Music Sequencing using the Traveling Salesman Problem

Bachelor thesis

Jorn Smulders

University of Twente
Bachelor Industrial Engineering and Management
University supervisor: Rogier Harmelink
Company supervisor: Koen van der Brink
August 20 2024

Preface

Dear reader,

You are about to read the bachelor thesis *Music Sequencing using the Traveling Salesman Problem*. This research has been conducted on behalf of Awaves BV in Enschede as the final assignment for my bachelor Industrial Engineering and Management at the University of Twente.

Throughout my bachelor's program, I learned that solutions created within the course are needed within every industry. As a DJ and music curator, I created the interest to put this into practice. When I came across Awaves, I was immediately excited. In particular, they were using technology used to improve the quality of music. Despite the unfortunate circumstance that the company had to cease operations, I am grateful to have been given the opportunity to contribute to this concept.

Without a doubt, I would like to thank my first supervisor, Rogier Harmerlink, for guiding me through this project. I have lost track of how many times we have met to enthusiastically discuss the topic. It is incredibly admirable how supportive and understanding you were during the process, even when personal circumstances became more challenging. The feedback that you have given has also gained me valuable new insights to take away with me. I would also like to thank my second supervisor, Eduardo Lalla for being a big supporter of the concept at the poster market and providing me with essential feedback for finishing my thesis.

Finally, I would like to thank my friends and family for believing in me. For their patience in listening to my theories, for their interest and contribution to the practical assessment, which helped me to gain right confidence and to get inspired to start this research.

I hope you find reading my thesis as inspirational as I found writing it!

Jorn Smulders

Enschede, August 2024

Management Summary

Awaves B.V., a student startup at the University of Twente, has developed an AI DJ to improve music quality at student house parties. Despite implementing features such as automatic transitions and live user feedback, the average streaming time remains under 8 minutes, indicating a mismatch between the product and users' needs. The audible errors mentioned by users, contributing to this dissatisfaction, include inconsistent genre, limited genres, inconsistent music, inconsistent energy levels, unpopular and low-quality music, and poor transitions. Awaves assumes that the errors are caused by the steps involved in creating a digital DJ mixtape: downloading music (recommendation), storing annotations and metadata (classification), deciding the playlist direction (sequencing), and adapting mixing techniques (mixing). Here, the unsystematic sequencing process was identified as the most important core problem and improving this process is expected to increase the coherence and flow of the music playlist. Therefore, the main research question of this thesis is formulated as follows:

What is the optimal way to increase music satisfaction by improving the sequential flow through automated playlist sequencing?

To investigate how the sequencing process should be optimised, the whole process of creating a digital mixtape is mapped out, the relationships between the causes and effects are documented, and earlier solutions are summarised. Using this information we can select an optimisation approach, a Traveling Salesman Problem Heuristic. The TSP involves finding the shortest possible route to visit each city once. In the context of this research, TSP is adapted to optimise the sequence of songs in a playlist, where songs are treated as points in a Euclidean space and transitions between songs are used as distances. The created model considers multiple audio features as variables influencing musical flow, such as tempo, energy, danceability, valence and key. However, due to the practical and computational scope of this research, valence and key have yet to be selected for the validation.

With this model, three solution methods are implemented and evaluated. The brute force algorithm, the nearest neighbour heuristic and the evolutionary algorithm (Excel Solver). With brute force, the optimal distance was found, and with the heuristic and evolutionary method, a sub and near-optimal sequence were found. Running this experiment for multiple scenarios showed promising potential in improving music sequencing. Here, all the solution methods were able to decrease the total distance with nearly fifty percent. Nevertheless, validations are still in the preliminary stages, requiring more repetitions, varied playlists, and diverse scenarios with different quantities of songs to assess the method's effectiveness robustly. The solution methods also been practically validated by creating four anonymous mixtapes, which were ranked by volunteers. The data of the assessment showed us that the shortest distance was ranked as the most preferred. Given the subjective nature of this research, several challenges were faced in executing this assessment. For instance, the length of the mixtape, the setting where the mixtapes where listened and the focus levels of the volunteers. Therefore, more validation is recommended based on the feedback.

In general, we conclude that automating the music sequencing process using optimisation models like the Traveling Salesman Problem can significantly improve the distance between audio features and therefore, musical flow and user satisfaction. While the technical validation of the solution methods is promising, further research and validation are necessary to realise the potential of this approach.

Table of Contents

1. Introduction to the problem

The first chapter introduces the reader to the research conducted at Awaves BV. The structure of this chapter is done according to phases one and two of the (MPSM)(Heerkens, 2017). In this method, the action problem is defined as the problem that has to be solved. Then, the core problem is found by creating a problem cluster. Solving this core problem influences the action problem by closing the gap between the norm and reality of the company. By knowing the action and core problem, the research cycle is constructed. Hence, Section 1.1 gives a brief description of the company. Section 1.2 introduces the company's problem and provides the action problem. In Section 1.3, the problem is defined, and a core problem is chosen. Last, in Section 1.4, the research questions of the research cycle are formulated.

1.1 Company

Awaves BV is a student startup founded at the University of Twente. With the consistent need for music at parties, the founders of Awaves have noticed that the quality of the music played could be improved. Being aware that hiring a DJ comes with expensive equipment and musical knowledge, the founders wanted to develop a cheaper solution that aligns with the newest technology. The mission behind the company is to develop an automated DJ to improve the quality of music at student house parties. Its operations are based on building an AI-driven algorithm that can automatically make transitions and implement live user feedback to create a DJ effect. One of the concepts that the company has developed is Awaves Play, a music streaming application. The goal behind this concept is to experience a live digital DJ at student house parties. At the start of this research, the web application Awaves Play has successfully launched, and has been live for over 11 months (since September 2022), and strives to grow users and receive investments to grow their operations.

1.2 Problem introduction

After nearly one year, the average streaming time per Awaves Play user remains below 8 minutes, equivalent to a maximum of three songs. This trend indicates that the features of automatic transitions and live feedback are not significantly enhancing user engagement. Based on these insights, Awaves BV thinks there is a potential mismatch between the product and the user's needs. To solve this problem, the company tried some improvements. They implemented a voting system to choose a certain mood and added user preferences, such as specific songs and mashups. On top of that, they tried some variations within music occasions, such as business events and gym locations. However, the improvement in overall streaming time has been minimal.

➔ *Therefore, the action problem is the low average streaming time* per user.

Although users are enthusiastic about the functionality and the design of the web application, they are not satisfied enough to use the web application more often. During direct feedback sessions, the standout aspect of the problem was that the dissatisfaction and reluctance to use the web application did not come from the functionality or the design but from its main feature, the music selection. Here, the lack of satisfaction was summed up by the following audible errors: *inconsistent genre, limited genres, inconsistent music energy level, unpopular music, low-quality music, no flow, no build-up and poorly executed transitions*.

To facilitate music streaming for Awaves Play, the company downloads the music files/tracks from a record pool into its database. A record pool is a platform where music is downloaded for commercial uses, allowing the company to play music at lower costs and with minimal copyright issues. In creating

a digital DJ mixtape, the music curators and mixing programmers typically follow these steps: 1. download the music, 2. store annotations and metadata, 3. decide a logical music playlist, and 4. adapt a mixing technique. After the company tested the application for functionality and execution of the mixing techniques, the music was played correctly. Which indicates that the action problem is most likely caused by the other three steps.

1.3 Problem definition

Now that we understand the company well, we can start to define the underlying problems. According to (Heerkens, 2017), when the action problem is found, the core problems must be identified. First, to determine the core problems, a problem identification is executed. This step explains why this problem exists. Second, the causes and effects are connected, and possible core problems are stated while creating a problem cluster. Last, out of the core problems, the most important core problem is chosen to be solved, and the gap between the norm and reality is stated.

1.3.1 Problem identification

To define the problem and find the core problems, it is essential first to understand why this scenario stands out. From the concept of Awaves Play, we may assume that it shifted the responsibility in music playing from the DJ to the consumer. Where a live DJ has full responsibility over the music, a competing streaming platform, e.g. Spotify, gives full responsibility to the user. By combining these concepts, a new division of responsibility has to be determined. The steps of creating a digital mixtape, which involves downloading, storing, playlisting and mixing, fall to Awaves BV's programmers and curators. These steps make them responsible for determining the target audience, analysing the songs, setting a musical direction and ensuring the accuracy of the mix. What makes it challenging is that the programmers and curators have to know in advance which complaints they have to prevent, while live DJ can adjust feedback live. So, currently, when the mix has audible errors, as mentioned in the problem introduction, Awaves cannot take live responsibility to correct the music. This leads to users hearing music that does not fit their taste, with their only choice to exit the application. Moreover, giving the responsibility to the user by providing, for instance, a skip button is not allowed by legal issues of having a radio streaming license. That is why the company can only focus on improving its music quality.

1.3.2 Problem Cluster

To provide an insight into connections between causes and effects of the problem, a problem cluster is created. We already know from the problem introduction that the action problem is caused by dissatisfaction with the played music. This dissatisfaction is described by specific audible errors mentioned in the problem introduction, highlighting the quality loss of the digital DJ of Awaves Play. In order to construct the problem cluster, the three steps of creating a mixtape and audible errors are connected. Then, the possible causes and solutions are added in search of their respective core problems. An overview of this process is given in the following table:

Awaves step →	Audible errors →	Possible cause →	Possible solution
Downloading music	Limited genres, unpopular music, low-quality music	Unknown preferences	Survey or case study music preferences
Store annotations and metadata	Inconsistent genre, poorly executed transitions	Incomplete music database	Custom classification system
Decide direction playlist	No flow, no build-up, inconsistent energy level	Incoherent music playlist	Automatic sequencing algorithm

Table 1: Problem identification

Downloading the music is the step in which all the music is collected. If there are some user complaints about a missing genre or the unpopularity or quality of a song, this is likely related to the decisions made during this step. A possible cause of the problems during this step is that the user's preferences need to be discovered. The reason why these preferences have yet to be discovered is that Awaves Play has a broad target audience. The group of students they are targeting has yet to be defined within specific genres or styles. As they have experienced within the process of implementing song preferences, the requests were quite broad. Usually, the intention is to solve the problem by direct user feedback. User feedback is an effective way to adjust specific preferences to increase user satisfaction. However, with an average streaming time below 8 minutes, minimal or no feedback is received.

Following the download step, the songs are classified using specific **annotations and metadata**. If this step is not perfectly executed, some songs may be played at inappropriate times. This leads to issues such as an inconsistent genres or transitions that sound offbeat. At the moment, Awaves is using a custom classification system, which is responsible for music annotation and adding extra metadata, such as if it is a local song or playable for everyone. A possible reason for these problems is that the classification database still needs to be completed. One of the causes of an incomplete database is the reliability of external data. Although a large amount of musical data is collected from record pools, it is sometimes accurate. As well, not every song is available; for instance, a lot of Dutch songs cannot be found in the record pools. That is why the curators and programmers must spend extra time identifying and correcting missing data. When the company is more experienced with conducting those corrections, some corrections are done faster.

When the previous steps are finished, **the direction of the playlist** is decided. When no flow, build-up or inconsistent energy level is experienced, the cause is likely to be found within the logic of the music played. A possible cause is the incoherence of the playlist. An incoherent playlist can originate from the fact that as well as the downloading problem, there needs to be more focus on a specific target audience. Having a database with random songs makes it hard to decide which song to play when automatically. As discussed in the problem introduction, some solutions were implemented where the audience could vote for a particular mood or a specific occasion was targeted. The outcome of the solutions was that the demand for music was not incoherent, but the music within these situations. Besides defining the target audience, the sequencing process of the playlist can also be improved. Currently, the music is sorted by metadata as genre, tempo and key to create a harmonic flow between

the songs. The approach within this process is relatively primitive and unsystematic. Primarily because incorporating two variables at the same time introduces a new complexity in maintaining a coherent harmony.

