Resume

The paper proposes a novel system, OurTunes: a collaborative playlist system designed to produce fair music playlists for groups while minimizing required input and ensuring smooth transitions.

Each user can join an OurTunes session with their own Spotify playlists, allowing users to align their preferences with the social context. OurTunes analyzes each playlist, plotting all songs in a high-dimensional feature space based on metadata such as Mood and Genre supplied by Cyanite.ai. Using this data, similar music is grouped closely, ensuring smooth transitions. The playlist is then built iteratively, starting with clustering all songs using k-means to find potential starting points. Each subsequent song is chosen from the k nearest neighbors using a k-d tree, with selection probabilities weighted to favor currently underrepresented users (i.e., those whose songs have been chosen less often). This dynamic weighting ensures Fairness.

To give users more control of the sound, OurTunes includes a "Wish song" feature: users may request a desired track from the entire Spotify catalog at any time. As new Wish songs are added, an ordered sequence of Wish songs to be played is determined, first moving towards the biggest cluster of Wish songs and then the shortest path through the remaining Wish songs. Between Wish song fulfillments, the system weighs songs closer to the next Wish song higher, ensuring that all Wishes can be accommodated. Users can also view how the Genres will change as the playlist progresses in the app.

In extensive simulations using 35 real "party" playlists (group sizes 2–35, 1,000 runs each), OurTunes outperformed both Round robin and Shuffle on Cumulative Distance (sum of all distances between songs) and Satisfaction (average distance of each chosen song by OurTunes to users' closest song), while achieving better Fairness for larger groups. A real-world user study with seven participants in a three-hour party setting complemented these findings: participants reported feeling heard without excessive interaction, experienced smooth genre and mood continuity (predominantly electronic dance/energetic), and valued the Wish song feature for its balance of control and automation, while also requesting the ability for either more Wish songs or more insight into when their Wish songs would be played.

OurTunes demonstrates that combining clustering, traversal to nearby songs, and user-weighting can produce socially satisfying, low-effort group playlists. Future work will refine smaller implementation issues, promote minority tastes or wishes to enhance Fairness further, and explore how different group sizes and how well each user knows one another can affect the requirements for this collaborative playlist system.

OurTunes: Improving Collaborative Consensus Playlist Making

Christian Søndergaard Thor
cthor20@student.aau.dk
Aalborg Universitet
Aalborg, Denmark

Daniel Dencker Jepsen
djepse20@student.aau.dk
Aalborg Universitet
Aalborg, Denmark

Lucas Bjørn Tranum
ltranu20@student.aau.dk
Aalborg Universitet
Aalborg, Denmark

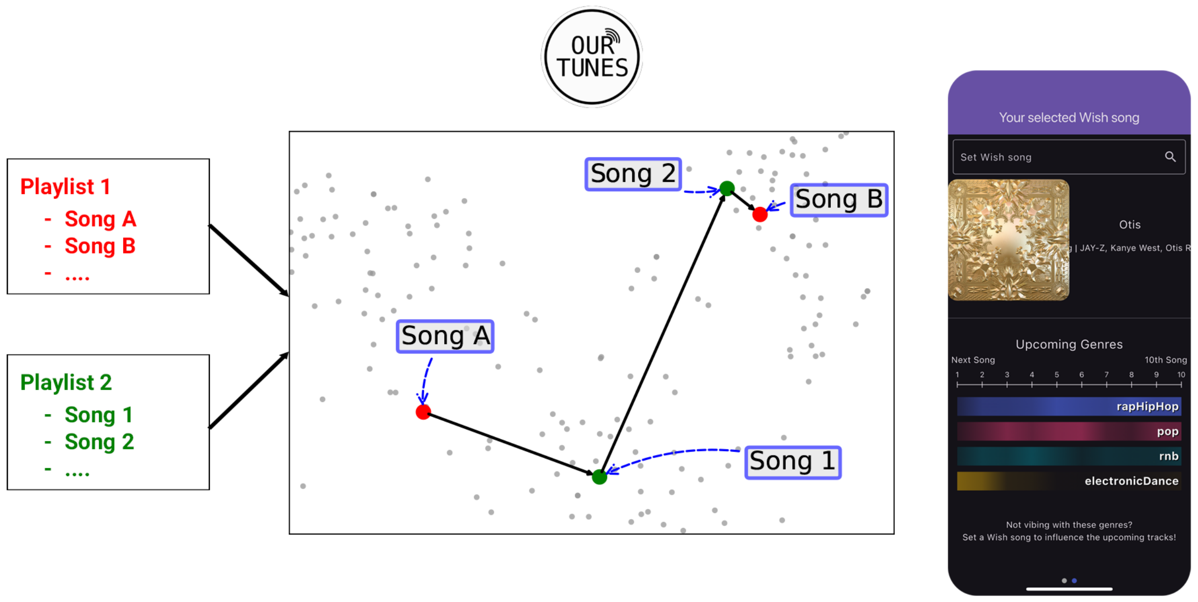


Figure 1: A path through the OurTunes song space, derived from multiple playlists. The OurTunes app is shown to the right.

Abstract

Collaborative playlist creation in group settings often involves challenges in balancing individual preferences and achieving group consensus. Most existing systems rely heavily on user interaction or fail to fairly represent all participants, making the process cumbersome and suboptimal. In this paper, we propose OurTunes, a system that generates group playlists by leveraging individual preferences and implicit consensus mechanisms. OurTunes clusters songs using k-means to generate potential starting points in the feature space alongside using k-nearest neighbors (KNN) to traverse the feature space based on the users. The system incorporates weighting techniques to ensure fair representation across participants while maintaining variety and cohesion.

Quantitative experiments have been used to evaluate the system on metrics such as Fairness, Cumulative distance, and Satisfaction, demonstrating improvements compared to Shuffle and Round robin algorithms. These findings indicate that OurTunes can facilitate collaborative playlist creation with reduced manual input, improved group satisfaction compared to the two aforementioned algorithms, and better Fairness compared to Shuffle. A user test was also conducted that showed positive results but with points to optimize for future iterations.

Keywords

Music, Group Recommendation, Playlist, Collaborative playlist

1 Introduction

Collaborative playlists/sessions have become increasingly popular over the years, and the driving factor here seems to be both social and practical in purpose[19]. One of the primary examples of a collaborative playlist/session is listening to music at a party. Creating party playlists is a task that many partake in, either as a group or as an individual. Companies like Spotify have created features to alleviate some problems with making playlists as a group. Features such as collaborative playlists and Spotify Jam[25] help group participants interact with each other through music sharing. However, this requires explicit choices made by users, which can be cumbersome[19]. Furthermore, as user preferences change a lot in group contexts, it can be hard to make a satisfying playlist. These collaborative sessions often end with only a few people contributing to the playlist, leaving many preferences from other people off the table. Lastly, listening to self-chosen music positively affects people, so the more they can exert their preferences in a group setting, the better the listening experience[14]. To alleviate some of these downsides, systems have been implemented that provide users with the ability to vote on preferred songs and, through that, have explicit consensus on the playlist[29][17]. However, this comes at a cost, as users must constantly micro-manage and vote to have their preferences heard.

However, there are ways to circumvent explicit consensus. One way is to read a crowd, for example, using visual or auditory analysis, whether they are satisfied[22][21]. However, this kind of data is hard to capture and contextualize. Many music recommendation systems use other data such as music metadata[26][1]. Music metadata is used to recommend new songs to users through mood or more tangible information such as artists and genre. The recommendations can be based on some pathing through a similarity space or some designated area in that space[26]. The problem with recommendation systems is that they are mostly tailored only to the individual user at this moment. Therefore, the recommendations are highly individual and cannot be used in a group setting in a valid way[16].

This paper aims to achieve a fair and satisfying group consensus based on individual preferences with minimum input. Our solution allows users to submit playlists of their choice as a basis for individual preferences, which creates a big dataset of possible songs from the users. As a side effect, users can also choose a playlist that fits into the context of the session, so it fits the session, i.e., choosing a party playlist if they are attending a party. OurTunes then chooses a path through this space using weighting techniques to ensure that the music type does not change too much from song to song and that the distribution of played songs is fair. OurTunes also chooses songs with similar sound profiles that users agree on more often rather than taking songs that only represent a minority of a session. We used our own quantitative metrics to quantify whether our algorithm is meaningfully better at a glance in picking more closely related music while being more fair to all participants. Lastly, a user test was also conducted for the system, in which participants would participate in a real party setting using their own playlists, experiencing the system in real-time and influencing the path taken through the song space by stating Wishes in a mobile app. Initial feedback suggests that our method has strong potential to create fair and enjoyable playlists for medium-sized groups of acquaintances. OurTunes was positively received by the test participants; however, there was room for improvement, namely, in how fair the system was, to increase the social aspect even further.

2 Related Works

Early work related to collaborative music listening started with Flytrap[8] and r-MUSIC[29] in 2002 and 2004, respectively. These present two avenues for facilitating collaborative music sessions. Flytrap does it entirely implicitly, whereas r-MUSIC is completely explicit. Flytrap collects data from user listening patterns using genre and artists as their focal point and then suggests songs similar in genre to the last one played that provide the most satisfaction. They derive satisfaction by having an agent for each user that votes on their behalf on songs based on the data collected and then, using the probability distribution of the votes, select a song. Meanwhile, r-MUSIC takes the other path, where they let the users suggest songs, and then they vote on the songs suggested that they want to hear. R-MUSIC has some complexity in how the weights of the votes work, where users who suggest less liked songs, measured by how many other users vote on their songs, have a low weight and vice versa.

The Smart Party[13][12] presents a collaborative music listening experience, where different rooms play different music according to the music preferences of the users in that specific room. While only tested in simulations, the results showed that grouping people with shared tastes improved satisfaction and fairness. However, this requires multiple rooms with individual speakers, each requiring a dedicated router and a special handheld device for each guest's preferences, creating a theoretically personalized experience but an impractical setup for the host.

Systems that focus on music exploration by visualizing users' music have also been created[3]. One such system is proposed by Beatrix et al., 2015[26]. Their proposed system visualizes songs on a map for the user by fetching a set of features such as mood, genre, style, and vocals. Then, employ dimensionality reduction on these features using T-SNE[27]. The dimensionality-reduced set of songs is then visualized in a 2D map, which shows how closely related songs are in their features and, by extension, in their sound. They found that users generally preferred to stay in similar areas rather than move through the space in a path. This indicates that users prefer songs that are similar to each other rather than taking an exploratory approach to music types when listening by themselves.

Another system that deals with collaborative music sessions is SRMAC[28]. This system uses genres and audio features fetched from Spotify to create group-based playlists. For genres, a graph is created based on all the different artists present on the user's playlists, and their respective genres are created. Then, based on how often a link between an artist and a specific genre is repeated, the weight of that genre will be higher. All other features were weighted depending on the context; for example, the danceability feature is weighed higher in "party" contexts. All these respective weighted features are then used to query Spotify's recommendation API to find songs that fit these weights. Their results indicated a potential issue with the familiarity of the songs being relatively low, which could be a positive or a negative depending on the purpose of the session. Lastly, they tested a feature where users could set the desired values of the Spotify features. Here, the general feedback was more positive, as it gave users greater control over the system.

