Multiple Attribute List Aggregation and an Application to Democratic Playlist Editing

Eyal Briman(✉) and Nimrod Talmon

Ben-Gurion University, Beersheba, Israel
briman@post.bgu.ac.il

Abstract. We present a social choice model that incorporates time-based constraints, where the goal is to produce an ordered list that satisfies both agent preferences (based on approval ballots) and global constraints. First, we analyze the general model, showing that it is generally NP-hard, but admits polynomial-time algorithms for a special case; we also develop heuristic solutions for the general case. Furthermore, we explore potential applications of the model and demonstrate its relevance by focusing on the use case of democratic playlist editing. In this scenario, our aim is to generate a playlist that reflects agent preferences for a given set of musical tracks while also considering soft constraints regarding the sequencing and transitions of tracks over time. We illustrate how the problem of democratic playlist editing can be translated into our model, and present simulation results where we apply our heuristics to solve specific instances of the problem. We contend that our results are promising, not only for the specific use case of democratic playlist editing, but also for a plethora of other use cases that we introduce here.

1 Introduction

Social choice theory examines collective decision-making processes by aggregating individual preferences to reach a collective choice or outcome [2]. In the standard setting of single-winner elections, a single alternative is chosen from a set of alternatives, based on group preferences. Moving beyond single-winner elections, in multi-winner elections—which formally generalize single-winner elections – a set of candidates or alternatives is selected, following agent preferences [12]. Going even further, in participatory budgeting – which formally generalizes multi-winner elections – a set of projects, each with its cost, is selected while respecting agent preferences and a given budget.

We argue that advancements in social choice many times correspond to the desire of having collective decision-making tools that aggregate agent preferences and output *increasingly complex outputs*. Our proposed social choice model can be viewed from this angle; and, correspondingly, the aggregation methods that we develop can be seen as computational tools that are able to aggregate agent preferences and output fairly structurally-involved outputs. In particular, we consider a social choice setting that consists of the following ingredients:

V. Malvone and A. Murano (Eds.): EUMAS 2023, LNAI 14282, pp. 1–16, 2023.
https://doi.org/10.1007/978-3-031-43264-4_1

- A set of elements out of which a subset shall be selected (similar to a multi-winner election);
- where these elements have certain numerically-valued attributes that shall be taken into consideration (formally generalizing participatory budgeting);
- and where the selected subset of elements shall be ordered (similarly to ranking elements, such as in proportional ranking [28]).

We describe our setting formally in Sect. 3. Essentially, we formulate a multi-objective optimization problem that balances between two considerations:

- First, the ordered subset (i.e., list) that we output shall respect agent preferences (we model this aspect by aiming to maximize the score of the output list according to some multi-winner voting rule);
- second, the trend or pattern of the attribute values over time within the output list (we model this aspect by aiming to minimize the distance of the corresponding patterns to "ideal", predefined patterns).

Through the use of mathematical optimization techniques and the development of effective heuristics, our model provides a robust framework for tackling diverse use-cases that involve multi-attribute decision problems in which an output list is to be agreed upon. While in this paper we concentrate on a specific application – namely, of democratic playlist editing – below we first describe several potential applications of our model. Consider the following applications:

- **Democratic Planning:** Consider the task of some cooperative manufacturing plan for highly logistic complex products, sensitive products, or products with occasional or seasonal demand. We aim to optimize different attributes based on the social choice, ideal demand changing over time, the ideal stock and inventory changing over time, and different measures such as service quality type 1 or 2, which will ideally change over time to minimize costs.
- **Democratic Scheduling:** Democratic scheduling of jobs in a cooperative using a multi-attribute approach that includes, e.g., job type and its corresponding agent groups.
- **Democratic Media:** Creating content that serves different functions for various attributes over time, for example, a TV series or a movie. The amount of stress/relief or happiness/sadness that the show/movie creates in the viewer's experience as the season or movie evolves can be measured.

In Sect. 3 we provide our formal model, which we argue fits all these applications. Then, in Sect. 7 we describe a different application over which we demonstrate the applicability of our model as well as the suitability and quality of several heuristic solutions that we propose for the model. The concrete application that we concentrate on in this paper is of *democratic playlist editing*:

- **Democratic Playlist Editing:** Producing a musical playlist - perhaps to accompany a podcast, movie, TV show, theater or dance, where there should be changes in different attributes over time, such as tempo, loudness, and emotions that will be expressed through the music (using chords, scales, and other musical tools).

1.1 Paper Structure

After discussing some related work (in Sect. 2), we go on to describe our formal model (in Sect. 3).

Then, as the aspect of our model that corresponds to respecting agent preferences is both crucial to our model as well as general, in Sect. 4 we describe how to capture different multi-winner rules in our model. We go on to provide a computational analysis of our general model (in Sect. 5) and to describe various general heuristics that we propose for solving instances of the general model (in Sect. 6).

We continue to describe the problem of democratic playlist editing in detail and in a formal way (in Sect. 7) and to report on computer-based simulations that we have performed on real-world and artificial data to evaluate the relevance and the quality of our algorithms (in Sect. 8).

