



UNIVERSITÀ DEGLI STUDI DI TRIESTE

DIPARTIMENTO DI MATEMATICA E GEOSCIENZE

INTELLIGENZA ARTIFICIALE E DATA SCIENCE

Musica Genetica

Autore:

Leonardo Angellotti

Relatore:

Luca Manzoni

Anno Accademico 2023/2024

Contents

1	Introduzione	3
2	Nozioni di Base	5
2.1	nozioni algoritmiche	5
2.1.1	fitness-function	5
2.1.2	modello markoviano	5
2.1.3	artificial neural network	6
2.2	nozioni musicali	7
2.2.1	Caratteristiche di una Nota Musicale	7
2.3	scala armonica	7
2.4	melodia	9
2.5	accordi	9
2.6	File MIDI	10
3	Metodi di generazione	11
3.1	generazione melodia	11
3.1.1	armonia	13
3.1.2	ritmo	14
3.2	Algoritmo Genetico	14
3.2.1	musica evolutiva	14
3.2.2	Prime applicazioni dell'EC alla musica	15
3.3	L-System	16
4	progetto di musica genetica	19
4.0.1	Run-Evolution	20
4.0.2	fitness-function	22
4.0.3	risultati	25
5	Conclusioni	30

A	Appendice A	32
A.1	Dettagli aggiuntivi	32
B	Appendice B	33
B.1	Altro materiale supplementare	33

Chapter 1

Introduzione

Sebbene Stravinsky sia famoso per aver affermato: “i buoni compositori prendono in prestito e i grandi compositori rubano”,

C'è una grande differenza tra “impossibile” e “difficile da immaginare”. Il primo riguarda il problema; il secondo riguarda te.

La parola 'Algoritmo' deriva dai concetti di algebra e aritmetica dei matematici del IX secolo d.C., dalle culture arabe e greche a quelle indiane. La Musica Algoritmica ha una storia altrettanto lunga nell'era pre-informatica, dai lavori 'processuali' di George Brecht (Drip Music, 1962), Stockhausen (Setz die Segel zur Sonn, 1970), Xenakis (Metastaseis, 1971) fino a George Lewis (Voyager), ecc.

Gli algoritmi di Intelligenza Artificiale (AI) sono una delle tecnologie più ampiamente utilizzate e potenti del ventunesimo secolo. Sebbene la definizione esatta di AI si sia costantemente evoluta dagli anni '50 di Alan Turing alle attuali architetture di Deep Learning come Alpha Go, un aumento dell'accessibilità ha portato a un'assimilazione più rapida di una vasta gamma di algoritmi AI nella nostra cultura.

Tali modelli AI per attività creative sono attualmente agli inizi e seguono quasi esclusivamente il modello di imitare il dataset su cui sono addestrati.

Nel campo della creazione musicale, tali modelli di generazione automatica della musica sono mirati alla produzione di massa di contenuti musicali. Questo ha portato a molti esempi di sistemi di musica generativa che producono musica come fosse un problema di ottimizzazione ben definito. I corali di Bach (Deep Bach), la Modellazione della Musica Popolare (Folk RNN) ecc. sono alcuni esempi di modelli altamente vincolati che acquisiscono rappresentazioni dei pattern su cui sono addestrati.

Modellare la musica è un problema difficile per due ragioni principali:

(a) Le gerarchie semantiche sono complesse, soggettive e operano su più scale temporali (dell'ordine di secondi a diversi minuti);

(b) Le rappresentazioni audio a tassi di campionamento dell'udito umano rendono le sequenze da modellare davvero lunghe, ovvero una tipica canzone di 4 minuti in qualità CD (44 kHz, 16-bit) ha oltre 10 milioni di timesteps.

Negli ultimi cinque anni sono stati fatti molti progressi nella modellazione della musica per sistemi generativi nelle sue rappresentazioni simboliche e d'onda. Magenta di Google (2016 - 2020), MuseNet di OpenAI (2019) e Jukebox (2020), Flow Machines di Sony CSL (2012), il Universal Music Translator di Facebook AI Research (2019) e Deep Composer di Amazon (2019) sono alcuni degli sforzi delle grandi aziende tecnologiche che utilizzano risorse massicce come grandi server e ampi dataset.

Sebbene ci sia molto interesse tra accademici e sviluppatori di software nel creare il prossimo set di strumenti creativi utilizzando l'AI per la creazione musicale, ciò non si è riflesso nell'assimilazione delle idee AI da parte dei musicisti di oggi. Solo un ristretto numero di musicisti, limitato ai programmatori, sta utilizzando questi strumenti basati su codice nel loro processo creativo.

