Large Language Models for Designing Participatory Budgeting Rules

Nguyen Thach
University of Nebraska-Lincoln
Lincoln, NE, USA
nate.thach@huskers.unl.edu

Xingchen Sha
Northwestern University
Evanston, IL, USA
xingchen.sha@u.northwestern.edu

Hau Chan
University of Nebraska-Lincoln
Lincoln, NE, USA
hchan3@unl.edu

ABSTRACT

Participatory budgeting (PB) is a democratic paradigm for deciding the funding of public projects given the residents' preferences, which has been adopted in numerous cities across the world. The main focus of PB is designing *rules*, functions that return feasible budget allocations for a set of projects subject to some budget constraint. Designing PB rules that optimize both utility and fairness objectives based on agent preferences had been challenging due to the extensive domain knowledge required and the proven trade-off between the two notions. Recently, large language models (LLMs) have been increasingly employed for automated algorithmic design. Given the resemblance of PB rules to algorithms for classical knapsack problems, in this paper, we introduce a novel framework, named LLMRule, that addresses the limitations of existing works by incorporating LLMs into an evolutionary search procedure for automating the design of PB rules. Our experimental results, evaluated on more than 600 real-world PB instances obtained from the U.S., Canada, Poland, and the Netherlands with different representations of agent preferences, demonstrate that the LLM-generated rules generally outperform existing handcrafted rules in terms of overall utility while still maintaining a similar degree of fairness.

KEYWORDS

Participatory Budgeting, Fairness, Large Language Models

ACM Reference Format:
Nguyen Thach, Xingchen Sha, and Hau Chan. 2026. Large Language Models for Designing Participatory Budgeting Rules. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/LGEP3560

1 INTRODUCTION

Participatory budgeting (PB) has been widely adopted in many cities across the world, e.g., from the U.S. and the U.K. to regions as diverse as Brazil, Spain, and South Africa [18, 34, 38]. Originally introduced for deciding the distribution of public funding (or budget) to public projects for residents given their preferences in a democratic manner [10], PB can also be applied to electing a representative committee (e.g., parliament) for voters [16], selecting validators in consensus protocols (e.g., blockchain) [11], and locating public facilities [39]. In the voting stage of a typical PB process[1], the social planner seeks to allocate a given budget to a set of projects (e.g., building new parks and clinics, installing public toilets, and improving street conditions) for serving agents from a region by considering their preferences on the projects and optimizing the overall utility based on agent preferences. In addition to utility considerations, the notion of *fairness* (e.g., *proportionality*), in which each agent should have roughly equal influence on the allocation outcome, has been emphasized to align with the spirit of democratic decision making [8, 29, 30].

Existing Efforts. Through immense domain expertise, almost all existing studies on PB aimed to manually design the following two types of PB rules and their corresponding implementation algorithms: (i) *utilitarian PB rules*, which return optimal or nearly optimal budget allocations subject to some utility objective [6, 40, 41], and (ii) *proportional PB rules*, which find budget allocations provably satisfying certain fairness properties [4, 29, 30]. Unfortunately, it has been shown that a trade-off exists between notions of utility and fairness [14, 26]—PB rules that focus on utility maximization (resp. fairness) are far from being fair (resp. optimal) with respect to the budget allocations. In fact, proportional rules may return budget allocations with severely compromised utility objective values even for relaxed definitions of fairness [33]. For example, the Method of Equal Shares (MES) [30], one of the most recognized proportional rules, finds budget allocations that satisfy strong fairness properties (e.g., EJR in special settings [30]) but typically does not spend the entire budget, which naturally results in a significant drop in the overall utility. Thus, in this paper, we strive to address the following question.

> *Can we design PB rules focusing on budget allocations that are empirically optimal for a given utility objective while being approximately fair?*

Our Approach and Associated Challenges. Large language models (LLMs) have shown remarkable capabilities in algorithmic design [28] and programming [23]. Notably, integrating LLMs into an evolutionary search procedure to iteratively refine heuristic algorithms has yielded impressive results in various combinatorial optimization problems such as traveling salesman problems and knapsack problems [27, 35, 45]. The idea from these works is to maintain a set of heuristics (coded in some programming language) with good optimization performance on an evaluation dataset of problem instances and iteratively prompt LLMs to generate new heuristics using existing ones as references. Because the problem of maximizing the utility objective subject to the budget constraint resembles classical knapsack problems [5], leveraging LLMs to design PB

This work is licensed under a Creative Commons Attribution International 4.0 License.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/LGEP3560

[1] The computational social choice literature has been focusing on the voting stage of PB extensively [33], which is typically preceded by a shortlisting stage [32].

rules via the stated idea appears to be a plausible approach to our question.

There are, however, three challenges to address when employing existing LLM-based methods. (Challenge I) These methods only consider single-objective problems[2], whereas we are optimizing for both utility and fairness objectives. (Challenge II) Previous LLM-based methods require a fine-grained performance measure that can capture small improvements during search for ensuring convergence at desirable algorithms [35]. In our case, while most utility objectives can provide a rich enough performance measure, since equivalent objectives for quantifying how fair the allocations returned from PB rules are (with respect to some fairness definition) have yet to exist, it is unclear how fairness properties can be effectively enforced for LLM-generated PB rules. (Challenge III) Existing LLM-based methods assume the availability of an efficient evaluation procedure such that the performance of the generated algorithms can be quickly computed. Unfortunately, it is well-documented that verifying whether a given budget allocation satisfies standard fairness properties is intractable (i.e., coNP-complete [2, 3]). Therefore, even when a suitable fairness objective addressing Challenge II exists, using it to evaluate the "fairness" of LLM-generated PB rules by computing its value for all training problem instances would naturally be intractable as well, rendering the evaluation procedure prohibitively expensive.

Contributions. We aim to address the stated three challenges and thereby enable the use of LLMs for designing PB rules. More specifically, our contributions are as follows:

- In response to Challenge II, we introduce a novel fairness objective inspired by Strong-EJR, a desirable fairness property in PB literature [12], to which we refer as *Strong-EJR approximation*, to quantify the degree of fairness of the budget allocations returned from a given LLM-generated PB rule. To account for Challenge III, we define this objective based on our provably equivalent definition of Strong-EJR, the verification of which can be done with reduced complexity.
- To further address Challenge III, we devise techniques to speed up the computation of Strong-EJR approximation, resulting in a linear-time complexity with respect to the number of agents that is typically large in practice.
- Equipped with the aforementioned tools, we develop the first LLM-based framework, named LLMRule, for automating the design of heuristic PB rules that are essentially fair (i.e., yield high values of Strong-EJR approximation) without significant compromise on the overall utility. We tackle Challenge I in the process by introducing a penalty term based on Strong-EJR approximation in the fitness function, originally defined with respect to *utilitarian social welfare* [17] as the utility objective, from which the performance of LLM-designed PB rules is measured. Additionally, LLMRule requires minimal domain knowledge and hyperparameter tuning (e.g., prompt design), which alleviates the manual efforts involved in contemporary PB studies.
- Using more than 600 real-world PB instances across Europe and North America, we empirically demonstrate that LLMRule outperforms existing proportional rules in terms of utilitarian social welfare while maintaining a similar degree of fairness with respect to Strong-EJR approximation on classical settings [8, 29] where agent preferences are represented as *approval ballots* (i.e., each agent submit a subset of projects they approve of) and the utility function is defined as (i) the number and (ii) the cost of selected and approved projects. Furthermore, the good performance of LLMRule also translates to more general settings with *cardinal ballots* (i.e., each agent submits a score for each project).