After identifying all possible causes, build the problem cluster. In the problem cluster, we can see how all the causes and effects are connected. Therefore, the core problems of the incoherent music playlist are the unsystematic sequencing process and the need for a target audience. The core problems of the unknown preferences are the need for a target audience and the minimal audience limit. Finally, the core problems of the incomplete music database are the limited knowledge about classification and the restricted availability of music and correct data. Resulting in the following problem cluster:

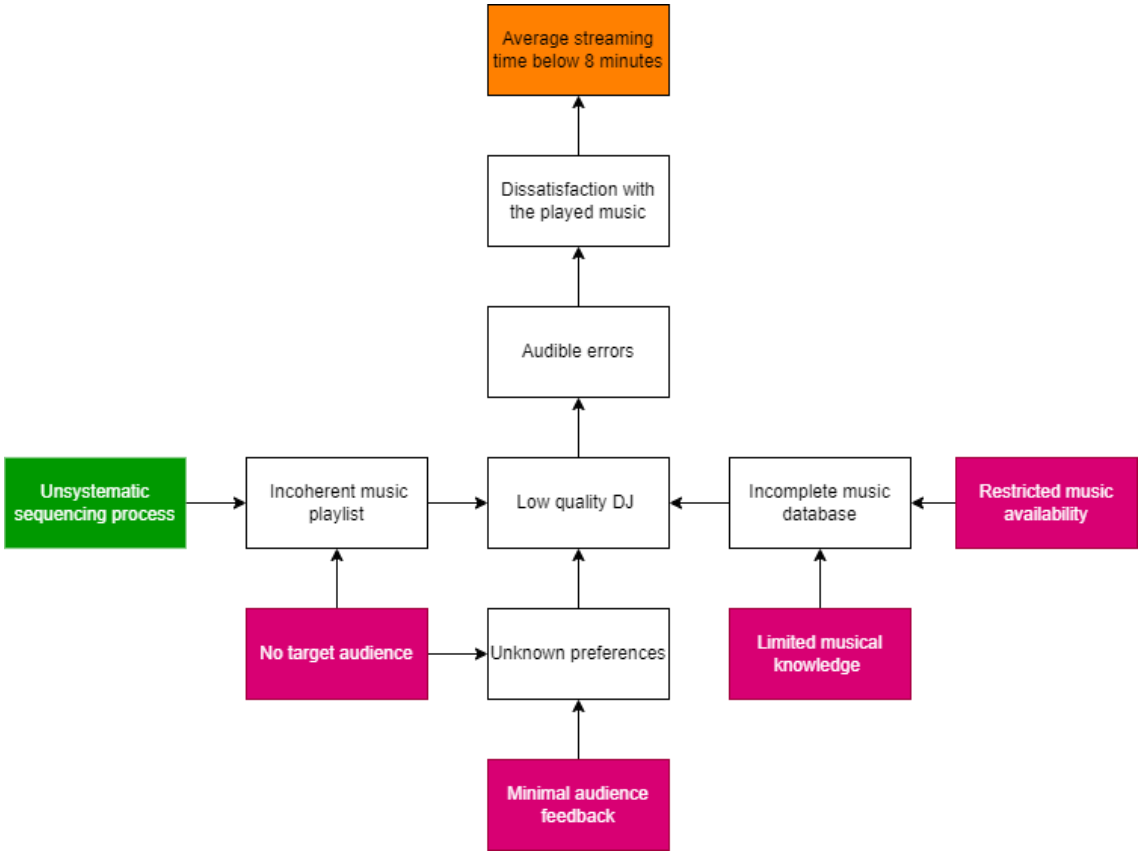

Figure 1: Problem Cluster

1.3.3 Choosing the core problem

Based on the methodology of the MPSM (Heerkens, 2017), only one problem is chosen as the problem that is solved. The most important problem, with the most minor restrictions due to unmanageable factors. For instance, solving the music availability includes some legal boundaries and deciding the target audience interferes with the founders' decisions. Nevertheless, there are still enough solutions to solve in order to increase the average streaming time:

- To **tackle the issue of minimal feedback,** a possible solution is to start an external survey or case study about the students' music preferences. Understanding audience preferences guides inclusion or exclusion criteria for specific groups from the target audience, thereby fitting the music more precisely to the user's taste.

- A custom classification system can be built to **reduce the musical knowledge needed** to increase the completeness of the music database. This system is independent of the external databases and gives Awaves control over the input and output of the annotations and metadata. The independence is achieved by an in-depth literature review about music classifications.
- A possible solution for **systemising the sequencing process** to improve coherence within the music playlist is to build an automatic sequencing algorithm. This algorithm deals with the complexity of multiple variables while maintaining a coherent harmony. An example of a problem that deals with these kinds of complexity is the traveling salesman problem, where the shortest route is determined by calculating the minimal distance between a certain number of locations.

The unsystematic sequencing process is chosen as the most important core problem in this research. Conducting a survey or case study would probably result in some interesting insights, but it needs more effectiveness on the dynamic and broad requests of the users. A custom classification system would provide the company with a solid foundation to control the consistency of the music database. The drawback is that this is a time-consuming procedure and requires external supervision for computer-related technical knowledge. However, constructing a sequencing algorithm is an achievable solution within the requirements. That is why it is chosen as the problem that is solved, and therefore highlighted in green.

Knowing the most important core problem enables us to define the gap between the norm and reality in more detail. Hence, as defined in the introduction, the reality is the 8-minute average streaming time. The streaming time is so low that minimal DJ mixing is executed before the user closes the application, and minimal streaming data is collected. The norm is that there must be at least some flow, achieved by decreasing the audible errors so that the users are satisfied enough to stay.

1.4 Research questions

According to the managerial problem solving method (Heerkens, 2017), after the problem definition the research approach is formulated. The approach requires the use of the research cycle, which describes the activities and knowledge that are needed to solve the problem. First, a research goal is stated. Next, the main research question and sub-research questions are formulated. At the end of this section, the scope and limitations are clarified.

1.4.1 Research goal

This research aims to find a way to optimise automatic playlist sequencing. This goal is achieved by exploring what kind of problem we face and how these problems have been solved. Followed by investigating how known knowledge might contribute to resolving this challenge. Ultimately, the research seeks to determine the feasibility of applying automation to this problem. The research is explanatory because relationships are researched, and no facts are established. Which is because we can already receive musical data and evidence about the low streaming time. Now, we want to know how these musical data can influence the streaming time. In order to provide an overview of all the relationships in a sequencing process, the following relationship model was used. The independent variable represents the core problem, and the dependent variables are the variables that are affected by the increase in the streaming time potentially.

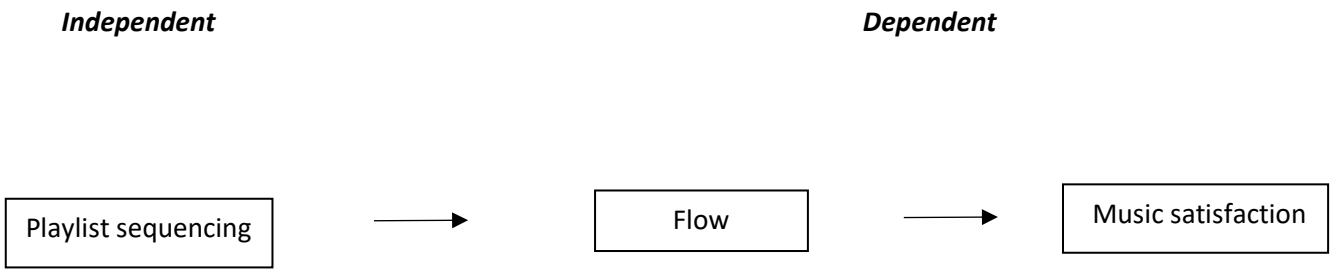

Figure 3: Relationship model

1.4.2 Research questions

The main research question is based on the research goal and how the variables are connected. The sub-research questions are created according to the remaining phases of the MPSM. Phase three focusses on analysing the problem. In phase four and five solutions are formulated, and a solution is chosen. In phase six the solution is implemented to the case of the company, and phase seven is based on evaluating the implementation. The main research question is defined as:

What is the optimal way to increase music satisfaction by improving the sequential flow through automated playlist sequencing?

With the corresponding sub-research questions:

- **How are digital mixtapes currently made?**

To create general awareness of the subject, exploratory research is done. Within this research, basic concepts and constructs about the subject are gathered. Here, the selected parts of the problem identification and cluster are re-examined. *(Phase three)*

- **How are digital mixtapes defined regarding playlist sequencing, flow and user satisfaction?**

In this research question, we zoom in on the sequencing part of creating a digital mixtape. While doing so, the relation with the other variables is researched. Causes are investigated, and the relationships between the problem and causes are documented. *(Phase three)*

- **Which models or methods exist in the literature for automatic music sequencing?**

A literature review is done to find what theory exists about automatic playlist sequencing. In order to select an appropriate theoretical perspective and theoretical framework, also, earlier solutions are investigated. *(Phase three)*

- **Which optimisation approaches are applicable to solve the problem?**

In this phase, approaches from within the scope are compared to earlier solutions of the theoretical framework. The solutions are described, followed by a decision-making process. Based on criteria the most applicable solution is selected. Then, the solution is explained in detail, including extra literature regarding this subject. *(Phase four) (Phase five)*

- **How is the solution constructed?**

Based on the chosen solution, the model is constructed, the data input is defined, and solution methods are explained. In addition, the implementation plan is drafted with a brief description of the needed activities. *(Phase six)*

- **How is the solution evaluated?**

Here, the chosen solution is evaluated. This is executed by comparing the affected situation to the desired situation. First, the technical validation is done by giving an example of the performance of all solution methods. Second, a brief sensitivity analysis is implemented with different data inputs. Last, the solution is practically validated by testing one of the scenarios with volunteers. *(Phase seven)*

1.4.3 Scope

The main scope of the research is to find out if there is an automated solution for playlist sequencing. As this is an Industrial Engineering and Management thesis, all solutions are within the theoretical perspective of this course. That means that in-depth information about artificial intelligence, machine learning and user data analysis is out of scope. Furthermore, this is a subjective concept and varies from person to person. Therefore, this study aims to improve the overall satisfaction of streaming music rather than trying to obtain specific, measurable outcomes (the number of songs streamed or the time spent listening). This approach acknowledges that the goal of streaming services is not just about delivering music efficiently but also about making sure the experience is personally meaningful and enjoyable for each user.

1.4.4 Limitations

The limitations of this research are categorised into three main areas: access, time, and availability. Currently, we only have access to knowledge about Excel Visual Basic as a programming language. In terms of literature access, we are restricted to academic papers. Given the commercial nature of the market, it is important to assume that relevant information exists in non-academic sources that we do not have access to. Usually, this research is designed to be completed within ten weeks, but more time is allocated to it due to preferences. However, conducting a survey about the model's effectiveness would take a lot of work to validate. As mentioned in Section 1.4.3 Scope, we would instead look at the overall satisfaction rather than the individual aspect. Additionally, we have chosen to focus on only one music genre as a subset of all music due to the variety of music genres, which was decided to ensure that the research is more contained. Availability is dependent on both data and the company's cooperation. Music data can be accessed through the Spotify API. Within this research, we assume that this data is accurate. Moreover, the involvement of Awaves in this research is less than usual. Due to managerial implications, a more independent approach is used in conducting this research.

2. Theoretical Framework

In this chapter, the first three sub-research questions are answered. In Section 2.1, we explain the overall DJ process. Section 2.2 links the relevant variables, and Section 2.3 summarises how this problem was solved earlier. The first research question we are answering is:

- **How are digital mixtapes currently made?**

2.1 How to Create a Digital Mixtape

To answer this question, we perform exploratory research. We gather relevant concepts and constructs from literature to understand the theoretical perspectives and terminology. The literature describes creating a mixtape using the following steps: 1. Recommendation 2. Classification 3. Sequencing 4. Mixing. Therefore, we choose to adopt this designation.

2.1.1 Recommendation

Within the literature, recommender systems are defined as: "*any system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options*" (Burke, 2002). The music recommendations are given by looking for similarities from one music to another or by giving preference from one user to another (Adiyansjah et al., 2019). By knowing historical data from users, charts are created, so those songs are found more easily. Recommender Systems use information filtering algorithms to select their music. Usually, those algorithms are selected from an extensive catalogue of music items, that are identified as most relevant to a target user (Schedl et al., 2015).

This process is described as → *Assisting the search for new music, which is helped by insights through user data.*

Within recommender systems, there are four different layers that applications take into account (Afchar et al., 2022):

- **Frontpage recommendation:** Recommending content at the home page without any user input
- **Music exploration/discovery:** Recommendations after a query. Which is item similarity in terms of artists' names, genres or lyrics.
- **Automatic playlist generation:** Playlists are automatically generated based on various criteria such as genre, mood, user behaviour or thematic content.
- **Automatic playlist continuation:** When a music playlist is given, songs are automatically added based on a sequence of seed tracks.

2.1.2 Classification

Within the literature, classification is defined as: "*the process of organizing data by relevant categories so that it may be used and protected more efficiently. On a basic level, the classification process makes data easier to locate and retrieve*" (De Groot, 2023). And **"***music classification is a music information retrieval (MIR) task whose objective is the computational understanding of music semantics*" (Won et al., 2021)

Therefore, this process is described as → *Translating music into an understandable language to make accurate decisions on the given information.*

Within the classification tasks, five types of data are extracted (Ng & Mehrotra, 2020):

- **Audio signal data**
  Musical features that are extracted directly from the audio signal, such as pitch, rhythm, and loudness.
- **Metadata and expert annotations**
  Descriptive information about songs that is added, such as genre, artist, and release year.
- **Audio attributes**
  Audio attributes of music content, e.g. danceability, energy and valence. These attributes help quantify and describe the track's acoustic characteristics and are used to understand how transitions in the audio attributes impact user perception of recommended music.
- **Social web data**
  Information that is obtained from social media platforms, including tags, ratings, and the social graph, which reflects user interactions and preferences.
- **Usage data**
  Data reflecting how users interact with music tracks, including play counts, skips, and inclusion in user-generated playlists.

2.1.3 Sequencing and Mixing

Within the literature, sequencing and mixing are closely related. According to Cliff (2000), the expertise of DJing highly depends on two skill levels: "*the macro-level of sequencing and the micro-level of mixing*." Deciding the order in which music tracks are played, involves considering the DJ's taste and the technical aspects, such as varying the music's tempo over the DJ set to match the energy levels desired for the event.

The Cambridge Dictionary (2024) defines a sequence as: "*a series of related things or events, or the order in which they follow each other*". Baccigalupo & Plaza (2006) claim: "*the quality of the recommendation depends mostly on the quality of past playlists: the more accurately they have been compiled by the users with a meaningful order, the more this order will be reflected in the output*". However, choosing the best option from an infinite amount of options within limited time is known to be difficult, even for experienced DJs (Hirai et al., 2018). As reported by Dias et al. (2017): "*Findings unveil that though the order of the songs is 'usually significant', there are no clear rules for sorting the songs in a playlist*."

Mixing is defined as: "*given a list of songs, a DJ selects a song with beats and sounds that are similar to a specific point in the currently playing song such that the song transition is seamless. Consequently the songs will be mixed as a consecutive song*" (Hirai et al., 2018). Mixing seamlessly is in the most basic version depending on the skill level of crossfading and beat-matching. Crossfading is fading out the volume of the outgoing song and fading in the volume of the incoming song. During the cross-fade, both songs are audible at the same time. That is where beat-matching is needed. When the both tracks are audible, the tempo has to be aligned. If that is not the case, a phase difference is heard. When the crossfading and beat-matching are executed correctly, the new song is mixed without any audible error. (Cliff, 2000).