In the Moodplay paper[2], a system is proposed to create a list of artists' recommendations based on the similarity of users' moods to specific artists. An algorithm proposed in this paper is a trail-based algorithm. This algorithm works by calculating the mood distance, which is based on metadata, and gives a set of weighted moods for each of the respective artists. Here, this mood distance is then weighted by users' previous trail marks, where each trail mark is generated depending on which artist the user chooses to listen to next. The users then choose which artist to listen to next. As a result, this algorithm will create a dynamic path through the mood space, from one mood to another. In the evaluation of this specific algorithm, users generally preferred navigating their music collection more freely in the mood space than using the trail-based algorithm. While the algorithm is implicit, the system as a whole is more explicit as it requires the user to choose the next artist after each time.

Another way to facilitate collaborative music sessions is using user ratings[11]. A paper that employed this found great success in the hit rate of their algorithm in their simulation study. However, as this requires explicit knowledge of liked and disliked songs, it is time-consuming to use and rare to have a user base that generally rates their music a lot. This paper also used a large dataset consisting of 20000 users, and these were clustered into groups, which consisted of 1500 users on average, which is far more than what this paper aims to cater to.

Lastly, some systems attempt to determine group preferences and play music based on these preferences[8][7]. However, these systems purely utilize the users' listening history to find similar music. While this can represent a user's music preferences quite well, it does not necessarily capture that user's preference in all social contexts that can involve music. *Reflektor* by Bauer et al., 2018 highlights how social contexts influence the chosen music, e.g., users picking jazz music for a card game night[4]. User opinions also change depending on whether they are in a group context compared to alone [16]. We aim to have users create playlists instead, allowing users to state their preferences in a way that aligns with their interpretation of the social context.

To end this section, we examine how people experience and partake in collaborative playlists. One paper did a survey where they tried to illuminate this, followed by interviews with a select few of the survey participants [19]. The researchers used a collaborative playlist framework for collaborative playlists to divide them into 3 categories: Practical, social, and cognitive purpose[20][15]. Firstly, practical purpose means that the collaborative playlist was constructed to serve some practical function, i.e., to a specific party like a school dance. Secondly, social purpose means sharing music and connecting with other users; thirdly, cognitive purpose is to experience new music. They reported that the purpose of the collaborative playlist for the participants was primarily practical and social. The reason a collaborative playlist was a participant's favorite showed the same pattern.

Regarding the favorite collaborative playlists of the participants, the participants said that *Recommended by streaming/AI* was severely less represented compared to other music content descriptors. The reason why it was lower than other content descriptors like *Is/Feels personalized* was that it did not fulfill the social purpose enough. The researchers surmised that the personal feeling conveyed by the songs chosen by other participants in collaborative playlists can help facilitate successful curation. Lastly, the interview of the participants illuminated some key factors in collaborative playlist curation. One of the reasons why creating a successful collaborative playlist is cumbersome is the inertia of creating one; it takes much work to get it off the ground. Many of them fail, according to participants, because of the initial work that needs to be done, coupled with the fact that only a few in a collaborative playlist actually participate in the curation. This was echoed by the survey, which showed that the action of *contribute* was often lower than the actions of *Listen/play alone* and *Listen/play with others*, especially if the participant was a non-initiator of the collaborative playlist. Therefore, many preferences are just left off the table when it comes to curation, and users simply listen instead of contributing to the curation, i.e., social loafing and the free-rider problem.

3 Initial Data Exploration and the OurTunes Algorithm

There are multiple problems that we try to solve when making collaborative sessions or collaborative playlists. One of them is that starting a collaborative playlist takes much effort, which So Yeon Park and Blair Kaneshiro (2021) described as inertia[19]. Overcoming the workload problem to get the playlist off the ground seems to be an essential part that systems like r-MUSIC[29] or something like Spotify Jam suffer from, as it requires a fair amount of input. The other problem is how personable the playlist is. As seen in the survey[19], people prefer music that fits the content descriptor of *Is/Feels Personal*, as people appreciate the effort of personally chosen songs, compared to the content descriptor of *Recommended by streaming/AI*, which was underperforming compared to most content descriptors. Therefore, a system that uses AI to recommend new music based on those preferences might also not be the solution we are looking for. People mostly create collaborative playlists based on the social and practical purposes, and the cognitive purpose seems to be a minor incentive to create collaborative playlists.

We want to create a system that solves the inertia problem while creating something personal. We want to support users in doing collaborative playlists or sessions faster while providing a more personal feel in the output and requiring minimal input from the user. To expedite this, we want to utilize the users' preferences much faster than something like r-MUSIC. There are ample preferences in people's personally constructed playlists that could benefit our system. This could give us a more personalized output compared to other systems, and people can also utilize the practical purpose of collaborating. For example, if one wants to have a party, the users can choose a playlist that fits into this context, therefore creating a collective party playlist. While the system does not explicitly try to derive contexts from the music, the users can choose a playlist that fits the context of the session, i.e., apply the practical purpose of collaborative music sharing. Most music streaming platform users often have personal playlists, and the Spotify user base is very vast[24]. First and foremost, we want our system to work with Spotify to have good user coverage. All of the above entails that the music in our system will be more personalized, and users can choose a playlist that fits into the context of the session. Coupled with the fact that the only input our system needs is a personal playlist, it should prove to be easy and fast, alleviating the inertia problem. Furthermore, as in many of the studies we have looked at, users like a degree of control. One study[28] found that the user experience increased after allowing them to exert their preferences on top of the Core algorithm manually. To this end, we implemented an easy-to-use feature, where a user can Wish for a song, and from that song, the system can, over time, move towards this song specifically, giving our pathing algorithm a direction. This should help users guide the experience and tailor it more to their wants.

To make our system a reality, we need data on the songs users want to play to make a path through the space of users' collective playlists. We chose Cyanite[9] as the source of music data, as it utilizes Spotify's song previews to analyze the songs and is therefore continually updated with new data and provides us with extensive data from the queried songs. Combining all participant playlists into a single multidimensional space represents the Core of the OurTunes algorithm. From this, the algorithm finds a path through this space that aims to play songs with similar features while also ensuring that something is played from every participant's playlist.

3.1 Music Data

Cyanite contains many different types of metadata. We use 6 of the features from this metadata that Cyanite provides. Below are the 6 chosen features:

- BPM
- Arousal/Valence
- Energy level
- Genre
- Mood

The Cyanite metadata contains a mix of numerical and categorical data. Most of the numerical features have a value between 0 and 1. However, one of the problems with the metadata is that the values are not normalized. For example, BPM is just the numerical value of BPM; hence, it has a much higher value than the other metadata, which are in the range of 0-1. Therefore, we normalize BPM to impact our algorithm equally to the other metadata. As there is no maximum value for BPM, we normalize it based on the current iteration's maximum. For example, if we have 100 songs and the max tempo of the set of songs is 165 BPM, then we normalize based on 165 BPM as the maximum value. Arousal/Valence is normalized similarly. These values are between -1 and 1, and we simply add 1 to both and then normalize based on the max value, which is 2.

Energy level is a categorical value constituted by the values Variable, Low, Medium, and High. These are simply converted into a corresponding numerical value. The values are converted into 0 for Variable, 0.33 for Low, 0.66 for Medium, and 1 for High. This way, it fits into our system with the other numerical values. We could have used one-hot encoding, but we found from previous experimentation that it segregated the data too much into the different categorical values.

Genre and Mood both consist of sub-metadata. We can take Genre as an example. Genre is divided into 15 Genres, each with its own value for a given song. The values are in the range of 0 and 1. A song can, for example, have 0.5 for the Genre pop and 0.7 for the Genre rapHiphop, which suggests that that particular song contains elements of both genres. This provides us with good coverage, as the songs are not segregated too much, as songs can be represented in multiple Genres.

As discussed previously, other papers have worked with visualization of music and playlist creation based on proximity between songs. Based on our testing, Cyanite metadata can also be used for this purpose. We established this using dimensionality reduction (T-SNE) to make the data points viewable in a 2D space, creating two color-coded visualizations of each song's highest Genre and Mood. This can be observed in Figure 2 on the next page.

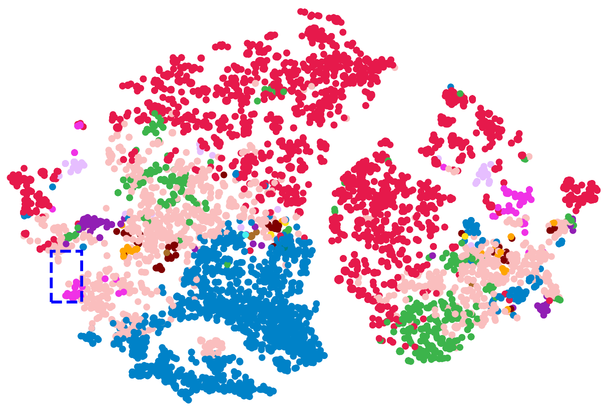
For the Genre song space, we see the most dominant Genres being rapHipHop, pop, and electronicDance with many other genres sprinkled in between. In Figure 2(a), we see pop being placed approximately in the middle of rapHipHop and electronicDance neighborhoods, which makes sense as pop usually is influenced by these two genres. For the Mood song space, seen in Figure 2(b), we primarily see energetic, sexy, happy, and uplifting as the most dominant Moods. The songs visualized are all derived from party playlists, so these labels fit well within that context.

We can see two sets of two songs in the local area in the left part of each song space. The zoom-in of this can be seen in Figure 2(d). The two sets consist of two different songs with almost the same values. This is because they are two versions of the same song, in this case, the normal version and a radio edit. This indicates that songs that are approximately the same have approximately the same feature set. Furthermore, songs from the same artist appear close to each other, as the four songs highlighted are all from the artist Daft Punk. This shows that songs close to each other resemble each other. This, coupled with the fact that users prefer clustering compared to an exploratory path throughout the music space, is the basis of our Core algorithm[26].

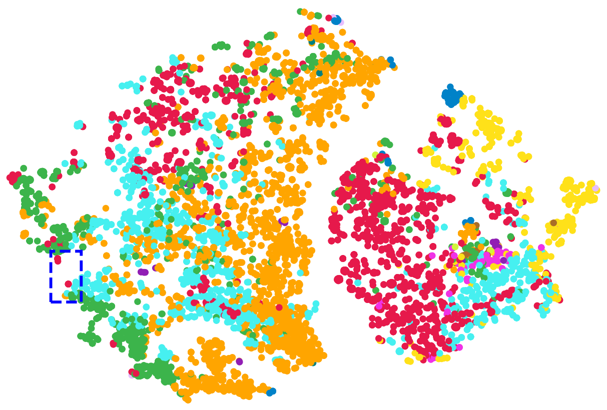
Although the T-SNE reduction shows that similar songs are near one another, we still needed a distance measure that describes how far or close the songs are. This is because we are working with relatively high dimensionality, in which T-SNE only preserved local areas and not the global structure of the space. We chose the Manhattan distance as our chosen distance measurement, as the Manhattan distance performs better in a high-dimensional space compared to something like Euclidean distance[10].

3.2 Core Algorithm

With the OurTunes feature space established, we can now dive deeper into how this space is used to generate a playlist for a group. The first task is to find a starting point/song in the space. Using K-Means to find clusters gives us multiple potential starting points with different vibes. We then find the centroid for each cluster and pick the song closest to the centroid, which will be one of the potential starting points of our playlist. The system randomly picks one of these potential starting points as the first song.