Outline. In the remainder of this paper, we provide the preliminaries for our considered PB problems in Section 2. Section 3 details our proposed LLMRule framework. We validate its efficacy in Section 4. We discuss related work in Section 5 and conclude our work in Section 6. From our full paper[3], readers can find complete related work in Appendix A, additional theoretical results in Appendix B, complete implementation details in Appendices C–F (which include all missing figures and listings), additional experiments involving synthetic data in Appendix G, and limitations and future work in Appendix H.

2 PRELIMINARIES

We present the classical settings of PB that are most studied in the literature [33]. Let $I = \langle \mathcal{P}, c, b \rangle$ be a PB instance, where $\mathcal{P} = \{p_1, \ldots, p_m\}$ is the set of m projects, $c : \mathcal{P} \to \mathbb{R}_{>0}$ is the cost function that assigns a cost $c(p) \in \mathbb{R}_{>0}$ to every project $p \in \mathcal{P}$, and $b \in \mathbb{R}_{>0}$ is the budget. Note that from a broader perspective, I represents the voting stage of a PB process[1]. For any subset of projects $P \subseteq \mathcal{P}$, we denote $c(P) = \sum_{p \in P} c(p)$ as its total cost.

Let $\mathcal{N} = \{1, \ldots, n\}$ be the set of n agents or voters involved in I. Each agent is associated with a *ballot* that represents their preferences over the projects in $\mathcal{P}$. In the most general setting, the agents are asked to submit a score for all projects, known as *cardinal ballots*. A cardinal ballot for an agent $i \in \mathcal{N}$ is defined as $A_i : \mathcal{P} \to \mathbb{R}_{\geq 0}$. Because collecting agent preferences from this type of ballot can be laborious especially when m is large [22], most real-life PB processes use *approval ballots* [5], where agents are only asked to submit a subset of projects they approve of. In other words, it is a special case of cardinal ballots: an approval ballot for i is defined as $A_i : \mathcal{P} \to \{0, 1\}$, where for any $p \in \mathcal{P}$, $A_i(p) = 1$ indicates that agent i approves of project p and $A_i(p) = 0$ otherwise. The vector $\boldsymbol{A} = (A_1, \ldots, A_n)$ forms a *profile* of ballots.

Based on this profile, the social planner seeks to find an allocation outcome of I, which is a budget allocation $\pi \subseteq \mathcal{P}$ such that $c(\pi) \leq b$.[4] A PB rule f is a function that takes an instance $I = \langle \mathcal{P}, c, b \rangle$ and a profile $\boldsymbol{A}$ as input and outputs a budget allocation π. Note that in general, a PB rule returns a subset of all feasible budget allocations i.e., $f(I, \boldsymbol{A}) \subseteq \{\pi \subseteq \mathcal{P} \mid c(\pi) \leq b\}$. We focus on *resolute* rules that always return a single budget allocation (via some tie-breaking strategy) and hence, with a slight abuse of notation, denote the output $\{\pi\}$ of a resolute rule by just π. Following existing utility

[2]By objective, we mean an explicit function inherent to the problem of interest, hence works (e.g., [43]) that consider empirical runtime as additional objectives do not apply.

[3]https://arxiv.org/abs/2602.09349

[4]In this paper, we focus on indivisible or discrete PB in which projects can only be fully funded or not at all.

maximization studies [6, 41], our goal is to find a resolute PB rule f that maximizes the overall utility of the agents, or formally the *utilitarian social welfare* [17]. For cardinal ballots, this objective is defined as

$$\omega(I, \boldsymbol{A}, f) = \sum_{i \in N} \sum_{p \in f(I, \boldsymbol{A})} A_i(p) \tag{1}$$

where we follow the standard *additivity* assumption [30] that equates the score of a budget allocation for an agent to the sum of the scores of the projects it contains. For approval ballots, the objective is

$$\omega[sat](I, \boldsymbol{A}, f) = \sum_{i \in N} sat_i(f(I, \boldsymbol{A})) \tag{2}$$

where $sat_i(\pi) = sat(\{p \in \pi \mid A_i(p) = 1\})$ and $sat : 2^{\mathcal{P}} \to \mathbb{R}_{\geq 0}$ is a satisfaction function for translating a budget allocation into a satisfaction level for the agents given their approval ballots [8]. We focus on two standard (additive) satisfaction functions [41]:

- Cardinality satisfaction function: $sat^{card}(P) = |P|$
- Cost satisfaction function: $sat^{cost}(P) = c(P)$.

At the same time, we also want f to be *fair*, i.e., satisfies some fairness property $\mathcal{X}$, in the sense that for every instance I and profile $\boldsymbol{A}$, the allocation outcome of the (resolute) rule $f(I, \boldsymbol{A})$ is guaranteed to satisfy $\mathcal{X}$. Formalizing fairness properties requires the following definitions of cohesive groups [31]. We start with cardinal ballots.

Definition 1 ((α, P)-Cohesive Groups). *Given an instance $I = \langle \mathcal{P}, c, b \rangle$ and a profile $\boldsymbol{A}$ of cardinal ballots, a non-empty group of agents $N \subseteq \mathcal{N}$ is said to be (α, P)-cohesive, for a function $\alpha : \mathcal{P} \to \mathbb{R}_{\geq 0}$ and a set of projects $P \subseteq \mathcal{P}$, if the following two conditions are satisfied:*

- *$\alpha(p) \leq A_i(p)$ for all $i \in N$ and $p \in P$, that is, α is lower-bounding the score of the agents in N;*
- *$\frac{|N|}{n} \cdot b \geq c(P)$, that is, N's share of the budget is enough to afford P.*

For simplicity, we restrict α to be the minimum positive score submitted by any agent in N for project $p \in \mathcal{P}$.[5] Similarly, cohesive groups for approval ballots are defined as follows.

Definition 2 (P-Cohesive Groups). *Given an instance $I = \langle \mathcal{P}, c, b \rangle$ and a profile $\boldsymbol{A}$ of approval ballots, a non-empty group of agents $N \subseteq \mathcal{N}$ is said to be P-cohesive, for a set of projects $P \subseteq \mathcal{P}$, if the following two conditions are satisfied:*

- *$A_i(p) = 1$ for all $i \in N$ and $p \in P$, that is, every agent in N approves all projects in P;*
- *$\frac{|N|}{n} \cdot b \geq c(P)$, that is, N's share of the budget is enough to afford P.*

Ideally, all agents in N should deserve at least as much satisfaction as they all agree P would offer them (captured by $\boldsymbol{A}$). The following fairness property, termed *strong extended justified representation* (Strong-EJR) [2, 12], captures this idea.

Definition 3 (Strong-EJR for Cardinal Ballots). *Given an instance $I = \langle \mathcal{P}, c, b \rangle$ and a profile $\boldsymbol{A}$ of cardinal ballots, a budget allocation π is said to satisfy strong extended justified representation (Strong-EJR) if for all $P \subseteq \mathcal{P}$ and all (α, P)-cohesive groups N, and all $i^\star \in N$, we have:*

$$\sum_{p \in \pi} A_{i^\star}(p) \geq \sum_{p \in P} \min_{i \in N} A_i(p).$$

Definition 4 (Strong-EJR for Approval Ballots). *Given an instance $I = \langle \mathcal{P}, c, b \rangle$, a profile $\boldsymbol{A}$ of approval ballots, and a satisfaction function sat, a budget allocation π is said to satisfy Strong-EJR[sat] if for all $P \subseteq \mathcal{P}$ and all P-cohesive groups N, we have $sat_i(\pi) \geq sat_i(P)$ for all agents $i \in N$.*

While appealing, budget allocations satisfying Strong-EJR may not exist for a given I in general [2, 12]. Hence, several weaker fairness properties exist in the literature, notably extended justified representation (EJR) [2, 30], which requires only one member of each cohesive group to enjoy the deserved satisfaction.