The process of sequencing is described as → *Ordering the songs logically, within the selected music of the recommendation and classification phase.* Mixing is described as → *Smoothening the transitions of the sequenced songs, to make the moment of changing the song less observable.*

2.1.4 Answer research question

A digital mixtape is made by adapting the recommendation, classification, sequencing, and mixing. This process involves a series of decisions where the mixtapes are customised. These decisions are made

before and during the music performance and are dependable on existing knowledge and data. To meet the demand of the music listener while having a limited supply of music and user data, some problems are more challenging than others. Playlist sequencing is such an unsolved problem.

2.2 Playlist sequencing

In this part, we zoom in on the process of playlist sequencing. According to (Ng & Mehrotra, 2020): "*Music streaming is inherently sequential in nature, with track sequence information playing a key role in user satisfaction with recommended music*". Now that we have enough knowledge about 'sequence information' and the recommendation steps of the DJ process, we want to find out how creating mixtapes is related to the flow and satisfaction of playlist sequencing. Therefore, we answer the following research question:

- **How are digital mixtapes defined regarding playlist sequencing, flow and user satisfaction?**

2.2.1 Playlist

In the literature, playlist generation is defined as: "*given a pool of tracks, a background, knowledge database, and some target characteristics of the playlist, create a sequence of tracks fulfilling the target characteristics in the best possible way*" (Bonnin & Jannach, 2014), p.3). Before defining playlist sequencing, there has to be a clear distinction between two types of playlists:

- **User-curated list (Database)**

The user curated playlist is seen as the classic Spotify playlist. This playlist is defined as "*a repackaging of music in a form native to streaming platforms*". Curated by either a person or an algorithm (Bonini & Gandini, 2019). A repackaged user or algorithmically created list belongs in this research to the recommendation and classification step and can therefore be seen as the input for our solution.

- **Queue of songs that are played (Sequence)**

The second type of playlist is defined as: "*a set of songs meant to be listened to as a group, usually with an explicit order*" (Fields, 2012). The process of arranging a set of songs into a specific order enhances the listening experience. Unlike a user-curated list, which may focus solely on song selection, this playlist emphasises the importance of the songs' order (Furini, 2021). Therefore, this 'queue' of songs seen as a filtered and sequenced version of the user-curated list and is the output of our solution.

2.2.2 Flow

Flow is a "*psychological state characterized by complete absorption and focused attention in an activity, where individuals experience a sense of control and lose track of time. It is a state of optimal experience that occurs when the challenges of an activity match an individual's skills, leading to a state of deep engagement and enjoyment*". Flow was described by Mihaly Csikszentmihalyi (1990).

Despite the definition above, flow is decomposed in multiple ways. To find the most suitable definition for our problem, we stated the most appropriate by Chirico et al. (2015):

- **Intensity of Flow:** The degree to which individuals experience flow, characterised by deep immersion, enjoyment, and engagement in the activity.
- **Flow as a State:** Refers to the transient, immediate experience of flow that occurs in specific situations or activities where individuals feel fully engaged and immersed.

2.2.3 Satisfaction

The Cambridge Dictionary (2024) defines satisfaction as: "*a pleasant feeling that you get when you receive something you wanted, or when you have done something you wanted to do*". Depending on the situation, there are different levels of satisfaction. The satisfaction is influenced by the "*extent to which the list matches its intended purpose; fulfils the desired target characteristics; or is in line with the user's expectations, including aspects of taste, context, or mood*" (Fields 2011). The desired properties found are (Dias et al., 2017):

- **Popularity**: Popular and trendy songs should be included if a large group has to be satisfied.
- **Freshness**: The playlist should have something new to keep the listener captivated.
- **Homogeneity and diversity**: There is a demand for a balance between the homogeneity and diversity of a playlist.
- **Musical Features**: People value features differently. Some might prefer more energetic, and others want more peaceful music.
- **Transition and Coherence:** The quality of the DJ sequencing and mixing does determines the overall satisfaction. An interesting finding is that the second half of the music selection usually has a lower coherence.

2.2.4 Answer research question

Changing a sequence can influence the flow experienced by the user. Flow is seen as a focus level that reflects the match between the music that is played and the intensity of music listening. If a flow is experienced, a user could experience some satisfaction by receiving something that is wanted. By stimulating flow, the user is in a state of meeting expectations and feeling a sense of control. Experiencing flow can potentially, therefore, have a positive influence on user satisfaction.

2.3 Automatic playlist sequencing algorithms

In this section, we answer the third research question. First, we briefly address how algorithms have been developed over the years. Second, the most used problem-solving methods are explained. Third, the evaluation methods of these solutions are summarised. Finally, an answer is given to the research question. The third research question that we deal with is:

- **Which models or methods exist in the literature for automatic sequencing?**

2.3.1 History

From the literature, we notice that the development of sequencing algorithms was already underway before the rise of extensive streaming services. "*The term playlist was first used around the beginning of the 20th century when the radio was introduced*"(Dias et al., 2017). Concepts of mixing and coherence already gained relevance within concert programs around 1850. Particular inventions, such as audio recording and playback have made it possible to share music with a larger audience without the need for a live DJ or artist. Around 1970, the concept of continuous mixing was introduced together with the "*elimination of space between songs played back in sequence*" (Assante, 2008). When portable audio devices emerged, the usage of cassette tapes grew. Cassette tapes made combining and reordering different songs into personal mixtapes possible. This development made mixtapes the starting point for the recommendation and discovery of new music. The playlist took a further step during the transition from analogical to digital audio recording. With the formation of the World Wide Web and the MP3, sharing mixtapes without the physical constraints became possible. Today, we are familiar with web-based storage playlists, where music is streamed via a service (Spotify, Deezer) or bought online (Apple Music, Beatport). To give an overview of the history, the following figure was taken from the source:

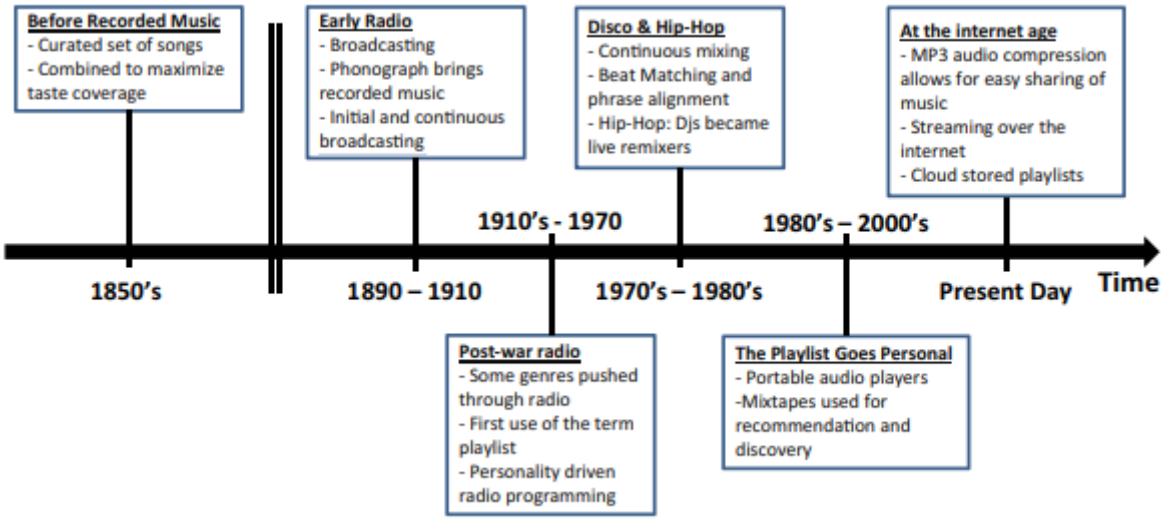

Figure 4: History of the Playlist (Dias, 2017)

Now that we know the timeline of the music playlist, we take a closer look at the consequences of these developments. "*The issue of generating automatic sequences of music titles that satisfy user preferences*" was brought up early (Aucouturier & Pachet, 2002, p.1.). Then, the main issues that emerged were the loss of expressiveness and incapacity to handle large music catalogues (scalability) (Balkema, 2009). Which is because the "*problem of generating a playlist given a catalogue of titles with musical metadata, and a set of arbitrary constraints is a combinatorial problem. It its full generality, the problem is NP-hard, as it boils down to a finite domain constraint satisfaction problem*" (Aucouturier & Pachet, 2002, p.1.), which still seems to be the case.

2.3.2 Methods

To solve the NP-hard problem, as discussed in the previous section, some solution methods have been constructed to tackle these issues. Within the literature, it is found that the following methods are used to generate playlists. Below, these methods are summarised, including their advantages and disadvantages (Bonnin & Jannach, 2014)(Dias et al., 2017):

- **Similarity-Based Algorithms:** Approaches that generate playlists by selecting and ordering tracks based on their similarities. Distance functions depending on the available data. The advantage is its scalability; the disadvantage is its homogeneity.
- **Collaborative Filtering**: Techniques that utilise community-provided ratings or preferences to recommend tracks or playlists. The advantage is that it adapts to preferences; the disadvantage is that it requires much data.
- **Frequent Pattern Mining**: Identifying common patterns or associations among tracks in existing playlists to inform new playlist compositions. The advantage is that it can reproduce what already exists; the disadvantage is that the quality depends on the input.
- **Statistical Models**: Utilising statistical methods to track selection and sequence, such as Markov and latent variable models. The advantage is that plenty of algorithms exist; the disadvantage is that the learning process is time-consuming.

- **Case-Based Reasoning**: Leveraging past cases or instances of playlist generation to solve new playlist creation challenges. The advantage is low computational complexity; the disadvantage is low scalability.
- **Discrete Optimisation**: Formulating playlist generation as an optimisation problem where the goal is to meet explicitly defined constraints. The advantage is that most targets are satisfied when background knowledge is accurate; the disadvantage is that most solutions are computationally complex and hard to scale.
- **Hybrid Techniques:** Combining multiple algorithms or techniques to leverage their strengths and mitigate their weaknesses. The advantage is overcoming individual limitations; the disadvantage is that it is more expensive and time-consuming than a sub-optimal solution.

2.3.3 Answer research question

There exist methods for the development of automatic playlist sequencing. Even more, these developments already started a century ago. As this is an NP-hard problem, there are some issues that still need to be resolved. Over the years, some solutions have emerged who have developed the music industry as we know it today. In the next chapter, the methods above are compared to the known approaches in the field of Industrial Engineering and Management.

3. Solution: Traveling Salesman Problem

In this chapter, the solution approach to the core problem is chosen. In Section 3.1, the possible solutions are presented. In Section 3.2 the solutions are compared to the problem. In Section 3.3 the chosen solution is explained further. In Section 3.4, the mathematical formulation of the solution is shown. In Section 3.5, additional literature is discussed. Within this chapter, the following research question is answered:

- **Which optimisation approaches are applicable to solve the problem?**

3.1 Optimisation approaches

With the theoretical framework of the previous chapter, we can choose a fitting solution from knowledge within the scope. Here, planning is known as one of the most complex tasks in operations management because the number of possible options increases rapidly as the number of activities and processes increases. We require is that we need an approach, which can automate playlist sequencing with available data. First, we are familiar with optimisation tools for dynamic and linear programming. Dynamic programming usually obtains solutions by working backwards from the end of a problem toward the beginning, thus breaking up a significant, unwieldy problem into a series of more minor, more tractable problems (Winston, 2004). Linear programming attempts to maximise or minimise the linear function of decision variables while satisfying some constraints. Second, we know planning heuristics for scheduling and the traveling salesman problem. Scheduling determines the sequence in which work is to be tackled, where some operations require a detailed timetable showing when jobs should start and when they should end (Slack et al., 2016). The traveling salesman problem, also known as the Vehicle Routing Problem, focuses on minimising the total routing cost with a fixed number of vehicles (Golden et al., 1984). However, finding the optimal solutions for significant problems in a reasonable time is still hard. Therefore heuristics "common sense methods" are applied. Last, we know about the stochastic process of queuing and Markov chains. A stochastic process is the study of how a random variable changes over time (Winston, 2004). The table below has been created to give a better overview of the approaches. Here, the existing methods of the literature from Chapter 2 are linked:

Known Approaches	**Purpose**	**Aligning Method (Section 2.3)**	**Input**	**Dangers**
Dynamic programming	Optimise decisions	Discrete optimisation	Decisions	Too many decisions
Linear programming	Optimise objective	Discrete optimisation	Objective function	No specific objective
Scheduling heuristics	Allocate resources	Discrete optimisation	Priorities and availabilities	Unknown priorities and unlimited availability
Traveling salesman problem heuristics	Determine optimal route	Similarity-based algorithms	Locations, distances, capacity	Abstractness music
Queuing/ Markov	Optimise service processes	Statistical method	Arrival and service ratios	Scarcity data and non-deterministic

Table 2: Overview approaches

By choosing the solution, we have to consider the dangers that come with combining this approach with a new case. The danger of choosing discrete optimisation as dynamic or linear programming is that scalability could be better. When more songs or variables are added, the computation time increases with increasing decisions, or the objective function loses its specific objective. Also, scheduling suffers from scalability due to the company's extensive music database. In addition, there are no clear priorities defined between songs, mainly because user preferences still need to be discovered. The danger of the traveling salesman problem heuristics as a similarity-based algorithm is the tendency to homogeneity, where music becomes too abstract. Finally, the risk of considering a stochastic solution as the Markov chains is that the user data is scarce. Knowing probabilities is necessary to solve a non-deterministic problem.

