CS517 Theory of Computation - Final Project
Constraint-Based Playlist Generation via SMT Solving

Alena Makarova
School of Electrical Engineering and Computer Science
Oregon State University
makarova@oregonstate.edu

Pratik Khanal
School of Electrical Engineering and Computer Science
Oregon State University
khanalp@oregonstate.edu

Abstract

We present a constraint-based approach to automatic playlist generation using SMT solvers. Our tool constructs a playlist of fixed duration (e.g., 30 minutes), selecting and ordering tracks from a larger set of candidates to satisfy multiple global constraints. Each song is annotated with metadata, including duration, valence (emotional positivity), energy (intensity), genre, and popularity. The playlist must (1) stay within a specified total duration range, (2) ensure smooth transitions in both valence and energy between adjacent songs (bounded jumps), and (3) meet or exceed a target average popularity score. The complexity arises from not just filtering songs based on individual attributes but from selecting and sequencing them in a way that satisfies global constraints across the entire playlist, including smooth emotional transitions and duration limits—turning it into a combinatorial optimization problem. We model the problem as an SMT optimization task using Z3, combining Boolean and arithmetic constraints. Although our ultimate goal is to minimize emotional roughness, our formulation focuses on finding feasible playlists that satisfy user-defined upper bounds on roughness, duration, popularity, and transitions. We evaluate our approach on real Spotify-derived metadata and demonstrate that it can efficiently generate high-quality playlists for moderate input sizes with clear constraint satisfaction and interpretable outputs.

1 Introduction

Playlist generation is key for music streaming platforms like Spotify, Apple Music, and YouTube Music. Users often seek thematic or genre-based playlists and emotionally coherent experiences—such as a relaxing morning mix, an upbeat workout sequence, or a moody study playlist. While heuristic- or ML-based approaches (e.g., collaborative filtering, clustering, and reinforcement learning) can capture user taste or similarity between tracks, they often lack explicit control over how playlists evolve in time. For instance, maintaining smooth transitions in emotional intensity (valence and energy) or ensuring no artist or genre dominates too much is difficult to guarantee using learned models alone.

Inspired by structured planning problems in AI and music informatics, we propose a constraint-programming-based solution to this task. Our goal is to build a playlist of fixed duration (e.g., 30 minutes) that satisfies multiple user-specified constraints on song metadata, such as:

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

- Total playlist duration within a target range,
- Smooth transitions in valence and energy between adjacent songs,
- A minimum average popularity score.

This formulation turns playlist generation into a sequencing problem with interdependent constraints and is thus NP-hard. We model it as an SMT optimization problem using the Z3 solver.[1]

In this paper, we first define the constrained playlist generation problem and explain why it is NP-hard. We then reduce it to an instance of SMT by encoding selection and ordering with Boolean variables and emotional and popularity constraints with real-valued arithmetic. Next, we describe the implementation of our solver-backed tool that generates playlists from real Spotify data. We evaluate the tool's performance and scalability on these datasets. Finally, we discuss current limitations, tradeoffs, and directions for future work.

TODOs:

- Expand related work, survey existing ML/heuristic methods for playlisting.
- Clarify theoretical framing, maybe cite scheduling, knapsack, or TSP variants to support NP-hardness.
- Visualize examples of valid vs. invalid transitions
- Include more cases who cares about this problem.
- Add bibliography, references

Our goal is not just to generate playlists that work but also to show how using a constraint-based approach makes it easier to control the structure of the playlist. Unlike many machine learning methods, our approach allows users to clearly define what they want—such as limits on energy changes or popularity—and get results that follow those rules in a way that's easy to understand. We use an SMT solver because the problem involves logical conditions (e.g., song positions, no repetitions) and real-valued arithmetic (e.g., smooth transitions in valence and energy, duration bounds), making it hard to solve using basic SAT or greedy methods. SMT solvers are well-suited to handle this combination of Boolean and numeric reasoning, so we can encode and solve the problem declaratively.

