Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Composition and reconstruction of music with a help of an evolutionary algorithm

Bachelor's Thesis

2026
Oleksandr Piskovyi

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Composition and reconstruction of music with a help of an evolutionary algorithm

Bachelor's Thesis

Study Programme:	Applied Computer Science
Field of Study:	Computer Science
Department:	Department of Applied Informatics
Supervisor:	doc. RNDr. Mária Markošová, PhD.

Bratislava, 2026
Oleksandr Piskovyi

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Oleksandr Piskovyi
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Skladanie a rekonštrukcia hudby pomocou evolučných algoritmov
Composition and reconstruction of music with a help of evolutionary algorithm

Anotácia: Študent, ktorý sa pokúsi najprv zrekonštruovať a potom vytvoriť krátku skladbu pomocou evolučných algoritmov musí mať znalosti v niektorej hudobnej oblasti, aby si dokázal zvoliť aký typ hudby bude skladať. Môže použiť napr. genetický algoritmus a správne zadefinovať ohodnocovaciu funkciu. Očakáva sa, že študent algoritmus naprogramuje.

Cieľ: Cieľom je vytvoriť krátku skladbu pomocou niektorého typu evolučného algoritmu, poprípade zrekonštruovat úsek niektorej skladby.

Literatúra: Barz-Beielstein, Menhen, Branke, Evolutionary algorithmms, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery · May 2014, DOI:10.1002/widm.1124

Kľúčové slová: evolučné algoritmy, fitnes, účelová funkcia, chromozóm

Vedúci: doc. RNDr. Mária Markošová, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 05.03.2025

Dátum schválenia: 29.09.2025 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Oleksandr Piskovyi
Study programme:	Applied Computer Science (Single degree study, bachelor I. deg., full time form)
Field of Study:	Computer Science
Type of Thesis:	Bachelor´s thesis
Language of Thesis:	Slovak
Secondary language:	English

Title: Skladanie a rekonštrukcia hudby pomocou evolučných algoritmov
Composition and reconstruction of music with a help of evolutionary algorithm

Annotation: Študent, ktorý sa pokúsi najprv zrekonštruovať a potom vytvoriť krátku skladbu pomocou evolučných algoritmov musí mať znalosti v niektorej hudobnej oblasti, aby si dokázal zvoliť aký typ hudby bude skladať. Môže použiť napr. genetický algoritmus a správne zadefinovať ohodnocovaciu funkciu. Očakáva sa, že študent algoritmus naprogramuje.

Aim: Cieľom je vytvoriť krátku skladbu pomocou niektorého typu evolučného algoritmu, poprípade zrekonštruovat úsek niektorej skladby.

Literature: Barz-Beielstein, Menhen, Branke, Evolutionary algorithmms, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery · May 2014, DOI:10.1002/widm.1124

Keywords: evolučné algoritmy, fitnes, účelová funkcia, chromozóm

Supervisor:	doc. RNDr. Mária Markošová, PhD.
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	doc. RNDr. Tatiana Jajcayová, PhD.
Assigned:	05.03.2025
Approved:	29.09.2025

doc. RNDr. Damas Gruska, PhD.
Guarantor of Study Programme

.....
Student

.....
Supervisor

Acknowledgments: I would like to acknowledge the open-source community for providing tools and resources that made this work possible. I also thank the band The Hellp for creating music that inspired part of this work.

Abstrakt

Táto práca sa zaoberá kompozíciou a rekonštrukciou krátkych hudobných fragmentov pomocou evolučného algoritmu. Cieľom práce bolo navrhnúť a implementovať systém, ktorý generuje kandidátske hudobné ukážky, modifikuje ich prostredníctvom evolučných operátorov a vyberá lepších kandidátov na základe zvolenej fitnes funkcie. Práca produkuje krátke hudobné výstupy v dvoch typoch úloh: rekonštrukcia známeho hudobného fragmentu a generovanie hudby bez jednej pevne stanovenej cieľovej ukážky.

Hlavným zameraním práce je štúdium vplyvu rôznych fitnes funkcií na tento proces. Experimenty porovnávajú niekoľko spôsobov hodnotenia kandidátskych ukážok: podobnosť s cieľovou nahrávkou v audio reprezentácii, hodnotenie lokálnych zvukových vlastností, symbolické štatistické hodnotenie hudobnej štruktúry a hodnotenie na základe textového popisu. Výsledky ukazujú, že výber fitnes funkcie silne ovplyvňuje rýchlosť prehľadávania, výpočtovú náročnosť a charakter vytvorených výstupov. Niektoré metódy hodnotenia sú praktickejšie pre rekonštrukciu cieľovej ukážky, zatiaľ čo iné sú vhodnejšie pre samotné generovanie.

Kľúčové slová: evolučné algoritmy, fitnes, účelová funkcia, chromozóm

Abstract

This thesis deals with the composition and reconstruction of short musical fragments using an evolutionary algorithm. The goal of the work was to design and implement a system that generates candidate musical pieces, modifies them through evolutionary operators, and selects better candidates according to a chosen fitness function. The work produces short musical outputs in two types of tasks: reconstruction of a known musical fragment and generation of music without one fixed target piece.

The main focus of the thesis is to study the influence of different fitness functions on the process. The experiments compare several ways of evaluating candidate pieces: similarity to a target recording in an audio representation, evaluation of local audio properties, symbolic statistical evaluation of musical structure, and evaluation based on a textual description. The results show that the choice of fitness function strongly affects search speed, computational cost, and the character of the produced outputs. Some evaluation methods are more practical for reconstruction of a target piece, while others are more suitable for generation.

Keywords: evolutionary algorithms, fitness, objective function, chromosome

Contents

List of Figures

Introduction

Music composition is usually understood as a creative human activity, but it can also be approached computationally. In algorithmic composition, a computer system generates musical material according to rules, models, or search procedures. Evolutionary algorithms are one possible approach: instead of writing the final piece directly, the system generates many candidate fragments, evaluates them, and gradually modifies the population.

The topic of this thesis is the composition and reconstruction of music with the help of an evolutionary algorithm. This task is nontrivial because music does not have one simple objective measure. Even a short fragment can be judged by melody, rhythm, repetition, timbre, similarity to a target, or subjective listener preference. Therefore, the central practical question is not only how to generate candidates, but also how to evaluate them.

This thesis implements a system for producing short musical fragments. The work considers two related tasks: reconstruction of a known target recording and generation of a new fragment without a fixed target. The main experimental focus is the comparison of several fitness functions used to guide the evolutionary process.

Chapter 1

Evolutionary Algorithms

This chapter is based mainly on the overview of evolutionary algorithms by Bartz-Beielstein et al. [1].

1.1 Basic Principle

Evolutionary algorithms are search and optimization methods inspired by Darwinian evolution. Instead of just taking one single solution and tweaking it, these algorithms work with a group of possible solutions known as a population.

The algorithm does not "understand" the problem in the way a human does. Instead, each candidate solution to a problem is assigned a fitness value — a numerical score that indicates how well one solves the problem. Optimal solutions are then approximated through a continuous cycle of variation and selection.

The practical behavior of the algorithm depends heavily on three key factors: the search operators (how candidates are changed), the control flow, and the representation (how the problem is encoded).

Here is how the standard evolutionary cycle works:

1. **Initialize the Population**. The algorithm generates an initial set of solution candidates, which form the starting population.

2. **Evaluate Fitness**. A function assigns a fitness value to every individual, a measurement of how well it solves the given problem.

3. **Select Parents**. To generate the next wave of solutions, the algorithm selects parent candidates from the population. This selection is biased toward individuals with better fitness values.

4. **Apply Variation**. New candidates are created using search operators. Recombination (or crossover) rearranges existing information from the parents, while

mutation introduces random changes to explore completely new areas of the search space.

5. **Selection Stage**. Finally, the algorithm must decide which solutions survive to form the next generation. This cycle continues repeatedly until specific termination criteria are fulfilled.

In this thesis, the term "evolutionary search" refers to this repeated cycle of initialization, evaluation, variation, and selection.

1.2 Main Concepts

An individual is one single point in the search space, representing one candidate solution. In this thesis, one individual corresponds to one candidate musical fragment.

The "genotype" is the internal code or string that the algorithm actually modifies. The "phenotype" is the realized, expressed form of that individual. For example, in a simple melody-generation task, the genotype could be a list of numbers encoding pitches and durations, while the phenotype is the actual melody you hear — an audio wave. This distinction matters because often implementations use a mapping process to turn the genotype into the phenotype before evaluating it.

The "fitness function" assigns the numerical score to an individual, to be then used in the selection stage. In this thesis, the software component that computes this score is called an "evaluator". A reconstruction evaluator may reward similarity to a known target audio, while a generation evaluator may reward agreement with a textual description, or alignment with well-known musical principles.

