Technische Hochschule
Brandenburg
University of
Applied Sciences

Fachbereich Informatik und Medien

Anwendung interaktiver evolutionärer Algorithmen zur Erzeugung von Schlagzeugrhythmen

Arbeit zur Erlangung des akademischen Grades Master of Science

Vorgelegt von: Mario Kaulmann

am: 15.10.2019

Erstbetreuer: Dipl.-Inform. Ingo Boersch

Zweitbetreuer: Prof. Dr. rer. nat. Martin Christof Kindsmüller

Zusammenfassung

Interessante Schlagzeugrhythmen zu finden ist eine kreativ anspruchsvolle Aufgabe. Es gibt die Möglichkeiten verschiedene Instrumente zu verschiedenen Zeitpunkten zu spielen. Die Anordnung der zu spielenden Instrumente in einem Zeitverlauf muss dabei wiederholbar sein und dem Schlagzeuger gefallen.
Zur Unterstützung bei diesem Prozess werden interaktive evolutionäre Algorithmen vorgeschlagen. Durch die Interaktivität kann der Nutzer die Suche steuern und die Abwandlungsoperatoren des evolutionären Algorithmus erzeugen neue Vorschläge. Ein Schwerpunkt dieser Arbeit liegt auf der Benutzeroberfläche. Diese soll die Ermüdung des Nutzers gering halten und auch zur Nachvollziehbarkeit des evolutionären Algorithmus beitragen.
Theoretische Grundlagen werden erläutert, inspirierende Arbeiten betrachtet, die Konzeption und Umsetzung eines Demonstrationsprogramms beschrieben und Versuche mit dem Programm dokumentiert und ausgewertet.

Abstract

To find interesting drum rhythms is a high creativity task. It is possible to use a variety of different instruments at different times. The order of the instruments to play over time has to be repeatable and has to please the drummer.
Interactive evolutionary algorithms are suggested to support this task. Due to the interactivity a user can control the search for new rhythms and the variation operators of the evolutionary algorithm create new proposals. The user interface is a main topic for this work. It should decrease the user fatigue and help to understand the evolutionary algorithm.
Theoretical fundamentals will be presented, inspiring works will be shown, the concept and implementation of an application are described and experiments with this application are documented and evaluated.

Danksagung

An dieser Stelle möchte ich mich bei denjenigen bedanken, die mich bei dieser Arbeit unterstützt haben.

Ich bedanke mich bei Herrn Boersch und Prof. Dr. Kindsmüller, dass sie diese Arbeit betreut und begutachtet haben und hilfreiche Anregungen dazu beigetragen haben.

Außerdem bedanke ich mich bei Sven Herzfeldt, für die Erstellung der Sound-Dateien für die Instrumente, die mit der Technik Flame gespielt werden.

Ein großer Dank gilt Florian Seeliger dafür, dass er als Schlagzeuger (1,5 Jahre Unterricht) die Anwendung in verschiedenen Stadien der Entstehung getestet und wertvolles Feedback gegeben hat.

Dank gebührt auch den Personen, die an den Versuchen teilgenommen haben.

Mario Kaulmann
Brandenburg an der Havel, 15.10.2019

Inhaltsverzeichnis

1 Einleitung

Zitat: People don't know what they like, but they always like what they know. (Dieter Lange)
Übersetzt: Menschen wissen nicht, was sie wollen/mögen, aber sie wollen/mögen das was sie kennen.

1.1 Motivation

Wenn man das Schlagzeugspielen lernt, beginnt man häufig damit bekannte Standardrhythmen und Figuren auf einer Trommel zu lernen. Fortschreitend werden dann neue Rhythmen einer speziellen Musikrichtung gelernt. Dadurch wird ein Schlagzeuger geformt, den man mit den Etiketten seiner gelernten Stilrichtungen versehen kann. Das Verlassen der Nischen, die der Schlagzeuger bedienen kann, erfordert es nicht in Kategorien zu denken.
Wenn man für sich neue Rhythmen erkunden möchte, kann man dazu technische Hilfe zum Beispiel durch Drum-Loop-Programme in Anspruch nehmen. Diese bieten einem Benutzer die Möglichkeit, durch zum Beispiel Klicken in einem Gittermodell, einen Rhythmus zu erzeugen und abzuspielen. Dabei ist ein hohes Maß an Kreativität von Nöten, wenn man einen Rhythmus erzeugen will, der einem gefällt. Des Weiteren muss dabei berücksichtigt werden, wenn man den Rhythmus nachspielen möchte gibt es Beschränkungen, die durch die Physis des Menschen gegeben sind.
Weiterhin besteht die Möglichkeit Rhythmen über Internetplattformen und Bücher zu beziehen. Das erfordert allerdings wiederum, dass man gezielt etwas fachbezogenes, zum Beispiel eine Stilrichtung oder eine Technik, sucht. Das gleiche geht ebenfalls indem man nach Vorbildern oder nach bestimmten Liedern sucht, deren Spielart einen gut gefällt.
In dieser Arbeit wird ein Verfahren mit interaktiven evolutionären Algorithmen vorgeschlagen, mit dem Rhythmen erkundet werden können, die man zum Üben verwenden kann. Durch die Interaktivität wird der Nutzer eingeladen eine Entdeckungsreise in der Welt der Rhythmen zu erleben. Durch die Unbefangenheit des evolutionären Algorithmus kann der Nutzer dabei auf Rhythmen stoßen, die sich von dem unterscheiden, was er gelernt hat. Die

Interaktion mit dem Programm und eine Darstellung der Entwicklung des Fortschreitens können dabei als treibende Kräfte genutzt werden für einen selbst neuartige Rhythmen zu entdecken.
Ein weiterer Vorteil bei der Nutzung evolutionärer Algorithmen ist das Prinzip, dass ähnliche Lösungen ähnliche Bewertungen haben. Das bedeutet, dass Rhythmen, die von dem Algorithmus vorgeschlagen werden und als gut bewertet werden die Grundlage für kommende Rhythmusvorschläge bilden und damit eine hohe Wahrscheinlichkeit besteht, dass die neuen Vorschläge ebenfalls gut bewertet werden. Die Ähnlichkeit der Rhythmen kommt auch der Art entgegen, wie Schlagzeuger Rhythmen lernen können. Es ist einfacher ein Rhythmusmuster zu lernen, das auf einem bekannten Muster basiert.

1.2 Ziel der Arbeit

Das Ziel der Arbeit ist es ein Programm zur Erkundung von Schlagzeugrhythmen zu erstellen, dass ein interaktives evolutionäres Verfahren anwendet. Dabei sollen Rhythmen entdeckt werden, die der Nutzer so interessant findet, dass er diese lernen möchte. Die vom Nutzer ausgewählten Rhythmen sollen als Noten und als Audiosignal ausgegeben werden.
Das interaktive evolutionäre Verfahren soll eine Mensch-Maschinen-Schnittstelle bereitstellen, die eine Bewertung der Rhythmen ermöglicht, die Entdeckungstour steuerbar und nachvollziehbar macht und dabei einfach zu bedienen ist. Im speziellen werden nur Rhythmen angezeigt, die der Physis eines Menschen entsprechend spielbar sind. Außerdem sollen nur Elemente zum Einsatz kommen, die der Nutzer möchte. Damit ist gemeint, dass nur die Schlagzeugteile, wie Trommeln und Becken, verwendet werden, sowie Schlagtechniken, die der Nutzer ausgewählt hat, um Rhythmen zu erkunden.

1.2.1 Einfachheit

Ein Nutzer, der gewohnt ist mit Computern zu arbeiten soll ohne extra Anleitung in der Lage sein das Programm zu bedienen.

1.2.2 Individualisierbarkeit

Es soll dem Nutzer möglich sein, das Programm so zu erweitern, dass es mit seinen eigenen Instrumenten arbeitet. Diese können ergänzend zu den Instrumenten hinzugefügt werden, die als Grundelemente dienen.

1.2.3 Kontrolle

Der Nutzer soll jederzeit die Kontrolle über die Entwicklung des Verlaufs haben. Er soll die Möglichkeit haben zu beeinflussen, wie stark die Änderungen zwischen den einzelnen Stufen ausfallen und auch selber Individuen vorschlagen und manuell modifizieren können.

1.2.4 Nachvollziehbarkeit

Durch geeignete Darstellungen soll die Nachvollziehbarkeit des Verlaufs im Programm gewährleistet werden. Außerdem werden Daten gespeichert, die auch für Auswertungen in der Nachbereitung analysiert werden können.

1.3 Aufbau der Arbeit

In Kapitel 2 wird die Aufgabenstellung analysiert, dabei werden Begriffe erläutert und die Funktionen des Programms, das zu dieser Arbeit entwickelt wird, werden festgehalten.
Kapitel 3 widmet sich den Grundlagen, die zum Verstehen dieser Arbeit benötigt werden.
In Kapitel 4 werden ähnliche Arbeiten betrachtet, um einen kleinen Einblick zu bekommen, was andere Leute in diesem Gebiet bereits versucht haben und um Anreize zu bekommen, was man selbst ausprobieren kann.
In Kapitel 5 wird auf die Konzeption eingegangen. Dabei wird erläutert, welche Verfahren und Mittel eingesetzt werden und warum.
Kapitel 6 beschreibt die Realisierung. In diesem Kapitel wird auf die konkrete Umsetzung im Programm eingegangen.
Kapitel 7 beschäftigt sich mit der Evaluation. Hier werden Versuche vorgestellt und ausgewertet, die als Beleg dienen, dass die Konzepte und die Umsetzung in Form des Programms funktionieren.
In Kapitel 8 wird die Arbeit kurz zusammengefasst und ein Ausblick auf mögliche Folgearbeiten zu dieser Arbeit gegeben.

2 Aufgabenanalyse

2.1 Aufgabenstellung

Das Ergebnis dieser Arbeit soll eine gebrauchstaugliche Demonstrationsanwendung für evolutionäre Algorithmen sein. Das Anwendungsgebiet soll die Erzeugung von Schlagzeugrhythmen sein, mit der Besonderheit, dass diese zum selbstständigen Üben verwendet werden können. Dabei werden Fähigkeiten und Instrumente des Nutzers berücksichtigt. Der Nutzer soll die Möglichkeit haben, eigene Individuen zu erzeugen, die Einfluss auf die Evolution haben.
Ein Schwerpunkt dieser Arbeit ist die Mensch-Maschine-Schnittstelle. Diese soll auf Grundlage des Standes der Forschung, sowie mit eigenen Ideen umgesetzt werden. Schwierigkeiten bei der Aufgabe bestehen in der Nutzerermüdung und der Eigenschaft der Individuen zeitsequenziell zu sein.
Die Demonstrationsanwendung soll modular aufgebaut sein und sämtliche Umgebungesvoraussetzungen mitliefern, die zur Inbetriebnahme erforderlich sind. Das kann zum Beispiel die benötigte Java-Version sein. Die Parameter des Programms sollen deklarativ und einfach änderbar repräsentiert werden. Die Änderung der Parameter soll im laufenden Betrieb der Anwendung möglich sein. Damit ist gemeint, dass man den evolutionären Algorithmus pausieren kann und dann die Parameter umstellen kann. Beispielsweise kann man die Populationsgröße ändern. Des Weiteren soll der Demonstrator geeignet dokumentiert und evaluiert werden. Zur technischen Dokumentation werden die Quellcodes mit Kommentaren versehen, sowie Methoden erläutert.
Da es sich bei der Anwendung um eine Demonstrationsanwendung handelt, werden Möglichkeiten bereitgestellt, die den Ablauf der Evolution nachvollziehbar machen. Eine Möglichkeit kann die Bereitstellung des Stammbaums sein.

2.2 Begriffsklärungen

Der Begriff *gebrauchstauglich* ist so zu verstehen, dass die Anwendung dem Benutzer eine zufriedenstellende Nutzung zur effizienten Erreichung seines Ziels ermöglicht. Der Benutzer

kann einer von zwei Benutzergruppen zugeordnet werden.
Die eine Gruppe sind Schlagzeuger. Das Ziel eines Schlagzeugers ist es Rhythmen zu erzeugen, die für ihn neu und interessant sind.
Die andere Gruppe sind Informatiker. Das Ziel eines Informatikers ist es die Vorgänge des Evolutionären Algorithmus nachzuvollziehen. Wobei auch die Auswirkungen verschiedener Parameter getestet werden können.
Der Begriff *Demonstrationsanwendung* ist so zu verstehen, dass die Anwendung als Umsetzung der theoretischen Konzepte dieser Arbeit funktionieren soll. Es werden keine kommerziellen Zwecke damit verfolgt.
Der Begriff *Schlagzeugrhythmus* ist so zu verstehen, dass es sich um eine wiederholbare Abfolge von Klängen erzeugt von Schlaginstrumenten handelt.

2.3 Anforderungen

In diesem Abschnitt werden Funktionen konkret definiert, die durch das Programm umgesetzt werden müssen. Dafür werden fünfstellige bis sechsstellige Funktionsbezeichner eingeführt, die aus einem Buchstaben und vier Ziffern bestehen. Der Buchstabe steht für eine der folgenden Kategorien.
Das sechste Zeichen ist optional und kennzeichnet einen Zusatz zu einer Anforderung, um diese näher zu erläutern. Dabei handelt es sich um kleingeschriebene Buchstaben.
E: evoulutionärer Algorithmus
G: graphische Benutzeroberfläche
D: Dokumentation
S: sonstiges

2.3.1 Funktionen evolutionärer Algorithmus

/E0001/
Der zyklische Ablauf eines interaktiven genetischen Algorithmus wird umgesetzt. Dieser umfasst Interaktion des Nutzers, Selektion und Durchführung der Abwandlungsoperationen.
/E0001a/
Die Interaktion des Nutzers besteht in der Bewertung (Fitnessbestimmung) der Individuen, Möglichkeit eigene Individuen zu erzeugen, Änderung der Hyperparameter des evoultionären Algorithmus, das Anstoßen der Erzeugung einer neuen Generation und das Beenden der Suche nach neuen Individuen.

/E0001b/
Die Selektion basiert auf Rängen.
/E0001c/
Die Abwandlungsoperatoren (Mutation und Crossover) werden in ihrer ursprünglichen und in erweiterten Versionen umgesetzt.

/E0002/
Die Initialisierung der Startpopulation erfolgt durch eine Mischung aus zufällig erzeugten Individuen und musterbasierten Individuen. Das Verhältnis wird durch den Nutzer bestimmt.

/E0003/
Individuen sollen so gestaltet sein, dass sie für einen Menschen, ohne körperliche Einschränkungen, umsetzbar sind.

2.3.2 Funktionen der Benutzeroberfläche

/G0001/
Die Benutzeroberfläche teilt das Programm in drei Phasen. Phase eins ist die Vorbereitung des evolutionären Algorithmus, Phase zwei ist der evolutionäre Algorithmus und Phase drei ist die Präsentation des Ergebnisses des evolutionären Algorithmus.
/G0001a/
In Phase eins werden Parameter für die Initialisierung des evolutionären Algorithmus festgelegt. Diese umfassen Instrumente und Techniken, die benutzt werden sollen, sowie Hyperparameter des evolutionären Algorithmus (Populationsgröße, Länge der Genotypen, etc.).
/G0001b/
In Phase zwei werden die Elemente aus **/E0001a/** ermöglicht.
/G0001c/
In Phase drei werden die ausgewählten Rhythmen graphisch ausgegeben und die akustische Ausgabe ermöglicht.

/G0002/
Die Ausgabe der Individuen (Phänotyp) wird graphisch als Notenbild und akustisch ermöglicht.

/G0003/

Die Anordnung der Individuen ermöglicht es dem Nutzer seine Präferenzen zu erkennen.

/G0004/
Die Bewertung der Individuen wird möglichst einfach gestaltet.

2.3.3 Funktionen der Dokumentation

/D0001/
Die Dokumentation ermöglicht es die Entstehung der Individuen nachzuvollziehen.
/D0001a/
Ein Stammbaum ermöglicht eine globale, grobe Ansicht der Entwicklung über die Zeit.
/D0001b/
Eine Ansicht zu einem Knotenpunkt im Stammbaum ermöglicht das lokale Nachvollziehen der Anwendung eines Abwandlungsoperators.

/D0002/
Es wird eine Statistik zu dem evolutionären Algorithmus als Übersicht erstellt.

2.3.4 sonstige Funktionen

/S0001/
Es wird die Möglichkeit zur Verfügung gestellt die Arbeit zu unterbrechen und zu einem beliebigen Zeitpunkt fortzusetzen.

/S0002/
Es werden Instrumente zur Verfügung gestellt, die als Grundlage genutzt werden können.

/S0003/
Es wird die Möglichkeit geschaffen, Arbeiten zu individualisieren, indem eigene Instrumente hinzugefügt werden können.

/S0004/
Es wird ermöglicht, Hyperparameter zum evolutionären Algorithmus während des Programmablaufs zu verändern.

3 Grundlagen

In diesem Kapitel werden die technischen und fachlichen Grundlagen beschrieben, die für diese Arbeit benötigt werden. Die Grundlagen zu evolutionären Algorithmen wurden hauptsächlich aus den Quellen [Tak01], [BHS07], [AEE15] und [Wei15] bezogen.
Es wird zuerst auf Begriffe eingegangen, die im Zusammenhang mit evolutionären Algorithmen genutzt werden. Anschließend werden die Besonderheiten von Fitness und Selektion erläutert. Danach wird die genutzte Ausprägungsform von evolutionären Algorithmen, genetische Algorithmen, kurz vorgestellt. Der darauf folgende Abschnitt stellt dann die Besonderheiten von interaktiven evolutionären Algorithmen vor.

3.1 Begriffe zu evolutionären Algorithmen

Evolutionäre Algorithmen dienen der Suche nach Lösungen nach dem Vorbild der natürlichen Evolution und sind somit Teil des Themengebiets der künstlichen Evolution.
Nachfolgend werden Begriffe erläutert, die zur Fachsprache des Themengebiets gehören.
Die Suche nach Lösungen mittels evolutionärer Algorithmen funktioniert im allgemeinen so, dass Lösungsvorschläge gemacht werden. In Anlehnung an die natürliche Evolution werden diese Lösungsvorschläge Individuen genannt. Die Menge der aktuell existierenden Individuen wird als Population bezeichnet. Dabei bilden Individuen, die innerhalb eines definierten Zeitabschnitts zur Population dazukommen eine Generation.
Für Individuen gibt es zwei verschiedene Erscheinungsformen. Die eine ist der Phänotyp. Dabei handelt es sich um die Ausprägung des Individuums, wie es in seiner Umgebung vorkommt. Die andere Erscheinungsform ist der Genotyp. Dabei handelt es sich um die Kodierung des Individuums. Diese ist die Arbeitsgrundlage für den Algorithmus, zur Erschaffung neuer besserer Individuen.
Der Genotyp eines Individuums besteht aus Genen, deren Ausprägungsformen als Allele bezeichnet werden [Wei15]. “Die Gesamtheit aller Allele in einer Population wird auch als Genpool bezeichnet.”[Wei15].
Um das Ziel neue bessere Individuen zu erzeugen, erreichen zu können, gibt es Bewer-

tungsfunktionen (Fitnessfunktionen), die auf den Phänotyp eines Individuums angewendet werden. Dadurch wird die Güte eines Individuums bestimmt, im Kontext evolutionärer Algorithmen wird diese Fitness genannt. Die Fitness eines Individuums hat Einfluss darauf, ob das Individuum zur Erzeugung der nächsten Generation verwendet wird. Die Auswahl der Individuen zur Erzeugung der nächsten Generation wird als Selektion bezeichnet. Die ausgewählten Individuen der aktuellen Generation werden als Eltern bezeichnet.
Die Erzeugung der nächsten Generation basiert auf Abwandlungsoperatoren ("Variation Operators" [AEE15, Kapitel 3.2.5]), von denen es zwei verschiedene gibt. Die durch diese Operatoren entstehenden Individuen werden Nachkommen, oder Kinder, genannt.
Die Mutation ist eine Operation, die auf dem Genotyp eines Individuums aus der Menge der Eltern ausgeführt wird. Dabei handelt es sich um eine kleine, zufällige Änderung.
Die Rekombination, oder auch Crossover genannt, ist ein Operator, bei dem mindestens zwei Genotypen aus der Menge der Eltern miteinander gemischt werden. Dadurch entstehen dann, ein bis Anzahl der verwendeten Eltern, Nachkommen. Üblicherweise werden für die Rekombination zwei Eltern benutzt.
Die Abbildung 3.1 zeigt einen Schritt eines evolutionären Algorithmus beispielhaft mit Farben im RGB-Farbraum.

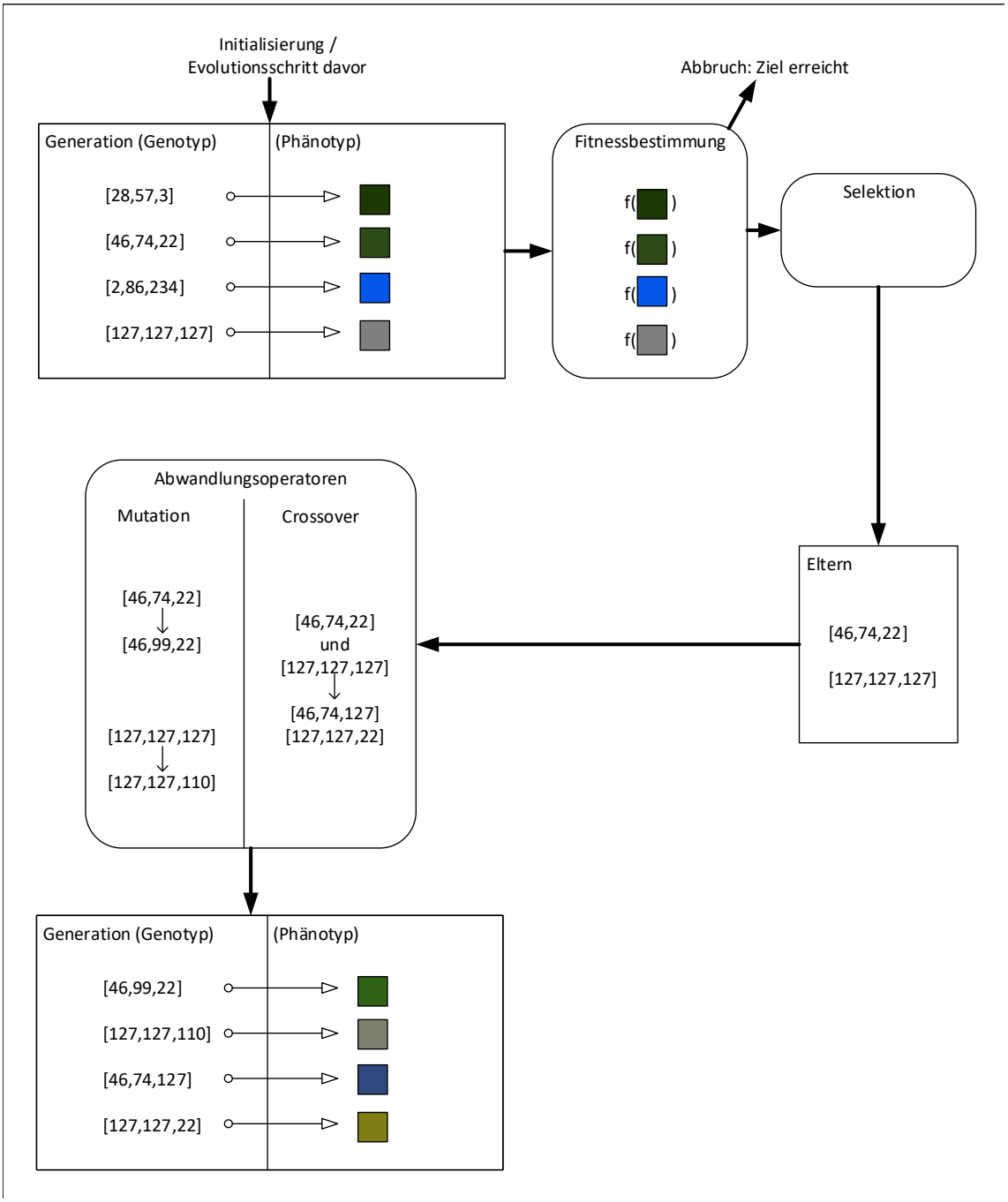

Abbildung 3.1: Darstellung eines Schritts in einem evolutionären Algorithmus mit Beispiel im RGB Farbraum. Jedes Individuum hat 3 Gene, die Allele sind die Zahlen von 0 bis 255.

3.2 Fitness

Die Güte eines Individuums wird durch die Fitness angegeben. Folgende drei Arten der Fitness werden in [BHS07] unterschieden:

- berechenbare Fitness: Die Fitness wird aus den Eigenschaften des Individuums berechnet.
- interaktive Fitness: Die Fitness wird durch ein Lebewesen bestimmt.
- implizite Fitness: Die Fitness wird durch das Zurechtkommen des Individuums in einer vorgegebenen Umgebung bestimmt.

Des Weiteren kann es vorkommen, dass ein Individuum mehrere voneinander unabhängige Anforderungen erfüllen muss um eine hohe Güte zu haben. In diesem Fall ist es möglich, die Fitness des Individuums über folgende Methoden zu bestimmen [BHS07]:

- zufällige Auswahl einer Fitnessbewertung
- Bildung der gewichteten Summe aller Fitnessbewertungen
- Pareto-Optimierung: Individuen Ränge zuordnen, nach dem Prinzip, dass ein Individuum, das in allen Fitnessbewertungen gleich und in einer Bewertung besser ist als die anderen einen höheren Rang hat.

3.3 Selektion

Die Selektion ist der Prozess, bei dem die Individuen ausgewählt werden, die ihre Gene weitervererben dürfen.
Folgende Algorithmen werden in [BHS07] genannt:

- Roulette-Selektion: Die Wahrscheinlichkeit eines Individuums ausgewählt zu werden entspricht dem prozentualen Anteil an der Gesamtfitness der Generation
- Stochastic Universal Sampling: Auswahl von m Individuen durch äquidistante Abstände der auszuwählenden Individuen und zufälliger Wahl des Startpunkts. Abbildung 3.2 zeigt den Unterschied zwischen Roulette-Selektion und Stochastic Universal Sampling.
- rangbasierte Selektion: Ordnung der Individuen nach Rang und anschießend wird die Position in der Liste zur Auswahl verwendet
- Tunierselektion: Es wird eine Tuniergruppe mit k Individuen geformt. Das beste Individuum der Tuniergruppe wird ausgewählt
- $(\mu + \lambda)$ / (μ, λ)-Strategie: μ steht für die Anzahl der Eltern, λ steht für die Anzahl der Nachkommen. Bei der "+"-Strategie werden aus den Eltern und Nachkommen die besten ausgewählt, bei der ","-Strategie wird nur aus den Nachkommen ausgewählt.

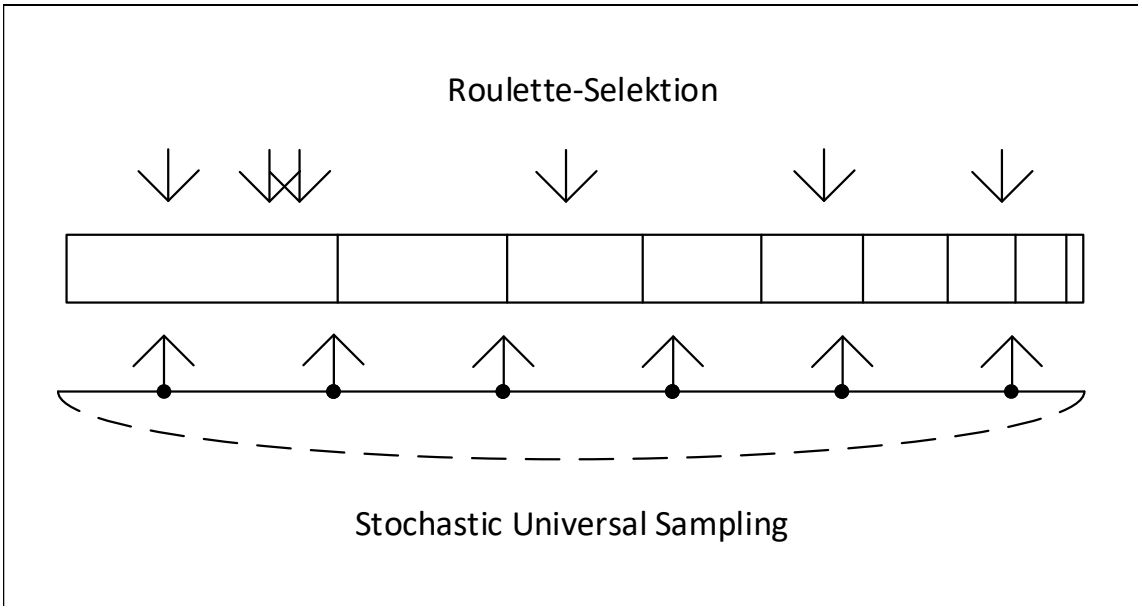

Abbildung 3.2: Der Streifen in der Mitte repräsentiert die Individuen. Jeder Abschnitt darin entspricht dem prozentualen Anteil eines Individuums an der Gesamtfitness. Oben ist eine mögliche Auswahl der Individuen mittels Roulette-Selektion zu sehen. Unten ist das Verfahren Stochastic Universal Sampling im Vergleich zu sehen. Mit beiden Methoden werden hier sechs Individuen gewählt.

Zusätzlich zu den Auswahlverfahren kann man auch die Elitestrategie anwenden, wodurch das beste Individuum garantiert in der nächsten Generation dabei ist.

3.4 Genetische Algorithmen

Bei der Anwendung genetischer Algorithmen werden Individuen als Bitvektoren oder Parametervektoren fester Länge kodiert. Das Ende des Algorithmus ist erreicht, wenn eine Mindestqualität erreicht wurde, die maximale Anzahl an Generationen erzeugt wurde oder wenn keine Verbesserungen mehr erreicht werden.
Je nach Kodierung und Problemstellung müssen die Umsetzungen der genetischen Operationen geeignet gewählt werden. Dadurch wird sichergestellt, dass keine ungültigen Lösungen erzeugt werden können.
In Kapitel 4 des Buches [AEE15] werden die Zusammenhänge zwischen der Kodierung der Individuen und den Abwandlungsoperationen beschrieben.
Für Individuen, die als Bitvektoren kodiert sind wird die Mutation so umgesetzt, dass eine Mutationswahrscheinlichkeit bestimmt wird mit der ein Bit gekippt wird. Bei dieser Operation wird diese Wahrscheinlichkeit für jedes Bit angewendet. Die Mutationswahrscheinlichkeit wird gering gewählt, um nur kleine Veränderungen zu erzeugen.

Auf der Ebene von Parametervektor basierten Individuen kann die Mutation auf die gleiche Weise erfolgen, mit dem Unterschied, dass der ersetzende Wert aus einer Menge von zulässigen Werten zufällig ausgewählt wird. Diese Operation wird in [AEE15] als “Random Resetting” (zufälliges Ersetzen) als Operation für Integer-Repräsentationen genannt.
Crossover-Operationen kommen bei Bitvektor kodierten Individuen in drei Standardformen vor [AEE15]. Abbildung 3.3 veranschaulicht die Crossover-Operationen.
Eine der Formen ist der Ein-Punkt-Crossover. Dieser funktioniert so, dass von zwei Individuen die ersten n Bits genommen werden und die übrigen Bits zwischen beiden Individuen getauscht werden. Die Individuen werden also an der Stelle zwischen der Position n und $n + 1$ getrennt. Auf diese Weise entstehen zwei neue Individuen mit dem Anfang des einen Elternteils und dem Ende des anderen Elternteils.
Eine weitere Form, von der der Ein-Punkt-Crossover eine Spezialform ist, ist der n-Punkt-Crossover. Dabei werden zwei Eltern genommen und n Schnittstellen bestimmt. Die entstehenden Teile werden dann abwechselnd getauscht und nicht getauscht.
Die dritte Form ist das uniforme Crossover. Dabei werden keine Schnittstellen zwischen den Individuen gesucht, sondern wie bei der Mutation wird mit Hilfe einer Wahrscheinlichkeit festgelegt, ob ein Bit zwischen den beiden Individuen getauscht werden soll.
Diese Crossover-Operationen lassen sich auf die gleiche Weise auch auf Individuen anwenden, die als Parametervektoren kodiert sind.

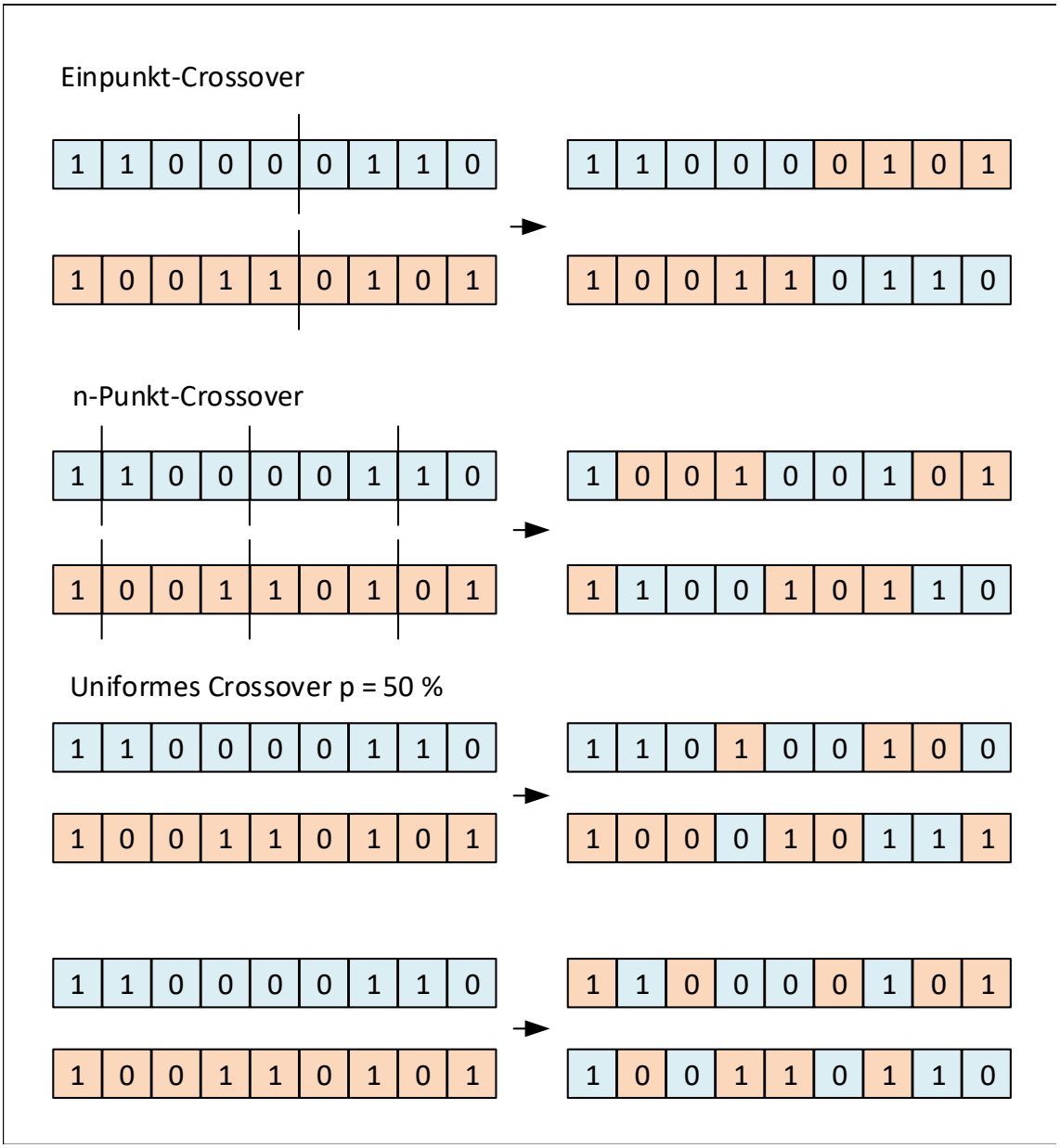

Abbildung 3.3: In Anlehnung an die Abbildungen in Abschnitt 4.2 aus [AEE15]. Abbildung zur Darstellung der 3 Standard-Crossoveroperationen für Bitvektor kodierte Individuen. Die Farben markieren die Zugehörigkeit zu dem Elternindividuum. Zwei Beispiele wurden für universelles Crossover angegeben um den Unterschied zu n-Punkt-Crossover zu verdeutlichen. Nämlich, dass bei gleichem Wahrscheinlichkeitswert unterschiedlich viele Trennstellen entstehen können.

Für Kodierungen, bei denen es darum geht, eine Reihenfolge zu finden, Permutations-Kodierungen genannt, gibt es Mutations- und Crossover-Operationen, bei denen darauf geachtet wird, dass keine Werte verschwinden.

Die folgenden Mutationsoperationen werden für Individuen in Permutations-Kodierung in

[AEE15] vorgestellt.

Die “Swap Mutation” (Tausch Mutation) funktioniert so, dass zwei Gene des Individuums getauscht werden. Die “Insert Mutation” funktioniert so, dass zwei Gene bestimmt werden und das zweite Gen wird direkt nach dem ersten Gen angefügt. Die anderen Gene werden entsprechend verschoben. Die “Scramble Mutation” (Gerangel Mutation) funktioniert so, dass zufällig Gene des Individuums ausgewählt werden und dann durcheinandergebracht werden.

Crossover-Operationen für Individuen in Permutations-Kodierung müssen so konzipiert sein, dass beim Austausch der Gene die Werte jeweils in der gleichen Anzahl, wie zuvor vorkommen. Eine Operation, die das gewährleistet ist der “Partially Mapped Crossover” (PMX-Crossover). Dieser funktioniert so, dass zwei Trennstellen zufällig ausgewählt werden. Dann wird das mittlere Stück zwischen den Individuen ausgetauscht. Da die beiden entstandenen Individuen gegebenenfalls nicht mehr gültig sind, da sich die Anzahl pro Wert geändert hat werden Ersetzungsregeln, durch Vergleich der Ersetzungen der Werte beim Austausch der Mittelstücke, erstellt. Die Ersetzungsregeln werden dann angewendet, um aus den Kinderindividuen wieder gültige Lösungsvorschläge zu erzeugen.