We conclude in Sect. 9 with a discussion on the implications of our research and on promising future research directions.

2 Related Work

In this section, we discuss relevant related work in three areas: multi-attribute social choice, proportional ranking, and multi-attribute scheduling. Indeed, these three areas of study are the basis for the model presented in this work.

First, however, we would like to mention some related work that does not directly fall within these three areas:

– First, as our model can be viewed as a time-based social choice model, we mention ongoing work regarding the aggregation of continuously-changing agent preferences [1].
– Second, in our work, we draw inspiration from work that combines social choice aspects with recommendation systems, such as the work of Burke et al. [7], that evolves around the exploration of dynamic fairness-aware recommendation systems using multi-agent social choice.
– Third, our multi-attribute setting also has some connections with work on the group activity selection problem [9].

2.1 Multi-attribute Social Choice

Multi-attribute social choice involves aggregating preferences of a group over alternatives with multiple attributes. Our work focuses on a multi-attribute setting where each element has a vector of numerical attribute values [5,8,16]. Various works explore social choice settings with different attributes, such as democratic parliamentary elections aiming for proportional representation of attributes like gender and race in society [4,6,22,23,27].

2.2 Proportional Ranking

Traditional approval voting ranks alternatives based on the percentage of approving voters, but it may not accurately reflect the preferences of the population. The proportional ranking model addresses this by interleaving alternatives supported by different groups of agents, reflecting relative popularity. It emphasizes sorting candidates to ensure proportional committees, considering candidate diversity and order. This model finds applications in recommendation systems, hiring, committee elections, and liquid democracy [28]. In our work, we aim to generate rankings with a broader scope than Skowron et al. [28]. Specifically, in Sect. 7, we consider proportionality in collaborative playlist editing, where tracks are ranked based on acoustic features, popularity, and proportional representation. We also mention other related works [14,18].

2.3 Multi-attribute Scheduling

The single-machine scheduling problem involves finding the optimal order of tasks on a single machine to minimize the total completion time or other objectives. Multi-attribute scheduling extends this by considering multiple objectives and constraints. It aims to find a schedule that satisfies constraints while optimizing objectives like completion time or resource utilization [15,20]. Our model generalizes the multi-attribute single-machine problem, emphasizing multi-objective optimization and incorporating the social choice aspect.

3 Multi-Attribute List Aggregation (MALA)

In this section, we present the formal model of Multi-Attribute List Aggregation (MALA). An instance of MALA consists of the following ingredients:

1. A set of y attributes, denoted with their index $q \in [y]$.
2. A set of m elements, $C = \{c_1, \ldots, c_m\}$. Each element c_i, $i \in [m]$, for each attribute $q \in [y]$, has some numerical value; we define c_i^q to be the numerical value of element i for attribute q (so, in particular, $c_i^q \in \mathbb{R}$).
3. A value $k \le m$, $k \in \mathbb{N}$; this is the desired size of the list that is the output of the instance.
4. A set of z so-called *Ω constraints*, denoted by $\{\Omega_1, \ldots, \Omega_z\}$. Below we describe what is an Ω-constraint: in particular, an Ω-constraint is defined by a tuple (q, F, d, w); next we describe what are q, F, d, and w:
   – $q \in [y]$ is the index of some attribute.
   – $F := \{f_1, \cdots, f_t\}$ is a family of t vectors, each of length k; formally, $f_\ell \in \mathbb{R}^k$, $\ell \in [t]$. We use a square brackets notation for vectors; i.e., for $\ell \in [t]$, $s \in [k]$, we write $f_\ell[s]$, $f_\ell[s] \in \mathbb{R}$, to denote the value of the s'th element of f_ℓ. (Intuitively, each of these vectors corresponds to some ideal behavior of the output list with respect to attribute q).

- d is a metric that returns a distance between two real-valued vectors of length k. Formally, $d : \mathbb{R}^k \times \mathbb{R}^k \rightarrow \mathbb{R}$; i.e., d is a function that takes two vectors of length k and returns a numeric value that we interpret as their distance; and it shall be a metric. (Intuitively, the metric d quantifies how close-to-ideal is the output list to at least one of the ideal vectors, with respect to the attribute q; note that we will use d only to evaluate the distance between some possible solution and some vector of F.)
- $w \in \mathbb{R}$, is the weight that the Ω-constraint gets. (Intuitively, it corresponds to the importance of that Ω-constraint.)

An instance of MALA as described above defines a cost for each possible *solution* to it. To describe what it is, let *solution* be some possible solution to an instance of MALA; first, formally, *solution* shall satisfy the following:

- $solution \in C^k$; i.e., *solution* is a vector of k elements, each from C.
- For $s_1 \neq s_2$, it holds that $solution[s_1] \neq solution[s_2]$; i.e., there can be no repetitions.