Un'altra limitazione è l'incapacità di guidare e modificare i modelli AI per personalizzare l'output all'interno delle strutture disponibili. I metodi attuali sono rigidi e consentono solo una singola mappatura dai parametri di input alla musica generata, con l'architettura fissata al momento dell'addestramento.

Un potenziale vantaggio degli algoritmi AI rispetto ad altri sistemi di musica computazionale che seguono un insieme di istruzioni è la capacità di interrogare regioni di rappresentazione con più o meno probabilità. Ciò consente paradigmi un costante flusso tra Exploitation e exploration.

Questo genere di modelli possono quindi essere utilizzati per esplorare scelte creative meno probabili da parte degli artisti.

"Se hai un computer che elabora combinazioni casuali di note musicali, un essere umano che ha sufficiente intuizione e tempo potrebbe benissimo cogliere un'idea o due. Un artista dotato, d'altra parte, potrebbe ascoltare la stessa composizione casuale e uscirne con un'idea completamente nuova, una in grado di innescare una nuova forma di melodia." - Margaret Boden (2016, intervista con IBM)

Struttura della tesi.

Nella tesi vengono riportate inizialmente le nozioni di base utili alla comprensione degli argomenti trattati. Vengono poi discussi i principali metodi di generazione musicale correnti, e gli studi associati. Infine viene presentato un algoritmo genetico, per la generazione di una sequenza musicale, discutendone i risultati conseguiti. Nelle conclusioni vengono raccolti i traguardi raggiunti, e le limitazioni da superare, nella costruzione algoritmica di nuovi modelli.

Chapter 2

Nozioni di Base

2.1 nozioni algoritmiche

2.1.1 fitness-function

Misura di Prestazione: La fitness function fornisce un punteggio numerico che rappresenta l'efficacia di una soluzione candidata.

Questo punteggio guida l'algoritmo nel processo di selezione e ottimizzazione. L'obiettivo è massimizzare o minimizzare la fitness function, a seconda del problema. Ad esempio, in un problema di massimizzazione, una soluzione con un punteggio di fitness più alto è preferita, mentre in un problema di minimizzazione, si preferiscono punteggi più bassi.

Negli algoritmi genetici, le soluzioni con punteggi di fitness più alti hanno una probabilità maggiore di essere selezionate per la riproduzione e la generazione di nuove soluzioni (prole).

2.1.2 modello markoviano

In un modello markoviano un nuovo stato dipende esclusivamente dallo stato corrente, senza tener conto degli stati passati.

Formalmente, questa proprietà si esprime come:

$$P(X_{n+1} = x | X_n = x_n, X_{n-1} = x_{n-1}, \dots, X_0 = x_0) = P(X_{n+1} = x | X_n = x_n)$$

dove X_i rappresenta lo stato del sistema al tempo i .

Il modello è costituito da un insieme finito o numerabile di stati. Le tran-

sizioni tra stati sono governate da probabilità di transizione che possono essere rappresentate in una matrice di transizione.

Per una catena di Markov con N stati, la matrice di transizione P è una matrice $N \times N$ dove l'elemento P_{ij} rappresenta la probabilità di transizione dallo stato i allo stato j .

Le righe della matrice devono sommare a 1:

$$\sum_{j=1}^N P_{ij} = 1$$

Un modello markoviano può raggiungere una distribuzione stazionaria, dove le probabilità di occupazione degli stati non cambiano più nel tempo. Questo accade quando il sistema ha raggiunto un equilibrio.

2.1.3 artificial neural network

Una Artificial Neural Network (ANN), o rete neurale artificiale, è un modello computazionale ispirato alla struttura e al funzionamento del cervello umano, progettato per riconoscere schemi complessi e prendere decisioni basate su input di dati.

Le ANNs sono particolarmente utili in vari campi dell'intelligenza artificiale, come il riconoscimento di immagini, il riconoscimento vocale, l'elaborazione del linguaggio naturale e molti altri.

Funzionamento di una ANN

Forward Propagation: Durante la propagazione in avanti, gli input vengono trasmessi attraverso i vari strati della rete, moltiplicando i valori degli input per i pesi delle connessioni e applicando la funzione di attivazione. Questo processo continua fino a raggiungere lo strato di output, producendo una previsione.