Selection directs the evolutionary search by favoring high-quality candidates. However, the variation operators—mutation and recombination—are what actually produce the diversity needed to find new solutions. Recombination mixes material that is already present in selected parents, while mutation introduces random changes that may create candidates not reachable by recombination alone.

1.3 Genetic Algorithms and Genetic Programming

While "evolutionary algorithms" is an umbrella term, there are several variants.

Genetic Algorithms (GAs) traditionally focus on fixed-length representations, such as binary strings, which were heavily inspired by the biological genetic code.

Genetic Programming (GP), on the other hand, is used for the automatic generation of computer programs or structured representations. A key advantage of GP is that it does not use fixed-length representations; instead, solutions are represented as branching trees of varying depths.

The implementation in this thesis is closer to genetic programming. Musical candidates are represented as trees rather than flat, fixed-length strings. Some nodes describe musical material, while others describe transformations.

1.4 Example in Music

Consider a very small melody consisting of four notes. A simple genotype could be:

```
[(60, 1), (62, 1), (64, 1), (67, 1)]
```

Here each pair contains a MIDI pitch and a duration in beats.

MIDI uses integer note numbers to represent pitch; for example, note 60 corresponds to middle C, and increasing the number by one raises the pitch by one semitone [2].

The corresponding phenotype is the melody heard when these notes are played. A mutation operator might change the third pitch from 64 to 65 to explore a new sound. A recombination operator might take the first two notes from one parent and the last two notes from another parent and swap them.

The fitness function then decides whether the new melody is better. The algorithm does not compose by following explicit musical rules step by step; the solution emerges purely from the variation-selection cycle. A more detailed discussion of concrete systems for evolutionary music composition is given in chapter 3.

Chapter 2

Goals of the Thesis

The assigned topic of this thesis is the composition and reconstruction of music with the help of an evolutionary algorithm. The practical goal is therefore to produce short musical artifacts with the help of evolutionary search, as introduced in chapter 1. We approach this goal through the design and comparison of fitness functions. The central question is how different fitness implementations affect the musical artifacts that are produced.

We do not try to prove that one fitness function is universally best for music. Instead, we study how different evaluators influence the search process and the resulting musical artifacts.

The produced artifacts are studied in two settings:

- reconstruction of a known target recording,
- generation of a new musical fragment without one exact target piece.

2.1 Reconstruction and Generation

The reconstruction setting has a concrete target. A candidate is considered better if it becomes more similar to a selected reference recording. This makes the task easier to formulate as an optimization problem, but it also raises the question of how similarity should be measured. Two candidates may be similar in timbre but not in melody, or similar in rhythm but not in overall structure.

The generation setting has no single correct output. A candidate is evaluated according to more general criteria, such as statistical musical structure or agreement with a text description. This setting is closer to open-ended composition, but the interpretation of fitness is less direct. For example, a candidate may match the selected statistical or semantic criterion while still lacking properties that a listener would consider musically important.

The distinction between these two settings is important. It makes clear that reconstruction and generation require different kinds of evaluators and that their fitness values cannot be compared as if they measured the same quantity.

Chapter 3

Related Work and Background

This chapter reviews previous approaches that are related to the goal of this thesis: creating or reconstructing short musical artifacts with the help of an evolutionary algorithm. The emphasis is not only on what these works implement, but also on what problem they encounter when evaluating music and how they try to solve it. This is important because the main experimental focus of this thesis is the design and comparison of fitness functions.

3.1 Evolutionary Approaches to Music Composition

Algorithmic composition is a broad area that includes rule-based, stochastic, evolutionary, statistical, and learning-based methods for producing musical material. Nierhaus gives an overview of this broader field and includes genetic algorithms among the paradigms used for automated music generation [3]. Methods based on evolutionary algorithms are relevant here because they do not need to construct the final piece in one deterministic step. They can instead work with many candidate musical fragments, evaluate them, modify them, and gradually keep material that scores better.

An early example of evolutionary music generation is GenJam by Biles, which uses a genetic algorithm to generate jazz solos [4]. The goal of the system is a creative generation rather than reconstruction of a fixed target. Its design choice is the use of human feedback: individuals are judged interactively, and this feedback becomes the basis for fitness. This addresses the main difficulty of music generation directly, because human listeners can judge musical quality more flexibly than a simple mathematical formula. The limitation is that interactive evaluation is slow, subjective, and hard to repeat in large experiments.

Marques et al. also apply genetic algorithms to music composition [5]. Their work is relevant because it shows the more automatic alternative: musical candidates can be generated and evaluated computationally, without requiring a human listener at

every generation. The difficulty is then moved into the design of the fitness function. If the automatic score rewards only a narrow property of the music, the algorithm may improve that score without producing a convincing musical result.

3.2 The Fitness Function Problem

The works above illustrate the central problem of evolutionary music systems: the evolutionary algorithm can optimize only the score it is given. If the score comes from a human listener, it may reflect musical judgment well, but it is expensive and difficult to reproduce. If the score is computed automatically, it is faster and repeatable, but it may capture only part of musical quality.

This problem is directly related to our work. Evolutionary algorithms can explore a large number of candidates, but a human listener in the middle of every evaluation would make such search very slow. Human ratings would also introduce subjective variation that is difficult to control in repeated runs. For this reason, the implemented system uses automatic evaluators. This does not mean that automatic fitness is assumed to be a complete model of musical quality. Instead, the experiments compare several evaluator types and analyze what each one encourages or fails to encourage.

3.3 Statistical and Corpus-Based Evaluation

One way to avoid direct human evaluation is to compare a candidate with statistical regularities observed in a reference collection of music. Manaris et al. present a corpus-based hybrid approach to music analysis and composition [6]. In their work, the word corpus means a collection of musical pieces used as reference material. Their system extracts statistical features from this reference material, including features related to Zipf's law (the observation that in many natural collections, the frequency of an item is inversely proportional to its rank), and trains artificial music critics. A candidate can then be scored according to how well its statistics fit the learned reference behavior.

The goal of this approach is to obtain an automatic critic that is still grounded in existing music. This is a useful compromise: the evaluator is repeatable and computationally usable, but it is not merely an arbitrary hand-written rule. The limitation is that closeness to reference statistics does not guarantee musical form or listener preference. A generated fragment can have a plausible distribution of pitches or intervals and still sound repetitive, structurally weak.

This work motivates the symbolic statistical evaluator in the present thesis. The implemented version is not a direct reproduction of Manaris et al.; it uses synthetic Zipf-like and non-Zipf-like distributions rather than a real music reference collection

of music. It is therefore treated as inspired by that corpus-based line of work, whose purpose is to test how a computationally efficient symbolic statistical fitness function behaves in comparison with other evaluators.

3.4 Similarity-Based Reconstruction

Reconstruction differs from open-ended generation because there is a reference artifact. The problem is not only to produce music that sounds plausible, but to produce music that resembles a known target. This shifts the evaluation problem from musical quality in general to musical similarity. The difficulty is that similarity can be defined in several ways: two candidates may be similar in pitch content, rhythm, timbre, texture, or learned audio representation, while still differing in other musically important aspects.

Fréchet Audio Distance was proposed by Kilgour et al. as a metric for evaluating music enhancement algorithms [7]. The method compares distributions of audio embeddings rather than raw audio samples. An embedding is a numerical vector that represents an object, such as a piece of audio, in a space where distances or similarities can be computed. This is relevant because it shows a way to compare musical audio in a representation that is less tied to sample-by-sample waveform differences. The limitation for this thesis is that the original metric is intended for comparing sets of audio examples. When the idea is adapted to a single target piece, the interpretation must be carefuller.

The reconstruction experiments in this thesis follow the broader idea of evaluating similarity in an audio representation. One evaluator compares a whole candidate to a target recording, while another compares local audio chunks. Both approaches are useful because they provide automatic reconstruction objectives, but both also inherit the usual risk of similarity-based fitness: improving the numerical similarity does not necessarily mean that the candidate reconstructs all musically meaningful aspects of the target.

3.5 Learned and Semantic Fitness Functions

Recent work also explores the use of learned models as fitness functions. This direction is relevant when the desired result is hard to describe with simple rules. Ostermann et al. use large language models as fitness functions in evolutionary algorithms for music generation [8]. Their work is important for this thesis because it treats a pretrained model not only as a generator, but as an evaluator that can guide the search.

The advantage of this approach is that a learned model can represent broader semantic information than a simple symbolic statistic. A candidate can be judged with

respect to a description, style, or high-level musical intention. The limitation is that such fitness functions are less transparent. It may be hard to know exactly why one candidate receives a higher score than another, and the score may still fail to enforce musical structure.

The text-conditioned experiment in this thesis belongs to this family of ideas. It uses a pretrained music-text representation as one possible evaluator inside the same implementation of the evolutionary algorithm as the other experiments.