Below we describe the *cost* of a possible solution *solution*:

- First, the cost of a solution *solution* with respect to a specific Ω-constraint $\Omega = (q, F, d, w)$ is, roughly speaking, the weight (w) multiplied by the distance (according to d) between the values of the elements of the solution for the attribute q to the vector of F that is the closest to it; formally, we define:

$$cost(solution, \Omega) = w \cdot min_{f \in F} d(f, solution^q),$$

  where $solution^q \in \mathbb{R}^k$ is the vector containing the values of the elements of the solution with respect to the attribute q; formally, $solution^q[s] := solution[s]^q$, $s \in [k]$.
- Second, the cost of a solution *solution* with respect to an instance of MALA $MALA$ (that contains z Ω-constraints) is defined naturally as the summation of its cost with respect to each of the Ω-constraints; formally, we define:

$$cost(solution, MALA) = \sum_{i \in [z]} cost(solution, \Omega_i).$$

Given an instance of MALA $MALA$, we are looking for a solution *solution* of minimum cost; formally, we are looking for the following:

$$\arg \min_{\substack{solution \in C^k \\ solution[s_1] \neq solution[s_2] \\ \text{for } s_1 \neq s_2}} cost(solution, MALA)$$

Consider the following toy example.

Example 1. Jimmy would like to prepare food to take to work and needs to decide what to bring to each of his three meals, during the day. His decision is based on 3 attributes: (1) personal preferences, (2) calories, (3) and sugar.

His candidates are: (1) apple, (2) orange, (3) omelette sandwich, and (4) tuna sandwich. Each candidate is defined by its unique attributes' values. Jimmy also sets his ideal attribute values for each one of the three meals and the importance of each attribute. Thus, the formal instance – denoted by $MALA$ – is given by (with some specific data):

- $y = 3$;
- $k = 3$
- Ω-constraints:
  - q_1= apple, q_2= orange, q_3= omelette sandwich, q_4= tuna sandwich.
  - F_1= $\{f_1\}$, where $f_1[1] = 10, f_1[2] = 10, f_1[3] = 10$;
  - F_2= $\{f_2, f_3\}$, where $f_2[1] = 80, f_2[2] = 600, f_2[3] = 60$, and where $f_3[1] = 100, f_3[2] = 500, f_3[3] = 80$;
  - F_3= $\{f_4, f_5\}$, where $f_4[1] = 0, f_4[2] = 20, f_4[3] = 0$, and where $f_5[1] = 0, f_5[2] = 0, f_5[3] = 0$;
  - d_i is the ℓ_1 norm, for each $i \in [3]$;
  - $w_1 = 1, w_2 = 0.5, w_3 = 0.8$.
- $C = \{c_1, c_2, c_3, c_4\}$ with jimmy's preferences given by: $c_1^1 = 7$, $c_1^2 = 95$, $c_1^3 = 19$, $c_2^1 = 4$, $c_2^2 = 60$, $c_2^3 = 12$, $c_3^1 = 5$, $c_3^2 = 530$, $c_3^3 = 15$, $c_4^1 = 10$, $c_4^2 = 570$, $c_4^3 = 50$;

Now let us observe a possible *solution*$_1$: [apple, omelette sandwich, orange] from the perspective of the cost of each of the Ω-constraints:

$$\begin{aligned} &cost(solution_1, MALA) \\ &\quad = \sum_{i=1}^{3} w_i \cdot (min_{f_\ell \in F_i} d_i(f, solution^{q_i}) \\ &\quad = 1 \cdot (|10 - 7| + |10 - 5| + |10 - 4|) \\ &\quad + 0.5 \cdot (|100 - 95| + |500 - 530| + |80 - 60|) \\ &\quad + 0.8 \cdot (|0 - 19| + |20 - 15| + |0 - 12|) = 70.3. \end{aligned}$$

4 Committee Scoring Rules Using MALA

Next, we will demonstrate the versatility of the MALA model for various voting rules. Our model is specifically designed for democratic settings, making it applicable to a wide range of use cases, including committee elections. One notable example is the Democratic Playlist Editing problem, which we will discuss in detail later.

Approval Voting and Borda Count: In the case of selecting a committee, the ideal approval score or Borda count would be achieved if all "n" agents voted or ranked the same "m" candidates, indicating a consensus. To model this, we represent the perfect approval score and Borda count as a vector of size "k," where each candidate's score is a constant value representing the number of votes or ranking positions they received from the "n" agents. Thus, this vector serves as an upper bound for calculating the cost of a given solution. Formally, we define:

- Each candidate's voting score is given by: voting score$(c_1) = \sum_{i=1}^{n} c_{i,1} \cdots$, voting score$(c_m) = \sum_{i=1}^{n} c_{i,m}$, and $c_{i,j} \in 0, 1$ or $c_{i,j} \in 0, 1, \cdots, k$ indicates the approval or ranking of candidate $j \in [m]$ by agent $i \in [n]$.
- $F_1 = f_1$, where in approval voting $f_1[s] = n$, and in Borda count $f_1[s] = n \cdot k$, for all $s \in [k]$.
- $d =$ any metric distance, such as ℓ_1 or ℓ_2.