Definition 5 (EJR for Cardinal Ballots). *Given an instance $I = \langle \mathcal{P}, c, b \rangle$ and a profile $\boldsymbol{A}$ of cardinal ballots, a budget allocation π is said to satisfy extended justified representation (EJR) if for all $P \subseteq \mathcal{P}$ and all (α, P)-cohesive groups N, there exists $i^\star \in N$ such that:*

$$\sum_{p \in \pi} A_{i^\star}(p) \geq \sum_{p \in P} \min_{i \in N} A_i(p).$$

Definition 6 (EJR for Approval Ballots). *Given an instance $I = \langle \mathcal{P}, c, b \rangle$, a profile $\boldsymbol{A}$ of approval ballots, and a satisfaction function sat, a budget allocation π is said to satisfy EJR[sat] if for all $P \subseteq \mathcal{P}$ and all P-cohesive groups N, there exists $i \in N$ such that $sat_i(\pi) \geq sat_i(P)$.*

3 THE LLMRULE FRAMEWORK

We start this section with an overview of our proposed framework, which is inspired by the popular EoH method [27] in automated algorithmic design. LLMRule maintains a population of h heuristic rules, denoted as $R = \{f_1, \ldots, f_h\}$. It adopts an evolutionary search procedure, iteratively searching for rules that yield better trade-offs between utility and fairness objectives. Each rule $f_j \in R$ is evaluated on a set of problem instances and assigned a fitness value $q(f_j)$ that indicates its probability of being selected as reference for generating new rules. The remainder of this section is organized as follows: Section 3.1 provides our provably equivalent definition of Strong-EJR for reducing the complexity of its verification (Challenge III); using this definition, Section 3.2 introduces Strong-EJR approximation, a novel fairness objective used for informing the design of empirically fair utilitarian PB rules (Challenge II); Section 3.3 describes our techniques for efficiently computing Strong-EJR approximation, which in turn expedites the training of LLMRule (Challenge III); Section 3.4 elaborates on LLMRule, in which the fitness function q for evaluating PB rules in terms of both objectives is defined (Challenge I).

3.1 Efficient Strong-EJR Verification

Based on Definitions 3 and 4, naïvely verifying Strong-EJR for a given π requires inspecting all possible combinations of projects P and groups of agents N pertaining to P, followed by comparing the satisfaction between P and π for each agent in N. Clearly, this exhaustive process takes $O(2^{m+n} \times n)$, which is intractable. In the

[5] We only consider nontrivial PB instances with at least one cohesive group, $\max_{i \in N} A_i(p) > 0$ for any $p \in \mathcal{P}$, and $\sum_{p \in \mathcal{P}} c(p) > b$.

following, we lay the groundwork for reducing this complexity. For brevity, we use P-cohesive groups to also refer to (α, P)-cohesive groups for cardinal ballots when the context is clear, where α is fixed to the smallest positive score agents can assign to projects.

Definition 7 (Maximal P-cohesive group). *A P-cohesive group is maximal, denoted as $N^{\max}$, if $N^{\max} \supseteq N$ for any other P-cohesive group N.*

Theorem 1. *Given $P \subseteq \mathcal{P}$ and two P-cohesive groups N and N' where $N' \subseteq N$, if $sat_i(\pi) \geq sat_i(P)$ for all $i \in N$, then $sat_{i'}(\pi) \geq sat_{i'}(P)$ for all $i' \in N'$.*
Likewise, if $\sum_{p\in\pi} A_{i^\star}(p) \geq \sum_{p\in P} \min_{i\in N} A_i(p)$ for all $i^\star \in N$, then $\sum_{p\in\pi} A_{i'}(p) \geq \sum_{p\in P} \min_{i\in N'} A_i(p)$ for all $i' \in N'$.

Proof. By contradiction, assume there exists $i' \in N'$ such that $sat_{i'}(\pi) < sat_{i'}(P)$. Since $N' \subseteq N$, it follows that there exists $i \in N$ such that $sat_i(\pi) < sat_i(P)$, which is a contradiction. The same applies to the statement for cardinal ballots. □

Theorem 1 implies that when verifying Strong-EJR, only checking the maximal P-cohesive group $N^{\max}$ for a given $P \subseteq \mathcal{P}$ is sufficient, which leads us to the following revision of Definitions 3 and 4.

Definition 8 (Strong-EJR for Cardinal Ballots (Updated)). *Given an instance $I = \langle \mathcal{P}, c, b\rangle$ and a profile $\boldsymbol{A}$ of cardinal ballots, a budget allocation π is said to satisfy strong extended justified representation (Strong-EJR) if for all $P \subseteq \mathcal{P}$ and all $i^\star \in N^{\max}$, the corresponding **maximal** (α, P)-cohesive group, we have:*

$$\sum_{p\in\pi} A_{i^\star}(p) \geq \sum_{p\in P} \min_{i\in N} A_i(p).$$

Definition 9 (Strong-EJR for Approval Ballots (Updated)). *Given an instance $I = \langle \mathcal{P}, c, b\rangle$, a profile $\boldsymbol{A}$ of approval ballots, and a satisfaction function sat, a budget allocation π is said to satisfy Strong-EJR[sat] if for all $P \subseteq \mathcal{P}$, we have $sat_i(\pi) \geq sat_i(P)$ for all agents $i \in N^{\max}$, the corresponding **maximal** P-cohesive group.*

Finding $N^{\max}$ for approval ballots and for cardinal ballots with $\alpha = 1$ (i.e., by searching in $\boldsymbol{A}$ the maximal set of agents that approve or assign positive scores to all projects in P) only takes $O(n|P|)$ or $O(nm)$, meaning the verification of Strong-EJR can be effectively done in $O(2^m \times (mn+n))$ or $O(2^m \times mn)$. Notably, the exponentiation term with n is dropped, which is substantial as $n \gg m$ in realistic PB processes [7, 15].

Remark 1 (Applicability to EJR Verification). *Theorem 1 does not hold for existential quantifiers. Formally, if there exists $i \in N$ such that $sat_i(\pi) \geq sat_i(P)$, then we are not guaranteed to have $i' \in N'$ such that $sat_{i'}(\pi) \geq sat_{i'}(P)$, particularly when $i \in N \setminus N', N' \subset N$. The same applies to cardinal ballots. Hence, EJR verification does not enjoy similar complexity reduction.*

Remark 1 is consistent with [2], which states that the verification of EJR given a budget allocation is coNP-complete. Additionally, our remark also holds for relaxations of EJR, e.g., EJR up to one project (EJR-1) [30] and EJR up to any project (EJR-X) [8]. Please refer to Appendix B for our similar remark on proportional justified representation (PJR) [36], an even weaker variant of Strong-EJR.

