A Mathematical Approach to DJ Transitions

Wyatt Mowery, Kevin Zhang, Michael Zhang

Massachusetts Institute of Technology (MIT)

wmowery@mit.edu, kevbev@mit.edu, mmz@mit.edu

May 13, 2025

1 Abstract

This project proposes a more efficient and objective way for DJ's to find the next song during a live set. DJ's work with playlists possibly consisting of hundreds of songs and must be able to find and transition to another song within 60 seconds. This short timeframe given to parse through a long list of songs leads to many un-optimal choices, leading to transitions between songs that may not flow as well as other options would have. Our project provides a solution for DJ's by vectorizing a playlist of songs. By finding vector representations of the different qualities of a song that allow for seamless transitions, we allow DJ's to input the current song they are playing and our algorithm will return the top k songs that are most similar to the current song. This ensures an objective and mathematically-based way to identify the best song for the smoothest transition. To test our hypothesis that our algorithm will provide a better song, we conducted experiments with DJ's from MIT's campus. We compared their choice of song with our algorithm's and proved that our algorithm returned better songs, which we measured with our song norm. Future work would be attempting to have our algorithm also find the timestamps Ultimately, our research provides an alternative way for DJ's to select songs. This would make live sets overall more fluid and entertaining with better music and better transitions.

2 Methodology

2.1 Objective

The goal of this paper is to develop a mathematical method for DJ's to identify transitions from one song to another. This section will detail the methodology we employed to identify transitions between songs.

2.2 Song Feature Extraction and Problem Formation

All audio files (.mp3 or .wav) were stored in a "songs" directory, in which we used a Python package Librosa, to extract features from the song. We extracted the 1-dimensional tempo vector that recorded beats per minute (BPM), the 26-dimensional timbre vector that recorded the mood of the song, the 12-dimensional key vector which recorded the key in a which a song was played in, the 7-dimensional spectral contrast vector which measured the energy of the song, as well as the 2-dimensional loudness vector that measured the mean and standard deviation of energy.

In total, for each song in the "songs" directory, we converted the song into a 48-dimensional vector that we added to an A matrix, that will serve as a bank of available songs to transition to. We made a b vector as well, which is the vectorized song in which we want to transition from. From here we use regression and gradient descent strategies to find x, the optimal song

that is the best transition to any song in A, from the song b.

2.2.1 Vector Representation Details

We would like to give a more in depth explanation as to what each of the features in our vectorized song represent. Beats per minute is a relatively straightforward statistic that measures the speed of a song. The timbre vector is gained by extracting the Mel-Frequency Cepstral Coefficients, which measure extremely short term power-spectrum's of a song (20 millisecond windows). We then take the log of these values to more accurately represent how humans hear sound, and then apply a discrete cosine transformation over different frequencies. These different frequencies mean that the first few MFCC's capture broader shapes in the timbre while the higher order MFCC's represent very fine grained spectral details. When we take the mean over the entire song, it allows us to collapse all these short frame measurements into a single summary vector for the entire length of the song. The key vector is calculated by extracting the chroma features of a song, which represent the intensity of each of the 12 different pitches across the song. This allows us to find songs that will be able to harmonize well. The energy vector is measured by calculating the differences in peak vs valley across 7 different frequencies, and this allows us to find songs that have similar sharpness and contrast so that we can keep the energy level consistent. This is helped by our final feature, which is the loudness vector, which is measured by taking the root mean square of the audio signal over the duration of the song, which further helps us find songs with similar energies as we transition. These 5 features allow us to capture all the things that are important to creating high quality transitions.