PAV Score: The PAV (Proportional Approval Voting) score assigns scores to candidates based on the number of votes they receive, with the goal of allocating seats to candidates proportionally to their support, while also considering the number of available seats. In our model, each agent approves or disapproves of certain elements. We introduce a PAV cost constraint to represent the "loss" of the potential PAV score. This model can also be extended to OWA-based rules [13]. For each agent $v_j \in v_1, \cdots, v_n$ voting on elements $c_1, \cdots, c_m$, we create an Ω-constraint, and its distance will be the "PAV-cost," reflecting the "loss" of the potential PAV score. The following definitions apply:

- $q \in [m]$ corresponds to every agent.
- $solution[s]^j = 1$ if agent j approves of candidate i and 0 otherwise, for all $solution \in C^k$.
- $F = \{f_1\}$, $f_1 = \{[1]^k\}$.
- $$d(x, y) = \begin{cases} \sum_{j=1}^{k-\ell_1(x,y)} \frac{1}{j}, & \ell_1(x, y) < k \\ 0, & else \end{cases}$$

Given such individual agent Ω-constraints, adding a weight vector of all "1" results in the realization of PAV as an instance of MALA.

5 Computational Analysis

To study the computational complexity of MALA we consider its decision variant, in which we are given an additional input that is the maximum total cost for which existence of a solution above is to be decided. First, we observe that, following the formulation described above of PAV as a MALA instance, NP-hardness is established [3]. Next we show that MALA is also NP-hard even with only 2 constraints.[1]

[1] There is a delicate point here with respect to the representation of the input. We discuss consequences of this to different applications in Sect. 9, but here, for the formal hardness statement and proof, it is crucial to describe the representation of the input that affects the length of the input. So, in particular, it is sufficient to assume that the F vectors in the input are given explicitly, while the d metrics are given as black-boxes of length $O(1)$.

Theorem 1. *The MALA Decision Problem is NP-hard.*

Proof. We provide a reduction from the subset-sum problem [21], where an instance X containing x_i, $i \in [n]$ is a "yes-instance" if a subgroup $X' \subset X$ exists that satisfies $|X'| = \frac{n}{2}$ and $\sum_{x_i \in X'} x_i = \frac{\sum_{x_i \in X} x_i}{2} = \frac{B}{2}$. To build an input for the MALA decision problem given the subset-sum input, we set the following Ω-constraints:

- **q** - We have a MALA problem with two identical attributes: q_1, q_2 having $c_i^1 = c_i^2 = x_i$ for all i.
- **F** - We set $F_1 = f_1$ where $f_1^j = 0$, and $F_2 = f_2$ where $f_2^j = M$, for all $j \in [k]$ (vector's length), where $M = \sum_{x_i \in X} x_i$.
- **d** - We create two distances based on the ℓ_1 distance between two given vectors:

$$d_1(vector_1, vector_2) = \begin{cases} 1, & \ell_1(vector_1, vector_2) > \frac{B}{2} \\ 0, & \ell_1(vector_1, vector_2) \le \frac{B}{2} \end{cases}$$

$$d_2(vector_1, vector_2) = \begin{cases} 1, & \ell_1(vector_1, vector_2) > \frac{M \cdot n}{2} - \frac{B}{2} \\ 0, & \ell_1(vector_1, vector_2) \le \frac{M \cdot n}{2} - \frac{B}{2} \end{cases}$$

- **Weights**- We set $w_1 = w_2 = \frac{1}{2}$.

We formulate the MALA model as a decision problem - Given all candidates C and: Ω-constraints, we want to determine if there exists a subset $X' \subseteq X$ such that $\sum_{i=1}^{2} cost(X', \Omega_i) = 1$. If such a subset X' exists, then $\sum_{x_i \in X'} x_i = \frac{\sum_{x_i \in X} x_i}{2} = \frac{B}{2}$. Conversely, if $c_{i,1} = c_{i,2}$, for all i, $F_1 = F_2$, $d_1 = d_2$, and $X = q_1 = q_2$, we can reduce the problem to the subset sum problem where $|X| = n$. In this case, if there exists a subset $X' \subseteq X$ such that $|X'| = \frac{n}{2}$ and $\sum_{x_i \in X'} x_i = \frac{\sum_{x_i \in X} x_i}{2} = \frac{B}{2}$, then $\sum_{i=1}^{2} cost(X', \Omega_i) = 1$. Thus, the MALA decision problem is NP-hard even with just two attributes, contradicting our polynomial assumption of the problem.

Observation 1. *It is worth noting that there are some polynomially-computable cases of MALA. For example, when the task is to choose an ordered sub-list of musical tracks that contains tracks with the most votes and is ordered by a non-ascending or non-descending tempo (speed of the track).*

6 Algorithms

Since our problem has been shown to be NP-hard, we have developed several heuristic algorithms to obtain a good solution within a reasonable time frame. In this study, we have chosen to test two main algorithms: Genetic and Simulated Annealing. Both of these heuristics are suitable for solving similar combinatorial optimization problems that have complex search spaces and multiple objectives.