3.2 Strong-EJR Approximation

Given that Strong-EJR can now be verified much more efficiently, we introduce the following fairness objective based on Definitions 8 and 9. Given a PB instance I and a profile $\boldsymbol{A}$, we define *Strong-EJR approximation* achieved by a resolute PB rule $f : (I, \boldsymbol{A}) \to \pi$ as

$$\phi(I, \boldsymbol{A}, f) = \frac{1}{s} \sum_{P\in S} \min_{i^\star \in N^{\max}} \min\left\{1, \frac{\sum_{p\in f(\cdot)} A_{i^\star}(p)}{\sum_{p\in P} \min_{i\in N} A_i(p)}\right\} \quad (3)$$

for cardinal ballots and

$$\phi[sat](I, \boldsymbol{A}, f) = \frac{1}{s} \sum_{P\in S} \min_{i\in N^{\max}} \min\left\{1, \frac{sat_i(f(\cdot))}{sat_i(P)}\right\} \quad (4)$$

for approval ballots where S is the set of all maximal (α, P)-/P-cohesive groups $N^{\max}$ from $(I, \boldsymbol{A})$ and $s = |S|$, which is bounded by $[1, 2^m)$.[5] Clearly, $\phi \in [0, 1]$, with 1 being perceived as "fair".[6]

Unfortunately, by Remark 1 (and Remark 2 in Appendix B), similar definitions for weaker fairness properties such as EJR (and PJR) are not applicable.

3.3 Accelerating LLMRule with Efficient Computation of Strong-EJR Approximation

Following Section 3.1, computing Strong-EJR approximation also takes $O(2^m \times mn)$. Notice this procedure can be split into two steps:

$$O(2^m \times mn) = O(\underbrace{2^m \times mn}_{\text{(i)}} + \underbrace{2^m \times n}_{\text{(ii)}}) \quad (5)$$

where (i) is required for finding all s maximal cohesive groups and (ii) for computing the satisfaction ratio (term in braces from Equations 3 and 4) for all agents from them. Because (i) does not require f, it can be done separately beforehand i.e., prior to training LLMRule. Thus, the complexity can be effectively reduced to $O(2^m \times n)$.

We now seek to improve the exponential term with m knowing that not every P has at least one cohesive group. We now refer to $N^{\max}$ as simply N in the remainder of this section for readability. We make the following observation based on the definition of cohesive groups (Definitions 1 and 2).

$$\frac{|N|}{n} \cdot b \geq c(P)$$
$$\gamma = \frac{|N|}{n} \cdot \frac{b}{c(P)} \geq 1.$$

In other words, the larger γ, the more "deserving" N gets to have some satisfaction in the final allocation outcome. From this observation, we can sort P-cohesive groups N by descending γ, where ties are broken based on the cumulative approval score $\prod_{p\in P} app(p, \boldsymbol{A})$, where $app(p, \boldsymbol{A}) = |\{i \in \mathcal{N} \mid A_i(p) = 1\}|$. Notice $|N|/n$ is the frequency of P in $\boldsymbol{A}$, which is consistent with the definition of *support* in frequent itemset mining [19]. Hence, we can use any algorithm in this field to obtain a sorted list of maximal cohesive groups. Let $supp_P$ be the support for P and consider the standard Apriori algorithm [21]. Because $supp_P = \frac{|N|}{n} \geq \frac{c(P)}{b}$ for any P-cohesive group N, the minimum support is $\min_{p\in\mathcal{P}} c(p)/b$, i.e., the cost of the cheapest project in $\mathcal{P}$ over b. We use this value as input to the

[6] The upper bound is not necessarily tight since Strong-EJR is unsatisfiable in general [2].

Apriori algorithm. Its output is a set of 2-tuples $(P, supp_P)$, which is used to compute γ, then search for N (which takes $O(mn)$ as discussed). Given that Apriori runs in $O(2^m)$ [21], step (i) in Equation 5 still maintains a complexity of $O(2^m + 2^m \times mn) = O(2^m \times mn)$ in total. Once the resulting list of s maximal cohesive groups is sorted, given f, we can proceed to step (ii) in Equation 5 i.e., compute ϕ starting from P-cohesive group N with largest γ. Overall, computing Strong-EJR approximation in this way while training LLMRule takes only $O(s \times n) < O(2^m \times n)$. When considering instances with large m and approval ballots in particular, an upper limit of top-σ maximal cohesive groups with $\sigma < s \ll 2^m$ can be specified in exchange for exhaustiveness.

3.4 LLMs for Designing PB Rules

Rule Representation. We first describe the components to represent a PB rule.

(1) The rule description comprises a few sentences in natural language. It is created by LLMs and encapsulates a high-level thought. An example is provided in Figure 1 (top).
(2) The code block is an implementation of the PB rule (Figure 1, bottom). It should follow a pre-defined format (see Listing S2) so that it can be identified and seamlessly integrated into the evolutionary framework. In the experiments, we choose to implement it as a Python function, though any programming language should work.
(3) Each rule is assigned a fitness value to represent its priority in the population, which is used for selection and population management. We elaborate our definition of fitness based on utility and fairness objectives shortly.

Evolutionary Search. To begin the evolution process, we inform the LLM of the PB problem of interest and instruct it to design a new rule by first presenting the description of the generated rule and the corresponding code block (see Figure S4, top). We repeat h times to generate h initial rules. From these initial rules, offspring rules are generated by prompting the LLM with specific instructions, called *prompt strategies*, that either explore or modify existing rules. Exploration strategies focus more on exploring uncharted area in the space of heuristics by conducting crossover-like operators on parent heuristics (Figure S4, left). Modification strategies refine a parent heuristic by modifying, adjusting parameters, and removing redundant parts (Figure S4, right). For selecting parent rule(s) from a population R of h rules when generating an offspring rule, we first rank them according to their fitness, then randomly select one or multiple rules f_j with probability $\Pr(f_j) \propto 1/(\rho_j + h)$, where ρ_j is its rank in R. Once h offspring rules are generated, we select the h best rules from the current population (comprising parent and offspring rules) to form a population for the next iteration. The evolution process terminates when the evaluation resource (e.g., number of populations) is depleted.

Prompt Refinement. Since prompt strategies dictate the diversity in the population and play a vital role in producing high-quality heuristics [28], thoughtful prompt engineering is necessary, which often requires certain degree of domain knowledge (especially for modification ones). To alleviate this, we incorporate a prompt refinement module wherein prompt strategies are iteratively refined as well. In particular, when the fitness values of the top-l rules in the population remain unchanged for t consecutive units of evaluation resource, which indicates stagnation during the search, we refine the prompt strategies from each type as follows.

- **Exploration:** Given the current set of exploration strategies, ask the LLM to design a new one that is different from them as much as possible (Figure S5, top).
- **Modification:** Given the current set of modification strategies, ask the LLM to design a new one that is different from them as much as possible. (Figure S5, bottom).

By this means, two new prompt strategies, one from each type, are added after calling this module. To maintain the size of the set of prompt strategies, during the evolution process, we assign each prompt strategy a score defined as the average fitness value of the top-d individuals produced by it, and keep those with highest scores for generating offspring rules.