3.2 Selected approach

In order to decide which solution approach we are using, the MoSCoW rules are applied (Heerkens, 2017). Hence, we can prioritize what needs to be done now and what can potentially wait till later. In the table, demands and requirements are classified into different categories:

MoSCoW	**Automatic Playlist Sequencing**
Must Have	No delay
Should have	Scalable
Could have	Adaptive
Want to have but not have this time around	Optimal sequence

Table 3: MoSCoW rules

From the problem case, we know that the application Awaves Play functions well. Implementing a new solution might affect the system's functionality by, for instance, increased computation times. The following arguments have been chosen by the MoSCoW rules:

- **No delay** is a must-have requirement. When a delay is experienced, the application Awaves Play automatically performs worse than before implementation. Therefore, no delay (minimal computation time) is a must-have requirement.
- **Scalability** is the second important requirement. While improving the sequence, the target of the action problem is to increase the average streaming time. If the solution is not scalable, the system still fails after minor improvement.
- **Adaptivity** of the solution is the third important requirement. As mentioned in the possible core problems and literature, user preferences within the music industry are dynamic. That is why a possible solution could only work temporarily and loses its value over time. Also the effect of satisfaction can decrease by experienced homogeneity.
- **The search for the optimal solution** is the least important priority. Knowing that there exists a widespread taste of users, the optimal solution remains subjective. Within scheduling, providing an optimal solution is rarely attempted and somewhat satisfied by an acceptable solution.

Decision

Considering the should-have criteria, given that only the traveling salesman problem heuristic remains as a scalable and potential solution. In addition, the traveling salesman problem already has similarities with music sequencing. Namely, the DJ has to find the most logical order of songs to make the music sound coherent, whereas the traveling salesman has to find the shortest/cheapest route to deliver its packages. Besides, in the algorithm, the salesman can only visit each city once, while during DJ mixing, it is preferable to play not the same song again. Based on this information, we chose this approach to develop further in this research.

3.3 The Traveling Salesman Problem (TSP)

The chosen approach for solving the problem is a traveling salesman problem heuristic. Which is a classic optimisation problem, and belongs to the class of NP-hard problems (Jünger et al., 1995). The problem involves a salesman who has to visit a set of cities and return to the starting city, minimising the total distance of cost travelled. Where at Vehicle Routing Problems, cities can be visited multiple times, here the goal is to find the most efficient route that visits each city exactly once. A well-known example of a TSP heuristic is the nearest neighbour. Furthermore, variants of the traveling salesman problem exist, such as subtours and the distinction between symmetric and asymmetric distances. For the scope of this research, we keep it to the most basic version of the problem:

Dantzig-Fulkerson notation (Laporte, 1992):

$$min \sum_{i \neq j}^{n} d_{ij} t_{ij}$$

Subject to:

$$\sum_{i=1 i \neq j}^{n} t_{ij} = 1 \ \forall \text{i} \in \text{N}$$

$$\sum_{j=1 j \neq i}^{n} t_{ij} = 1 \ \forall \text{j} \in \text{N}$$

For all proper subsets $V \subset N, V \neq \emptyset$

$$\sum_{i \neq j}^{n} t_{ij} \leq |V| - 1$$

$$t_{ij} \in \{0,1\}, \text{i}, \text{j} \in N$$

Where:

- $\text{N} = \{1,2, \ldots, \text{n}\}$ is the set of locations
- d_{ij} is the distance between i and j
- t_{ij} is a decision variable with a value of 1 if and only if the salesman goes from i to j, 0 otherwise.
- Each location N can only be visited once.

3.4 Existing Applications

Before we start constructing the problem model, we first want to dive once again into the literature to discover if the traveling salesman problem approach has already been used with music sequencing. If that is the case, what are the necessary components that are considered? In the following table,

some applications of the traveling salesman problem in combination with music sequencing are highlighted:

Tool	Method	Chosen Variables	Goal	Creator
Travellers Sound Player	Audio and web-based similarities	Timbre, Instrumentation, Social background artist.	Music recommendation	(Pohle et al., 2007)
Music Map	Mapping songs into a two-dimensional map	Artist name, Song Length, Tempo	Reflect subjective preferences	(Hartono & Yoshitake, 2013)
Graph problem	Graph traversal using tempo, key and timbre	Key, Tempo, Timbre	Replicate professional music curators	(Bittner et al., 2016)
Smart Playlist Shuffle	2opt script in Python	Acousticness, Danceability, Energy, Instrumentalness, Loudness, Speechiness, BPM, Valence	Smooth transitioning	(Hildén, 2020)
Playlist Optimalisation	Nearest neighbour with multiple dimensions	Key, Tempo, Energy	Visualisation track ordering	(Holtes, 2021)

Table 4: TSP music sequencing applications

We have found from the literature that over the years, some articles have been published about the traveling salesman problem application to a music playlist. For instance, the first article used the TSP to order an extensive music database on a portable MP3 player logically, where a physical wheel was added to navigate within the route. The second article researched creating smaller neighbourhoods based on audio features to reflect user subjective preferences and solve them with a TSP algorithm. Moreover, the fifth article performed a TSP solution together with a principle component analysis to use all variables simultaneously.

Although these articles do not go in-depth about our case, they still provide an inside look at how this has been used or will be used in the future. What stands out is that each article has its interpretation of implementing the traveling salesman problem and chooses music variables differently. For example, the two oldest articles aim to reduce the of the music database's randomness. The third article is trying to replicate professional artists by reducing audible errors. Moveover, the last two articles investigate how they can increase song coherence. Therefore, we might assume that the interpretations and chosen variables may have evolved with the available data and resources. In addition, the second article recommended further research: "We believe that there is a strong psychological correlation between the selections of the attributes with the satisfaction of the users with respect to the automatic generation of the playlist". They are indicating the enduring relevance of this solving method.

3.5 Answer research question

To answer the research question; only some optimisation approaches are equally applicable. We have decided that delay and scalability are requirements that need to be met in the solution, and therefore the, traveling salesman problem heuristic is chosen as our solving method. This method, combined with automatic playlist sequencing is familiar, however the goals behind the problem cases vary widely. Hence, the necessary components are a well-considered selection of chosen variables, and a compatible application of the traveling salesman problem.

4. Construction model

This chapter aims to construct the model. In 4.1 TSP as a Music Sequencer, the mathematical model is modified, and the growth of decisions is explained. In Section 4.2 & 4.3, the model input and distance function are discussed. In Section 4.4, the solution methods are constructed and in Section 4.5, the results are highlighted. Answering the following research questions:

- **How is the solution constructed?**
- **How is the solution evaluated?**

4.1 The TSP as a Music Sequencer

Before constructing the model, the mathematical model needs to be modified. To adapt the traveling salesman problem for sequencing of music playlists, the following rephrases have been made to the mathematical model:

- The city N that has to be visited is renamed with song S
- The distance d_{ij} is the distance between the chosen song variables.
- The salesman is changed by DJ.

Resulting in the following model:

$$min \sum_{i \neq j}^{s} d_{ij} t_{ij}$$

Subject to:

$$\sum_{i=1 i \neq j}^{s} t_{ij} = 1 \ \ \forall \text{i} \in \text{S}$$

$$\sum_{j=1 j \neq i}^{s} t_{ij} = 1 \ \forall \text{j} \in \text{S}$$

For all proper subsets $V \subset S, V \neq \emptyset$

$$\sum_{i \neq j}^{s} t_{ij} \ \leq |V| - 1$$

$$t_{ij} \in \ \{0,1\}, \text{i}, \text{j} \ \in S$$

Where:

- S = {1,2, …, n} is the set of *songs*
- d_{ij} is the distance between the *variables from song* i and j [*]
- t_{ij} is a decision variable with a value of 1 if and only if the *DJ* goes from i to j, 0 otherwise.
- Each *song S* can only be chosen once.

[*] See 4.3 Distance

4.2 Data input (Audio Features)

Now that we have defined the mathematical model, we can start by defining the model's input. The data input needed are the musical songs' respective audio features. As mentioned in Chapter 2, audio features are labels for extracted data from the classification step. Using Spotify API data, we can access an extensive database of classified songs. The features that are considered most important are tempo, energy, danceability, valence and key. tempo is necessary for the beat-matching process of song mixing. Energy, danceability and valence are used to determine the direction of the mix. Moreover key is used to make the music sound harmonically correct. As we have seen in the existing applications and notice within the Spotify API, these features currently appear to be the most relevant. Further explanations of audio features are given (Spotify for developers, 2024):

Tempo

The tempo is the overall estimated pace of a track in beats per minute (BPM). In musical terminology, tempo is the speed of a given song and derives directly from the average beat duration. This audio feature sets the mood (Kody, 2021). Also, here, it is best to stick with the same tempo. After a moment, it gets too uniform, and a change has to be made. An overview of some examples is given below:

GENRE	BPM
Hip Hop	85–95 BPM
Glitch Hop	105–115 BPM
Techno	120–125 BPM
House	115–130 BPM
Electro	128 BPM
Dubstep	140 BPM (with a half time, 70 BPM feel)
Drum and Bass	174 BPM

Figure 5: Genre BPM (Kody, 2021)

Energy

Energy is a measure from 0.0 to 1.0, representing a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.

Danceability

Danceability describes how suitable a track is for dancing based on a combination of musical elements, including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is the least danceable, and 1.0 is the most danceable.

Valence

Valence is a measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry)

Key

The key is the note or chord the track is in. Integers are used to map keys using standard Pitch Class* notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D.

(*Pitch Class Notation is a way of naming the tones of the musical chromatic scale without regard to octaves. In this notation, the 12 unique tones of the Western musical system are identified by a number from 0 to 11.) (Wikipedia, Pitch Class)

Mixed In Key is an example of a software application used by DJs and music producers to analyse the music classification of songs and tracks. One of their inventions is the Camelot Wheel, shown in the figure below, a version of the circle of fifths. This wheel guides them to choose the right song on their key. The rule of thumb is to stick to the same key to make the music sound harmonically correct. However, after a while, this becomes homogenous. The solution is to go up or down one increment in a straight line on the wheel.

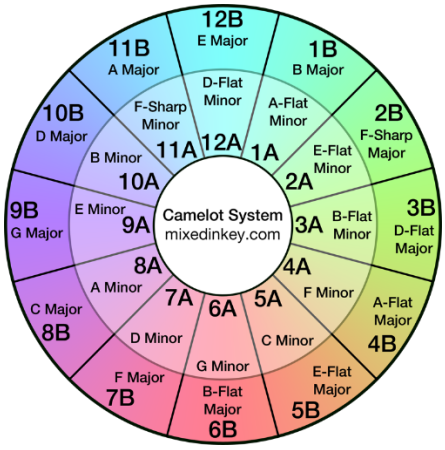

Figure 6: Camelot Wheel (Mixed in Key)

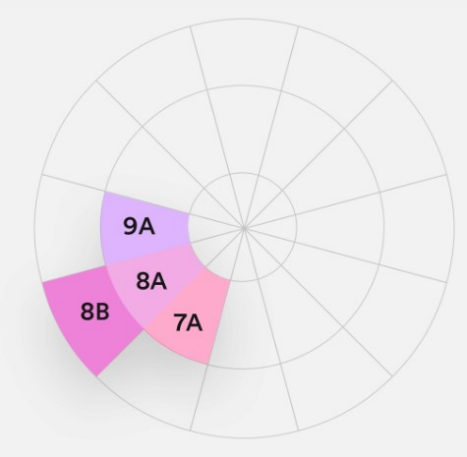

Figure 7: Solution harmonic sound (MiK)

4.3 Distance

After defining the most relevant audio features, we discuss the distance function. Due to the different scales, also different distances have to be determined. In this research, we use two kinds of distances: The Euclidian distance and the Key distance.

4.3.1 Euclidian distance

The Euclidian distance is known as its most general metric, where mainly the Pythagorean theorem is used to determine the distance:

$$d(P,Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$$

Where $P = (p_1,p_2)$ and $Q = (q_1,q_2)$ are two different songs with two chosen audio features.

However, for this problem, we are adding a normalisation formula to fit the audio features within the calculations. If needed the distance can be as well be calculated for multiple dimensions.

Normalisation

The data is normalised before we actual distance is calculated. Which is to make sure the chosen songs have a relatively equal value concerning the other coordinates. Which is done by the following formula:

$$\frac{value - \min}{\max - \min}$$

Figure 8: Normalisation formula

In the table below a practical example of value normalisation is given. Variable X is an example of an energy, danceability, or valence type of value, and variable Y, is an example of a tempo type of value. As we can see at norm X and norm Y the lowest value now has the value zero, and the highest value has the value one. Each value in between gets a relative value in between.

Variable X	**Variable Y**	**Norm X**	**Norm Y**
0,1	100	0	0
0,4	120	0,5	0,4
0,75	140	0,8125	0,8
0,9	150	1	1

Table 5: Example value normalisation

Multiple dimensions

Within the original traveling salesman problem, only two coordinates are needed. However, within music theory, more than two variables are relevant simultaneously. For this situation, we have to first look at the theory of dimensions. The Euclidian distance namely applies to multiple dimensions (Black, 2004). This means that the new distance is calculated by adding one dimension to the formula. The principle holds that the shortest distance can also be found if the distance is calculated.

For three dimensions: $d(P,Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2}$

Where $P = (p_1, p_2, p_3)$ and $Q = (q_1, q_2, q_3)$ are two different songs with three chosen audio features.

And for N dimensions: $d(P,Q) = \sqrt{\sum_{i=1}^{N}(p_i - q_i)^2}$

Where $P = (p_1, p_2, \dots, p_n)$ and $Q = (q_1, q_2, \dots, q_n)$ are two different songs with n chosen audio features.

The advantage of using this theory, is that the calculated distance becomes more representable to reality. The disadvantage is that it comes at the expense of simplicity in terms of possible scenarios and visualisation. To give an impression, a scenario with one extra dimension (3D-model from an existing application) is visualised in Figure 9. Due to the scope and limitations this research will only focus on two dimensions.