3.6 Position of This Thesis

The related work shows several ways to address the evaluation problem in evolutionary music: human feedback, automatic rules, statistical critics trained from reference music, similarity measures, and learned semantic evaluators. Each option solves part of the problem and introduces a different limitation. Human feedback is musically meaningful but hard to scale. Statistical fitness is computationally efficient and repeatable but partial. Similarity-based reconstruction has a clear target but depends on the chosen representation of similarity. Learned evaluators allow more abstract goals, but their scores can be opaque.

This thesis builds on these ideas in order to produce short musical artifacts with an evolutionary algorithm. The work is practical in that it implements a system for generation and reconstruction, and experimental in that it compares several evaluator types within the same implementation.

Chapter 4

Design and Implementation

The goals in chapter 2 require an implementation that can produce short musical artifacts, support both reconstruction and generation, and allow several fitness functions to be compared. This chapter describes how the implemented system satisfies these requirements.

The implementation was designed with four main properties in mind. First, it must represent musical fragments in a form that can be modified by mutation and recombination. Second, the representation should be expressive enough to describe simple musical structure, such as repetition, transposition, phrase sequencing, and multiple voices. Third, the same individual representation should be usable with different fitness functions. Some fitness functions work with symbolic music, others need MIDI files, and others need rendered audio. Fourth, the experiments must be reproducible enough to compare runs with different random seeds and generation counts.

4.1 System Overview

The implemented system follows one main flow:

```
individual tree -> SymbolicScore -> MIDI/audio -> fitness value
```

The individual tree is the genotype modified by the evolutionary algorithm. It does not contain audio samples directly. Instead, it describes musical operations such as creating a motif, adding a rest, joining phrases, repeating, or transposing material, stretching it in time, assigning an instrument, and combining voices in parallel.

Before evaluation, each individual tree is compiled into a `SymbolicScore`. This is the common representation used by all experiments. It contains tempo, total duration, voices with instrument assignments, and note events with pitch, onset, duration, and velocity. From this form the system can either export MIDI, render audio, or compute symbolic statistics directly.

The compiled individual is then scored by the selected fitness function. The result is a scalar fitness value, optionally accompanied by diagnostic metrics. The search procedure itself only needs the scalar fitness value. This makes it possible to test different fitness functions while keeping the same basic individual representation and evolutionary search procedure.

4.2 Candidate Representation

Each individual in the population is represented internally as a tree. The implementation distinguishes three node levels: phrase nodes, voice nodes, and score nodes. Phrase nodes describe material inside one musical line. Voice nodes assign an instrument to a phrase. Score nodes combine voices or scores into a parallel structure. This type distinction is also used during variation: a phrase subtree can be replaced only by another phrase subtree, a voice subtree by another voice subtree, and a score subtree by another score subtree.

The basic phrase nodes are motifs and rests. A motif stores a sequence of MIDI pitches, a sequence of durations in beats, and a velocity value (MIDI velocity controls the loudness or attack strength of a note). For example, a motif with pitches 60, 62, and 67 and durations 1, 1, and 2 describes three notes: middle C for one beat, D for one beat, and G for two beats. A rest node represents silence with a specified duration.

Other phrase nodes build larger structures from smaller ones. A sequence node places one phrase after another. A repeat node repeats a phrase a fixed number of times. A stretch node changes the duration and timing of a phrase by a fixed factor. A transpose node shifts all pitches by a number of semitones. These nodes allow the algorithm to modify musical material above the level of individual notes.

At the voice level, an assignment node connects a phrase with an instrument. The implemented instruments are sine, pluck, pad, bass, and percussion. At the score level, voices can be combined in parallel. A simple individual can therefore describe, for example, a pluck phrase playing together with a slower bass phrase. This is the main reason for using a tree representation rather than a flat note list: repeated and transformed musical material remains explicit in the individual structure.

Random initialization follows the same representation. Motifs are sampled from a bounded MIDI pitch range, a small set of note durations, fixed velocity values, and motif lengths between one and eight notes. Repeat counts, stretch factors, transposition intervals, and instruments are also sampled from bounded sets. These bounds define the practical search space explored by the experiments.

4.3 Compilation to Symbolic Score

The tree representation is useful for search, but later components need a uniform musical form. Therefore, every valid individual is compiled into a `SymbolicScore`. A symbolic score contains a tempo, total duration, and a list of voices. Each voice contains its instrument and an ordered list of note events. A note event stores pitch, onset in beats, duration in beats, and velocity.

Compilation is recursive. A motif becomes note events with accumulated onsets. A rest contributes duration but no notes. A sequence shifts the second phrase after the first one. A repeat copies the child phrase with increasing offsets. A stretch changes both onsets and durations. A transpose changes pitches while leaving timing unchanged. Voice compilation applies the selected instrument, and score compilation combines voices in parallel.

The important point is that all fitness functions receive the same canonical form after compilation. A symbolic fitness function can inspect note events directly. A text-conditioned fitness function can export the score to MIDI. An audio fitness function can render it to a WAV file. This design connects the practical goal of producing music with the experimental goal of comparing different fitness functions.

4.4 Output Formats

The system supports two output formats derived from `SymbolicScore`. MIDI export creates a standard MIDI file containing tempo information, tracks, note messages, and instrument mappings. MIDI is used for symbolic artifacts and for fitness functions that operate on symbolic music files.

Audio-based evaluators do not operate on the tree directly. After compilation to `SymbolicScore`, the score is passed through an audio-synthesis pipeline that renders it as a WAV file. In the implementation, this pipeline uses Supriya and SuperCollider [9, 10]; installation details are given in chapter A.

Because both MIDI and audio are derived from the same symbolic score, the experiments do not use unrelated internal formats. The individual is always created and modified as a tree, compiled into a symbolic score, and only then converted into the format required by the chosen fitness function.

4.5 Evolutionary Search Procedure

The search procedure uses a generational evolutionary algorithm. It is configured by a random seed, population size, number of generations, tournament size, elitism count,

crossover rate, mutation rate, maximum tree depth, maximum node count, maximum duration, and target tempo.

At the start, the algorithm samples random valid individual trees. In each generation, individuals are compiled into symbolic scores and evaluated by the selected fitness function. The best individuals can be copied directly into the next generation according to the elitism setting. The remaining population is filled with offspring produced from selected parents.

Parent selection is implemented as tournament selection. A small group of individuals is sampled from the population, and the individual with the highest fitness is selected as a parent. This gives better individuals a higher chance of being used while still preserving some diversity.

Crossover and mutation are adapted to the tree representation. Crossover exchanges compatible subtrees between two parents. Mutation either replaces a subtree with a newly generated subtree of the same kind or changes a local parameter, such as a pitch, duration, repeat count, stretch factor, transposition interval, or instrument. After variation, the resulting individual must still satisfy the depth, node-count, and duration constraints.

4.6 Fitness Function Interface

To compare several fitness functions, the search procedure uses a common interface. In the code, each fitness function is represented by an evaluator. The evaluator receives a `SymbolicScore` and returns an `EvaluationResult`. This result contains one scalar fitness score and optional diagnostic metrics.

The scalar score is used for selection and ranking. The diagnostic metrics are stored for later analysis. For example, an evaluator may return not only the final fitness value, but also similarity, penalty, event count, or other quantities useful for interpreting the run.

4.7 Constraints and Reproducibility

The implementation uses hard constraints on tree depth, node count, and duration. These constraints are necessary because the representation has variable size. A repeat node can multiply duration, a sequence can extend a phrase, and parallel score nodes can increase the number of voices. Without limits, the algorithm could spend time on individuals that are too large or too slow to evaluate.

This also addresses a standard issue in genetic programming: variable-size tree representations can grow without a corresponding improvement in fitness, a phenomenon

known as bloat [11]. In this system, uncontrolled growth would increase compilation cost, audio rendering time, and neural-model inference cost. The implementation therefore uses hard depth, node-count, and duration limits as boundaries for valid individuals. A softer alternative would be parsimonious fitness, where larger individuals receive a fitness penalty, but hard limits were chosen here to keep every evaluated individual within predictable computational bounds.

Reproducibility is supported through explicit random seeds and saved run summaries. The search procedure uses a seeded random number generator for initialization, selection, crossover, and mutation. Experiment scripts also seed other libraries where relevant. Each run can write a summary containing the configuration, best fitness, diagnostic metrics, generation summaries, output directory, and produced artifact paths.

Chapter 5

Fitness Functions and Evaluators

The previous chapter described the common implementation used by the experiments. This chapter describes the fitness functions used inside that implementation. The purpose is to make clear what each evaluator rewards, what assumption it makes about music, and whether it is intended mainly for reconstruction or for generation.

5.1 Fitness Function and Evaluator

In this thesis, a *fitness function* is the scoring rule optimized by the evolutionary algorithm. It maps an individual, after compilation to a `SymbolicScore` and possibly to MIDI or audio, to one numerical value. Higher fitness values are always treated as better by the search procedure. An *evaluator* is the implementation of such a scoring rule in the program. Each evaluator returns an `EvaluationResult`, which contains the scalar fitness value used for selection and optional diagnostic metrics used later for analysis.