Fitness Evaluation. The evaluation of each newly generated PB rule f involves running it on a set of K problem instances from which its fitness value is computed. Since we aim to maximize both utility (Equations 1 and 2) and fairness (Equations 3 and 4) objectives, the associated fitness function should include both terms. Unfortunately, existing works on LLMs have yet to consider multi-objective combinatorial optimization to our best knowledge. In response, we circumvent this challenge by introducing a penalty term in the fitness function whenever f is not empirically fair. Formally, given a heuristic PB rule f that takes an instance I and a profile $\boldsymbol{A}$ of ballots as input and outputs a budget allocation $\pi \subseteq \mathcal{P}$,

Rule Description (generated using the initial E1 prompt strategy)

A new strategy will utilize a geometric mean approach combining approval rates and a cost-to-budget proportion, enabling a balanced evaluation for project prioritization.

```python
import numpy as np
def priority(project_costs, budget, approval_mat):
    '''
    Args:
    project_costs (np.ndarray): Array of length M
        storing the project costs.
    budget (int or float): A positive value.
    approval_mat (np.ndarray): An N-by-M matrix where
        each row contains a sample of binary approvals
        for the respective projects.

    Returns:
    scores (np.ndarray): Array of priority scores for
        the projects.
    '''
    approval_rates = np.mean(approval_mat, axis=0)
    cost_to_budget_ratio = project_costs / budget
    scores = np.sqrt(approval_rates) * (1 / (1 +
        cost_to_budget_ratio))

    return scores
```

Figure 1: High-level description of an LLM-generated PB rule for approval ballots with $sat = sat^{cost}$ and its associated code.

its fitness value $q(f)$ is defined as[7]

$$q(f) = \frac{1}{K}\sum_{k=1}^{K}\Big[\ \omega'\Big(I^{(k)}, \boldsymbol{A}^{(k)}, f(\cdot)\Big) - \theta\Big(\phi\big(I^{(k)}, \boldsymbol{A}^{(k)}, f(\cdot)\big)\Big)\Big], \tag{6}$$

where

$$\omega'(I, \boldsymbol{A}, f) = \frac{\omega(I, \boldsymbol{A}, f)}{\omega(I, \boldsymbol{A}, \pi^{\mathrm{OPT}})}$$

is the utilitarian social welfare of f relative to the one yielded from the optimal budget allocation π^{OPT} (obtained by computing the welfare maximizer) and θ is a function of Strong-EJR approximation ϕ yielded from f:

$$\theta\Big(\phi(\cdot)\Big) = \begin{cases} 1 & \text{if } \phi < \varepsilon \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

for some value $0 \le \varepsilon \le 1$. That is, we penalize f's fitness value if its associated Strong-EJR approximation is less than some predefined threshold. Because ω' is bounded within $[0, 1]$, a PB rule with a negative fitness value implies that it is not empirically fair (subject to Strong-EJR). Setting $\varepsilon < 1$ relaxes the fairness constraint.

4 EXPERIMENTS

4.1 Experimental Setups

All experiments were conducted under Ubuntu 20.04 on a Linux virtual machine equipped with NVIDIA GeForce RTX 3050 Ti GPU and 12th Gen Intel(R) Core(TM) i7-12700H CPU @2.3GHz. We chose GPT-4o mini as our pretrained LLM. The code for our implementation in Python 3.10 is available on GitHub[8].

Dataset. We obtain 617 real-world PB instances from Pabulib[9] [15], which were held in the U.S., Canada, Poland, and the Netherlands with either approval or cardinal ballots. We include all instances satisfying the following criteria: having 3 to 25 projects, not "fully funded"[5], and not "experimental" i.e., synthetic. Note that Pabulib only includes a special type of cardinal ballots, called cumulative ballots, in which voters are allocated a specific number of points that they can distribute among the projects ($\sum_{p\in\mathcal{P}} A_i(p) \le 1$). For each of the two types of ballots, we further categorize the dataset into small and large PB instances based on n, with small instances having fewer than 1,000 voters. Table 1 specifies our employed dataset. Notably, we train LLMRule using the small instances, then test the learned rules on the large instances, i.e., out-of-distribution (OOD) testing. Due to a plethora of small PB instances with approval ballots, we only use those held in the U.S. for training and the rest for in-distribution (ID) testing.

Baselines. We consider the following well-studied PB rules in the literature that were implemented in the Pabutools library[10] [15].

- MaxUtil: the Additive Utilitarian Welfare Maximizer [41], which returns π^{OPT}.
- GreedUtil: the Greedy Utilitarian Rule [41].

[7]For brevity, we omit the $[sat]$ parameterization in ω and θ for approval ballots.
[8]https://github.com/AnonyMouse3005/LLM-PB
[9]https://pabulib.org/
[10]https://github.com/COMSOC-Community/pabutools

Table 1: Number of PB instances obtained from Pabulib. Data used for training LLMRule are shaded in green, and data for OOD testing are unshaded. Approval ballots also include an ID test set (shaded in gray).

Ballot type	Small (n <1,000)		Large (n ≥1,000)
Cardinal	63		122
Approval	74	133	225

- MES: the Method of Equal Shares [30]. For approval ballots, we run MES for $sat = sat^{cost}$ and $sat = sat^{card}$, denoted respectively as MES$[sat^{cost}]$ and MES$[sat^{card}]$. Because MES typically does not spend all the given budget, i.e., it is not *exhaustive*, we additionally run MES with three completion methods: 'Add1' (special case of 'Exh2' in [30]) along with 'Add1U' [15] and 'Add1UM' that respectively use GreedUtil and MaxUtil for exhausting the remaining budget.

For approval ballots, we also consider two standard proportional rules other than MES:

- SeqPhrag [29]: the Sequential Phragmén Rule.
- MaximinSupp [4]: the Maximin Support Rule, also known as Generalised Sequential Phragmén Rule.

By default, these rules are irresolute, hence we make them resolute by implementing lexicographic tie-breaking.

In terms of which rules satisfy which fairness property, for cardinal ballots, MES satisfies EJR-1 [30] and PJR-1 [29]. For approval ballots,

- Strong-EJR$[sat]$: None [33]
- EJR$[sat]$: MES$[sat^{card}]$ [30]
- EJR-X$[sat]$: MES$[sat]$ for any DNS function sat[11] [8]
- EJR-1$[sat]$: MES$[sat]$ for any additive sat [30]
- PJR$[sat]$: None
- PJR-X$[sat]$: MES$[sat]$ for any DNS function sat[11], SeqPhrag, MaximinSupp [8]

Note that we do not consider the theoretically appealing "greedy cohesive rule" introduced in [30], which has been shown to satisfy all listed fairness properties except Strong-EJR, since it runs in time exponential in n and therefore highly impractical.

Implementation Details. We consider three problem settings: approval ballots with (i) $sat = sat^{cost}$ and (ii) $sat = sat^{card}$, and (iii) cardinal ballots. That is, the utility objectives are defined as (i) $\omega[sat^{cost}]$ and (ii) $\omega[sat^{card}]$ (Equation 2), and (iii) ω (Equation 1). In each setting, we use LLMRule to design greedy rules, in which the task for the LLM is to design a priority function for assigning a score to each project. This function takes a list of project costs, the budget constraint, and the ballot profile as input and outputs the scores for the projects. Using these scores, the allocation outcome $\pi \subseteq \mathcal{P}$ is constructed by greedily adding the projects until none of the remaining projects are affordable given the current budget. Please refer to Appendix C for further implementation details.

Our motivations for designing greedy rules are twofold. *Flexibility*: Greedy rules can also serve as effective completion methods (e.g., 'Add1') [15] for proportional rules that are non-exhaustive

[11]Both sat^{cost} and sat^{card} are DNS functions (defined and proved in [8]).

such as MES. *Practicality*: GreedUtil is in fact the most widely used rule in the real world [33].