Backward Propagation (Backpropagation): Durante l'addestramento, l'errore tra la previsione e il valore reale viene calcolato e retropropagato attraverso la rete per aggiornare i pesi. L'algoritmo di backpropagation utilizza il calcolo del gradiente per minimizzare l'errore, adattando i pesi in modo iterativo.

2.2 nozioni musicali



Figure 2.1: generica linea melodica di uno spartito

2.2.1 Caratteristiche di una Nota Musicale

Altezza (Pitch):

L'altezza di una nota è la frequenza del suono che produce, determinando quanto sia acuto o grave.

Le note sono rappresentate su un pentagramma, dove la posizione verticale della nota indica la sua altezza.

Ad esempio, le note su righe o spazi più alti sul pentagramma hanno altezze maggiori.

Le note musicali fondamentali nell'ottava sono Do, Re, Mi, Fa, Sol, La, Si.

Durata:

La durata di una nota indica per quanto tempo viene suonata.

Di seguito un'immagine esemplificativa:

Intensità (Velocity):

L'intensità di una nota si riferisce a quanto forte o piano viene suonata.

Sono indicazioni come *p* (piano) per suonare piano, *f* (forte) per suonare forte, *mf* (mezzo forte), *mp* (mezzo piano), e altre variazioni come crescendo (aumentando l'intensità) e decrescendo (diminuendo l'intensità).

2.3 scala armonica

La scala maggiore è una delle scale diatoniche più comuni e ha un suono generalmente percepito come "felice" o "luminosa".

La scala maggiore segue uno schema specifico di toni (T) e semitoni (S). Lo schema è:

T - T - S - T - T - T - S

Ad esempio, la scala di Do maggiore (C maggiore) è:

Do (C) - Re (D) - Mi (E) - Fa (F) - Sol (G) - La (A) - Si (B) - Do (C)

Segno grafico	Nome	Durata
	intero (o semibreve)	4/4
	metà (o minima)	2/4
	quarto (o semiminima)	1/4
	ottavo (o croma)	1/8
	sedicesimo (o semicroma)	1/16
	trentaduesimo (o biscroma)	1/32
	sessantaquattresimo (o semibiscroma)	1/64

Figure 2.2: tabella riassuntiva della durata delle note

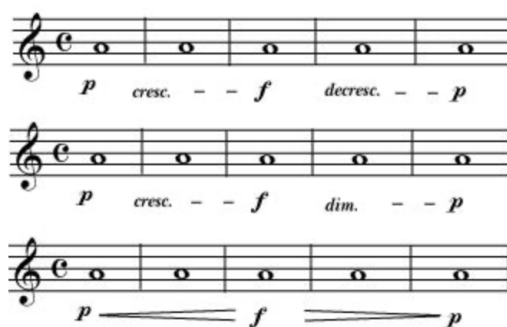


Figure 2.3: simbolismo dinamiche in uno spartito

La scala minore ha un suono generalmente percepito come "triste" o "scuro".

Segue uno schema diverso rispetto alla scala maggiore:

T - S - T - T - S - T - T

Ad esempio, la scala di La minore naturale (A minore naturale) è:

La (A) - Si (B) - Do (C) - Re (D) - Mi (E) - Fa (F) - Sol (G) - La (A)

2.4 melodia

Una melodia è una sequenza di note musicali che è percepita come un'unità singola e coerente.

Essa è una componente fondamentale della musica, essendo spesso la parte più riconoscibile e memorabile di un brano musicale.

La melodia può essere cantata, suonata su uno strumento o realizzata in una combinazione di voci e strumenti.

Una melodia è composta da una serie di note che vengono suonate in successione. Queste note possono variare in altezza (pitch), durata (rhythm), intensità (velocity).

Il contorno melodico si riferisce al profilo delle altezze delle note, ovvero come le altezze delle note salgono e scendono nel corso della melodia. Questo può includere salti, scali, ripetizioni e altre variazioni.

Una melodia è spesso divisa in frasi musicali, che sono come le frasi di un discorso. Ogni frase ha un inizio, uno sviluppo e una conclusione.

Un motivo è una piccola sequenza di note che ha significato tematico e viene spesso ripetuta e sviluppata nel corso di un brano.

La melodia è spesso basata su una scala musicale, come una scala maggiore o minore, che fornisce un quadro armonico e tonale.

2.5 accordi

Gli accordi sono combinazioni di tre o più note suonate simultaneamente o in sequenza arpeggiata, che creano un'armonia.