Figure 1 depicts a radar chart of our 48 dimensional vector representation for 3 different songs. We started with Party in the USA by Miley Cyrus (Blue) and ran our algorithm to find the best transition candidate. The result we received was Butterfly Effect by Travis Scott (red). In order to show how our algorithm finds songs that are similar across the key attributes we described above, we randomly selected

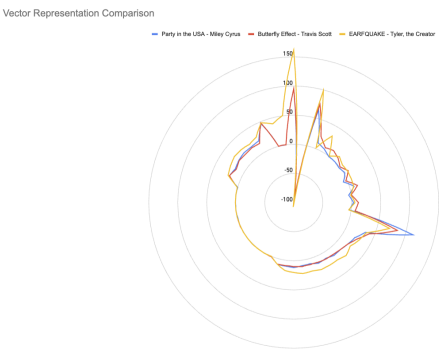

Figure 1: Song Radar Chart Visualization

a third song, Earfquake by Tyler, the Creator (Yellow). You can see that the blue and red lines, representing the songs we determined to be good matches, follow each other quite closely as you rotate around the chart, while the yellow line, representing a random choice, deviates quite severely from the other two, particularly in the tempo and key sections of the chart, the two features we will end up valuing above features like energy.

2.3 Constraint Design

Though we vectorized each song into a 48-dimensional vector, it must be noted that particular dimensions in each "song vector" play a more important role in an optimal transition than others. Namely, tempo and key play the biggest role in determining a good DJ transition, thus we place constraints on these two values. We define that for a transition to be valid, the selected song must be within $\pm$ 10% of the original song. Additionally, the selected song must be within $\pm$ 1 semitone of the original song. Should these constraints be violated, the song will be excluded from consideration as a candidate for the optimal x, or transition.

2.4 Optimization via Constrained Gradient Descent

To identify the best transition vector x, we solve the following optimization problem:

$$\min_x \|Ax - b\|_W^2 + \lambda \|x\|_W^2 \quad \text{s.t.} \quad x_i \geq 0, \quad \sum_i x_i = 1$$

- $A \in \mathbb{R}^{48 \times n}$: feature matrix of candidate songs
- $b \in \mathbb{R}^{48 \times 1}$: feature vector of the input song
- $x \in \mathbb{R}^{n \times 1}$: weight vector representing how much each song should contribute to the transition
- $\|\cdot\|_W$: A custom **song norm** that accounts for the relative importance of different features during a transition (see section 3.5 for more details)
- λ: regularization parameter, set to 0.01

Note that in the optimization problem, we use the constraint $x_i \geq 0$ since the vector x should tell us how much of each song we should have in the optimal transition, and we cannot have a negative amount of a song.

2.4.1 Solving via Gradient Descent

The gradient of the objective function is as follows:

$$L(x) = \|Ax - b\|_W^2 + \lambda \|x\|_W^2$$

$$L(x) = \|W \cdot (Ax - b)\|_2^2 + \lambda \|W \cdot x\|_2^2$$

$$\nabla_x L = 2A^\top (W^2 \cdot (Ax - b)) + 2\lambda x$$

At each iteration of gradient descent, the gradient is computed and used to update x in the negative direction scaled by a factor of η, the learning rate. Afterwards, x is projected back to the probability simplex, or normalized such that $\sum x_i = 1$.

2.4.2 Importance of the Probability Simplex Projection

We used probability simplex in order to ensure that our results were usable. The probability simplex "standardizes" our song vector x to ensure that our probability distribution is wholly positive and that all our probabilities sum to 1. This probability simplex is defined below:

$$\Delta^{n-1} = \left\{ x \in \mathbb{R}^n \,\middle|\, x_i \geq 0 \text{ for all } i,\ \sum_{i=1}^{n} x_i = 1 \right\}.$$

This ensures that our x vector is a convex combination of songs. From a DJ's perspective, this ensures our code finds the top 3 songs of highest similarity, and not an arbitrary signal reconstruction.