4.2 Results

Approval Ballots. Table 2 and Figure 2 show results for approval ballots. In both settings, compared to proportional rules (SeqPhrag, MaximinSupp, and all MES variants), LLMRule yields the best utilitarian social welfare while still maintaining a similar degree of fairness, which is interesting given that the rules generated from LLMRule are greedy in nature. Figure 1 provides an example of such rule (from one of the three runs of LLMRule) for sat^{cost}. Following the terminology in [33], the rule generated by LLMRule can be formalized as $\{\text{Greed}(I, \triangleright) \mid \triangleright \text{ is compatible with } \sqrt{app/n}/(1 + c/b)\}$. That is, p precedes p' in the order defined by $\triangleright$, denoted as $p \triangleright p'$, if and only if $\sqrt{app(p, A)/n}/(1 + c(p)/b) \geq \sqrt{app(p', A)/n}/(1 + c(p')/b)$ (recall from Section 3.3 that $app(p, A) = |\{i \in \mathcal{N} \mid A_i(p) = 1\}|$). For comparison, the baseline greedy utilitarian rule, which only concerns utility optimization and is hence significantly less fair, is defined as

Table 2: Results for approval ballots with (top) $sat = sat^{cost}$ and (bottom) $sat = sat^{card}$ on ID and OOD test sets. Utilitarian/proportional rules are respectively shaded in gray/green.

PB rule	$\omega'[sat^{cost}]$ (ID)	ϕ (ID)	$\omega'[sat^{cost}]$ (OOD)	ϕ (OOD)
MaxUtil	1	0.643	1	0.536
GreedUtil	0.978	0.750	0.971	0.686
SeqPhrag	0.710	0.955	0.695	0.956
MaximinSupp	0.696	0.956	0.691	0.960
MES[sat^{cost}]	0.524	0.831	0.470	0.806
MES[sat^{cost}]-Add1	0.766	0.934	0.775	0.917
MES[sat^{cost}]-Add1U	0.780	0.938	0.797	0.927
MES[sat^{cost}]-Add1UM	0.780	0.938	0.797	0.927
MES[sat^{card}]	0.419	0.855	0.391	0.824
MES[sat^{card}]-Add1	0.710	0.951	0.697	0.955
MES[sat^{card}]-Add1U	0.732	0.955	0.722	0.959
MES[sat^{card}]-Add1UM	0.732	0.955	0.722	0.958
LLMRule	0.802	0.940	0.814	0.901
MES[sat^{cost}]-LLMRule	0.775	0.951	0.774	0.943
MES[sat^{card}]-LLMRule	0.752	0.955	0.768	0.947

PB rule	$\omega'[sat^{card}]$ (ID)	ϕ (ID)	$\omega'[sat^{card}]$ (OOD)	ϕ (OOD)
MaxUtil	1	0.896	1	0.926
GreedUtil	0.964	0.962	0.984	0.967
SeqPhrag	0.943	0.960	0.959	0.960
MaximinSupp	0.959	0.961	0.977	0.964
MES[sat^{cost}]	0.763	0.854	0.732	0.820
MES[sat^{cost}]-Add1	0.942	0.942	0.945	0.925
MES[sat^{cost}]-Add1U	0.961	0.948	0.970	0.936
MES[sat^{cost}]-Add1UM	0.961	0.947	0.971	0.936
MES[sat^{card}]	0.759	0.870	0.719	0.834
MES[sat^{card}]-Add1	0.939	0.956	0.957	0.959
MES[sat^{card}]-Add1U	0.966	0.961	0.984	0.964
MES[sat^{card}]-Add1UM	0.967	0.961	0.985	0.964
LLMRule	0.975	0.956	0.986	0.941
MES[sat^{cost}]-LLMRule	0.971	0.957	0.985	0.948
MES[sat^{card}]-LLMRule	0.972	0.960	0.987	0.953

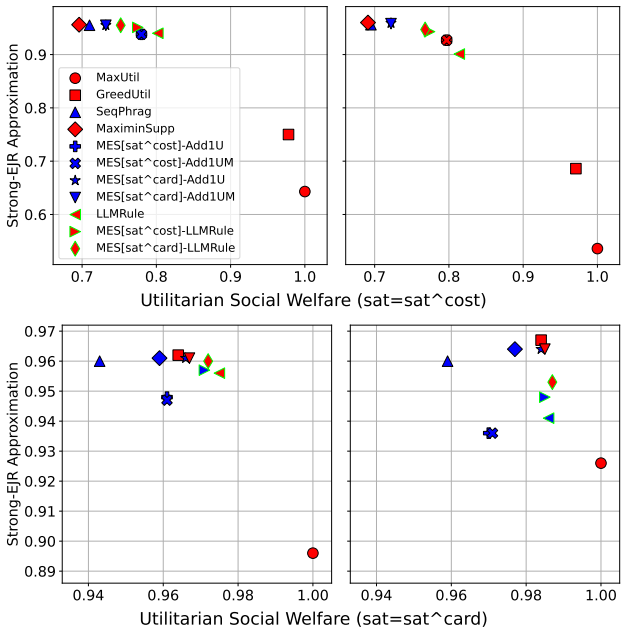


Figure 2: Utilitarian social welfare vs. Strong-EJR approximation for approval ballots on (left) ID and (right) OOD test sets. The Pareto front is highlighted red. Rules with LLMRule are bordered green. MES and MES-Add1 (either *sat*) were omitted due to being Pareto dominated in all settings.

GreedUtil$(I, A) = \{\text{Greed}(I, \triangleright) \mid \triangleright \text{ is compatible with } app\}$. Furthermore, the good performance of LLMRule on ID test instances also applies to OOD ones, indicating the generalizability of LLM-generated rules to larger instances with more agents.

When using the same greedy rules from LLMRule as completion methods for MES (MES[sat]-LLMRule)[12], as expected, we observe an improvement in ϕ in exchange for a decrease in ω. Notably, compared to other completion methods ('Add1', 'Add1U', 'Add1UM'), LLMRule yields the best ϕ for $sat = sat^{cost}$ and the best ω for $sat = sat^{card}$ (please refer to Figure S8 for more succinct plots).

Overall, the designed rules from LLMRule achieve impressive utility-fairness trade-offs and excel either as standalone greedy utilitarian rules or completion methods for non-exhaustive rules. Further experiments involving synthetic data in Appendix G solidify our claim.

Cardinal Ballots. Table 3 and Figure 3 show results for the more general setting with cardinal ballots. Surprisingly, LLMRule yields the best utilitarian social welfare among the baselines (beside the utilitarian maximizer), even surpassing the greedy utilitarian rule. Meanwhile, the respective Strong-EJR approximation approaches 1, meaning the learned rules are still (empirically) fair. We provide an example of such rule in Figure S7. Compared to the previously shown rule for approval ballots, the greedy scheme from this rule appears to be more sophisticated: after computing the scores, it further ensures projects that bring the highest overall utility are

[12]Only MES is non-exhaustive in our considered baselines. We use the exhaustive variant of SeqPhrag and MaximinSupp (originally non-exhaustive) from Pabutools.

Table 3: Results for cardinal ballots. Utilitarian rules and proportional rules are respectively shaded in gray and green.

