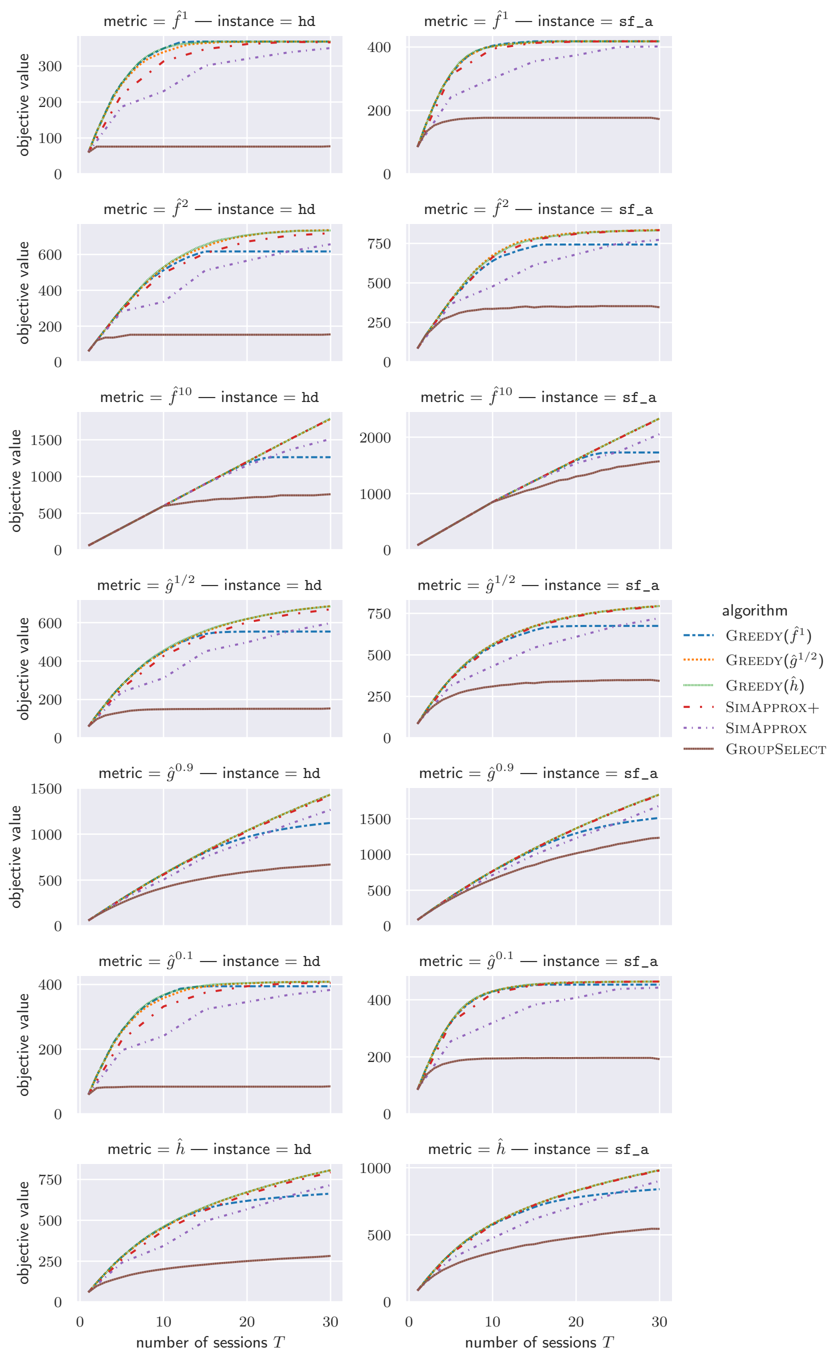
E.3 Additional Experiments

First, we will investigate the impact of different ILP timeouts on our greedy algorithms. Below we show one plot for each out of five algorithms. In each case, the x axis indicates the timeout per ILP call in seconds. The y axis indicates the sum of obtained marginals over 20 sessions, normalized at the maximum timeout of 180 seconds for ease of readability. (For SIMAPPROX and SIMAPPROX+, we sum the marginals for the different functions being optimized.)