In order to enforce this constraint during each iteration of gradient descent, we projected each x_i onto the simplex using a Euclidean projection algorithm. The projection procedure sorts the entries of x_i at each step, computes a threshold such that subtracting θ from each component of x results in a vector that is non-negative and sums to one, and then clips negative values to zero. These steps ensure that the vector x is a valid probability distribution of songs at each step. Formally, the project is implemented as follows:

$$\theta = \frac{1}{\rho + 1} \left(\sum_{j=1}^{\rho} u_j - 1 \right), \quad x_i' = \max(x_i - \theta, 0),$$

Here, u is the sorted vector x in descending order and rho is the largest index satisfying $u_j - \theta > 0$.

Before adding in a probability simplex to our algorithm, the optimal x vector we derived for a given song contained negative aspects of other songs, and the probability distributions never summed to one due to the lack of standardization. Even when we added in code to only take in positive values, the songs our algorithm returned could be shown to have little to no difference with randomly selecting songs. With this fix, however, we were able to ensure that the x vector returned only positive song contributions that summed to one, and thus have valid similar songs returned.

2.4.3 Importance of Regularization

We also chose to employ regularization due to its importance in producing unique and reasonable solutions. Without regularization, the problem reduces to solving for:

$$\min_x \|Ax - b\|_W^2 \quad \text{s.t.} \quad x_i > 0, \quad \sum_i x_i = 1$$

However, if the number of songs n is less than 48, the solution space to $Ax = b$ contains infinitely many minimizers leading to the issues of non-uniqueness. This may lead to convergence towards extreme solutions and general overfitting which may exploit noise or small features in b that are not meaningfully represented across the song bank matrix, A.

By adding the regularizer $\lambda \|x\|_2^2$ to the loss function, a unique solution is guaranteed even when $n < 48$. Additionally, this regularizer term would help prevent overfitting and would cause the optimizer to spread the weight across a range of songs instead of on just one or two songs, unless those songs clearly are the best match of course.

2.5 Defining the Song Norm

As certain features are more important in deciding good DJ transitions, we introduce a weighted norm, which will be referred to as the **song norm**. This norm increases the influence that certain features, namely the tempo and the key of a song, have when calculating the distance between songs. This is needed as it ensures that the loss function properly penalizes mismatches in these vital dimensions more so than in others.

Suppose the difference between a predicted song x and the optimal song b is defined as $z = Ax - b \in \mathbb{R}^{48}$. Let $W \in \mathbb{R}^{48}$ be a vector of positive weights representing how much each feature of a song should be weighted. The song norm is defined as following:

$$\|z\|_W := \sqrt{\sum_{i=1}^{48} W_i^2 z_i^2} = \|W \cdot z\|_2$$

$W \in \mathbb{R}^{48}$ is initialized as a column vector of all ones, which assigns equal importance to all features initially. We then set $W_0 = 6$ where index 0 corresponds to the tempo of a song, the highest weighted feature. Additionally, we then changed indices W_2 through W_5 to be 2, as the lower order MFCC's represent the broader spectral envelope of the song and we would like to weight those higher than the finer grained information delivered by the higher order MFCC's. Lastly, we set W_{14} through W_{25} to be 4, as these indices correspond to the key of a song, as the key is the second most important factor in providing a smooth transition from song to song. The remainder of the features remain unweighted, meaning they have weight 1.

2.5.1 Proving the Song Norm is a Valid Norm

We will first prove positive definiteness meaning $\|z\|_W \geq 0$ and $\|z\|_W = 0 \iff z = 0$. Since each $W_i^2 > 0$ and $z_i^2 \geq 0$, there sum will always be non-negative, and thus $\sum_{i=1}^{48} W_i^2 z_i^2$ will always be non-negative and $\|z\|_W = 0$ only when $z = 0$.