Gli accordi sono fondamentali nella musica, fornendo la struttura armonica su cui si basano melodie e progressioni.

Possono essere maggiori o minori, in base alla distanza tra le note.

Un problema fondamentale nella generazione di musica tramite algoritmi, è la scelta nella sequenza di una progressione di accordi, che descrive lo stile musicale di una canzone e la sua "orecchiabilità".

Esempi di Progressioni di Accordi usati comunemente in musica.

Progressione I-IV-V-I in Do Maggiore:

Accordi: Do (C) - Fa (F) - Sol (G) - Do (C)

Uso comune nelle canzoni pop e rock (il "giro di "Do)

Progressione ii-V-I in Jazz:

Accordi: Re minore settima (Dm7) - Sol settima (G7) - Do maggiore settima (Cmaj7)

Molto usata nel jazz per creare risoluzioni armoniche sofisticate.

2.6 File MIDI

Un file MIDI (Musical Instrument Digital Interface) è un formato di file standardizzato utilizzato per rappresentare informazioni musicali in forma digitale.

Invece di memorizzare il suono effettivo come fanno i file audio, un file MIDI memorizza i dati relativi alle note, agli strumenti, alle dinamiche e ad altre istruzioni musicali che possono essere riprodotte da dispositivi elettronici compatibili con il MIDI, come sintetizzatori, sequencer, computer e software di notazione musicale.

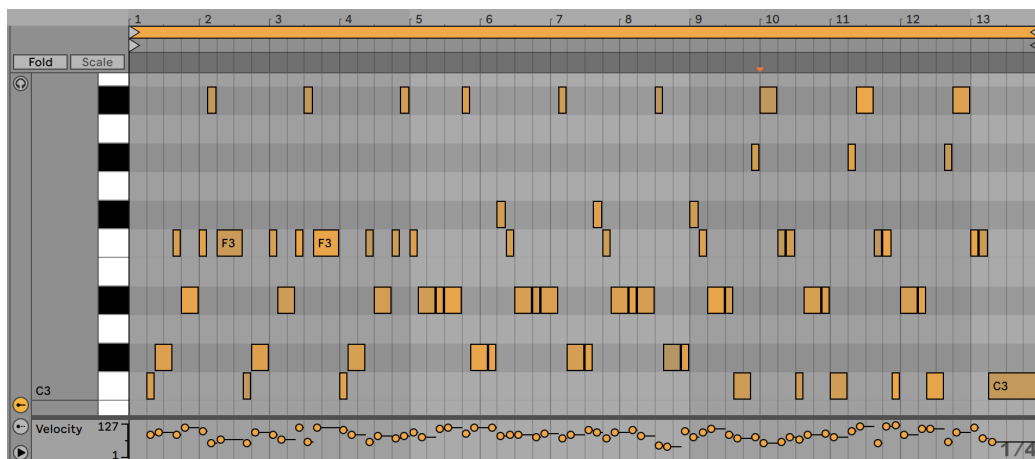


Figure 2.4: esempio di un file midi importato in Ableton

Chapter 3

Metodi di generazione

3.1 generazione melodia

Quando si considera il problema della generazione della musica, la forma più semplice di esercizio che viene in mente è la composizione di melodie monofoniche senza accompagnamento.

Nella maggior parte dei sistemi di generazione di melodie, l'obiettivo è comporre melodie con caratteristiche simili allo stile scelto.

Questi sistemi dipendono da una fitness-function che valuta le sequenze generate.

Tale funzione di fitness è spesso basata sulla somiglianza con un determinato corpus, stile o brano, o dipende in generale dalle regole teoriche musicali.

I primi tentativi di generare melodie con i computer risalgono al 1956, quando Pinkerton costruì un **modello markoviano**, il “Banal Tune-Maker”, basato su un corpus di 39 semplici filastrocche, tuttavia il modello tendeva ad essere ripetitivo.

Un altro problema nell'uso delle catene di Markov risiede nell'eccessiva somiglianza con la melodia originale, e dunque nel plagio.

Il compromesso tra la composizione di brani simili a lavori esistenti, contro la necessità di doverne creare di nuovi e creativi, è difficile da ricercare.

Si ricorda a tal proposito una citazione di Stravinsky, celebre compositore Russo:

“i buoni compositori copiano, i grandi compositori rubano”.

Riferendosi a quest'idea tuttavia le macchine non hanno ancora la capacità di distinguere tra il furto astuto e il plagio totale.