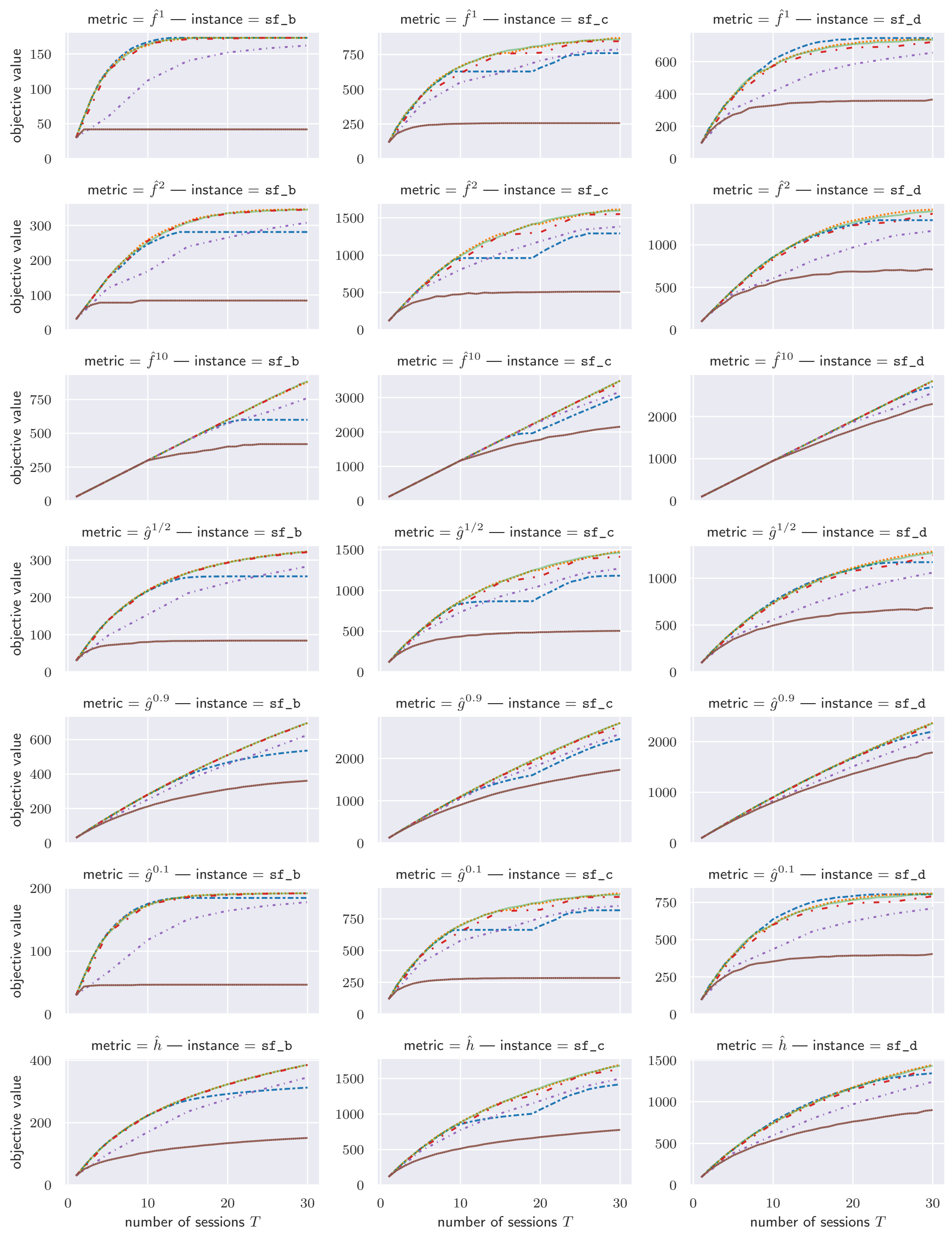


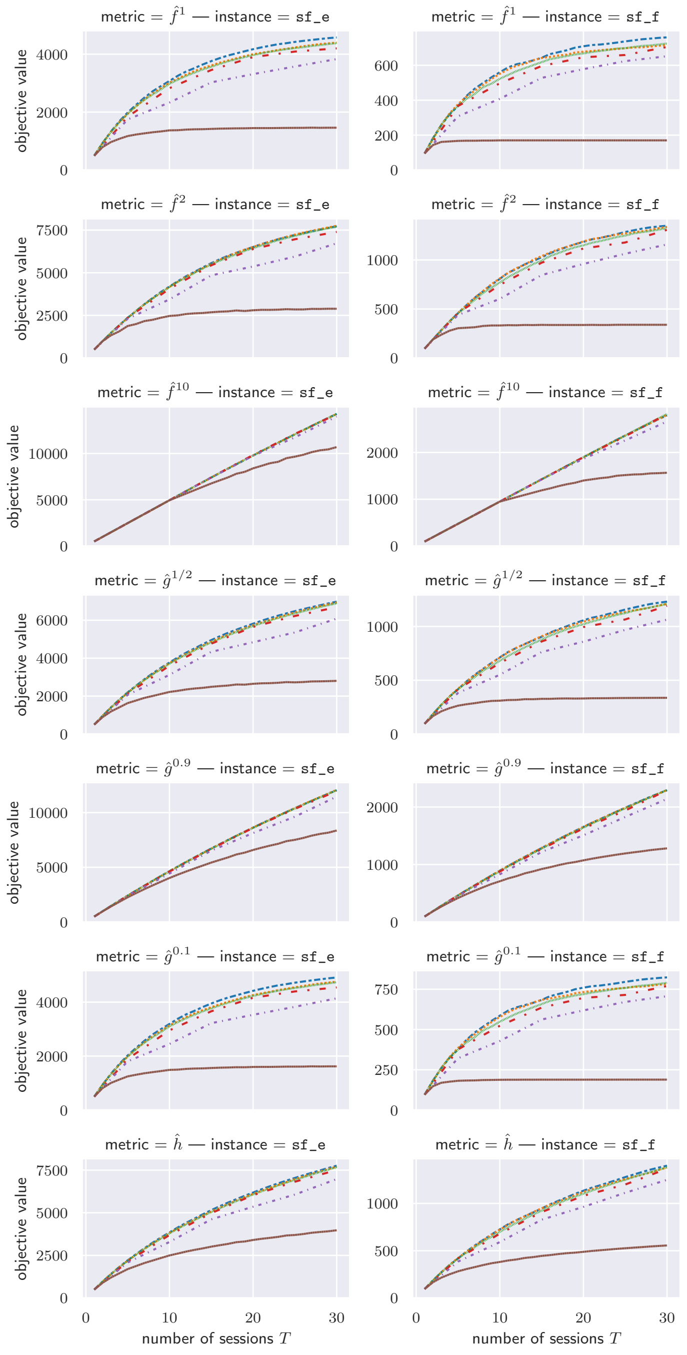
As one expects, the general trend across most lines is that larger timeouts lead to better optimization. This is, however, not always the case; in particular, instance `sf_c` exhibits nonmonotone behavior where shorter optimization times are more effective, and this is the case to an extreme degree when we greedily optimized the objective $\hat{f}^1$. Other than in these extreme cases, a timeout of 60 seconds achieves at least around 95% of the sum of marginals of a timeout of 180 seconds.