We will next prove homogeneity meaning $\|\alpha z\|_w = |\alpha| \cdot \|z\|_w$. This is proven by observing:

$$\|\alpha z\|_W = \sqrt{\sum W_i^2 (\alpha z_i)^2} = \sqrt{\alpha^2 \sum W_i^2 z_i^2} = |\alpha| \cdot \|z\|_W$$

Lastly, we must prove that the triangle inequality holds true, meaning that for all $z_1, z_2 \in \mathbb{R}^{48}$,

$$\|z_1 + z_2\|_w \leq \|z_1\|_w + \|z_2\|_w$$

This is proven true by observing:

$$\|z_1 + z_2\|_W = \|W \cdot (z_1 + z_2)\|_2 \leq \|W \cdot z_1\|_2 + \|W \cdot z_2\|_2 = \|z_1\|_W + \|z_2\|_W$$

Therefore, since the song norm $\|z\|_W$ satisfies all positive definiteness, homogeneity, and the triangle inequality, it is a usable and valid norm.

2.6 Interpreting the Transition Vector

The output vector $x \in \mathbb{R}^{nx1}$where $\sum x_i = 1$ represents a similarity distribution over the n available songs in A (not including the current song b). Each entry $x_i \in [0, 1]$ represents how much the i^{th} song of A contributes to the optimal transition from song b.

To select the best candidate for the transition, we compute the *argmax* of x which returns the index i of the song that contributes the most towards the transition from song b. The corresponding feature vector is simply the i^{th} column of the matrix A in which we can use to trace back to the name of the song that serves as the optimal transition from song b. In practice, we may choose to take the top k highest values in x and provide a ranked list of transition candidates to allow for flexibility and creativity from DJ's.

3 Experiments and Results

3.1 Evaluating our Algorithm

In order to test our alrogithm's efficacy, we randomly chose 5 songs and asked 5 MIT DJ's to choose the next best song given our entire playlist. To make the trial as close to a live set as possible, we gave the 5 MIT DJ's 3 minutes to review the list of all the songs in the playlist. Then, we gave them the name of one of the 5 songs and gave them 30 seconds to choose their next best song. We soon realized that the MIT DJ's generally did not have a good sense of what song to choose next. Many of them claimed to simply choose more or less "randomly" during sets and only a few paid minor attention to song details like beats per minute. As a result, our team chose to continue this project by simply randomly selecting 5 different songs from the playlist to simulate a DJ's choices.

We then compared the difference in song norms between our algorithm's top choice of song against the 5 randomly chosen songs. The results are shown below:

From the chart and the graph, it is very clear that our algorithm (blue) selects much better song matches compared to a human (red) who more or less randomly selects songs. Furthermore, our algorithm consistently has a 85-95

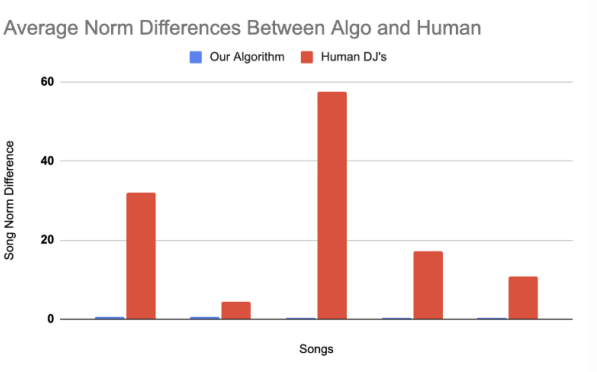

Figure 2: Difference in Song Norm between Human and Algorithm

To test statistical significance, we used a one-sided Welch's t-test. We compared the 25 random songs to the 5 songs returned by our algorithm. Our resulting t-score was 3.98, which correlates with a p-value of p = 0.00027. Because our p-value is much less than 0.01, we concluded with 99% confidence that there is statistical significance between the similarities with the songs that our algorithm returns and random human song choices. These values are shown in table 1 below.

3.2 Additional Visualization

Additionally, we wanted a way in which we could visualize the distances of the vectors from each other, thus, we ran Principle Component Analysis to reduce the dimensionality from 48 dimensions to 3 dimensions. Subsequently, we ran a 3 dimensional visualization to visualize the distance from the song we are transitioning from b, to the optimal song we found using our method, as pictured below in figure 3 and figure 4.