To simplify the testing process, we make the assumption that each vector family, denoted as F, which describes optimal behavior over time or sequence of some feature q, has a finite set. (This assumption will be discussed further in the Outlook section.) In each of the heuristics explained here, we aim to find the ordered sub-group of k elements out of a total group of m elements that would minimize the weighted summation of costs defined by Ω-constraints.

Genetic Algorithm - The algorithm begins by generating an initial population of candidate solutions, each represented as a set of parameters. The cost of each solution is calculated based on the problem at hand. The population is then sorted based on the descending cost, and the best solution is identified. In each iteration, the algorithm updates the population size using adaptive population sizing techniques, which adjust the number of solutions in the population based on their performance. The crossover and mutation probabilities, which control the exploration and exploitation of the search space, are also updated adaptively [24]. New solutions are generated through crossover or mutation operations, ensuring that there are no repetitions among the solutions. The cost of each new solution is calculated, and the population is sorted again. If the best solution in the new population has a lower cost than the current best solution, it is updated accordingly. The algorithm continues iterating until the specified total run time is reached. Finally, the best solution found throughout the iterations is returned as the output of the algorithm.

Simulated Annealing - The algorithm starts by generating an initial random solution. It then sets the initial cooling rate and temperature, which are problem-dependent and determine the exploration-exploitation balance. Additionally, a number of iterations for a random start are specified to allow for more diverse exploration. During each iteration, the algorithm calculates the cost of the current solution and compares it to the minimal cost found so far. If the current cost is lower, the minimal cost is updated accordingly. The algorithm also computes the probability of accepting a worse solution based on an adaptive cooling rate [19]. If the probability allows accepting a worse solution, the minimal cost is updated. To explore the search space, a random element and index are generated. If the generated element is included in the solution, it is swapped with the element at the generated index. Otherwise, the element at the generated index is replaced with the generated element. The temperature is decreased using the cooling rate, gradually reducing the exploration ability of the algorithm. The process continues until the specified total run time is reached. Finally, the best solution found throughout the iterations is returned as the output of the algorithm.

7 The Democratic Playlist Editing Problem

The focus now shifts to the democratic playlist editing issue, specifically the problem of creating a playlist with a specific logic or theme. This problem, known as the Democratic Automated Generation Playlist Problem, involves a group of friends attempting to create a playlist. In this section, we will discuss this problem and its formulation using MALA. The Automated Generation Playlist Problem involves creating a playlist by selecting tracks from a given list based on their musical attributes, such as scale, key, tempo (beats per minute), time signature, loudness, valence (optimism), danceability, and more. The Democratic Automated Generation Playlist extends the original problem by allowing a community to vote on whether to include tracks in a playlist. Playlist editors and critics usually look for the following three measures in a playlist [17,26]:

1. **Coherence of tracks** - Listeners tend to like playlists with tracks that correspond (musically and lyrically) to each other homogeneously. In order to model the coherence of a feature using MALA, we must find the vector of some constant value. It will serve as a reference to measure the extent of the coherence of the attribute's behavior over time or over a sequence of events. Thus, we need to search all the positive constant vectors in order to find the most suitable one for $Solution[q]$ over time. Formally:
   - q corresponds to c_i^q.
   - $F = \{f_1, f_2, \cdots, f_t\}$ having $f_i[s] = p, i \in [t], p \in \mathbb{R}$, for all $s \in [k]$.
   - $d = \ell_1$ or ℓ_2
2. **Smooth transitions between two consecutive tracks** - Smooth transitions are highly valued by users as they provide a seamless progression of attributes, whether in sequence or over time. The primary objective of these transitions is to maintain a consistent flow while minimizing abrupt changes in attributes value's direction over time.
   - $q_{j \in 0,\cdots,k-1}$ address a particular attribute, and e, the maximum explicit number of direction changes it values can undergo.
   - $F = \{f_1, f_2, \cdots, f_t\}$ having $f_i = \mathbb{R}^k, i \in [t]$, such that $0 \leq \sum_{s=2}^{k-1} \delta(sign(f_i[s] - f_i[s-1]), sign(f_i[s+1] - f_i[s])) \leq e \in [k-1]$, $\delta(x, y) = 0$ if $x = y$, else $\delta(x, y) = 1$.
   - $d = \ell_1$ or ℓ_2.

   This modeling of smooth transitions as well as coherence, is in contradiction to our initial assuming of a finite set of functions F. Because the modeling of these qualities depend on the attribute's behaviour and the actual suggested tracks; we do not have a way of predicting what would be the ideal set of functions, and so we can only approximate by giving a few reasonable functions (a finite set of vectors).
3. **Diversity** - Like many democratic parliaments that ensure seats for different groups in society (such as women and minorities) to maintain a proportional representation of society, a playlist should aim for representation of different attributes of the given tracks, including genres (e.g., jazz, pop, rock, reggae, Brazilian, Afro-beat, Indian), scales (e.g., major and minor), time signatures (e.g., even and odd beat division of a track), and ranges of tempo (e.g., Largo (very slow), Adagio (slow), Andante (medium-slow), Moderato (medium), Allegro (medium-fast), and Presto (fast)).While the two measures

before where based on numerical valued attributes, this measure is based on categorical valued attributes. Formally:

- q corresponds to $c_i^q \in [r]$ having $r = |P^q|$, and P^q to be the set of categories associated with attribute q.
- $F = \{f_0, f_1, \cdots, f_t\}$, $f_i = \{f_i[s] | s \in k, f_i[s] \in [r], freq(f_i[s], f_i) = \pi_{p \in P^q}\}$ having π_p to be the optimal proportion of each category of attribute q.
- $d = \ell_1$ or ℓ_2

In conclusion, the Democratic Automated Generation Playlist Problem involves selecting an ordered subset of k tracks from a set of total m musical tracks that satisfy both the user's own taste and some musical constraints.