On the next pages, we will evaluate the schedules produced by six algorithms: greedy algorithms optimizing $\hat{f}^1, \hat{g}^{1/2}, \hat{h}$, SIMAPPROX+, SIMAPPROX, and GROUPSELECT. We will evaluate all algorithms on all instances and on numbers of sessions that vary between 1 and 30. Most notably, we evaluate the quality of schedules in terms of a whole range of objectives, including objectives $\hat{f}^r$ for small and large r, geometric objectives for small and large bases, and the harmonic objective. Across all these experiments, we can see that the schedules obtained by greedily optimizing either $\hat{g}^{1/2}$ or $\hat{h}$ lead to the best objective values or very close to the best (in particular, greedily optimizing $\hat{f}^1$ performs a bit better when evaluated on $\hat{f}^1$). The GROUPSELECT baseline is uniformly beaten by all other algorithms; greedily optimizing $\hat{f}^1$ tends to fall back for larger numbers of sessions T.



metric = $\hat{f}^1$ — instance = hd
metric = $\hat{f}^1$ — instance = sf_a
metric = $\hat{f}^2$ — instance = hd
metric = $\hat{f}^2$ — instance = sf_a
metric = $\hat{f}^{10}$ — instance = hd
metric = $\hat{f}^{10}$ — instance = sf_a
metric = $\hat{g}^{1/2}$ — instance = hd
metric = $\hat{g}^{1/2}$ — instance = sf_a
metric = $\hat{g}^{0.9}$ — instance = hd
metric = $\hat{g}^{0.9}$ — instance = sf_a
metric = $\hat{g}^{0.1}$ — instance = hd
metric = $\hat{g}^{0.1}$ — instance = sf_a
metric = $\hat{h}$ — instance = hd
metric = $\hat{h}$ — instance = sf_a
objective value
number of sessions T
algorithm
Greedy($\hat{f}^1$)
Greedy($\hat{g}^{1/2}$)
Greedy($\hat{h}$)
SimApprox+
SimApprox
GroupSelect


metric = $\hat{f}^1$ — instance = sf_b
metric = $\hat{f}^1$ — instance = sf_c
metric = $\hat{f}^1$ — instance = sf_d
metric = $\hat{f}^2$ — instance = sf_b
metric = $\hat{f}^2$ — instance = sf_c
metric = $\hat{f}^2$ — instance = sf_d
metric = $\hat{f}^{10}$ — instance = sf_b
metric = $\hat{f}^{10}$ — instance = sf_c
metric = $\hat{f}^{10}$ — instance = sf_d
metric = $\hat{g}^{1/2}$ — instance = sf_b
metric = $\hat{g}^{1/2}$ — instance = sf_c
metric = $\hat{g}^{1/2}$ — instance = sf_d
metric = $\hat{g}^{0.9}$ — instance = sf_b
metric = $\hat{g}^{0.9}$ — instance = sf_c
metric = $\hat{g}^{0.9}$ — instance = sf_d
metric = $\hat{g}^{0.1}$ — instance = sf_b
metric = $\hat{g}^{0.1}$ — instance = sf_c
metric = $\hat{g}^{0.1}$ — instance = sf_d
metric = $\hat{h}$ — instance = sf_b
metric = $\hat{h}$ — instance = sf_c
metric = $\hat{h}$ — instance = sf_d
objective value
number of sessions T
0
10
20
30



Recall that, in all experiments so far, the representativeness constraints were derived from GROUPSELECT, and as a result were probably relatively easy to satisfy. In this section, we repeat some of our experiments with substantially tighter representativeness constraints. Specifically, for each instance, we use an ILP to compute a strengthening of the constraints derived before, namely one that, summed up over all features, decreased the gap between lower and upper quotas by the largest amount. The resulting constraints are substantially tighter: the gap between upper quota and lower quota, averaged over features, shrunk by an amount of at least 1.9 in every instance (median: 2.5, maximum: 5.4). Given that these constraints are now maximally tight, we would expect the ILP solver to face a more challenging task.

As expected, the ILP solver requires more time in the sense that, in some of the iterations of the greedy algorithm or of SIMAPPROX, the ILP solver is not able to find any feasible solution within 120 seconds (we then let the ILP solver run until some feasible solution is found). For the objective $\hat{g}^{1/2}$, for example, 3 of our 7 instances still compute 30 sessions in less than $30 \cdot 120$ seconds, 3 other instances exceed this amount by 7%, 26%, and 37%. The clear outlier is the instance `sf_e` (which has nearly 3 times more panel members than the second-largest instance), which requires around 3 hours for 30 sessions — 3 times as much time as implied by our timeouts.