Song	Mean Random Dist.	Best Dist.	Improvement vs. Random
DNA by Kendrick Lamar	32.04	0.6653	97.9%
One More Time by Daft Punk	4.42	0.5784	86.9%
Fire Burning by Sean Kingston	57.44	0.2866	99.5%
Baby by Justin Bieber	17.21	0.4485	97.4%
Hotline Bling by Drake	10.77	0.3043	97.2%

Table 1: Comparison of Best Match Distance to Mean Random Distance

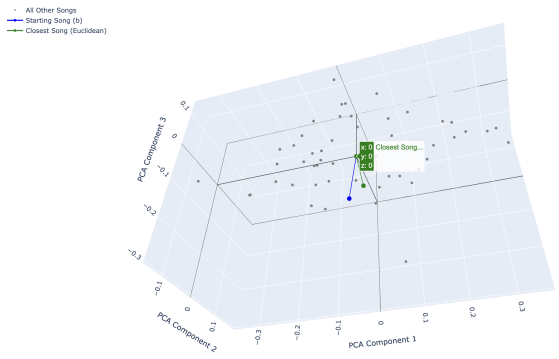

Figure 3: Angle 1

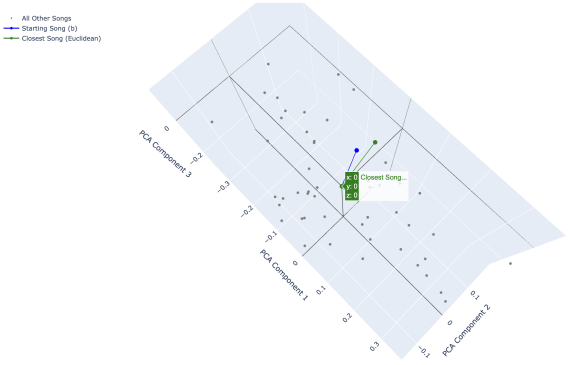

Figure 4: Angle 2

4 Discussion

Our results demonstrate that a mathematical, feature-driven approach to DJ transitions can outperform human intuition and are a vast improvement over a random choice. We would like to provide further context for why we found creating the "song norm" to be necessary to creating an algorithm is able to preform strongly against other selection methods.

4.1 Song Norm vs. Dot Product

In evaluating the quality of song recommendations produced by our algorithm, we compared the returned songs to a target song with a dot product similarity and song norm difference. While our algorithm did demonstrate a statistically significant improvement upon song norms, there was no significant difference in dot product similarity. The following section talks about the difference between the song norm and dot product and why our algorithm only returned statistical significance in one of the two.

Our weighted norm distance measures overall similarity across tempo, timbre, harmony, spectral contrast, and loudness. Our algorithm directly aims to minimize this weighted norm distance, which we have shown it statistically does.

On the other hand, the dot product between two songs measures the raw alignment between the two vectors without accounting for magnitude. While the dot product is often used as a measurement for similarities, un-normalized dot products do not work as well in high-dimensional spaces, like our project (48 dimensions). This is because the dot product will be dominated by large feature values rather than actual meaningful structural overlap. For instance, two songs may be nearly identical in structure but differ

in volume, which results in a small dot product despite high perceptual similarity. On the other hand, two songs that are both fast, but with little other similarities may be reflected with a larger dot product.

Ultimately, this mismatch arises because the dot product fails to normalize for vector length. If we had somehow normalized all of our song vectors beforehand, the dot product would have been a better tool to measure song similarities. Thus, the lack of statistical significance in the dot product does not reflect a failure in our algorithm, but rather shows metric misalignment. Our optimization algorithm seeks to directly minimize norm distance, and not the dot product.

4.2 Future Work

We believe that this algorithm could be the foundation of a successful, full-stack application that makes becoming a DJ easier and more accessible for all, however there are some features that would need to be added.