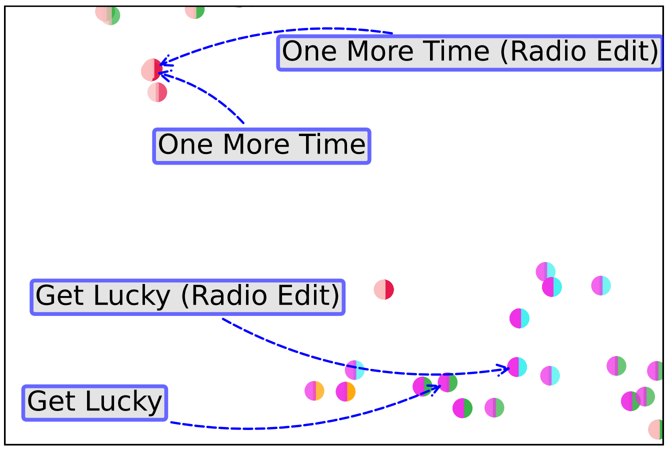
(a) OurTunes song space color coded to primary Genre.



(b) OurTunes song space color coded to primary Mood.

Color	Genre	Mood
🟦	ambient	aggressive
🟫	blues	calm
🟨	classical	chilled
🟦	electronicDance	dark
🟧	folkCountry	energetic
🟪	funkSoul	epic
🟦	jazz	happy
🟪	latin	romantic
🟩	metal	sad
🟥	pop	scary
🟥	rapHipHop	sexy
🟪	reggae	ethereal
🟩	rnb	uplifting
🟥	rock	
🟩	singerSongwriter	

(c) Color coding table. Genre and Mood values are independent.



(d) Zoomed-in inset of OurTunes song space. Left half color is primary Genre and right half color is primary Mood.

Figure 2: Scatterplot from 5000 songs in the OurTunes feature space based on Cyanite data, dimensionally reduced using T-SNE. A zoomed-in inset shows two pairs of songs of nearly identical songs (the radio and non-radio version) by the same artist (Daft Punk) in close proximity to each other.

For each song to be added, the system uses k-nearest neighbors (KNN) to find several close songs, depending on the number of participants, and then randomly selects one of the retrieved candidate songs. OurTunes tracks how much each user has "won" a round, i.e., their song was chosen. We then take the inverse of these values to construct our user weights to increase the chance that users with low win count win subsequent rounds. This weight is then multiplied by the candidate songs' distributed chance to encourage our algorithm to choose songs from users that have not been heard recently. This improves the fairness of the system and non-deterministic playlists. To speed up the querying of the nearest neighbors, we store all the data in a k-d tree[5]. The overall pseudocode for our Core algorithm can be seen in Algorithm 1.

Algorithm 1 OurTunesCoreAlgorithm

```
Require: $audioFeatures$
Require: $startPoint$
Require: $k$                                   ▷ querySize
Require: $numSongs \geq 0$                     ▷ Playlist size
  $playList \leftarrow set(startPoint)$
  $songs \leftarrow []$
  while $len(songs) < numSongs$ do
      $songsWithWeights \leftarrow$
          Query $k$ songs with weights
      $total \leftarrow sumWeights(songsWithWeights)$
      $randVal \leftarrow rand(0, total)$
      $weightAccum \leftarrow 0$
      $chosenSong \leftarrow$
          weighted random choice of $songsWithWeights$
  end while
  $songs.Insert(ChosenSong)$
```

3.3 Wish Songs

Another important feature of OurTunes is the ability for users to impact the path the system takes through the song space by choosing specific Wish songs to move to a specific part of the song space. On a high level, this is done by finding the shortest path through currently chosen Wish songs (a traveling salesman problem). OurTunes then finds the sub-paths between these Wish songs using a modified version of the Core algorithm.

In more detail, the system first determines the order in which the Wish songs should be played. As mentioned, we order Wish songs based on the shortest path, but we also want to ensure that if many users agree on what type of music they want to hear, then these Wish songs are prioritized. We do this by clustering Wish songs to see if we have any similar Wish songs and then prioritize the biggest cluster. We use K-Means to cluster where the number of clusters to find is determined using the Kneedle algorithm[23], an algorithm based on the elbow method that finds the optimal number of clusters. We did not want a minority cluster to be more prioritized, so we set a minimum value for the share of the cluster. If a cluster has less than 33% of the amount users in the session, then these clusters are not prioritized, and we simply find the shortest path through all the Wish songs. If there is a cluster with more than a 33% share, then this cluster will be the first stop in the Wish song path. The Wish songs inside a cluster are ordered based on the shortest path that visits all clustered Wish songs. For the remaining Wish songs outside the cluster, OurTunes finds the shortest path after visiting the clustered Wish songs.

Once the sequence of Wish songs has been determined, the system begins moving towards those Wish songs. Overall, the Wish algorithm works similarly to the Core algorithm. We query songs based on a dynamic query range, then filter out songs that do not decrease the distance to the Wish song we are moving towards, ensuring directionality. On top of this, we include a new weight, which gives a higher weight to songs that decrease the distance to the Wish song we are moving towards. This weight is calculated by taking the normalized ratio of a given song's distance and the maximum distance raised to a rather large exponent. This way, we value and choose songs closer to the current Wish song in the queue more often, allowing OurTunes to get to the Wish song quickly. As mentioned, the query range is dynamic, which means it increases or decreases depending on the number of Wish songs, as a higher share of Wish songs is seen as an indication that users are dissatisfied with the current songs being played. Lastly, Ourtunes always chooses the Wish song if it is inside the queried songs.

When all Wish songs have been visited, the system returns to using the Core algorithm as described in Algorithm 1; that is, it returns to having no directionality and just choosing songs in the current area. The complete pseudocode for the Wish song part of the algorithm can be seen in Algorithm 2. This implementation of the Wish song algorithm currently has some issues that were missed during initial development. One issue was that the number of queried songs scaled faster than intended. It also has weighting issues that make it choose songs further away than intended. Lastly, the Wish songs of the participants were, in some cases, removed before being played. A more in-depth explanation of the above-mentioned issues and their implications will be covered in Section 6.1.

Algorithm 2 Determine sequence of Wish songs

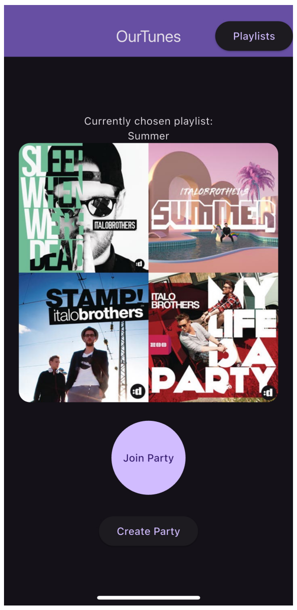
```
Require: $Wishsongs$
Require: $startPoint$
Require: $numSongs$
Require: $threshold \geq 0$,
Require: $SongCounts$
 1: $totalSongPath \leftarrow set(startPoint)$
 2: $cluster \leftarrow$ Select cluster with highest share above threshold
 3: if $cluster \neq$ nil then
 4:     $songs \leftarrow$ Shortest path through cluster Wish songs
 5:     $totalSongPath$.Add($songs$)
 6: end if
 7: if $|totalSongPath| < numSongs$ then
 8:     $songs \leftarrow$ Shortest path through non cluster Wish songs
 9:     $totalSongPath$.Add($songs$)
10: end if
11: if $|totalSongPath| < numSongs$ then
12:     $extras \leftarrow$ Use Core Algorithm to query remaining songs
        missing
13:     $totalSongPath$.Add($extras$)
14: end if
15: return $totalSongPath$
```

3.4 The OurTunes app

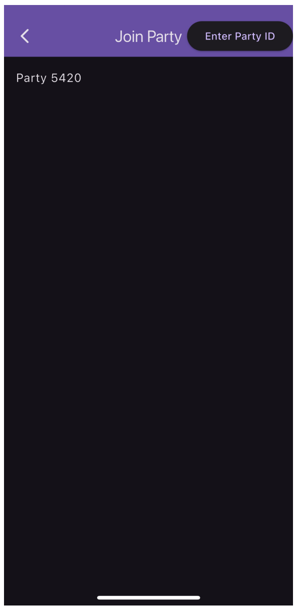
To properly conduct a user test for OurTunes, we need a way for participants to interact with the system. The overall goal of the app is to complement the minimal interaction aspect of the algorithm by making all interactions simple and easy to use.

We developed the app in Flutter for both Android and iOS, wherein users can join a party with their own playlists and choose Wish songs. Bluetooth Low Energy (BLE) was utilized to make each participant's phone advertise the party to nearby devices. The purpose of this is twofold: allowing users to quickly join an OurTunes party nearby without requiring the host to share a link with the participants or show a QR code, along with checking whether any participants have left the session if they cannot be scanned anymore. This enabled us to scan for nearby devices from the host device and remove them from the party if not scanned for a longer period, freeing both the participant and the party host from manually removing inactive people from the party. Playlists could also be added to the app by sharing a playlist from Spotify to our app, using the built-in smartphone share functionality. Screenshots of the app from the participants' perspective can be seen in Figure 3.

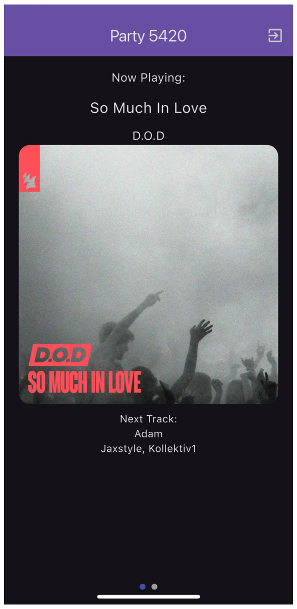
When participating in a party, the app is comprised of two screens. The first screen shows the currently playing song, along with its cover, and also displays the upcoming song. For the host, there is also a play/pause button. The second screen shows a user's currently selected Wish song with a search bar above, allowing users to search Spotify's entire library of songs. Wish songs are not restricted to existing songs in users' playlists. Below the selected Wish song is the Genre Breakdown, which shows the upcoming genres with gradients. These gradients are the main Genres for each of the next 10 potentially upcoming songs. The more solid the gradient, the more that particular Genre was represented in that song. The Genre data are taken directly from the same data used by the algorithm and serve as an approximation of how the upcoming songs sound. We deemed that users would have an easier time differentiating songs based on Genre rather than Mood, as something like a happy song does not convey much information. The relevant genres to show were determined by the sum of each Genre for the 10 songs and filtering out low values. This would allow Genres to appear despite only being relevant in a single song. Withholding the concrete names of the next 10 songs was a deliberate choice, as these songs can change considerably if someone changes their Wish song. We wanted to avoid the experience where users would see an upcoming song they liked, and the next time they would check, it could have been removed because the path through the space got changed. Lastly, the app was designed to run in the background and allowed reconnections to the party in case the app was closed. Rejoining a party would not reset the user's chosen playlist or their selected Wish song.