2 Problem Definition

This project aims to generate a music playlist of a fixed duration (e.g., 30 minutes), where the selected and ordered tracks satisfy a set of user-specified constraints. Given a set $\mathcal{S}$ of songs, each annotated with metadata including duration, valence (emotional positivity), energy (intensity), popularity, and genre, we aim to construct an ordered sequence P of songs such that:

- The total duration of the playlist is within a specified range (e.g., 1800 ± 5 seconds).
- The difference in valence and energy between each pair of adjacent tracks does not exceed a given threshold.
- The average popularity score across all selected tracks is at least a user-defined minimum.

This problem isn't just about picking songs that individually meet specific requirements. It also involves deciding how to arrange those songs so that the playlist follows specific rules — like smooth changes in energy and mood from one song to the next or ensuring the playlist hits a target duration and popularity. That makes the problem harder because we must consider which songs to include and in what order. These sequencing and combination rules make it a combinatorial optimization problem, similar to well-known NP-hard problems like subset-sum (for matching the total duration) and scheduling or path-finding problems (for controlling song transitions). We use SMT solvers to handle both parts together because we're working with real-valued data and logic-based constraints simultaneously.

TODOs:

[1] `https://github.com/PKR-808/cs517-2025`

- Add a formal mathematical definition of the problem.
- Include examples showing a valid vs. invalid playlist with respect to valence or duration.

3 Reducing the Playlist Problem to SMT

We reduce the constrained playlist generation problem to an SMT instance using Z3. The solver selects and arranges a subset of songs to satisfy all given constraints. The encoding uses a mix of Boolean and real-valued variables.

Let $\mathcal{S}$ be the set of available songs, and let k be the maximum number of songs allowed in the final playlist (i.e., the number of time slots). For each song i and slot j, we define a Boolean variable x_{ij} which is true if and only if song i is placed in slot j.

We also define real-valued helper variables for valence and energy at each slot j, denoted by v_j and e_j, respectively.

The main components of the encoding are:

- **Duration constraint:** Let d_i be the duration (in seconds) of song i. The total duration of selected songs must fall within a specified tolerance window around a target time T:

$$T - \delta \leq \sum_{i,j} d_i \cdot x_{ij} \leq T + \delta$$

- **Valence/Energy smoothness:** For each adjacent pair of slots $(j, j+1)$, the difference in valence and energy must not exceed pre-defined thresholds:

$$|v_{j+1} - v_j| \leq \epsilon_v \quad \text{and} \quad |e_{j+1} - e_j| \leq \epsilon_e$$

- **Popularity constraint:** Let p_i be the popularity of song i. The average popularity across the playlist must meet a given threshold $P_{\min}$:

$$\frac{1}{k} \sum_{i,j} p_i \cdot x_{ij} \geq P_{\min}$$

Objective: While the ideal goal is to minimize emotional "roughness" (i.e., the sum of valence and energy changes between consecutive tracks), our formulation treats this as a decision problem: can we construct a playlist whose roughness does not exceed a given threshold R?

$$\sum_{j=1}^{k-1} |v_{j+1} - v_j| + |e_{j+1} - e_j| \leq R$$

TODOs:

- Add a small example showing a simplified encoding with three songs and two slots.
- Add a diagram or table visualizing x_{ij} variable structure.
- Explain Z3 implementation details
- Justify why this formulation requires SMT
- Add the mathematical proof

4 Scalability and Optimization Techniques

In this section, we explain how we encode instances of the playlist generation problem into SMT, describe the structure and size of the resulting formulas, and present the techniques we use to make the solver scale to realistic input sizes.

4.1 Encoding and Formula Structure

We introduce a real-valued variable `roughness`, representing the total emotional variation in the playlist, defined as the sum of absolute differences in valence and energy between adjacent tracks. Rather than minimizing this value directly, we constrain it to be less than or equal to a threshold R. We can approximate the optimal playlist by iteratively solving the SMT instance with decreasing values of R.