Con quest'intuizione è stata poi introdotta la variabile *MaxOrder*, ad indicare

il massimo ordine di sottosequenze consentito in una sequenza generata, con l'intento di limitare le ripetizioni eccessive di materiale.

Esistono diverse tipologie di tali vincoli di controllo, ad esempio la volontà che una sequenza sia globalmente ascendente o che segua invece un intorno di note arbitrario.

Davismoon ed Eccles (2010) sono stati alcuni dei primi ricercatori a inquadrare la generazione musicale come un problema di ottimizzazione combinatoria con un modello di Markov.

Per valutare la musica generata, il loro sistema costruisce un secondo modello di Markov, riducendo poi al minimo la distanza euclidea tra il modello originale e il nuovo modello.

Per risolvere questo problema di minimizzazione è stato utilizzato un processo di *simulated-annealing*.

Comporre una melodia monofonica può sembrare un compito semplice rispetto alla partitura di un'intera sinfonia.

Tuttavia, le melodie sono più che semplici movimenti tra le note, normalmente possiedono una **struttura a lungo termine**.

Negli ultimi anni, alcune ricerche hanno dimostrato l'efficacia dell'utilizzo di tecniche come il deep learning per rafforzare una consecuzione coerente.

Horner e Goldberg (1991), pionieri nell'applicazione degli algoritmi genetici (GA) nella composizione musicale, affrontano il problema del "**interpolazione tematica**", ovvero la trasformazione di un pattern musicale iniziale in uno finale, durante una durata specificata.

Per farlo hanno utilizzato un algoritmo genetico, un tipo di metaeuristica, diventato popolare negli anni '70.

generalmente si genera un insieme (chiamato popolazione) di soluzioni, e attraverso una combinazione (simile a quella genetica) se ne generano di nuove.

Sebbene esista molto lavoro sulla generazione di accordi data una melodia, alcuni studi si concentrano viceversa, sulla generazione di una melodia che si adatti a una sequenza di accordi.

Moorer (1972), ad esempio, genera prima una sequenza di accordi, e poi una linea melodica su questa.

Le note della melodia sono limitate solo a quelle dell'accordo corrispondente in un dato momento.

Ad ogni punto si decide, sulla base di un modello markoviano, di invertire i frammenti melodici basati sull'accordo, oppure di copiarne da quello prece-

dente.

Le brevi melodie risultanti tuttavia hanno un suono estraneo, ciò è legato al fatto che l'approccio melodico, non è quello utilizzato dagli esseri umani, infatti il sistema non discrimina le sequenze "non familiari".

3.1.1 armonia

L'armonia è definita da una relazione verticale tra note, che suonano contemporaneamente, e orizzontale, cioè la relazione delle note nel tempo, come le transizioni tra accordi.

Il sistema può essere valutato secondo criteri come, la somiglianza con un genere musicale, o con la musica di un particolare artista.

Il sistema deve generare sequenze riconoscibili nel genere target, ma non "sostanzialmente uguali".

La maggior parte degli studi adotta approcci basati sul training-set per determinare possibili accordi per un dato segmento melodico, e per garantire una corretta transizione tra questi.

Questi sistemi comunemente producono uno scheletro armonico, e utilizzano modelli predefiniti per creare trame ritmiche e arrangiamenti.

Lee e Jang (2004) hanno utilizzato il modello di Markov, integrato con programmazione dinamica, per determinare il pattern armonico di una melodia canticchiata dall'utente;

le probabilità di transizione dalle note correnti vengono apprese da 150 canzoni.

Simon (2008) ha adottato un approccio simile; allenando il proprio sistema su 298 brani di vari generi come jazz, rock, pop, blues e altri.

Entrambi i sistemi vengono valutati tramite feedback soggettivo, cioè da un umano e non da una qualche fitness-function, derivanti da sessioni di ascolto.

Uno svantaggio di questo approccio è che le sequenze di accordi generate tendono ad essere generiche e indistinte nello stile.

Steedman (1984) ha studiato le melodie blues con lunghezza di 12 battute, definendo un piccolo insieme di regole, è riuscito a produrre delle progressioni di accordi tipiche nello stile.

Come notato nell'articolo, non c'è stato alcun tentativo esplicito di generare buone progressioni di accordi o di evitarne di cattive.

3.1.2 ritmo

Nei sistemi di generazione musicale, il ritmo deriva spesso dalla durata intrinseca di ciascuna nota.