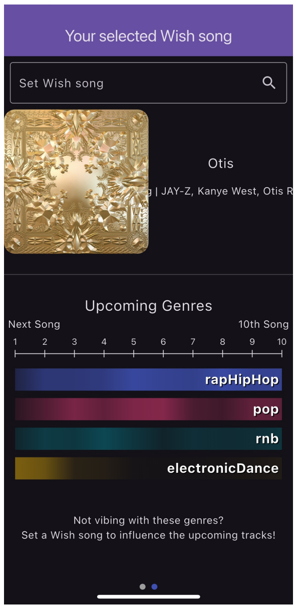
(a) Frontpage of OurTunes



(b) Joining a nearby OurTunes party



(c) Currently playing song view



(d) Selected Wish song and overall genres of the next 10 potentially upcoming songs. Users can search the entire Spotify catalog for a Wish song.

Figure 3: The OurTunes app.

4 Measuring Recommendations

To effectively recommend music to a group, defining what makes a good recommendation in a group context is essential. Here, we draw inspiration from another study in which they use the metrics of Satisfaction and Fairness to evaluate the recommendations of their proposed system[12]. The Cumulative distance traveled will also be measured to get a sense of how much the algorithms jump in total.

Our system defines the Satisfaction metric as the distance between any chosen song by OurTunes and the closest song for each user. The farther the distance, the higher the Satisfaction metric, meaning a user's song is farther away. This means a low Satisfaction value is better than a high Satisfaction value. This builds from our assumption that users prefer songs closely related in audio features and enables us to score each song chosen by our algorithm. It should be noted that Satisfaction only applies to the simulation study of the system. As such, it does not represent how users feel about the played songs. Such evaluation will be done in our user test in Section 6.

The Cumulative distance measure indicates whether the path taken through the space is composed of small or large jumps. A high Cumulative distance means that the jumps taken in the path are generally higher compared to a low Cumulative distance. This also captures a scenario in which the Satisfaction metric fails, wherein the system could perform a long jump through the space but still end up with all users' songs being in close proximity. This metric tells us whether we achieve the attribute of smooth changes in music we aim to achieve.

Lastly, we define Fairness as the distribution of chosen songs. The more evenly songs are chosen between all users, the more fair our algorithm is. Here, we use the Gini coefficient[6]. The Gini coefficient, in general, is based on the concept of Lorenzo curves, which is the curve generated by the cumulative proportion function. We consider this to be an important metric because, as described in the survey study[19], the social purpose mentioned in the study is more pronounced for successful collaborative playlists.

The Gini coefficient is the area between the Lorenzo curve and the cumulative proportion function for the even distribution. It ranges between 0 and 1, where 0 indicates a completely even distribution, while 1 is the opposite. This measurement describes whether the generated playlists evenly represent users. As described here, the Fairness metric only applies to our simulation study. Real users might find a playlist entirely fair, regardless of the Gini coefficient.

5 Experiments

To evaluate the OurTunes algorithm described in Section 3, we have decided to perform a simulation study based on the metrics described in Section 4. We are conducting these experiments to gain insight into how the OurTunes algorithm compares to Round robin, which chooses one song one by one from each user, and Shuffle, which just continuously picks a random song.

5.1 Data Collection for Experiments

To conduct these experiments, we first needed a set of baseline playlists for our simulation study. To do this, we set the following list of criteria for finding the different playlists for our test data:

(1) The overall context of a playlist should be "party". Some examples of genres for such a context could be pop, hip-hop, dance, rock-pop, and rock-indie.
(2) The playlists must contain songs from more than 8 different artists.
(3) Only one playlist per individual user
(4) Playlist has to be made by individuals, i.e., not created by Spotify or other organizations/artists
(5) All playlists must originate from Denmark.
(6) The playlists are discovered by querying Spotify playlists with the search term "fest" (The Danish word for party) and "dakkedak" (A word that describes highly rhythmic music with a high degree of bass[18], i.e., party music).

These criteria have been chosen to ensure that the playlists have a wide variety within and between each other but are not too far apart from the test context and that the playlist authors' nationalities are the same, as that is how this system will most likely be used. The playlists are used to determine whether OurTunes creates a playlist with lower Cumulative distance, better Satisfaction, and better Fairness. In total, we found 35 different playlists using the above-mentioned criteria, which varied from a total length of 58 to 794 songs, which provides a reasonable base.

5.2 Procedure for experiments

Two other algorithms have been tested against the OurTunes algorithm to evaluate our algorithm in the experiments. We need points of comparison to see whether OurTunes performs better or worse than other algorithms. We decided to use the following algorithms for the experiment:

- **Round robin:** Create a playlist of 100 songs by repeatedly going through all individuals and choosing a random song from their playlist.
- **Shuffle:** Choose 100 songs randomly from all users to generate a playlist based on existing playlists.
- **OurTunes:** Creating a playlist of 100 songs using the OurTunes algorithm

We chose 100 songs as the sample size because the average length of a song was 3.18 minutes in our dataset. We deemed that a reasonable playlist for a party is around 5 hours, which is around 100 songs of playtime. Furthermore, we chose to test the algorithms on several test cases. As we have 35 playlists, we elected to run our experiment with 2, 5, 7, 10, 20, and 35 playlists to see whether the algorithms were better at some specific sample sizes. We ran 1000 iterations for each test case to lower the influence of potential outliers. The specific playlists chosen for each iteration are also random, apart from the test case for 35 users, as that is the total dataset size.

5.3 Results of Experiments

Now that we have established the data we used for our simulation test, the results of the respective experiments will be presented. It is important to reiterate that these results are based on simulation studies. Therefore, these results should not be seen as definitive results but instead as preliminary results that indicate whether or not this algorithm has any merit for further testing. Furthermore, these tests are conducted purely using the Core algorithm of OurTunes in Section 3.2 and do not include any simulation of participants picking Wish songs. All results are accompanied by a graph showing the mean of the results, along with error bars visualizing the standard deviation.

5.3.1 *Experiment for the Cumulative Distance Metric.*

For the Cumulative distance experiment, as mentioned in Section 4, we measure the Cumulative distance traveled for the 3 aforementioned algorithms. The experiment results can be seen in Figure 4.

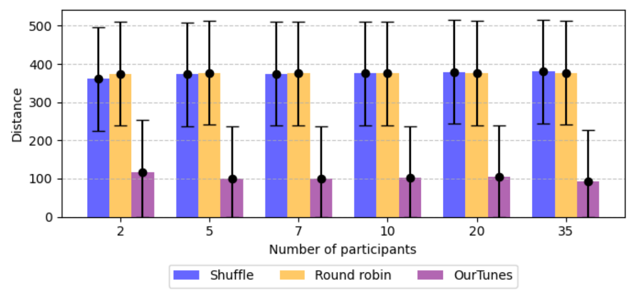


Figure 4: Cumulative distances over 1000 iterations for playlist length of 100. Lower is better.

While all methods have a large spread of Cumulative distances from the mean, we can see that the OurTunes mean is consistently much lower across all group sizes. The upper deviation of OurTunes is almost equal to the lower deviation of both Shuffle and Round robin. This shows that OurTunes only considers the nearest neighbors for any given song. The mean value for OurTunes also slightly decreases with increasing group sizes, as the song space is more

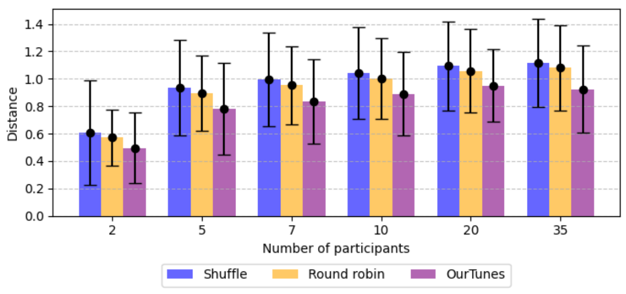


Figure 5: Satisfaction of 1000 iterations per user from current song to user's closest song. Lower is better.

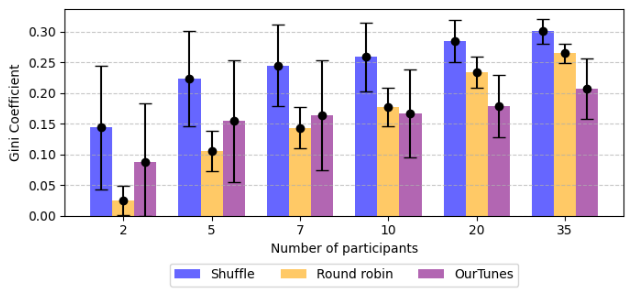


Figure 6: Gini coefficients of 1000 iterations for the distribution of the proportion of songs added by each user. Lower is better.

dense, and therefore, the nearest neighbors will be closer to the query point. Shuffle and Round robin perform almost identically, as they both choose random points in the space without regard to the distances from the current to the next song chosen.

5.3.2 Experiment for the Satisfaction Metric. This experiment aimed to determine the average distance for each user from the chosen song of an algorithm to the users' closest song for 100 songs.

Figure 5 shows that the median Satisfaction of OurTunes is overall lower across all group sizes, albeit only by roughly ~ 0.2. It is difficult to assess how large this difference is, as it is highly dependent on the song space for that simulation. In a dense song space, where most of the songs are very similar, overall distances would naturally be smaller than that of a more sparse song space. One would need to capture the dissimilarity of the included playlists to determine how large the difference is.

Regardless of the impact, it still shows that OurTunes performs better on average in all cases to some degree. The smallest gap is seen for the smaller group sizes, most notably for 2 users. The problem arises when 2 users have too dissimilar playlists. Currently, OurTunes may continue playing only one user's songs as only their songs are inside the KNN search of OurTunes. This points to further optimization when the number of participants is lower. As our project aims toward larger group sizes, we see this as a good result for our initial experiments. For larger party/group sizes, we see better Satisfaction compared to the other algorithms. OurTunes generally stays in more dense areas of the song space for longer periods, leading to songs being closer to each other.

5.3.3 Experiment for the Fairness Metric. For the Fairness experiment, as mentioned in Section 4, we will look at the Gini coefficient, which measures the inequality of the distribution of songs added in proportion to the number of users counted so far. The results can be seen in Figure 6, where it was tested on the aforementioned 3 algorithms.

Overall, the OurTunes Fairness mean is much lower than Shuffle in all cases but falls behind Round robin for smaller group sizes. Furthermore, the deviation is much larger around the mean in all cases for OurTunes. While one might believe that Round robin represents the theoretically fairest system, it does not account for songs existing in multiple playlists. This effect is minimal on smaller group sizes, as even a single song being only on two playlists would represent a majority of the group. The theoretically most fair system would need to keep track of how many times a song has been played from each user, which is precisely a part of what OurTunes does.

Despite this, OurTunes still has a much larger deviation around the mean in all cases. We believe there is room for improvement, but the spread is also a natural consequence of not accommodating a minority preference. Certain combinations of playlists in our simulated sessions resulted in some playlists having very different feature sets compared to the majority, causing very few songs to be played from this minority. This can greatly affect the Gini coefficient, causing the large deviation seen in our results across the board.

This issue also showcases how our Satisfaction and Fairness metrics are connected. While we could make OurTunes much more "fair" by simply finding the nearest track for each user and picking among them, this would affect the Satisfaction metric as songs from a minority could be much further away, decreasing the Satisfaction for each user. We went with a middle-ground approach that attempts to strike a balance between improved Satisfaction and Fairness for larger group sizes. It remains to be discovered in the user test which metrics users value more.