Each instance is encoded as an SMT problem using Z3. The decision variables are:

- Boolean variables x_{ij} indicating whether song i is placed in slot j of the playlist.
- Real-valued variables v_j and e_j for the valence and energy of the track at slot j.
- Auxiliary real-valued variables zv_j and ze_j to represent the absolute valence and energy changes between consecutive slots.

The total number of Boolean variables is $O(n \cdot k)$, where n is the number of input tracks and k is the playlist length. The number of arithmetic constraints is linear in k for valence, energy, and duration, and proportional to the number of genre groups and slots for genre constraints.

4.2 Optimization Techniques and Tricks

To reduce solving time and memory usage, we apply the following optimizations:

- **Track Sampling:** We limit the size of the candidate set $\mathcal{S}$ by sampling a random subset of tracks (typically 40–100 songs) before building the SMT formula. This reduces the number of x_{ij} variables and the total formula size.
- Placeholder for the optimization tricks

4.3 Performance and Practical Limits

Placeholder

Dataset[2]

TODOs for this section:

- Provide the pseudo algorithm of the code
- Revise the tricks subsection according to the code
- Revise Performance subsection - move it to the experimental section?
- Add the reference to the dataset source
- Add a table comparing solve time for different values of n and k.
- Discuss memory footprint of large SMT instances.
- Provide examples of unsatisfiable configurations (how we diagnose them?).

5 Experimental Evaluation

We evaluated the performance of our SMT-based playlist generation tool on a real-world dataset of 200 tracks collected from the Spotify API. Each track was annotated with duration, valence, energy, popularity, and genre metadata. We measured solve times under varying input sizes and constraint settings to assess both scalability and constraint satisfiability.

5.1 Setup

All experiments were run on a laptop with an Intel i7 processor (2.6 GHz) and 16 GB RAM. The SMT solver used was Z3. For each test, we varied the number of available tracks (n), maximum playlist length (k), and constraint tightness (valence/energy delta thresholds).

[2] `https://www.kaggle.com/datasets/amitanshjoshi/spotify-1million-tracks`

5.2 Input and Output

Input format: A JSONL file containing one JSON object per song, derived from a filtered CSV of Spotify metadata. This includes fields such as $track\_id$, $duration\_sec$, $valence$, $energy$, $popularity$, and $genre$. Command-line parameters configure constraints (e.g., duration target, valence delta, popularity threshold, slot count, etc.).

Output format: A structured JSON file containing:

- The ordered list of selected tracks (with slot number, metadata, and start time),
- The total playlist duration,
- The value of the minimized objective (emotional roughness),
- Or a failure message if the constraints are unsatisfiable within the timeout.

5.3 Results

Placeholder for the results

TODOs for this section:

- Revise Results subsection
- Include plots of solve time vs. number of input tracks, playlist length, and valence/energy delta.
- Add histogram of objective values for different settings.
- Include table of settings (n, k, deltas) vs. solve time.
- Add example of a satisfiable vs. unsatisfiable configuration.
- Evaluate variation across multiple random seeds or track samples.

6 Conclusion

This project demonstrated how SMT solvers can be applied to a structured real-world problem—playlist generation—by formulating it as a constrained optimization task over song metadata. Using Boolean and real-valued variables, we encoded constraints over playlist duration, emotional transitions (valence and energy), popularity, and genre into an SMT instance. Our solver-backed tool, implemented using Z3, produces valid and interpretable playlists, offering users more control than traditional heuristic or machine learning methods.

In future work, we plan to support soft constraints and preferences (e.g., preferring some songs but not requiring them), adaptive profiles for individual users, and multi-objective optimization—such as balancing emotional smoothness with popularity or novelty. We also intend to benchmark our approach against ML-based playlist generation systems to evaluate comparative quality and user satisfaction. Ultimately, this project shows that constraint-based methods can complement existing music recommendation technologies by offering transparent, controllable, and rigorous playlist generation.

References

A Appendix