PB rule	ω'	ϕ
MaxUtil	1	0.921
GreedUtil	0.961	1
MES	0.454	0.921
MES-Add1	0.909	1
MES-Add1U	0.945	1
MES-Add1UM	0.946	1
LLMRule	0.972	0.993
MES-LLMRule	0.970	0.995

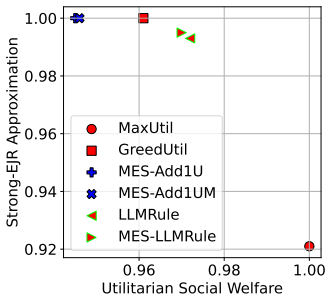


Figure 3: ω' vs. ϕ for cardinal ballots. The Pareto front is highlighted red. Rules with LLMRule are bordered green.

prioritized. Notice that the prompt strategy responsible for producing this rule is newly introduced via our prompt refinement module, which indicates its capability of facilitating the design of novel rules.

5 RELATED WORK

We discuss the related work focusing on the considered PB settings and LLMs for algorithmic design. For a comprehensive review of the PB literature, we refer readers to the surveys [5, 33]. For LLMs, please refer to [28]. We follow the notations from Sections 2 and 3 when applicable. Appendix A expands on the quantification of fairness and existing data-driven approaches for designing PB rules.

The Design of Participatory Budgeting Rules. As previously mentioned, existing PB rules mainly fall into two types: *utilitarian PB rules* and *proportional PB rules*. Utilitarian PB rules concern the overall utility of the agents. For approval ballots, Talmon and Faliszewski [41] consider the max greedy and proportional greedy PB rules defined with respect to the utilitarian social welfare under the cardinality (sat^{card}) and cost (sat^{cost}) satisfaction functions. While maximizing the utilitarian social welfare under sat^{card} can be done in polynomial time, they show that it is weakly NP-hard under sat^{cost}, for which a pseudo-polynomial time algorithm is provided. For cardinal ballots, Fluschnik et al. [17] consider the problem of utility maximization as a variant of the knapsack problem and propose similar greedy rules based on the knapsack literature [25].

Proportional PB rules focus on the fairness of the resulting budget allocations with respect to agent preferences. These rules ensure any sufficiently large and cohesive group of voters should secure allocation outcomes proportional to its size, which is well captured by Strong-EJR. However, this ideal definition of fairness is not satisfiable in general [12]. Therefore, existing works consider weaker but satisfiable fairness definitions, most notably EJR [2, 30] and PJR [36] as well as their relaxations, e.g., EJR-1 [30], EJR-X [8], PJR-1 [29], PJR-X [8]. Unfortunately, it has been theoretically shown that the "price of fairness" exists [13, 14]. More specifically, [14] prove that if f satisfies EJR[sat^{card}], then for any instance I and profile $\boldsymbol{A}$ of approval ballots, the lower bound of $\omega'[sat^{card}](I, \boldsymbol{A}, f)$ is $\frac{c_{\min}}{n \cdot b} \left\lfloor \frac{b}{c_{\max}} \right\rfloor$, where $c_{\min} = \min_{p \in \mathcal{P}} c(p)$ and $c_{\max} = \max_{p \in \mathcal{P}} c(p)$. In other words, proportional rules are subject to poor worst-case utilitarian social welfare, which calls for the design of rules that yield better trade-offs between utility and fairness objectives.

Large Language Models for Algorithmic Design. LLMs have been applied in diverse tasks, including mathematical reasoning [1], code generation [23], scientific discovery [42], and algorithm design [28]. FunSearch [35] integrated LLMs into an evolutionary search framework for mathematical function design. EoH [27] adopted LLMs to automatically generate, combine and improve heuristics in an evolutionary search. [20] proposed ADAS, or Automated Design of Agentic Systems, that aims to automatically generate powerful agentic system designs by using meta agents to program new agents. The significance of integrating LLMs into iterative search frameworks, particularly for challenging design tasks requiring environmental evaluations, has been emphasized in [44]. Despite these advancements [28], the application of LLMs to designing PB rules remains unexplored.

Applying existing LLM-based methods, e.g., EoH [27], requires addressing three challenges as previously stated in our introduction. Among them, ensuring the verification of the fairness property of interest runs in tractable time is the most important. Even in the special case of PB (i.e., multiwinner voting, where every project has unit cost and the budget allocation is required to be exhaustive), both EJR and PJR were shown to be coNP-complete to verify given a budget allocation [2, 3]. While stronger variants of EJR and PJR may be verifiable in polynomial time, e.g., EJR+ and PJR+ [9] (though not as strong as Strong-EJR), this is not the case in general, e.g., as in full justified representation (FJR) and full proportional justified representation (FPJR) [24]. EJR+ and PJR+ were originally introduced for multiwinner voting settings [9] and could be extended to PB [9, 37]. However, it is clear that the demonstrated polynomial-time verification procedure of either fairness property (from the proofs of Propositions 2 and 6 in the paper), which hinges on the assumption that every project has unit cost (i.e., each candidate occupies a seat if voted), no longer applies to PB. In this paper, we circumvent this challenge by devising techniques to enable the fast computation of our newly introduced fairness objective based on our provably equivalent definition of Strong-EJR with reduced verification complexity.

6 CONCLUSION

In this paper, we propose LLMRule, the first LLM-based framework for automating the design of PB rules. Through our experiments on various real-world PB instances, we demonstrate that LLMRule can generate rules that perform well in terms of both utility and fairness objectives while requiring minimal domain knowledge and prompt design. By this means, LLMRule can significantly reduce manual efforts involved in contemporary research on PB.

ACKNOWLEDGMENTS

This work is supported by the National Institute of General Medical Sciences of the National Institutes of Health [P20GM130461], the Rural Drug Addiction Research Center at the University of Nebraska-Lincoln, and the National Science Foundation under grants IIS:RI #2302999 and IIS:RI #2414554. The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies.

REFERENCES

[1] Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157* (2024).

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

[3] Haris Aziz, Edith Elkind, Shenwei Huang, Martin Lackner, Luis Sánchez-Fernández, and Piotr Skowron. 2018. On the complexity of extended and proportional justified representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[4] Haris Aziz, Barton E. Lee, and Nimrod Talmon. 2018. Proportionally Representative Participatory Budgeting: Axioms and Algorithms. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (Stockholm, Sweden) *(AAMAS '18)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 23–31.

[5] Haris Aziz and Nisarg Shah. 2020. Participatory budgeting: Models and approaches. In *Pathways Between Social Science and Computational Social Science: Theories, Methods, and Interpretations*. Springer, 215–236.

[6] Dorothea Baumeister, Linus Boes, and Tessa Seeger. 2020. Irresolute approval-based budgeting. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. 1774–1776.

[7] Niclas Boehmer, Piotr Faliszewski, Łukasz Janeczko, Andrzej Kaczmarczyk, Grzegorz Lisowski, Grzegorz Pierczyński, Simon Rey, Dariusz Stolicki, Stanisław Szufa, and Tomasz Wąs. 2024. Guide to Numerical Experiments on Elections in Computational Social Choice. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, Kate Larson (Ed.). International Joint Conferences on Artificial Intelligence Organization, 7962–7970. https://doi.org/10.24963/ijcai.2024/881 Survey Track.

[8] Markus Brill, Stefan Forster, Martin Lackner, Jan Maly, and Jannik Peters. 2023. Proportionality in approval-based participatory budgeting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 5524–5531.