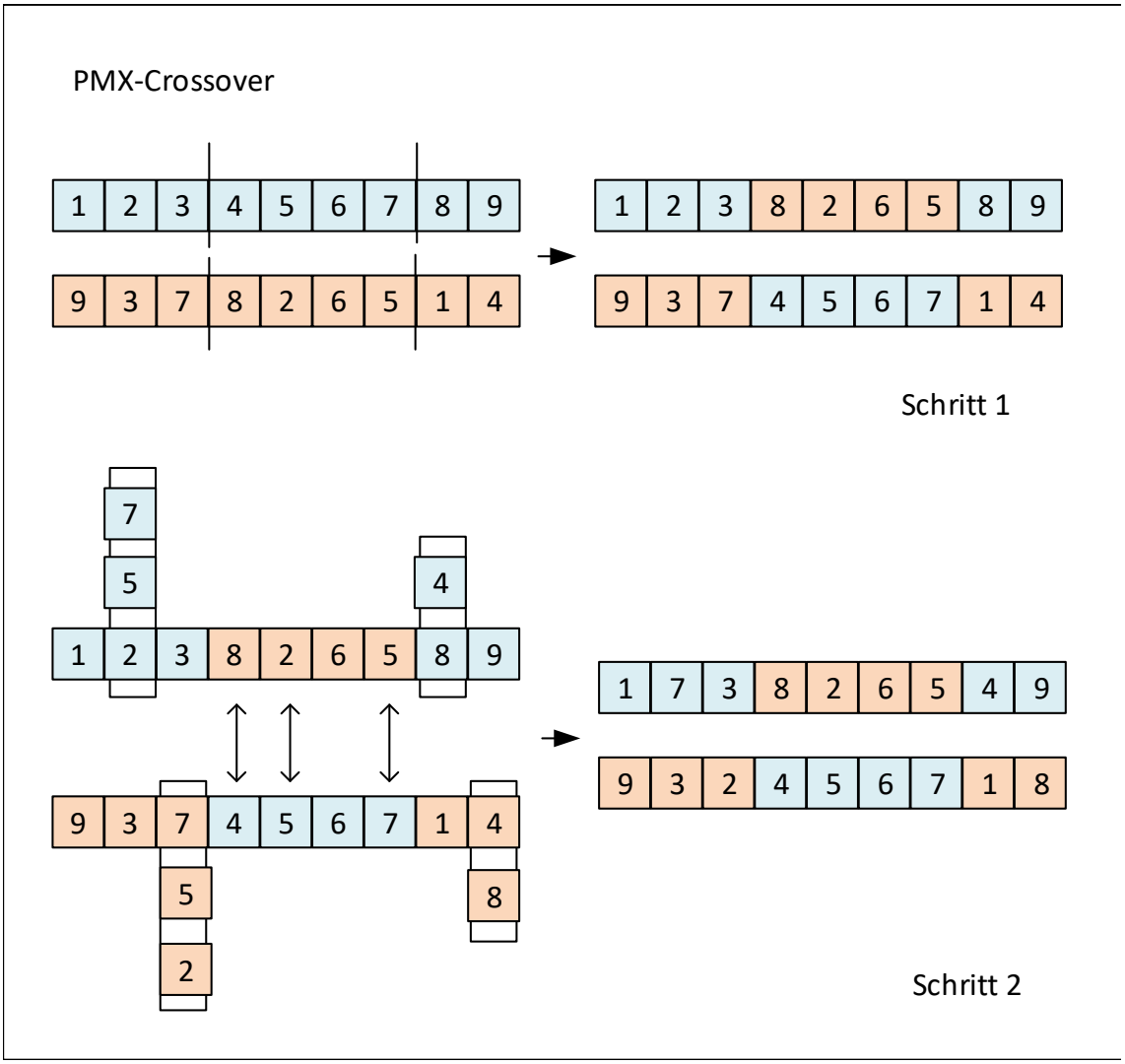

Abbildung 3.4: In Anlehnung an das Beispiel aus Abschnitt 4.5.2 aus [AEE15]. Beispiel für die PMX-Crossover-Operation. Oben ist Schritt 1 zu sehen, wobei ungültige Individuen entstehen. Unten ist Schritt 2, die Korrektur zu sehen. Bei der Korrektur werden die Ersetzungsregeln erzeugt (Pfeile zwischen den direkt ausgetauschten Teilen) und dann auf die nicht ausgetauschten Teile angewendet (Balken mit Zahlen an den entsprechenden Stellen). Die Pfeile sind so zu verstehen: Das Auftreten einer 8 wird im ersten Individuum durch eine 4 ersetzt. Das Auftreten einer 2 wird im ersten Individuum durch eine 5 ersetzt. Das Auftreten einer 5 wird im ersten Individuum durch eine 7 ersetzt. Für das zweite Individuum gelten die Regeln andersherum.

3.5 Interaktive evolutionäre Algorithmen

Bei interaktiven evolutionären Algorithmen wird der Mensch in den Zyklus der evolutionären Algorithmen mit einbezogen. Dabei werden Zielsysteme basierend auf einer abbildenden Beziehung zwischen den Eigenschaftsparametern und psychologischem Raum optimiert [Tak01].

Die Einbeziehung des Menschen ist dann notwendig, wenn es schwierig oder unmöglich ist eine Bewertung der Individuen zu berechnen. Darum werden interaktive evolutionäre Algorithmen oft im Kontext der Kunst, Mode und Musik eingesetzt. Der alternative Ansatz zu evolutionären Algorithmen ist ein Modell eines Nutzers zu erstellen und dadurch die Bewertungen durchführen zu lassen. Dieser Ansatz wird in [Tak01] als analytischer Ansatz bezeichnet.
Verglichen mit einem evolutionären Algorithmus ist das globale Optimum eines interaktiven evolutionären Algorithmus kein Punkt, sondern eine Fläche, da es unterschiedliche System-Ausgaben geben kann, die von Nutzern nicht unterschieden werden können und damit psychologisch gleich sind [Tak01].

3.5.1 Einflussnahme des Menschen

In der Arbeit [Tak01] wird die Einflussnahme des Menschen als Grundlage der Definition von interaktiven evolutionären Algorithmen benutzt.
In der engen Fassung der Definition ist ein interaktiver evolutionärer Algorithmus ein evolutionärer Algorithmus, bei dem die Fitnesswerte für die Individuen von einem Menschen festgelegt werden.
In der erweiterten Fassung der Definition ist ein interaktiver evolutionärer Algorithmus ein evolutionärer Algorithmus, bei dem es eine Mensch-Maschine-Schnittstelle zur Optimierung der Individuen gibt.

3.5.2 Probleme

In [Tak01] werden drei Hauptprobleme von interaktiven evolutionären Algorithmen beschrieben.
Problem eins ist die Nutzerermüdung. Ein Nutzer kann verglichen mit einer Maschine nur sehr wenige Bewertungen vornehmen. Das Bewerten ist ein Prozess, der viel Aufmerksamkeit verlangt und dadurch sehr anstrengend ist für den Nutzer.
Das zweite Problem ist, dass die Suche nach Problemlösungen nur mit kleinen Populationen und wenigen Generationen möglich ist. Die kleinen Populationen resultieren daraus, dass ein Mensch weniger Dinge in einem geringeren Tempo als Maschinen erfassen und vergleichen kann.
Das dritte Problem ist herauszufinden, wie man zeitsequenzielle Individuen mit weniger Müdigkeit und weniger Arbeitszeit bewerten kann.
Die gute Nachricht ist, dass interaktive evolutionäre Algorithmen nicht viele Generationen

benötigen, um zufriedenstellende Ergebnisse zu erzielen [Tak01].

3.5.3 Nutzeroberfläche

Die Müdigkeit wird stark dadurch beeinflusst, wie einfach es ist Individuen eines interaktiven evolutionären Algorithmus zu bewerten und Rückmeldungen an den evolutionären Algorithmus zu geben [Tak01]. Daraus resultiert, das die Mensch-Maschine-Schnittstelle ein wichtiges Element für die Verringerung der Nutzerermüdung ist. Diese sollte so konzipiert sein, dass ein Nutzer möglichst so lange mit dem Programm arbeitet, bis eine ihn zufriedenstellende Lösung erreicht ist.
In der Studienarbeit [Dre11] sind ein paar Konzepte beschrieben, wie Benutzeroberflächen zur Bewertung aufgebaut sind. Dabei bezieht sich die Arbeit [Dre11] auf ausgewählte Beispiele. In [Dre11] wird zwischen klassischen Modellen unterschieden, die in Kacheln angeordnet sind, wobei in jeder Kachel ein Individuum und die entsprechenden Bewertungselemente sind, und innovative Ansätze, welche auf der Positionierung oder Positionsänderung der Individuen basieren. Außerdem wird das paarweise Vergleichsmodell beschrieben, wobei man sich immer zwischen zwei Individuen entscheidet, welches besser ist.
Des Weiteren werden Konzepte für eine einfache Benutzeroberfläche gezeigt. Folgende Konzepte für einfache Benutzeroberflächen werden in [Dre11] erläutert:

- Anzahl der zu bewertenden Individuen gering halten. (Doppelbewertungen und ungültige Individuen vermeiden)
- Bewertung mit möglichst wenig Interaktionen
- nur Phänotyp darstellen (nur bewertungsrelevante Teile), Genotyp bleibt dem Nutzer verborgen
- KISS[1]-Prinzip einhalten
- abwechslungsreicher Evaluationsprozess
- jederzeit erkennbares Ziel
- Anzahl der Bewertungsstufen gering halten

3.5.4 Lösungsansätze zur Benutzerermüdung

In [Tak01] wird auf verschiedene Möglichkeiten eingegangen, wie die Ermüdung des Benutzers reduziert werden kann.

[1] KISS: Keep it simple and stupid

Eine Möglichkeit ist die Vorhersage der Fitnesswerte. Auf diese Weise kann man die Populationsgröße, die bei interaktiven evolutionären Algorithmen verglichen mit evolutionären Algorithmen ohne Interaktion durch die Fähigkeiten des Menschen stark begrenzt ist, größer auswählen. Um den Nutzer nicht zu überlasten, kann man diesem dann die Individuen präsentieren, für die eine hohe Fitness vorhergesagt wird.
Eine weitere Möglichkeit sind Varianten, die die Nutzerschnittstelle dynamisch machen. In [Tak01] sind für Arbeiten mit akustisch wahrnehmbaren Individuen Vorschläge enthalten, wie zum Beispiel, dass man verschiedene Soundquellen benutzt zur Wiedergabe der Individuen. Eine höhere Dynamik der Nutzerschnittstelle kann auch dadurch gegeben sein, wenn man dem Nutzer jederzeit das beste Individuum markiert und zum Abspielen bereit hält. Dadurch kann jederzeit der Vergleich zum besten Individuum gewährleistet werden.
Eine Untersuchung zur Dynamik in [Tak01] vergleicht die Effektivität davon, den Nutzer frei entscheiden zu lassen, welches Individuum er hören möchte, und der Vorgabe, welches Individuum gehört werden soll. Es wird nach subjektiven Tests gezeigt, dass es effektiver ist, dem Nutzer die Reihenfolge vorzugeben, in der die Individuen abgespielt werden, statt ihn selbst entscheiden zu lassen.
Bei längeren akustisch wahrnehmbaren Individuen von großer Länge wird die Echtzeit Bewertung vorgeschlagen, nach dem Vorbild des Projekts “GenJam”, das in [Tak01] als Beispiel dient. Dabei wird die Bewertung eines Individuums Stück für Stück vorgenommen.
Als weitere Möglichkeit die Nutzerermüdung zu reduzieren wird vorgeschlagen aktive Eingriffe in den evolutionären Algorithmus zu erlauben. Dabei kann man zum Beispiel dem Nutzer erlauben bestimmte Gene mit Werten zu belegen.
Außerdem kann man den Nutzer aktiv Eingreifen lassen, indem man den Suchraum visualisiert. Dabei hat der Nutzer dann die Möglichkeit grob einzuschätzen, wo sich optimale Individuen befinden können.

3.6 fachliche Grundlagen

Das Fachgebiet dieser Anwendung ist Rhythmik mit Schlaginstrumenten. In diesem Abschnitt werden die notwendigen musikalischen Konzepte für diese Arbeit erläutert.

3.6.1 Instrumente

Die Nutzung von Schlaginstrumenten kann auf verschiedene Weisen erfolgen. Sie können mit den Händen direkt gespielt werden oder mit Hilfsmitteln. Als Hilfsmittel werden die für

Schlagzeuger üblichen Mittel verstanden, im wesentlichen Schlagzeugstöcke, üblicherweise Sticks genannt, und Fußmaschinen. Von beiden gibt es unterschiedliche Ausführungen, die den Klang und den Spielkomfort des Schlagzeugers beeinflussen, in dieser Arbeit wird davon ausgegangen, dass ein Schlagzeuger nur gewöhnliche Sticks und einfache Fußmaschinen für Basstrommel und Hi-Hat hat. Diese Hilfsmittel werden in Abbildung 3.5 gezeigt.

Abbildung 3.5: Standardbedienelemente mit denen Schlaginstrumente eines Schlagzeugs benutzt werden können. Auf dem linken Bild sieht man ein Einzel-Pedal an einer Kuhglocke im Vordergrund und eine Hi-Hat-Fußmaschine im Hintergrund. Die Hi-Hat-Fußmaschine ist mit einer Stange verbunden, die bewirkt, dass das obere Becken der Hi-Hat nach unten geht, wenn man das Pedal tritt. Auf dem rechten Bild sieht man einfache Sticks.

Schlaginstrumente können in zwei Kategorien nach der Klangerzeugung unterschieden werden. Die eine Kategorie sind Fellklinger (Membranophone). Das sind die Instrumente, bei deren Tonerzeugung eine Membran genutzt wird. Die andere Kategorie sind Selbstklinger (Idiophone). Das sind Instrumente, bei denen der Klang durch die Schwingung des Instru-

mentkörpers selbst erzeugt wird.

Als Grundlage für diese Arbeit wird ein Standardschlagzeug definiert, das aus den Komponenten besteht, die in Tabelle 3.1 den Kategorien der Klangerzeugung zugeordnet sind. Abbildung 3.6 zeigt zur Verdeutlichung, wie das Standardschlagzeug aufgebaut sein kann.

Tabelle 3.1: Die Tabelle zeigt die Instrumente, die in dieser Arbeit das Standardschlagzeug bilden mit Einordnung in die entsprechende Kategorie der Klangerzeugung und Zuordnung des Hilfmittels, das zur Nutzung eingesetzt wird.

Instrument	**Kategorie**	**Bedienhilfsmittel**
Bass-Trommel	Membranophone	Fußmaschine
Snare	Membranophone	Sticks
kleine Tom	Membranophone	Sticks
mittlere Tom	Membranophone	Sticks
große Tom	Membranophone	Sticks
Hi-Hat	Idiophone	Fußmaschine / Sticks
Crash Becken	Idiophone	Sticks
Ride Becken	Idiophone	Sticks
Splash Becken	Idiophone	Sticks
Kuhglocke	Idiophone	Fußmaschine

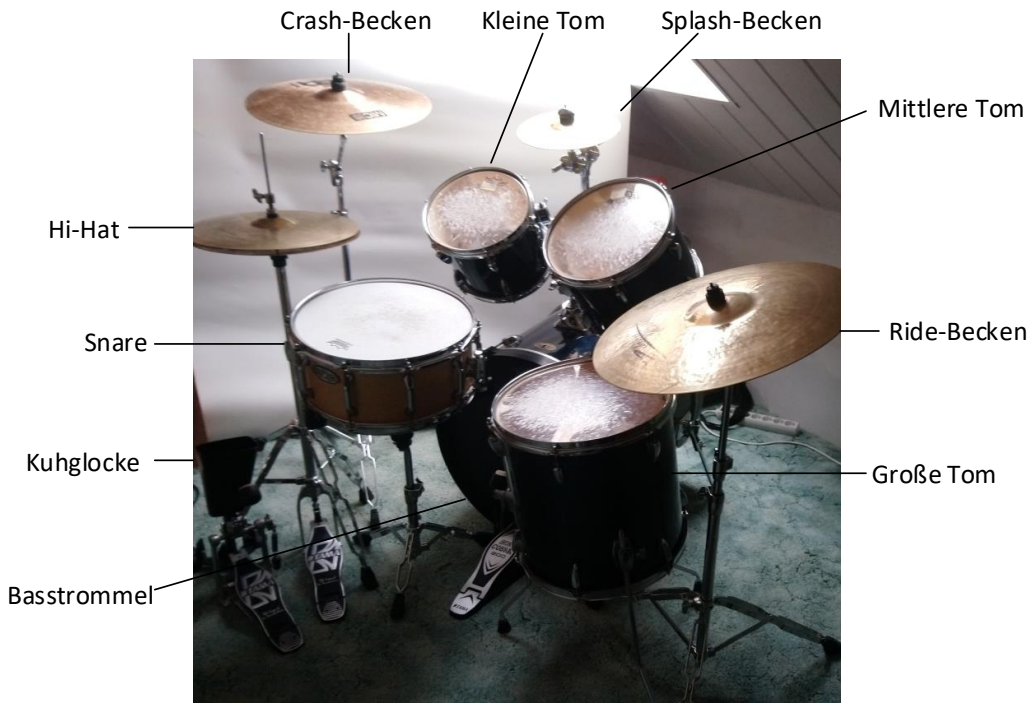

Abbildung 3.6: Abbildung und Benennung der Elemente, die das Standardschlagzeug ausmachen in einer gängigen Anordnung der Instrumente.

Die Nutzung der Instrumente kann mit verschiedenen Techniken erfolgen. In dieser Arbeit werden Techniken betrachtet, die Auswirkungen auf den Klang haben. Die Nutzung einer Technik ist auch davon abhängig, mit welcher Art Bedienhilfsmittel man ein Instrument nutzt. Die Bedienhilfsmittel für die Instrumente des definierten Standardschlagzeugs sind in Tabelle 3.1 eingetragen. Nachfolgend werden Techniken erläutert, die zum Grundrepertoire dieser Arbeit gehören.

Eine Technik ist der einfache Schlag. Dabei wird ein Instrument mit einem Stick einmal geschlagen oder die Fußmaschine an einem Instrument einmal getreten.

Eine weitere Technik ist der Flame. Dabei wird ein einfacher Schlag mit einem leiseren einfachen Schlag, der kurz vorher erfolgt kombiniert. Ein Flame erzeugt einen wuchtigeren Klang, als ein einfacher Schlag.

Ein Sidestick (oft auch Cross-Stick) ist eine Technik, bei der ein Stick quer über eine Trommel gelegt wird und dann mit dem Ende das über den Rand der Trommel geht auf den Rand der Trommel geschlagen wird. Zur Veranschaulichung zeigt Abbildung 3.7, wie die Handhaltung bei einem Sidestick ist.

Abbildung 3.7: Diese Abbildung zeigt, wie ein Sidestick gespielt wird.

Ein Rimshot ist eine Technik, bei der eine Trommel mit einem Stick so geschlagen wird, dass dabei der Rand der Trommel und das Fell getroffen werden.
Für die Hi-Hat gibt es abhängig vom Zustand des Instruments verschiedene Techniken. Wenn die Hi-Hat offen ist, dann kann man einen einfachen Schlag mit einem Stick ausführen oder durch das Treten der Fußmaschine. Im geschlossenen Zustand kann man einen einfachen Schlag mit einem Stick ausführen, wobei sich der Klang von dem einfachen Schlag im offenen Zustand unterscheidet.

3.6.2 Notensystem und Taktart

Das für diese Arbeit relevante Notensystem ist das binäre Notensystem. Dieses hat die Eigenschaft, dass die Dauer eines Notenwertes zwei Notenwerten des nächst kleineren Notenwertes entspricht. Die Abbildung 3.8 zeigt den Aufbau der Notenpyramide bis zur Sechzehntelnote.

Beispielsweise hat eine ganze Note die gleiche Dauer, wie zwei halbe Noten. Der Unterschied ist, dass die Note dann aber zweimal gespielt wird.

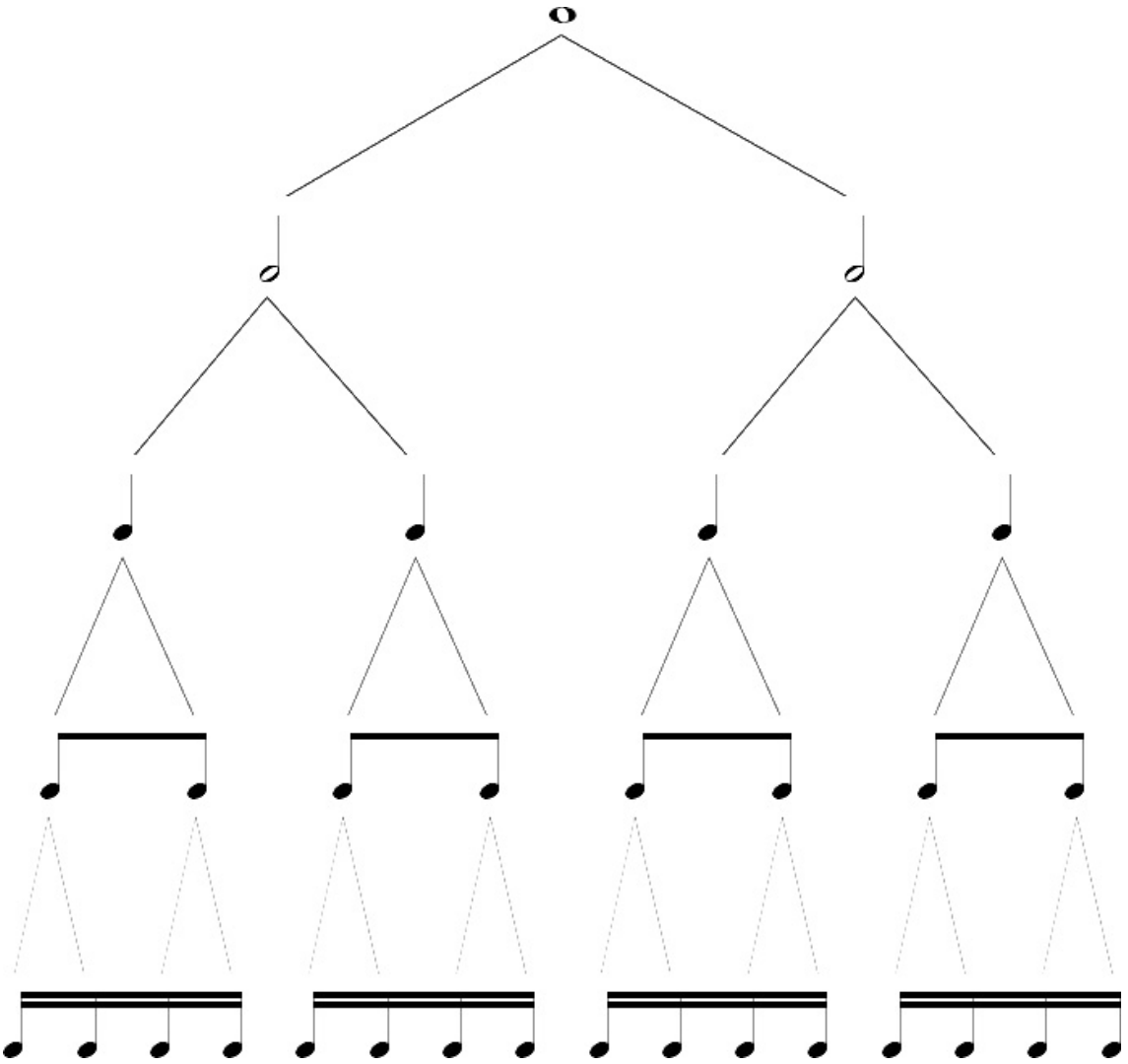

Abbildung 3.8: Abbildung aus [Rhy]. Die Rhythmuspyramide zeigt die Notenwerte. Dabei sieht man, wie viele Noten eines Wertes benötigt werden, um einen anderen Wert darzustellen.

Die Besonderheit bei Schlaginstrumenten ist, dass die Dauer des Klangs meistens nicht berücksichtigt wird. Hier ist es meistens nur relevant, wie oft Instrumente in dem entsprechenden Zeitintervall gespielt werden sollen.
Um Noten übersichtlich darzustellen, werden diese in Takten gruppiert. Die Taktart ist dabei als Bruch angegeben, wobei der Nenner aussagt, welcher Notenwert die Grundlage für den Takt bildet und der Zähler aussagt, wie viele Noten des Notenwertes einen Takt bilden. Pro Zähler wird auch von einer Zählzeit gesprochen.

In dieser Arbeit wird als kleinster Notenwert die Sechzehntelnote verwendet und die Taktarten basieren auf Viertelnoten. Das bedeutet, dass 4 Sechzehntelnoten pro Zählzeit benutzt werden können.

3.6.3 Geschwindigkeit

Die Geschwindigkeitsangabe erfolgt in bpm ("Beats per minute"). Als Beat wird in dieser Arbeit die Zählzeit genutzt. Die Angabe 60 bpm würde bedeuten, dass man 60 Zählzeiten in einer Minute hat, also 60 Viertelnoten, was 240 Sechzehntelnoten pro Minute entspricht. Umgerechnet auf 4/4-Takte sind das 15 Stück in einer Minute.

4 Ähnliche Arbeiten

Dieses Kapitel referenziert Arbeiten, die Inspiration zur Umsetzung des Projekts lieferten. Das Kapitel ist in die Teilaspekte gegliedert, zu denen Arbeiten gefunden wurden. Arbeiten zu den betreffenden Teilaspekten werden kurz vorgestellt und anschließend eingeschätzt in Bezug auf den Wert für die eigene Arbeit.

4.1 Thematisch ähnliche Arbeiten

In der Arbeit [YNOF11] wurden mittels interaktiven genetischem Algorithmus Fill-Ins[1] zu einem Standardrhythmus erzeugt. Dazu standen sieben Sounds, inklusive kein Sound, zur Verfügung, mit deren Hilfe eine Abfolge von acht Sechzehntelnoten erzeugt wurde. Der Genotyp wurde durch eine Abfolge von acht Zahlen von null bis sechs dargestellt. Die Präsentation erfolgte mittels MIDI-Ausgabe über Kopfhörer. Die Bewerter durften mit Zahlen von eins bis sieben bewerten, wobei eins bedeutet, dass das Fill-In nicht gefällt und sieben, dass es sehr gut gefällt.

Bei der Arbeit [YNOF11] sind Individuen lediglich Abfolgen von acht einzeln gespielten Instrumenten, bzw. Pausen. Das vereinfacht die Darstellung, mindert allerdings auch die Vielfalt der möglichen Lösungen, wodurch schneller der Eindruck entstehen kann, dass gleiche Individuen erzeugt werden. Denn durch die Linearität[2] kann es auf subjektiver Ebene dazu kommen, dass verschiedene Positionen in Individuen nur binär unterschieden werden, nach Schlag und kein Schlag.
Das Problem ist hier zu stark vereinfacht worden. Besser ist es, wenn die Möglichkeit berücksichtigt wird, dass mehrere Instrumente gleichzeitig bedient werden können. Denn durch das gleichzeitige Spielen verschiedener Instrumente entstehen auch neue subjektive Eindrücke. Beispielsweise können zwei Instrumente zusammen harmonisch wirken.

[1] Ein Fill-In ist eine Unterbrechung des aktuell gespielten Rhythmus, durch ein abweichendes Muster.
[2] Lineariät bedeutet beim Schlagzeugspielen, dass Instrumente nicht gleichzeitig gespielt werden.

4.2 Nutzerschnittstelle

Die Arbeit [MI18] befasst sich mit der Komposition von Musik mit Hilfe von interaktiven evolutionären Algorithmen in Verbindung mit Google's Magenta[3] Projekt. Die GUI für die Auswahl der Individuen wurde angelehnt an einen Einkaufsablauf implementiert. Es werden anfangs alle Individuen als runde Symbole in einem neutralen Bereich angezeigt. Die Bewertung erfolgt durch das Ziehen der Symbole in einen entsprechenden Bereich.
In einem ersten Schritt werden die guten in den Einkaufswagen gelegt und die anderen aussortiert. In einem zweiten Schritt wird der Einkaufswagen sortiert, nach kaufen und nur im Wagen behalten.

Die Nutzer-Bewertung ist in zwei zwei-Optionen Entscheidungen unterteilt, was die Bewertung für den Nutzer einfacher macht [MI18].
Das Abspielen der Musikstücke erfolgt durch Anklicken eines Individuums. Abbildung 4.1 zeigt die GUI des Projekts.

[3] https://magenta.tensorflow.org

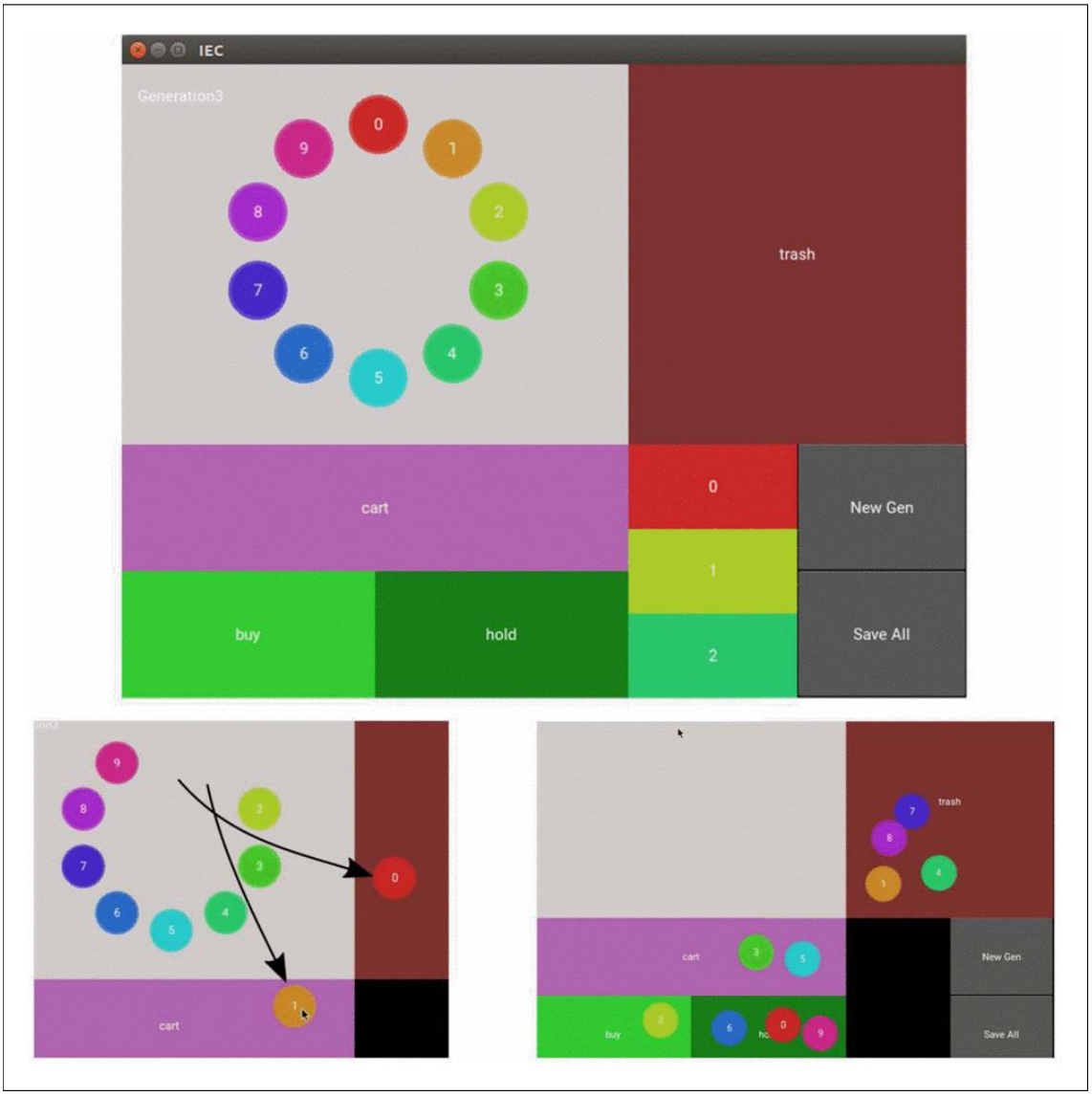

Abbildung 4.1: Fig. 2 aus [MI18]. GUI in Anlehnung an den Ablauf beim Einkaufen. Oben: Anfangszustand; Unten links: Schritt 1 Einkaufswagen oder weg; Unten rechts: Schritt 2 Sortierung innerhalb des Einkaufswagens

Die Arbeit [SI11] befasst sich damit mit Hilfe von interaktiven genetischen Algorithmen Musikempfehlungen zu machen. Dabei wird die Tendenz des Nutzers visuell in einem kubischen 2D Raum dargestellt.

Es macht Freude, wenn die Tendenz der Musikauswahl visualisiert wird [SI11]. Eine solche Visualisierung hilft dem Nutzer seine Tendenzen zu verstehen und erleichtert die Musikauswahl [SI11].

Die Individuen werden als Symbole in einem Streudiagramm dargestellt. Die Individuen haben verschiedene Eigenschaften, von denen der Nutzer sich die Eigenschaften aussuchen kann, die an der X-Achse und an der Y-Achse angezeigt werden sollen. Es gibt vier verschiedene

Farben, die die Bewertung der Individuen repräsentieren. Abbildung 4.2 zeigt die Darstellung des Raums.

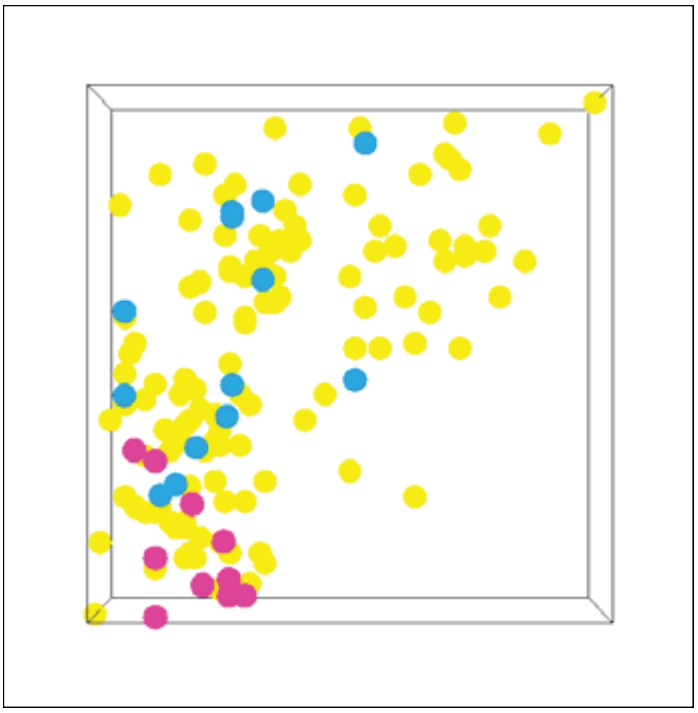

Abbildung 4.2: Ausschnitt von Figure 2 aus [SI11]. Darstellung des Raums in MusiCube. Die Farben bedeuten alle unterschiedliche Zustände der Musikstücke. (rot: gehört und positiv bewertet; blau: gehört und negativ bewertet; gelb: Ausgangszustand)

In der Arbeit [HT00] werden die Möglichkeiten von visualisierten interaktiven Algorithmen besprochen und ein Experiment mit selbstorganisierenden Karten vorgestellt. In Abbildung 4.3 wird dargestellt, wie sich ein visualisierter interaktiver Algorithmus von einem interaktiven Algorithmus unterscheidet.

Bei der Visualisierung wird der Suchraum im zweidimensionalen Raum dargestellt. Menschen haben die Fähigkeit eine gesamte Verteilung von Individuen im zweidimensionalen Raum auf makroskopischer Ebene zu erkennen, was nicht durch einen evolutionären Algorithmus interpretiert werden kann [HT00]. Beim visualisierten interaktiven Algorithmus nimmt der Nutzer die Rolle des Bewerters ein und wählt bessere Individuen als Kandidaten aus [HT00]. In dieser Arbeit wird ein Experiment gemacht, bei dem selbstorganisierende Karten zur Abbildung der Individuen aus einem n-dimensionalen Raum in den zweidimensionalen Raum benutzt werden. Dabei wurde festgestellt, dass es viel Zeit in Anspruch nimmt, wenn man die selbstorganisierenden Karten nach jeder Generation neu trainieren muss.

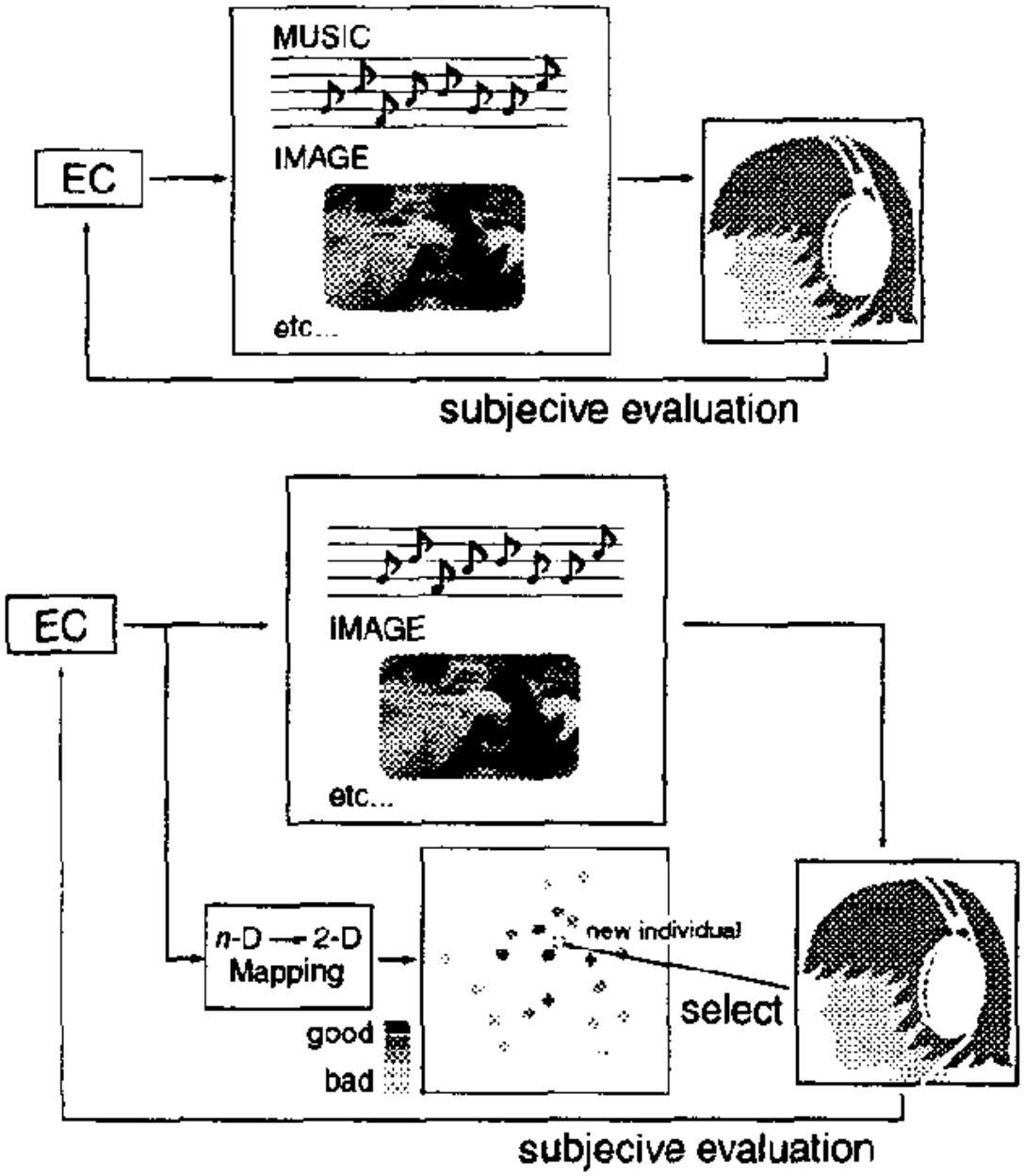

Abbildung 4.3: Figure 2 aus [HT00]. Darstellung eines interaktiven Algorithmus (oben) verglichen mit einem visualisierten interaktiven Algorithmus (unten). Dabei ist erkennbar, dass der Nutzer in der visualisierten Variante die weitere Rolle besitzt Individuen auszuwählen.

Die in der Arbeit [MI18] vorgestellte GUI bietet eine einfache Variante der Bewertung. Das Aufteilen in zwei Phasen der Bewertung erscheint allerdings überflüssig zu sein. So wie Individuen direkt im Müll landen können, kann man sich ebenfalls direkt entscheiden, ob man etwas kauft, oder im Warenkorb behalten möchte.

Der Ablauf hat in [MI18] einen etwas monotonen Charakter, da der Zyklus immer gleich zu sein scheint:

- Individuen in der neutralen Zone
- Bewertung Phase 1
- Bewertung Phase 2
- Start von vorne (neue Generation) oder abbrechen

Die Darstellung in [SI11] hat das interaktive Element, dass man die Ansicht wechseln kann, indem man die Eigenschaften an den Achsen ändert. Außerdem bleibt der Fortschritt immer erhalten. Bei dem Erstellen der nächsten Generation verschwinden die Punkte nicht, die die Individuen älterer Generationen darstellen. Allerdings kann es hier sein, dass Punkte die Individuen darstellen durch andere verdeckt werden. In Abbildung 4.2 ist das deutlich zu sehen.
Die in Idee aus [HT00] den Suchraum im zweidimensionalen Raum darzustellen mit Hilfe von selbstorganisierenden Karten hat gegenüber der Visualisierung aus [SI11] den Nachteil, dass der Nutzer die Räume selber interpretieren muss, während bei Streudiagrammen mit definierter x- und y-Achse diese Interpretation abgenommen wird. Hinzukommt, dass der Nutzer eine dynamischere Benutzeroberfläche hat, wenn er die Parameter der Darstellung ändern kann und dass selbstorganisierende Karten einen Trainingsprozess benötigen, der die Phase der Interaktion hinauszögert. Der Vorteil der selbstorganisierenden Karten wäre, dass diese alle Attribute in die Berechnung der Position mit einbeziehen.