Two pretrained neural-network model families are used for this purpose. CLAP (Contrastive Language-Audio Pretraining) models compare audio and text by mapping them into embeddings [12, 13]. CLaMP (Contrastive Language-Music Pretraining) models are similar in spirit, but are designed for music information retrieval across modalities such as symbolic music and text [14, 15].

5.2 Global CLAP Audio Similarity

The first reconstruction evaluator uses a CLAP model to convert the whole rendered individual and the target recording into audio embeddings. CLAP models learn a shared representation for audio and language, but they also provide audio embeddings that can be compared directly [12, 13]. In this evaluator, both the target recording and the rendered individual are represented by one CLAP audio embedding.

The target embedding is computed once at the beginning of the run. During evaluation, the individual is rendered to an audio file and passed through the CLAP. The fitness value is the cosine similarity between the target embedding and the individual embedding:

$$f(x) = \frac{e(x) \cdot e(t)}{\|e(x)\|\|e(t)\|}, \tag{5.1}$$

where x is the individual, t is the target recording, and $e(\cdot)$ denotes the CLAP audio embedding. A higher value means that the model represents the two audio recordings as more similar.

This evaluator is simple and directly connected to the reconstruction goal. It does not require a symbolic transcription of the target; the target can be any audio recording. Its main weakness is that a single global embedding compresses the whole recording into one vector. Therefore, two pieces may obtain a high similarity score even if they differ in local melody, rhythm, or structure. The score should therefore be interpreted as audio-representation similarity, not as a complete measurement of musical reconstruction quality.

5.3 Chunk-Wise CLAP Distance Inspired by FAD

The second reconstruction evaluator is inspired by Fréchet Audio Distance (FAD), which compares distributions of audio embeddings rather than individual audio samples [7]. The original FAD metric is designed for comparing sets of audio examples. We adapt the idea to a single target by splitting both the target and the rendered individual into fixed-length audio chunks.

Each chunk is passed through a CLAP. This gives a set of embeddings for the target and a set of embeddings for the individual. For each set, the evaluator computes a mean vector and covariance matrix. The distance is then computed in the same general form as a Fréchet distance between two Gaussian distributions:

$$D = \|\mu_t - \mu_x\|^2 + \mathrm{Tr}\left(\Sigma_t + \Sigma_x - 2(\Sigma_t\Sigma_x)^{1/2}\right), \tag{5.2}$$

where μ_t, Σ_t describe the target chunks and μ_x, Σ_x describe the individual chunks. Because the evolutionary algorithm maximizes fitness, the evaluator returns the negative objective:

$$f(x) = -(D + p_{\text{length}}). \tag{5.3}$$

The length penalty discourages individuals that are much shorter or longer than the target recording.

This evaluator is more local than the global CLAP similarity evaluator. Instead of comparing only one embedding per piece, it compares distributions of chunk embeddings. This may better capture whether the generated audio has similar local content throughout the piece. Compared with global CLAP similarity, the extra cost comes mainly from computing CLAP embeddings for several chunks of each rendered candidate, rather than one embedding for the whole candidate. The score is also less transparent. It is still an adaptation: the experiment does not reproduce canonical dataset-level FAD, because it uses chunks from one target and one individual rather than two larger audio collections. The small number of chunk embeddings also means that the covariance estimate should be interpreted as a practical search signal rather than as a robust population statistic.

5.4 Symbolic Statistical Evaluator

The symbolic statistical evaluator is inspired by the work of Manaris et al. on automatic music analysis and composition [6]. Their approach uses a reference collection of existing musical pieces and studies statistical regularities in musical material, including Zipf-like rank-frequency distributions. A Zipf-like distribution means that the most frequent symbols occur much more often than lower-ranked symbols, following a decreasing curve when symbols are ordered by frequency. The evaluator implemented in this thesis does not use such a reference collection. Instead, it tests whether symbolic event distributions in an individual resemble synthetically generated Zipf-like distributions. For that reason, it is described as inspired by the work of Manaris et al., rather than as a reproduction of their method.

The evaluator operates directly on the `SymbolicScore`. It extracts five symbolic views of the music: pitches, pitch classes, melodic intervals, inter-onset gaps, and pitch-duration pairs. For each view, the evaluator counts token occurrences, sorts the counts from most frequent to least frequent, and converts the ranked count distribution into a feature vector. This vector describes the size and shape of the distribution: log-transformed token count, log-transformed number of distinct tokens, distinct-to-total ratio, concentration in the most frequent tokens, normalized entropy, Gini coefficient, fitted log-log Zipf slope, fit-quality terms, and the fraction of singleton tokens.

For each symbolic view, a small ridge-regularized linear critic is trained before the search starts. Positive training examples are sampled from noisy Zipf-like distributions, while negative examples are sampled from alternative distributions such as uniform, collapsed, exponential, or Dirichlet-sampled distributions. During evaluation, the candidate's feature vector is normalized with the training mean and scale and passed through the corresponding critic. This produces one scalar critic score for each symbolic

view.

The final fitness value is the mean critic score minus penalties:

$$f(x) = \frac{1}{m} \sum_{i=1}^{m} c_i(x) - p_{\text{material}} - p_{\text{duration}}, \tag{5.4}$$

where $c_i(x)$ is the score of the i-th symbolic critic. The material penalty discourages individuals with too few notes or too few distinct onsets. The duration penalty discourages individuals that exceed the configured maximum duration.

The advantage of this evaluator is that it is fast and interpretable. It can be optimized without rendering audio or calling a large model. Its limitation is that symbolic statistics are only a partial description of music. An individual can have plausible pitch or interval distributions while still lacking phrase-level structure, tension, repetition, or expressive shape. This evaluator is therefore useful as a generation objective, but not as a complete model of musical quality.

5.5 Text-Conditioned CLaMP Evaluator

The text-conditioned evaluator is used for generation rather than reconstruction. Instead of comparing an individual with a target recording, it compares the individual with a text prompt. This follows the broader idea of using learned models as fitness functions for evolutionary music generation, where the goal is specified by a model-mediated judgment rather than by an explicit hand-written rule [8].

The implementation uses CLaMP 3, a contrastive model that maps music and text to a shared embedding space [14, 15]. In this thesis it is not used to retrieve items from a database. It is used only to compute a similarity score between an exported MIDI file and the text prompt. Two pretrained checkpoints are tested:

- **SAAS** (Symbolic-Audio-Aligned Score): a broader CLaMP 3 checkpoint trained with both symbolic and audio alignment stages. For this thesis, it serves as the more general multimodal variant.

- **C2**: a checkpoint intended for symbolic music inputs such as sheet music or MIDI. Since the evolved individuals are exported as MIDI, C2 is the cleaner conceptual match for the generation experiments.

The difference between these checkpoints is evaluated empirically in chapter 7.

Each individual is exported to MIDI and compared with the prompt text using the selected checkpoint. The evaluator receives a similarity score and subtracts penalties for insufficient material and unsuitable duration:

$$f(x) = s_{\text{CLaMP}}(x, q) - p_{\text{material}} - p_{\text{duration}}, \tag{5.5}$$

where q is the text prompt and s_{CLaMP} is the model similarity between the exported MIDI and the prompt.

The strength of text-conditioned evaluation is that it allows the search goal to be stated semantically. This is useful for open-ended generation, where there is no single correct target recording. Its weakness is that a text prompt does not fully define musical form. A fragment may align with the prompt in the learned model representation while still being structurally weak when heard by a listener.

5.6 Comparison of Evaluator Assumptions

The evaluators differ not only in implementation cost, but also in what they assume a good result should be. The global CLAP evaluator assumes that reconstruction can be guided by closeness of whole-piece audio embeddings. The chunk-wise CLAP distance assumes that reconstruction is better described by a distribution of local audio embeddings. The symbolic statistical evaluator assumes that generated music should contain nonuniform symbolic regularities. The text-conditioned evaluator assumes that generation can be guided by similarity between symbolic music and a natural-language description.

Evaluator	**Task**	**Optimized score**	**Main limitation**
Global CLAP similarity	Reconstruction	Cosine similarity to target audio embedding	One vector may miss local musical structure
Chunk-wise CLAP distance	Reconstruction	Negative chunk-distribution distance with length penalty	Expensive and not canonical dataset-level FAD
Symbolic statistical critic	Generation	Mean symbolic critic score minus penalties	Statistics do not guarantee musical form
Text-conditioned CLaMP	Generation	Prompt similarity minus penalties	Semantic match does not fully specify structure

Table 5.1: Summary of the implemented fitness functions.

Because each evaluator measures a different property, their raw fitness values are

not directly comparable. A value from the symbolic statistical evaluator and a value from the CLaMP evaluator do not mean the same thing. The comparison in this thesis therefore focuses on behavior within each evaluator: whether the search improves its own objective, how stable the runs are, what artifacts are produced, and what limitations become visible.