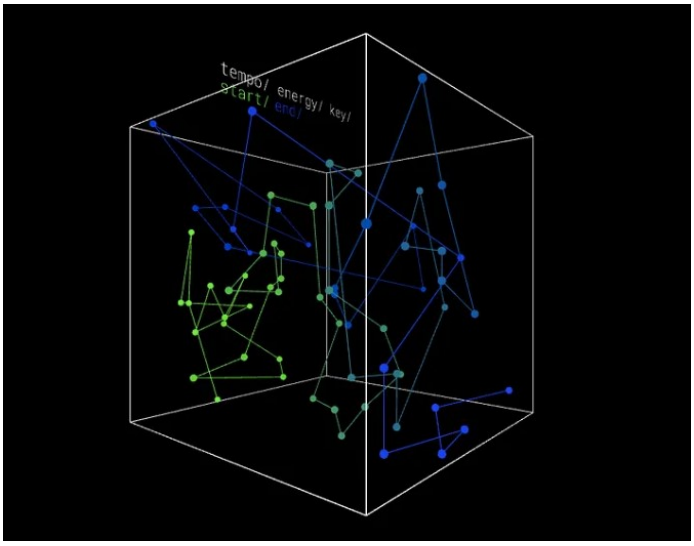

Figure 9: Visualisation of a TSP solution (Holtes, 2021)

4.3.2 Key distance

The key variable is unique because of its chromatic scale, as mentioned in Section 4.2 Key. Instead of using the Euclidean distance, a new distance has to be used to find the optimal sequence. To have a clear overview the key variables' position, we use of the Camelot Wheel. Which is an ordinal scale with cyclic properties. In Figure 6, it is shown that every movement around the wheel is counted as one step. So, all possible steps range from zero to a maximum of seven steps. The following figure gives an example of one versus seven steps is given, indicating a significant difference in harmony.

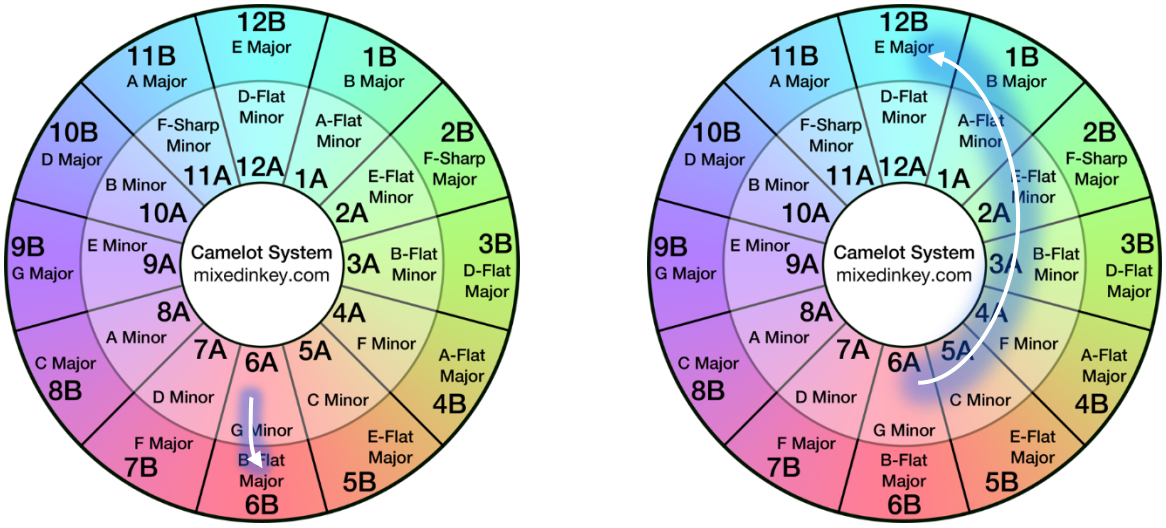

Figure 10: Distance Key Camelot Wheel (Mixed in Key)

As we can see in Figure 10, one step represents distance one, and the maximum distance is distance seven. This gives in total a total of eight possible distances. If the distance is counted eight or more, there is a shorter route somewhere else. When the set of songs is equally distributed in key, the following probabilities of a key distance are used. Also, here, the distances are normalised.

Distance	Probability	Normalisation
0	1/24	0
1	3/24	1/7

2	1/6	2/7
3	1/6	3/7
4	1/6	4/7
5	1/6	5/7
6	3/24	6/7
7	1/24	7/7
7<	0	Not applicable

Table 6: Distance Key

Decision

Regarding the scope of this study, we decided to only use the Euclidean distance for the remainder of this research. Despite the fact that key being an vital audio feature and the applicability to use the TSP approach, it brings new complications when combining the key distance with the Euclidean distance. First, because the Euclidian distance is based on a fixed location in the Euclidian space and the key distance based on a relative position of the starting point in a circle. Second, it comes along with the computational problem that key is already using more than one dimension. In addition, we have to consider that key is a qualitative variable and the Euclidean distance variables are quantitative, and therefore weights are necessary to balance the distance values.

4.4 Solution methods

With all the information about the solution and data input collected, the solution methods are selected. First, a random sequence used as a representative of the old situation. Next, we use brute force in order to find the optimal distance. Because brute force runs into computational complexities two near-optimal alternatives are chosen: The nearest neighbour heuristic and the evolutionary method. In the table below the advantages and disadvantages of the solution methods are highlighted.

Solution Method	**Random (Old method)**	**Brute Force**	**Nearest Neighbour**	**Evolutionary**
Advantage	Unlimited song capacity	Optimal	Simple	Near-optimal
Disadvantage	Unsystematic	Limited song capacity	Sub-optimal	Complex

Table 7: Comparison of solution methods

4.4.1 Brute force

One of the most robust methods of solving the traveling Salesman Problem is brute force, which is simply said: visiting every option possible. The minimum of songs needed for a mix is two, and from then, a first transition is made. However, the amount of permutations grows exponentially. In the evaluation, we decided to use the amount of eight songs. For a mixtape, assuming that the average song length is three minutes, this would mean a length of (8 * 3 minutes) 24 minutes. If we want to hear the optimal sequence, it would already take nearly two years (40320*24 min) to listen to all possible sequences. Which emphasises that the original way runs into computation complexities. The advantage of this method is that the optimal solution is always found. The disadvantage is the limited

song capacity that can be used, which is not favourable in case of scalability. For more clarity, we can check the following permutation table.

Number of songs (n)	Permutations (n!)
1	1
2	2
3	6
4	24
5	120
6	720
7	5040
8	**40320**
9	362880
10	3628800
11	39916800
12	479001600

Table 8: Example Permutations

4.4.2 Nearest Neighbour (NN)

As we have seen in 4.4.1 Brute force, exact solutions are only sometimes possible. That is when we have to switch to heuristics. As mentioned at 3.3 Traveling Salesman Problem, one of the well-known TSP heuristics is the nearest neighbour. The nearest neighbour entails constructing a route by adding the nearest city to the end of the route (van der Heijden & van der Wegen, 2017). Here, the algorithm chooses a starting point (e.g. i_1), checks for a song that is closest to i_1 (e.g. i_2), and connects this city with i_1. After that this step is repeated until the last song which has not been visited yet is added to the end of the tour. The advantage of this method is the computational simplicity. The disadvantage is the sub-optimality of the generated answer. Figure 11 gives an example of how this process evolves.

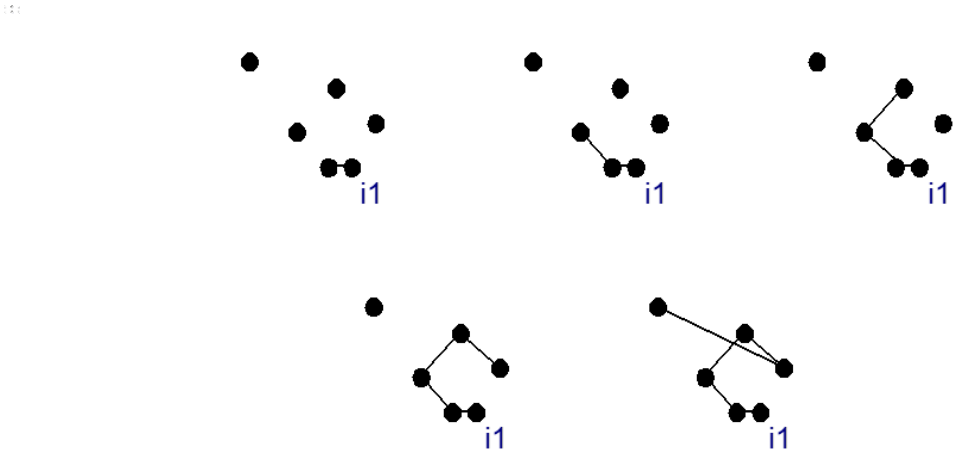

Figure 11: Example Nearest Neighbour (van der Heijden & van der Wegen, 2017)

4.4.3 Evolutionary (Excel Solver)

When searching for an Excel implementation of the traveling salesman problem, the evolutionary method of Excel Solver is mentioned multiple times as an alternative solution method to brute force (Rasmussen, 2011)(Gusron, 2023)(Sam, 2022). The method is a genetic algorithm (Young, 2024), that works by starting with a random population of input values. The outcomes of these values are compared to the target value (minimal distance). Then, the values closest to the target are used to produce offspring for the second population. Which goes on until there is very little change in the objective function. In the figure below, this process is visualised:

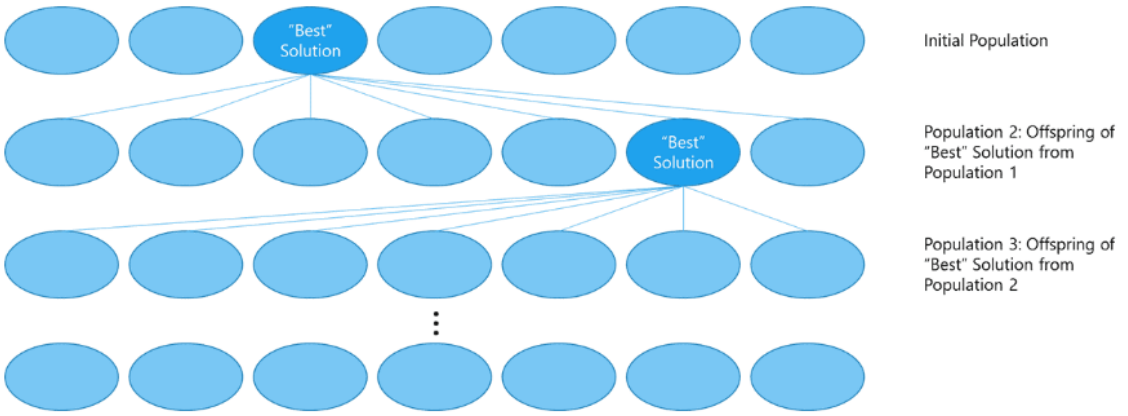

Figure 12: Evolutionary method (Young, 2024)

4.4.4 Implementation plan (pseudocode)

The implementation plan is drafted to describe steps that are executed to measure the solution methods. Using this plan generates four sequences based on each solution method. The execution of this plan can be found in the Appendix A.

General (Random)

1. Create a table with the randomly chosen songs from 1 to S and select two or more Euclidian distance audio features.
2. Choose the first song as a starting point.
3. Create a second table where the distance (d) is calculated based on the Euclidian distance.

- **Brute Force**

4. Create a new Excel sheet and put down the permutations of every possible sequence. For instance, with three songs (1,2,3) that is: {(1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), (3,2,1)}
5. Combine the second table (Distance table) with the sequences of the new Excel sheet.
6. Calculate the total distance of every sequence by summing the individual distances.
7. Find the minimum distance of all the total distances, which is the optimal distance.

- **Nearest Neighbour (VBA)***

* The Visual Basic code can be found in Appendix B

4. Take parameter MinDistance and give it a large value, e.g. 10000.
5. Calculate the distance between the first and second song of the selection.
6. Check if distance is smaller than the large number.
7. If it is smaller than MinDistance, the value is updated.
8. Repeat this for every possible distance.
9. Mark the song with the smallest distance as visited, e.g. Visited(False) → Visited(True)
10. Create a third table where the value of the next visited song is placed.
11. Create a loop where it is checked if the city is not visited yet.
12. Repeat step 4 until all songs are marked as visited.

- **Evolutionary**
  4. Create a third table with a chosen sequence and the matching distance from the second table.
  5. Sum the total distance under the third table.
  6. Use the Excel solver to calculate[*] the minimal distance of the sequence. (Select the evolutionary solving method and choose that songs can only be chosen once).

4.5 Results

After executing the solution methods in Excel and algorithms to the Visual Basic coder, we were able to read some results. In this section, we briefly visualise the music sequencing model and summarise of the findings. First, an example of the solutions' performance is given. Next, a sensitivity analysis is executed for different scenarios. Last, the methods were implemented in an actual mixtape and assessed based on user preferences.

4.5.1 Assumptions

In conducting this evaluation, several decisions are made based on assumptions. These assumptions were necessary to generalise the results and manage the complexity of the study.

- **Song Selection:** The songs selected for the evaluation are collected from Spotify playlists based on three different genres: EDM (Electronic Dance Music), Hip-Hop and Rock. Assuming that these popular genres have different audio feature values.
- **Variable Selection:** The selected variables are filtered by their practical and computational applicability. Valence is a variable that values musical positiveness. However, this subject is neither used nor reflected yet in the company. On the other hand, key is a valuable variable, but, as discussed in 4.3.2 Key distance, combining this feature with the others will create a new mathematical problem. Therefore, both variables are excluded. We assume that the other variables are valuable enough to test the effectiveness of the solution methods.
- **Number of songs:** The number of songs selected is nine, based on the maximum available permutations for brute force in Excel. Given that the first song is fixed, 8! (40320) different sequences are possible. We assume that 40320 possible sequences are representable for comparing the solution methods.
- **Solver Settings**: The input settings are based on literature within Excel solver. In addition, the amount of iterations decreased with the extension of the Excel file due to the computational power of the laptop. Assuming that the answers to the evolutionary method may differ with different settings.