4.3 Nutzerermüdung

Da es sich um ein zeitsequenzielles Problem handelt, ist die Länge der Individuen von Bedeutung. In der Arbeit [MI18] wurde die Länge der Individuen auf zwei Takte festgelegt, unter Berücksichtigung dessen, dass die Schwierigkeit steigt, je länger Musikstücke sind.
In der Arbeit [NO14] wird ein System vorgestellt, das dabei helfen soll Videos zusammenzufassen, wobei ein Video entsteht, das die Zusammenfassung darstellt. Ein Genotyp besteht hier aus einer Reihenfolge von Videoclips.
Dem Nutzer wird dabei erlaubt selbst die Reihenfolge der Videoclips zu verändern, durch hinzufügen, löschen und Reihenfolge ändern. Außerdem bewertet der Nutzer die Videos qualitativ. Bei einem Video das unbewertet bleibt, wird die qualitative Bewertung durch eine Ähnlichkeitsfunktion berechnet. Die Fitness ist allerdings ebenfalls von einer quantitativen Fitness abhängig, die durch das System bestimmt wird.

Die Länge der Individuen auf maximal zwei Takte zu beschränken, wie in [MI18], ist sowohl

aus Sicht der Nutzerermüdung, als auch im Kontext der Anwendung sinnvoll. Rhythmen bestehen gewöhnlich aus kurzen sich wiederholenden Teilen.
Die Interaktion mit dem Programm, wie in [NO14] ist ebenfalls eine sehr gute Idee. Wenn der Nutzer einen Rhythmus findet, der ihm gut gefällt, aber ggf. Kleinigkeiten störend sind, sollte er diese auch manuell ändern können, statt darauf warten zu müssen den entsprechenden Rhythmus vorgeschlagen zu bekommen. Dadurch kann die Zufriedenheit des Nutzers und damit die Akzeptanz des Programms erhöht werden.
Die Bewertung der Rhythmen teils von objektiven Eigenschaften abhängig zu machen, oder die Benutzerwertung zu schätzen kann dazu führen, dass sich der Nutzer übergangen fühlt und somit aus dem Prozess der Entwicklung ausgegrenzt fühlt. Es kann der Eindruck entstehen, dass die Entscheidungen des Nutzers nicht mehr wichtig sind.

4.4 Hyperparameter genetischer Algorithmus

In der Arbeit [VFC15] wird versucht neue Melodien durch genetische Algorithmen zu erzeugen. Die Grundlage dieser Arbeit wird durch die Musiktheorie gebildet.
Die Startgeneration wird hier durch kurze Phrasen erzeugt, die durch Experten validiert wurden. Dadurch soll garantiert werden, dass die Melodien, die entstehen musikalische Kriterien erfüllen. Des Weiteren sind die Abwandlungsoperationen nicht zufällig, sondern intelligent implementiert. Das bedeutet, dass diese auf Grundlage von Methoden der Musiktheorie implementiert sind. Dadurch bleibt sichergestellt, dass die musikalischen Kriterien erfüllt bleiben.

Das Erzeugen von Melodien, wie in [VFC15], ist ein komplexeres Problem, als Schlagzeugrhythmen zu erzeugen, da eine Melodie ebenfalls einen Rhythmus hat und dabei noch weitere Kriterien erfüllen muss. Des Weiteren unterscheidet sich die Arbeit [VFC15] von dieser Arbeit in dem Punkt, dass es kein interaktiver genetischer Algorithmus ist.
Es scheint eine gute Idee zu sein, die Startgeneration nicht komplett zufällig zu erzeugen, um die Suche nach neuen Rhythmen effektiver zu gestalten. Dafür können Muster in die Rhythmen integriert werden. Gegebenenfalls vertraute Muster in der Startgeneration wiederzufinden, kann die Ermüdung des Nutzers verringern, weil schneller Rhythmen erzeugt werden können, die dem Nutzer gefallen können.
Auf das Einführen von intelligenten Abwandlungsoperationen nach Grundlagen der Musiktheorie wird allerdings verzichtet, um die Möglichkeiten dessen, was vorgeschlagen wird nicht zu limitieren. Diese Limitierung wird in der Arbeit [VFC15] ebenfalls als nicht gut

bewertet.

4.5 Eigenschaften von Rhythmen

In der Arbeit [NHJ15] werden Wege für die Berechnung des Abstands zwischen verschiedenen Rhythmen vorgestellt. Das Ziel der Arbeit ist es mit Hilfe eines genetischen Algorithmus ähnliche Rhythmen zu einem vorgegebenen Rhythmus zu erstellen. Dabei wird auf die "edit distance" eingegangen, die die Anzahl der nötigen Änderungen zählt, um von einer Zeichenkette zu einer anderen zu kommen. Die "Hamming distance" wird als die Anzahl der Positionen beschrieben, die sich unterscheiden. Um diese Distanzen Anwenden zu können wird vorgeschlagen einen Rhythmus als Bit-String zu sehen, bei dem die Positionen der Instrumente hintereinander angereiht werden.
Die Arbeit [KFV13] stellt 40 Eigenschaften zur Beschreibung von Rhythmen vor. Diese Eigenschaften basieren auf der Annahme, dass die Rhythmen nur mit drei Instrumenten (Hi-Hat, Basstrommel und Snare) erzeugt werden. Außerdem werden die Instrumente unterschieden in starker Einfluss auf den Rhythmus und schwacher Einfluss auf den Rhythmus. Die Instrumente mit starken Einfluss auf den Rhythmus sind die Snare und die Basstrommel. Den kleinen Einfluss hat die Hi-Hat, diese ist eher für den Puls zuständig. Die Lautstärke eines Instruments kann in sechs Stufen variiert werden, oder das Element ist nicht gesetzt. Die Eigenschaften sind im wesentlichen auf die Dichte, Synkopierung, Symmetrie und das Verhältnis zwischen starken und schwachen Einfluss zurückzuführen, sowie Abweichungen und Mittelwerte der Lautstärken.

Die in [NHJ15] eingeführte Idee der Abstandsberechnung ist sehr interessant, da man damit eine Mutationsoperation implementieren kann, die auf der Ähnlichkeit der einzelnen Allele beruht. Die Betrachtung des Genotyps als Bit-String, der hintereinander die Positionen der Instrumente darstellt ist in dieser Arbeit nicht geeignet. Dabei könnte es vorkommen, das bei Mutationen physisch für Menschen nicht umsetzbare Rhythmen entstehen.
Die Betrachtung der Instrumente in [KFV13] in schwachen Einfluss und starken Einfluss, ist interessant, zur Erzeugung einer initialen Population, die auf Mustern basiert. Dabei kann man definieren, an welchen Stellen ein schwacher oder ein starker Einfluss gesetzt sein muss, um ein entsprechendes Muster in einem Rhythmus integriert zu bekommen. Die Unterscheidung nach Einfluss kann noch präziser erfolgen. In [KFV13] waren die Instrumente mit dem starken Einfluss die Snare und die Basstrommel. Diese kann man nach Tonhöhe unterscheiden. Die Snare wäre dabei das hohe Instrument und die Basstrommel wäre das

tiefe Instrument. Die Instrumente mit schwachen Einfluss haben den Charakter, den Puls darzustellen, oder auch den Beat zu erhalten.
Des Weiteren sind einige Eigenschaften aus [KFV13] in ähnlicher Form für dieses Projekt nutzbar. Allerdings scheint die Anzahl 40 etwas zu hoch zu sein. Durch eine so hohe Anzahl an Eigenschaften kann sich ein Nutzer schnell überfordert fühlen.

4.6 Nutzerfeedback

In der Arbeit [MI18] gab es die Nutzerrückmeldung, dass das Endergebnis schlechter war, als die dritte oder vierte Generation.

Diese Rückmeldung wirft die Frage auf, warum ein nutzbares Ergebnis erst nach einer festgelegten Anzahl von Wiederholungen gegeben werden soll. Es erscheint sinnvoll zu sein dem Nutzer zu ermöglichen Ergebnisse aus jeder Generation als mögliches Ergebnis verfügbar zu machen.

In der Arbeit [YNOF11] gab es die Rückmeldung, dass mit fortschreitenden Generationen, gleiche Individuen erzeugt wurden.

Eine solche Erfahrung kann für Nutzer frustrierend sein und den Sinn ihrer Tätigkeit infrage stellen. In der Arbeit [YNOF11] wurde mit festgelegten Parametern für den evolutionären Algorithmus gearbeitet, die Mitwirkung des Nutzers bestand ausschließlich in der Fitnessbestimmung der Individuen. Es kann sinnvoll sein dem Nutzer Einfluss auf die Parameter der Mutation und des Crossovers zu gewähren, damit der Prozess mehr Interaktivität bekommt und der Nutzer mehr Einfluss auf die Ergebnisse hat.

In der Arbeit [NHJ15] gab es die Rückmeldung, dass Rhythmen, bei denen erkennbar ist, dass diese sich aus anderen entwickelt haben durch kleine Änderungen sehr interessant sind, im Vergleich zu Rhythmen, die total anders sind. Allerdings darf die Änderung auch nicht zu klein sein.

Diese Rückmeldung gibt Anlass, eine geeignete Mutationsoperation zu implementieren. Das Maß der Veränderung zu finden ist dabei das Problem, da die Veränderung nicht zu gering ausfallen darf, um interessant zu sein.

5 Konzeption

In diesem Kapitel wird auf die Entwicklungsmethoden und Planung des Projekts eingegangen.

5.1 Überlegungen

1. Die Bezeichnung der Erscheinungsform eines Individuums ist abhängig von der Betrachtung. Der Genotyp ist die Bauanleitung für den Phänotyp eines Individuums. Zur Erzeugung des Phänotyps braucht es einen Übersetzer, der den Genotyp in den Phänotyp übersetzt. Das bedeutet, dass der Genotyp in diesem Projekt die Objekte sind, die durch den Kompiler erzeugt werden. Der Phänotyp kann das Audiosignal, das durch das Objekt zur Wiedergabe erzeugt wird, sein, aber auch ein Notenbild, das durch das Objekt zur graphischen Ausgabe erzeugt wird.
Aus einer anderen Betrachtungsweise kann man die Ergebnisse dieses Programms als Genotyp interpretieren. Der Nutzer wäre dann der Übersetzer, wenn er versucht nach dem Notenbild oder dem Audiosignal den Rhythmus zu erzeugen. Der Phänotyp wäre dabei das Audiosignal, das entsteht, wenn der Nutzer übt.
2. Dieses Projekt ist dafür gedacht, kreativ mit einem Menschen zusammenzuarbeiten. Das bedeutet, dass dieses Programm nicht konvergiert, dass es also keinen Punkt gibt, an dem ein Maximum erreicht werden kann. Die Ergebnisse können die entsprechenden Rhythmen darstellen, die der Nutzer am Ende umsetzt oder auch nur Ideen zur Umsetzung von Rhythmen darstellen, also nur die Grundlage dessen bilden, was der Nutzer als neuen Rhythmus lernt. Das gängige Paradigma, dass man eine bestimmte Anzahl von Generationen durchgeht und bewertet oder den Prozess der Lösungssuche endgültig abschließt, wenn man mit einer Lösung zufrieden ist, soll in dieser Arbeit nicht zur Anwendung kommen. Stattdessen kann man die Suche unterbrechen, mit seinen Ergebnissen arbeiten und zu einem späteren Zeitpunkt fortfahren.
Es ist also nicht zielführend Lerner zu trainieren, die die Präferenzen des Nutzers lernen und damit indirekt den Nutzer steuern, sondern besser dem Nutzer einen hohen Freiheitsgrad einzuräumen, damit er das Programm und damit die künstliche Evolution steuert.

Die Freiheit des Nutzers ist allerdings auch in seinem Sinne zu begrenzen, um zu Vermeiden dass Frust aufkommt, oder der Nutzer sehr schnell ermüdet. Als Beispiel soll hier die Generationsgröße dienen. Diese wird auf Werte im Intervall [3,15] festgelegt. Die untere Grenze ist drei, da sonst zu wenig Individuen für die Selektion in der nächsten Generation bereit stünden. Die Obergrenze ist 15, damit der Nutzer sich nicht mit zu vielen neuen Rhythmen auf einmal konfrontiert sieht und die Nutzerermüdung nicht schnell ansteigt. Die Generationsgröße soll allerdings auch jederzeit im Programm änderbar sein, sodass der Nutzer für spätere Generationen den Wert verändern kann.
Die Umsetzung des Projekts erfolgt daher nach der erweiterten Definition aus Abschnitt 3.5.1.

5.2 Programmiersprache

Als Programmiersprache für dieses Projekt wird Java eingesetzt. Der Vorteil dabei ist, die Plattformunabhängigkeit. Des Weiteren kommen keine fremden Bibliotheken für evolutionäre Algorithmen zum Einsatz, da diese oft nicht für interaktive evolutionäre Algorithmen konzipiert sind und außerdem immer ein hohes Maß an Anpassung erfordern, da die Kodierung der Genotypen stark vom Anwendungsfall abhängig ist.

5.3 Entwicklungsvorgehen

Die Entwicklung des Projekts erfolgt dynamisch und versionsbasiert. Das bedeutet, dass jeder Entwicklungsschritt eine neue Version des Java-Projekts entspricht. Die Änderungen zur vorherigen Version sind in den Kommentaren der Klasse mit der main-Methode festgehalten. Jede Version des Projekts wird manuell selbst getestet, sodass die Lauffähigkeit der Versionen gewährleistet bleibt. Des Weiteren werden Tests von einem fachlichen Anwender, ohne technischen Hintergrund, durchgeführt, wodurch Feedback zu der Gebrauchstauglichkeit entsteht. Auf diese Weise werden auch Probleme an der Nutzerobefläche schnell gefunden, wodurch die Gebrauchstauglichkeit erhöht werden kann und Störfaktoren, die zu einer schnelleren Ermüdung führen erkannt werden können.
Das Ziel bleibt dabei weiterhin die Erfüllung der Anforderungen, die im Abschnitt 2.3 definiert wurden.

5.4 Speicherbarkeit und Individualisierbarkeit

Zur Umsetzung der Anforderungen zur Speicherbarkeit und Individualisierbarkeit, bietet sich das Konzept des Workspaces an. Ein Workspace entspricht dabei einer Suche nach neuen Rhythmen.
Dabei ist der Ablauf so definiert, dass bei Programmstart abgefragt wird, mit welchem Workspace man arbeiten möchte. Sofern dieser Workspace nicht existiert, wird ein leerer neuer Workspace erstellt. In dem Workspace werden sämtliche Daten bezüglich einer Suche nach Rhythmen in Verzeichnissen strukturiert gespeichert.
Das Konzept des Workspaces bietet auch den Vorteil, dass man die Ergebnisse seiner Arbeit leicht mit anderen austauschen kann. Dazu muss man nur den Workspace kopieren.

5.5 Individuenrepräsentation und Lösungsraum

Die Repräsentation der Individuen ist eine Abfolge von gleichzeitig spielbaren Schlaginstrumenten und Pausen. Dabei haben alle Individuen die gleiche feste Länge.
Ein Allel wird in diesem Programm definiert als eine Kombination gleichzeitig spielbarer Instrumente. Das Allel, das keine Instrumente enthält repräsentiert die Pause.
Als kleinster Notenwert wird die Sechzehntelnote genommen. Die Länge der Genotypen (s) ergibt sich aus der Anzahl der Takte (aT), dem Zähler bei der Taktart (Ta) und dem Faktor 4 für eine Auflösung als Sechzehntelnoten.

$$s = aT \cdot Ta \cdot 4$$

Den Lösungsraum, die Anzahl der möglichen verschiedenen Individuen (n), zu berechnen ist ein Problem, das als Variation mit Wiederholung klassifizierbar ist. Zur Lösung wird die Länge der Individuen und die Anzahl der möglichen Allele (a) benötigt. Die Formel dafür lautet:

$$n = a^s$$

5.6 Programmablauf

Dieser Abschnitt beschreibt den Ablauf des Programms.

1. Der Nutzer erzeugt oder wählt einen bestehenden Workspace

1a. Der Nutzer wählt die Parameter für die Rhythmen aus.

2. Das Programm erzeugt oder lädt alle Allele aus denen die Rhythmen aufgebaut werden können.

3. Das Programm erzeugt eine Startgeneration oder lädt die gespeicherten Generationen.

4. Der Nutzer bewertet die Individuen der aktuellen Generation, erzeugt eigene Individuen, guckt sich den Stammbaum an, ändert Parameter oder guckt sich die Statistik an, oder Beendet das Programm.

5. Das Programm führt Abwandlungsoperationen aus und erzeugt dadurch eine neue Generation.

6. Wiederholung der Schritte 4. und 5. bis der Nutzer in Schritt 4. abbricht.

7. Ausgabe des Ergebnisses in Form von geschriebenen Noten und Möglichkeit des Hörens.

8. Ende des Programms oder Beginn von 1.

Abbildung 5.1 stellt den Programmablauf zur Verdeutlichung der aufgezählten Schritte dar.

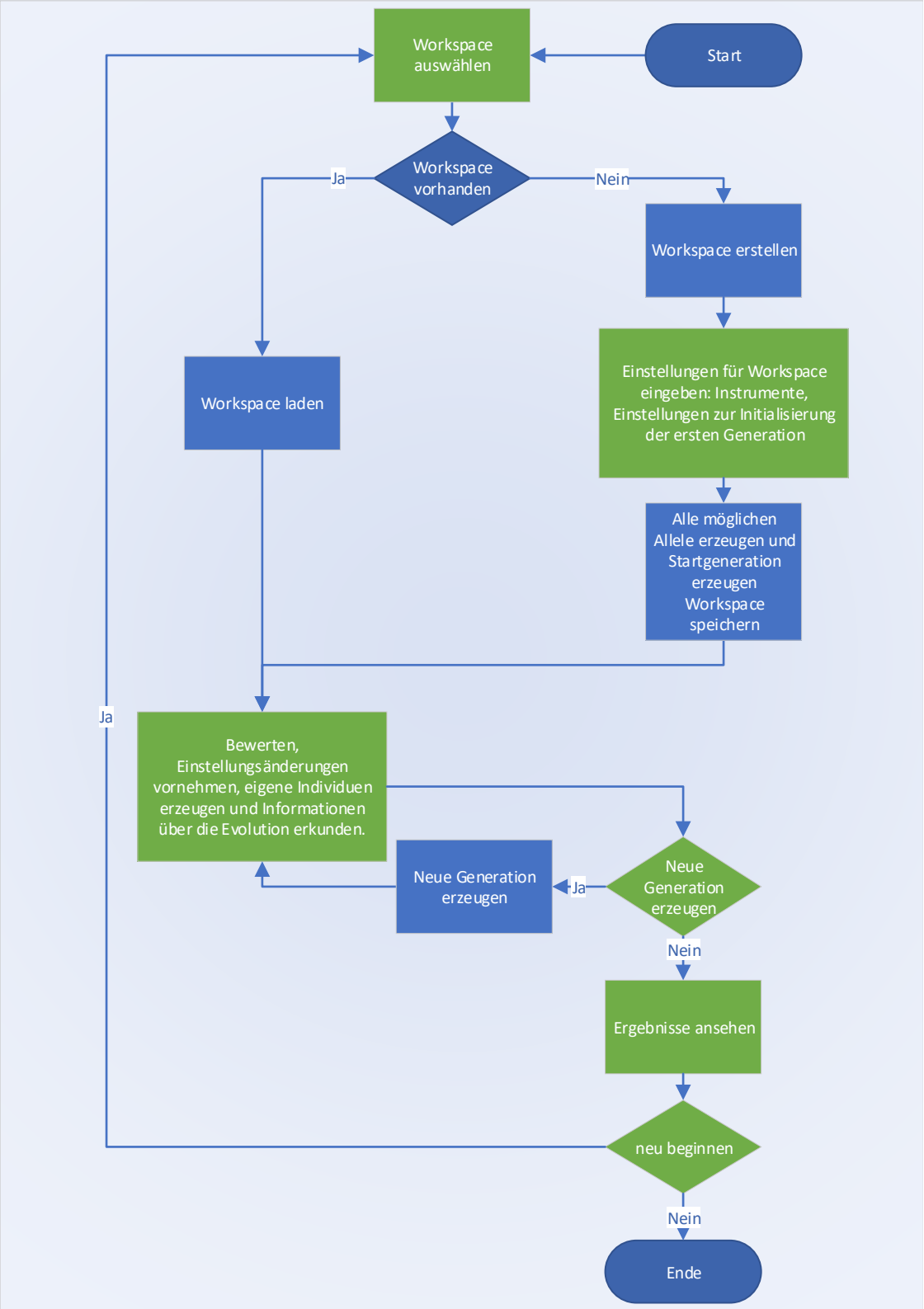

Abbildung 5.1: Skizze, die den Ablauf darstellt. Die blauen Rechtecke zeigen Prozesse und Entscheidungen durch das Programm. Die grünen Rechtecke zeigen Prozesse, bei denen der Nutzer eine GUI bedient.

5.7 Nutzerschnittstelle

Die Benutzerschnittstelle ist eingeteilt in das Rahmenkonstrukt, das für die Vor- und Nachbereitung des evolutionären Algorithmus benutzt wird und der Evolutionsnutzerschnittstelle, das die Steuerung des evolutionären Algorithmus ermöglicht.

5.7.1 Regeln für die Benutzerschnittstelle

Die in Abschnitt 3.5.3 beschriebenen Regeln sollen umgesetzt werden. Des Weiteren wird darauf geachtet, dass die Benutzeroberfläche der Nutzergruppe angepasst sein muss. Für einen Schlagzeuger kann nicht vorausgesetzt werden, dass er die Fachsprache zu evolutionären Algorithmen kennt. Außerdem sollte die Variabilität seiner Einstellungsmöglichkeiten eingeschränkt werden, damit der Fokus auf sein Ziel nicht verloren geht.

5.7.2 Rahmenkonstrukt

Im ersten Schritt muss der Benutzer Parameter für die Initialisierung des evolutionären Algorithmus festlegen. Dafür wird ein Fenster zur Festlegung der folgenden Parameter angezeigt.

- Anzahl der Takte (1 oder 2)
- Taktart ($\frac{3}{4}$, $\frac{4}{4}$, $\frac{5}{4}$)
- Instrumente
- Techniken
- Generationsgröße ([3,15])
- Prozentsatz der Rhythmen, die nach Mustern erzeugt werden ([0,100])

Abbildung 5.2 zeigt eine Skizze des Fensters, zur Einstellung der Parameter für die Initialisierung des evolutionären Algorithmus.

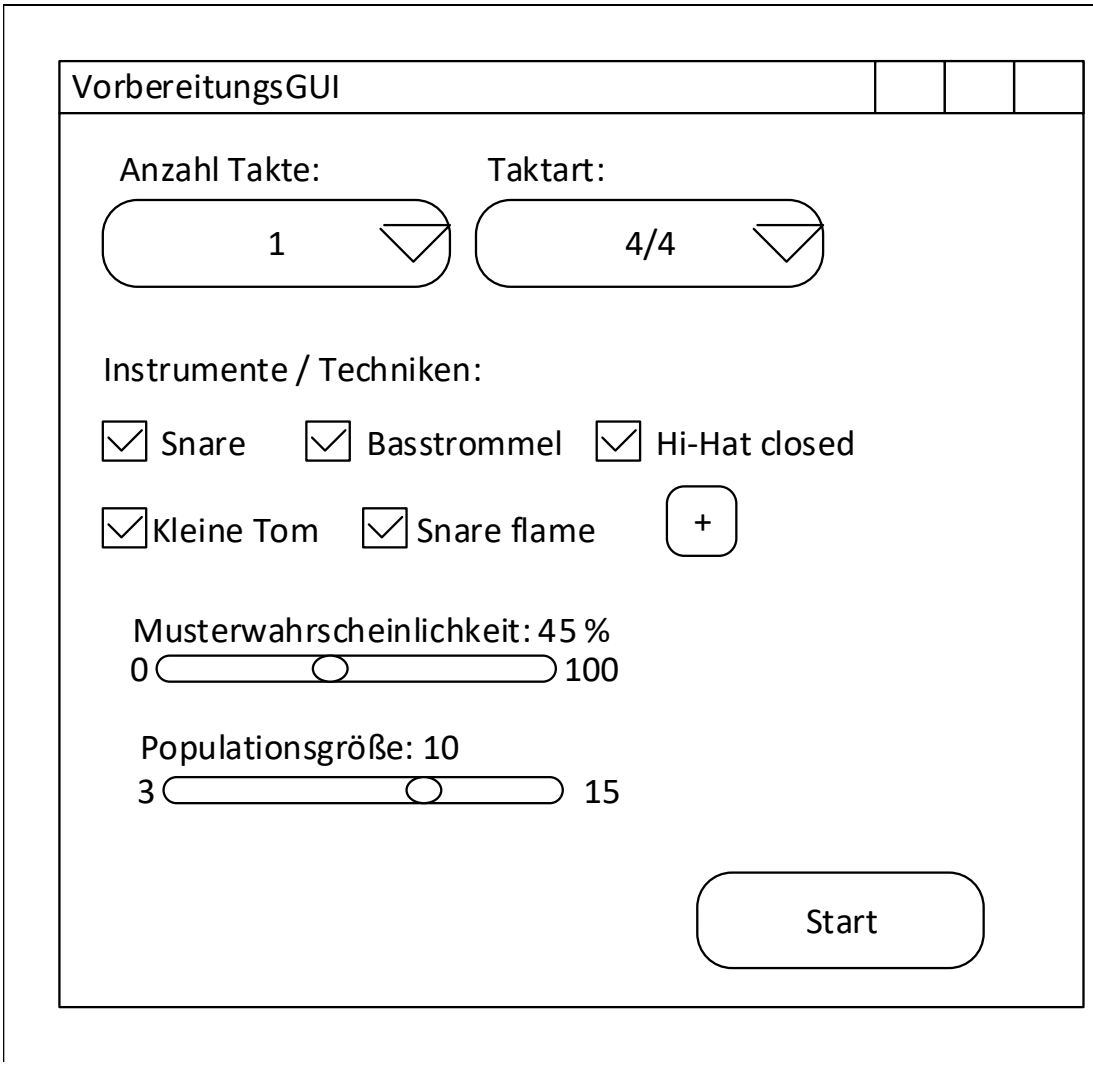

Abbildung 5.2: Skizze zur VorbereitungsGUI. Diese deutet alle Elemente an, die benötigt werden. Die Festlegung wird mit Auswahlbuttons (Anzahl Takte und Taktart), Auswahlboxen (Techniken und Instrumente) und Schiebereglern (Musterwahrscheinlichkeit und Populationsgröße) getroffen. Des Weiteren existiert ein Button (+) zum Hinzufügen neuer Instrumente und Techniken und der Button zum Starten des evolutionären Algorithmus.

Nach der Festlegung der Parameter startet der evolutionäre Algorithmus und die Evolutionsnutzerschnittstelle wird angezeigt.

Nach Abschluss des evolutionären Algorithmus werden die erzeugten Rhythmen für den Nutzer angezeigt. Dafür gibt es ein Fenster, das dem Nutzer die Ausgabe als Audiosignal und als Notenbild ermöglicht. Abbildung 5.3 zeigt eine Skizze zum abschließenden Fenster.

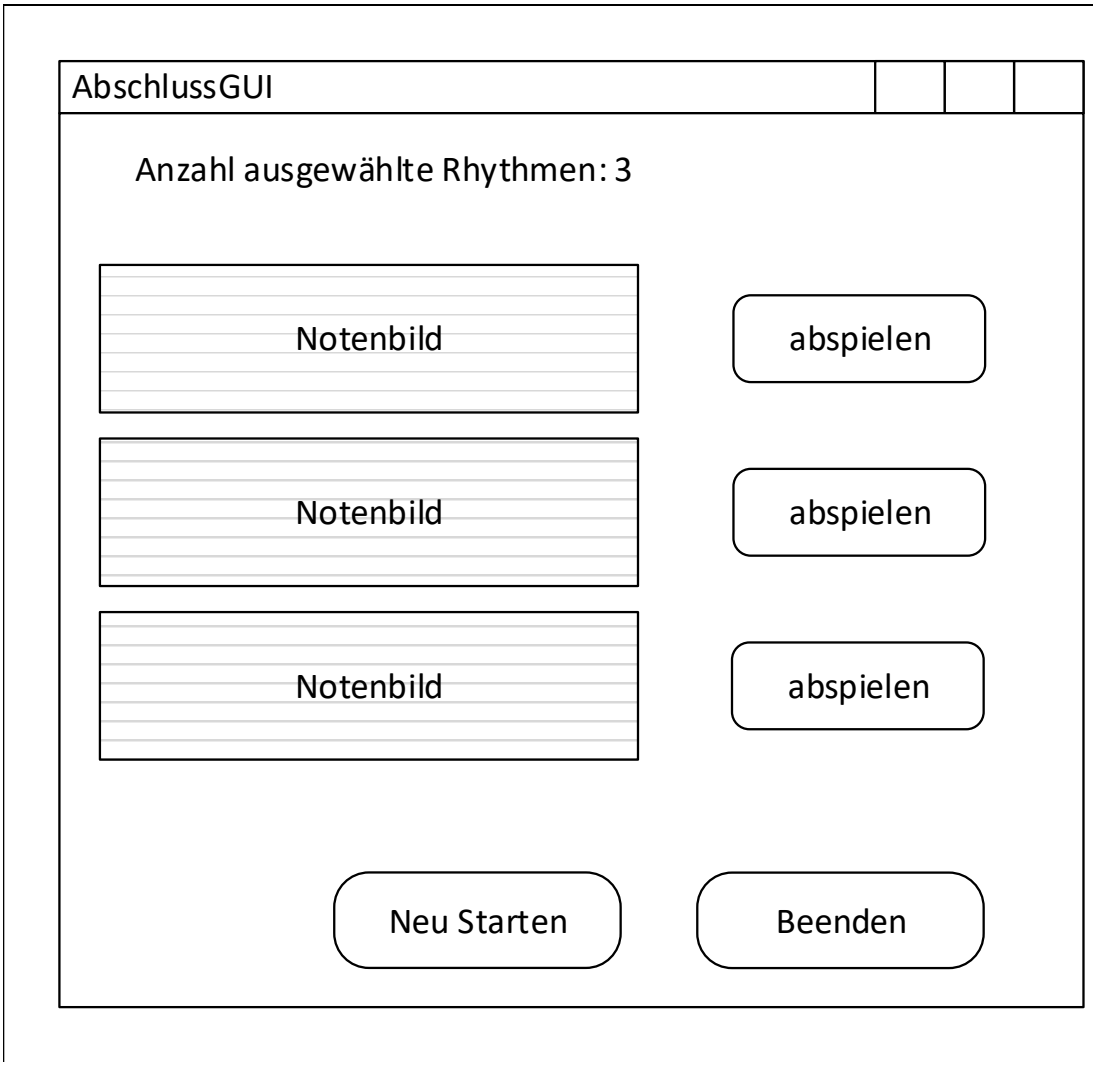

Abbildung 5.3: Skizze zur AbschlussGUI. Diese deutet alle Elemente an, die benötigt werden. Auf der Linken Seite sind immer die Notenbilder, auf der rechten Seite sind die Buttons zum Abspielen der Rhythmen. Unten gibt es 2 Buttons: zum Neu Starten der Anwendung und zum Beenden der Anwendung.

5.7.3 Evolutionsnutzerschnittstelle

Die Nutzerschnittstelle soll die Möglichkeit bieten, ähnlich wie in [MI18] die Bewertung der Individuen in Form des Einkaufsparadigmas vorzunehmen. Die Darstellung der Individuen soll zusätzlich in einer Form stattfinden, mit deren Hilfe der Nutzer eine Orientierung bekommt, welche Eigenschaften einen interessanten Rhythmus für ihn ausmachen. Möglich wäre das zum Beispiel entsprechend der Darstellung, wie sie in [SI11] gewählt wurde. Abbildung 5.4 zeigt eine Skizze zum Aufbau des Fensters zur Evolutionssteuerung.

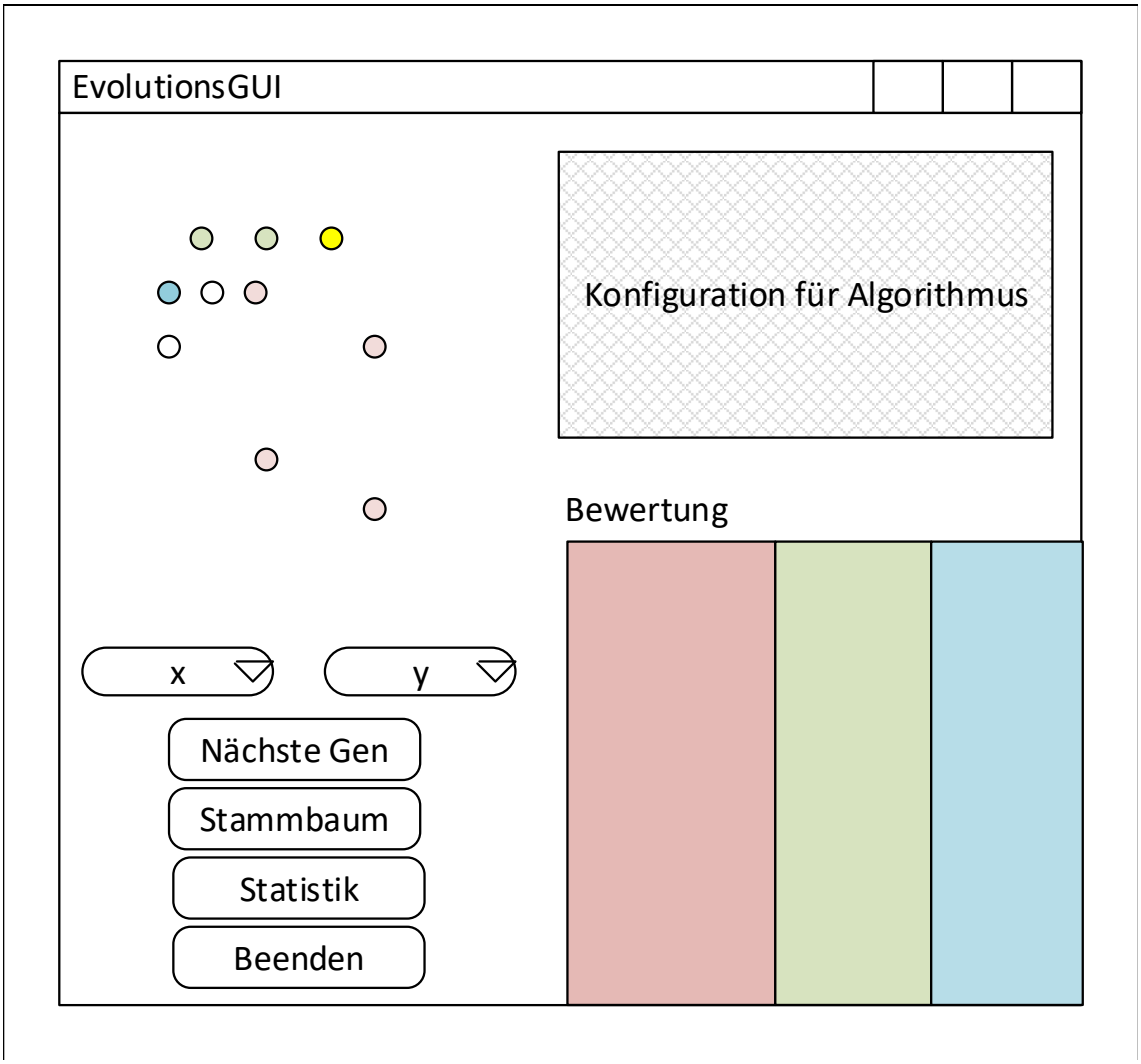

Abbildung 5.4: Skizze zur EvolutionsGUI. Diese deutet alle Elemente an, die benötigt werden. Oben links Punkte, die die Individuen repräsentieren angeordnet nach ihren Eigenschaften in der Fläche. Die Farbe steht für die aktuelle Bewertung (gelb: gehört und nicht bewertet, weiß: nicht gehört und nicht bewertet, rot: schlecht bewertet, grün: gut bewertet und blau: gut bewertet und ausgewählt). Oben Rechts Möglichkeit Parameter der Evolution anzupassen. Unten rechts, Felder zum Draufziehen der Punkte, zum Bewerten. Unten links Buttons, zur Steuerung (x- und y-Button zur Auswahl der Eigenschaften, die an den Achsen abgetragen werden. Nächste Gen, zur Durchführung des nächsten Evolutionsschritts. Stammbaum-Button zur Anzeige des Stammbaums. Statistik-Button zur Statistikanzeige und Beenden-Button, zum Abbrechen der Evolution.).

Das Fenster bietet Konfigurationen, die die Änderbarkeit der Populationsgröße, die Auswahl anderer Crossover-Algorithmen und eines stärkeren Mutationsalgorithmus, sowie die Möglichkeit das Verhältnis zwischen Mutation und Crossover zu ändern beinhalten.

Als Grundlage zur Darstellung der Individuen in der Fläche werden objektive Merkmale

benutzt, die aus den Genotypen berechnet werden können. Dadurch ist gewährleistet, dass die Merkmale nicht vom Nutzer abhängen, auch wenn sich die Präferenzen ändern, also bleiben die Eigenschaften eines Punktes immer gleich, unabhängig von der Bewertung oder der Generation. Ein Punkt wird bei gleicher Auswahl der x- und y-Achsenbelegung immer an der gleichen Stelle erscheinen.

5.7.3.1 Konfiguration des evolutionären Algorithmus

Die Konfiguration des evolutionären Algorithmus muss an die Benutzergruppe angepasst dargestellt werden. Ein Informatiker soll die Möglichkeit haben Parameter ausgiebig zu testen und darf diese so frei wie möglich wählen. Für einen Schlagzeuger ist es sinnvoll Beschreibungen zu finden, die ihm die Wirkung seiner Einstellungen darlegen und vordefinierte Werte einzustellen, sodass nur noch entschieden werden kann, welche Art der Änderung vorgenommen werden soll. Die konkreten Einstellungen bleiben dem Nutzer damit verborgen. Ein Schlagzeuger soll die Crossoverwahrscheinlichkeit und damit auch die Mutationswahrscheinlichkeit festlegen können. Außerdem soll er die Viertelnoten zwischen 2 Rhythmen tauschen können, eine große Änderung hervorrufen können, was ähnlich ist, wie neu Initialisieren und Mutationen mit kleinen Änderungen hervorrufen können. Außerdem soll er den Grenzwert für neue Rhythmen in einer Generation festlegen können. Zur Bestimmung, was eine kleine Änderung ausmacht, wurde ein Versuch durchgeführt, der in Abschnitt 7.1 beschrieben wird.

Tabelle 5.1: Die Optionen, die ein Schlagzeuger zur Beeinflussung des evolutionären Algorithmus hat.

Einflussmöglichkeit	**Beschreibung**
Crossoverwahrscheinlichkeit	Wahrscheinlichkeit, dass 2 Rhythmen vermischt werden
Mutationswahrscheinlichkeit	Wahrscheinlichkeit, dass 1 Rhythmus verändert wird
Zählzeiten-Crossover	Tausch der Viertelnoten zweier Rhythmen
neu Initialisieren	Vollkommen neue Rhythmen erzeugen
kleine Schritte Multimutation	Rhythmen mit kleinen Änderungen erzeugen
Generationsgröße	Grenzwert für die Anzahl der neuen Rhythmen

5.7.3.2 Stammbaum

Um dem Nutzer das Fortschreiten des evolutionären Algorithmus aufzeigen zu können gibt es einen Stammbaum. In diesem Fenster soll der Nutzer sehen können, das seine Bewertungen

Auswirkungen haben und damit auch dazu beitragen, dass die Motivation hoch bleibt Rhythmen zu suchen. Die Abbildung 5.5 zeigt die Darstellung des Stammbaums.