In generale, esistono meno sistemi di generazione ritmica rispetto a sistemi melodici.

Tokui e Iba (2000) hanno proposto il sistema CONGA, che usa algoritmi genetici per produrre modelli ritmici, usando l'utente umano come funzione fitness.

Brevi frammenti di schemi ritmici formano gli elementi cromosomici nell'algoritmo; subendo poi trasformazioni come il crossover e mutazioni.

Un algoritmo genetico è stato implementato anche da Ariza (2002), generando ritmiche derivanti da variazioni genetiche.

La funzione fitness è consistita nel calcolare la distanza tra il ritmo generato e l'originale, fornito dall'utente, attraverso differenti misure di distanza.

3.2 Algoritmo Genetico

3.2.1 musica evolutiva

L'applicazione dei metodi di *computazione evolutiva* ai compiti musicali ha cominciato ad emergere all'inizio degli anni '90.

Ci sono tre criteri fondamentali da considerare: l'ambito del problema, la rappresentazione individuale e la misura dell'idoneità.

L'ambito del problema definisce esattamente quale tipo di "musica" si desidera sviluppare; si è intenzionati a creare melodie, armonizzazioni o progressioni di accordi?

Nella ricerca è fondamentale la codifica del dominio, e quindi i limiti o lo stile musicale, entro cui l'algoritmo ricerca.

Nella rappresentazione individuale ci si chiede come rappresentare la musica: audio, spartiti stampati, messaggi MIDI (Musical Instrument Digital Interface)?

comunemente, per una più facile gestione dei dati, si ricorre al formato MIDI, dove il genoma contiene valori di pitch e durate.

Infine la questione della misura di idoneità: supponendo che due individui rappresentino brani musicali nel dominio desiderato, cosa rende uno "migliore" dell'altro?

Uno dei vantaggi dell'applicazione di algoritmi genetici è l'aggiornamento continuo di una popolazione di soluzioni, piuttosto che un individuo, in modo che le relazioni tra individui possano essere inserite nel processo di ricerca. Questo aspetto è utilizzato in numerosi studi, che espongono l'intera popolazione finale nella ricerca di un risultato.

3.2.2 Prime applicazioni dell'EC alla musica

Horner e Goldberg furono tra i primi ad applicare un algoritmo genetico al processo di composizione musicale considerando il problema della “transizione tematica”.

In questo si considerava la trasformazione di una frase melodica in un'altra, ed i genomi rappresentavano la serie di transizioni necessarie.

La fitness-function si misurava quanto bene il modello finale corrispondeva al modello desiderato.

Il sistema tuttavia è stato utilizzato per creare transizioni piuttosto che creare effettivamente nuova musica.

Indipendentemente dalla rappresentazione utilizzata, la valutazione della musica è spesso un compito soggettivo e, come tale, una misura di fitness affidabile e solida può essere difficile da definire.

Molti dei primi ricercatori aggirarono questo problema impiegando l'uso di un giudice umano, e tali sistemi sono noti come **Interactive EC** (IEC).

Una funzione di fitness basata sull'uomo era stata precedentemente utilizzata con successo nell'evoluzione delle immagini, ma gli autori hanno notato una differenza pratica:

più immagini possono essere visualizzate contemporaneamente, e molto rapidamente, mentre la musica è un fenomeno temporale, e quindi tutti gli individui devono essere ascoltati in sequenza, uno dopo l'altro.

Ciò si traduce in un costo molto elevato nell'utilizzo di un essere umano come funzione di fitness.

John Biles creò il sistema GenJam che utilizzava un algoritmo genetico per evolvere gli assoli jazz.

GenJam ha utilizzato due popolazioni indipendenti: una per le misure di fitness e una per la generazione di sequenze.

L'utente ha valutato queste frasi **in tempo reale** come “buone” o “cattive”. Il sistema ha quindi utilizzato l'intero aggregato di frasi e misure per costruire un assolo jazz.

In studi successivi, si è addestrata una ANN a fungere da funzione di fit-

ness, eliminando così la necessità dell'ascoltatore umano.

Tuttavia, in questo caso è riscontrato che i risultati erano in qualche modo carenti e si è stabilito che gli esseri umani ascoltano e sperimentano la musica in modi complessi e sottili, che non possono essere percepiti da modelli statistici come le ANN.

Negli studi che hanno fatto uso di ANN come fitness-function, gli autori hanno limitato il loro dominio, considerando solo armonie basate su tre accordi nella tonalità di Do maggiore, e creando solo melodie di quattro battute come durata.