[*] The *settings can be found in Appendix C*

4.5.1 Performance

First, a simple technical validation is executed. Here, an EDM playlist is selected together with the variables energy and danceability*. The optimal solution is calculated by using the Euclidean distance (4.3.1) formula and compared to the old situation (random) and the two solution methods (NN & Excel Solver). In Figure 12, a visual representation of the sequences is given. What stands out is that the old situation has multiple line intersections, and all the solution methods have nearly no intersections. Moreover, it is noticeable that Brute Force and Solver differ in one song in their sequence. In Table 9, the distances of the sequences are collected and compared. As expected from the theory, all solutions have a smaller distance than the random sequence.

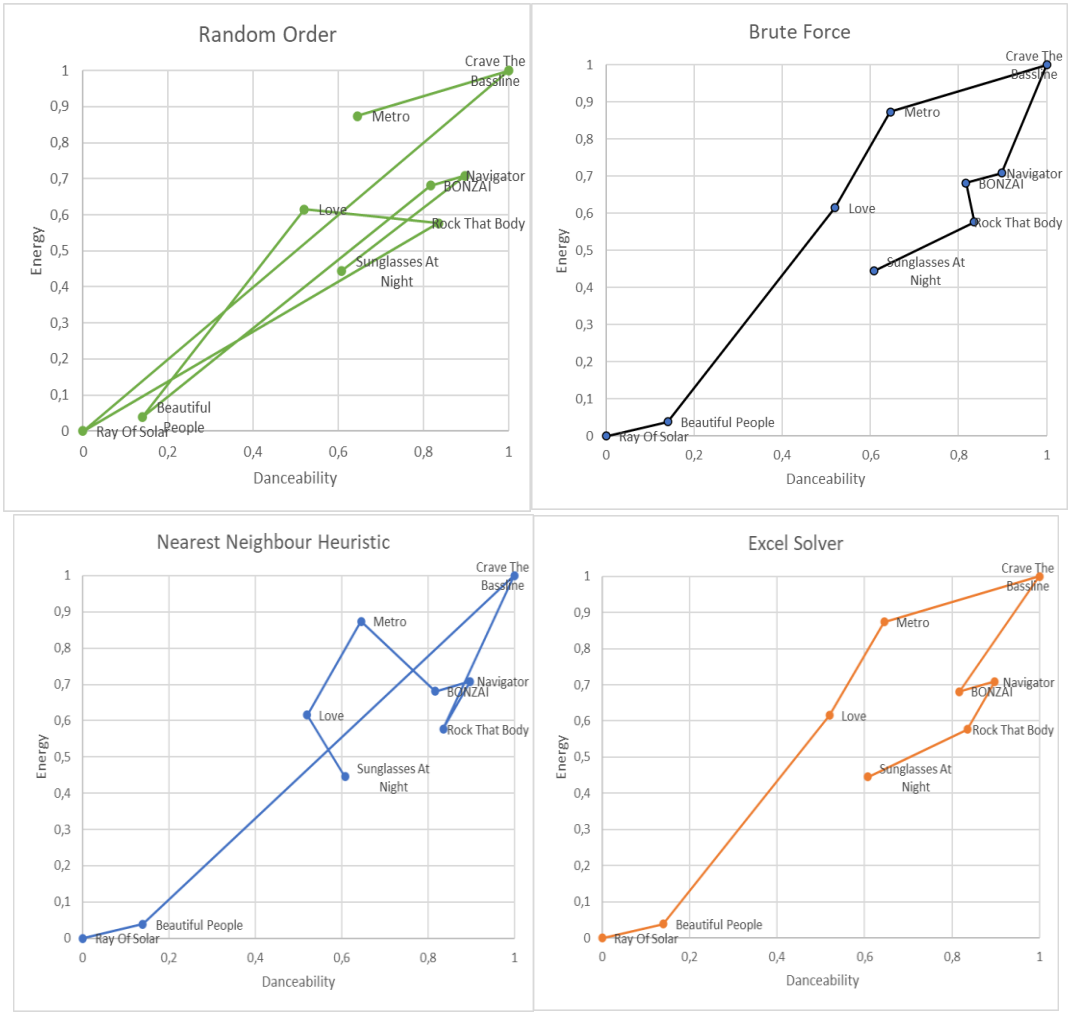

Figure 12: Sequence solution methods

Method	Random	Brute force	Nearest Neighbour	Solver
Distance	5,227	2,263	2,858	2,362

Table 9: Comparison of solution methods

* See Appendix E

4.5.2 Sensitivity analysis

For the sensitivity analysis, three different playlists are combined with three different song variable combinations. The collected playlists each consist of a different music genre. EDM, the company's primary genre, was selected first. Next, Hip-Hop and Rock are chosen as comparable popular genres. The variables left (Danceability, Energy and Tempo) are chosen for the different scenarios. As shown in Table 10, EDM songs have a wide range of danceability, a high energy level and a relatively high tempo. The Hip-Hop songs have a high danceability, broad energy level and an intermediate tempo. Last, the Rock songs have a relatively low danceability rating, a broad energy level and a broad tempo range.

Genre	**EDM**	**Hip-Hop**	**Rock**
Danceability	(0,355-0,786)	(0,737-0,852)	(0,414-0,586)
Energy	(0,802-0,984)	(0,547-0,821)	(0,631-0,945)
Tempo	(125-140) BPM	(93-117) BPM	(82-130) BPM

Table 10: Input variables

The following table compares all the different playlists to the solution methods. The first scenario corresponds to the performance example in Figure 12 and Table 9. In addition, the maximum distance, calculated by brute force, is added to indicate how the random solution performs. The most striking point of this analysis is that all the solutions outperform the random solution. It is also noticeable that the nearest neighbour heuristic sometimes performs better than the Excel solver. Furthermore, no exceptional results were noticed.

Playlist (variables)	**Random**	**Brute Force**	**Nearest Neighbour**	**Excel Solver**	**Max distance**
EDM (Dance, Energy)	5,227	2,263	2,858	2,362	6,273
EDM (Tempo, Energy)	5,425	2,769	3,107	2,915	6,523
EDM (Dance, Tempo)	5,753	2,799	2,799	3,135	6,601
Hip-Hop (Dance, Energy)	4,334	2,649	2,896	2,655	6,322
Hip-Hop (Tempo, Energy)	4,501	2,63	2,779	2,693	5,479
Hip-hop (Dance, Tempo)	5,002	2,657	2,885	2,657	6,05
Rock (Dance, Energy)	5,145	2,649	2,649	2,649	6,248
Rock (Tempo, Energy)	5,774	2,718	3,264	2,718	6,518
Rock (Dance, Tempo)	5,807	2,387	2,392	2,729	7,589

Table 11: Sensitivity analysis

4.5.3 Practical validation

To practically validate the solution, a group of four different sequences have been compiled into a mixtape:

1. Shortest distance (*Brute Force*)
2. Sub-optimal distance (*Nearest Neighbour*)
3. Random distance
4. Maximum distance (*Brute Force*)

Here, the same songs and sequences at 4.1.5 Performance were used for this analysis (EDM, Danceability & Energy). During the setup of this validation, some adjustments were made to make the assessment manageable:

- First, the solver sequence has been omitted to prevent ambiguity about the similarity between the solver and the brute force sequence. Instead, the maximum distance, calculated by brute force, has been added as a "worst-case scenario".
- Second, to decrease the total length of the mixtape, the duration of the songs was shortened to 7 bars (~13 seconds); here, the first drop (most intense section) of every song was selected as a fragment.
- Third, a pause of 1 bar (~1,5 seconds) was added between the songs to intensify the focus on the sequence instead of the transitions.

Subsequently, 23 volunteers from within the social environment were asked to rank the sequences based on their preferences from 1 (best) to 4 (worst). The titles of the samples were made anonymous. The results of this assessment are shown in the frequency table below, where each point represents the number of times the sample is given that respectable ranking. The implementation expected that the shortest distance would perform best, the sub-optimal distance second, the random distance third, and the maximum distance last. What stands out from this practical assessment is that the shortest distance (Min) has a significant difference compared to the others. The maximum distance (Max) is ranked with second highest average ranking, the random (Random) sequence third and the nearest neighbour (NN) is ranked fourth.

Sample	**Rank #1**	**Rank #2**	**Rank #3**	**Rank #4**	**Average Score**
NN (1)	3	3	11	6	2,867
Random (2)	4	7	3	9	2,739
Max(3)	6	6	7	4	2,391
Min (4)	10	7	2	4	2

Table 12: Frequency analysis & average ranking

4.5.4 Answer research question

In this chapter, we combined all the information gathered from the previous chapters, constructed the model and introduced solution methods to implement. Here, we discovered that despite the possibilities (multiple dimensions and the key distance), the model quickly runs into new computational challenges and therefore, had to be excluded. Later, during the evaluation, the solutions were theoretically validated, and the chosen methods improved in sequence distance. The sensitivity analysis shows that this is still the case for different scenarios. Last, volunteers have been practically validated four sequences, resulting in a different ranking than expected.

5. Discussion model

This chapter discusses a brief report of the model. Its purpose is to look critically at the decisions that have been made through this research.

General remarks

This research has shown that the traveling salesman problem is used for optimising automatic sequencing. Based on the model and the definition of flow, we might also predict that this solution leads to an increased music satisfaction. What stands out most is the convenience of building the model and proving the concept. Later, we faced the challenge of validating all possible scenarios. While generating one answer, it is likely to assume that more scenarios are valuable to improve music satisfaction. The traveling salesman problem usually minimises the total distance because every extra kilometre has extra costs. In this model we calculated the distance by its most basic form. In reality with music optimisation, the distance becomes subjective, which means that every optimal solution differs for each situation. Nevertheless, there are still some aspects to be critical of before we can draw further conclusions:

Technical Validation

First, the validations performed thus far are still at a preliminary stage. Increasing the number of repetitions, testing with a greater variety of playlists, and exploring more scenarios with different quantities of songs could enhance the robustness of the results. Within the limitations of this research, we had to restrict ourselves to the most basic form.

Song selection

Second, constructing the model showed that some songs have a large deviation on one or more variables, which has a significant influence on the normalisation. Such outliers raise the question whether songs need to be filtered in advance based on these values. However, how the songs should be filtered is something we have not yet investigated. To guide this explanation, two examples are provided in Figures 21 and 22, where the effect of the outliers can be spotted (Each dot represents a song).

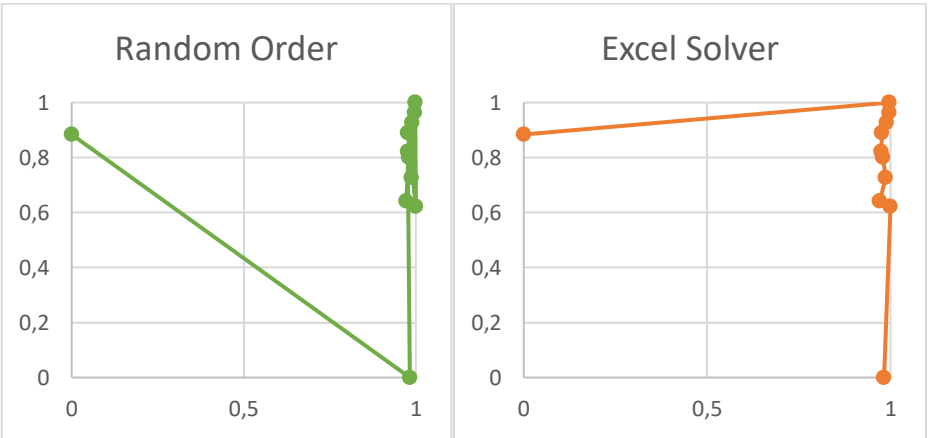

Figure 21: Example Single Outlier

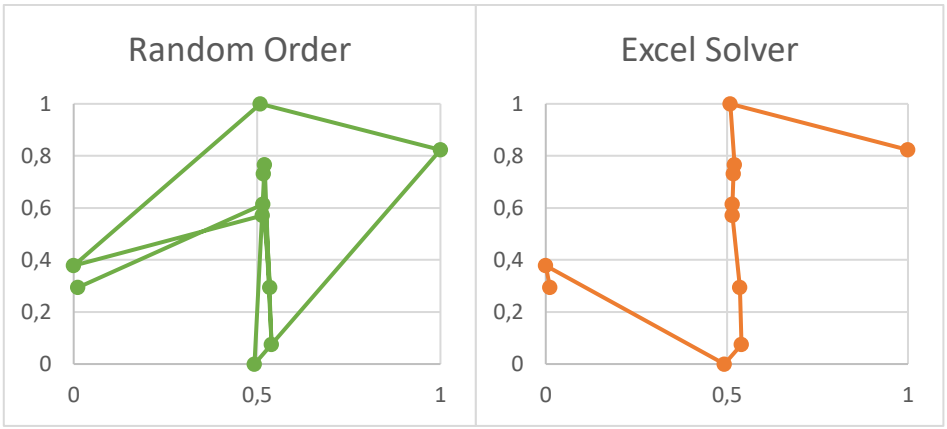

Figure 22: Example Both-sided Outlier

Dependent on Recommendation and Classification

Third, we cannot deny that the sequence output depends on the musical input's recommendation quality. In this research, we have chosen arbitrary songs and an arbitrary first song. However, in reality the song selection has a direct influence on the overall experience of the played music. On the other hand, sequence output also depends on the classified data. Two of the chosen audio features are objective (tempo, key), and the other three are subjective (danceability, energy, valence). The result is that the sequence is based on both objective and subjective decisions, which can influence the reliability of the optimal solution.

Input Variables (weights)

Fourth, we also have decided on which features/variables to include. Since we do not know which are more effective in which situation, we have now treated them equally in the model. In reality, it is conceivable that the effectiveness can fluctuate and that, for instance, there are differences in satisfaction with optimised sequences per genre. In addition, it is yet to be determined how many variables there are needed to be included simultaneously. The more variables, the more representative the outcome, but more variables also result in that distances are drawn to the mean. Hence, we may lose the overview of why songs are sequenced and when.