Example 2. A community of 3 music lovers decided to collaboratively edit a three sized democratic playlist (i.e., k = 3) out of a 5 sized list of tracks (i.e., candidates). Each track has the following musical features and meta-data. After each member of the group has selected three tracks from the given list of tracks we assemble them into a binary-requirements-matrix where 1 represents if a track i was chosen by agent j:

Number	Name	Artist	Album	BPM	Scale	Loudness
1	"I Wish"	Stevie Wonder	"Songs In the Key of Life"	106	B flat minor	−10.4
2	"Thriller"	Michael Jackson	"Thriller"	139	E flat minor	−3.7
3	"So What"	Miles Davis	"Kind of Blue"	138	C major	−17.27
4	"Oye Como Va"	Carlos Santana	"Abraxas"	128	G major	−13.21
5	"Reelin' in the Years"	Steely Dan	"Can't Buy a Thrill"	135	D major minor	−17.34

	Track 1	Track 2	Track 3	Track 4	Track 5
agent 1	0	1	1	0	1
agent 2	0	1	1	1	0
agent 3	1	0	1	0	1

We aim to create a 3-track playlist that maximizes: (1) social welfare of the listeners while maintaining (2) coherence in beats per minute (BPM) among the tracks. This coherence ensures that all tempos of tracks are close to either 120 (medium tempo) or 180 BPM (fast tempo). We will explore both a potential "good" solution and a potential "bad" (or "expensive") solution for the given example, using distance ℓ_1 and measured by explicit F vectors:

- $F_{\text{welfare}} = F_1 = \{f_1\}$ where $f_1[1] = 3, f_1[2] = 3, f_1[3] = 3$
- $F_{\text{BPM coherence}} = F_2 = \{f_2, f_3\}$ where $f_2[1] = 120, f_2[2] = 120, f_2[3] = 120$, and where $f_3[1] = 180, f_3[2] = 180, f_3[3] = 180$.

Social welfare is given the weight w_1=5 and BPM coherence of tracks is given the weight of w_2=0.5.

- Good solution: $\sum_{i \in [2]} \text{cost}([track5, track3, track2], \Omega_i) = 5 \cdot \text{cost}([track5, track3, track2], \Omega_1) + 0.5 \cdot \text{cost}([track5, track3, track2], \Omega_2) = 5 \cdot (|2-3| + |3-3| + |3-2|) + 0.5 \cdot (|135-120| + |138-120| + |139-120|) = 36.$
- Worst solution: $\sum_{i \in [2]} \text{cost}([track1, track2, track4], \Omega_i) = 5 \cdot \text{cost}([track1, track2, track4], \Omega_1) + 0.5 \cdot \text{cost}([track1, track2, track4], \Omega_2) = 5 \cdot (|1-3| + |2-3| + |1-3|) + 0.5 \cdot (|106-120| + |139-120| + |128-120|) = 45.5.$

8 Experimental Analysis

To evaluate the quality of the heuristics discussed earlier, we conducted a simulation of the Democratic Playlist Editing problem modeled as a MALA-optimization problem.

8.1 Experimental Design

We generated ten 700-track playlists from Spotify's "Top 10,000 Songs Of All Times" playlist[2]. For each instance, we applied the discussed algorithms to find an ordered sub-list of 250 tracks that minimized the cost within a ten-minute run. The heuristic parameters were set as follows: simulated annealing and sequential simulated annealing with a temperature of 1000°, an initial cooling rate of 0.003, and a random start every 1000 iterations. Genetic Algorithm was initialized with an initial crossover probability of 0.85, an initial population size of 100, and a maximum population of 5000. The costs were normalized using a 100-random algorithm, which generated 100 random permutations and selected the one with the minimal cost. This allowed us to calculate the average cost per minute for all 10 instances.

We selected three audio features from Spotify's API out of 13 available features [10]: Energy (0.0 to 1.0 score representing intensity and activity), Tempo (measured in beats per minute), and Danceability (0.0 to 1.0 score representing suitability for dancing). These features were chosen for their significant impact on playlist formation.