[9] Markus Brill and Jannik Peters. 2023. Robust and Verifiable Proportionality Axioms for Multiwinner Voting. In *EC '23* (London, United Kingdom) *(EC '23)*. ACM, New York, NY, USA, 301. https://doi.org/10.1145/3580507.3597785

[10] Yves Cabannes. 2004. Participatory budgeting: a significant contribution to participatory democracy. *Environment and urbanization* 16, 1 (2004), 27–46.

[11] Alfonso Cevallos and Alistair Stewart. 2021. A verifiably secure and proportional committee election rule. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*. 29–42.

[12] Nikhil Chandak, Shashwat Goel, and Dominik Peters. 2024. Proportional aggregation of preferences for sequential decision making. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 9573–9581.

[13] Edith Elkind, Piotr Faliszewski, Ayumi Igarashi, Pasin Manurangsi, Ulrike Schmidt-Kraepelin, and Warut Suksompong. 2022. The Price of Justified Representation. *Proceedings of the AAAI Conference on Artificial Intelligence* 36, 5 (Jun. 2022), 4983–4990. https://doi.org/10.1609/aaai.v36i5.20429

[14] Roy Fairstein, Dan Vilenchik, Reshef Meir, and Kobi Gal. 2022. Welfare vs. Representation in Participatory Budgeting. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems* (Virtual Event, New Zealand) *(AAMAS '22)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 409–417.

[15] Piotr Faliszewski, Jarosław Flis, Dominik Peters, Grzegorz Pierczyński, Piotr Skowron, Dariusz Stolicki, Stanisław Szufa, and Nimrod Talmon. 2023. Participatory Budgeting: Data, Tools and Analysis. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, Edith Elkind (Ed.). International Joint Conferences on Artificial Intelligence Organization, 2667–2674. https://doi.org/10.24963/ijcai.2023/297 Main Track.

[16] Piotr Faliszewski, Piotr Skowron, Arkadii Slinko, and Nimrod Talmon. 2017. Multiwinner voting: A new challenge for social choice theory. *Trends in computational social choice* 74, 2017 (2017), 27–47.

[17] Till Fluschnik, Piotr Skowron, Mervin Triphaus, and Kai Wilker. 2019. Fair knapsack. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 1941–1948.

[18] B Gutiérrez. 2018. Madrid as a democracy lab, 2017. *URL https://www. opendemocracy. net/democraciaabierta/bernardo-guti-rrez/madrid-as-democracy-lab. Accessed* (2018), 05–02.

[19] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. 2007. Frequent pattern mining: current status and future directions. *Data mining and knowledge discovery* 15, 1 (2007), 55–86.

[20] Shengran Hu, Cong Lu, and Jeff Clune. 2024. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435* (2024).

[21] Liusheng Huang, Huaping Chen, Xun Wang, and Guoliang Chen. 2000. A fast algorithm for mining association rules. *Journal of Computer Science and Technology* 15 (2000), 619–624.

[22] Sheena S Iyengar and Mark R Lepper. 2000. When choice is demotivating: Can one desire too much of a good thing? *Journal of personality and social psychology* 79, 6 (2000), 995.

[23] Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2024. A Survey on Large Language Models for Code Generation. *arXiv preprint arXiv:2406.00515* (2024).

[24] Yusuf Hakan Kalayci, Jiasen Liu, and David Kempe. 2025. Full Proportional Justified Representation. In *AAMAS 2025* (Detroit, MI, USA). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1070–1078.

[25] Hans Kellerer, Ulrich Pferschy, and David Pisinger. 2004. *Knapsack Problems*. Springer. https://doi.org/10.1007/978-3-540-24777-7

[26] Martin Lackner and Piotr Skowron. 2020. Utilitarian welfare and representation guarantees of approval-based multiwinner rules. *AI* 288 (2020), 103366.

[27] Fei Liu, Tong Xialiang, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. 2024. Evolution of Heuristics: Towards Efficient Automatic Algorithm Design Using Large Language Model. In *Forty-first International Conference on Machine Learning*.

[28] Fei Liu, Yiming Yao, Ping Guo, Zhiyuan Yang, Xi Lin, Xialiang Tong, Mingxuan Yuan, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. 2024. A systematic survey on large language models for algorithm design. *arXiv preprint arXiv:2410.14716* (2024).

[29] Maaike Los, Zoé Christoff, and Davide Grossi. 2022. Proportional Budget Allocations: Towards a Systematization. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, Lud De Raedt (Ed.). International Joint Conferences on Artificial Intelligence Organization, 398–404. https://doi.org/10.24963/ijcai.2022/57 Main Track.

[30] Dominik Peters, Grzegorz Pierczyński, and Piotr Skowron. 2021. Proportional participatory budgeting with additive utilities. *Advances in Neural Information Processing Systems* 34 (2021), 12726–12737.

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

[32] Simon Rey, Ulle Endriss, and Ronald de Haan. 2021. Shortlisting Rules and Incentives in an End-to-End Model for Participatory Budgeting. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Zhi-Hua Zhou (Ed.). International Joint Conferences on Artificial Intelligence Organization, 370–376. https://doi.org/10.24963/ijcai.2021/52 Main Track.

[33] Simon Rey, Felicia Schmidt, and Jan Maly. 2023. The (computational) social choice take on indivisible participatory budgeting. *arXiv preprint arXiv:2303.00621* (2023).

[34] Anja Röcke. 2014. *Framing citizen participation: Participatory budgeting in France, Germany and the United Kingdom*. Springer.

[35] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. 2024. Mathematical discoveries from program search with large language models. *Nature* 625, 7995 (2024), 468–475.

[36] Luis Sánchez-Fernández, Edith Elkind, Martin Lackner, Norberto Fernández, Jesús Fisteus, Pablo Basanta Val, and Piotr Skowron. 2017. Proportional justified representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[37] Ulrike Schmidt-Kraepelin, Warut Suksompong, and Markus Utke. 2025. Discrete Budget Aggregation: Truthfulness and Proportionality. *arXiv preprint arXiv:2505.05708* (2025).

[38] Anwar Shah. 2007. Public Sector Governance and Accountability series. *World Bank, Washington* (2007).

[39] Piotr Skowron, Piotr Faliszewski, and Jérôme Lang. 2016. Finding a collective set of items: From proportional multirepresentation to group recommendation. *AI* 241 (2016), 191–216.

[40] Gogulapati Sreedurga, Mayank Ratan Bhardwaj, and Yadati Narahari. 2022. Maxmin Participatory Budgeting. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, Lud De Raedt (Ed.). International Joint Conferences on Artificial Intelligence Organization, 489–495. https://doi.org/10.24963/ijcai.2022/70 Main Track.

[41] Nimrod Talmon and Piotr Faliszewski. 2019. A framework for approval-based budgeting methods. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 2181–2188.

[42] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. 2023. Scientific discovery in the age of artificial intelligence. *Nature* 620, 7972 (2023), 47–60.

[43] Shunyu Yao, Fei Liu, Xi Lin, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. 2025. Multi-objective evolution of heuristic using large language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 27144–27152.

[44] Rui Zhang, Fei Liu, Xi Lin, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. 2024. Understanding the Importance of Evolutionary Search in Automated Heuristic Design with Large Language Models. In *International Conference on Parallel Problem Solving from Nature*. Springer, 185–202.

[45] Zhi Zheng, Zhuoliang Xie, Zhenkun Wang, and Bryan Hooi. 2025. Monte Carlo Tree Search for Comprehensive Exploration in LLM-Based Automatic Heuristic Design. In *Forty-Second International Conference on Machine Learning*.