Capacity limitations

Fifth, despite a few related articles, the amount of research on this subject could be much higher, leading to capacity limitations in the early stage of the implementation. For simplicity, Excel VBA was chosen as the primary programming tool. However, the integration of multiple functions and calculations presented challenges to automation. Moreover, some scenarios still need to be explored even with additional time dedication to this research. For instance, combining key to the Euclidian distance and using more than two variables have yet to be investigated.

Practical validation

Last, the implementation was facing several challenges to create a realistic validation. Despite shortening the songs, the volunteers indicated that they found it difficult to hear and remember the differences between the mixtapes. Besides, we might assume that the setting, where the mixtape was listened to, could influence the experience of the variables. For this validation, the chosen variables were danceability and energy, where it is imaginable that the effect of the variables can differ when the volunteer was, for instance, sitting on a chair or walking outside. The volunteers also stated that they had different focus points, even though they were asked the same. Some focused on the overall feeling of the mixtape, while others were focusing on the individual transitions of songs.

As stated in this research, subjectivity plays a significant role and every situation could need a different assessment. Therefore, it is challenging to provide significant conclusions about whether the theory behind the solution improves music satisfaction.

6. Conclusions & recommendations

In this chapter, we are answering the main research question:

What is the optimal way to increase music satisfaction by improving the sequential flow through automated playlist sequencing?

While doing so, we summarise all the chapters above in 6.1. In Section 6.2, we give recommendations for further research.

6.1 Conclusion

In this research, we have analysed the problem of the case company Awaves B.V. After defining the core problem of lacking a systematic sequencing process, we went through the literature on how a digital DJ mix should be automatically sequenced. In the literature, we found that the sequencing process is part of four steps of how to DJ: Recommendation, Classification, Sequencing and Mixing. While zooming in on the third step, we discovered that investing in sequencing can stimulate flow experience and increase user satisfaction. With this background knowledge, we went through the literature again to find out whether any research has already been done to improve automatic sequencing. Within the literature, we found the beginning period of the concept of music sequencing and how this has technologically been developed over time. Out of these developments we stated the most relevant methods nowadays are used to optimise sequencing.

In the second part of the research, we combined the theoretical framework with the optimisation approaches within our scope. Hence, we decided which solution approach to use. This was the traveling salesman problem heuristic due to its convenience in applying, scalability, and similarities with the DJ process. Next, we defined the mathematical model and summarised some existing applications. After that, we constructed a model and implemented three solution methods to compare with Brute force, the nearest neighbour heuristic and Evolutionary (Excel Solver). All solution methods were able to automatically sequence a music playlist while using a traveling salesman problem approach. Based on the existing applications, five variables were selected: tempo, energy, danceability, valence, and key. During the construction, we understood that the audio feature key needed a different approach. For this, we used the Camelot wheel to determine the distance. However, this variable appeared computationally complex to combine with the other variables, so it was excluded from the evaluation.

In the end, the implementation of the solution was evaluated. A theoretical validation, a sensitivity analysis and a practical validation were executed. The solution methods performed better than the old situation (random), and repeating this for multiple scenarios gave the same results. The execution of the practical validation showed the difficulty of measuring the subjective side of this research, indicating that more research needs to be done. In this assessment, the sequence with the minimal distance was ranked highest, then the maximum distance, followed by the random sequence and nearest neighbour heuristic.

Based on the theoretical framework and the implementation of the model, we showed that we are closer to solving the action problem, which is the low streaming time. The proposed solution methods can decrease an exponential amount of decisions and find the (sub)optimal music sequence. Consequently, music playlists can be automatically sequenced, while improving the flow experience and increasing user satisfaction.

6.2 Recommendations

Despite the proposed solution's potential, this research also indicates that some analysis and testing still need to be done. Therefore, we propose the following recommendations for further research:

- **Solve the other core problems:** As mentioned in the discussion, recommendation and classification play a significant role on sequence's results. Both steps are difficult to ignore in the practical assessment. Having more information about the music selection would make the research more compact. Therefore, the other core problems in named in the problem introduction are recommended to be prioritized after this research.
- **Examine Brute Force First**
  As we have seen in the results, brute force only faces capacity limitations until nine songs. While the evolutionary method and nearest neighbour are possible alternatives, the optimal solution can be helpful to conclude subjective questions first. For instance, how many songs are needed to experience flow? How much does each variable contribute to flow? Given the strong preference for the shortest route in the practical validation, brute force could provide more certainty of an improved sequence.
- **Define music selection rules:** For this research, we used a discrete playlist, but the playlist is not always defined. There are even more sequences possible. To tackle this issue, we recommended a study examining the maximum (and perhaps minimum) distance between variables. Which is seen as a song filter for certain complex combinations. If we have a preselection, we are confident that the transitions are always at least within a specific boundary and that outliers can be prevented. For example, EDM users would prefer to have the tempo difference within five beats per minute. Defining this rule would exclude already some possible sequences. Furthermore, it could be the case that the energy level is not wanted to be decreased, as in Figure 10. If that is the case, songs with a high energy level must be filtered out as possible starting songs.
- **Explore more use cases:** Due to the scalability of this solution, this model might also be applicable to other use cases. For example, the optimal sequence can be compared to sequences created by live DJs. Reaching out to DJs may also provide new feedback for input variables. Furthermore, users can be asked when they experience a positive or negative feeling during the mixtape. The setting of the listening situation can be determined, for instance, a student house party and the individual transitions can also be ranked.
- **Combine with stochastic models**: Another idea is to combine this model with non-deterministic models, such as Markov chain applications. For now, we have only dealt with a deterministic playlist, but in practice, we can imagine that the demand for music is more dynamic and probabilistic. When more user data is available, stochastic calculations can make this model more realistic. In other words, once the behaviour of users is known, several routes will be built, predicted and even corrected. Hypothetically, the model will be further developed with AI and machine learning algorithms already possessing this information.

7. References

Adiyansjah, Gunawan, A. A. S., & Suhartono, D. (2019). Music recommender system based on genre using convolutional recurrent neural networks. *Procedia Computer Science*, *157*, 99–109. https://doi.org/10.1016/j.procs.2019.08.146

Afchar, D., A. B. Melchiorre, M. Schedl, R. Hennequin, E. V. Epure, and M. Moussallam. (2022). Explainability in music recommender systems. *AI Magazine 43*: 190–208. https://doi.org/10.1002/aaai.12056

Aucouturier, J.-J., & Pachet, F. (2002). Scaling up music playlist generation. *In Proceedings of ICM*: 105–108. https://doi.org/10.1109/ICME.2002.1035729

Baccigalupo, C., & Plaza, E. (2006). Case-Based Sequential Ordering of Songs for Playlist Recommendation. *Advances in Case-Based Reasoning:* 286-300 https://doi.org/10.1007/11805816_22

Balkema, J. W. (2009). Design and realisation of an efficient content based music playlist generation system. [PhD Thesis - Research external, graduation UT, University of Twente]. Twente University Press (TUP). https://doi.org/10.3990/1.9789036528863

Bittner, R. M., Gu, M., Hernandez, G., Humphrey, E. J., Jehan, T., Mccurry, P. H., & Montecchio, N. (2017). Automatic Playlist Sequencing and Transitions. *In Proceedings of the 18th International Society for Music Information Retrieval Conference* (ISMIR 2017).'

Black, P. E. (2004). Euclidean distance. In P. E. Black (Ed.), *Dictionary of Algorithms and Data Structures* (online). Retrieved from https://www.nist.gov/dads/HTML/euclidndstnc.html

Bonini, T., & Gandini, A. (2019). "First Week Is Editorial, Second Week Is Algorithmic": Platform gatekeepers and the platformization of music curation. *Social Media + Society 5(4)*, 1–11. https://doi.org/10.1177/2056305119880006

Bonnin, G., & Jannach, D. (2014). Automated generation of music playlists: Survey and experiments. In *ACM Computing Surveys 47(2)*: 1-35. Association for Computing Machinery. https://doi.org/10.1145/2652481

Burke, R. (2002) Hybrid recommender systems: Survey and experiments. *User Modeling and UserAdapted Interaction 12*, 4, 331–370. https://doi.org/10.1023/A:1021240730564

Chirico, A., Serino, S., Cipresso, P., Gaggioli, A., & Riva, G. (2015). When music "flows". State and trait in musical performance, composition and listening: A systematic review. In *Frontiers in Psychology* (Vol. 6). Frontiers Media S.A. https://doi.org/10.3389/fpsyg.2015.00906

Cliff, D. (2000). Hang the DJ: Automatic Sequencing and Seamless Mixing of Dance-Music Tracks. *HP Laboratories Technical Report*, 104, 2000 . Retrieved from https://www.researchgate.net/publication/228580488

Codeacademy, (2024), *Normalisation*, Retrieved from https://www.codecademy.com/article/normalization

Csikszentmihalyi, M. (1990). Flow: The psychology of optimal experience. New York, NY: Harper and Row

Dias, R., Gonçalves, D., & Fonseca, M. J. (2017). From manual to assisted playlist creation: a survey. *Multimedia Tools and Applications*, *76*(12), 14375–14403. https://doi.org/10.1007/s11042-016-3836-x

De Groot, J. (2023), What is Data Classification? A Data Classification Definition. Retrieved from: https://www.digitalguardian.com/blog/what-data-classification-data-classification-definition

Fields, B. (2011). Contextualize Your Listening: The Playlist as Recommendation Engine. Ph.D. Dissertation. Department of Computing Goldsmiths, University of London, London, UK.

Fields, B. (2012). A brief history of playlist generation. 5th Recommender Stammtisch. Retrieved from: https://speakerdeck.com/bfields/a-brief-history-of-playlist-generation

Furini, M. (2021). Automatic music playlist generation based on music-programming of FM radios. *2021 IEEE 18th Annual Consumer Communications and Networking Conference, CCNC 2021*. https://doi.org/10.1109/CCNC49032.2021.9369526

Golden, B., Assad, A., & Levy, L., & Gheysens, F. (1984). The Fleet Size and Mix Vehicle Routing Problem

Gusron. R. (2023). Solving Traveling Salesman Problems Using Excel Solver Evolutionary Algorithm. Medium. Retrieved from: https://medium.com/@rihot_gusron/solving-traveling-salesman-problems-using-excel-solver-evolutionary-algorithm-e8deea89ca42

Hartono, P., & Yoshitake, R. (2013). Automatic Playlist Generation from Self-Organizing Music Map. *Journal of Signal Processing*, *17*(1), 11–19. https://doi.org/10.2299/jsp.17.11

Heerkens, H., & van Winden, A. (2017). Solving managerial problems systematically (1st ed.). Groningen, Netherlands: Noordhoff.

Hildén, F. (2020) Smart playlist shuffle using Travelling Salesman. Dev. https://dev.to/felixhilden/smart-playlist-shuffle-using-travelling-salesman-58i1

Hirai, T., Doi, H., & Morishima, S. (2018). Latent topic similarity for music retrieval and its application to a system that supports DJ performance. *Journal of Information Processing*, *26*, 276–284. https://doi.org/10.2197/ipsjjip.26.276

Holtes, G. (2021) Playlist Optimisation as a Traveling Salesperson Problem. Medium. https://medium.com/dataseries/playlist-optimisation-as-a-traveling-salesperson-problem-f31f73c417fa

Jünger, M., Reinelt, G., & Rinami, G. (1995). *The Traveling Salesman Problem* (Vol. 7).

Kody, A. (2021). Using Different Tempos To Make Beats For Different Genres. Izotope. Retrieved from: https://www.izotope.com/en/learn/using-different-tempos-to-make-beats-for-different-genres.html

Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2), 231–247. https://doi.org/10.1016/0377-2217(92)90138-y

Ng, A., & Mehrotra, R. (2020). Investigating the Impact of Audio States & Transitions for Track Sequencing in Music Streaming Sessions. *RecSys 2020 - 14th ACM Conference on Recommender Systems*, 697–702. https://doi.org/10.1145/3383313.3418493

Pohle, T., Pampalk, E., & Widmer, G. (2005). Generating similarity-based playlists using traveling salesman algorithms. In *Proceedings of the 8th International Conference on Digital Audio Effects* (DAFx 2005).

Rasmussen, R. (2011). Travelling Salesman Problem in Spreadsheets - a Guided Tour. *International Review of Economics Education*, 10(1), 94-116. https://doi.org/10.1016/S1477-3880(15)30037-2

Sam, A. "TSP - Travelling Salesman Problem - for 10 Cities - Problem definition and Solution in Excel" Youtube, 8 jun 2022, Retrieved From: https://www.youtube.com/watch?v=51fp2cSlVks

Schedl et al., (2015). On the Influence of User Characteristics on Music Recommendation Algorithms. https://doi.org/10.1007/978-3-319-16354-3_37

Slack, N., Brandon-Jones, A., & Johnston, R. (2016). Operations Management (8th ed.). Pearson.

Van der Heijden, M.C. & Van der Wegen, L.L.M. (2017). *Vehicle Routing – Lecture Notes*

Winston, W. L. (2004). Operations Research: Applications and Algorithms (4th ed.). Brooks/Cole.