Chapter 6

Experimental Design

The previous chapters described the implementation and the individual fitness functions. This chapter describes how the experiments were organized. Detailed parameter tables belong to chapter B; this chapter only explains which settings matter for interpreting the results.

6.1 Comparison Principle

The experiments were designed to compare evaluator behavior inside the same basic implementation. All experiments use the tree representation described in chapter 4, the same compilation step into `SymbolicScore`, and the same generational search procedure with tournament selection, elitism, crossover, and mutation. The main changing component is the evaluator.

This does not mean that every experimental setting is identical. Some evaluators are much more expensive than others, and the search settings were adjusted to keep the experiments computationally feasible. The comparison is therefore not a perfect isolated test of only one variable. It is a practical comparison of how several evaluator types behave in the implemented system.

6.2 Experiment Tracks

The study is divided into two tracks. The reconstruction track uses a known target recording: the first 30 seconds of "U" by The Hellp [16], consisting of a monophonic piano introduction with no vocals. The commercial target recording is used only as an experimental reference and is not included in the submitted attachments. In this setting, an individual is better if it is more similar to the target according to the selected evaluator. The reconstruction experiments use global CLAP audio similarity and chunk-wise CLAP distance. The target was selected partly to avoid bias toward

well-known or benchmark compositions: its opening is conventional enough to serve as a clear monophonic piano target, but its non-benchmark production context may interact with general-purpose audio embeddings differently from classical, synthetic, or dataset-curated targets.

The generation track does not use one exact target piece. Instead, an individual is scored according to a more general criterion. The symbolic statistical evaluator rewards symbolic distributions similar to synthetic Zipf-like examples. The text-conditioned CLaMP evaluator rewards agreement with a text prompt. These experiments are reported separately from reconstruction because their fitness values have different meanings.

6.3 Shared Search Settings

All runs use a target tempo of 120 BPM, tournament size 3, and elitism count 1. Random seeding is explicit. The main seed is 2026, and repeated comparison runs also use seeds 2027 and 2028 where applicable. This allows a result to be distinguished from a single unusually strong or weak run.

The audio reconstruction experiments use larger populations and deeper trees than the generation experiments. Their variation rates are also more conservative, because each audio evaluation requires rendering and neural-network inference. The generation experiments use smaller populations and shallower trees, but stronger variation. This makes the comparison less clean than a fully controlled evaluator-only experiment, but it keeps the runs computationally feasible.

6.4 Experiment Matrix

Table 6.1 gives a high-level overview of the executed experiments. The exact numeric settings are listed separately in chapter B.

6.5 Measured Quantities

The main measured quantity is the best fitness reached by the search. Because each evaluator defines its own scale, these values are interpreted only within one evaluator family. A CLAP similarity score, a symbolic statistical score, and a CLaMP similarity score do not measure the same thing.

Additional metrics are stored to help interpret the optimized values. Audio experiments store similarity or distance values and penalties. The symbolic statistical experiment stores critic scores, event counts, onset counts, and penalties. The CLaMP

Experiment	Task	Evaluator	Generations	Artifact
Global audio similarity	Reconstruction	Global CLAP audio similarity	25, 50, 100; repeated at 50	WAV
Chunk-wise audio distance	Reconstruction	Chunk-wise CLAP distance	10, 20, 40; repeated at 20	WAV
Symbolic statistics	Generation	Symbolic statistical evaluator inspired by Manaris et al.	6, 12, 24; repeated at 12	MIDI
CLaMP 3 SAAS	Generation	Text-conditioned CLaMP evaluator with SAAS checkpoint	10, 25, 50; repeated at 25	MIDI
CLaMP 3 C2	Generation	Text-conditioned CLaMP evaluator with C2 checkpoint	10, 25, 50; repeated at 25 and 50	MIDI

Table 6.1: Experiment matrix used in the result comparison.

experiments store prompt similarity, event counts, onset counts, and penalties. Runtime is recorded for every run because computational cost is one of the practical differences between evaluators.

The produced musical artifacts are also part of the comparison. Reconstruction experiments produce rendered WAV files. Generation experiments produce MIDI files. These artifacts are necessary because a high internal fitness value does not by itself prove that the result is musically convincing.

6.6 Comparison Method

The comparison focuses on five questions:

1. Does the search improve the objective when the generation budget is increased?
2. How stable is the result across repeated seeds?
3. How much runtime is required for the improvement?
4. Do penalties remain active in the best individuals?
5. What kind of musical artifacts are produced?

This rule is especially important for the two CLaMP variants. The SAAS and C2 checkpoints should not be compared as if their raw scores were on one shared absolute scale. Their comparison is based instead on convergence behavior, runtime, stability across seeds, and the character of the produced artifacts.

Chapter 7

Results

This chapter reports the results of the experiment matrix described in chapter 6. The results are interpreted within each evaluator family, because the raw fitness values of different evaluators are not on a shared scale.

7.1 Overview of Completed Runs

All runs used in the matrix completed successfully. The overall result supports the main experimental claim of the thesis: changing the evaluator changes the behavior of the evolutionary search. Some evaluators produced fast and stable improvement, while others were slower, more expensive, or harder to optimize under the same practical budget.

Table 7.1 summarizes the most important repeated settings. These are the settings used for the main stability comparison because they were executed with several seeds.

Evaluator	**Generations**	**Mean best fitness**	**Std.**	**Mean runtime**
Global CLAP similarity	50	0.4885	0.0149	618.51 s
Chunk-wise CLAP distance	20	−1.4300	0.0080	1251.18 s
Symbolic statistical evaluator	12	0.5029	0.0241	1.13 s
CLaMP 3 SAAS	25	0.3524	0.0311	537.94 s
CLaMP 3 C2	25	0.1042	0.0321	487.90 s
CLaMP 3 C2	50	0.1301	0.0315	881.35 s

Table 7.1: Representative repeated settings from the completed experiment matrix.

7.2 Reconstruction Results

The global CLAP audio similarity experiment produced the clearest reconstruction result among the two audio-based evaluators. At 50 generations, the repeated runs achieved mean best fitness 0.4885 with standard deviation 0.0149 and mean runtime 618.51 seconds. The 25-generation setting was weaker. The single 100-generation run reached best fitness 0.4912, compared with the repeated 50-generation mean of 0.4885.

The chunk-wise CLAP distance experiment behaved differently. Its repeated 20-generation setting achieved mean best fitness -1.4300 with standard deviation 0.0080. A single 40-generation run improved further to -1.3267, but this longer setting was not repeated across seeds and should therefore be interpreted more cautiously. Compared with global CLAP similarity, the chunk-wise method is much more computationally expensive. It also retained nonzero penalties in several completed runs, which indicates that duration or material mismatch still affected the best individuals.

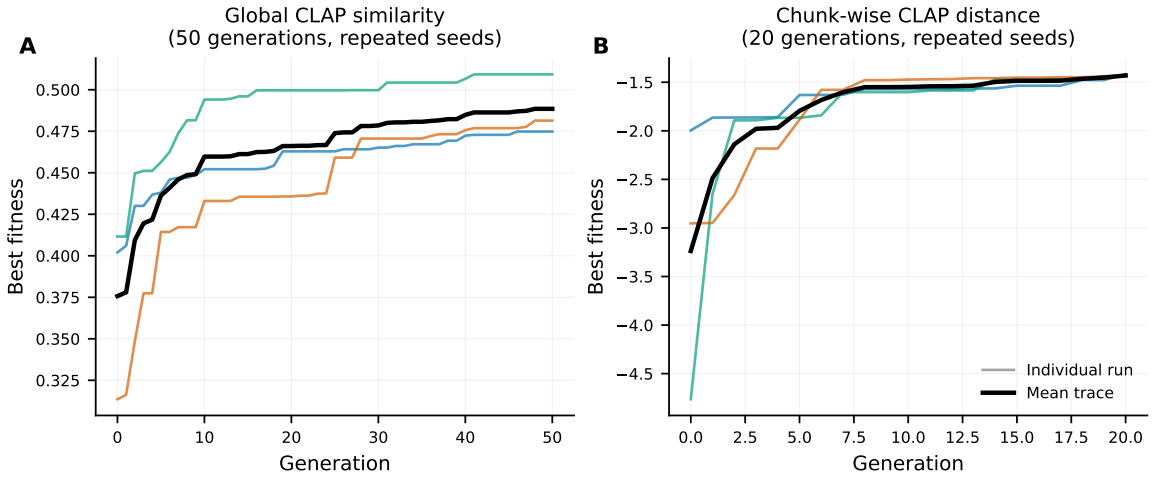

Figure 7.1: Reconstruction convergence traces for the repeated settings. Panel A shows global CLAP similarity at 50 generations across three seeds. Panel B shows chunk-wise CLAP distance at 20 generations across three seeds. Thin colored lines represent individual runs, and the black line represents the mean trace.