Currently, each song is represented by a single summary vector derived from the average of time-varying features across the entire track. This global representation ignores local temporal dynamics that often determine ideal transition points. A natural extension of this work would be to retain the full time series matrix of extracted features (e.g., MFCCs, chroma, spectral contrast) and compute local similarities between segments of songs. This would allow us to also suggest timestamps that represent points in the two songs where the transtion between the two would be smoothest, further helping DJ's make quicker decisions during a live set. Futher updates to the algorithm would be the potential to look ahead to what the next transition might be to ensure fluidity over the entirety of the set.

We also would need to provide a more user-friendly interface. There is no front end work on this project yet and if this were to be an application pushed to the public that would have to change. Allowing users to connect to popular streaming services such as Apple Music or Spotify would make this app much more accessible. This would allow the interface to become much simpler, as a DJ could input the song currently playing and receive the suggestions from the playlists they have created.

The last step would be to try and implement reinforcement learning techniques that would learn the style of the user over time. It would be able to learn certain genres that the particular user tends to gravitate towards as well as favorite songs they always play.

5 Conclusion

This paper introduces a novel, mathematically based way to identify transitions between songs. This algorithm encodes songs into a high-dimensional feature space based off of their tempo, timbre, key, and energy. From there we use constrained gradient decent along with a "song norm" that weights the more important features in determining a smooth transition, tempo, key, and broad spectral envelope, to find the songs that are the best candidates for a transition based off of these methods.

Our experimental results show our algorithm has a significant difference over randomly choosing a song and leads to finding songs that are extremely closely aligned across the board. These results give us confidence that the algorithm developed here, with some additional extensions and features, could be the foundation for an extremely useful tool in DJ's lives.

References

[1] "Librosa: Audio and Music Signal Analysis in Python," *Librosa Documentation*, 2025. [Online]. Available: https://librosa.org/doc/latest/index.html.

[2] B.-Y. Chen, W.-H. Hsu, W.-H. Liao, M. A. Martínez-Ramírez, Y. Mitsufuji, and Y.-H. Yang, "Automatic DJ Transitions with Differentiable Audio Effects and Generative Adversarial Networks," *arXiv preprint arXiv:2110.06525*, 2021. [Online]. Available: https://arxiv.org/abs/2110.06525.

[3] T. Kim, M. Choi, E. Sacks, Y.-H. Yang, and J. Nam, "A Computational Analysis of Real-World DJ Mixes Using Mix-To-Track Subsequence Alignment," *arXiv preprint arXiv:2008.10267*, 2020. [Online]. Available: https://arxiv.org/abs/2008.10267.

[4] E. Liebman and P. Stone, "DJ-MC: A Reinforcement-Learning Agent for Music Playlist Recommendation," *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2015. [Online]. Available: https://www.ifaamas.org/Proceedings/aamas2015/aamas/p591.pdf.

[5] C. Deruty, "Intuitive Understanding of MFCCs," *Medium*, 2019. [Online]. Available: https://medium.com/@derutycsl/intuitive-understanding-of-mfccs-836d36a1f779.

[6] D. Abbattista, V. W. Anelli, T. Di Noia, C. Macdonald, and A. V. Petrov, "Enhancing Sequential Music Recommendation with Personalized Popularity Awareness," *arXiv preprint arXiv:2409.04329*, 2024. [Online]. Available: https://arxiv.org/abs/2409.04329.

[7] C.-Y. Chiu, M. Müller, M. E. P. Davies, A. W.-Y. Su, and Y.-H. Yang, "An Analysis Method for Metric-Level Switching in Beat Tracking," *arXiv preprint arXiv:2210.06817*, 2022. [Online]. Available: https://arxiv.org/abs/2210.06817.

[8] "Mix-To-Track Subsequence Alignment Code Repository," *GitHub*, 2025. [Online]. Available: https://github.com/mir-aidj/djmix-analysis.