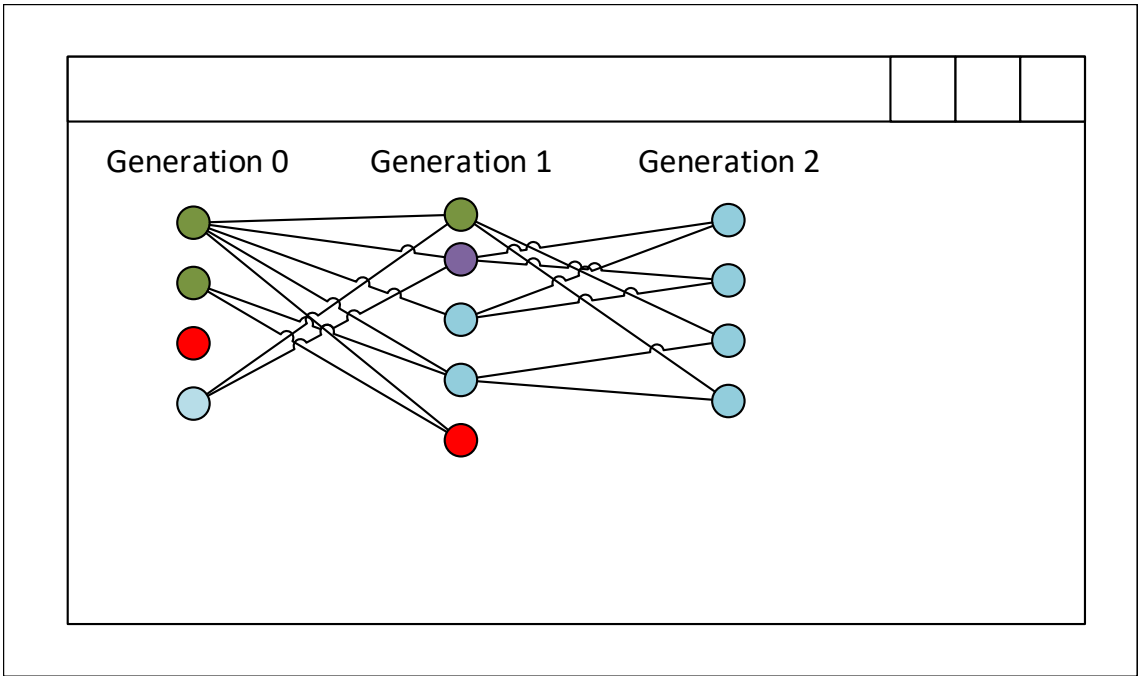

Abbildung 5.5: Skizze des Stammbaums. Die Kreise repräsentieren die Individuen. Die Farbe repräsentiert die Bewertung. Grün ist gut bewertet, blau ist nicht bewertet, rot ist schlecht bewertet und lila ist als Favorit bewertet. Die Linien kennzeichnen die Verwandtschaftsverhältnisse.

Um dem Stammbaum ein interaktives Element hinzuzufügen und um die Entwicklung besser nachvollziehen zu können ist es sinnvoll die Kreise anklickbar zu machen. Eine Funktion kann die Verwandtschaftsverhältnisse eines Kreises hervorheben, während eine andere Funktion eine Detailansicht zu der Entstehung eines Kreises bieten kann.
Die Detailansicht eines Crossover zeigt dabei die Notenbilder der Eltern, sowie des Kindes und färbt Bestandteile so ein, dass die Zugehörigkeit zu dem entsprechenden Elternteil sichtbar wird. Bei der Detailansicht einer Mutation können die Teile, die mutiert worden sind hervorgehoben werden, um den Unterschied zu dem Elternindividuum zu verdeutlichen. Bei einer Erzeugung kann lediglich das Individuum präsentiert werden, das erzeugt wurde.

5.7.3.3 Eigenes Individuum erzeugen

Um dem Nutzer einen großen Einfluss auf den evolutionären Algorithmus zu geben kann dieser selbst Individuen vorschlagen. Zur Steuerung dieses Prozesses wird ihm ein Fenster angezeigt, das ein Gitter anzeigt, mit Positionen, an denen ein Instrument gesetzt werden kann. Damit die Rahmenbedingung eingehalten wird, dass nur physisch umsetzbare Rhythmen erzeugt

werden muss eine Kennzeichnung eingeführt werden, die anzeigt, welche Felder noch setzbar sind und welche nicht. Die Abbildung 5.6 zeigt eine Skizze des Fensters.
Eine weitere sinnvolle Nutzung des Fensters ist das Erstellen eines eigenen Individuums auf Grundlage eines bereits bekannten Individuums. Dabei müsste die das Fenster beim Aufruf direkt den entsprechenden Rhythmus abbilden. Diese Art der Erzeugung eines Rhythmus würde einer Mutation durch den Nutzer entsprechen.

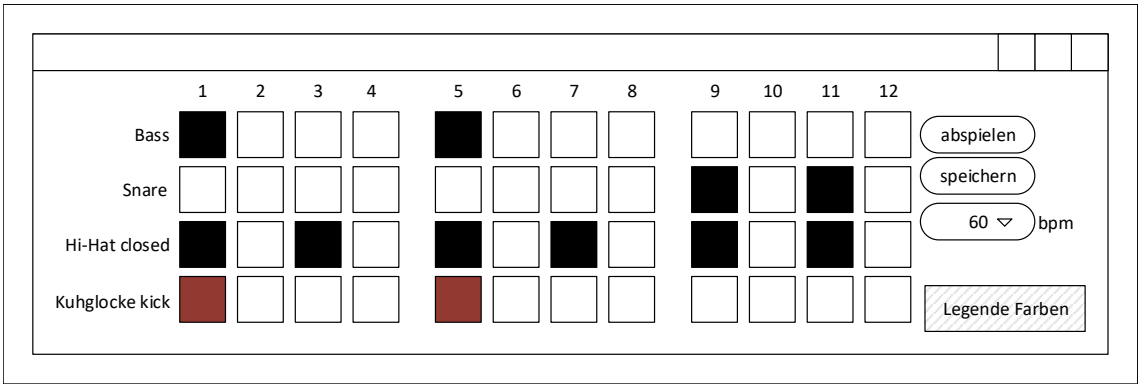

Abbildung 5.6: In Anlehnung an Fig. 1 aus [VLN+16]. Skizze zum Fenster mit der GUI zum Erstellen des eigenen Rhythmus. Die Felder zeigen die Positionen an, an denen ein Instrument im Rhythmus gesetzt werden kann. Die Farbe schwarz bedeutet, dass das Instrument gesetzt ist. Rot bedeutet, dass das Instrument nicht gesetzt werden kann und weiß bedeutet, dass ein Instrument gesetzt werden kann. Des Weiteren gibt es eine Legende, die die Farben erklärt, damit der Nutzer nicht verwirrt ist und Buttons zum Abspielen, Speichern und Ändern der Abspielgeschwindigkeit.

5.8 Evolutionärer Algorithmus

5.8.1 Vorbereitung

Aus den vorgegebenen Techniken und Instrumenten können alle Allele berechnet werden, die möglich sind zu spielen. Ein Allel ist spielbar, wenn es einem Menschen möglich ist die Instrumente, die dieses Allel bilden gleichzeitig zu benutzen. Da es Techniken gibt, die sich gegenseitig ausschließen gleichzeitig gespielt zu werden, gibt es eine Liste, die die unmöglichen Kombinationen enthält. Ein Beispiel ist, dass man eine Hi-Hat nicht gleichzeitig geschlossen und offen spielen kann.
Die Berechnung aller möglichen Allele vor Beginn des evolutionären Algorithmus vermindert den Rechenaufwand, gegenüber der Methode, dass bei der Erzeugung eines neuen Rhythmus immer geprüft werden muss, ob dieser Rhythmus überhaupt spielbar ist, erhöht allerdings den Speicherbedarf. Der erhöhte Speicherbedarf ist bei modernen Computern nicht so pro-

blematisch, wie die Auswirkungen einer erhöhten Laufzeit, die durch immer neue Prüfungen der Genotypen notwendig wäre.
Die Allele werden in Gruppen gespeichert. Jede Gruppe enthält alle Allele, die die gleiche Anzahl an Instrumenten haben. Es gibt fünf mögliche Gruppen. Diese gehen von null bis vier Instrumente. Die Gruppe mit null Instrumenten enthält immer genau ein Element, das die Pause repräsentiert.
Die Speicherung der Allele in Gruppen bietet den Vorteil, dass man auf diese Weise einfach eine Mutationsoperation implementieren kann, die auf der Anzahl der Instrumente beruht.
Die Erzeugung der Listen erfolgt nach dem im Listing 5.1 gezeigten Muster.

```
gruppe_0 = erstelleGruppe(0);

for( i = 1; i <= 4; i++ ){
  foreach(Allel al : gruppe_(i-1)){
    foreach(Instrument instrument : instrumenteListe){
      if(al.addAble(instrument))
        gruppe_i.add(
        al.addInstrument(instrument));
    }
  }
}
```

Listing 5.1: Pseudocode zur Erstellung der Allelgruppen.

Die Prüfung, ob ein Instrument zu einem Allel hinzufügbar ist, prüft, ob das Instrument bereits enthalten ist, ob beim Hinzufügen ein Konflikt entstehen würde, der durch die Konfliktliste dokumentiert ist, und ob durch das Hinzufügen die Anzahl der gleichzeitig im Einsatz befindlichen Hände und Füße jeweils größer als zwei ist. In diesen drei Fällen schlägt die Prüfung fehl.

5.8.2 Initialisierung

Die Initialisierung basiert auf den durch den Nutzer festgelegten Werten: Anzahl der Takte, Taktart und der Musterwahrscheinlichkeit.
Die Musterwahrscheinlichkeit gibt an, wie wahrscheinlich es ist, dass ein Genotyp auf Grundlage eines Musters initialisiert wird. Die Funktion Individuen mit Muster zu initialisieren wird eingeplant, um Rhythmen in der initialen Generation zu ermöglichen, die bekannte Muster enthalten, um die Ermüdung der Nutzer zu verringern.
Ein Genotyp der ohne Muster initialisiert wird, wird durch einen Zufallsgenerator zusammengesetzt.

Ein Genotyp, der mit Muster initialisiert wird, wird ebenfalls durch einen Zufallsgenerator zusammengesetzt, mit dem Unterschied, dass dieser darauf achtet, dass an bestimmten Positionen des Genotyps nur Allele infrage kommen, die ein oder mehrere bestimmte Instrumente enthalten. Dafür haben Instrumente die Attribute: Hochelement, Tiefelement und Beatelement. Die Benennung der Attribute basiert auf der Unterscheidung in Instrumente mit starken Einfluss auf den Rhythmus und schwachen Einfluss auf den Rhythmus, wie in Abschnitt 4.5 beschrieben.
Ein Muster kann mit drei Listen beschrieben werden. Jede der Listen enthält die Positionen der entsprechenden Elemente in einem Rhythmus.

5.8.3 Selektion

Die Selektion erfolgt auf Basis der den Individuen vom Nutzer zugeordneten Bewertungen. Die Bewertungen bilden dabei Ränge ab. Es gibt insgesamt drei verschiedene Ränge. Zu Rang eins zählen alle Individuen, die mit eins oder zwei bewertet wurden. Zu Rang zwei gehören alle Individuen, die mit null bewertet wurden, also die nicht bewerteten. Zu Rang drei gehören alle, die mit minus eins bewertet wurden.
Der höchste Rang ist Rang eins und bekommt die höchste Wahrscheinlichkeit, ausgewählt zu werden. Rang zwei bekommt eine niedrigere Wahrscheinlichkeit und Rang drei die kleinste Wahrscheinlichkeit ausgewählt zu werden. Nachdem der Rang ausgesucht wurde, wird ein Individuum mit diesem Rang zufällig ausgewählt. Anschließend wird auf die gleiche Weise ein zweiter Elternteil ausgewählt. Wenn dieser zufällig identisch mit dem ersten Elternteil ist, wird die Auswahl wiederholt.
Nach der Auswahl zweier Eltern wird ein Abwandlungsoperator angewendet. Wenn es die Mutation ist, wurde der zweite Elternteil umsonst ausgewählt. Die Selektion mit anschließendem Abwandlungsoperator wird so oft durchgeführt, bis die neue Generation voll ist.

5.8.4 Abwandlungsoperatoren

Es werden grundlegend die klassischen Abwandlungsoperatoren Einpunkt-Crossover und Mutation mit Änderung eines Allels implementiert.
Nicht implementiert werden Operationen, die auf Permutation basieren, weil die Mutation den Effekt haben soll, neue Allele in den Genpool einzubringen, oder zu entfernen und die Crossover-Operationen nicht anwendbar sind.
Als zusätzliche Mutationsoperation wird eine Mutationsoperation implementiert, die alle Allele des Genotyps durchgeht und mit einer Wahrscheinlichkeit, die vom Nutzer festgelegt

wird ändert. Diese Operation wird in dieser Arbeit Multimutation genannt. Die Multimutation wird in zwei Varianten umgesetzt.
In der einen Variante werden kleine Änderungen vorgenommen. Das bedeutet, dass das zu ersetzende Allel nur durch eins ersetzt werden kann, das durch Hinzufügen oder Entfernen eines Instruments aus dem zu ersetzenden Allel entstehen kann.
In der zweiten Variante kann das zu ersetzende Allel durch ein beliebiges Allel ersetzt werden. Wenn man diese Operation mit einer sehr hohen Änderungswahrscheinlichkeit benutzt, wirkt diese wie eine Neuerzeugung der entstehenden Kindindividuen, da die Ähnlichkeit zwischen Elternindividuum und Kindindividuum sehr gering ausfallen kann.
Außerdem wird der Dreipunkt-Crossover in zwei Varianten umgesetzt.
In der ersten Variante ist er symmetrisch. Dabei entstehen vier gleichlange Teile, die ausgetauscht werden. Aufgrund der Beschaffenheit des Genotyps ist die Taktart und Anzahl der Takte irrelevant für das funktionieren der Aufteilung in vier gleichlange Teile, da durch die Auflösung als Sechzehntelnote immer durch vier teilbare Genotyplängen entstehen.
In der zweiten Variante ist der Dreipunkt-Crossover asymmetrisch. Dabei entstehen vier unterschiedlich lange Teile pro Elternteil, die ausgetauscht werden.
In Anlehnung an die Theorie zu den Notenwerten wird eine Crossoveroperation umgesetzt, der zwischen zwei Rhythmen die Zählzeiten austauscht. Diese Operation setzt eine Schnittstelle bei den Individuen im Abstand von vier Allelen.

5.9 Eigenschaften eines Individuums

Die Berechnungen der objektiven Eigenschaften erfolgen auf Grundlage des Patterns. Ideen für berechenbare Eigenschaften sind in Anlehnung an [KFV13] in der Tabelle 5.2 zu finden.

Tabelle 5.2: Eigenschaften eines Rhythmus nach dem Vorbild aus [KFV13]: Die Eigenschaften beziehen sich immer auf Membranophone, Idiophone und Pausen.

Eigenschaft	**Erläuterung**
Membranophoneabstand	arithmetischer Mittelwert des Abstands bis zum nächsten Auftreten eines Membranophones
Idiophoneabstand	arithmetischer Mittelwert des Abstands bis zum nächsten Auftreten eines Idiophones
Mittlere Belegung	arithmetischer Mittelwert der Summen aus eingesetzten Händen und Füßen pro Note
Pausenverhältnis	Verhältnis von Pausen zu nicht Pausen
Pausenabstand	arithmetischer Mittelwert des Abstands bis zum nächsten Auftreten einer Pause
Pausen pro Zählzeit	arithmetischer Mittelwert der Pausen pro Zählzeit
Anzahl Membranophone	Anzahl der vorkommenden Membranophone im Rhythmus
Anzahl Idiophone	Anzahl der vorkommenden Idiophone im Rhythmus
Anzahl Pausen	Anzahl der vorkommenden Pausen im Rhythmus
Mem/Idio Verhältnis	Verhältnis von Membranophonen zu Idiophonen

Abbildung 5.7 zeigt an einem Beispiel, wie die Abstandswerte ermittelt werden. Dabei ist zu beachten, das ein Rhythmus eine wiederholbare Abfolge von Klängen ist. Also ist der Abstand vom letzten Auftreten eines Elements bis zum ersten Auftreten ebenfalls zu berücksichtigen.

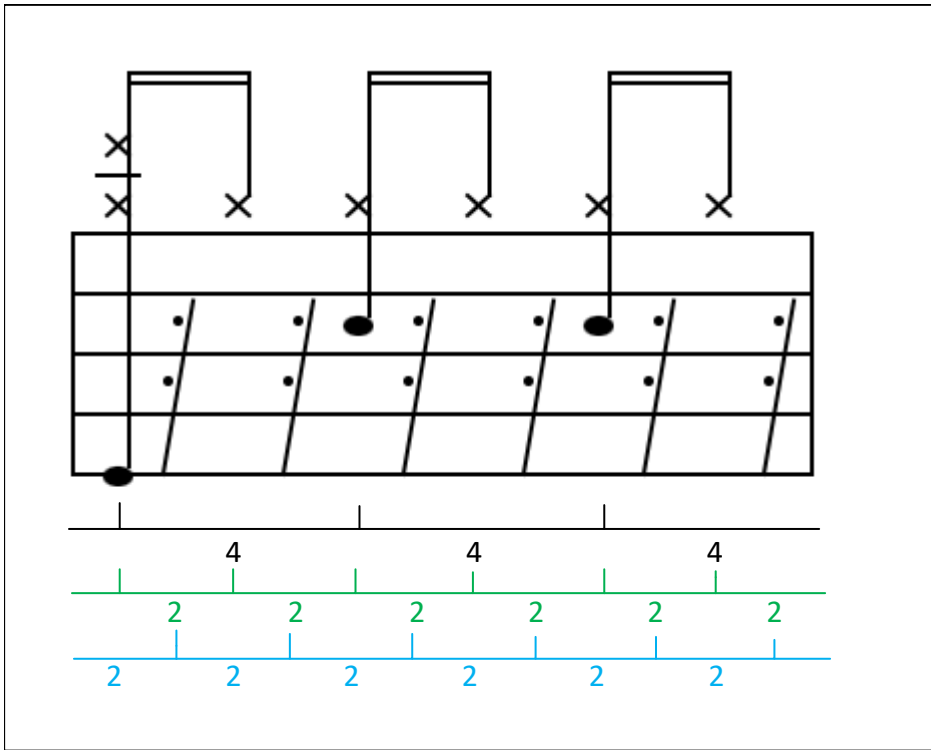

Abbildung 5.7: Dieses Bild zeigt beispielhaft, wie der Abstand ermittelt wird. In schwarz, der Membranophoneabstand, in grün der Idiophoneabstand und in blau der Pausenabstand.

5.10 Phänotyp

Der Phänotyp hat zwei Ausprägungsformen.

Die erste Form ist das Notenbild. Dabei handelt es sich um die Darstellung des Rhythmus in Form von Noten. Für jedes Instrument ist ein Symbol und die Notenlinie, oder der Zwischenraum zweier Notenlinien, definiert. Abbildung 5.8 zeigt die Kodierung der Position eines Instruments.

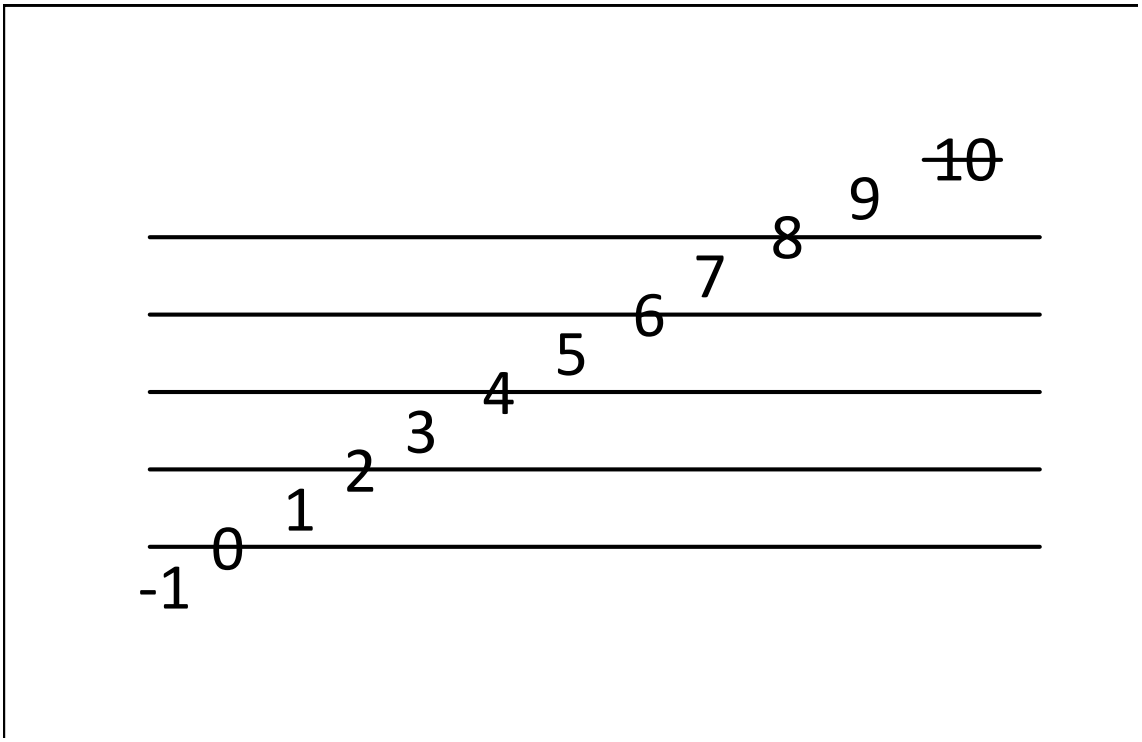

Abbildung 5.8: Darstellung des Positionscodes für Instrumente. Als Orientierung wird die unterste Linie des Systems benutzt. Gerade Zahlen sind auf einer Linie, ungerade Zahlen sind zwischen zwei Linien.

Zur Festlegung der Position und des Symbols für ein Instrument wurde die Kodierung von [Kod] benutzt.

Die zweite Form des Phänotyps ist die akustische Ausgabe. Daür gibt es zu jedem Instrument eine Wave-Datei, die das akustische Signal des Klangs dieses Instruments enthält. Die Übersetzung des Genotyps in den akustischen Phänotyp wird mit Hilfe von einzelnen Tonaufnahmen von Schlagzeugkomponenten zu einem Audiosignal zusammengesetzt. Das Zusammensetzen der Tonaufnahmen klingt natürlich und könnte dadurch einen positiven Effekt auf den Benutzer haben. Tonaufnahmen können von der Webseite freewavesamples[1] bezogen werden.

Das Abspielen eines Rhythmus erfolgt so, dass alle Allele der Reihe nach durchgegangen werden und für jedes Allel die Wiedergabe der WAV-Datei der Instrumente gestartet wird. Nach der Wiedergabe eines Allels wird eine Verzögerung (v) in Millisekunden eingebaut, deren Länge sich durch die Angabe des Wertes Beats pro Minute (bpm) nach folgender Formel berechnet:

$$v = \frac{\frac{60000}{bpm}}{4}$$

[1] https://freewavesamples.com/sample-type/drums/

Die 60000 ist eine Minute in Millisekunden geteilt durch die Anzahl der Beats pro Minute um den Wert der Verzögerung einer Viertelnote zu bekommen und dann nochmal durch 4 geteilt, um den zeitlichen Abstand der Sechzehntelnoten zu erhalten.

5.11 Stammbaum

Der Stammbaum wird so gespeichert, dass pro Generation die Entstehungen der Individuen gespeichert wird. Es gibt drei Formen, der Entstehung.
Die Erzeugung, damit sind alle Individuen gemeint, die ohne Eltern entstehen, also durch Zufall bei der Initialisierung oder durch den Nutzer selber erstellt wurden.
Die Mutation ist die Entstehungsart, bei der ein Individuum zufällig an einer oder mehreren Stellen geändert wird, sodass ein neues Individuum entsteht.
Das Crossover ist die Entstehungsart, bei der zwei Individuen durch die Vermischung von zwei Individuen früherer Generationen entstehen.
Durch diese Art der Speicherung ist direkt jede Eltern-Kind-Beziehung gespeichert und kann bei der Darstellung genutzt werden.
Außerdem werden die wichtigen Stellen zur Nachvollziehbarkeit der Veränderung ebenfalls gespeichert. Das bedeutet, dass zum Beispiel für eine Mutation gespeichert wird, welches Individuum der Elternteil ist, welches Individuum das entstandene Kind ist und an welchen Stellen die Änderungen stattfanden.

5.12 Statistik

Die Statistik wird erfasst, um die globalen Veränderungen bei der Evolution fassbar zu machen.
Dazu gehören die vorkommenden Allele pro Generation, die Anzahl der Genotypen, die Änderungen in der Konfiguration der Hyperparameter der Evolution.
Durch das Erfassen der Statistik wird es möglich einen Überblick darüber zu bekommen, wie schwierig es ist etwas zu finden, das dem Nutzer gefällt, da die Größe des Lösungsraums ausrechenbar ist und es kann bestimmt werden, wie groß der Fortschritt bei der Suche nach interessanten Rhythmen ist.

6 Realisierung

Dieses Kapitel beschreibt die Umsetzung des Projektes.

6.1 Softwarearchitektur

Das Programm soll die in dieser Arbeit besprochenen Konzepte umsetzen und muss daher keine allgemein einsetzbaren Programmbausteine erzeugen.
Die folgende Aufzählung zeigt die Java-Pakete des Projekts und ihren Inhalt:

- rahmen: Pakete und Klassen, die zur Umsetzung des Rahmens der Anwendung benutzt werden
- genGui: Pakete und Klassen, die zur Umsetzung der Nutzerschnittstelle während der Evolution genutzt werden
- genAlgorithmus: Pakete und Klassen, die zur Umsetzung des genetischen Algorithmus genutzt werden
- dokumentation: Pakete und Klassen, die zur Umsetzung der Dokumentation genutzt werden
- konfiguration: Klasse, die die Konfigurationswerte repräsentiert
- datenschnittstelle: Klassen, die den Zugriff auf die Ressourcen ermöglichen

Außerdem gibt es noch ein Ressourcen-Verzeichnis, das Dateien enthält, die fest zum Programm dazugehören, dazu gehören Dateien, die die Standardinstrumente beschreiben, Sound-Dateien, zu den Standardinstrumenten, Rhythmusmuster und Bilder, die als Cursor eingesetzt werden. Die Struktur und der Inhalt des Ressourcen-Verzeichnisses sind in Abbildung 6.1 zu sehen. Die Rhythmusmuster sind gruppiert in Ordnern gespeichert, die die jeweilige Taktart repräsentieren. In jedem Ordner liegen Dateien mit nummerischen Namen, die jeweils ein Muster enthalten.
Zur Nutzung des Programms sind 18 Standardinstrumente integriert, die das definierte

Standardschlagzeug darstellen, inklusive einiger Variationen mit den vorgestellten Schlagtechniken. Zu jedem Instrument gibt es eine beschreibende Konfigurationsdatei und eine Sound-Datei im WAV-Datei Format.

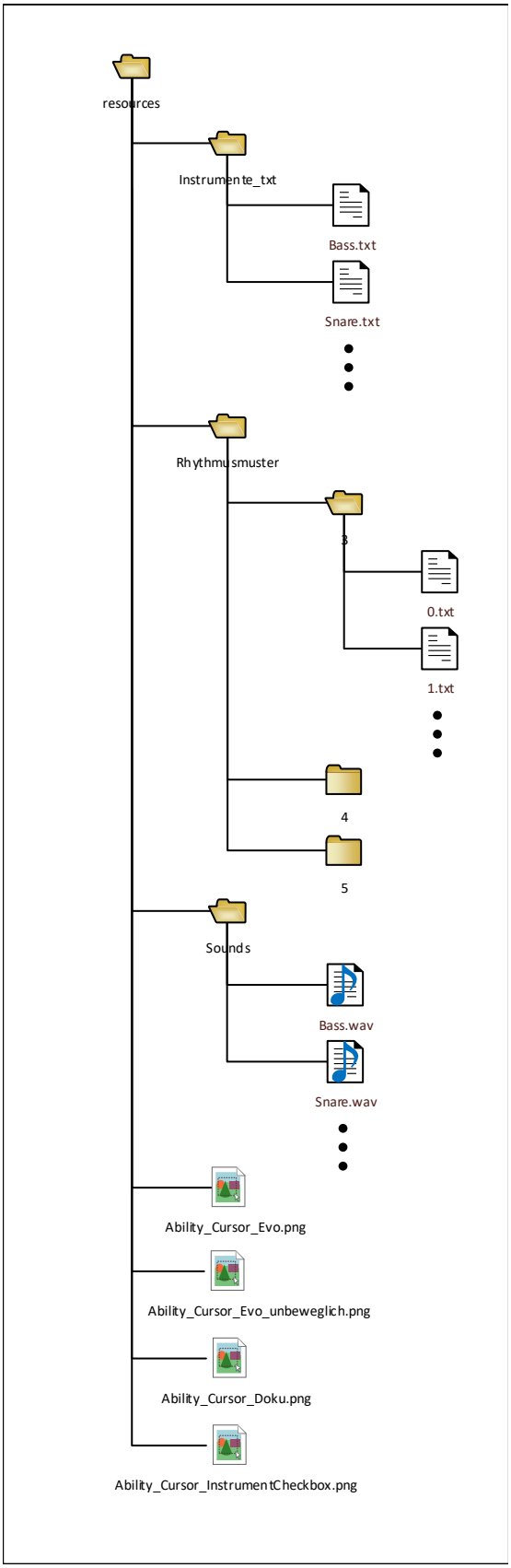

Abbildung 6.1: Struktur des Ressourcen-Verzeichnisses. Es enthält Daten zu Rhythmusmustern, Instrumenten, die als Standardinstrumente dienen und Bilddateien, die die selbst erstellten Cursor enthalten.

6.2 Individualisierung

Um dafür zu sorgen, dass die Ergebnisse individuell sind und auch zu einem späteren Zeitpunkt verfügbar sind und weiterverarbeitbar, wird bei Programmstart immer verlangt, dass ein Workspace ausgewählt wird. Sollte keiner vorhanden sein, wird einer erstellt.
Ein Workspace ist so aufgebaut, dass es für die Daten, die in serialisierter Form gespeichert werden, um das Laden der Arbeit zu ermöglichen, im “Daten”-Verzeichnis abgelegt sind. Daten, die den Verlauf des evolutionären Algorithmus dokumentieren sind im Verzeichnis “Dokumentation” zu finden. Die dort abgelegten Dateien haben die Besonderheit, dass sie in einer für Menschen lesbaren Form gespeichert sind und so auch von anderen Programmen nutzbar sind. Im Verzeichnis “eigene Instrumente” werden die Daten zu Instrumenten gespeichert, die vom Nutzer selbst hinzugefügt wurden. Im “Export”-Verzeichnis liegen Daten, die in einem nutzbaren Format von Menschen weitergenutzt werden können. Die Erzeugung dieser Daten muss vom Nutzer aus dem Programm heraus angestoßen werden. Die Abbildung 6.2 zeigt die Struktur eines Workspaces.

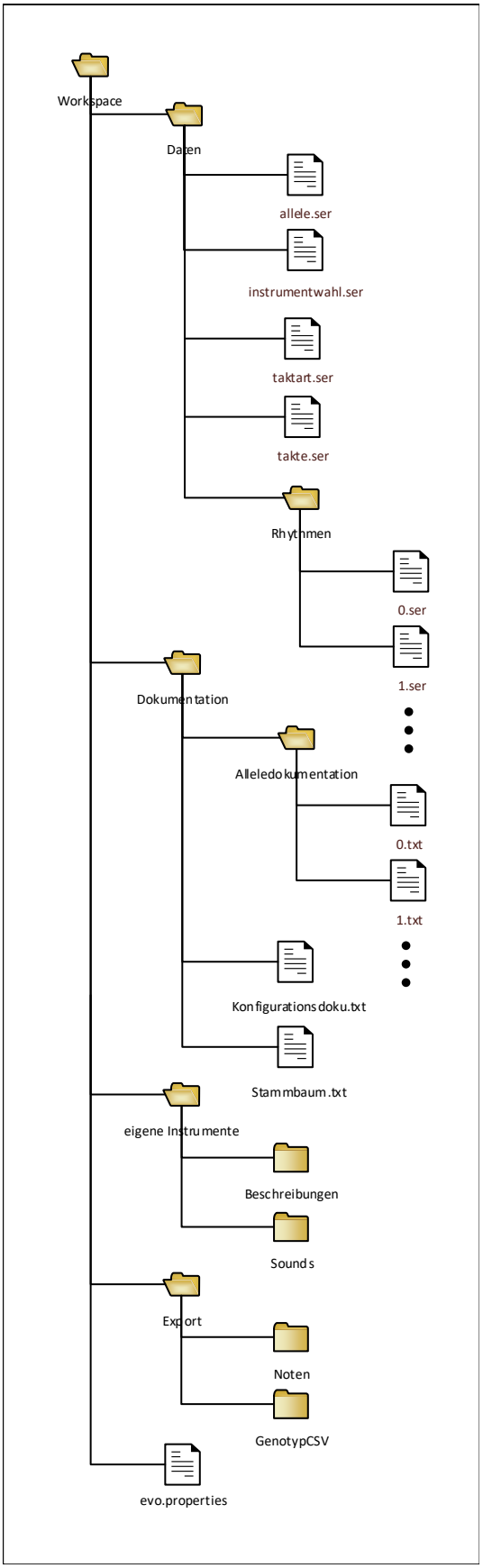

Abbildung 6.2: Struktur eines Workspaces.

6.3 Datenschnittstelle

Die Datenschnittstelle dient der Verwaltung der Daten, die zu den Ressourcen des Programms gehören. Diese Daten umfassen Rhythmusmuster und Instrumente, die als Standardinstrumente ausgewählt wurden.
In dem Java-Paket “datenschnittstelle” sind vier Klassen zur Verwaltung dieser Daten enthalten. Die Klasse “Instrument” stellt ein Instrument und die Klasse “Rhythmusmuster” stellt ein Rhythmusmuster im Programm dar. Die Klassen “InstrumentVerwalter” und “RhythmusmusterVerwalter” enthalten eine Liste mit allen zu verwaltenden Objekten, die genutzt werden können und enthalten die Methoden zum Verwalten der entsprechenden Daten. Mit der Klasse “InstrumentVerwalter” können darüber hinaus auch Instrumente aus dem Workspace verwaltet werden, die vom Nutzer hinzugefügt wurden.
Listing 6.1 zeigt den Inhalt einer Rhythmusmusterdatei für ein 4/4-Takt-Muster. Die Musterlänge ist immer für zwei Takte ausgelegt.

```
0 Hoch=4,12,20,28
  Tief=0,8,16,24
2 Beat=0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30
```

Listing 6.1: Inhalt einer Textdatei zu einem 4/4-Takt-Muster.

Diese Rhythmusmusterdateien werden durch den “RhythmusmusterVerwalter” geladen und die entsprechenden Rhythmusmusterobjekte erzeugt.
Listing 6.2 zeigt den Inhalt der Instrument-Datei für die Snare.

```
0 Name/ID=Snare
  NutzCode=1
2 SymbolCode=0
  PositionsCode=5
4 Idiophone=false
  Sound_Pfad=/Sounds/Snare.wav
6 BeatElement=false
  TiefElement=false
8 HochElement=true
```

Listing 6.2: Inhalt einer Textdatei zu dem Instrument Snare.

Die Instrumentdateien werden durch den “InstrumentVerwalter” geladen und die entsprechenden Instrumentobjekte erzeugt.

6.4 Genetische Klassen

In der Programmstruktur sind die genetischen Klassen “Allel” und “Genotyp” in dem Paket “genKlassen” enthalten. Diese dienen der programminternen Darstellung von Individuen. Ein Allel ist eine Liste von Instrumenten, bei der die Reihenfolge keine Rolle spielt. Die Klasse “Allel” ist so konzipiert, dass nur Allele entstehen können, bei denen zum gleichzeitigen Benutzen der Instrumente nicht mehr als zwei Hände und zwei Füße gebraucht werden. Des Weiteren gibt es Kombinationen aus Techniken, die nicht erlaubt sind. Es ist auch nicht zulässig, dass ein Instrument doppelt vorkommt. Wenn eins der drei Kriterien bei der Erzeugung eines Allelobjekts nicht erfüllt sein sollte, wird eine entsprechende Exception erzeugt. Diese Exceptions können die “ChecksummeUeberschrittenException”, “InstrumentVerbotenException” oder die “InstrumentBereitsEnthaltenException” sein. Das Listing 6.3 zeigt die Methode mit der Instrumente zu einem Allel hinzugefügt werden.

```
0  /**
   * Die Methode hinzufuegen benutzt die Methode add von Arraylist um das Element
2  * hinzuzufuegen. Ausserdem wird die Nutzsumme gesetzt. Sollte ein nicht
   * gueltiges Allel entstehen, wird eine Exception erzeugt. <br>
4  * <br>
   * Ein ungueltiges Allel ist eins, welches ein Instrument mehrfach enthaelt,
       eine unzulaessige
6  * Kombination von Instrumenten enthaelt oder wenn der Nutzcode fuer die Haende
       oder Fuesse groesser
   * als 2 ist.
8  *
   * @param in -> Instrument
10 * @return
   * @throws ChecksummeUeberschrittenException
12 * @throws InstrumentBereitsEnthaltenException
   * @throws InstrumentVerbotenException
14 */
   public boolean hinzufuegen(Instrument in)
16 throws ChecksummeUeberschrittenException, InstrumentBereitsEnthaltenException,
       InstrumentVerbotenException {
     if (this.contains(in))
18     throw new InstrumentBereitsEnthaltenException();
     // Pruefen, ob das Instrument in der Verbotsliste ist
20   for (Instrument instrument : this) {
       try {
22       if (verbotsliste.get(instrument).contains(in)) {
           throw new InstrumentVerbotenException();
```

```
24          }
        } catch (NullPointerException e) {
26      // Tue nichts, weil das bedeutet, dass das Instrument nicht in der
        Verbotsliste
        // ist
28      }
    }
30  if (pruefeZufuegbarkeit(in))
      setzeNutzsummen(in, true);
32  else
      throw new ChecksummeUeberschrittenException();
34  boolean erg = super.add(in);
    return erg;
36 }
```

Listing 6.3: Methode zum Hinzufügen eines Instruments zu einem Allel. Diese Methode wirft Exceptions, wenn das Hinzufügen nicht zulässig ist.

Der Genotyp ist als eine Liste von Allelen umgesetzt, bei denen die Reihenfolge eine zeitliche Abfolge darstellt. Außerdem enthält die Klasse die Information, wie das Individuum bewertet wurde, die Taktart und die Anzahl der Takte, woraus die Länge des Genotyps ableitbar ist, die Generationsnummer und eine ID, zur eindeutigen Identifizierung des Individuums.

6.5 Phänotypklassen

In dem Java-Paket “phaenotypKlassen” sind die Klassen, die zur Darstellung eines Individuums genutzt werden. Die Darstellung kann visuell und akustisch erfolgen.

6.5.1 visuelle Ausgabe

Die Notendarstellung ist die visuelle Darstellung eines Individuums. Sie basiert auf dem Schema aus [Kod]. Die Kodierungen aller Instrumente des Standardschlagzeugs im Notenbild sind in Anhang E zu sehen.
Die Klasse “NotenBild” bietet die Möglichkeit das Individuum komplett in schwarz, mit Markierungen in einer Farbe und mit Markierungen in zwei Farben darzustellen. Die Größe des Notenbildes ist dabei abhängig von der Größe der Genotypen, deren Länge durch die Anzahl der Takte und die Taktart festgelegt ist.
Beim Zeichnen des Notenbildes wird zuerst das Notengitter gezeichnet und anschließend der Genotyp sequenziell durchgegangen. Für jedes Allel werden die Instrumente durchgegangen

und aus dem Positionscode die Position auf der vertikalen Achse bestimmt. Des Weiteren wird der Positionscode genutzt um zu ermitteln, ob eine Hilfslinie gezeichnet werden muss, die dazu dient, einfacher Noten lesen zu können, die über oder unter den fünf Linien des Notengitters positioniert sind. Die Position des Allels im Genotyp ist die Position auf der horizontalen Achse für jedes Instrument, das Teil des Allels ist. Zur Ermittlung des Symbols, das gezeichnet werden muss, wird der Symbolcode des Instruments verwendet. Wenn ein Allel eine leere Liste ist, dann wird ein Pausensymbol gezeichnet.
Nachdem der Genotyp abgearbeitet wurde, werden die Linien gezeichnet, die der Gruppierung der Noten in Viererblöcke dient. Abbidlung 6.3 zeigt ein Beispiel für ein Notenbild, das durch das Programm erzeugt wurde.