7.3 Generation Results

The symbolic statistical evaluator produced the most computationally efficient generation results. The repeated 12-generation setting achieved mean best fitness 0.5029 with standard deviation 0.0241, while the single 24-generation run reached 0.5434. Runtime remained very small: the repeated setting took about 1.13 seconds on average. This makes the symbolic statistical evaluator useful as a computationally efficient generation baseline.

At the same time, the structural statistics show that its fitness value does not determine one unique musical form. The repeated 12-generation runs had noticeable variation in event count and onset count. This is expected: the evaluator rewards symbolic statistical properties, not a fixed melody or fixed structure.

The text-conditioned CLaMP experiments produced a more mixed result. Under the SAAS checkpoint, repeated 25-generation runs achieved mean best fitness 0.3524 with standard deviation 0.0311, and the single 50-generation run reached 0.4345. Event count and onset count were comparatively stable across the repeated medium-budget runs. This suggests that the SAAS setting was relatively friendly to the evolutionary search, even though it is not the cleanest conceptual match for symbolic music retrieval.

The C2 checkpoint was introduced because it better matches the symbolic modality of the evolved individuals. The repeated 25-generation C2 runs achieved mean best fitness 0.1042 with standard deviation 0.0321, while the repeated 50-generation C2 runs achieved mean best fitness 0.1301 with standard deviation 0.0315. These values should not be compared directly with SAAS as absolute scores. Within the C2 configuration itself, however, the larger generation budget produced only weak improvement. Event-count and onset-count variation also remained higher than in the SAAS runs.

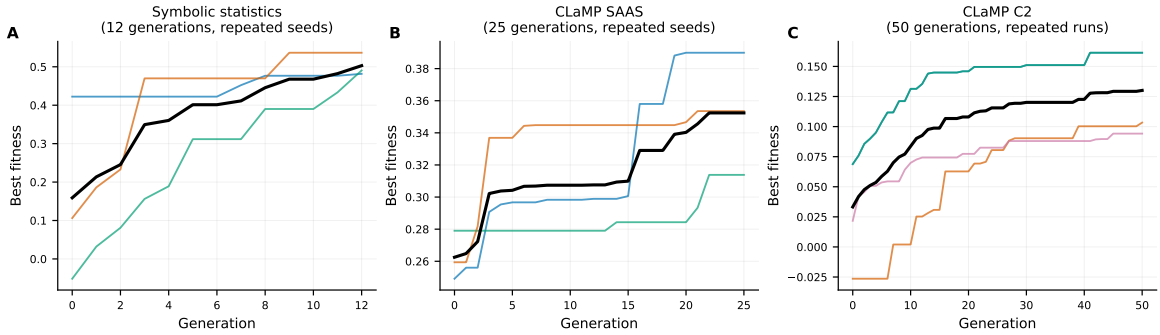

Figure 7.2: Generation convergence traces for the repeated settings. Panel A shows the symbolic statistical evaluator at 12 generations across three seeds. Panel B shows the CLaMP 3 SAAS setup at 25 generations across three seeds. Panel C shows the CLaMP 3 C2 setup at 50 generations across repeated runs.

7.4 Runtime and Stability

Runtime differed strongly between evaluator families. The symbolic statistical evaluator was by far the fastest, because it operates directly on symbolic events and does not require audio rendering or neural-network inference. The audio-based evaluators were the slowest, especially the chunk-wise CLAP distance, which must render audio, split it into chunks, compute several embeddings, and compare their distribution.

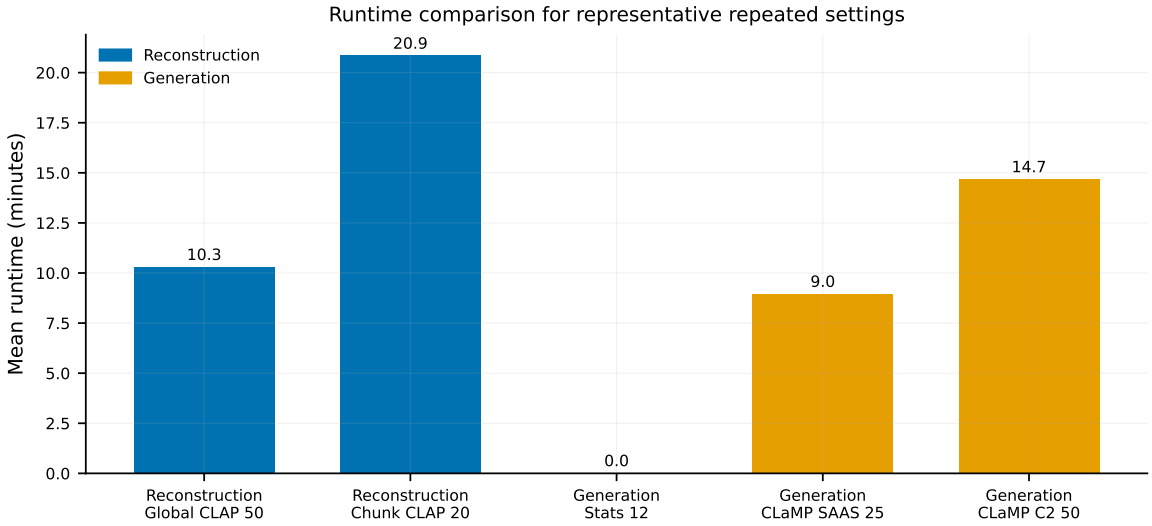

Figure 7.3: Mean runtime comparison for representative repeated settings. Reconstruction settings are shown together with generation settings to highlight the computational difference between audio-based and symbolic evaluators.

Seed variability was moderate in most repeated settings. Global CLAP similarity had low variation at 50 generations. The chunk-wise CLAP distance was also numerically stable at 20 generations, but this stability came with high runtime. The symbolic statistical evaluator had low fitness variation, but larger variation in structural metrics. For CLaMP, raw fitness values are checkpoint-specific, so stability is interpreted mainly within each variant. The C2 runs showed higher structural variability than the SAAS runs.

7.5 Generated Musical Artifacts

The experiments produced both audio and MIDI artifacts. The reconstruction experiments produced WAV files that can be compared with the target recording. The generation experiments produced MIDI files. These artifacts are important because the optimized fitness value is only the internal objective of the search. It must be checked against the actual musical output.

The global CLAP reconstruction artifacts are the most straightforward result to interpret: they are produced by directly optimizing similarity to a target audio recording. The chunk-wise reconstruction artifacts are more expensive to obtain and should be interpreted with their remaining penalties in mind. The symbolic statistical artifacts show that the search can quickly produce material matching the selected symbolic statistics, but the score does not guarantee richer musical form. The CLaMP artifacts show the promise and difficulty of text-conditioned generation: the prompt gives an

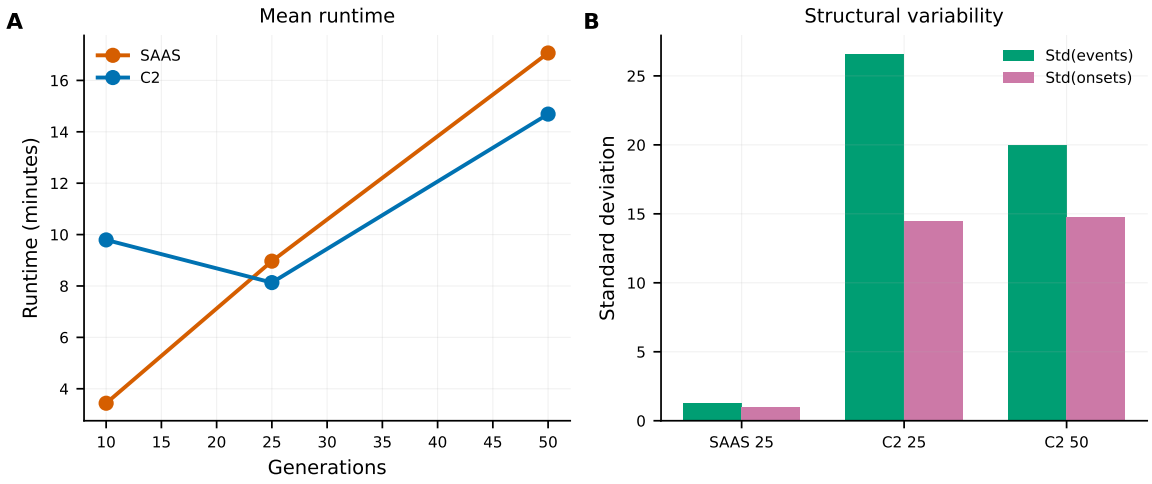

Figure 7.4: Comparison of the two CLaMP 3 checkpoint variants using quantities that do not depend on the raw checkpoint-specific fitness scale. Panel A shows mean runtime. Panel B shows structural variability, measured by the standard deviation of event count and onset count.

open-ended goal, but the resulting score still does not fully define what should sound musically convincing.