Approval scores were added to each track using an artificial society of 20 agents. An algorithm was used to generate a list of 700 integers, ensuring a sum of 5000. The algorithm randomly and uniformly generated approval ballots, divided the remaining sum by the remaining iterations, and updated the list accordingly. If there was a remaining sum, it was distributed incrementally to randomly selected indices until reaching zero. Next, we generated 10 families of functions (represented as vectors), each one containing 50 different 250-sized vectors. These functions within each family can be divided into two types:

[2] https://open.spotify.com/playlist/1G8IpkZKobrIlXcVPoSIuf.

1. We aimed for an ideal behavior of certain attributes over time, specifically smooth transitions for BPM and energy, with 1–3 direction changes to ensure smoothness along the playlist. This resulted in a total of 6 families of functions.
2. We defined an ideal static value to measure the coherence of attributes, where changes in direction are 0 and the slope (i.e., the size of change between two consecutive tracks) is 0. This applies to BPM, danceability, energy, and approval score, with the latter only including one vector/function of "all 20s".

We developed an algorithm that takes in the minimum number of direction changes, a range for the first variable in the vector (distributed uniformly), a range for the slope between consecutive variables in the vector (distributed uniformly), and the minimum and maximum values to set the range of legal values in the vector. The algorithm generates the initial direction (+ or −) uniformly and k indexes in the range of 2–248, where k is the number of direction changes and $index_i - index_{i+1} \geq 2$. Whenever the algorithm reaches one of the indexes, or if the value of the current variable is greater than the maximum value or smaller than the minimal value, a change of direction will occur. Finally, we set the distance d as ℓ_1 and generated a weight for each Ω-constraint combination, including:

1. Energy weights for 0, 1, 2, and 3 changes of direction over time. These weights were generated uniformly between 1 to 3, taking into account the involvement of energy in creating a playlist.
2. Tempo weights for 0, 1, 2, and 3 changes of direction over time. These weights were generated uniformly between 0.0001 to 0.001, as tempo was deemed to be a feature of less importance in our simulation.
3. Danceability weight for 0 changes of direction over time. These weights were generated uniformly between 3 to 4, aiming to create a highly danceable playlist.
4. Approval score weight for 0 changes of direction over time. These weights were generated uniformly between 4 to 5, with the intention of creating a playlist of popular tracks.

We also tested a greedy algorithm as an additional reference, wherein we preformed a local search for the most suitable track to fit into every location in the list. Finally, we performed another simulation to measure the time required for the simulated annealing algorithm to reach half the value of zRandom on 10 different instances of different sizes (selected randomly), ranging from 30 tracks to 240, in jumps of 30.

8.2 Results and Analysis

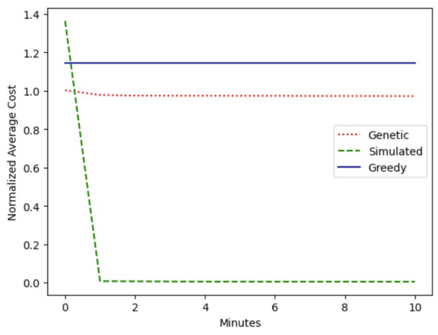

Fig. 1. Algorithms's Cost Over a 10 min Run.

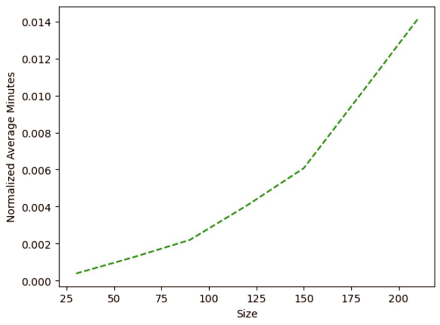

Fig. 2. SA Average Time VS Size of Playlist - until $SA\ cost = 0.5 \cdot (zRandom\ cost)$.

The results in Fig. 1 show that the simulated annealing algorithm achieved the lowest normalized cost quickly, with genetic algorithms performing worse. Figure 2 confirms that the simulated annealing algorithm takes longer to reach half the value of zRandom as the instance size increases. These results suggest that the simulated annealing algorithm explores a wider range of solutions compared to the more restrictive genetic algorithm, which converges slower due to maintaining parental order. Alternatively, the suboptimal tuning of initial parameters, such as adaptive crossover probability and population size, may explain the genetic algorithm's performance and could be improved with additional tuning techniques. These results suit the findings of Piotr Faliszewski et al [11] on effective heruistics for committee scoring rules.

9 Outlook

In future research, we suggest exploring the following directions:

- **Logical constraints:** Currently, we assume finite family functions F, limiting the model's applicability. Extending the model with logical predicates (P) as Ω-constraints could overcome this limitation and allow for the formulation of desired attribute behaviors over time.
- **Time-axis:** In playlist editing, considering variable song durations would require treating functions F as continuous functions, with the optimization problem involving a constraint on the summation of selected elements' time. This change from our current assumption of equal song durations can offer added flexibility to the process.
- **Several dimensions:** While our model is one-dimensional, considering a variant with multiple dimensions could be valuable for democratic editing of graphical illustrations or other multidimensional scenarios.
- **Heuristic solutions:** While we employed three standard heuristics, exploring alternative algorithmic solutions, such as constructing a linguistic model tailored to the MALA instance using Ngrams [25], may yield improved results.
- **Computational analysis:** Investigating both tractable and intractable special cases of MALA can enhance our understanding of the model's applicability and provide insights into its computational complexity, in particular by examining different types of distances in the Ω constraints.