To our surprise, tightening the representativeness constraints did not have a clear negative effect on the approximation certificates in the equivalent of Fig. 1, shown in Fig. 3. Instead, some guarantees became a bit better and some a bit worse, leading to a similar overall picture. Though this comparison is not quite on equal terms since the experiment with tightened constraints used greater running time, we would have expected a clear negative effect since we expected the ILP solver to get less far in bounding the optimality gap.

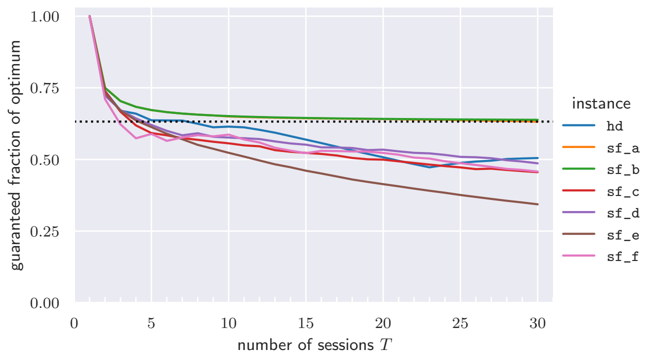
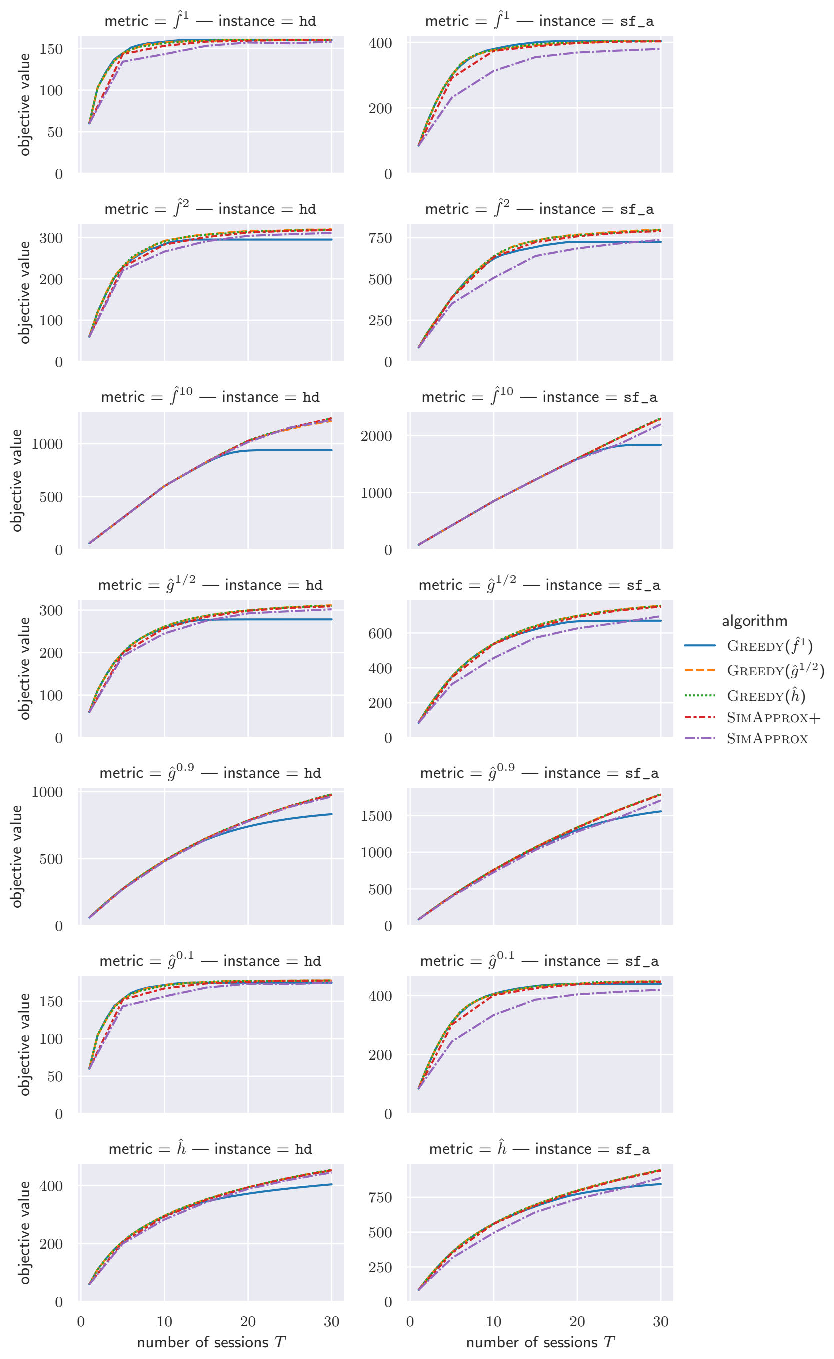
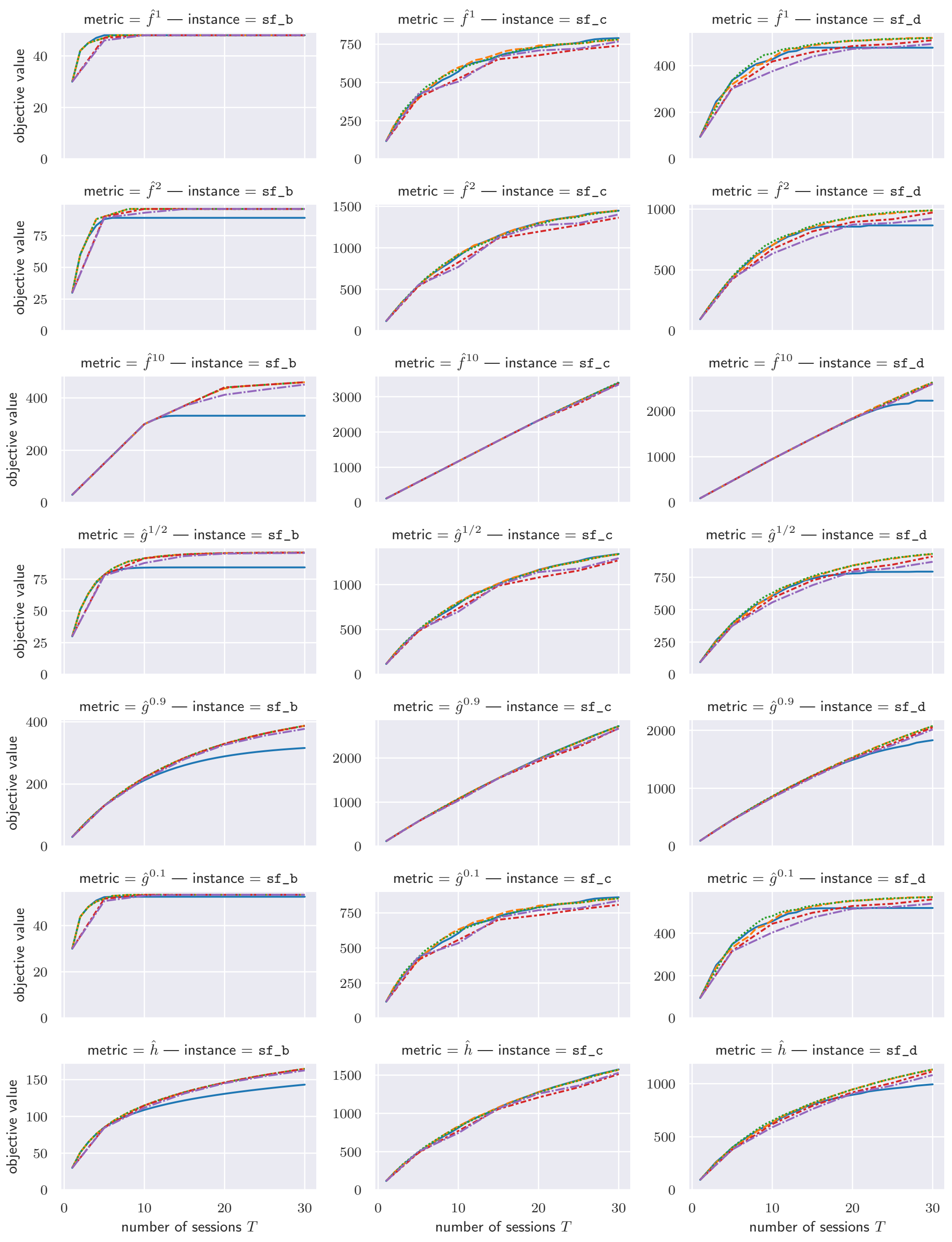