7.6 Summary of Main Observations

The completed runs support the following observations:

1. The evaluator strongly affects runtime and optimization behavior.
2. Global CLAP similarity currently gives the most practical reconstruction result.
3. Chunk-wise CLAP distance is useful but expensive under the current budget.
4. The symbolic statistical evaluator is a very fast generation baseline.
5. The CLaMP 3 C2 checkpoint is conceptually cleaner for symbolic input, but showed weak within-checkpoint improvement and higher structural variability in the current experiments.

Chapter 8

Discussion

The results show that the choice of evaluator is not a small implementation detail in evolutionary music generation. It affects what the search rewards, how quickly the search improves, how expensive the run becomes, and what kind of musical artifact is produced. This supports the main direction of the thesis: the assigned goal of producing short music with an evolutionary algorithm can be studied meaningfully through the design and comparison of fitness functions.

8.1 Effect of the Evaluator

All experiments used the same basic representation and search procedure, but the results differed substantially. The global CLAP similarity evaluator produced clearer reconstruction behavior than the chunk-wise CLAP distance under the available budget. The symbolic statistical evaluator was much faster than the learned-model evaluators. The CLaMP experiments showed that a semantically richer objective can be useful, but also harder to optimize.

This means that the evaluator shapes the practical search problem. It does not only assign a number after the music has been produced. It decides which individuals are selected as parents and therefore which musical properties are preserved or amplified in later generations. A change of evaluator can therefore change the direction of the whole search, even when the representation and variation operators remain the same.

8.2 Reconstruction and Generation

The reconstruction experiments are easier to interpret because they have a fixed target recording. In this setting, the global CLAP similarity evaluator was the more practical option. It is simpler than the chunk-wise distance and produced a better runtime-to-result trade-off. The chunk-wise distance remains interesting because it compares

distributions of local audio chunks, but the current implementation is expensive and still affected by duration or material penalties.

The generation experiments are less direct. There is no single correct output, so the evaluator defines what kind of musical property should be rewarded. The symbolic statistical evaluator rewards event distributions. This makes it fast and interpretable, but also narrow. The CLaMP evaluator rewards agreement with a text prompt. This is closer to open-ended composition, but the prompt does not fully specify melody, structure, or development.

The comparison between the CLaMP SAAS and C2 checkpoints is especially important. C2 is the cleaner conceptual match for symbolic input, because the evolved individuals are exported as MIDI. However, it was not easier to optimize in the current experiments. This shows that conceptual fit and search behavior are related, but not the same thing.

8.3 Fitness and Musical Quality

The experiments also show why optimized fitness should not be treated as the same thing as musical quality. A fitness function measures only the property it was designed to measure. Global CLAP similarity measures closeness in a learned audio representation. The symbolic statistical evaluator measures selected symbolic regularities. CLaMP measures similarity between a symbolic artifact and a text description according to a learned model.

Each of these measurements is useful, but partial. A high score can indicate that the search found something relevant to the evaluator, but it does not guarantee that the result is musically convincing to a listener. For that reason, the produced artifacts remain an important part of the thesis. The numerical results show how the search behaves; the artifacts show what kind of music the search actually produced.

The weaknesses visible in the results follow a common pattern. Evaluators that are computationally efficient and transparent (symbolic statistics) capture only surface regularities. Evaluators that are semantically richer (CLaMP) are harder to optimize and less transparent. Audio-based evaluators (global and chunk-wise CLAP) sit between these extremes: they reward genuine audio similarity, but their scores are shaped by the embedding model rather than by explicit musical criteria. This pattern is not a flaw of any single evaluator—it reflects the underlying difficulty of the fitness-design problem discussed in chapter 3.

8.4 Limitations

The comparison is not a fully controlled evaluator-only experiment. The audio evaluators are much more expensive than the symbolic evaluator, and the experimental settings were adjusted to keep the runs feasible. This is why the strongest conclusions are within each task track: comparing the two reconstruction evaluators, comparing generation evaluators, and comparing CLaMP variants within the same family.

Another limitation is that the analysis relies mainly on internal fitness values, runtime, penalties, structural metrics, and inspection of produced artifacts. A more complete evaluation would include a listening study or another external assessment of musical quality. Such evaluation would be slower and less repeatable, but it would address the question that automatic metrics cannot fully answer: whether listeners perceive the produced music as convincing.

Future work could also replace or supplement the chunk-wise covariance distance with Kernel Audio Distance (KAD). Chung et al. propose KAD as a Maximum Mean Discrepancy-based, distribution-free alternative to FAD with better behavior under limited sample sizes [17]. In this thesis, KAD remains a future option rather than an implemented evaluator.

Conclusion

This thesis studied the composition and reconstruction of short musical artifacts with the help of an evolutionary algorithm. The work approached the assigned topic through the design and comparison of fitness functions. Instead of proposing one universal music generator, it implemented one common evolutionary system and used it to compare several evaluators.

The implementation represents each individual as a tree, compiles it into a symbolic score, and then evaluates it through symbolic, MIDI-based, or audio-based views of the same musical artifact. This made it possible to run both reconstruction experiments, where the goal is similarity to a target recording, and generation experiments, where the goal is agreement with a more general statistical or text-conditioned criterion.

The results show that the evaluator strongly affects the behavior of the search. For reconstruction, global CLAP audio similarity was the most practical evaluator in the current implementation, while the chunk-wise CLAP distance was more expensive and harder to optimize cleanly. For generation, the symbolic statistical evaluator was a very fast baseline, while the CLaMP experiments showed the promise and difficulty of using semantic text-conditioned evaluation. The comparison between the CLaMP SAAS and C2 variants showed that a conceptually better match to symbolic music does not automatically produce an easier search problem.

The main conclusion is that in evolutionary music systems, the fitness function does more than measure the result. It helps define what kind of music the search can find. A useful evaluator must therefore be considered not only by what it is intended to measure, but also by how it behaves when optimized repeatedly by an evolutionary algorithm.

The thesis also produced concrete musical artifacts in both task settings. This is important for the assigned goal: the work does not remain only a theoretical comparison of metrics, but demonstrates that the implemented evolutionary search can produce short reconstruction and generation outputs. At the same time, the artifacts and results show that automatic fitness values are only partial descriptions of musical quality.

Future work could extend the experiments in several directions. The comparison could be strengthened by using more uniform search settings across evaluator families, by testing larger generation budgets, or by combining several fitness signals into one

multi-objective setup. A listening study would also be valuable, because it would compare the automatic scores with human perception of the produced artifacts. Finally, the symbolic representation itself could be expanded to support richer rhythm, harmony, and longer musical form.

Bibliography

[1] Thomas Bartz-Beielstein, Juergen Branke, Jorn Mehnen, and Olaf Mersmann. Evolutionary algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4:178–195, 05 2014.

[2] MIDI Manufacturers Association. MIDI 1.0 detailed specification. `https://midi.org/midi-1-0-detailed-specification`, 1996. Document Version 4.2.1.

[3] Gerhard Nierhaus. *Algorithmic Composition: Paradigms of Automated Music Generation*. Springer, Vienna, 2009.

[4] John A. Biles. Genjam: A genetic algorithm for generating jazz solos. In *Proceedings of the International Computer Music Conference*, pages 131–137. International Computer Music Association, 1994.

[5] M. Marques, V. Oliveira, S. Vieira, and A. C. Rosa. Music composition using genetic evolutionary algorithms. In *Proceedings of the 2000 Congress on Evolutionary Computation*, pages 714–719. IEEE, 2000.

[6] Bill Manaris, Patrick Roos, Penousal Machado, Dwight Krehbiel, Luca Pellicoro, and Juan Romero. A corpus-based hybrid approach to music analysis and composition. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 839–845. AAAI Press, 2007.

[7] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms. In *Proceedings of Interspeech 2019*, pages 2350–2354. ISCA, 2019.

[8] Fabian Ostermann, Jonas Kramer, and Günter Rudolph. Using large language models as fitness functions in evolutionary algorithms for music generation. In *Proceedings of the AI Music Creativity Conference*. Zenodo, 2025.

[9] Trevor Bača. Supriya. `https://github.com/supriya-project/supriya`, 2026. Python API for SuperCollider.

[10] SuperCollider contributors. Supercollider. `https://supercollider.github.io/`, 2026. Audio synthesis programming language and sound server.

[11] Riccardo Poli, William B. Langdon, and Nicholas Freitag McPhee. *A Field Guide to Genetic Programming*. Lulu.com, 2008.

[12] Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, and Huaming Wang. CLAP: Learning audio concepts from natural language supervision. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1–5. IEEE, 2023.

[13] Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1–5. IEEE, 2023.

[14] Shangda Wu, Dingyao Yu, Xu Tan, and Maosong Sun. CLaMP: Contrastive language-music pre-training for cross-modal symbolic music information retrieval. In *Proceedings of the 24th International Society for Music Information Retrieval Conference*, pages 157–165, 2023.