6 User Test

We wanted to test the OurTunes system in a real party setting to follow up on our simulation experiments. To do this, we set up a user test of 7 participants. The participants are all friends of the authors, ranging from 24-27 years of age, and consisted of 5 males and 2 females. Before the test, participants were asked to create a playlist for a party containing music they would like to hear. There were no requirements for the length of the playlist, but we recommended between 10 and 200 songs. The total song space consisted of 587 songs, including duplicates, with the shortest playlist consisting of 35 songs and the longest playlist consisting of 180 songs.

On the day of the test, participants were only instructed on how to download the app and add their playlist. The authors hosted the OurTunes session using a specialized "host mode" of the app, wherein the host did not influence the system in any way. As such, all music played was only picked from the participants' playlists and Wish songs. The test was held on a three-day trip to a vacation home on the first day of the trip. The test started after dinner, although technical difficulties led us to restart the session. The full session where we tested OurTunes started approximately at 20:45 local time. The test lasted approximately 3 hours and 16 minutes, with 54 songs played. It ended with a semi-structured group interview that covered the functionality of OurTunes and their experience using it. The interview was recorded, transcribed, and coded to find overarching opinions describing usage and feelings regarding OurTunes. We also logged all the songs played and the Wish songs and playlists added so we could do a data-oriented analysis of the session.

6.1 Data Analysis

To get an overview of how the user test session went, we examined it from a data-oriented perspective: we looked at the paths taken, the distribution of participants, and the change in Mood, Genre, and BPM over the course of the party. To get a handle on the smoothness of the transitions, we will also look at the distance jumped for each transition in the user test.

6.1.1 Transitions and Composition of the User Test Session. Firstly, to get insights into the overall composition of the session and the transitions made between the songs in the session, we elected to look at Genre, Mood, and BPM features from Cyanite.

To understand the overall composition of the Genre and Mood features present in the song space, we calculated the share of Genres and Moods in the session. We used the highest Genre and Mood features and the second highest above a threshold of 0.2. This was done to account for Cyanite's data, which usually have high values in multiple features, and to ensure that if a song was exclusive to a certain Mood or Genre, the secondary values would not be counted. The distribution of these can be seen in Table 1.

Genre		**Mood**	
Genre	Share (%)	Mood	Share (%)
electronicDance	49.13	energetic	63.62
pop	34.31	uplifting	36.21
rock	15.86	happy	33.79
rapHipHop	15.00	aggressive	18.28
rnb	9.13	sexy	14.83

Table 1: Genre and Mood composition for song space (including all Wish songs and not-played songs)

We also visualized how the OurTunes algorithm changed between different Genres over time by plotting the primary and secondary Genres for the songs played in the session in Figure 7. In general, we see that the songs played consist mainly of the Genres electronicDance(∼ 50.0%), pop(∼ 31.48%), rapHipHop(∼ 29.62%) and rock(∼ 12.96%). Here, ∼ 44% of the songs had no secondary Genre when accounting for the threshold. This indicates that a high degree of the songs played were associated with only one specific Genre.

Overall, the Genre composition for songs played correlates well with the overall composition of the song space seen in Table 1. The Genre rapHipHop slightly deviates from the overall composition by being overrepresented in songs played compared to the total song space. To investigate this, we queried the 60 closest songs of each song played in the session and looked at the Genre distribution to get a general idea of the areas around the path OurTunes took in the session. Here, the Genres electronicDance (∼ 60.04%), pop(∼ 28.81%), rapHipHop(∼ 20.59%), rnb(∼ 10.13%) and Rock(∼ 8.94%) were the Genres most represented. This shows that rapHipHop and rnb songs were more represented in the surrounding space of the path than in the total song space.

For the overall Mood of the session, we, similarly to the Genre analysis, looked at the primary and secondary Moods of the different songs, which can also be seen in Figure 7. We can see that the represented Moods of the songs in the session were mainly energetic(∼ 90.74%), uplifting(∼ 53.70%), aggressive(∼ 22.22%) and happy(∼ 18.51%). The Moods energetic and uplifting were overrepresented compared to the song space composition in Table 1. We found the main reason for this to be that the Wish songs requested (played and not-played) had mostly the energetic(∼ 66.66%) and uplifting(∼ 39.39%) Mood. Wish songs had a much higher chance of being chosen than normal songs, and OurTunes chooses closely related songs, which naturally influences the overall result.

	Genre		**Mood**	
	1st	2nd	1st	2nd
1	rock	metal	energetic	aggressive
2	electronicDance	rapHipHop	energetic	aggressive
3	pop	rock	energetic	uplifting
4	electronicDance	pop	energetic	happy
5	pop	rapHipHop	energetic	uplifting
6	electronicDance	rapHipHop	energetic	uplifting
7	electronicDance	pop	energetic	uplifting

Table 2: Most and second most represented genres and moods for each participant playlists.

To get an overview of how the Genres and Moods of the songs played matched up with the participants' playlists, we found the most and second most represented Genres and Moods for the songs for each participant's playlist. These can be seen in Table 2. The table shows that all participants, except participant 1, had electronicDance or pop as their primary Genre, while, for the Mood, all participants had energetic as their primary Mood. This indicates a relatively homogeneous Genre and Mood composition, with only one participant deviating from the norm. The Mood and Genre of the participants' playlists generally align well with the Moods and Genres of the songs played. We see this from the fact that the

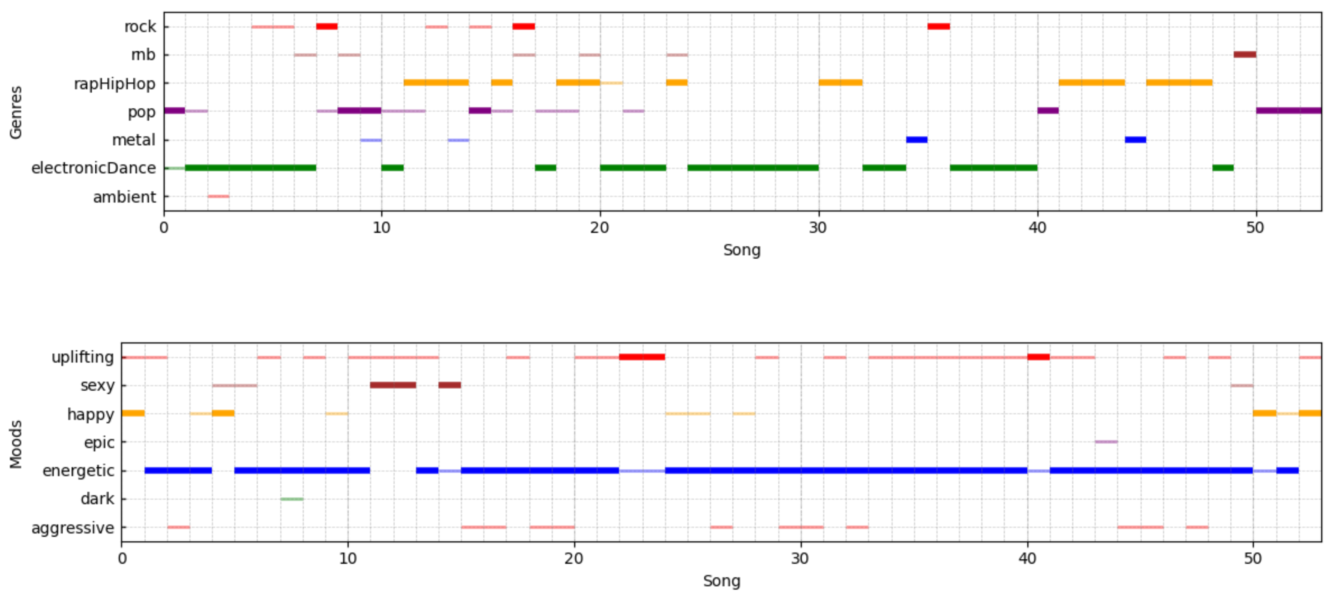


Figure 7: Genre (top) and Mood (bottom) for each played song. Thick lines indicate primary feature and thin lines indicate secondary feature if present(over a value of 0.2).

most played Genres were electronicDance, pop, and rapHipHop, which are more represented in Table 2. The same follows for Mood as energetic and uplifting were the most prevalent moods in the played songs and the aforementioned table.

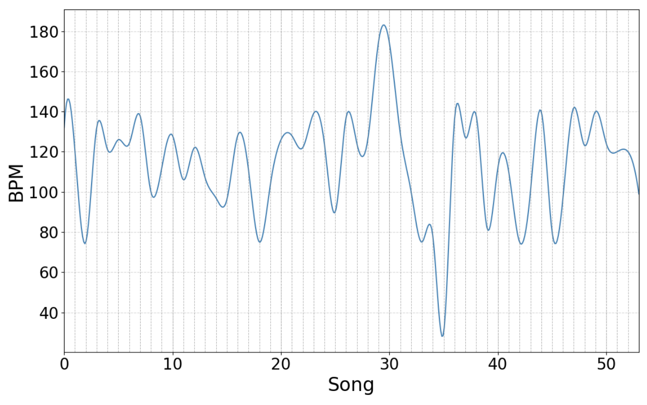


Figure 8: BPM for all songs played at the party

To get an idea of the played songs' tempo, we will look at the BPM range, which can be seen in Figure 8. Here, we found that the BPM range was mainly between 80 − 140, with many above 100. This correlates well with the primary Genre(eletronicDance) and Mood(energetic). The only notable outlier was song number 35, which had a BPM, according to Cyanite of ∼ 30. When we looked up the concrete song, we found that the actual BPM was much higher. This indicates that the features retrieved from Cyanite do not always accurately describe a given song, which could lead to less smooth transitions from a user's perspective.

Now, to gain insight into how far the transitions leaped between songs for the OurTunes user test, we compared the individual jump distances between each song for the median run of Shuffle and the OurTunes user test. Here, this is plotted in Figure 9.

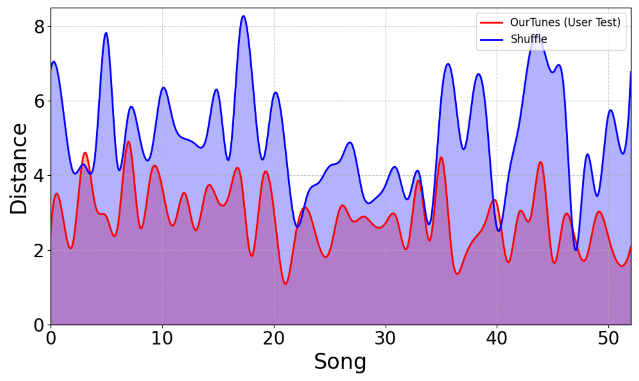


Figure 9: Distance of transition jumps for OurTunes user test session and a median iteration of Shuffle.

We can see that the OurTunes algorithm generally makes shorter jumps between each song, with only a few exceptions. These few exceptions are primarily due to OurTunes moving towards a Wish song in a more sparse region of the space, increasing the jump's distance.

6.1.2 Distribution of participants. Now that we have covered the overall composition of the space, we will look at how the songs were distributed between the participants. The concrete song counts for each participant, including Wish songs and songs on their playlist, can be viewed in Table 3.

	Track Statistics	
Participant	**Song Count**	**Playlist Size**
Participant 1	5	69
Participant 2	9	66
Participant 3	8	99
Participant 4	13	179
Participant 5	10	95
Participant 6	4	35
Participant 7	6	37

Table 3: Distribution of songs played and playlist sizes per participant.