Won, M., Spijkervet, J., & Choi, K. (2021). Music Classification: Beyond Supervised Learning, Towards Real-world Applications. https://doi.org/10.48550/arXiv.2111.11636

Young, C. (2024). Excel solver: Which solving method should I choose? Engineer Excel. Retrieved from: https://engineerexcel.com/excel-solver-solving-method-choose

Appendix A: Model Excel & Dataset

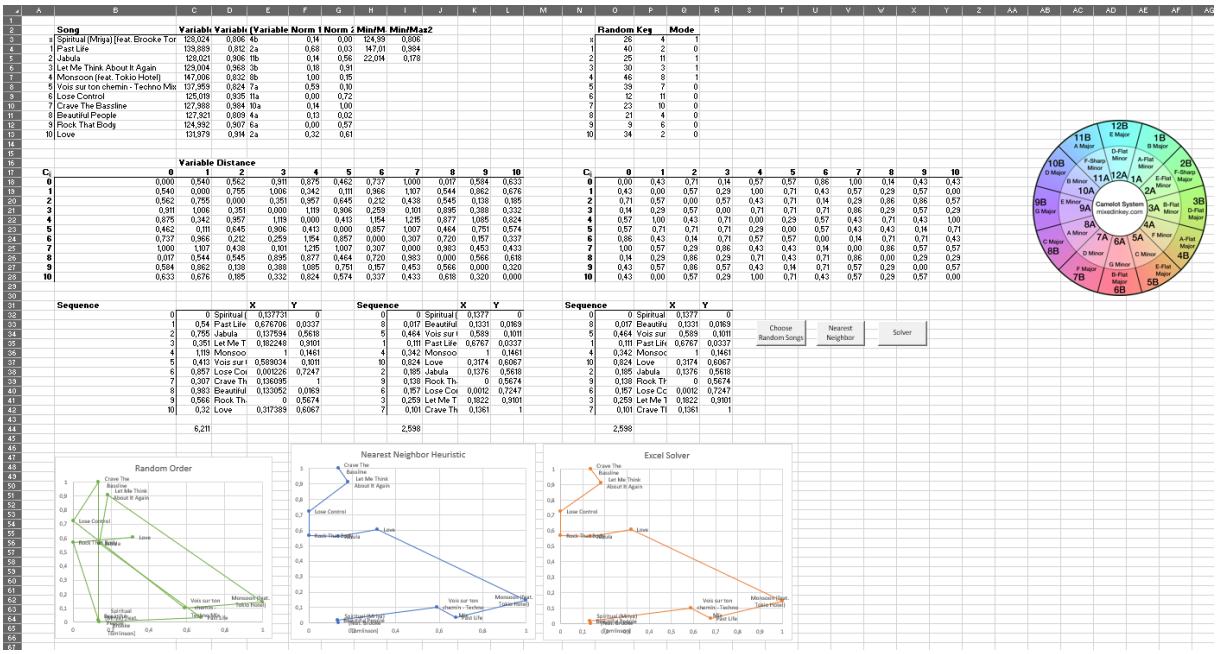

Figure 23: Model Excel

Appendix B: VBA Code Nearest Neighbour

```
Sub NearestNeighbour()

    Dim StartX As Double
    Dim StartY As Double
    Dim NearestX As Double
    Dim NearestY As Double
    Dim MinDistance As Double
    Dim Distance As Double
    Dim NearestIndex As Long
    Dim TotalRows As Long
    Dim i As Long
    Dim j As Long
    Dim Visited() As Boolean

    TotalRows = 12 'Total songs -2


    StartX = Range("F3").Value 'Input coordinates
    StartY = Range("G3").Value

    ReDim Visited(1 To TotalRows)

    For i = 2 To TotalRows

    MinDistance = 100000 'Random large number

    For j = 2 To TotalRows
        If Not Visited(j - 1) Then 'Checks if city is visited'
        Distance = Sqr((Cells(j + 1, 6) - StartX) ^ 2 + (Cells(j + 1, 7) - StartY) ^ 2) 'Calculate distance

    If Distance < MinDistance Then 'Checks if distance is smaller than minimal distance
        MinDistance = Distance  'If smaller, update new minimal distance'
        NearestX = Cells(j + 1, 6) 'Update coorninates'
        NearestY = Cells(j + 1, 7)
        NearestIndex = j 'Continues until all unvisited cities have been compared
    End If
    End If
    Next j

    Visited(NearestIndex - 1) = True  'Changes smallest song to visited

    StartX = NearestX 'Changes start value to last visited song
    StartY = NearestY

    Cells(i + 30, 11).Value = NearestX  'Fills in the closest coordinates
    Cells(i + 30, 12).Value = NearestY
    Cells(i + 30, 8).Value = NearestIndex - 2

Next i

End Sub
```

Figure 24: VBA Code Nearest Neighbour

Appendix C: Settings Excel Solver

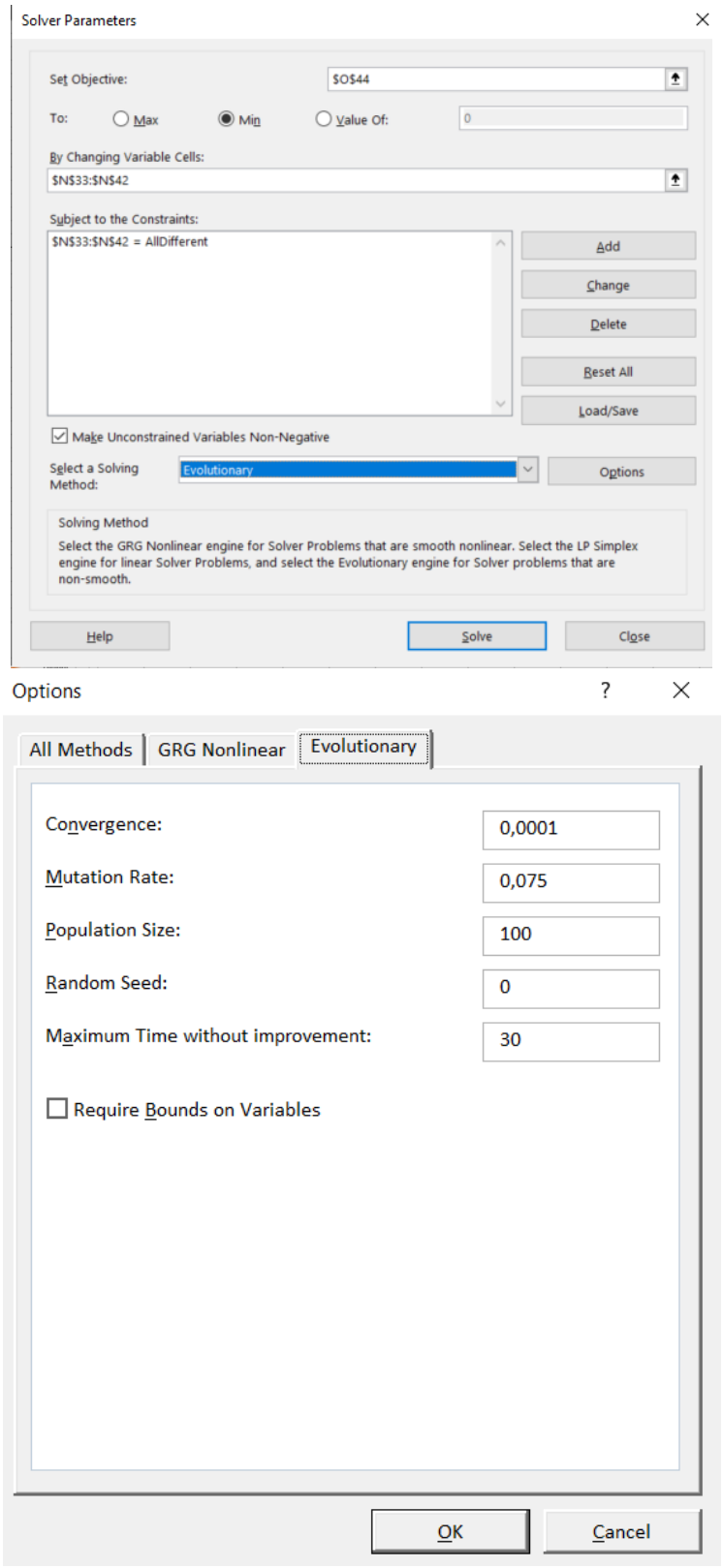

Figure 25: Solver Parameters

Appendix D: Key Data Translation

Mode	Key	Pitch	Mixed in Key
0	11	B	10a
1	2	D	10b
0	6	F#	11a
1	9	A	11b
0	1	C#	12a
1	4	E	12b
0	8	G#	1a
1	11	B	1b
0	3	D#	2a
1	6	F#	2b
0	10	A#	3a
1	1	C#	3b
0	5	F	4a
1	8	G#	4b
0	0	C	5a
1	3	D#	5b
0	7	G	6a
1	10	A#	6b
0	2	D	7a
1	5	F	7b
0	9	A	8a
1	0	C	8b
0	4	E	9a
1	7	G	9b

Figure 26: Key Data Translation

Appendix E: Playlists Sensitivity Analysis

EDM Song	Danceability	Energy	Tempo
Sunglasses at night	0,617	0,883	140,012
Navigator	0,742	0,931	128,012
BONZAI	0,707	0,926	134,973
Beautiful People	0,415	0,809	127,921
Love	0,579	0,914	131,979
Rock That Body	0,715	0,907	124,992
Ray Of Solar	0,355	0,802	135,017
Crave The Bassline	0,786	0,984	127,988
Metro	0,633	0,961	125,033

Table 13: EDM Playlist

Hip-Hop Song	Danceability	Energy	Tempo
Personality	0,74	0,667	93,001
Come Ova	0,795	0,547	116,748
Best Wel Chill	0,808	0,72	100,012
Stiekem '24	0,827	0,634	104,003
Keeper	0,792	0,616	93,931
Wahala (feat. Olamide)	0,852	0,566	96,937
Lie to You	0,835	0,646	102,988
Tokkoh	0,823	0,821	100,034
Ver	0,737	0,709	102,033

Table 14: Hip-Hop Playlist

Rock Song	Danceability	Energy	Tempo
SPINE	0,414	0,934	149,98
*that b*tch don't even kno my name...*	0,538	0,647	158,092
Everything and Nothing	0,586	0,928	132,976
Sick of Being Young	0,52	0,899	150,107
Take A Bite	0,537	0,631	91,007
Perfume	0,505	0,891	93,003
Edge of the Earth	0,494	0,751	160,426
MORE	0,58	0,666	92,01
lightweight	0,492	0,945	154,978

Table 15: Rock Playlist

Appendix F: Practical Validation

Person	1	2	3	4
1	1	4	3	2
2	4	2	1	3
3	1	3	2	4
4	4	2	3	1
5	2	4	1	3
6	3	2	1	4
7	4	3	1	2
8	4	3	1	2
9	4	2	1	3
10	3	4	1	2
11	3	4	2	1
12	2	4	1	3
13	3	1	4	2
14	4	2	3	1
15	4	2	3	1
16	3	2	4	1
17	2	3	1	4
18	4	1	3	2
19	2	3	1	4
20	1	4	3	2
21	3	4	2	1
22	4	3	1	2
23	4	1	3	2

Table 16: Individual results Ranking test

Amount	1	2	3	4
NN (1)	3	3	11	6
Random (2)	4	7	3	9
Max (3)	6	6	7	4
Min (4)	10	7	2	4

Table 17: Results Validation Test

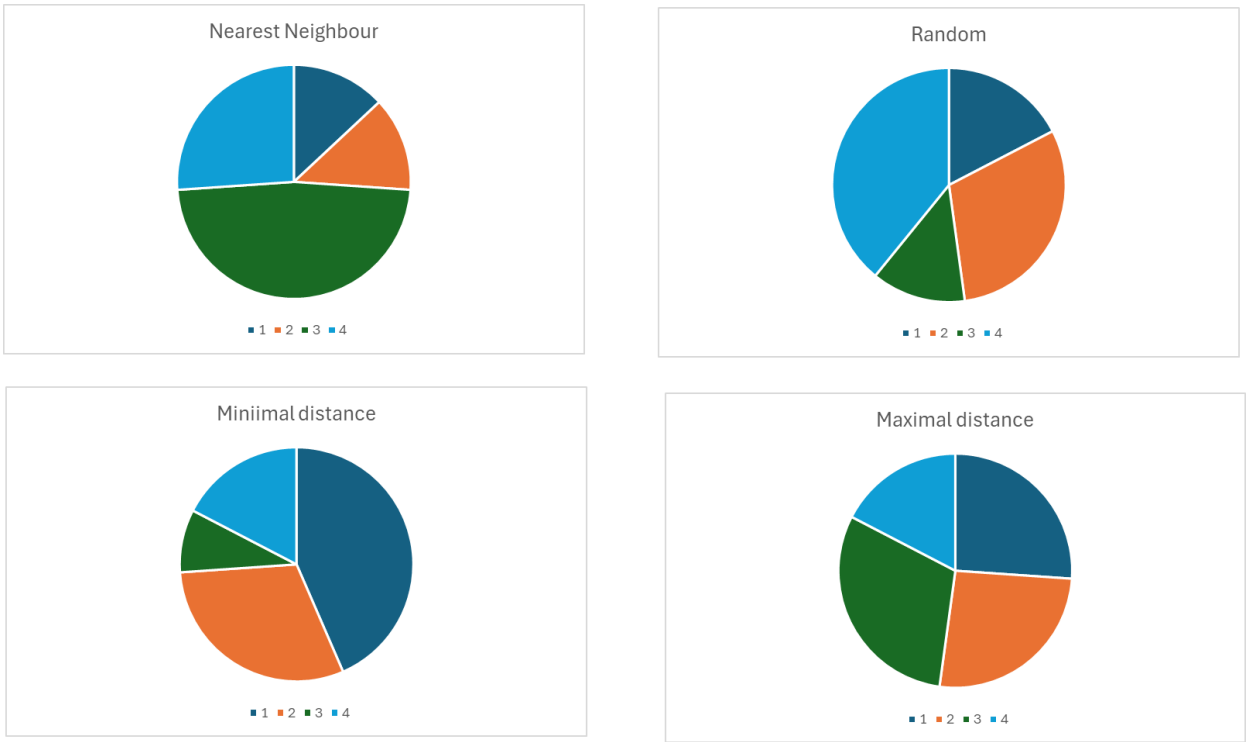

Figure 27: Chart Pie Results Validation Test