La composizione è stata poi suddivisa in “mattoni” costituiti da ritmo, melodia e armonia.

Per ogni blocco è stata utilizzato un algoritmo genetico per generare la popolazione, ed una ANN è stata addestrata, da esempi forniti dall'utente, per giudicare i genomi candidati.

Gli autori hanno riconosciuto che questa combinazione di alg. gen. e ANN potrebbe produrre delle pseudo composizioni, ma le limitazioni applicate al dominio di addestramento da parte dell'utente hanno limitato fortemente il modello.

Durante gli anni '90 molti professionisti hanno applicato l'alg. gen. ad aspetti della generazione musicale, ma nessun metodo è risultato ideale rispetto ad altri.

Le misure di fitness basate su un corpus musicale o su un insieme di regole musicali possono soffrire di pregiudizi.

Gli esperimenti che utilizzano un corpus musicale per ricavare una misura di fitness saranno sempre limitati dal dominio scelto, limitando così il potenziale creativo del sistema.

Tali misure guidano l'evoluzione, ma indipendentemente da quanto coinvolta o sfaccettata sia una data misura di fitness, è sicuramente *impossibile dire che questa sia la misura definitiva della musica.*

3.3 L-System

Gli L-system (o Lindenmayer system) sono un tipo di sistema formale utilizzato principalmente per modellare la crescita di piante e altri organismi.

Sono stati introdotti da Aristid Lindenmayer nel 1968 e sono composti da un alfabeto di simboli che, usati come insieme di regole di produzione, vengono impiegati per generare stringhe, a loro volta contenenti altri simboli, appli-

cando dunque un approccio ricorsivo.

È necessario un assioma (o stringa iniziale) da cui iniziare, e meccanismi per interpretare le stringhe generate come strutture geometriche o altre forme di output.

Nella generazione musicale, gli L-system possono essere usati per creare strutture musicali complesse e organizzate.

Alfabeto: I simboli dell'alfabeto possono rappresentare note musicali, durate, dinamiche, articolazioni, e altri elementi musicali.

Assioma: La stringa iniziale può essere una sequenza di note o eventi musicali.

Le regole descrivono come trasformare ogni simbolo in una nuova sequenza di simboli musicali.

Alfabeto: A, B

Assioma: A

Regole di produzione:

$A \rightarrow AB$

$B \rightarrow A$

In questo esempio:

1. Partiamo dall'assioma "A".

2. Applichiamo le regole:

- A diventa AB.

- AB diventa ABA.

- ABA diventa ABAAB.

e così via.

Ora, mappiamo i simboli alle note musicali:

- A = Do

- B = Re

La sequenza generata può essere interpretata come una melodia: Do, Do-Re, Do-Re-Do, Do-Re-Do-Do-Re, e così via.

Gli L-system così permettono di generare pattern musicali ricorsivi e auto-simili che possono essere sia prevedibili che ma anche variabili.

Nel paper Growing music, Peter Worth e Susan Stepney, hanno sperimentato in modo più approfondito questo sistema.

Gli L-System permettono una maggiore sensibilità per il *contesto*, cioè il rapporto tra le note lungo la sequenza, facendo crescere conseguentemente la pianta in modo diverso in base a gli elementi nell'intorno.

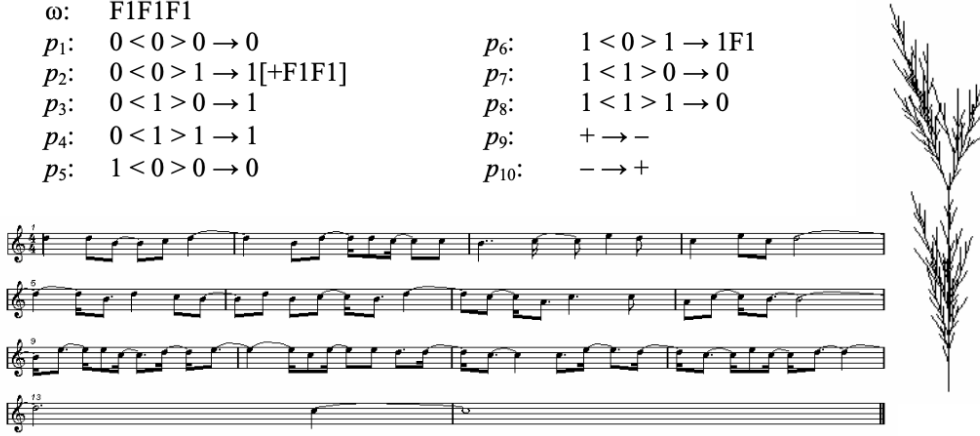
Questo potrebbe essere utile in musica, quando ad esempio il pezzo generato potrebbe raggiungere un climax o interrompersi bruscamente.