The table shows that the user test session had a fairly even distribution, with a calculated Gini coefficient of ~ 0.2078. However, we found that some participants had a clear difference in their win count. For participants with a low win count, the primary reasons were that they had either a low share of songs in the song space or mostly had songs in their playlist that did not align with the primary or secondary Genre of the total song space, as was the case for participant 1. The only participant with a higher win count was mainly because they had many songs in the song space, and many of their songs were the most heard Genre, i.e., electronicDance.

During the session, participants requested a total of 27 unique Wish songs (33 with duplicate/overlapping requests). A total of 17 Wish songs were played, equal to roughly half.

Picking songs that overlap between users can help improve Fairness, as it is a fairer choice than picking single-user songs, so it should be an obvious choice to pick these when possible. The playlists in the user test session had minimal overlap in songs, with only 14 songs present in multiple playlists, which is 2.4% of the total song space. Of these 14 overlapping songs, only one was played during the session, which is 1.9% of the songs played. We can only surmise that either the participants with these overlapping songs already had a high win count or the overlapping songs were not near the path that OurTunes took.

6.1.3 Exploration of Song Space. Using different dimensionality reduction techniques, we tried to explore the concrete path taken in the user test session. The implementation issues mentioned in Section 3.3 came to light during this exploration. Here, this resulted in a more disorganized path being taken for the user test session.

Firstly, the current implementation can, in some instances, after a Wish song has been added, still have that Wish song as part of the potential candidates for choosing the next song. This leads to all weights being 0, due to our current implementation of the weight calculation described in Section 3.3. Consequently, because of an unfortunate fallback value, the song farthest away is chosen.

Secondly, there are cases where, after a new Wish song is set, some of the previous Wish songs that have not been heard are removed. This behavior can be seen in Figure 10.

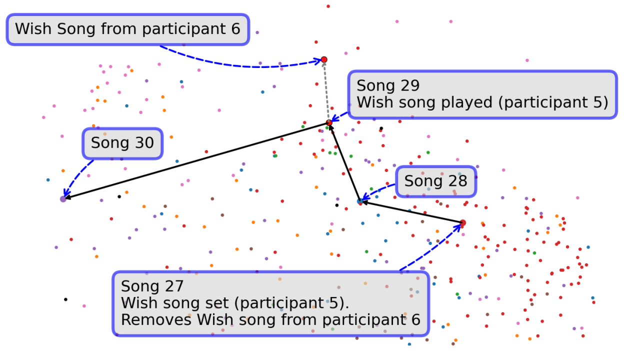


Figure 10: Path showcasing a Wish song sat at song 27, removing the Wish song from Participant 6. The Grey dotted arrow shows one possible path a correct version could have taken.

Despite the potential severity of these flaws, it only had a minor effect on the result. We fixed the issues and ran multiple simulations with the same playlists and Wish songs in the same order to approximate the potential impact of these issues. Our findings show that most participants chose Wish songs that were already close to the song currently playing. This resulted in the total number of Wish songs played being, at most, 3 songs more than was the case for the user test session. The Cumulative distance was also lower for the fixed version, indicating that the system performed smaller jumps. There were no meaningful differences in the Fairness and Satisfaction metrics.

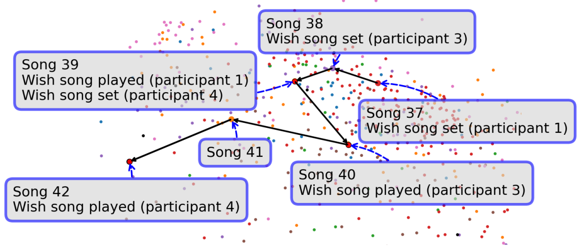


Figure 11: Wish song path close to the end of the party, where Participant 1,3 and 4 got their Wish songs heard. We see that both participant 1 and 3 picked a Wish song close to the currently playing song.

In most cases, the system operated as expected. We will now look at one case and see what it can tell us about how that sub-path progressed. A section of the sub-path taken during the session can be seen in Figure 11. This part of the path was close to the end of the session, wherein Participants 1, 3, and 4, respectively, put on Wish songs, which were all heard.

Here, we can draw a couple of interesting insights. Firstly, in the case of participants 1 and 3, their Wish song was played as the next possible song, which is an indication that they picked Wish songs close to the song that is currently playing. We saw this theme of users selecting Wish songs that were close to the already playing songs throughout the session. Afterward, Participant 4 put on their Wish song while song 39 played. Here, the algorithm takes a clear leap toward the Wished song for each transition.

6.2 Interview

The interview lasted 1 hour and 10 minutes. The general consensus on the system was positive, with minor grievances about the Genre Breakdown, the Wish song feature, and the song choices. In the subsections below, we will delve further into these grievances and into the parts where the system excels and/or works as intended.

6.2.1 General experience. As mentioned above, the system was generally perceived positively. This came to light throughout the interview when we talked about the song choices and the app's different features. All participants felt like they got their songs heard from time to time, which they all felt was a great feeling. This can be seen from quotes such as:

- *"That was one of my songs, which was something a little different(genre wise). Maybe a little rock-ish. I think that was pretty nice"*
- *"I think, that I experienced, that some of my songs from the playlist(participant's playlist), got played here and there... sprinkled in between... I think that was actually kind of nice to see"*

It was not only that OurTunes played their songs, but also the fact that they felt their songs were not pushed away or overlooked, as seen here: *"... So I actually do not feel like the choices and the songs I had on my playlist got pushed to the side or overlooked"*. People, therefore, seemed happy in the sense that they got their songs heard and that they were not overlooked.

Unprompted, the participants also began to compare our system with Spotify Jam. In particular, people echoed that it felt better than using the much-used Spotify Jam session feature on Spotify specifically, as participants' songs were often way down in the queue or could get skipped by other actors at a party. This can be seen in the following quotes:

- *"I was surprised by how quickly my songs got added when I added them to the Wish list because normally, when I use Spotify Jam session, a lot of other people have already added thousands of songs before me to the queue. Then you just get de-prioritized"*
- *"...But I actually do not feel that the choices and songs I had on my playlist got pushed away or overlooked. I actually think, on the contrary, that they got seen(the participant's songs) compared to if it was a Jam on Spotify we created"*
- *"Typically if you are at a party and people are just putting songs in the queue constantly(Spotify Jam) then your song never gets played, because somebody has put the same song in queue 5 times"*

One participant followed up on these statements by saying: *"When participant x put something they knew everyone liked, compared to Spotify Jam, then the algorithm knew that it was something everyone liked, then it(OurTunes) prioritized it. So it came on, and everyone liked it. I would categorize that as a success"*.

Overall, participants generally had a very positive view of the system because of how they got represented and not overlooked compared to other popular systems, such as the aforementioned Spotify Jam session. As one of the participants remarked *"I would say, it is a lot better than the alternative"*, it seems that the system at least outperformed other types of popular ways to create a party listening session.

6.2.2 Bias of the system. This was a contentious point for a lot of the participants. A lot of the participants felt that we were mainly in the same Genres all the time, as can be seen in quotes such as *"...because in the start we heard a lot of rock, which was maybe a smaller mutual genre between what people had in their playlists. After that, it changed between electronic, hip hop rap, electronic, hip hop rap, electronic, electronic, electronic. Where I was maybe one of them that was electronic and rap(people with that kind of music)"*. Generally, it seemed like people perceived that a lot of the music played was electronicDance but also understood that it was what many of the participants added to the collective session, as seen in Section 6.1, which was why that Genre was more pronounced. Despite this they still felt like they got their songs heard, they just wished for more diversity from time to time as seen in the following quotes:

- *"I would maybe have wanted that those who had genres that differed from the rest of us used the Wish functionality more, that they felt that their voice also mattered because then I think we would have gotten more variation than we did tonight"*
- *"I think that could be cool also just to try to... to make the music even more varied"*

However, a participant did share the opinion that the song choices had variety, but the variety mostly came inside the main Genres of the session illustrated in the following quote: *"...where there was this variation even though it was inside the same musical genre it was still different artists, which still made it a new experience"*.

6.2.3 Choice of music. As mentioned, our system aims to pick songs that are close to the song that was just heard. Through that, we expect songs to be similar from the previously heard song to the next. The participants did mostly find this to be the case with a few outliers as seen through the following quotes:

- *"I did not think we had any abrupt changes. I think we had some changes, but it has been fine... But I think that it has been very fair"*
- *"Yeah, well, I as well did not notice that there has been somewhere where we went from whatever kind of genre that was the most noisiest to a genre that was the least noisiest... So I*

did not feel at any point that I have been caught off guard by the changes that happened between songs or genres"
- *"Yeah, I think that sometimes the algorithm could not... could not have done it better because some of the songs we had were really high in tempo. There was a lot of tempo in them. How do you change from that? You cannot really succeed in that"*

However, a participant said that OurTunes could possibly have done a better job if it had considered the setting of the party: *"The algorithm did not really consider different factors such as the noise level of the media player. That could perhaps indicate that people did not want to hear songs with high tempos. Or maybe the speech noise level of the party, like the noise level of the speech in the room where the media player is"*. This could be a possible point of improvement for a later iteration of OurTunes, but it presents a lot of new problems in making such a system.

Another point that some of the participants felt would be a great addition to the system was to consider the song's language. Participants said that they wanted to hear more Danish music and tried to put on Wish songs that were Danish, but currently, OurTunes does not consider a song's language. This was shown through answers such as *"The algorithm did not at all take into account the language. I would have liked that. Like, for example, if everyone had something Danish on their playlist then it knew (OurTunes) it was a common denominator for what we all wanted to hear at some point at least... I think everyone can nod in agreement to that. I see some nodding heads out there"*. This was not a major contention for most of the participants. This was mostly echoed by 3 participants, who might seemingly have wanted Danish music to be more represented as its own category by our system.

One participant felt that it was anxiety-inducing to have some of their songs played, as they were weird in style and placed at a wrong point of the party(right in the beginning). These songs were not weird in the sense of their Cyanite features, but instead in the sense of lyrics, which our system does not account for, and hence, our system did not take this into account when choosing these songs. The confusion and feelings from the participants seemed to stem from their perceived understanding of the system as seen in this quote: *"The point is that I thought that it(OurTunes) would take everyone's playlists and then construct a new playlist based on all the participants' genres. That was how I got it explained by another participant"*. We have never discussed how the system works with any of the participants, so we can only assume here that some participants had an idea of how it worked and continued to share this perceived notion of our system with other participants. Another possibility for this unforeseen feeling from the participant might also stem from how the system starts. Currently, as explained in Section 3.2, our system performs clustering and then chooses the closest song to the centroid. The cluster chosen by our system is done randomly, and a possible optimization here could be to give the users more power over where to start the music-listening experience.

6.2.4 Opinions about the features. For this section, we mainly focus on the two features that participants can interact and use to make their choices, i.e., the Wish song feature and the Genre Breakdown. These features were generally positively received, but they also had some downsides we can delve into.