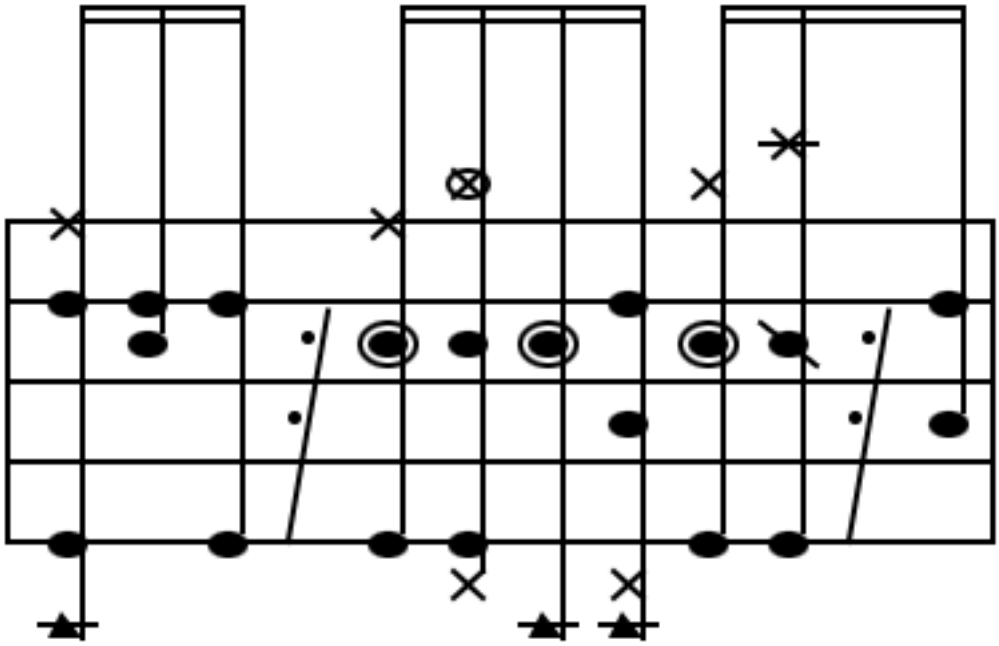

Abbildung 6.3: Beispiel für ein Notenbild das durch das Programm erzeugt wurde für einen Rhythmus im 3/4 Takt.

6.5.2 akustische Ausgabe

Die Klasse “RhythmusSpieler” ist für die akustische Ausgabe der Individuen zuständig. Damit das Programm nicht für die Dauer des Abspielens blockiert ist, erweitert die Klasse “RhythmusSpieler” die Klasse “Thread”.
Ein Rhythmus-Spieler spielt einen Rhythmus genau einmal.

Bei Threads ist darauf zu achten, dass diese nicht versuchen parallel eine Wiedergabe eines Individuums auszuführen. Um sich überlappende Wiedergaben zu vermeiden wurde ein Mechanismus eingebaut, der dafür sorgt, dass allen Rhythmus-Spielern bekannt ist, welcher Rhythmus derzeit abgespielt wird. Rhythmus-Spieler, die den gleichen Rhythmus abspielen wollen, reihen sich in einer Warteschleife ein. Sollte sich der Nutzer entscheiden einen anderen Rhythmus abspielen zu wollen, wird ein Schalter gesetzt, der allen Rhythmus-Spielern mit dem vorherigen Rhythmus signalisiert, dass das Abspielen abgebrochen werden soll.
Die Verzögerungen zwischen dem Abspielen der Allele wird in Form einer Klassenvariable "HashMap" umgesetzt. Schlüsselwerte sind die Geschwindigkeitswerte in bmp und die Werte der "HashMap" sind die Verzögerungen in Millisekunden. Da es Rhythmen gibt, die langsam besser wirken, als schnell, gibt es vier Abstufungen.

6.6 genetischer Algorithmus

Die Komponenten, zur Umsetzung des genetischen Algorithmus befinden sich im Paket "genAlgorithmus". Im wesentlichen handelt es sich um 2 Komponenten, die in unterschiedlichen Ausführungen genutzt werden können.
Die eine Komponente ist die Klasse "GenerationVerwalter". Dabei handelt es sich um eine abstrakte Klasse. Die Funktion dieser Komponente ist die Verwaltung der Genotypen, sodass die Generationsverhältnisse erkennbar sind. Dazu wurde eine "HashMap" genutzt, deren Schlüssel die Generationsnummer ist. Der Wert ist eine "HashMap" mit dem Schlüssel, ID des Genotyps, und als Wert der Genotyp.
Diese Klasse verfügt über Methoden zum Speichern und Laden von Generationen, sodass eine spätere Arbeit mit dem Workspace ermöglicht wird. Des Weiteren wird die Selektion auf Grundlage von Rängen in dieser Klasse durchgeführt. Die Ränge entsprechen den Bewertungen. Der höchste Rang sind die Genotypen mit der Bewertung eins oder zwei. Der zweithöchste Rang sind die Genotypen mit der Bewertung null und der niedrigste Rang sind die Genotypen mit der Bewertung minus eins.
Von dem "GenerationVerwalter" gibt es zwei Ausprägungsformen. Die Klasse "EinfacherGenerationVerwalter" ist für evolutionäre Algorithmen gedacht, bei denen die Population gleich der Generation ist.
Die Klasse "GenerationsuebergreifenderGenerationVerwalter" ist für evolutionäre Algorithmen gedacht, wo die Population aus mehreren Generationen besteht und folglich die neue Generation auf Grundlage von Individuen verschiedener Generationen entstehen kann.
Die zweite Komponente ist das "GAModul". Diese kommt in zwei Varianten vor. Die Trennung

in “GAModul” und “ErweitertesGAModul” hat den Grund, dass das erweiterte GA-Modul Methoden enthält, die von der grundlegenden und einfachsten Art der evolutionären Abwandlungsoperatoren abweichen.
Das erweiterte GA-Modul ermöglicht die Startgeneration mit festgelegten Mustern zu initialisieren, Mutationen, mit mehr als einer Änderungsstelle, wobei unterschieden werden kann zwischen leichten Änderungen und beliebigen Änderungen, und Crossover, der sich an den Zählzeiten orientiert und Crossover mit drei Bruchstellen, sowohl symmetrisch, als auch asymmetrisch. Des Weiteren wird hier die Methode zur Erzeugung eines durch den Nutzer erzeugten Individuums bereitgestellt.

6.6.1 Initialisierung

Die Initialisierung, die nur auf Zufall basiert wurde im “GAModul” umgesetzt. In der Methode “initialisieren” wird eine Schleife so oft durchgegangen, wie es Individuen in der Startgeneration geben soll. In der Schleife wird die Methode “erzeugeZufallsindividuum” aufgerufen. Der Quellcode der Methode wird im Listing 6.4 dargestellt. Bei dieser Methode wird ein Individuum aus zufällig ausgewählten Allelen zusammengesetzt. Die Auswahl der Allele erfolgt in zwei Schritten. Zuerst muss zufällig die Gruppe der Allele bestimmt werden. Die Gruppennummer entspricht der Anzahl der Instrumente im Allel. Anschließend wird ein Allel aus der Gruppe zufällig ausgewählt. Des Weiteren wird die Entstehung des Genotyps dokumentiert.

```
/**
* Methode zum Erzeugen eines neuen Individuums nach dem Zufallsprinzip.
*
* @param genNummer      -> Generationsnummer
* @param rhythmuslaenge -> Laenge der Rhythmen, Anzahl der Allele pro Rhythmus
* @param letzte         -> Markierung, ob es die letzte Operation ist
* @return
*/
public Genotyp erzeugeZufallsindividuum(int genNummer, int rhythmuslaenge,
    boolean letzte) {
  // Initialisierung eines neuen leeren Genotyps fuer ein Individuum
  Genotyp neuerGenotyp = new Genotyp(genNummer, k.getAnzTakte(), k.getTaktart
    ());
  // Schleife fuer die gesamte Rhythmuslaenge
  for (int j = 0; j < rhythmuslaenge; j++) {
   // Bestimmung der Gruppe des Allels (Anzahl Instrumente)
   List<Allel> instrumentzahl = al.get(r.nextInt(al.keySet().size()));
```

```
    // dem Genotyp ein zufaellig ausgewaehltes Allel aus der ausgewaehlten
     Gruppe
    // hinzufuegen
    neuerGenotyp.add(instrumentzahl.get(r.nextInt(instrumentzahl.size())));
  }
  // ID des Genotyps setzen
  neuerGenotyp.setId(GenerationVerwalter.rhythmenZaehler);
  // Anzahl der Rhythmen ist um 1 gestiegen
  GenerationVerwalter.rhythmenZaehler++;
  // Dokumentation vornehmen
  doku.trageDatensatzEin(neuerGenotyp, letzte);
  return neuerGenotyp;
}
```

Listing 6.4: Methode zur Erzeugung eines Zufallsindividuums. Die Variable “al” ist eine HashMap mit Listen von Allelen, die der Anzahl der enthaltenen Instrumente zugeordnet sind, die Variable “doku” verweist auf das Dokumentationsobjekt und die Variable “r” verweist auf das Objekt des Zufallsgenerators.

Außerdem gibt es die Initialisierung, die auf Mustern basiert. Dafür wird die Methode in der Klasse “ErweitertesGAModul” überschrieben. Listing 6.5 zeigt die Methode. Dabei wird zuerst ausgewertet, ob ein Individuum nach Muster entstehen soll, was von der in der Konfiguration eingestellten Wahrscheinlichkeit zur Mustererzeugung abhängig ist. Wenn ein Individuum nach Muster erzeugt werden soll, dann wird ein Muster aus der Musterliste ausgesucht und eine Schleife gestartet, bei der ein Individuum sukzessiv aufgebaut wird. Bei der Auswahl der Allele wird nun auf Listen zurückgegriffen, die die Allele mit Instrumenten mit dem entsprechenden Elementwert besitzen. Der Fall, dass in einem Muster ein Beat-Element gleichzeitig mit einem Hoch-Element, oder einem Tiefelement vorkommen kann wird dabei ebenfalls berücksichtigt. Die Entstehung des Individuums wird abschließend dokumentiert.

```
@Override
public Genotyp erzeugeZufallsindividuum(int genNummer, int rhythmuslaenge,
    boolean letzte) {
  Genotyp erg = null;
  // Wenn kein Muster verwendet werden soll, dann wird die Methode der
  // Mutterklasse benutzt
  if (k.getMusterWahrscheinlichkeitProzent() < r.nextInt(100)) {
    erg = super.erzeugeZufallsindividuum(genNummer, rhythmuslaenge, letzte);
  } else {
   /*
    * Erzeugung eines Individuums nach Muster
```

```
10      */
        erg = new Genotyp(genNummer, k.getAnzTakte(), k.getTaktart());
12      /*
        * Muster auswaehlen
14      */
        int musterNr = r.nextInt(rV.getRhythmusmuster().size());
16      // Fuer alle Positionen im Rhythmus
        for (int j = 0; j < rhythmuslaenge; j++) {
18        // Variable zur Erkennung eines gueltigen Allels
          boolean gueltig = false;
20        // Solange ein ungueltiges Allel ausgesucht ist
          while (!gueltig) {
22          // Check, ob es ein Beatelement ist
            if (rV.getRhythmusmuster().get(musterNr).getBeatListe().contains(j)) {
24            Allel allel = null;
              // Check, ob ein Tiefelement gebraucht wird
26            if (rV.getRhythmusmuster().get(musterNr).getTiefListe().contains(j))
        {
                allel = tiefElemente.get(r.nextInt(tiefElemente.size()));
28            }
              // Check, ob ein Hochelement gebraucht wird
30            if (rV.getRhythmusmuster().get(musterNr).getHochListe().contains(j))
        {
                allel = hochElemente.get(r.nextInt(hochElemente.size()));
32            }
              // Wenn Allel null, dann bedeutet das, dass ein beliebiges
        Beatelement gewaehlt
34            // werden kann. Andernafalls muss das vorher ausgesuchte Element
        auch ein
              // Beatelement enthalten.
36            if (allel != null) {
                if (beatElemente.contains(allel)) {
38                erg.add(allel);
                  gueltig = true;
40              }
              } else {
42              erg.add(beatElemente.get(r.nextInt(beatElemente.size())));
                gueltig = true;
44            }
            } else {
46            // Position gehoert nicht zum Muster --> zufaelliges Element
        auswaehlen
              List<Allel> instrumentzahl = al.get(r.nextInt(al.keySet().size()));
48            erg.add(instrumentzahl.get(r.nextInt(instrumentzahl.size())));
```

```
            gueltig = true;
50          }
          }
52      }
        erg.setId(GenerationVerwalter.rhythmenZaehler);
54      GenerationVerwalter.rhythmenZaehler++;
        doku.trageDatensatzEin(erg, letzte);
56    }
    return erg;
58  }
```

Listing 6.5: Methode zur Erzeugung eines Zufallsindividuums erweitert um Muster. Die Variable “al” ist eine HashMap mit Listen von Allelen, die der Anzahl der enthaltenen Instrumente zugeordnet sind, die Variable “doku” verweist auf das Dokumentationsobjekt, die Variable “k” ist das Konfigurationsobjekt und die Variable “r” verweist auf das Objekt des Zufallsgenerators.

6.6.2 Selektion

Die Selektion basiert auf Rängen. Dabei ist vorgesehen sich am Bewertungsmodell zu orientieren. Das Bewertungsmodell sieht vor, dass der Nutzer einen Rhythmus als schlecht, gut oder ausgewählt bewerten kann. Außerdem gibt es die Möglichkeit, dass Individuen unbewertet sind. Daraus ergeben sich drei Ränge, die für die Selektion von Bedeutung sind. Die Ränge bilden die schlechten, die unbewerteten und die guten, inklusive der ausgewählten, in je einer Gruppe pro Generation ab.
Bei Einsatz des “GenerationsuebergreifenderGenerationVerwalter” können zusätzlich Individuen der vorhergehenden Generationen berücksichtigt werden, sofern sie die Bewertung gut, oder ausgewählt, später zugeordnet bekommen.
Das Listing 6.6 zeigt die Methode zur Selektion. Dabei wird berücksichtigt, dass es sein kann, dass Rängegruppen nicht besetzt sind. Des Weiteren ist es wichtig, dass die Eltern nicht identisch sind und dadurch keine Endlosschleife entsteht, falls nur noch identische Individuen zur Auswahl stehen. Dass die Eltern nicht identisch sein sollen ist deshalb wichtig, damit bei einem Crossover nicht zwei identische Nachkommen entstehen.

```
0 /**
  * Die Selektion findet auf Grundlage einer HashMap statt, deren
      Schluesselwerte die Raenge der Individuen darstellen und die Werte die
      Listen der Individuen gleichen Rages.
2 *
  * @param raenge -> Map mit Schluessel: Rang und Wert: Liste mit Individuen des
```

```
4  *                Ranges
   * @return
6  */
   protected Genotyp[] selektiere(HashMap<Integer, ArrayList<Genotyp>> raenge) {
8    // Gruppen ermitteln
     ArrayList<Integer> schluessel = new ArrayList<Integer>(raenge.keySet());
10   // Gruppen sortieren
     Collections.sort(schluessel);
12   Genotyp[] eltern = new Genotyp[2];
     // Selektion Beginn
14   int i = 0;
     // Variable, die die Durchl\"aufe zaehlt, um eine Endlosschleife zu
       vermeiden
16   int durchlauf = 0;
     do {
18     // Wenn es nur eine Ranggruppe gibt
       if (schluessel.size() == 1) {
20       // Es ist also ein beliebiges Individuum von allen zur Verfuegung
       stehenden
         int pos = schluessel.get(0);
22       eltern[i] = raenge.get(pos).get(random.nextInt(raenge.get(pos).size()));
       }
24     // Wenn es nur 2 Ranggruppen gibt
       if (schluessel.size() == 2) {
26       // Zu 80 % Eltern aus der hoeheren Ranggruppe
         if (random.nextInt(10) < 8) {
28         int pos = schluessel.get(1);
           eltern[i] = raenge.get(pos).get(random.nextInt(raenge.get(pos).size())
       );
30         // Zu 20 Prozent aus der niedrigen Ranggruppe
         } else {
32         int pos = schluessel.get(0);
           eltern[i] = raenge.get(pos).get(random.nextInt(raenge.get(pos).size())
       );
34       }
       }
36     // Wenn alle 3 Ranggruppen vorhanden sind
       if (schluessel.size() == 3) {
38       // Zu 80 % aus der hoechsten Ranggruppe
         if (random.nextInt(10) < 8) {
40       int pos = schluessel.get(2);
         eltern[i] = raenge.get(pos).get(random.nextInt(raenge.get(pos).size()));
42       } else {
           // Zu 16 % aus der zweithoechsten Ranggruppe
```

```
44          if (random.nextInt(10) < 8) {
            int pos = schluessel.get(1);
46          eltern[i] = raenge.get(pos).get(random.nextInt(raenge.get(pos).size())
    );
            // Zu 4 % aus der niedrigsten Ranggruppe
48          } else {
              int pos = schluessel.get(0);
50            eltern[i] = raenge.get(pos).get(random.nextInt(raenge.get(pos).size
    ()));
            }
52        }
      }
54    // Erzwingt mindestens einen weiteren Durchlauf der Schleife, um den
    zweiten
      // Elternteil zu bestimmen
56    if (i == 0) {
        eltern[1] = eltern[0];
58    }
      // nachdem der erste Elternteil feststeht, muss nur der zweite noch
    gesucht
60    // werden
      i = 1;
62    // durchlaufzaehler erhoehen
      durchlauf++;
64    // Abfangen der Endlosschleife
      if (durchlauf >= 1000) {
66      Object[] genotypen = bekannteGenerationen.get(random.nextInt(
    bekannteGenerationen.values().size()))
        .values().toArray();
68      eltern[1] = (Genotyp) genotypen[random.nextInt(genotypen.length)];
      }
70  } while (eltern[1].equals(eltern[0])); // Sichergehen, dass die Eltern nicht
     gleich sind
// Selektion Ende
72 return eltern;
}
```

Listing 6.6: Methode zur Selektion von Elternindividuen. Die Variable “random” verweist auf das Objekt des Zufallsgenerators.

6.6.3 Crossoveroperationen

Es wurden vier Crossover-Operationen implementiert. Die grundlegende Variante mit genau einer Stelle, an der die Individuen getrennt werden ist in der Klasse “GAModul” enthalten. Das Listing 6.7 zeigt die Methode, die den Einpunkt-Crossover umsetzt. Dabei wird die Trennstelle zufällig bestimmt, wobei darauf geachtet wird, dass nicht der Wert null oder oder Rhythmuslänge minus eins genommen wird. Diese beiden Werte hätten in dieser Umsetzung zur Folge, dass die Kinder identisch zu den Eltern wären.

```
0  /**
   * Umsetzung des 1-Punkt-Crossover.
2  *
   * @param individuum1 -> erster Elternteil
4  * @param individuum2 -> zweiter Elternteil
   * @param letzte      -> Markierung, ob es die letzte Operation fuer diese
6  *                       Generation ist
   * @param generation  -> Generationsnummer
8  * @return
   */
10 public Genotyp[] kombiniere(Genotyp individuum1, Genotyp individuum2, boolean
       letzte, int generation) {
     Genotyp[] erg = new Genotyp[2];
12   // Trennstelle soll nicht 0 und nicht das letzte Symbol sein, weil sonst die
     // Eltern gleich den Kindern sind.
14   int trennstelle = r.nextInt((individuum1.size() - 1)) + 1;
     erg[0] = new Genotyp(generation, k.getAnzTakte(), k.getTaktart());
16   erg[1] = new Genotyp(generation, k.getAnzTakte(), k.getTaktart());
     // Kopieren des ersten Teils
18   for (int i = 0; i < trennstelle; i++) {
       erg[0].add((Allel) individuum1.get(i).clone());
20     erg[1].add((Allel) individuum2.get(i).clone());
     }
22   // Kopieren des zweiten Teils (Bezugsindividuum ist getauscht)
     for (int i = trennstelle; i < individuum1.size(); i++) {
24     erg[0].add((Allel) individuum2.get(i).clone());
       erg[1].add((Allel) individuum1.get(i).clone());
26   }

28  /*
     * Zaehler richtig setzen
30   */
     erg[0].setId(GenerationVerwalter.rhythmenZaehler);
32   GenerationVerwalter.rhythmenZaehler++;
```

```
    erg[1].setId(GenerationVerwalter.rhythmenZaehler);
34  GenerationVerwalter.rhythmenZaehler++;

36  ArrayList<Integer> positionen = new ArrayList<Integer>();
    positionen.add(trennstelle);
38  doku.trageDatensatzEin(individuum1, individuum2, erg[0], erg[1], positionen,
        letzte);

40  return erg;
}
```

Listing 6.7: Methode Einpunkt-Crossover. Die Variable "r" verweist auf das Objekt des Zufallsgenerators und die Variable "doku" verweist auf das Dokumentationsobjekt.

Die weiteren Crossover-Operationen sind der 3-Stellen-Crossover, in den Formen, dass die Teile die getauscht werden gleich lang sind (symmetrisch) und dass die Teile die getaucht werden unterschiedliche Längen haben können (asymmetrisch).
Die letzte Crossover-Operation ist angelehnt an die Notenwerte und trennt die Individuen so, dass die Viertelnoten (Zählzeiten) miteinander getauscht werden. Der symmetrische 3-Stellen-Crossover entspricht dem an Zählzeiten orientierten Crossover, wenn die Individuen Rhythmen im 4/4-Takt sind.
Diese Operationen sind Teil der Klasse "ErweitertesGAModul".
Das Vorgehen bei den erweiterten Crossover-Operationen entspricht dem Vorgehen beim Einpunkt-Crossover. Erst werden die Trennstellen ausgesucht, dann werden neue Individuen durch Vertauschen der Allele erschaffen, nach dem Vorbild in Listing 6.7.

6.6.4 Mutationsoperationen

Es wurden drei Mutationsoperationen implementiert. Die grundlegende Variante, dass ein Allel zufällig verändert wird ist Teil der Klasse "GAModul". Diese Operation funktioniert so, dass eine Kopie des Individuums angelegt wird, das mutiert werden soll. Danach wird die Bewertung der Kopie auf null gesetzt. Anschließend wird eine Stelle des Genotypen zufällig bestimmt und in der Kopie zufällig durch ein beliebiges Allel ersetzt. In Listing 6.8 ist die Umsetzung der Operation zu sehen.

```
0 /**
  * Ausfuehrung einer einfachen Mutation. Dabei wird genau eine Stelle des
2 * Individuums geandert.
```

```
 *
 * @param individuum -> Individuum, das geandert wird
 * @param al         -> Liste aller moeglichen Allele
 * @param letzte     -> Markierung, ob es die letzte Operation fuer die
 *                      Generation ist
 * @param generation -> Generationsnuummer
 * @return
 */
public Genotyp mutiere(Genotyp individuum, Map<Integer, List<Allel>> al,
    boolean letzte, int generation) {
  Genotyp erg = new Genotyp(individuum);
  // Die Bewertung wird auf Null zurueckgesetzt, ggf. vorhergesagt
  erg.setBewertung(Genotyp.NEUTRALBEWERTUNG);
  erg.setGenerationsNummer(generation);
  // Liste mit Allelen einer bestimmten Laenge bekommen
  List<Allel> allelListe = al.get(r.nextInt(al.size()));
  // Element aus der Liste mit Allelen bestimmter Laenge bekommen
  Allel aenderung = allelListe.get(r.nextInt(allelListe.size()));
  // Position fuer die Aenderung bestimmen
  int pos = r.nextInt(individuum.size());
  erg.set(pos, aenderung);

  erg.setId(GenerationVerwalter.rhythmenZaehler);
  GenerationVerwalter.rhythmenZaehler++;

  ArrayList<Integer> positionen = new ArrayList<Integer>();
  positionen.add(pos);
  doku.trageDatensatzEin(individuum, erg, positionen, letzte);

  return erg;
}
}
```

Listing 6.8: Methode Einpunkt-Crossover. Die Variable “r” verweist auf das Objekt des Zufallsgenerators und die Variable “doku” verweist auf das Dokumentationsobjekt.

Die anderen beiden Operationen mutieren ein Individuum an mehreren Stellen, basierend auf einem Wahrscheinlichkeitswert. Dabei wird die Kopie des Elternindividuums sukzessive durchgegangen und mit dem Wahrscheinlichkeitswert bestimmt, ob dass Allel ersetzt werden soll. Soll das Allel ersetzt werden, so wird in der einen Variante ein Allel zufällig ausgewählt, es besteht also auch die Chance, dass es identisch ist, bei der anderen Variante wird ein Allel

ausgesucht, dass sich von dem zu ersetzenden um genau ein Instrument hinzufügen, oder entfernen unterscheidet. Die Operationen zur Mutation an mehreren Stellen sind Teil der Klasse “ErweitertesGAModul”.

6.7 Nutzerschnittstelle

Um die Interaktionsmöglichkeiten bei einigen GUI-Elementen darzustellen wurden Piktogramme eingeführt, die den Cursor ersetzen und deutlich machen sollen, was getan werden kann.
Des Weiteren gibt es GUI-Elemente, die nur für die entsprechende Nutzergruppe angezeigt werden, oder auf die Nutzergruppe zugeschnitten angezeigt werden.
Für die Nutzergruppe der Informatiker gibt es zusätzlich in der Evolutions-GUI einen Button, zur Anzeige der statistischen Werte. In der Ansicht mit den Ergebnissen gibt es für Informatiker einen zusätzlichen Button, mit dem die Individuen in einem lesbaren Format exportiert werden können.
Die Einstellungsmöglichkeiten für die Parameter der Evolution wurden so gestaltet, dass sie von der entsprechenden Nutzergruppe verstanden werden können und im Rahmen der Zielsetzung des Nutzers nützlich sein können. In Abbildung 6.4 ist der Unterschied zwischen den Einstellungsmöglichkeiten für einen Schlagzeuger und einen Informatiker zu sehen.

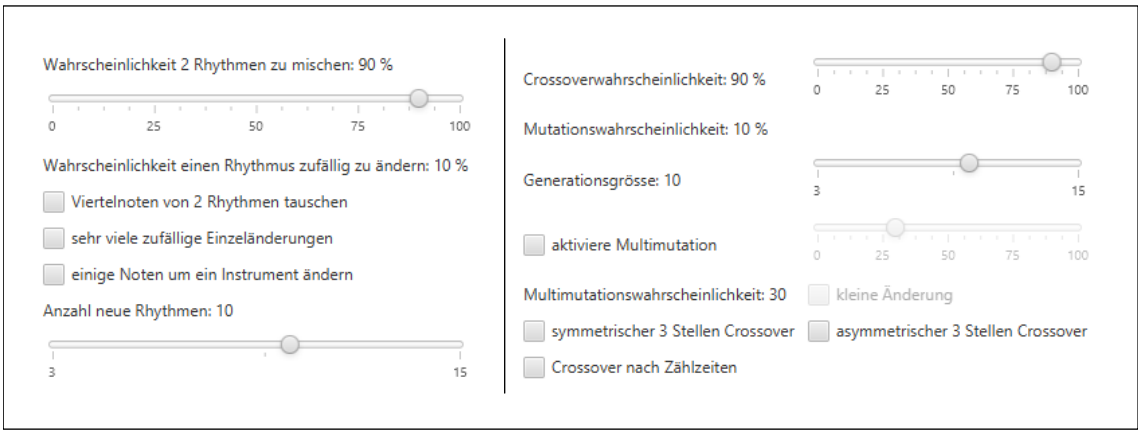

Abbildung 6.4: GUI-Elemente, zur Darstellung der Einstellungsmöglichkeiten. Links Ansicht für einen Schlagzeuger, rechts Ansicht für einen Informatiker.

6.7.1 Rahmen

Die Klassen, die für die Umsetzung des Rahmens der Anwendung genutzt werden befinden sich in dem Paket “rahmen”.

Das Paket “rahmen” ist so strukturiert, dass es für die GUI-Elemente das Paket “GUI”, für Eventhandler, die zur Steuerung des Programmablaufs beitragen das Paket “ablaufEventhandler” und für weitere Eventhandler mit Funktionen, die keinen Einfluss auf den Programmablauf haben, gibt es das Paket “eventhandler”.
Im Paket “rahmen” befindet sich nur die Klasse zum Starten des Programms. Die Klasse “Starter” enthält nur die Main-Methode und erweitert die Klasse “Application”, da es sich um eine JavaFX-Anwendung handelt.
Im Paket “GUI” befinden sich die Klassen die die Benutzerobeflächen des Rahmens bilden. Dazu gehören die Klasse “WorkspaceGUI”, die ein Fenster darstellt, mit dem ein Workspace als Arbeitsgrundlage erstellt oder geladen werden kann.
Die Klasse “VorbereitungGUI” zeigt einem Nutzer die Möglichen Einstellungen für den Fall, dass er einen neuen Workspace erstellt. Mit dieser Benutzeroberfläche zusammen gehört die Benutzeroberfläche “neuesInstrumentGUI”. Diese ermöglicht es in der Phase der Vorbereitung ein eigenes Instrument zum Pool der verfügbaren Instrumente des Workspaces hinzuzufügen.
Die letzte Klasse in dem Paket “GUI” ist die “AbschlussGUI”. Diese zeigt die Ergebnisse der Evolution an. Dabei handelt es sich um die ausgewählten Rhythmen in Form eines Notenbildes und abspielbar als akustisches Signal durch Anklicken eines Buttons.
Die Klassen in dem Paket “ablaufEventhandler” sind für die Übergänge zwischen den GUIs zuständig. Das Namensmuster bei diesen Klassen ist so, dass der Name mit dem Namen des Fensters beginnt, von dem aus zu einem anderen Fenster gewechselt wird. Zum Beispiel wird mit dem “WorkspaceFensterWechselHandler” ein Wechsel des Fensters von der WorkspaceGUI heraus aufgerufen. Abbildung 6.5 zeigt eine Darstellung der Umsetzung des Ablaufs des Programms.

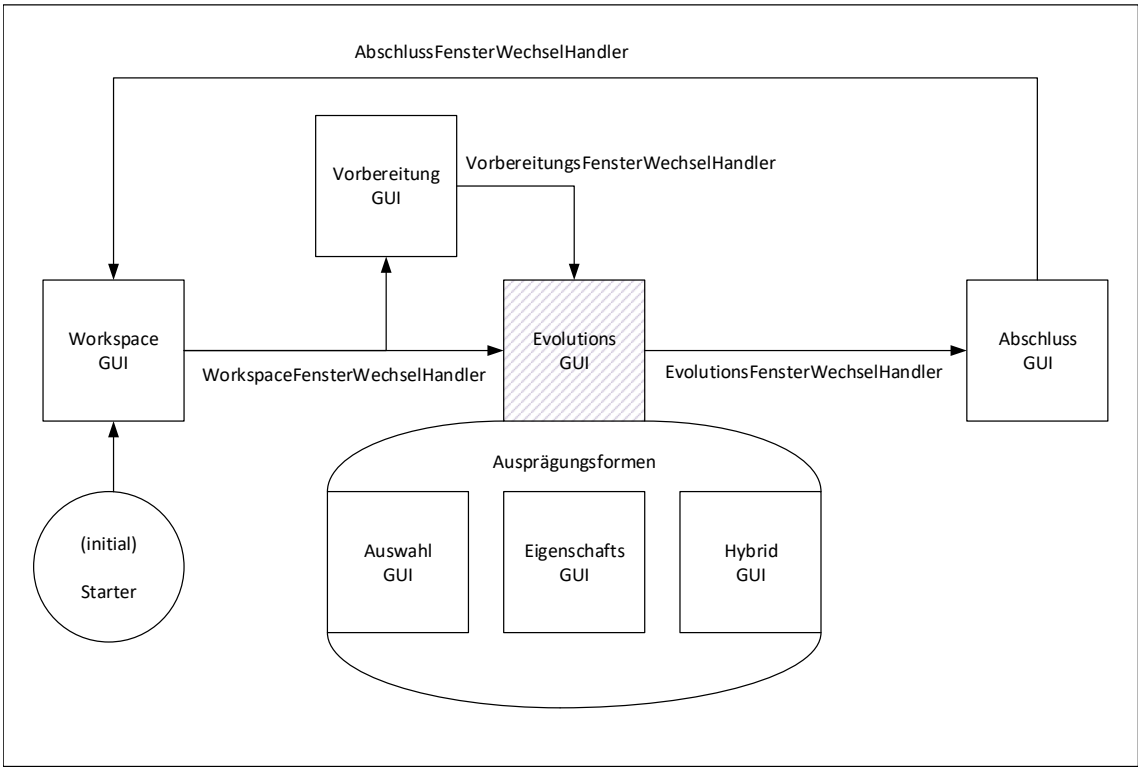

Abbildung 6.5: Darstellung der Ablaufsteuerungskomponenten. Initial wird die "Workspace-GUI" von der Starter-Klasse (keine GUI) aufgerufen. Dann wird die Steuerung von den Ablauf-Eventhandlern übernommen, die über die entsprechende GUI aufgerufen werden können. Die "EvolutionsGUI" wurde hervorgehoben, da diese nicht zum Rahmen der Anwendung gehört. Sie kann in einer der drei Ausprägungsformen im Programmablauf aufgerufen werden. Nicht im Bild zu sehen ist, dass man aus jeder GUI heraus das Programm beenden kann.

6.7.2 Evolutions-GUI

Die Bewertung der Individuen kann mit einer von drei verschiedenen GUIs erfolgen. Die Auswahl der GUI ist über den Parameter "darstellungsGUI" in der Datei "evo.properties" im Workspace möglich. Die Werte können 0 für die "AuswahlGUI", 1 für die "EigenschaftsGUI" und 2 für die "HybridGUI" sein. Der Standardwert ist 2.
Die erste GUI ist die "AuswahlGUI". Diese ist angelehnt an [MI18]. Sie bietet auf der einen Seite die Individuen als Kreise dargestellt und auf der anderen Seite Rechtecke, in die man die Kreise hineinziehen kann, um die Bewertung festzulegen. Die Positionsänderung der Kreise ist dabei permanent, bis der Workspace neu geladen wird. Bei Erzeugung einer neuen Generation verschwinden die Kreise und neue unbewertete Kreise werden erzeugt. Außerdem kann man sich das Notenbild ansehen und eine akustische Ausgabe starten. Abbildung 6.6 zeigt die "AuswahlGUI".

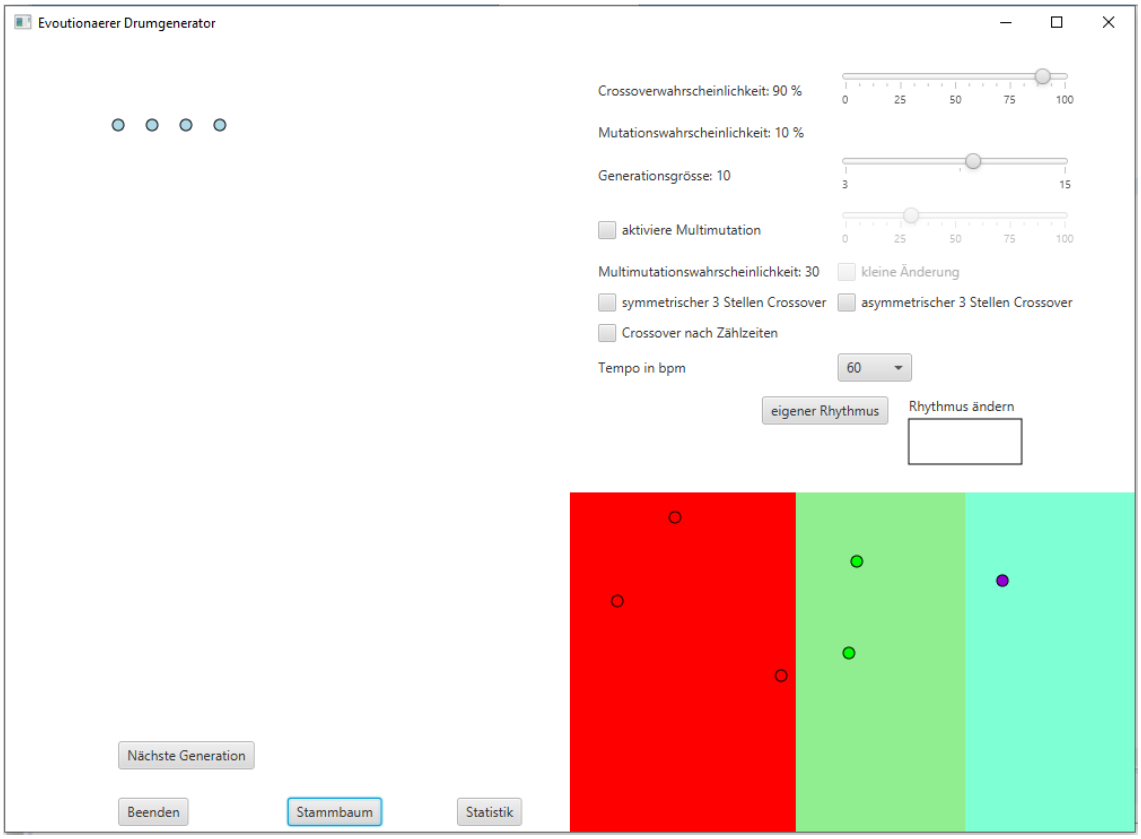

Abbildung 6.6: GUI, die auf Grundlage von den Ideen aus [MI18] erzeugt wurde. Diese Ansicht ist mit den Einstellungsmöglichkeiten für einen Informatiker. In der Abbildung wurden sechs Individuen bewertet und vier Individuen nicht bewertet.

Die zweite GUI ist die “EigenschaftsGUI”. Diese ist angelehnt an die GUI aus [SI11]. Dort kann man festlegen, welche Eigenschaft auf der horizontalen und der vertikalen Ebene als Grundlage für die Position des entsprechenden Kreises genutzt werden soll. Anders als bei der anderen GUI legt man die Bewertung hier nicht durch verschieben fest, sondern durch anklicken eines Buttons. Anschließend ist die Farbgebung des Kreises entsprechend der Bewertung. So entsteht ein optischer Eindruck davon, an welcher Stelle man ein gutes Individuum im Lösungsraum finden kann. Abbildung 6.7 zeigt die “EigenschaftsGUI”.

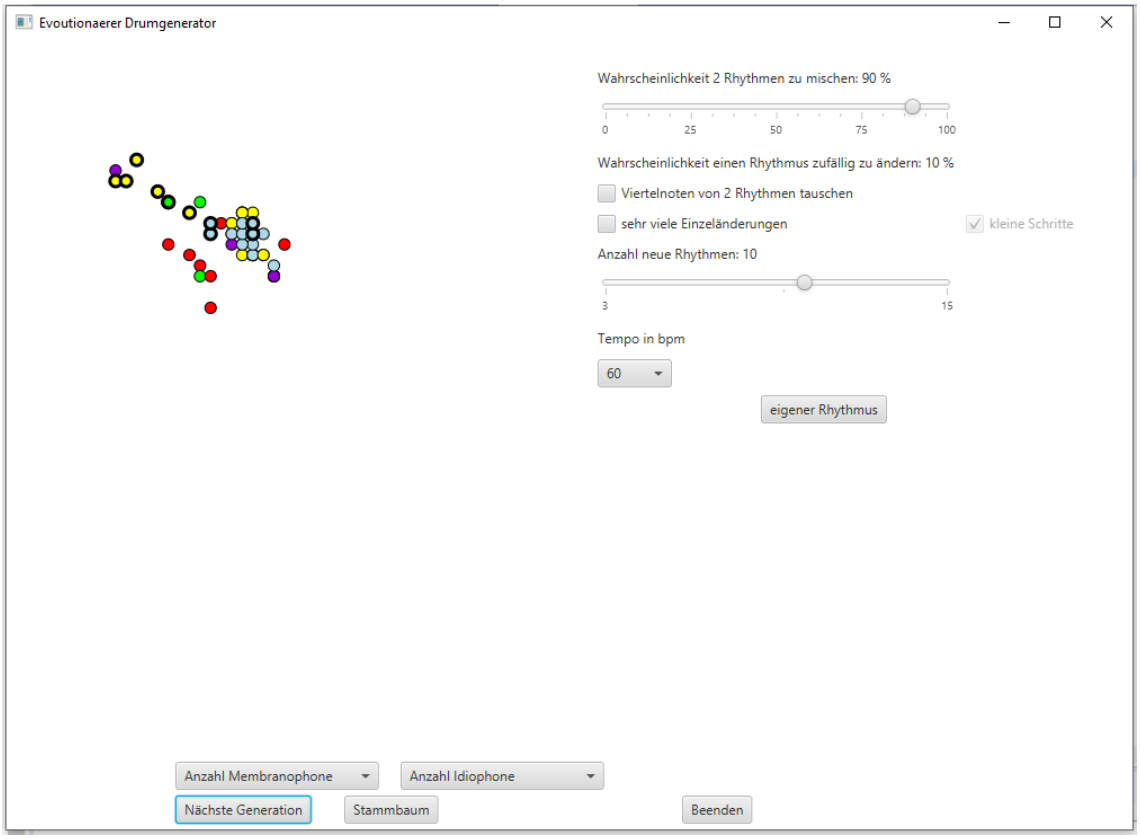

Abbildung 6.7: GUI, die auf Grundlage von den Ideen aus [SI11] erzeugt wurde. Diese Ansicht ist mit den Einstellungsmöglichkeiten für einen Informatiker. Die Kreise sind frei im Raum angeordnet, nach den Parametern, die für die Achsen ausgewählt wurden. Die aktuelle Generation wird mit dickeren Rändern hervorgehoben.