[15] Shangda Wu, Zhancheng Guo, Ruibin Yuan, Junyan Jiang, SeungHeon Doh, Gus Xia, Juhan Nam, Xiaobing Li, Feng Yu, and Maosong Sun. CLaMP 3: Universal music information retrieval across unaligned modalities and unseen languages. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2605–2625. Association for Computational Linguistics, 2025.

[16] The Hellp. U. Track on *LL*, 2024. Written and produced by Chandler Ransom Lucy and Noah Patrick Dillon.

[17] Yoonjin Chung, Pilsun Eu, Junwon Lee, Keunwoo Choi, Juhan Nam, and Ben Sangbae Chon. KAD: No more FAD! an effective and efficient evaluation metric for audio generation, 2025.

Appendix A

Installation and Running Instructions

The source code and experiment results are submitted together with the thesis as separate attachments. This appendix gives only the information needed to orient the reader in those attachments and to reproduce the experiment runs.

A.1 Repository Structure

The implementation is stored in the `evau` directory. The most important subdirectories are:

- `evau/shared`: shared implementation of the tree representation, symbolic score, MIDI export, audio renderer, evaluator interface, and search procedure;
- `evau/experiments`: experiment scripts used in the thesis;
- `evau/clamp3`: bundled CLaMP 3 code and model files used by the text-conditioned experiments;
- `evau/samples`: input samples, including the target audio used by reconstruction experiments;
- `evau/results`: generated run records, summaries, logs, figures, and produced artifacts.

The thesis text itself is stored separately in `thesis-template/src`.

A.2 Environment Setup

The Python part of the implementation requires Python 3.12 or newer. The experiments are run with `uv`. Each experiment script contains inline dependency metadata, and the shared package is used as an editable local package from `evau/shared`.

Audio rendering is performed through Supriya in SuperCollider non-realtime mode. Therefore, the audio-based experiments require a working SuperCollider installation in addition to the Python packages. The CLAP-based experiments also use pretrained neural-network models from the Hugging Face ecosystem. If the models are not already present in the cache, they are downloaded during the first run.

The CLaMP experiments use the bundled `evau/clamp3` directory. The experiment runner selects the CLaMP variant through the `CLAMP3_VARIANT` environment variable and calls `clamp3_score.py` internally.

A.3 Running the Experiment Matrix

The full experiment matrix is controlled by the matrix runner script in the experiments directory. From the `evau` directory, the prioritized matrix used for the thesis can be run with:

```
uv run experiments/run_thesis_matrix.py --mode prioritized
```

The same script can run only selected experiments. For example, the symbolic statistical experiment can be run with:

```
uv run experiments/run_thesis_matrix.py \
  --only exp_05_symbolic_stats
```

Individual experiment scripts can also be run directly. The shared command-line arguments are `-seed`, `-generations`, `-output-dir`, and `-summary-path`. For example:

```
uv run experiments/exp_05_symbolic_stats.py \
  --generations 12 --seed 2026
```

The CLaMP experiment additionally accepts the `-clamp-variant` argument:

```
uv run experiments/exp_06_clamp.py --clamp-variant c2
```

A.4 Output Files

The experiment matrix writes several result files into `evau/results`. The most important ones are:

- `EXPERIMENT_RUNS.csv` and `EXPERIMENT_RUNS.json`: one record per executed run;
- `EXPERIMENT_AGGREGATES.csv`: aggregated values grouped by experiment and generation count;

- `EXPERIMENT_REPORT.md`: generated textual report;
- `figures`: plots used in the results chapter;
- `artifacts`: generated WAV and MIDI outputs;
- `logs`: standard output and error logs for individual runs.

The generated music examples are not duplicated as a separate appendix. They are included in the submitted result attachment and are referenced through the result records.

Appendix B

Experiment Configuration Tables

This appendix lists the experiment parameters that are too detailed for the main text. The values come from the experiment scripts in `evau/experiments` and from the experiment matrix runner.

B.1 Shared Matrix Settings

Setting	**Value**
Target tempo	120 BPM
Selection method	Tournament selection
Tournament size	3
Elitism count	1
Main seed	2026
Additional repeat seeds	2027, 2028
Matrix runner	`evau/experiments/run_thesis_matrix.py`
Main result directory	`evau/results`
Run records	`EXPERIMENT_RUNS.csv`, `EXPERIMENT_RUNS.json`
Aggregated results	`EXPERIMENT_AGGREGATES.csv`

Table B.1: Settings shared by the experiment matrix.

B.2 Reconstruction Experiments

B.3 Generation Experiments

The text prompt used in both CLaMP configurations was:

"a calm, sparse, melancholic melody with gentle motion"

B.4 Stored Metrics

The values in these tables are configuration values. The measured results are reported in chapter 7 and stored in the submitted result attachment.

Setting	**Global CLAP similarity**	**Chunk-wise CLAP distance**	**Meaning**
Script	`exp_02_audio_global_similarity.py`	`exp_03_audio_chunk_distance.py`	Experiment file
Task	Reconstruction	Reconstruction	Uses target audio
Output artifact	WAV	WAV	Produced file type
Generation budgets	25, 50, 100	10, 20, 40	Tested run lengths
Repeated budget	50 generations	20 generations	Multi-seed setting
Population size	50	50	Number of individuals
Crossover rate	0.5	0.5	Probability of crossover
Mutation rate	0.2	0.2	Probability of mutation
Maximum tree depth	10	10	Structural limit
Maximum node count	30	30	Structural limit
Maximum duration	32 beats	128 beats	Duration limit
Sample rate	48000 Hz	48000 Hz	Audio rendering/evaluation
CLAP model	`laion/clap-htsat-fused`	`laion/clap-htsat-unfused`	Audio embedding model
Extra setting	Whole-audio embedding	3 second chunks	Evaluator detail

Table B.2: Configuration of the reconstruction experiments.

Setting	**Symbolic statistics**	**CLaMP 3 SAAS**	**CLaMP 3 C2**
Script	`exp_05_symbolic_stats.py`	`exp_06_clamp.py`	`exp_06_clamp.py`
Task	Generation	Generation	Generation
Output artifact	MIDI	MIDI	MIDI
Generation budgets	6, 12, 24	10, 25, 50	10, 25, 50
Repeated budget	12 generations	25 generations	25, 50 generations
Population size	24	12	12
Crossover rate	1.0	1.0	1.0
Mutation rate	1.0	1.0	1.0
Maximum tree depth	6	6	6
Maximum node count	28	28	28
Duration setting	24 beat maximum	24 beat target	24 beat target
Main evaluator input	Symbolic score	MIDI and text prompt	MIDI and text prompt
Variant argument	none	`-clamp-variant saas`	`-clamp-variant c2`

Table B.3: Configuration of the generation experiments.

Experiment	**Stored metrics**
Global CLAP similarity	Best fitness and CLAP similarity
Chunk-wise CLAP distance	Best fitness, objective value, CLAP distance, length penalty
Symbolic statistical evaluator	Best fitness, individual critic scores, mean critic score, event count, onset count, penalty
CLaMP 3 evaluator	Best fitness, prompt similarity, event count, onset count, penalty

Table B.4: Main diagnostic metrics stored by the experiment scripts.

Appendix C

Generated Music Examples

This appendix briefly characterises the musical artifacts produced by the experiments. Full audio and MIDI outputs are in the submitted result attachment under `evau/results/artifacts`. All descriptions below are the author's informal observations, not a formal listening study.

C.1 Reconstruction Outputs

Spectral inspection of the best global CLAP similarity outputs shows that the search successfully approached the frequency profile of the target recording: the dominant spikes in the spectrogram matched those of the target, indicating that the instrument and its pitch content were broadly identified. The structure of the best output, however, is far from the target. The target recording is repeating, monophonic, and predictable in its event organization, whereas the best evolved output is more chaotic. The tree representation can express repetition and transposition, so the simple target structure is in principle reachable, but the global CLAP evaluator did not drive the search strongly toward structural match within the tested budget.

The chunk-wise CLAP distance outputs did not show a clear advantage over the global CLAP outputs in informal listening. No consistent preference emerged, despite the higher computational cost of the chunk-wise evaluator.

C.2 Generation Outputs

The symbolic statistical evaluator produced listenable outputs already from the first generation. A recurring weakness was observed, however. On seed 2026, a degenerate subpart at the end of the best individual was carried from generation 6 to generation 24 without structural change; only its instrument assignment was altered. This subpart scored well under the statistical critics but was audibly unpleasant, illustrating the gap

between a symbolic-statistical score and human perception.

The text-conditioned CLaMP evaluators produced weaker outputs with high variance between generations. The search showed signs of many local optima and did not converge reliably under the tested budgets. This is consistent with the numerical results in chapter 7, where the CLaMP C2 runs in particular showed low mean fitness and high structural variability across seeds.

C.3 Listening Notes

No formal listening study was conducted. A structured perceptual evaluation with multiple listeners remains a direction for future work.