References

1. Anonymous personal communication. Continuous preference aggregation in one dimension (2023)
2. Arrow, K.J., Sen, A., Suzumura, K.: Handbook of Social Choice and Welfare, vol. 2. Elsevier, Amsterdam (2010)
3. Aziz, H.: Proportional representation in approval-based committee voting and beyond (2018)
4. Aziz, H.: A rule for committee selection with soft diversity constraints. Group Decisi. Negot. **28**(6), 1193–1200 (2019). https://doi.org/10.1007/s10726-019-09634-5
5. Bossert, W., Peters, H.: Multi-attribute decision-making in individual and social choice. Math. Soc. Sci. **40**(3), 327–339 (2000)
6. Bredereck, R., Faliszewski, P., Igarashi, A., Lackner, M., Skowron, P.: Multiwinner elections with diversity constraints. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)
7. Burke, R., Mattei, N., Grozin, V., Voida, A., Sonboli, N.: Multi-agent social choice for dynamic fairness-aware recommendation. In: Adjunct Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization, pp. 234–244 (2022)
8. Burke, R., Mattei, N., Grozin, V., Voida, A., Sonboli, N.: Multi-agent social choice for dynamic fairness-aware recommendation. In: Adjunct Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization, UMAP 2022 Adjunct, pp. 234–244. Association for Computing Machinery, New York, NY, USA (2022)

9. Darmann, A., Elkind, E., Kurz, S., Lang, J., Schauer, J., Woeginger, G.: Group activity selection problem. In: Goldberg, P.W. (ed.) WINE 2012. LNCS, vol. 7695, pp. 156–169. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35311-6_12
10. Duman, D., Neto, P., Mavrolampados, A., Toiviainen, P., Luck, G.: Music we move to: Spotify audio features and reasons for listening. PLoS ONE **17**(9), 1–18 (2022)
11. Faliszewski, P., Lackner, M., Peters, D., Talmon, N.: Effective heuristics for committee scoring rules. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)
12. Faliszewski, P., Skowron, P., Slinko, A., Talmon, N.: Multiwinner voting: a new challenge for social choice theory. Trends Comput. Soc. Choice **74**(2017), 27–47 (2017)
13. Faliszewski, P., Skowron, P., Slinko, A., Talmon, N.: Committee scoring rules: Axiomatic characterization and hierarchy. CoRR, abs/1802.06483 (2018)
14. Goldsmith, J., Lang, J., Mattei, N., Perny, P.: Voting with rank dependent scoring rules (2014)
15. Gupta, S.K., Kyparisis, J.: Single machine scheduling research. Omega **15**(3), 207–227 (1987)
16. Gupta, S., Jain, P., Saurabh, S.: Well-structured committees. In: Bessiere, C., (eds.) Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, pp. 189–195. International Joint Conferences on Artificial Intelligence Organization, vol. 7 (2020). Main track
17. Ikeda, S., Oku, K., Kawagoe, K.: Analysis of music transition in acoustic feature space for music recommendation. In: Proceedings of the 9th International Conference on Machine Learning and Computing, ICMLC 2017, pp. 77–80. Association for Computing Machinery, New York, NY, USA (2017)
18. Israel, J., Brill, M.: Dynamic proportional rankings. arXiv preprint arXiv:2105.08043 (2021)
19. Karabin, M., Stuart, S.J.: Simulated annealing with adaptive cooling rates. J. Chem. Phys. **153**(11), 114103 (2020)
20. Karger, D., Stein, C., Wein, J.: Scheduling algorithms (2010)
21. Karp, R.M.: Reducibility among Combinatorial Problems, pp. 85–103,D US, Boston, MA (1972)
22. Lang, J., Skowron, P.: Multi-attribute proportional representation. Artif. Intell. **263**, 74–106 (2018)
23. Lian, J.W., Mattei, N., Noble, R., Walsh, T.: Using order weighted averages to assign indivisible goods. In: The conference Paper Assignment Problem (2018)
24. Lobo, F.G., Lima, C.F.: A review of adaptive population sizing schemes in genetic algorithms. In: Proceedings of the 7th Annual Workshop on Genetic and Evolutionary Computation, GECCO 2005, pp. 228–234. Association for Computing Machinery, New York, NY, USA (2005)
25. McFee, B., Lanckriet, G.R.: The natural language of playlists. In: ISMIR, vol. 11, pp. 537–541 (2011)
26. Pauws, S., Eggen, B.: Realization and user evaluation of an automatic playlist generator. J. New Music Res. **32**(2), 179–192 (2003)
27. Sikdar, S.K.: Optimal Multi-Attribute Decision Making in Social Choice Problems. Rensselaer Polytechnic Institute (2018)
28. Skowron, P., Lackner, M., Brill, M., Peters, D., Elkind, E.: Proportional rankings, Markus Brill (2016)