Die dritte GUI ist der Hybrid aus den GUIs eins und zwei und entspricht der Abbildung 5.4 aus der Konzeption. Der grösste Unterschied ist, dass bei dem Rechtsklick auf einen Datenpunkt kein zusätzliches Fenster mit der Notendarstellung erscheint, sondern direkt die akustische Ausgabe startet außerdem wurde die Darstellung der Punkte erweitert. Die Darstellung der Punkte erfolgt nun in einem Gitternetz. Das Gitter ist dabei nicht zu sehen. Diese Darstellung bietet den Vorteil, dass sich keine Punkte überlappen können. Des Weiteren werden Individuen, die gleich sind herausgefiltert, sodass nur ein Datenkreis pro gleichen Individuum dargestellt wird. Abbildung 6.8 zeigt die “HybridGUI”.

Abbildung 6.8: GUI, die auf Grundlage der “AuswahlGUI” und der “EigenschfatsGUI” entstand. Diese Ansicht ist mit den Einstellungsmöglichkeiten für einen Informatiker. Die Kreise sind in einem unsichtbaren Gitternetz angeordnet. Es werden keine identischen Individuen angezeigt.

6.7.2.1 Individuenrepräsentation

Die Individuen werden mit Hilfe von Kreisen auf den GUIs repräsentiert. Dafür gibt es die abstrakte Klasse “DatenKreis”. Von dieser Klasse sind alle spezifischen Datenkreise zur Darstellung in den GUIs abgeleitet.

Die Klasse “DatenKreis” bietet die Bewertungsfarben als konstante Klassenvariablen an. Des Weiteren wird die Notenansicht des Individuums mit Abspielbutton als GUI-Element angeboten. Die Methode “impelementiereListener” wird als abstrakte Methode angeboten, welche im Konstruktor aufgerufen wird. Außerdem gibt es die abstrakte Methode “implementClick”, bei der das Klick-Ereignis implementiert werden soll.

Die Ableitung “BewegbarerDatenKreis” für die “AuswahlGUI” implementiert die Methode “impelementiereListener” so, dass die Methode “implementClick” und eine neue Methode “implementDragAndDrop” aufgerufen wird.

Die Methode “implementClick” erzeugt ein Fenster mit dem Notenbild, bei einem Rechtsklick.

Die Methode “implementDragAndDrop” erzeugt das Verhalten, dass man einen Kreis durch halten der linken Maustaste verschieben kann. Diese Methode ist abhängig von der Klasse “Bewertungsrechteck”. Ein “Bewertungsrechteck” hält die Informationen darüber, welche Bewertung ein Individuum zugeordnet bekommt, das in das Rechteck verschoben wurde und welche Farbe der Datenkreis annimmt, der es repräsentiert. Außerdem bietet es die Methode zur Püfung, ob sich ein Datenkreis komplett in dem entsprechenden “Bewertungsrechteck” befindet.
Die Ableitung “EigenschaftenKreis” ist für die “EigenschaftsGUI” und implementiert die Methode “implementiereListener” so, dass die Methode “implementClick” aufgerufen wird. Die “implementClick”-Methode ist so implementiert, dass ein Klick-Ereignis die Notenansicht öffnet. Die Notenansicht wird von dieser Klasse um drei Buttons zur Bewertung des Individuums erweitert. Außerdem gibt es einen Button um ein Idividuum manuell zu erzeugen, das auf dem aufgerufenen Rhythmus basiert.
Des Weiteren hat die Klasse eine Variable vom Typ Map, mit Strings als Schlüssel und Zahlenwerten als Werte. Dadurch werden die Eigenschaften im Programm repräsentiert, die als Grundlage zur Positionierung des Datenkreises genutzt werden.
Die Ableitung “BeweglicherEigenschaftenKreis” für die “HybridGUI” implementiert die Methode “implementiereListener” auf die gleiche Weise, wie die Klasse “BewegbarerDatenKreis”. Die dafür benötigte Methode “implementDragAndDrop” verhält sich ebenfalls ähnlich, mit dem Unterschied, dass beim Ziehen mit der Maus eine Kopie des Datenkreises entsteht, die verschoben wird. Wenn die Kopie losgelassen wird, dann verschwindet sie und der originale Datenkreis bekommt die entsprechende Farbe zugeordnet.
Die Methode “implementClick” implementiert das direkte Abspielen des Individuums, wenn man mit der rechten Maustaste auf den Datenkreis klickt.
Die Klasse “BeweglicherEigenschaftenKreis” hat ebenfalls eine Variable, mit den Eigenschaften eines Individuums, wie die Klasse “EigenschaftenKreis”.
Die Repräsentation der Individuen im Stammbaum erfolgt ebenfalls mit Hilfe einer von “Datenkreis” abgeleiteten Klasse. Diese Klasse ist die abstrakte Klasse “DokuKreis”.
Die Klasse “DokuKreis” implementiert die Methoden “implementiereListener” und “implementClick” und stellt die Methode “notenAnsicht” als abstrakte Methode zur Verfügung.
Mit der Methode “implementiereListener” wird die Methode “implementClick” aufgerufen. Die Methode “implementClick” implementiert das Verhalten, dass bei einem Rechtsklick die Notenansicht geöffnet wird und bei einem Linksklick die Beziehungen eines Individuums angezeigt werden.
Ein “DokuKreis” kennt die GUI-Elemente, die im Stammbaum seine direkten Nachbarn sind.

Das bedeutet, dass er die Linien kennt, die zu seinen Nachbarn führen und die "DokuKreise", die seine Nachbarn bilden, sortiert, nach Eltern und Kindern.
Des Weiteren verfügt jeder "DokuKreis" über einen Ankerpunkt vorne und hinten. Diese dienen als Start- bzw. Endpunkt für Verbindungslinien im Stammbaum.
Die Klassen, die von "DokuKreis" abgeleitet sind heißen "DokuCrossoverKreis", "DokuMutationsKreis" und "DokuErzeugungKreis". Diese Klassen implementieren die Methode "notenAnsicht" jeweils so, dass die Notenbilder der Eltern und der Kinder dargestellt werden inklusive eines Buttons, der zum Abspielen genutzt wird.

6.7.2.2 Eigenschaftenberechnung

Für die Positionierung der Datenpunkte in der "EigenschaftsGUI" und der "HybridGUI" werden Eigenschaften benötigt. Diese werden mit Hilfe der Klasse "Eigenschaftsberechner" berechnet. Diese Klasse implementiert das "Eigenschaftsmodul"-Interface, das die Methode "berechneEigenschaften" bereitstellt. Mit dieser Methode wird eine "Map" erstellt, die zu einem Individuum alle Eigenschaften enthält.
Die Klasse "Eigenschaftsberechner" berechnet die Eigenschaften, die in Abschnitt 5.9 beschrieben sind. Außerdem wird eine "HashMap" bereitgestellt, die alle Maximalwerte enthält. Die Maximalwerte sind immer von der Anzahl der Takte und der Taktart abhängig. In der Tabelle 6.1 sind die Formeln und die Maximalwerte pro Eigenschaft angegeben.

Tabelle 6.1: Die Tabelle zeigt die Eigenschaften, die von dem Programm für einen Genotyp berechnet werden mit der Formel und dem maximalen Wert für die Eigenschaft.

Eigenschaft	**Berechnung**	**Maximalwert** (max)
Mittlere Belegung	$\frac{\sum Belegungen}{Notenanzahl}$	4
Anzahl Membranophone	zählen	$Notenanzahl \cdot max_{MittlereBelegung}$
Anzahl Idiophone	zählen	$Notenanzahl \cdot max_{MittlereBelegung}$
Anzahl Pausen	zählen	$Notenanzahl$
Anzahl Pausen pro Beat	$\frac{Pausenanzahl}{Zählzeitanzahl}$	4
Membranophoneabstand	$\frac{\sum Membranophoneabstände}{MembranophoneabständeAnzahl}$	$Notenanzahl$
Idiophoneabstand	$\frac{\sum Idiophoneabstände}{IdiophoneabständeAnzahl}$	$Notenanzahl$
Pausenabstand	$\frac{\sum Pausenabstände}{PausenabständeAnzahl}$	$Notenanzahl$
Pausenverhältnis	$\frac{AnzahlPausen}{AnzahlNichtPausen}$	$Notenanzahl$
Mem/Idio Verhältnis	$\frac{AnzahlMembranophone}{AnzahlIdiophone}$	$Notenanzahl \cdot max_{MittlereBelegung}$

6.7.2.3 Datenpunktanzeige

In der “AuswahlGUI” sind die Datenpunkte in mehreren Reihen unabhängig von den Eigenschaften der Individuen die sie repräsentieren angeordnet.
In der “EigenschaftsGUI” sind die Datenpunkte frei in einem begrenzten Raum angeordnet, wobei die Anordnung auf den Eigenschaften des repräsentierten Individuums beruhen.
Für die “HybridGUI” gibt es eine Klasse, die die Anordnung der Datenpunkte so gestaltet, dass es keine Überlappungen geben kann. Diese Klasse ist die “Kreisordner”-Klasse.
Durch die Kreisordnerklasse wird der begrenzte Darstellungsraum in ein Gitter unterteilt, in dem jedes Feld einen Datenkreis groß ist. Zu Beginn sind alle Felder frei. Wenn eine Position für einen Datenkreis ermittelt werden soll, dann wird aus den übergebenen Koordinaten des Kreises die Position im Gitter ermittelt, die den geringsten Abstand hat. Diese Position überschreibt die konkreten Koordinaten des Datenkreises. Außerdem wird diese Position jetzt als nicht mehr verfügbar markiert, wodurch keine übereinanderliegenden Datenkreise entstehen können.

6.7.2.4 eigenes Individuum erzeugen

Zur Erstellung eines eigenen Individuums wurden für den Nutzer zwei Möglichkeiten implementiert. Er kann ein komplett neues Individuum erschaffen, oder auf Grundlage eines bestehenden Individuums ein neues erschaffen. Die zweite Option stellt die Umsetzung einer weiteren Mutationsoperation dar, die komplett vom Nutzer ausgeführt wird.
Zur Erzeugung eines eigenen Individuums, bekommt der Nutzer eine GUI zu sehen, in der er Instrumente für die entsprechenden Positionen in einem Rhythmus festlegen kann. Die GUI zeigt dem Nutzer an, welche Positionen frei, besetzt und nicht mehr besetzbar sind, unter Berücksichtigung der Regeln für die Erzeugung gültiger Rhythmen. Es gibt zwei Buttons in der GUI. Der “speichern”-Button fügt den erzeugten Rhythmus der aktuellen Generation hinzu und dokumentiert die Entstehung. Der “Abspielen”-Button spielt den Rhythmus so oft ab, wie er geklickt wurde. Während des Abspielens kann der Rhythmus auch direkt verändert werden. Abbildung 6.9 zeigt die GUI, die zum Erstellen eines eigenen Rhythmus zur Verfügung steht.

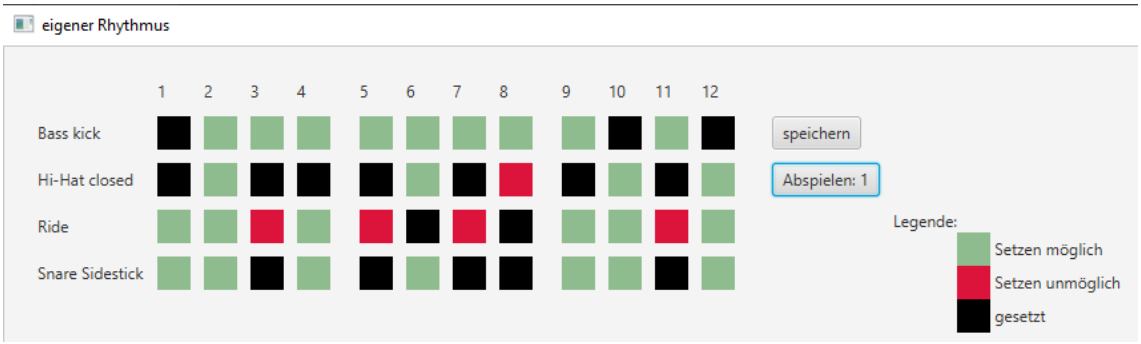

Abbildung 6.9: GUI zur Erzeugung eines eigenen Individuums. Der Abspielen-Button zeigt an, dass der Rhythmus gerade abgespielt wird. Die GUI enthält eine Legende, zur Erklärung der Farbbedeutung.

Wenn das Individuum gespeichert wird, dann wird es auf der Evolutions-GUI markiert und mit gut bewertet, da mit der Erzeugung eines Individuums ausgedrückt wird, dass es Einfluss auf den evolutionären Prozess nehmen soll. Die Dokumentation ist abhängig davon, ob als Grundlage ein anderer Rhythmus genutzt wurde, oder nicht. Wenn ein Rhythmus als Grundlage genutzt wurde, dann wird eine Mutation dokumentiert. Sonst wird eine Erzeugung dokumentiert.

6.8 Stammbaum

Der Stammbaum ist ein Mittel der Dokumentation des Ablaufs des evolutionären Algorithmus. Aus diesem Grund befinden sich alle Klassen die den Stammbaum betreffen in dem Paket “dokumentation”. Die Speicherung der stammbaumrelevanten Daten erfolgt in dem Verzeichnis “Stammbaum” im jeweiligen Workspace in Form einer CSV-Datei.
Eine Zeile in der Datei enthält in eckigen Klammern die Generation als ersten Wert, gefolgt von durch Semikolon getrennten Eintragungen über die Entstehung der Individuen. Als Entstehungsarten gibt es die Erzeugung, die Mutation und den Crossover.
Eine Erzeugung wird bei der Initialisierung verwendet und bei einem durch den Nutzer erzeugten Individuum, wenn dieses nicht auf einem bekannten Individuum basiert.
Die Entstehungsarten Mutation und Crossover haben die Gemeinsamkeit, dass die entstehenden Individuen auf Individuen vorhergehender Generationen basieren.
Die Tabelle 6.2 zeigt die möglichen Einträge für die Entstehungsarten.

Tabelle 6.2: Bezeichner mit Parametern in der CSV-Datei, die den Stammbaum speichert. Die Parameter stehen jeweils in runden Klammern, durch Komma getrennt. Die Bezeichner stehen für Erzeugung (erz), Mutation (mut) und Crossover (cross).

Bezeichner	**Parameter**
erz	ID des entstandenen Individuums
mut	ID Eltern, ID Kind, Mutationsstellen (durch Schrägstrich getrennt)
cross	ID Eltern1, ID Eltern2, ID Kind1, ID Kind2, Crossoverstellen (durch Schrägstrich getrennt)

Die Klasse, die zur Verwaltung der Datei benutzt wird ist die Klasse “StammbaumModul”. Diese erzeugt die CSV-Datei, erzeugt neue Einträge und ermöglicht das Einlesen der Datei, zur Darstellung in der GUI.

Das Einlesen der Datei erfolgt mit Hilfe von Klassen, die das Interface “Operation” implementieren. Die Namen der Klassen sind “Erzeugung”, “Mutation” und “Crossover”. Diese enthalten die entsprechenden Daten zu den Genotypen und den Trennstellen.

Zur Darstellung des Stammbaums gibt es die Klasse “DokuGUI” und die abstrakte Klasse “DokuKreis”, von der die Klassen “DokuErzeugungKreis”, “DokuMutationKreis” und “DokuCrossoverKreis” abgeleitet sind. Die Darstellung erfolgt tabellarisch, wobei die Generationen horizontal abgebildet sind und die Individuen in Form eines Kreises vertikal einsortiert sind. Die Farbe eines Kreises stellt die Bewertung des Individuums dar.

Die Kreise sind durch Linien verbunden, die das Verwandtschaftsverhältnis darstellen. Durch einen Linksklick kann auch das gesamte Verwandtschaftsverhältnis eines Individuums hervorgehoben werden. Ein Rechtsklick zeigt mittels Notenbild, wie das entsprechende Individuum entstanden ist.

An den Kreisen ist ein Zahlenwert, der die Bewertung des Individuums darstellt.

Um die Darstellung etwas schöner zu gestalten, werden generationsübergreifende Verwandtschaftsverhältnisse nicht direkt eingezeichnet, sondern durch einen “DokuKreisLink”. Dieser verweist auf einen Doku-Kreis einer weiter zurückliegenden Generation oder der gleichen Generation und lässt den entsprechenden Kreis aufleuchten, wenn man mit dem Mauszeiger drüber geht. So werden Linien vermieden, die quer durch das ganze Bild gehen und es damit unlesbar machen.

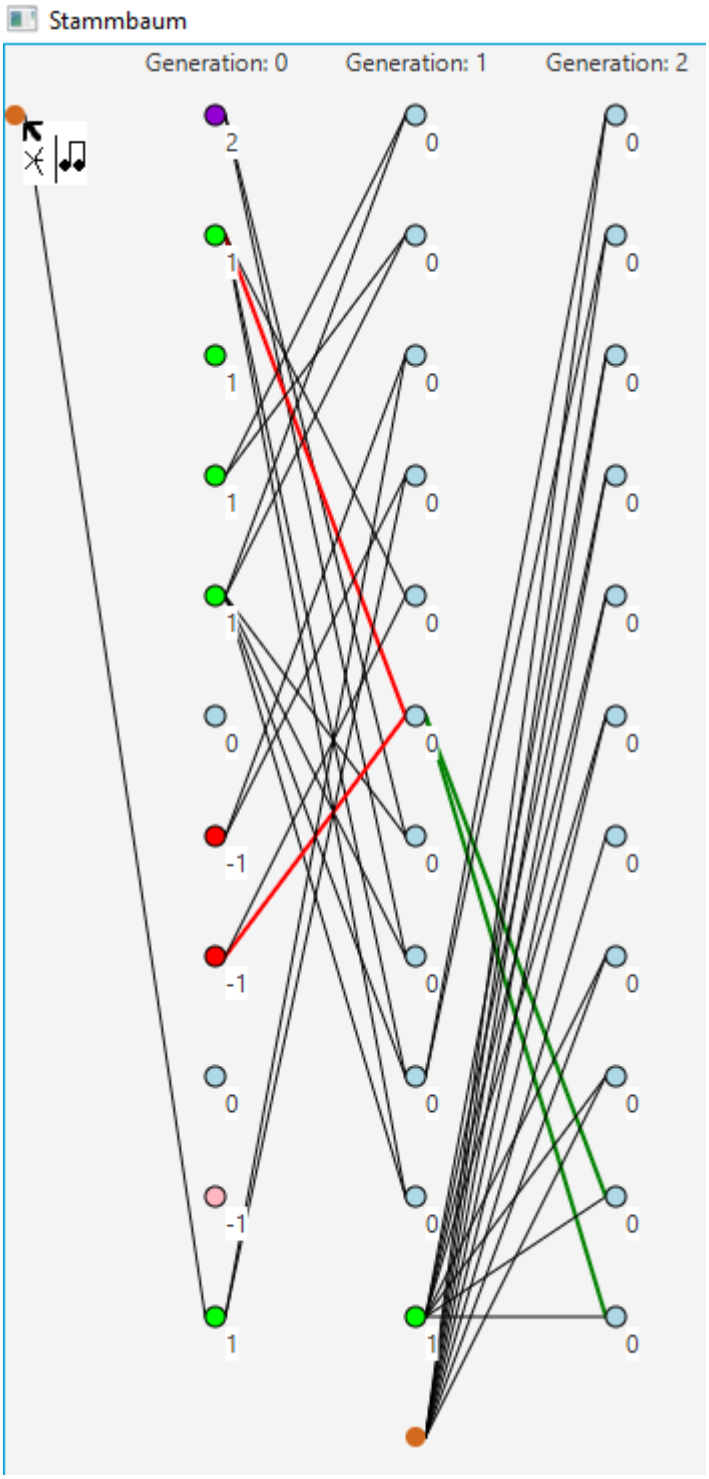

Abbildung 6.10: Dieser Stammbaum zeigt alle Möglichkeiten des Stammbaum auf. Die Farben entsprechen den Bewertungen der Individuen: blau ist ein unbewerteter Rhythmus, rot ein schlecht bewerteter Rhythmus, grün ein gut bewerteter Rhythmus, lila ein sehr gut bewerteter Rhythmus, hellbraun ist ein Link auf einen Rhythmus und rosa markiert den Kreis zu dem Link, wenn man mit dem Cursor über einen Link geht. Die Linien zeigen die Verwandtschaftsverhältnisse. Wenn Linien eingefärbt sind bedeutet rot eine Verbindung zu Vorfahren und grün eine Verbindung zu Nachkommen eines gewählten Individuums. Die Zahlen an den Kreisen sind die Bewertungen der Individuen.

6.9 Statistik

Die Statistik wird durch den Generationsverwalter geführt. In dieser Klasse sind alle relevanten Daten abgreifbar, die einen statistischen Überblick zu dem genetischen Algorithmus ermöglichen. Folgende Daten werden zur Verfügung gestellt.

- Anzahl der Individuen
- Allele im Genpool
- Gesamtzahl Gene
- Anzahl möglicher Allele
- Anzahl der Allele pro Individuum
- Anzahl der möglichen verschiedenen Individuen (Lösungsraum)

7 Evaluation

In diesem Kapitel werden Versuche zur Bewertung, ob die Aufgabenstellung erfüllt wurde, vorgestellt und ausgewertet. Die Versuche haben das Ziel die Qualität dieser Arbeit einschätzbar zu machen.
Im wesentlichen liegt der Fokus darauf, ob man die erzeugten Rhythmen zum Üben nutzen kann, dabei wird auch die Gebrauchstauglichkeit betrachtet in Verbindung mit der Nutzerermüdung, und ob Informatiker mit Hilfe der Demonstrationsanwendung evolutionäre Algorithmen nachvollziehen können.

7.1 Versuch zu kleinen Änderungen

7.1.1 Vorgehen

Die in Abschnitt 4.6 beschriebene Nutzerrückmeldung aus [NHJ15], dass kleine Änderungen interessant sind, da man eine Entwicklung nachvollziehen kann, aber zu kleine Änderungen uninteressant sind, ist Grundlage für das Experiment, zur Bestimmung des Wertes für die Wahrscheinlichkeit der Mutation eines Gens bei der Multimutation mit kleinen Schritten. Durch diesen Wert soll einem Schlagzeuger-Nutzer die Bedienung der Benutzeroberfläche erleichtert werden. Des Weiteren ist es ein Beleg dafür, dass man als Informatiker die Möglichkeit hat mit diesem Programm evolutionäre Algorithmen nachzuvollziehen und mit ihnen zu experimentieren.
Zur Bestimmung des Wahrscheinlichkeitswerts werden vier unterschiedliche Personen gebeten Rhythmen zu bewerten. Die Rhythmen basieren dabei immer auf dem gleichen Standardrhythmus, der in Abbildung 7.1 als Notenbild dargestellt ist. Die Teilnehmer sollen immer die Bewertung vornehmen, wie ähnlich der mutierte Rhythmus zum Standardrhythmus ist und wie interessant die Veränderung ist. Dabei kommt eine Bewertungsskala von eins (schwach) bis sieben (stark) zum Einsatz.

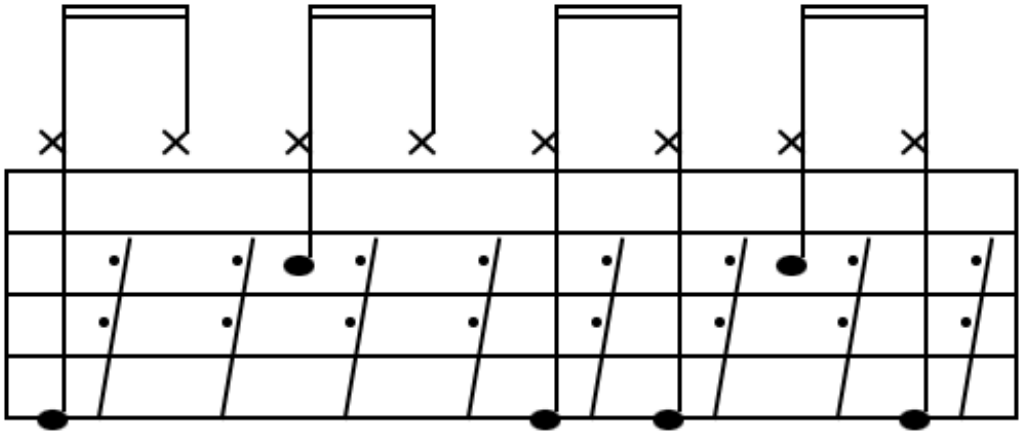

Abbildung 7.1: Notenbild des Standardrhythmus für das Experiment zur Bestimmung der Wahrscheinlichkeit zur Mutation eines Gens bei der Multimutation mit kleinen Schritten.

Des Weiteren wird überprüft, wie oft es keine Veränderung gibt.

Die Einstellungen für die Mutationswahrscheinlichkeit sind in der Tabelle 7.1 dargestellt. Bei den Experimenten werden die Werte 10, 20, 25, 30, 35, 40, 45, 50, 55 und 80 als Wahrscheinlichkeit in Prozent getestet.

Jede Testperson muss dabei drei Rhythmen pro Mutationswahrscheinlichkeitswert bewerten. Damit alle Testpersonen die gleichen Rhythmen bewerten, wird ein Workspace erzeugt und die entsprechenden Rhythmen werden auf Grundlage des Medianwertes zur Anzahl der Veränderungen ausgewählt. Die Testpersonen konnten dabei das Notenbild der Rhythmen nicht sehen. Sie durften immer anfordern, den Standardrhythmus erneut zum Vergleichen zu hören. In Anhang A ist zu sehen, welche Rhythmen zum Hören ausgewählt wurden.

Tabelle 7.1: Relevante Einstellungen für das erste Experiment, zur Bestimmung der Wahrscheinlichkeit zur Mutation eines Gens bei der Multimutation mit kleinen Schritten.

Einstellung	**Wert**
crossoverWahrscheinlichkeitProzent	0
aktiviereMultimutation	true
kleineAenderungenMultiMutation	true
generationsgroesse	10
multiMutationsWahrscheinlichkeitProzent	gesuchter Wert

7.1.2 Ergebnisse

Die Tabelle 7.2 zeigt die Werte für die objektiven Eigenschaften der mutierten Rhythmen. Hier liegt der Fokus darauf, wie viele Rhythmen von dem Standardrhythmus abstammen und wie viele Mutationen in den Rhythmen vorkommen.

Tabelle 7.2: Darstellung der objektiven Eigenschaften der Rhythmen im Experiment. Angegeben sind die Testwerte in Prozent, die Anzahl der Rhythmen, die aus dem Standardrhythmus entstanden sind und dabei nicht identisch sind (in Klammern: Anzahl der Rhythmen inklusive indentische Rhythmen), der arithmetische Mittelwert der Mutationen pro Rhythmus und die geordnete Reihe der Messwerte für die Anzahl der Mutationen pro Rhythmus mit fett markiertem Median und den Nachbarwerten.

Testwert	**Anzahl Rhythmen**	**arithmetisches Mittel Mutationen**	**Median Mutationen**
10	8 (9)	$\frac{16}{9} = 1,7\overline{7}$	0,1,1,**1**,**1**,**1**,2,4,5
20	9 (9)	$\frac{30}{9} = 3,3\overline{3}$	1,3,3,**3**,**3**,**3**,4,5,5
25	9 (9)	$\frac{36}{9} = 4$	2,3,3,**4**,**4**,**4**,5,5,6
30	8 (8)	$\frac{38}{8} = 4,75$	1,3,3,**5**,**5**,**6**,7,8
35	6 (6)	$\frac{38}{6} = 6,3\overline{3}$	5,5,**5**,**6**,**7**,10
40	10 (10)	$\frac{65}{10} = 6,5$	3,4,6,6,**7**,**7**,**7**,7,9,9
45	9 (9)	$\frac{74}{9} = 8,2\overline{2}$	4,7,8,**8**,**8**,**9**,10,10,10
50	10 (10)	$\frac{81}{10} = 8,1$	5,6,7,8,**8**,**9**,**9**,9,10,10
55	9 (9)	$\frac{78}{9} = 8,6\overline{6}$	5,6,6,**7**,**10**,**10**,10,11,13
80	9 (9)	$\frac{110}{9} = 12,2\overline{2}$	11,11,11,**12**,**12**,**13**,13,13,14

Die Tabelle 7.3 zeigt die arithmetischen Mittelwerte der Bewertungen der Testpersonen, zu den Fragen, wie ähnlich sie die Rhythmen im Vergleich zum Standardrhythmus empfanden und wie interessant sie die Veränderung bewerteten.

Tabelle 7.3: Darstellung der Ergebnisse nach der Befragung von vier Testpersonen. Zu jedem Testwert ist das arithmetische Mittel für den Ähnlichkeitswert und den Interessantwert berechnet worden.

Testwert	**Ähnlichkeit**	**Interessant**
10	$5,416\overline{6}$	$3,83\overline{3}$
20	$3,6\overline{6}$	$4,16\overline{6}$
25	$3,5$	$3,583\overline{3}$
30	$3,25$	$4,083\overline{3}$
35	$3,83\overline{3}$	$4,6\overline{6}$
40	$3,916\overline{6}$	$4,6\overline{6}$
45	$2,416\overline{6}$	$4,5$
50	$2,583\overline{3}$	$4,583\overline{3}$
55	2	$4,583\overline{3}$
80	$1,3\overline{3}$	$4,6\overline{6}$

7.1.3 Auswertung

Anhand der Messwerte ist zu sehen, dass die Anzahl der Mutationen im erwarteten Bereich liegen und die Multimutationsoperation damit korrekt funktioniert.
Die Eindrücke der Testpersonen zeigen, dass die Ähnlichkeit mit größerer Mutationswahrscheinlichkeit tendenziell sinkt, auch wenn die Werte bei 35 % und 40 % noch einmal steigen. Die Werte, die beschreiben, wie interessant eine Änderung wahrgenommen wurde, zeigen einen Sprung ab 35 %. Alle Werte davor sind im Bereich 3,8 bis 4,1. Danach sind die Werte im Bereich 4,5 bis 4,7 zu finden.
Auf Grundlage dieser Messwerte wird die Mutationswahrscheinlichkeit für das Programm auf 40 % festgelegt.
Bei diesen Messwerten ist allerdings zu berücksichtigen, dass die Anzahl der befragten Personen nur vier war, wodurch noch keine besonders sicheren Werte ermittelt werden konnten.

7.2 Probenutzung

7.2.1 Vorgehen

Für diesen Versuch bekam ein Testnutzer, der Schlagzeuger ist, das Programm zur Verfügung gestellt. Vorher gab es eine kleine Einweisung in das Programm. Wie es funktioniert und was man damit machen kann. Da sich das Programm noch in der Entwicklung befand, hat

der Testnutzer drei Versionen getestet. Der Testnutzer durfte das komplette Programm ohne Einschränkungen benutzen.
Dabei soll das Programm unter Bedingungen getestet werden, die den gewöhnlichen Lebensbedingungen eines Menschen entsprechen.

7.2.2 Workspace 1

Der Workspace 1 des Testnutzers wurde mit einer Populationsgrösse von 5 initialisiert und einer Musterwahrscheinlichkeit von 80%. Es wurden eintaktige Rhythmen im 4/4-Takt gesucht. Die Instrumente, die genutzt wurden sind “Bass kick”, “Crash”, “Hi-Hat closed”, “Hi-Hat kick”, “Kuhglocke kick”, “Ride”, “Snare”, “Snare flame”, “Splash”, “grosse Tom”, “grosse Tom flame”, “kleine Tom”, “kleine Tom flame”, “mittlere Tom” und “mittlere Tom flame”.
Der Stammbaum zu diesem Workspace ist in Abbildung 7.2 zu sehen. Aus dem Bild und den Einstellungen zur Initialisierung geht hervor, dass zwei Rhythmen in der Generation 0 durch den Nutzer erzeugt worden sein müssen. Des Weiteren wurde in der Generation 9 ein Rhythmus durch den Nutzer erzeugt.

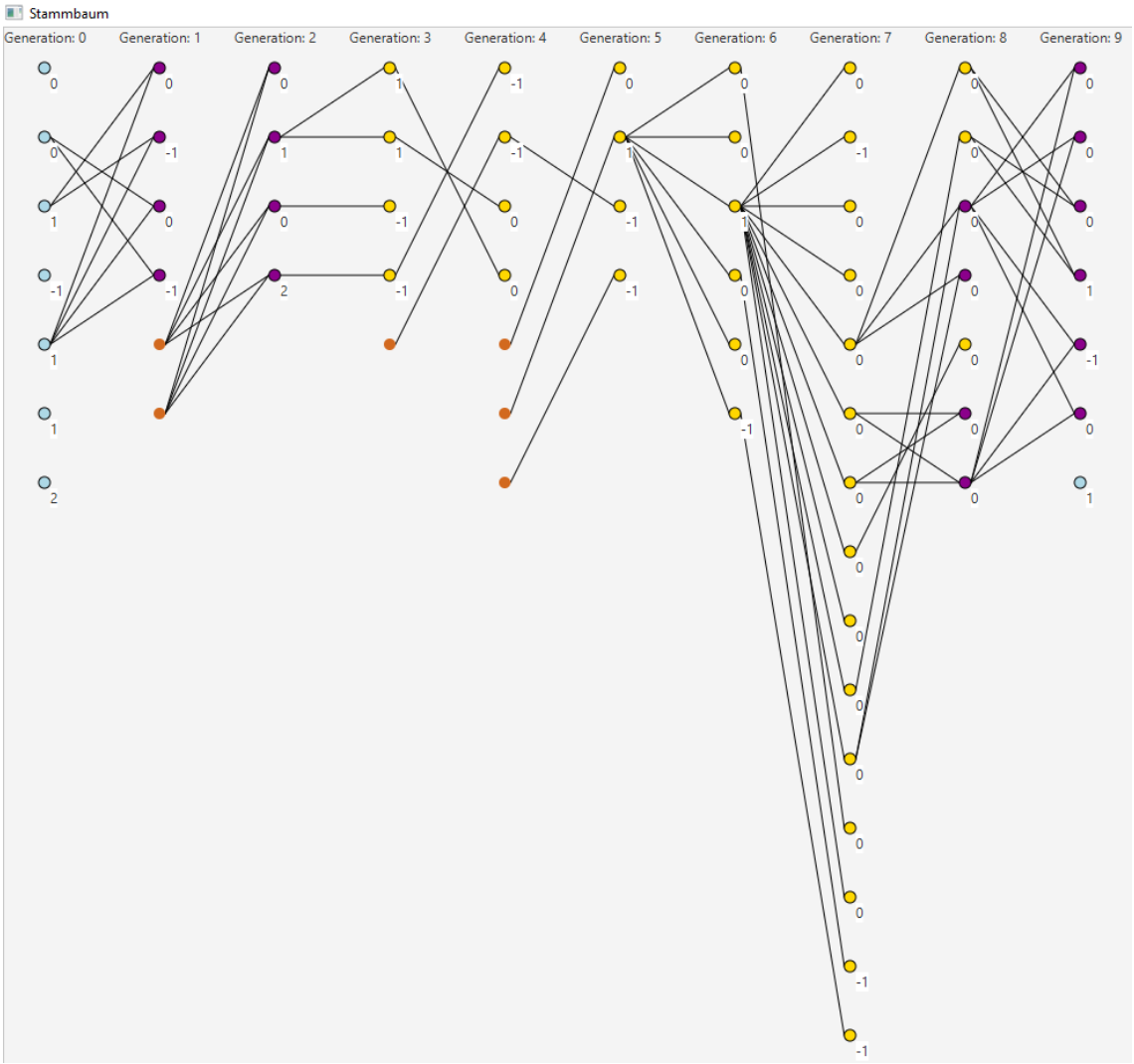

Abbildung 7.2: Stammbaum (ältere Programmversion) zu dem ersten Workspace, den der Testnutzer in seiner Testphase angelegt hat. Die Bedeutung der Farben ist die Art der Entstehung: blau = Erzeugung, lila = Crossover, gelb = Mutation, hellbraun = Link auf einen Kreis einer früheren Generation. Die Zahlen an den Kreisen entsprechen der Bewertung.

Im Stammbaum ist zu erkennen, dass der Nutzer zwei Rhythmen ausgewählt hat, die er selber üben möchte. Einer der Rhythmen wurde in der initialen Generation von ihm selbst erstellt, der andere Rhythmus ist durch Crossover in Generation 2 entstanden.
Des Weiteren ist am Stammbaum zu erkennen, dass der Testnutzer für Generation 3 bis 7 ausschließlich Rhythmen durch Mutation haben wollte.

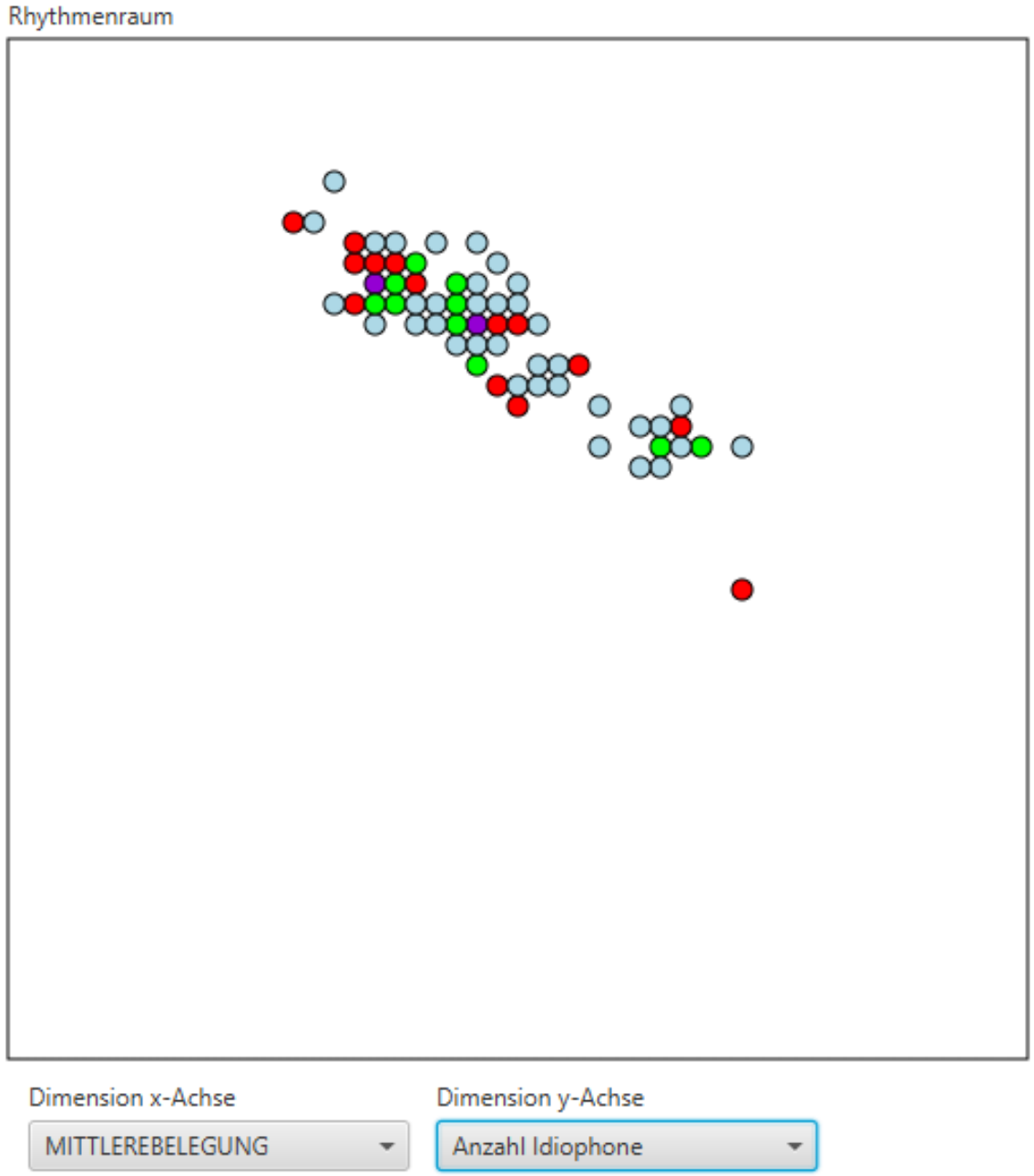

Abbildung 7.3: Rhythmenraum (ältere Programmversion) zu dem ersten Workspace, den der Testnutzer in seiner Testphase angelegt hat. Die Bedeutung der Farben entspricht der Bewertung: blau = nicht bewertet, lila = sehr gut, rot = schlecht und grün = gut.