As for the Wish song feature, most participants ended up using it. They used the feature much more than we expected them. All participants did, however, say that it granted them a degree of control over the system, as seen in the following quotes:

- *"As I said earlier, I felt that many of the songs I put on got represented after like two songs, so I kind of felt that the playlist was varied, but it was still me and x(references another participant) that kind of controlled the playlist"*
- *"I requested a song, and it came in the span of 4 songs"*
- *"I definitely think that I had a lot of control over what music got played as long as it was in the main genres that got played often tonight"*
- *"Okay, when there was a specific song, then I Wished for it. And it came. I would not say in the span of 2 songs. I would say 5-6-7 songs. That was completely fine"*

However, this is not the whole picture. Some participants felt that it did not grant them enough control as their songs never got heard, which is seen in the following quotes:

- *"I tried putting some different music on compared to what was played. It(Genre Breakdown) said a lot of eletronicDance, and then I thought, I want to hear some jazz. The jazz never got played, but that was probably because I was the only one that put it on, so I changed my Wish from jazz to something eletronicDance, and then my song came on 7 minutes later"*
- *"I tried putting on "For Pengene" as my Wish song, but it never came, so I ended up changing it to some hip hop and then it got played right after"*

The participants did not get all their Wishes fulfilled, especially if they did not coincide with the general consensus, but this might be a preferred behavior from the participants. If the song does not fit the current consensus of the rest of the participants, should that overrule what the rest of them want, or should it urge the participant to change their Wish to something that aligns more with the rest of the participants? This is an open-ended question that does not have a straightforward answer.

The participants voiced some possible points of optimizations that might help participants in making choices based on the current context:

- *"It could be nice if there was a way to say, hey now there is some sort of new direction, is there anyone that wants to jump on that? I think that could be cool"*
- *"I do not know if this is overcomplicating things, but the Wish feature is actually pretty good, but I do not know if it might be a bit too primitive. I do not know if you could make it so you have 3 Wish songs, and you somehow could weigh those songs differently?"*

One participant suggested maybe having some indicator of how likely a Wish song was to be played: *"but if you had some sort of indicator, what was most likely to be played, based on our playlists"*. There are many possible variations, and we will delve deeper into these in Section 8 later in the paper.

As for the Genre Breakdown, some participants felt it was somewhat hard to decipher:

- *"I thought that it was hard to read"*
- *"The diagram for the genres was borderline unreadable in the start"*

One of the reasons was a visual bug that happened to one participant we found out about after the session, but the other participants did not like the diagram. One of them echoed that *"I do not know if those fade colors should be combined into one line or that they should not fade at all"*. It seems that one participant probably thinks of Genres in binary. Either it is a Genre, or it is not that Genre. This could be alleviated in possible future iterations, which we will discuss in Section 8. Despite this, most of them got the idea behind it after a while, and most of them used the diagram to some extent:

- *"It had a direct effect on what I chose to recommend"*
- *"I noticed that at one point where there were 100% pop electronic and therefore I tried putting some rock on in the form of "My Chemical Romance""*
- *"I used it to get an overview over what kind of genres that got put together based on the choices we all made, but also to see when those choices would be represented"*

All in all, the features saw great use, but both had some caveats that need fixing, especially the Genre Breakdown. The Wish song feature functions as intended based on what we implemented, but it may have been too stringent based on the participants' feedback. Either that or provide new functionality that can help users make better decisions to exert control over the system. The Genre Breakdown's learning curve was too steep and hard to grasp quickly. It took some time before the participant understood it, which is definitely something to improve. This could be done by simplifying it even further, as it seemed too detailed to understand its meaning with little effort.

7 Discussion

Our project was largely a success based on our research, as test participants stated that it felt better compared to usual collaborative music sessions like Spotify Jam. We aimed to find a middle ground between a personally curated playlist and easy set-up in a group setting using our proprietary algorithm, which uses Cyanite's metadata, while aiming for as little manual input from users as possible yet still giving them a degree of control. We elected not to pick all the Cyanite features as some of the features were segregating the data too much and did not provide anything meaningful for our algorithm. An example of this is the Voice feature. This feature consists of 3 predictive labels between 0-1 that predict if the audio was female, male, or mostly instrumental. Here, we found that this feature segregated the space into these 3 distinctive labels. This led to a larger segregation in the song space than we liked, so we removed it. Another example is the Character feature. This feature consisted of 16 different predictive labels between 0-1 concerning the character of a given song. Although this feature did not segregate the song space too much, it appeared to affect the song space similar to the Mood feature described in Section 3 and was therefore not used.

We also found that the features needed normalization, so they did not impact our algorithm too much, and each feature had a balanced impact. After choosing appropriate features and normalizing them, we saw that songs close to each other also resembled each other. However, our chosen data did not catch all the minutiae of the songs, as seen in Section 6.2. The participants wanted a way to suggest or get more songs in a specific language, such as Danish. This is currently not a variable in our system; however, we do not think this will be beneficial as it will segregate our data too much because of how OurTunes works, similar to something like the Voice feature mentioned above.

As mentioned in Section 3.2, our algorithm picks a random cluster as its starting point. This was shown to be potentially problematic, as some test participants did not like the random starting point, which sometimes does not fit the context of the start of the session. This is a hard point to capture, as we cannot know the context of a session beforehand. If we were to implement a way to circumvent this problem, it would come at the cost of more user interaction and, therefore, increase the inertia, i.e., the problem of starting a collaborative session. There could be ways to alleviate this starting point problem in future iterations, which we will discuss in Section 8.

Furthermore, our playlist algorithm did seem to encounter some problems when it came to querying songs in the multidimensional song space. We did not consider the number of songs users have in their playlists. This did not prove problematic in our simulation test but did come to light in our user test. The system is decently good at choosing songs from different people. However, test participants with larger playlists had a higher frequency in how often their songs were chosen. The more songs a user has, the bigger the chance they have to be more represented in query range selection compared to someone with fewer songs in their playlists. As we do not currently have some way of counteracting this interaction, it leads to users with larger playlists being more presented, especially if people have more varying tastes, as in our user test. This relates to the aforementioned problem of OurTunes not being fair/social enough. This is one of the optimization points that we will outline in more detail in Section 8.

We based our system on choosing closely related songs and optimized the system around that; however, the relationship between the Satisfaction and Fairness metrics does not seem straightforward. As described in one study[26], users preferred music in clusters; however, this evidently relates only to single-user systems. We saw in our interview that participants would have liked it to represent users more equally, i.e., they wanted a more social experience. The Satisfaction metric is apparently not as important regarding collective playlists or sessions, as the social purpose outweighs this metric. Therefore, Fairness correlates better with a good experience, both in the sense that participants expressed positive feelings when their songs were chosen and wanted to hear more of other participants' songs. This is, of course, a give-and-take relationship as the participants agreed that minority preferences should not be able to influence the system too much to something where nobody has a similar song, i.e., low Satisfaction.

For our data analysis in Section 6.1, most of the main takeaways aligned well with what the participants stated in the interview in Section 6.2. From a purely data-oriented perspective, all participants had songs from both their primary Genres and Moods heard, which aligned well with the participants' experience, where they all felt that they got some of their songs heard. The data aligned well with the participants' overall experience: most transitions between songs were not too abrupt, and the Genres coincided nicely with participant perception of the heard Genres.

Another point revealed by the data is that some participants were less represented in the user test than others. This was, as mentioned previously, either due to them having a smaller number of songs in common with the majority or having a smaller playlist than other participants. However, this is not necessarily a problem, as one can easily imagine a scenario in which playing these songs would lead to none of the other participants having a similar song, thus leading to a less satisfying experience. This ties into the above paragraph, where the participants say they want to hear a higher share of different participants' songs but that there should still be a limit on how much they diverge from the other participants.

As for our interview in Section 6.2, one participant suggested deriving more context of the party through either the noise level of conversation or the noise level of the played music; this presents a couple of challenges. First, when it comes to the noise level of the conversations, we could implement a way to record the noise through the host's microphone on their device and then analyze this to get a better sense of the current context of the session. We feel that this is an overstep over what the app should do, as an app that records a whole session seems like an invasion of privacy, and we, therefore, are not aiming to implement such a feature. This also overlaps with OurTunes' use of personal playlists instead of using profiles from the users, as users have more control over what they are bringing to the party. This also allows users to align their music preferences with the social context. Secondly, as for the noise level of the played music, there are possible avenues to pursue here. However, the problem lies in that we could look at the sound level of the host device, but some devices have their own inputs to change the sound level. From our understanding, this is very dependent on the specific Bluetooth speaker. Therefore, a possible feature that leverages the sound level of the party can only be used with devices with these functionalities. We are therefore electing not to pursue this avenue, first and foremost, as we deem other optimization points more important.

Regarding existing systems, our algorithm, as it currently stands, contributes to the research space of playlist-making in several areas. First and foremost, while individual playlist-making and general music discovery have seen some work[26][3], collaborative playlist-making systems have seen relatively sparse research. However, there are some examples. One such system is SRMAC[28], which explores the creation of collaborative playlists through the user-chosen context and their already existing preferences regarding genres. This system does not investigate the aspect of creating a smooth transition between song types, ensuring Fairness, and was not compared against other methods, such as Round robin and Shuffle. As described in that paper, the users preferred to have a bigger degree of control than what the initial test system had.

We succeeded in ensuring smooth transitions, as outlined in the user test interview and the data analysis in Section 6 compared to other systems. However, we saw the same problem as SRMAC with insufficient control. Although we gave users a degree of control, the interview participants also expressed wanting even more control. This is a difficult balance to strike, as the system should not take too much time out of the hands of the users. As the participants expressed, it was also nice that they could leave it alone and use it when they wanted to, but they wanted more control and information to make better and more informed decisions. The participants suggested having the possibility of having more than one Wish song. We do not deem this to be in line with one of the intended goals of OurTunes, as we wanted to make a system that lowers interactions with the system. It was also suggested that more information be provided for users to utilize. This way, we could possibly alleviate this problem with the feeling of not having enough control, as each choice they make could be more meaningful, and therefore provide more control through one action instead of just giving them more actions to exert control with. Depending on how it is implemented, this could also align well with the social purpose. Possible solutions to increasing meaningful information will be discussed in Section 8.

Furthermore, some work has been done on more explicit consensus on collaborative playlist making, such as r-MUSIC[29]. This system introduces a voting system where users can vote on songs on a shared public playlist. Although this creates a very explicit consensus-based way of making collaborative playlists, our system focuses more on doing it implicitly with smooth transitions, for which our findings in the simulation studies show preliminary success. Furthermore, we also succeeded in requiring less input while still providing them with meaningful control, though, as outlined, they did not feel this was enough.

7.1 Limitations

First and foremost, there were some bugs in the implementation, which are outlined in Section 3.3. As we mentioned in Section 6.1, these bugs, while unfortunate, did not seem to have a major effect on either the Fairness or Satisfaction metric. Only the distance between songs and total number of Wish songs played were affected. Therefore, most of the main findings would likely have been the same without the bugs.

We also see that the song space matters quite a bit for the OurTunes algorithm's performance with respect to both the simulation metrics and distribution of users at a party. One such example is that of Participant 1 in the user test session. In this case, this participant got fewer of their songs heard because their playlist had a different

Genre and Mood composition than the rest of the participants. In general, this means that the main findings can very well change depending on how divergent the different participants' songs are.

Furthermore, the song space is only as good as the feature variables from Cyanite. Sometimes, as shown in Section 6.1, these values are not entirely accurate. We saw this with a song from the user test that had a much lower BPM than intended. This is further limited by Cyanite only analyzing the 30-second preview of each song from Spotify, which might be a section of the song that sounds very different from the rest.