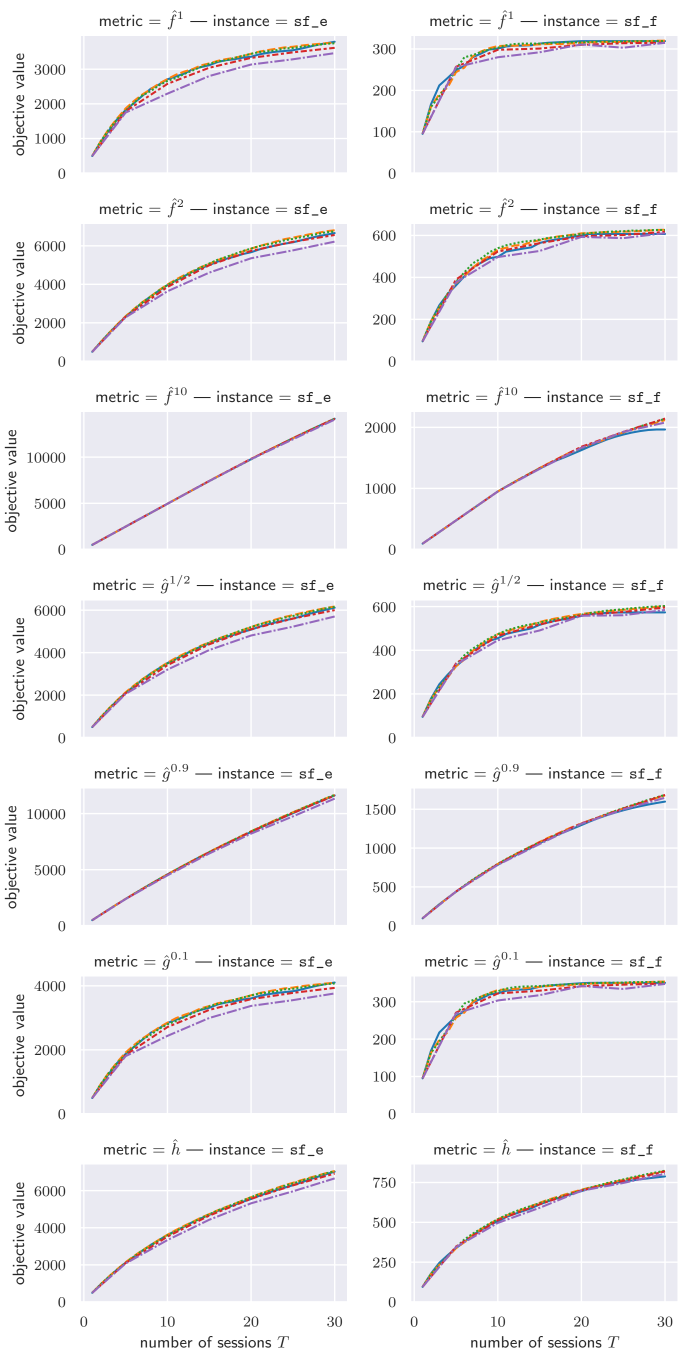
Figure 3: Approximation certificates for the greedy algorithm on $\hat{g}^{1/2}$, guaranteeing near-optimality, but for the maximally tight quotas. The dashed line marks $1 - 1/e$.