In Abbildung 7.3 sieht man die Verteilung der Rhythmen im Rhythmenraum.

7.2.3 Workspace 2

Der Workspace 2 des Testnutzers wurde mit einer Populationsgrösse von 3 initialisiert und einer Musterwahrscheinlichkeit von 100%. Es wurden eintaktige Rhythmen im 4/4-Takt

gesucht. Die Menge der Instrumente entspricht der Menge aus Workspace 1.
Der Stammbaum zu dem zweiten Workspace ist in Abbildung 7.4 zu sehen. Hier ist zu erkennen, dass alle Rhythmen von genau einem Rhythmus der Initialisierungsgeneration abstammen. In Generation 7 hat der Nutzer einen eigenen Rhythmus hinzugefügt. Dieser hat erst Einfluss auf die Generation 9 ausgeübt. Der Link in Generation 8 verweist auf den entsprechenden Rhythmus.
Am Stammbaum ist zu erkennen, dass der Nutzer im Vergleich zum Workspace 1 vermehrt Crossover eingesetzt hat.

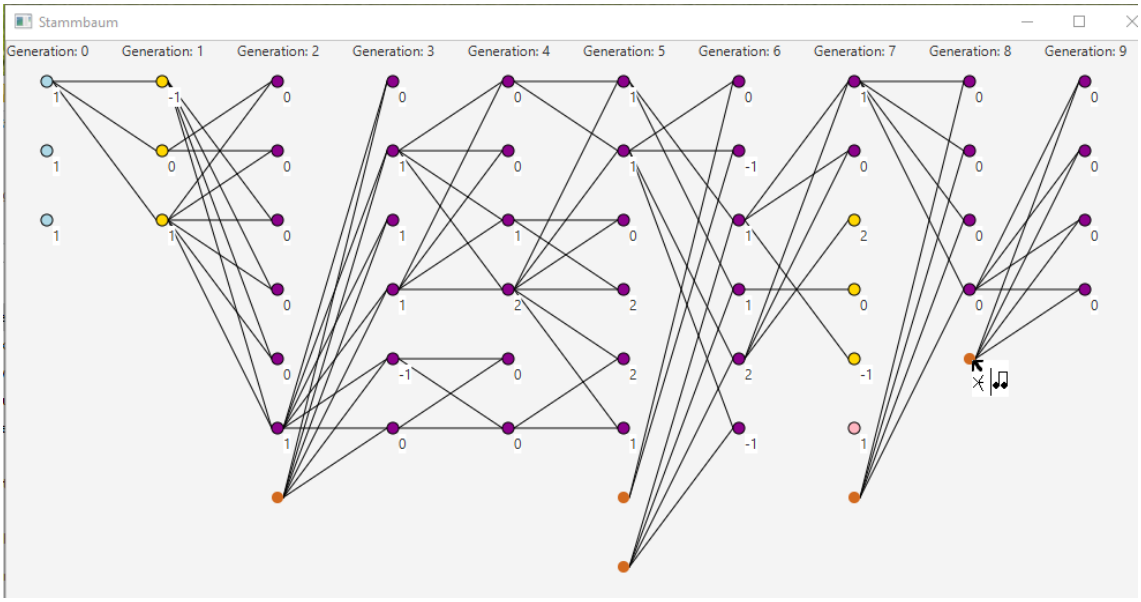

Abbildung 7.4: Stammbaum (ältere Programmversion) zu dem zweiten Workspace, den der Testnutzer in seiner Testphase angelegt hat. Die Bedeutung der Farben ist die Art der Entstehung: blau = Erzeugung, lila = Crossover, gelb = Mutation, hellbraun = Link auf einen Kreis einer früheren Generation, rosa = Kreis, der von dem Link referenziert wird, auf dem sich der Cursor befindet. Die Zahlen an den Kreisen entsprechen der Bewertung.

In der Abbildung 7.5 wird der Rhythmenraum angezeigt. Hier ist zu erkennen, dass fünf Rhythmen vom Nutzer ausgewählt wurden.

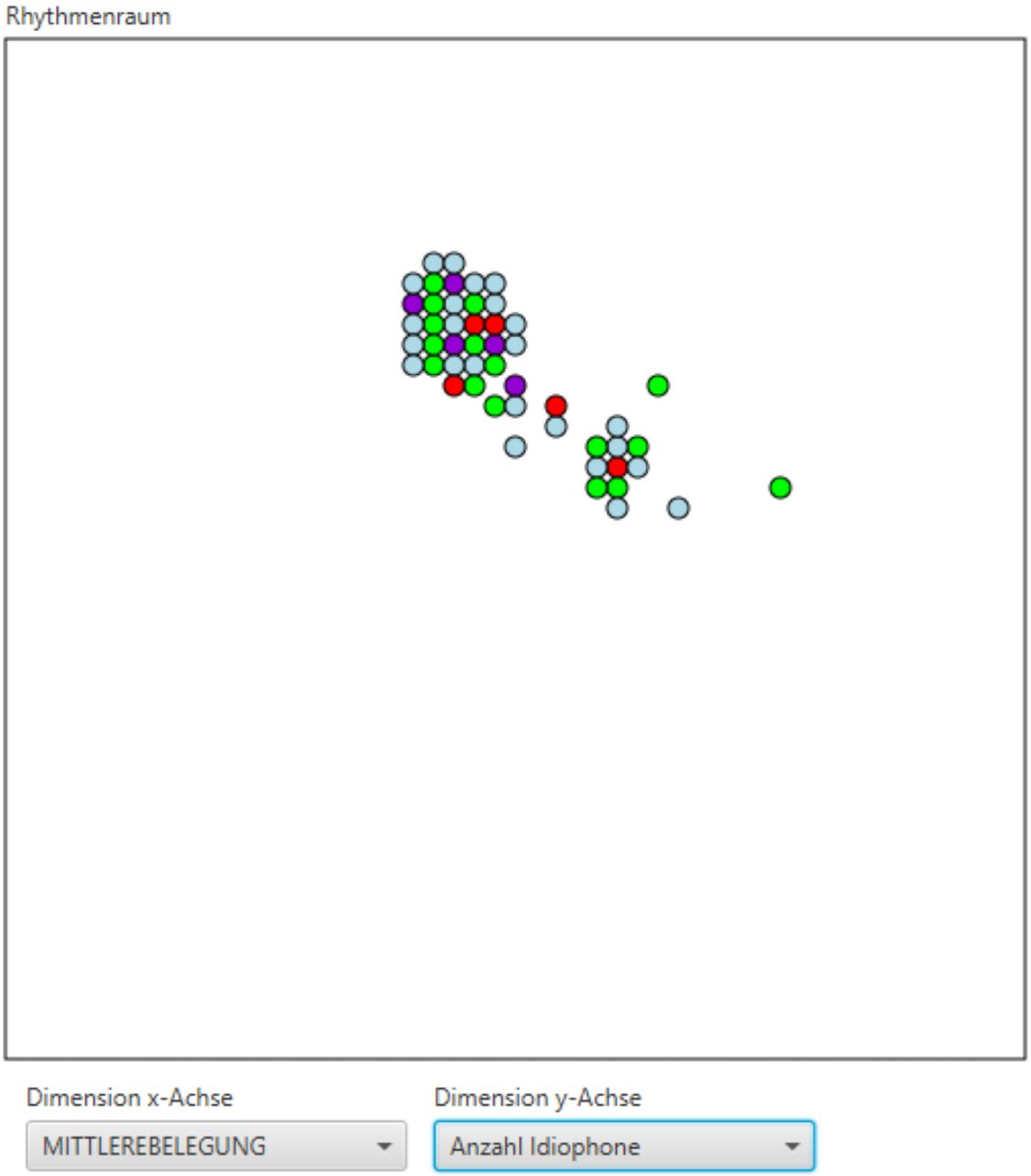

Abbildung 7.5: Rhythmenraum (ältere Programmversion) zu dem zweiten Workspace, den der Testnutzer in seiner Testphase angelegt hat. Die Bedeutung der Farben entspricht der Bewertung: blau = nicht bewertet, lila = sehr gut, rot = schlecht und grün = gut.

7.2.4 Workspace 3

Der Workspace 3 des Testnutzers wurde mit einer Populationsgrösse von 10 initialisiert und einer Musterwahrscheinlichkeit von 80%. Es wurden eintaktige Rhythmen im 4/4-Takt gesucht. Die Menge der Instrumente entspricht der Menge aus Workspace 1 plus “Snare

Sidestick" und "Hi-Hat open", aber ohne "Kuhglocke kick". Zur initialen Generation wurde ein Standardrhythmus hinzugefügt, der zur Erstellung der ersten Generation als einziger Rhythmus durch Multimutation mit kleinen Schritten und einer Mutationswahrscheinlichkeit von 80% genutzt werden sollte. Der Standardrhythmus ist in Abbildung 7.6 zu sehen.

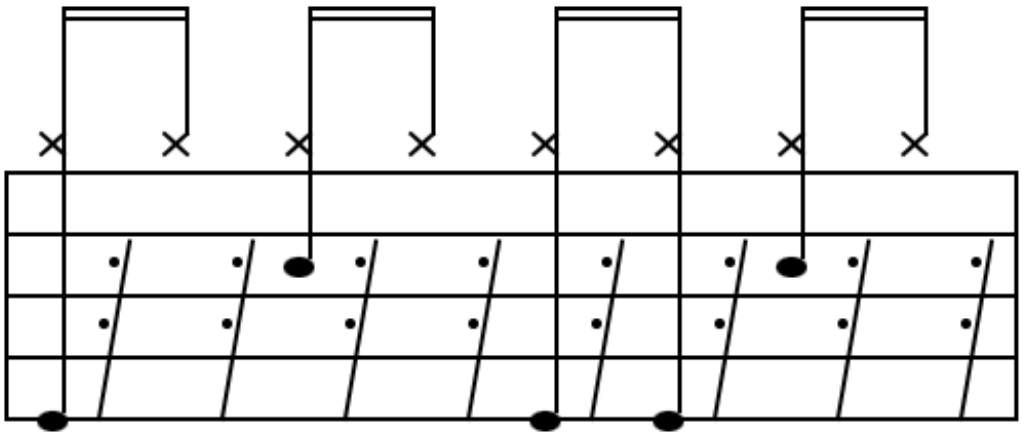

Abbildung 7.6: Standardrhythmus für den dritten Workspace.

Der Stammbaum in Abbildung 7.7 zeigt, dass zur initialen Generation ein Rhythmus hinzugefügt wurde, von dem alle Rhythmen der Generation 1 durch Mutation abstammen. Die durch den Nutzer hinzugefügten Rhythmen in Generation 1 sind durch Veränderungen schon bestehender Rhythmen entstanden (Das ist nicht im Stammbaum zu erkennen, bei dieser Programmversion.). Sieben Rhythmen wurden ausgewählt, als sehr gute Rhythmen. Davon sind zwei direkte Mutationen des Standardrhythmus und einer ist die Mutation der Mutation des Standardrhythmus.

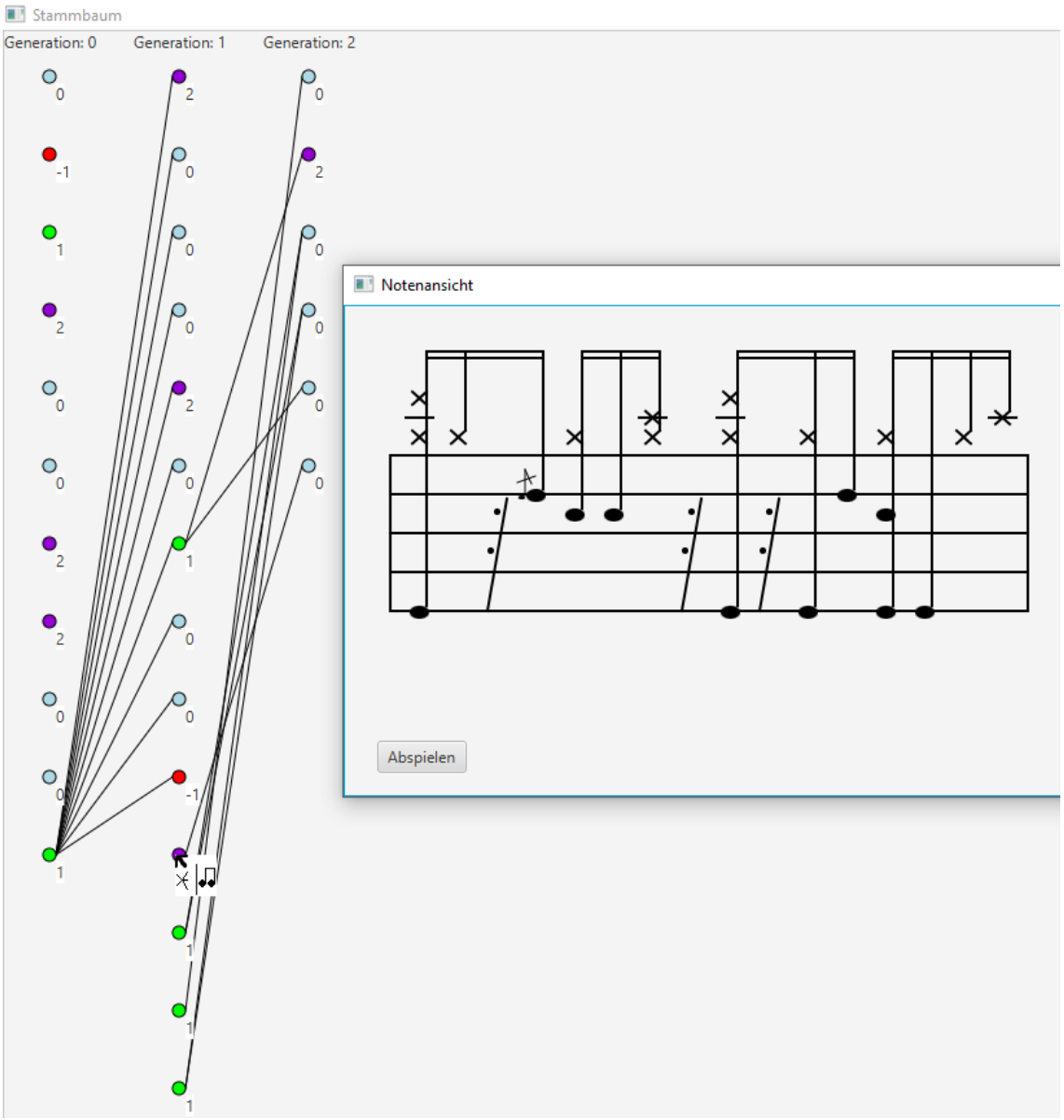

Abbildung 7.7: Stammbaum zu dem dritten Workspace, den der Testnutzer in seiner Testphase unter der Auflage einen Standardrhythmus variieren zu sollen angelegt hat. Die Bedeutung der Farben entspricht der Bewertung: blau = nicht bewertet, lila = sehr gut bewertet, rot = schlecht bewertet, grün = gut bewertet. Die Zahlen an den Kreisen entsprechen der Bewertung. Der Cursor zeigt auf den Rhythmus, dessen Detailansicht in dem überlappenden Fenster zu sehen ist.

In Abbildung 7.8 ist der Rhythmenraum zu sehen. Der Standardrhythmus ist der grüne Kreis ganz links.

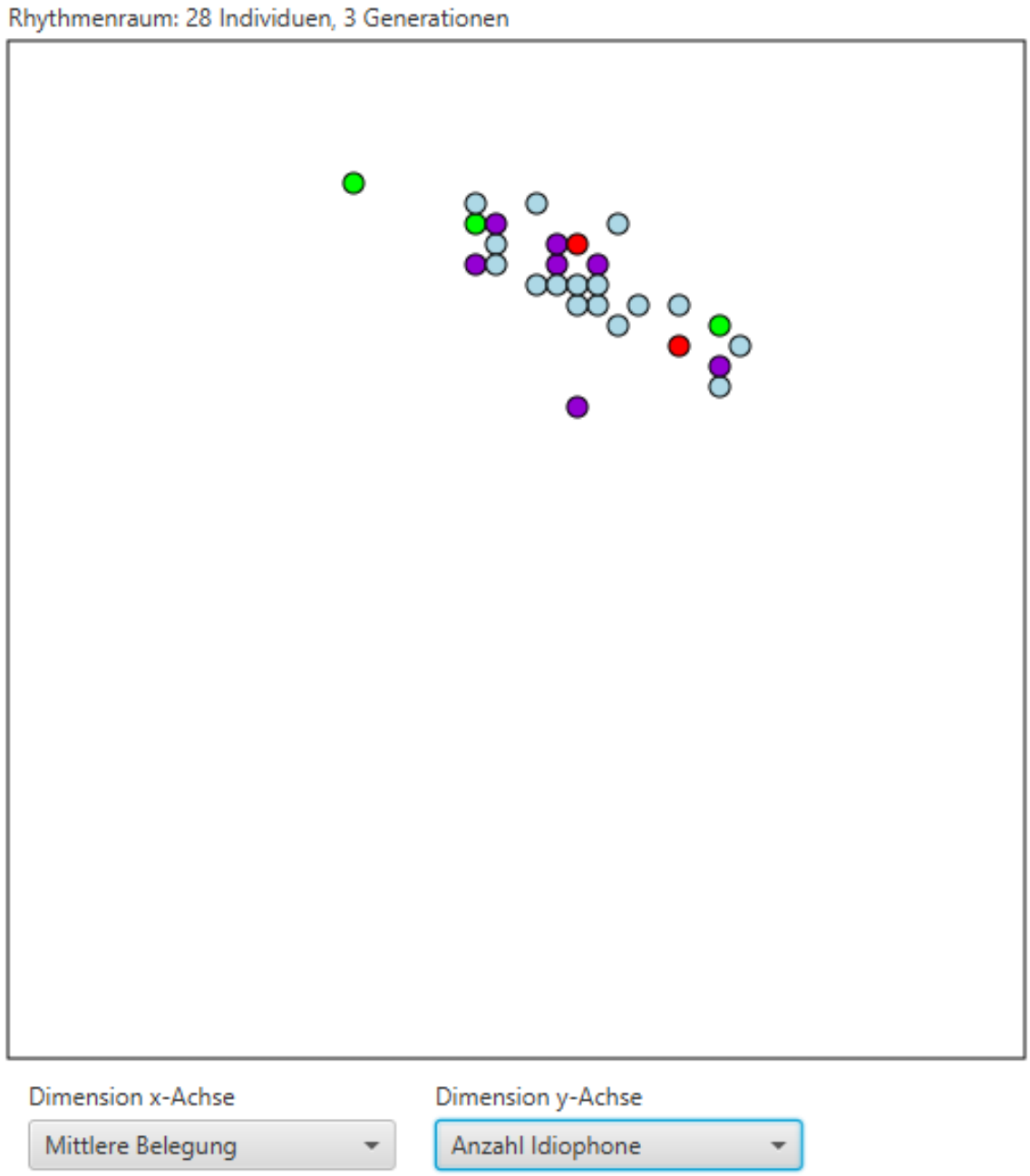

Abbildung 7.8: Rhythmenraum zu dem dritten Workspace, den der Testnutzer in seiner Testphase angelegt hat. Die Bedeutung der Farben entspricht der Bewertung: blau = nicht bewertet, lila = sehr gut, rot = schlecht und grün = gut.

Zu diesem Workspace wurde auch versucht, mindestens einen Rhythmus auf einem echten Schlagzeug zu lernen. Der Rhythmus ist in Abbildung 7.7 mit dem Cursor im Stammbaum markiert und in der Detailansicht geöffnet. Der Rhythmus wurde nachträglich so angepasst, dass er leichter wird zu üben. In Abbildung 7.9 ist zu sehen, wie der Rhythmus verändert wurde. Damit der Rhythmus einfacher wird zu spielen wurden die Splash- und Crash-Becken entfernt.

Nach etwas mehr Übung konnte der Rhythmus auch komplett mit den Splash- und Crash-Becken gespielt werden.

In Anhang B sind die Notenbilder aller ausgewählten Rhythmen zu sehen.

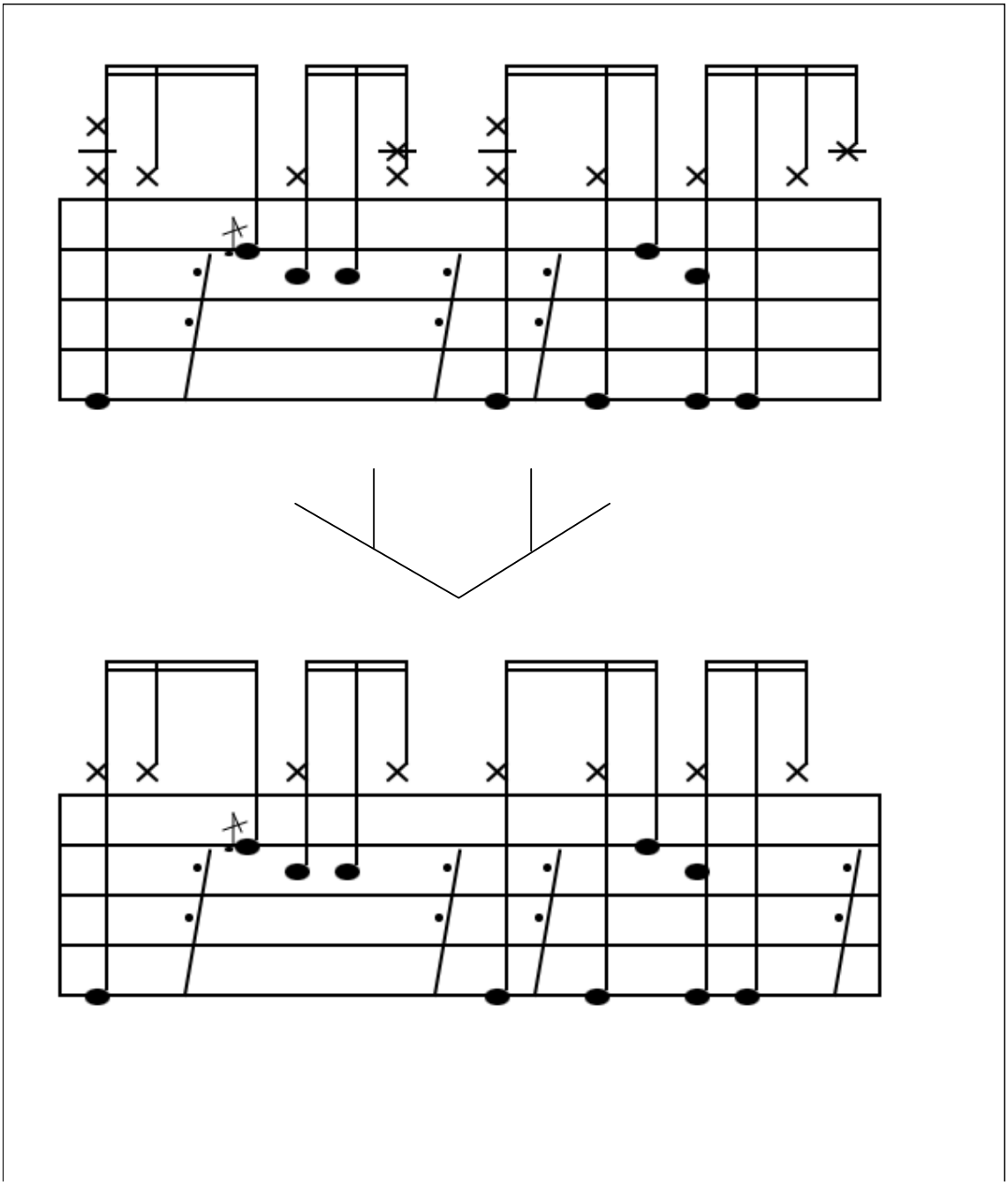

Abbildung 7.9: Rhythmusanpassung: Bei diesem Rhythmus wurden die Splash- und Crash-Becken entfernt, damit er etwas leichter wird zu spielen.

7.2.5 Auswertung

Es gibt drei Workspaces, die in dieser Zeit entstanden sind. Außerdem gibt es wertvolles Feedback, zu dem Programm.
Das Feedback umfasste Verbesserungsvorschläge, die zum größten Teil umgesetzt wurden und es wurde ein allgemeines Feedback zu dem Programm eingeholt.
Das Allgemeine Feedback brachte folgende Kritikpunkte durch den Nutzer hervor:

- Es verging viel Zeit, bis das Programm startklar war. (Die Nutzung eines Laptops stellte offenbar ein Hindernis dar, da das Gerät des Nutzers nicht besonders schnell hochgefahren ist.) Eine Umsetzung auf dem Smartphone wäre besser gewesen, weil das immer direkt da ist.
- Der Prozess neue Rhythmen zu suchen dauert sehr lange und es ist sehr viel Müll dabei.
- Die Bedienung war am Anfang nicht leicht, verbesserte sich mit der Zeit. (Mit den neuen Versionen)
- Das Programm ist nichts, was man unbedingt braucht. (eher ein Luxusprogramm)
- Das Programm fördert die Kreativität
- Die Benutzeroberfläche ist relativ übersichtlich, nicht überladen. (Auch durch Anpassungen über die Versionen nach Feedback)
- Es wurde positiv wahrgenommen, dass es die Erweiterungsfunktion (Instrument hinzufügen) gab. (Wurde allerdings nicht genutzt.)
- Es wurde positiv bewertet, dass man eigene Rhythmen hinzufügen konnte, das förderte die Kreativität. Außerdem wurde es genutzt um Fill-Ins zu erzeugen.
- Einige mathematische Ausdrücke für die Achsenbeschriftungsauswahl waren verwirrend.

An den Workspaces ist zu erkennen, dass die ausgewählten Rhythmen häufig eher zufällig auftreten. Allerdings ist es auch so, dass immer interessante Rhythmen gefunden wurden. Damit wird das Feedback des Testnutzers untermauert, dass das Programm einen Beitrag zur Kreativität leistet. Dass es immer auch Rhythmen gibt die genommen wurden, zeigt, dass die Suche nach Rhythmen zumindest so lange durchgehalten wurde, bis es etwas nutzbares gab.
Der aus Workspace 3 nachgespielte Rhythmus ist auch deshalb gut nachspielbar gewesen, da es sich um eine Mutation des in Workspace 3 verwendeten Standardrhythmus handelte.
In Workspace 1 wurden 62 Individuen in 10 Generationen erzeugt, in Workspace 2 wurden

50 Individuen in 10 Generationen erzeugt und in Workspace 3 wurden 28 Individuen in 3 Generationen erzeugt. An den Stammbäumen der Workspaces lässt sich auch erkennen, dass die Suche nach neuen Rhythmen länger durchgehalten wird, je kleiner die Generationen sind. Das könnte damit zusammenhängen, dass der Nutzer nicht mit sehr vielen neuen Rhythmen auf einmal überschüttet wird.

7.3 Übungstauglichkeit

7.3.1 Vorgehen

Zum Testen, ob sich das Programm zum selbstständig üben eignet wurden Versuche gemacht, bei denen einzelne Funktionen des Programms getestet wurden.
In einem ersten Versuch wurde das Programm mit einer initialen Generation genutzt, die von dem Programm erzeugt wurde. Dabei wurde so lange mit dem Programm gearbeitet, bis ein paar Rhythmen zum Üben gefunden wurden. Anschließend wurde versucht, diese Rhythmen zu lernen. Abbildung 7.10 zeigt die Parameter, die bei der Vorbereitungs-GUI ausgewählt wurden.

Abbildung 7.10: Einstellungen für den ersten Selbstversuch in der Vorbereitungs-GUI.

In einem zweiten Versuch wurde ein Standardrhythmus als Grundlage für die Evolution genutzt. Anschließend wurde das Programm so lange genutzt, bis ein paar Rhythmen gefunden wurden, die als Grundlage zum Üben verwendet werden sollten.

7.3.2 Evolutionsverlauf Versuch 1

Es wurden insgesamt acht Generationen erzeugt, wobei 19 Rhythmen als Favorit ausgewählt wurden. Insgesamt wurden 85 nicht identische Rhythmen von dem Programm erzeugt. In

Abbildung 7.11 ist die Verteilung der Rhythmen in der Ansicht zu sehen, die während der Erzeugung am meisten genutzt wurde.

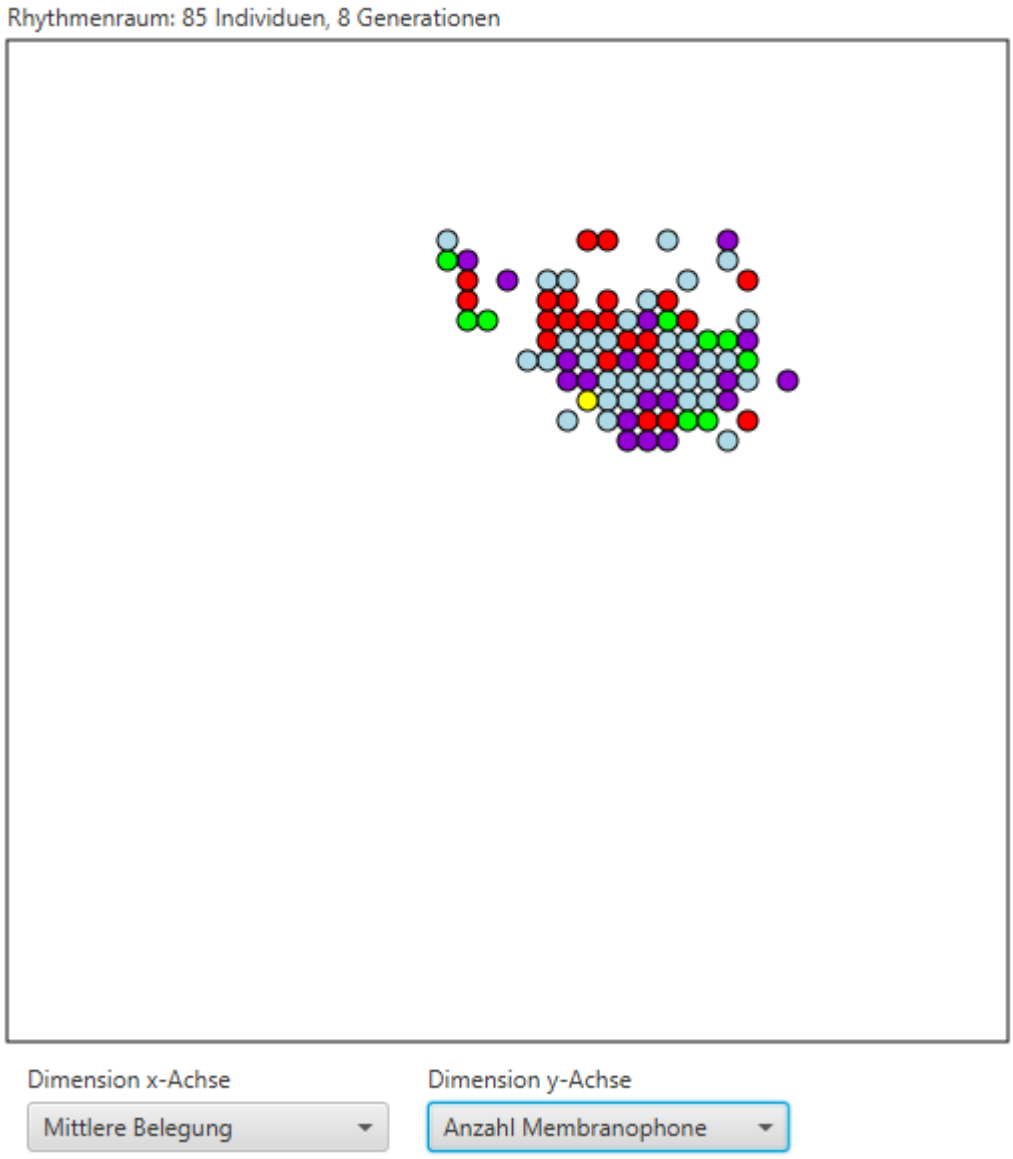

Abbildung 7.11: Rhythmenraum in Versuch 1. Die Farbe rot bedeutet schlecht bewertet, gelb/blau bedeutet nicht bewertet grün bedeutet gut bewertet und lila bedeutet möchte ich üben.

Abbildung 7.12 zeigt den Stammbaum des ersten Versuchs. Zu erkennen ist, dass es hier in jeder Generation mindestens einen Rhythmus gab, der als sehr interessant bewertet wurde.

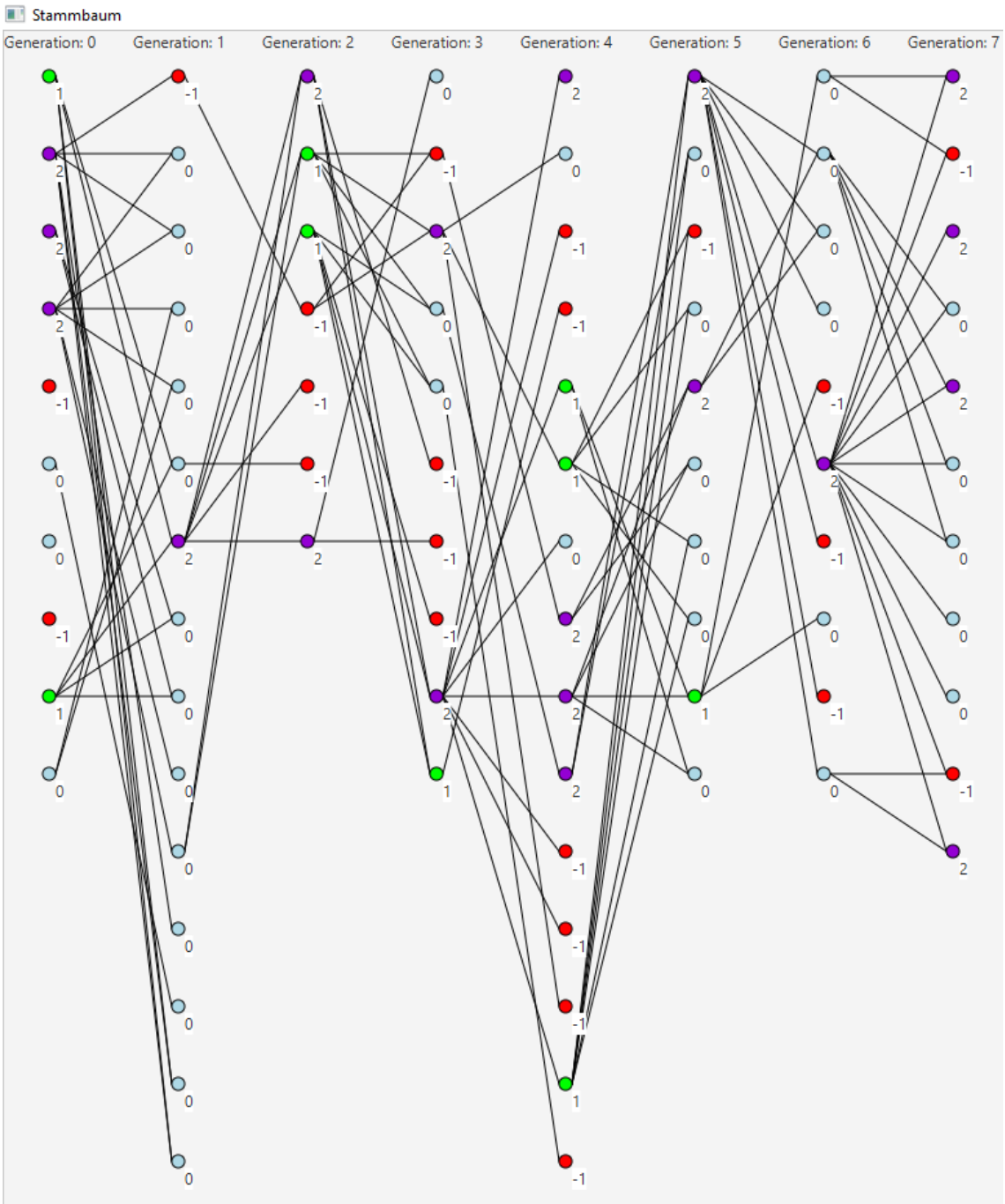

Abbildung 7.12: Stammbaum zum ersten Versuch. Die Farbe rot bedeutet schlecht bewertet, blau bedeutet nicht bewertet grün bedeutet gut bewertet und lila bedeutet möchte ich üben.

Nach Abschluss des Evolutionsprozesses wurden in der Abschluss-GUI noch drei Rhythmen aussortiert. Die Notenbilder der übrigen wurden auf zwei Blättern zusammengefasst und zum üben verwendet. In Anhang C sind die Blätter zu sehen.

7.3.3 Evolutionsverlauf Versuch 2

Im zweiten Versuch wurde ein Grundrhythmus, der in Abbildung 7.13 zu sehen ist, genommen und mit Multimutation in kleinen Schritten mutiert. Der Wert für die Mutationswahrscheinlichkeit eines Allels lag bei 40 %. Der Stammbaum in Abbildung 7.14 zeigt, dass alle Generationen auf dem einen Grundrhythmus basieren. Es wurden sieben Rhythmen ausgewählt zum üben. In Anhang D sind die ausgewählten Rhythmen und der Grundrhythmus zu sehen.

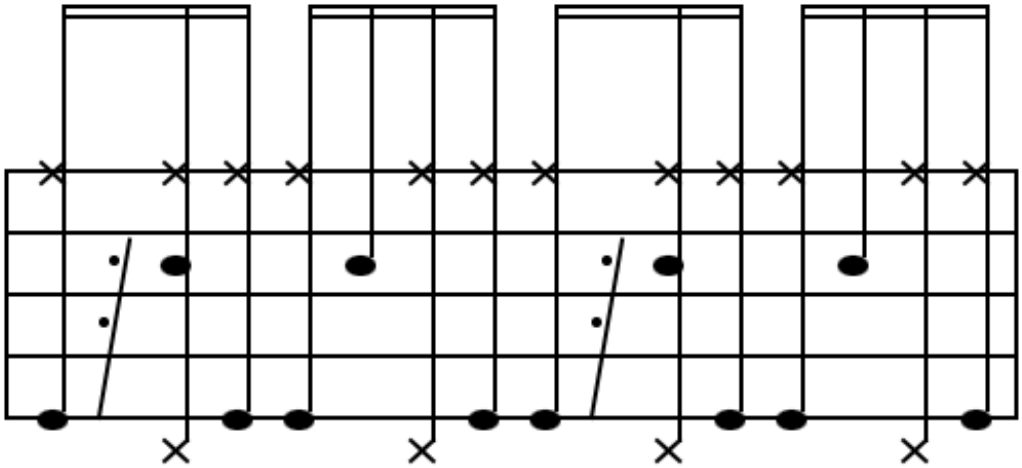

Abbildung 7.13: Grundrhythmus für Versuch 2.