The user test was also limited in the number of participants. Here, the main findings of this section could very well change if different participants were tested, more specifically how well the participants know one-another, along with their individual song preferences either being more overlapping or more diverging. Lastly, all of the participants were acquaintances with the authors, which might have led to some bias in the feedback given.

8 Future Works

OurTunes represents a small step towards a modern, easy, and ad hoc collaborative music listening experience with minimal interaction needed. Nevertheless, there is still much to be discovered that OurTunes does not cover. A natural first step would be to fix the implementation issues as highlighted earlier, and while the incorrect weights and reset of users' Wish songs are clear cases, the query range is more nuanced. Although it did not function according to our specifications, participants seemingly preferred the more varied songs rather than being stuck in the same space for too long. The exact ranges are to be determined, but something that represents a somewhat diverse selection yet is still in close proximity to a given song seems optimal.

The social aspect of OurTunes is also something that should be explored further. Based on our user test, users did want to listen to other participants' music, which overall suggests that Fairness is a more important metric than distances between songs. In particular, the participants wished to see other people's Wish Song, which we also believe would be a good improvement, as we would like the participants to coordinate their song selection. This would also help more extreme outliers have the possibility of being heard if there is an overall consensus.

– Participants also requested more options to make their Wish song choices more informed, either through more control with multiple Wish songs or more information about how far out their Wish songs were. One possibility is allowing users to set multiple Wish songs that are far apart to increase the odds of visiting outliers and enhance the social experience further. Another option is to show how likely their Wish song is to be played or how far away each song is in the search result for Wish songs. Both are potential options, but we lean towards the latter as it still ensures low interactability, so users do not spend too much time tinkering with the system. Furthermore, based on our user test, participants ended up using Wish Songs a lot more than anticipated, and allowing users to have multiple Wishes could transform the system back to a normal queue.

To further increase the Fairness of the system, we also want to add the ability to select which song to start from. The system already picks out multiple potential starting points but only picks one at random. A possible solution could be to give users the ability to pick from these starting points with a simple consensus vote while still keeping the ability to have the system pick one if so desired. However, this will be at the cost of inertia and more interactability, and we will have to see if users deem this worthwhile.

The Genre Breakdown of the upcoming songs was more confusing for participants than we had hoped. During the design phase, we tested different visualization methods, such as only showing a single Genre and a pie chart, but we settled on the gradients of upcoming Genres. Users eventually learned how it worked, but many preferred something more easily readable at a glance. While we do not have any concrete ideas for how this could look, we believe an ideal solution still involves withholding the specific songs from users, as this adds more excitement when one's own song appears and hides the potential disappointment of seeing one's own song far down the queue.

Lastly, our user test only included participants who knew each other. While we believe such a scenario would be the most common use case for OurTunes, it could also be used if none of the participants knew each other. Therefore, conducting a user test with nonacquaintances under the same conditions could be beneficial in illuminating even more minutiae in the space of collaborative playlist making. We hypothesize that participants would react differently and/or have different requirements than sessions consisting of acquaintances. Most notably, parties consisting of nonacquaintances might not care as much about listening to other participants' music, as demonstrated by one of our test participants describing how someone using Spotify Jam would queue five of their own songs.

9 Conclusion

The OurTunes algorithm introduces a novel way to create collaborative playlists implicitly with a degree of manual control that emphasizes smooth transitions and fairness with respect to the distribution of songs per user. Our preliminary findings in the simulation study show improvements compared to other algorithms with improvements in our Satisfaction and Cumulative distance metrics. Fairness results were mixed and showed improvements with larger group sizes. As for our interview, we saw that the participants preferred OurTunes over other systems like Spotify Jam; however, they wanted more control and better representation of the other participants. Overall, we made strides towards better collaborative playlist making by lowering the inertia and increasing the social purpose compared to other low inertia systems like SRMAC, but with points that can be improved to increase the social purpose further.

10 Acknowledgements

We would like to thank Henning Pohl for his valuable feedback and guidance during this project, as well as all the participants in our pilot and user test for their feedback. We would also like to thank Mali Michelsen for designing the logo and icon used for the mobile application.

References

[1] Alo Allik, Florian Thalmann, and Mark Sandler. 2018. MusicLynx: Exploring Music Through Artist Similarity Graphs. In *Companion Proceedings of the The Web Conference 2018* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 167–170. https://doi.org/10.1145/3184558.3186970

[2] Ivana Andjelkovic, Denis Parra, and John O'Donovan. 2019. Moodplay: Interactive music recommendation based on Artists' mood similarity. *International Journal of Human-Computer Studies* 121 (2019), 142–159. https://doi.org/10.1016/j.ijhcs.2018.04.004 Advances in Computer-Human Interaction for Recommender Systems.

[3] Maureen S. Y. Aw, Chung Sion Lim, and Andy W. H. Khong. 2013. SmartDJ: An interactive music player for music discovery by similarity comparison. In *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*. 1–5. https://doi.org/10.1109/APSIPA.2013.6694280

[4] Jared S. Bauer, Aubury L. Jellenek, and Julie A. Kientz. 2018. Reflektor: An Exploration of Collaborative Music Playlist Creation for Social Context. In *Proceedings of the 2018 ACM Conference on Supporting Groupwork*. ACM, New York, NY, USA, 27–38.

[5] Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM* 18, 9 (Sept. 1975), 509–517. https://doi.org/10.1145/361002.361007

[6] Michael T. Catalano, Tanya L. Leise, and Thomas J. Pfaff. 2009. Measuring Resource Inequality: The Gini Coefficient. *Numeracy* 2, 2 (2009). https://doi.org/10.5038/1936-4660.2.2.4

[7] Shih-Han Chen, Sok-Ian Sou, and Hsun-Ping Hsieh. 2024. Top-N music recommendation framework for precision and novelty under diversity group size and similarity. *Journal of intelligent information systems* 62, 1 (2024), 1–26.

[8] Andrew Crossen, Jay Budzik, and Kristian J. Hammond. 2002. Flytrap: intelligent group music recommendation. In *Proceedings of the 7th International Conference on Intelligent User Interfaces* (San Francisco, California, USA) *(IUI '02)*. Association for Computing Machinery, New York, NY, USA, 184–185. https://doi.org/10.1145/502716.502748

[9] Cyanite. 2024. The Power of Automatic Music Tagging with AI. https://cyanite.ai/2023/10/03/the-power-of-automatic-tagging-with-ai/

[10] D. 2001. On the Surprising Behavior of Distance Metrics in High Dimensional Spaces. In *Proceedings of the 8th International Conference on Database Theory (ICDT '01)*. Springer-Verlag, Berlin, Heidelberg, 420–434.

[11] Pedro Dias and João Magalhães. 2013. Music recommendations for groups of users. In *Proceedings of the 2013 ACM International Workshop on Immersive Media Experiences* (Barcelona, Spain) *(ImmersiveMe '13)*. Association for Computing Machinery, New York, NY, USA, 21–24. https://doi.org/10.1145/2512142.2512151

[12] Kevin Eustice, Amir Mohsen Jourabchi, Jason Stoops, and Peter Reiher. 2008. Improving User Satisfaction in a Ubiquitous Computing Application. In *2008 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*. 496–501. https://doi.org/10.1109/WiMob.2008.106

[13] Kevin Eustice, V. Ramakrishna, Nam Nguyen, and Peter Reiher. 2008. The Smart Party: A Personalized Location-Aware Multimedia Experience. In *2008 5th IEEE Consumer Communications and Networking Conference*. 873–877. https://doi.org/10.1109/ccnc08.2007.204

[14] Jenny M. Groarke and Michael J. Hogan. 2019. Listening to self-chosen music regulates induced negative affect for both younger and older adults. *PLOS ONE* 14, 6 (06 2019), 1–19. https://doi.org/10.1371/journal.pone.0218017

[15] Sang-Won Lee. 2021. Lost in Co-curation: Uncomfortable Interactions and the Role of Communication in Collaborative Music Playlists. *Proceedings of the ACM on Human-Computer Interaction* 5 (04 2021), 1–24. https://doi.org/10.1145/3449137

[16] Zsolt Mezei and Carsten Eickhoff. 2017. Evaluating Music Recommender Systems for Groups. arXiv:1707.09790 [cs.AI] https://arxiv.org/abs/1707.09790

[17] Kenton O'Hara, Matthew Lipson, Marcel Jansen, Axel Unger, Huw Jeffries, and Peter Macer. 2004. Jukola: democratic music choice in a public space. In *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques* (Cambridge, MA, USA) *(DIS '04)*. Association for Computing Machinery, New York, NY, USA, 145–154. https://doi.org/10.1145/1013115.1013136

[18] Den Danske Ordbog. 2025. Dictonary defintion of DakkeDak. https://ordnet.dk/ddo/ordbog?query=dakke-dak

[19] So Yeon Park and Blair Kaneshiro. 2021. Social Music Curation That Works: Insights from Successful Collaborative Playlists. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW1, Article 117 (April 2021), 27 pages. https://doi.org/10.1145/3449191

[20] So Yeon Park, Audrey Laplante, Jin Ha Lee, and Blair Kaneshiro. 2019. Tunes Together: Perception and Experience of Collaborative Playlists. In *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019*, Arthur Flexer, Geoffroy Peeters, Julián Urbano, and Anja Volk (Eds.). 723–730. http://archives.ismir.net/ismir2019/paper/000088.pdf

[21] Hamidreza Rabiee, Javad Haddadnia, Hossein Mousavi, Moin Nabi, Vittorio Murino, and Nicu Sebe. 2016. Emotion-Based Crowd Representation for Abnormality Detection. (2016).

[22] David A. Robb, Stefano Padilla, Britta Kalkreuter, and Mike J. Chantler. 2015. Moodsource: Enabling Perceptual and Emotional Feedback from Crowds. In *Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing* (Vancouver, BC, Canada) *(CSCW'15 Companion)*. Association for Computing Machinery, New York, NY, USA, 21–24. https://doi.org/10.1145/2685553.2702676

[23] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. 2011. Finding a Kneedle in a Haystack: Detecting Knee Points in System Behavior. 166 – 171. https://doi.org/10.1109/ICDCSW.2011.20

[24] Shubham Sing. 2024. Spotify Users Statistics 2025 - Subscribers Demographic & More. https://www.demandsage.com/spotify-stats/

[25] Spotify. 2024. Jam. https://support.spotify.com/us/article/jam/

[26] Beatrix Vad, Daniel Boland, John Williamson, Roderick Murray-Smith, and Peter Berg Steffensen. 2015. Design and Evaluation of a Probabilistic Music Projection Interface. In *Proceedings of the 16th International Society for Music Information Retrieval Conference*. 134–140.

[27] Laurens van der Maaten and Geoffrey Hinton. 2008. Viualizing data using t-SNE. *Journal of Machine Learning Research* 9 (11 2008), 2579–2605.

[28] João G. Bracaioli Vitório and Carlos N. Silla. 2023. Music Recommendation System for Shared Environments. In *2023 30th International Conference on Systems, Signals and Image Processing (IWSSIP)*. 1–5. https://doi.org/10.1109/IWSSIP58668.2023.10180270

[29] U. Wolz, M. Massimi, and E. Tarn. 2004. r-MUSIC, a collaborative music DJ for ad hoc networks. In *Proceedings of the Fourth International Conference onWeb Delivering of Music, 2004. EDELMUSIC 2004*. 144–150. https://doi.org/10.1109/WDM.2004.1358111