A final dimension of comparison are the objective values themselves. For this, we repeat the plots from the beginning of this section below, showing the objective value attained as a function of T, for the same combinations of objective metrics and instances. In roughly half of the instances, objective values are substantially lower than without constraints; for example, the number of unique meetings might be only half as large at $T = 30$ sessions with tightened constraints than with the original constraints. In the other half of instances, objective values are barely reduced. Since we chose *maximally tight* constraints, it is entirely plausible that some agents are bound to never appear on the same table (the easiest case would be an upper bound of 1 on a feature, which implies that two agents with this feature never meet), or that the lack of flexibility in the constraints means that a much smaller number of new meetings can be achieved per session. The tightened constraints seem to have such effects for some instances; on other instances, the tightened constraints seem to leave similar flexibility as the original constraints.



metric = $\hat{f}^1$ — instance = hd
metric = $\hat{f}^1$ — instance = sf_a
metric = $\hat{f}^2$ — instance = hd
metric = $\hat{f}^2$ — instance = sf_a
metric = $\hat{f}^{10}$ — instance = hd
metric = $\hat{f}^{10}$ — instance = sf_a
metric = $\hat{g}^{1/2}$ — instance = hd
metric = $\hat{g}^{1/2}$ — instance = sf_a
metric = $\hat{g}^{0.9}$ — instance = hd
metric = $\hat{g}^{0.9}$ — instance = sf_a
metric = $\hat{g}^{0.1}$ — instance = hd
metric = $\hat{g}^{0.1}$ — instance = sf_a
metric = $\hat{h}$ — instance = hd
metric = $\hat{h}$ — instance = sf_a
objective value
number of sessions T
algorithm
Greedy($\hat{f}^1$)
Greedy($\hat{g}^{1/2}$)
Greedy($\hat{h}$)
SimApprox+
SimApprox


metric = $\hat{f}^1$ — instance = sf_b
metric = $\hat{f}^1$ — instance = sf_c
metric = $\hat{f}^1$ — instance = sf_d
metric = $\hat{f}^2$ — instance = sf_b
metric = $\hat{f}^2$ — instance = sf_c
metric = $\hat{f}^2$ — instance = sf_d
metric = $\hat{f}^{10}$ — instance = sf_b
metric = $\hat{f}^{10}$ — instance = sf_c
metric = $\hat{f}^{10}$ — instance = sf_d
metric = $\hat{g}^{1/2}$ — instance = sf_b
metric = $\hat{g}^{1/2}$ — instance = sf_c
metric = $\hat{g}^{1/2}$ — instance = sf_d
metric = $\hat{g}^{0.9}$ — instance = sf_b
metric = $\hat{g}^{0.9}$ — instance = sf_c
metric = $\hat{g}^{0.9}$ — instance = sf_d
metric = $\hat{g}^{0.1}$ — instance = sf_b
metric = $\hat{g}^{0.1}$ — instance = sf_c
metric = $\hat{g}^{0.1}$ — instance = sf_d
metric = $\hat{h}$ — instance = sf_b
metric = $\hat{h}$ — instance = sf_c
metric = $\hat{h}$ — instance = sf_d
objective value
number of sessions T


metric = $\hat{f}^1$ — instance = sf_e
metric = $\hat{f}^1$ — instance = sf_f
metric = $\hat{f}^2$ — instance = sf_e
metric = $\hat{f}^2$ — instance = sf_f
metric = $\hat{f}^{10}$ — instance = sf_e
metric = $\hat{f}^{10}$ — instance = sf_f
metric = $\hat{g}^{1/2}$ — instance = sf_e
metric = $\hat{g}^{1/2}$ — instance = sf_f
metric = $\hat{g}^{0.9}$ — instance = sf_e
metric = $\hat{g}^{0.9}$ — instance = sf_f
metric = $\hat{g}^{0.1}$ — instance = sf_e
metric = $\hat{g}^{0.1}$ — instance = sf_f
metric = $\hat{h}$ — instance = sf_e
metric = $\hat{h}$ — instance = sf_f
objective value
number of sessions T