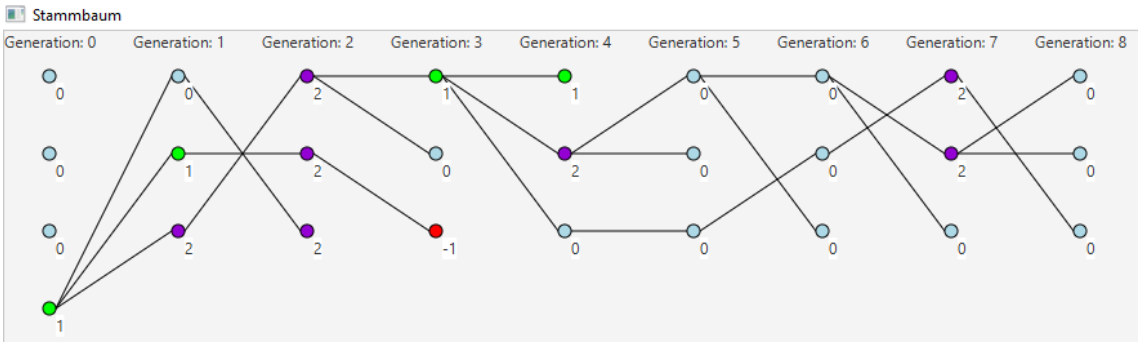

Abbildung 7.14: Stammbaum zum zweiten Versuch. Die Farbe rot bedeutet schlecht bewertet, blau bedeutet nicht bewertet grün bedeutet gut bewertet und lila bedeutet möchte ich üben. Alle Rhythmen wurden durch Multimutation in kleinen Schritten mit 40 % Mutationswahrschinlichkeit erzeugt.

7.3.4 Ergebnis

Bei Versuch 1 waren die Rhythmen sehr unterschiedlich von den Rhythmen, die man gewohnt ist zu spielen. Dadurch fällt das Lernen der Rhythmen nicht leicht und erzeugt Frust beim Versuch des Nachspielens in den ersten Versuchen.
Bei Versuch 2 entstanden alle Rhythmen aus einem Grundrhythmus. Dabei war die Ähnlichkeit der Rhythmen zu einem bekannten Rhythmus sehr hoch, was die Schwierigkeit die Rhythmen nachzuspielen senkte und damit auch die Motivation nicht so sehr sinken ließ.
Der Rhythmus, der direkt aus dem Grundrhythmus entstanden ist und ausgewählt wurde, konnte innerhalb einer Übungsstunde gespielt werden.
Insgesamt sollten neue Rhythmen nicht zu sehr von dem was dem Nutzer bekannt ist abweichen, damit das Lernen der Rhythmen nicht zu schwer ist. Wenn die Änderungen zwischen den Generationen gering ausfallen, dann kann der Verlauf im Stammbaum als Möglichkeit genutzt werden, eine aufbauende Übungsstrategie zu entwickeln, um die Rhythmen zu lernen. Die Fähigkeiten des Schlagzeugers sind allerdings ebenso ein Einfluss auf den Erfolg des Lernens, wie die Ähnlichkeit der Rhythmen.

8 Zusammenfassung und Ausblick

In diesem Kapitel werden die Ergebnisse der Arbeit zusammengefasst und ein Ausblick gegeben mit Ideen für Arbeiten, die darauf aufbauen können.

8.1 Zusammenfassung

Es sollte ein Programm entstehen, das Rhythmen, unter Anwendung interaktiver evolutionärer Algorithmen, erzeugen kann. Des Weiteren sollte die Entwicklung der Individuen nachvollziehbar dargestellt werden. Dazu wurden ähnliche Arbeiten zur Inspiration begutachtet und Grundlagen der evolutionären Algorithmen, sowie der benötigten musikalischen Kenntnisse erläutert.
Die Evaluation stützt sich auf einem Test der Funktionen für Informatiker, mit der ein Wert für die Multimutationsoperation ermittelt wurde. Durch den Stammbaum und die Funktion eigene Individuen zu erzeugen konnte ein entsprechender Versuch gestaltet werden.
Des Weiteren gab es einen Langzeitversuch, bei dem ein Schlagzeuger das Programm in verschiedenen Stadien der Entwicklung getestet hat. Dabei entstand Feedback, zur Verbesserung der GUI, Fehlermeldungen und auch Workspaces, die man auswerten konnte.
Der letzte Versuch zur Evaluation zeigte, dass man das Programm zum selbstständigen üben gut nutzen kann, wenn man sich Rhythmen erzeugen lässt, die ein gewisses Maß an Ähnlichkeit zu den Rhythmen aufweisen, mit denen der Nutzer vertraut ist.

8.2 Ausblick

Das Programm das zu dieser Arbeit entwickelt wurde bietet noch weitere Möglichkeiten interaktive evolutionäre Algorithmen zu untersuchen.
Beispielsweise kann man damit experimentieren, welchen Einfluss das gemeinsame Suchen mit anderen Personen nach neuen Rhythmen hat. Dabei kann man einen Workspace erzeugen, der von einer Person bearbeitet wird und anschließend an eine andere Person weitergegeben

wird. Diese Person arbeitet dann mit dem Workspace und gibt ihn wieder an die erste Person zurück. Nach mehreren Tauschvorgängen kann man die Teilnehmer nach ihren Erfahrungen befragen und auswerten, wie sich der Workspace von Tausch zu Tausch entwickelt hat, wenn man eine Kopie davon separat gespeichert hat. Ähnliche Versuche wurden bereits in anderen Arbeiten durchgeführt, um Lösungen für Gruppen zu finden. Beipielsweise wurden in den Arbeiten [Fuk16, FH16] Versuche mit mehreren Personen, die die Bewertungen für Individuen vornahmen gemacht. Die Hauptprobleme bei solchen Experimenten scheinen zu sein, dass die Geschmäcker der Personen verschieden sind und dass das Arbeitstempo beim Bewerten sehr unterschiedlich sein kann. Das Problem mit dem unterschiedlichen Arbeitstempo würde sich bei einem Vorgehen, mit Workspace-Austausch gegebenenfalls nicht ergeben, da nicht zeitgleich an dem Projekt gearbeitet wird.

Das Programm kann auch zum Testen weiterer Parameterkonfigurationen und Benutzeroberflächen benutzt werden. Für die Tests zu neuen Benutzeroberflächen müsste es entsprechend erweitert werden.

Denkbar sind auch Tests zu anderen Initialisierungsmethoden. Wenn man genug Testpersonen gehabt hat, kann man zur Initialisierung Rhythmen nehmen, die von anderen Personen als Favorit bewertet wurden. Diese Idee wurde beispielsweise für ein System zur Modellierung von Brillen in [SM16] umgesetzt.

Die Daten, die Informatiker exportieren können eignen sich auch dazu näher untersucht zu werden. Wenn man ausreichend Daten gesammelt hat, kann man versuchen Muster zu finden, die beliebt sind. Auf diese Weise könnten die Abwandlungsoperationen so implementiert werden, dass sie basierend auf diesen Mustern Individuen erzeugen. In [XyCMG10] werden beispielsweise so genannte “Gene Sense Units” benutzt um den evolutionären Prozess zu unterstützen, wobei das Anwendungsgebiet Vorhänge designen ist.

Mit Hilfe der exportierbaren Daten kann es auch ermöglicht werden Lerner zu trainieren, die dann angewendet werden, um den Nutzer phasenweise während des evolutionären Prozesses zu ersetzen. Beispielsweise wurden in der Arbeit [WXXY15] Support-Vektor-Maschinen eingesetzt um die Fitnessbewertung durch einen Pseudo Nutzer zu ersetzten.

Literaturverzeichnis

[AEE15] A. E. Eiben, James E. S.: *Introduction to Evolutionary Computing*. Springer-Verlag GmbH, 2015. – ISBN 3662448734

[BHS07] Boersch, Ingo ; Heinsohn, Jochen ; Socher, Rolf: *Wissensverarbeitung: Eine Einführung in die Künstliche Intelligenz für Informatiker und Ingenieure*. Spektrum Akademischer Verlag, 2007. – ISBN 9783827418449

[Dre11] Dreyer, Stephan: *Benutzerschnittstellen für die interaktive Evolution*. Internet. `http://ots.fh-brandenburg.de/downloads/abschlussarbeiten/sa_StephanDreyer.pdf`. Version: 2011. – Zugriff: 03.09.2019

[FH16] Fukumoto, Makoto ; Hatanaka, Takeshi: Parallel distributed Interactive Genetic Algorithm for composing music melody suited to multiple users' feelings. In: *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, IEEE, jun 2016

[Fuk16] Fukumoto, Makoto: Creation of Warning Sound by Vote of Multiple Users Based on Interactive Differential Evolution: Discussion toward Effective IECs Creating of Media Contents Suited to Multiple Users. In: *2016 Joint 8th International Conference on Soft Computing and Intelligent Systems (SCIS) and 17th International Symposium on Advanced Intelligent Systems (ISIS)*, IEEE, aug 2016

[HT00] Hayashida, N. ; Takagi, H.: Visualized IEC: interactive evolutionary computation with multidimensional data visualization. In: *2000 26th Annual Conference of the IEEE Industrial Electronics Society. IECON 2000. 2000 IEEE International Conference on Industrial Electronics, Control and Instrumentation. 21st Century Technologies and Industrial Opportunities (Cat. No.00CH37141)*, IEEE, 2000

[KFV13] Kaliakatsos–Papakostas, Maximos A. ; Floros, Andreas ; Vrahatis, Michael N.: evoDrummer: Deriving Rhythmic Patterns through Interactive Genetic Algorithms. Version: 2013. `http://dx.doi.org/10.1007/`

978-3-642-36955-1_3. In: *Evolutionary and Biologically Inspired Music, Sound, Art and Design*. Springer Berlin Heidelberg, 2013. – DOI 10.1007/978–3–642–36955–1_3, S. 25–36

[Kod] *Drum Notation Key - Legend & Guide*. Internet. https://www.onlinedrummer.com/drum-key/. – Zugriff: 24.07.2019

[MI18] Masuda, Naotake ; Iba, Hitoshi: Musical Composition by Interactive Evolutionary Computation and Latent Space Modeling. In: *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, oct 2018

[NHJ15] Nuanáin, Cárthach Ó ; Herrera, Perfecto ; Jordà, Sergi: Target-Based Rhythmic Pattern Generation and Variation with Genetic Algorithms. In: *Sound and Music Computing Conference 2015*. Maynooth, Ireland, 30/07/2015 2015

[NO14] Nakamura, Hiroshi ; Ono, Satoshi: Video summarization support by Interactive Evolutionary Computation. In: *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS)*, IEEE, dec 2014

[Rhy] *Die Rhythmus-Pyramide*. Internet. https://gitarren.blog/musiktheorie/das-notensystem-basics-1/die-rhythmus-pyramide/. – Zugriff: 26.09.2019

[SI11] Saito, Yuri ; Itoh, Takayuki: MusiCube. In: *Proceedings of the 2011 Visual Information Communication - International Symposium on - VINCI '11*, ACM Press, 2011

[SM16] Seyama, Takahito ; Munetomo, Masaharu: Development of a multi-player interactive genetic algorithm-based 3D modeling system for glasses. In: *2016 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, jul 2016

[Tak01] Takagi, H.: Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. In: *Proceedings of the IEEE* 89 (2001), Nr. 9. http://dx.doi.org/10.1109/5.949485. – DOI 10.1109/5.949485

[VFC15] Vargas, Francisco V. ; Fuster, Jose A. ; Castanon, Cesar B.: Artificial musical pattern generation with genetic algorithms. In: *2015 Latin America Congress on Computational Intelligence (LA-CCI)*, IEEE, oct 2015

[VLN+16] Vogl, Richard ; Leimeister, Matthias ; Nuanáin, Cárthach Ó ; Jordà, Sergi ; Hlatky, Michael ; Knees, Peter: An Intelligent Interface for Drum

Pattern Variation and Comparative Evaluation of Algorithms. In: *Journal of the Audio Engineering Society* 64 (2016), aug, Nr. 7/8, S. 503–513. `http://dx.doi.org/10.17743/jaes.2016.0016`. – DOI 10.17743/jaes.2016.0016

[Wei15] Weicker, Karsten: *Evolutionäre Algorithmen*. Vieweg+Teubner Verlag, 2015 `https://www.ebook.de/de/product/24198849/karsten_weicker_evolutionaere_algorithmen.html(Zugriff:16.01.2019)`. – ISBN 3658099577

[WXXY15] Wei, Bo ; Xia, Xuewen ; Xie, Chengwang ; Yu, Fei: An interactive evolutionary algorithm with adaptive evaluation mechanism. In: *The 27th Chinese Control and Decision Conference (2015 CCDC)*, IEEE, may 2015

[XyCMG10] Xiao-yan, Sun ; Chen, Jian ; Ma, Xiaoping ; Gong, Dunwei: Grid-based knowledge-guided interactive genetic algorithm and its application to curtain design. In: *2010 Second World Congress on Nature and Biologically Inspired Computing (NaBIC)*, IEEE, dec 2010

[YNOF11] Yamamoto, Ryota ; Nakashima, Shuta ; Ogawa, Shintaro ; Fukumoto, Makoto: Proposal for Automated Creation of Drum's Fill-In Pattern Using Interactive Genetic Algorithm. In: *2011 International Conference on Biometrics and Kansei Engineering*, IEEE, sep 2011

Abbildungsverzeichnis

Tabellenverzeichnis

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit zum Thema:

Anwendung interaktiver evolutionärer Algorithmen zur Erzeugung von Schlagzeugrhythmen

selbstständig angefertigt und ausschließlich die angegebenen Hilfsmittel und Literatur verwendet habe. Die Arbeit hat in dieser oder ähnlicher Form noch keinem anderen Prüfungsausschuss vorgelegen.

Brandenburg, den 14. Oktober 2019

Mario Kaulmann

A Workspacebeschreibung zu Versuch 1

Abbildung A.1 zeigt den Stammbaum, aus dem Workspace, der für Versuch 1 benutzt wurde. Diese Bild zeigt die Anzahl der Mutationen pro Rhythmus und markiert die Individuen gelb, die allen Teilnehmern zu Beurteilung vorgespielt wurden.
Generation 1 entspricht den Rhythmen mit 10 % Änderungswahrscheinlichkeit, Generation 2 entspricht den Rhythmen mit 20 % Änderungswahrscheinlichkeit, Generation 3 entspricht den Rhythmen mit 25 % Änderungswahrscheinlichkeit, Generation 4 entspricht den Rhythmen mit 30 % Änderungswahrscheinlichkeit, Generation 5 entspricht den Rhythmen mit 35 % Änderungswahrscheinlichkeit, Generation 6 entspricht den Rhythmen mit 40 % Änderungswahrscheinlichkeit, Generation 7 entspricht den Rhythmen mit 45 % Änderungswahrscheinlichkeit, Generation 8 entspricht den Rhythmen mit 50 % Änderungswahrscheinlichkeit, Generation 9 entspricht den Rhythmen mit 55 % Änderungswahrscheinlichkeit und Generation 10 entspricht den Rhythmen mit 80 % Änderungswahrscheinlichkeit.

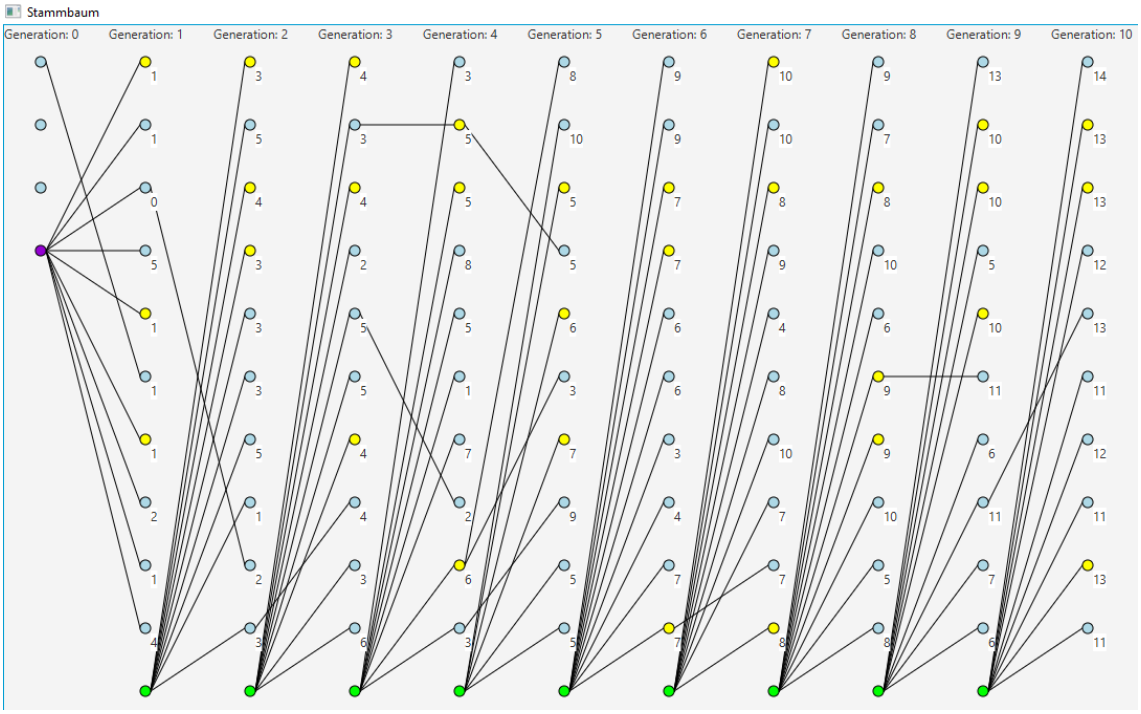

Abbildung A.1: Stammbaum zum Versuch, um die Ähnlichkeit zu bestimmen. In diesem Stammbaum stehen neben den Individuen die Zahlen, die angeben, wie viele Mutationen es gab. Die gelbe Farbmarkierung zeigt, dass ein Rhythmus zum Beurteilen genutzt wurde.

In Abbildung A.2 sind die Bewertungszettel für die Rhythmen zu sehen, die die Testpersonen ausgefüllt haben.

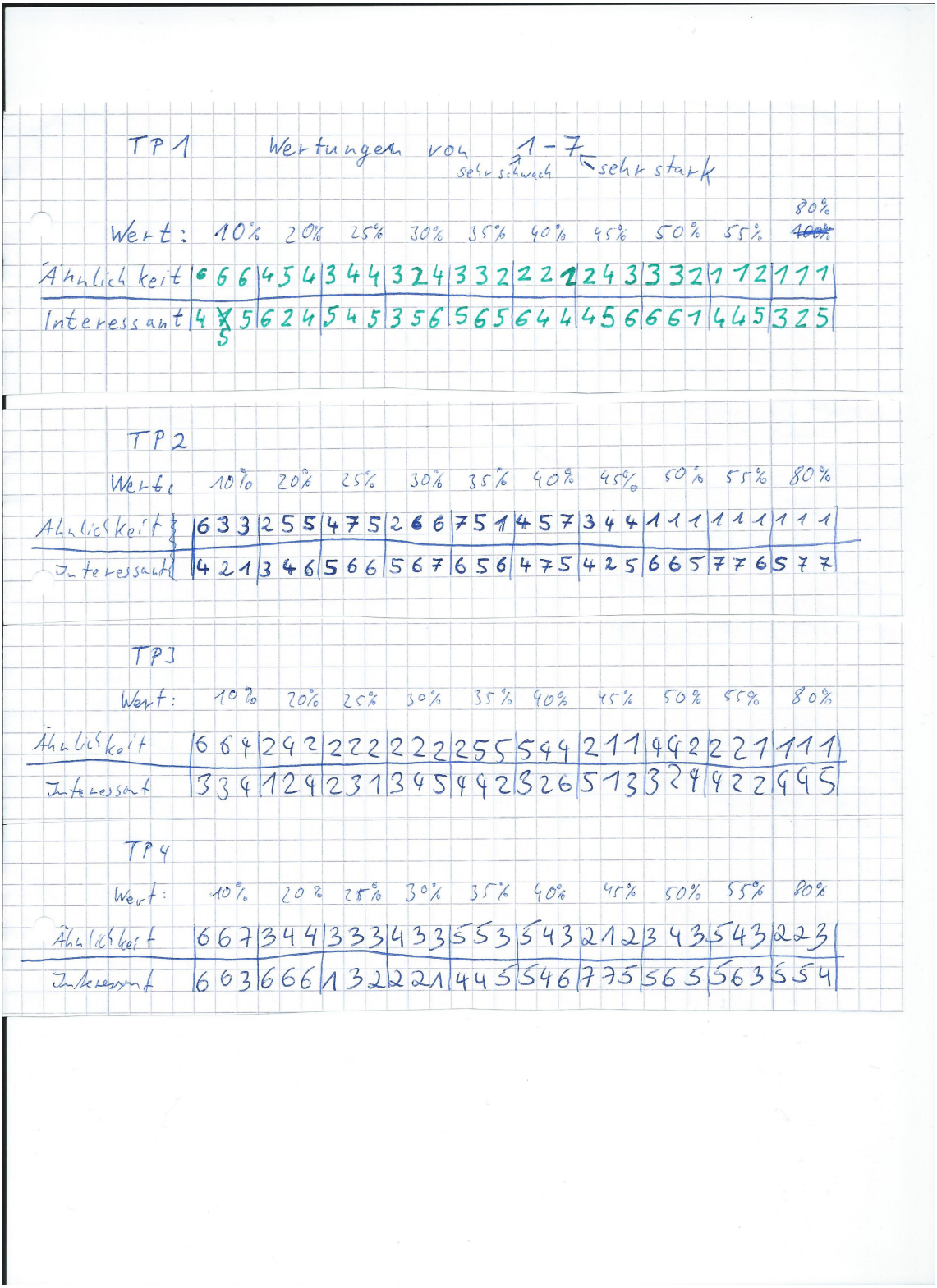

Abbildung A.2: Zettel mit den Eintragungen der Nutzer zur Befragung nach der Ähnlichkeit der Rhythmen.

B Rhythmen zu Workspace 3 in Versuch 2

Abbildung B.1 zeigt die Notenbilder, die zu den ausgewählten Rhythmen aus Versuch 2 Workspace 3 gehören.

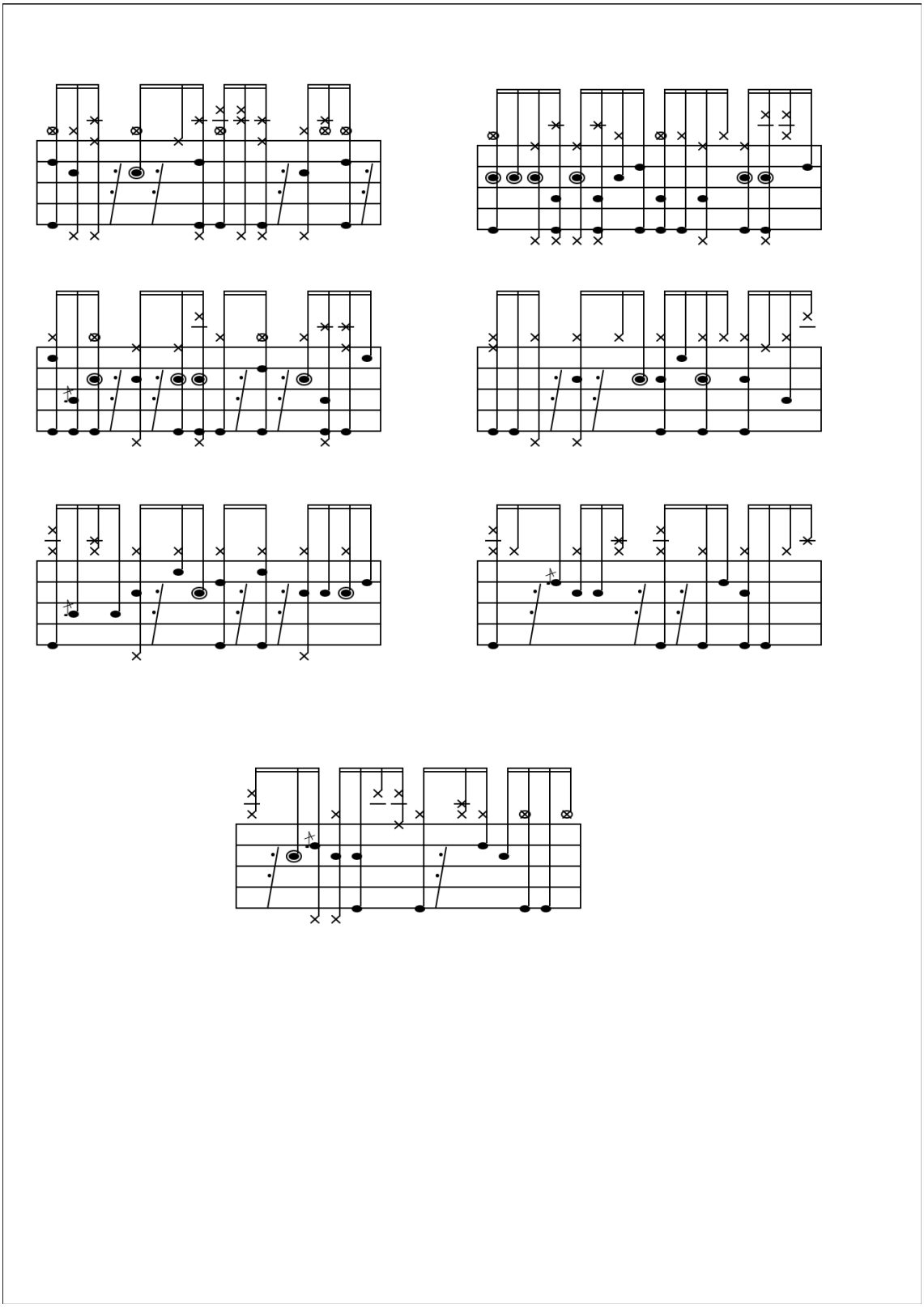

Abbildung B.1: Notenbilder der ausgesuchten Rhythmen aus Versuch 2 Workspace 3.

C Rhythmen zu Selbsttest 1

Abbildung C.1 und Abbildung C.2 zeigen die Notenbilder, die zu den ausgewählten Rhythmen aus Selbstversuch 1 gehören.

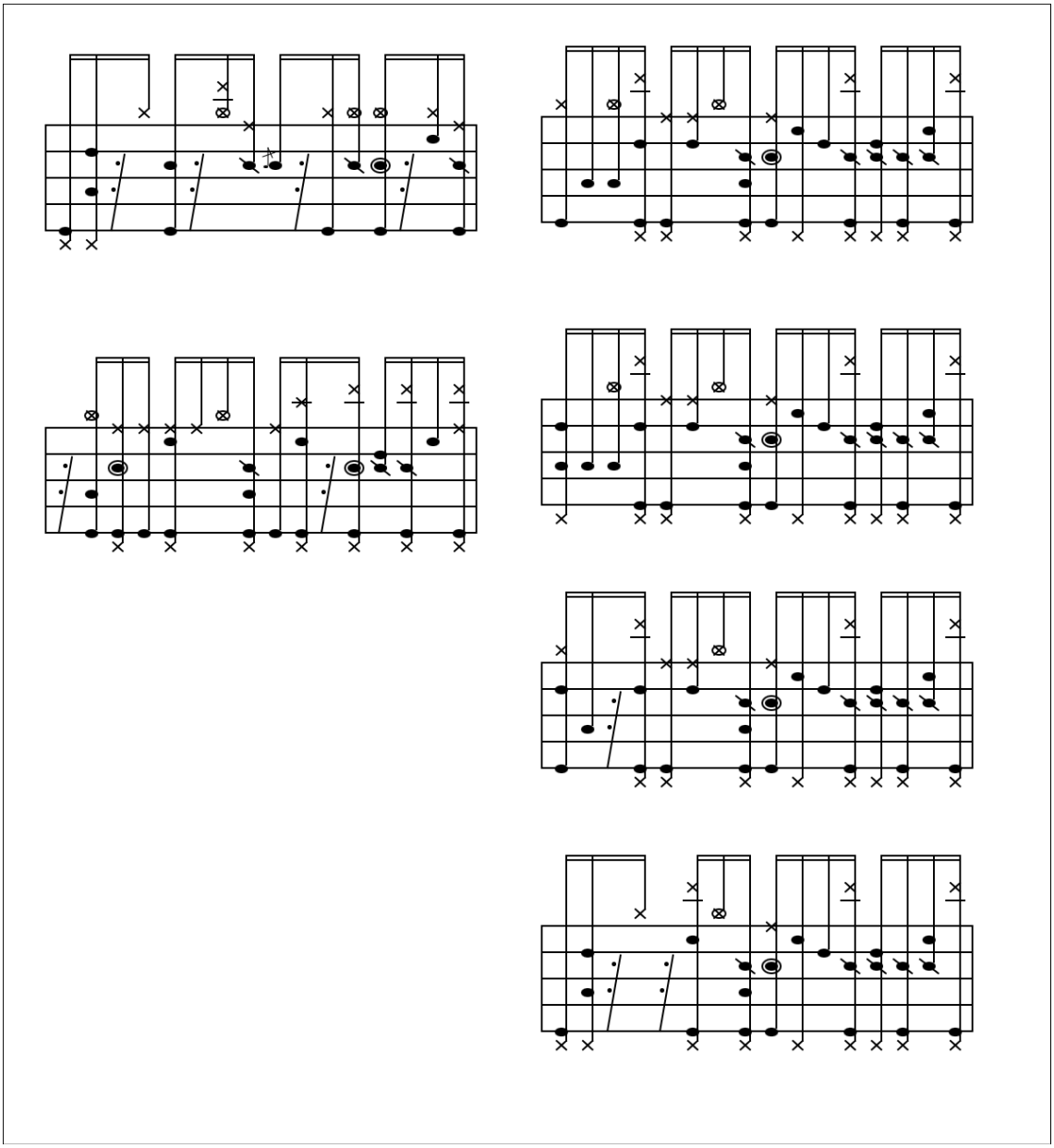

Abbildung C.1: Notenbilder der ausgesuchten Rhythmen in Selbstversuch 1.

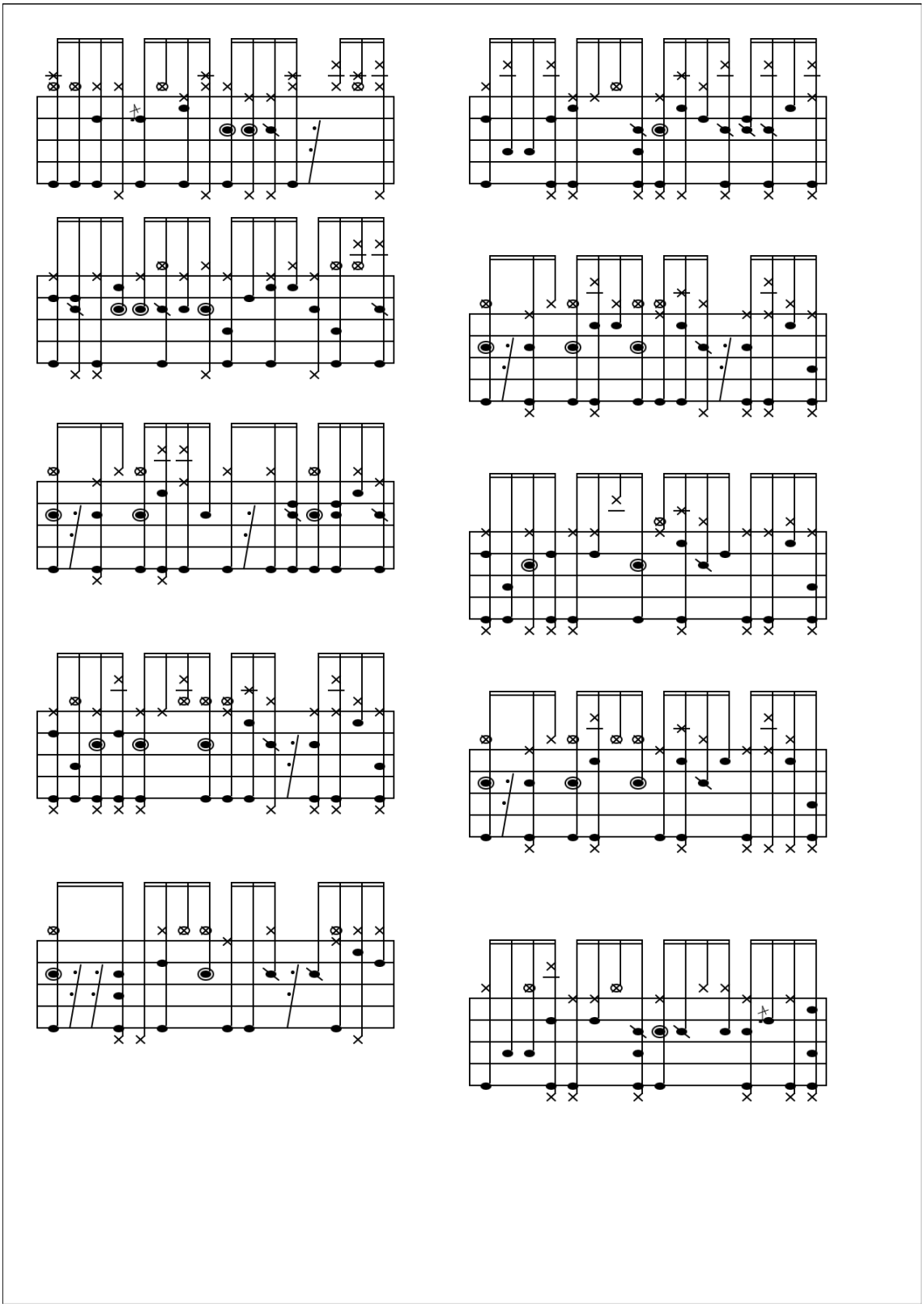

Abbildung C.2: Notenbilder der ausgesuchten Rhythmen in Selbstversuch 1.

D Rhythmen zu Selbsttest 2

Abbildung D.1 zeigt die Notenbilder, die zu den ausgewählten Rhythmen aus Selbstversuch 2 gehören.

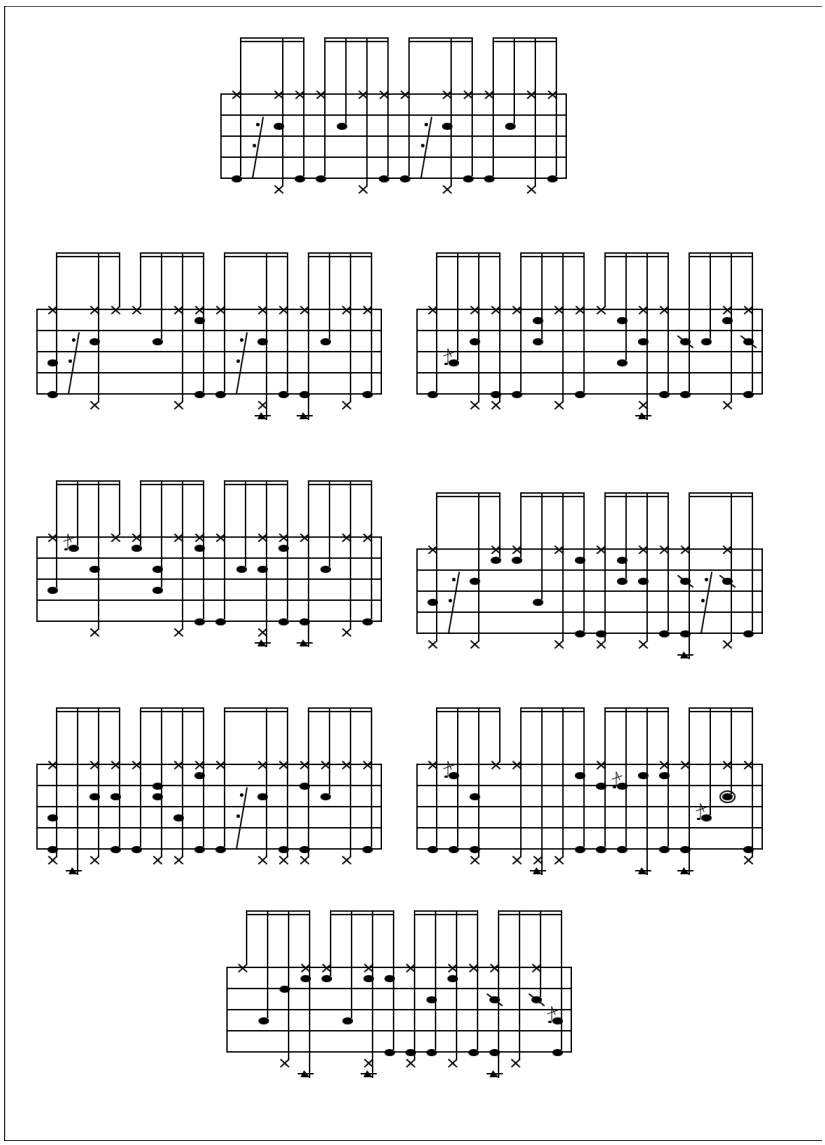

Abbildung D.1: Notenbilder der ausgesuchten Rhythmen in Selbstversuch 2.

E Aufschlüsselung der Instrumentkodierung

Abbildung E.1 zeigt die Kodierung der Standardinstrumente im Programm.

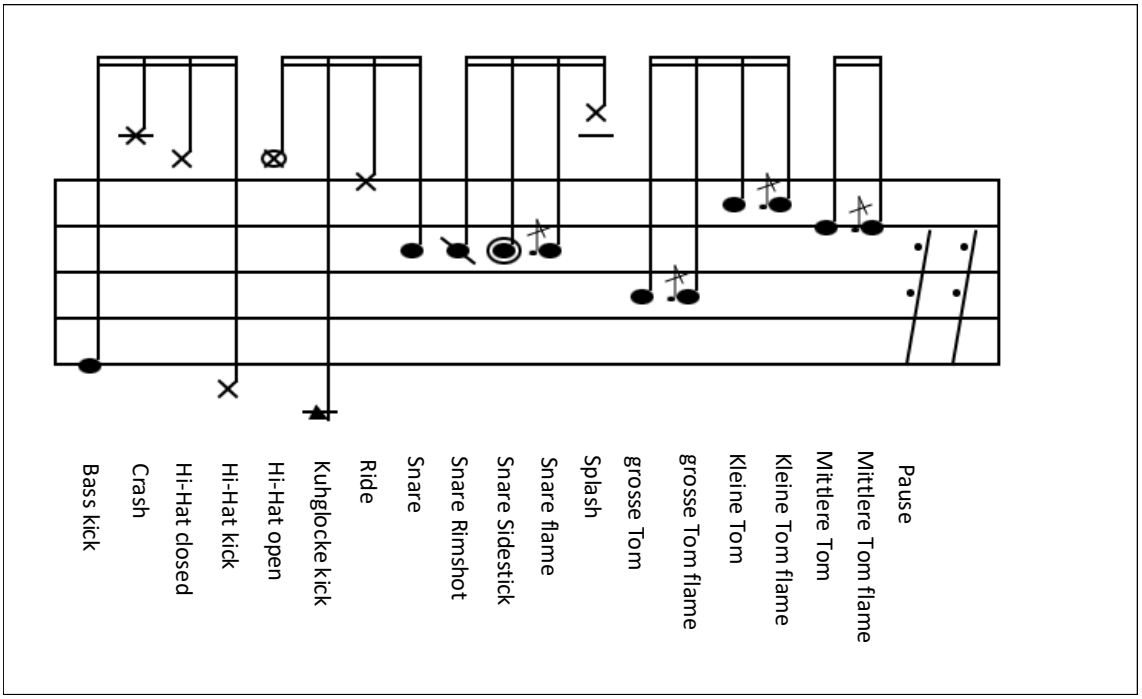

Abbildung E.1: Kodierung der Instrumente des Standardschlagzeugs im Programm.

F CD Inhalt

Beschreibung des Inhalts der beigelegten CD.

- JavaDoc-Verzeichnis: Dateien, der JavaDoc Dokumentation
- Programm-Verzeichnis: ausführbare Datei des Programms. (und Readme.txt)
- Quellcodes-Verzeichnis: alle Java-Projektversionen, die im Laufe der Entwicklung entstanden sind
- schriftliche Arbeit-Verzeichnis: PDF der Arbeit
- Uebungssounds-Verzeichnis: Sounddateien, als Nachweis, dass Rhythmen geübt wurden
- Workspaces-Verzeichnis: Workspaces, die für die Versuche genutzt wurden. Die Version des Programms, mit der sie geöffnet werden können liegt neben dem Workspace