Per far ciò che questo avvenga, la regola di produzione per un L-System viene applicata al simbolo solo se compare in uno specifico contesto (tra altri simboli).

La notazione $A \mid B \mid C$ indica che agisce se la stringa B è presente con A alla sinistra e C a destra.

Viene considerato il seguente L-System *sensibile al contesto*:

$\omega:$ F1F1F1 $p_1:$ $0 < 0 > 0 \rightarrow 0$ $p_2:$ $0 < 0 > 1 \rightarrow 1[+F1F1]$ $p_3:$ $0 < 1 > 0 \rightarrow 1$ $p_4:$ $0 < 1 > 1 \rightarrow 1$ $p_5:$ $1 < 0 > 0 \rightarrow 0$	$p_6:$ $1 < 0 > 1 \rightarrow 1F1$ $p_7:$ $1 < 1 > 0 \rightarrow 0$ $p_8:$ $1 < 1 > 1 \rightarrow 0$ $p_9:$ $+ \rightarrow -$ $p_{10}:$ $- \rightarrow +$
---	---



The figure shows a musical score with four staves. The first three staves contain a melody in 4/4 time, with notes and rests. The fourth staff is mostly empty, with a few notes at the end. To the right of the staves is a vertical illustration of a plant with many small, branching leaves, resembling a stylized tree or a large bush.

Figure 3.1: Descrizione dell'immagine

Questa melodia, e altre derivate in modo simile, suonano abbastanza “casuali”; ricordano un assolo jazz.

Non rientrano bene nella cadenza 4/4 in una partitura perché molte delle note hanno una durata irregolare.

Tuttavia la melodia ritorna sempre a un motivo o una frase principale, a volte trasposto o suonato in un punto diverso della battuta.

Ad esempio, nella partitura sopra, una serie di note nella prima battuta è ripetuta nella nona battuta, ma in un modo insolito (trasposto in avanti per un quarto di tempo) e aumentate di 2 semitoni.

Questo tipo di simmetria rispecchia la musica normalmente composta o improvvisata, ed è dovuta dal tipo di struttura ricorsiva ad albero.

Chapter 4

progetto di musica genetica

Viene ora introdotto un progetto di algoritmo genetico utile ad esemplificare fin quanto detto.

All'interno del progetto la funzione:

```
1
2 # Main function for running all helper functions and
   handling user input.
3 def genetic_algorithm(key, root, tempo, rythm):
```

scrive su disco 10 genotipi di sequenze di note, in formato MIDI, come risultato al dell'esecuzione dell'algoritmo genetico.

La funzione prende in input dall'utente

key: la scala utilizzata nella scelta delle note, può essere major o minor.

root: la nota che indica la tonalità della canzone, e sulla quale verrà costruita la scala (C, C, D, D, E, F, F, G, G, A, A, B)

tempo: il numero di battiti al minuto (bpm), ovvero la velocità di esecuzione, della composizione

rythm: il ritmo della composizione, ovvero la durata delle note, a scelta tra quattro predefiniti (rock, jazz, dance, bossa nova)

all'interno della funzione troviamo

```
1
2 # Build the scale based on the root note and
   chosen scale type
3 scale = buildScale(root, key)
```

Restituisce un array, dove ogni numero rappresenta il codice MIDI di ogni nota.

L'array rappresenta la scala, maggiore o minore, costruita partendo dalla nota fondamentale scelta.

Viene eseguita poi l'evoluzione vera e propria, attraverso la crescita del genoma.

4.0.1 Run-Evolution

```
1
2      # Run the genetic algorithm to evolve a melody
3      using the specified mutation rate and scale
4      evolvedMelody = runEvolution(MUTATION_RATE,
5                                    scale)
```

runEvolution implementa un algoritmo genetico che migliora in modo iterativo la popolazione di genomi musicali attraverso selezione, crossover e mutazione, l'esecuzione avviene finché non viene raggiunto un genoma con il punteggio MAX-FITNESS specificato.

Viene inizialmente creata una popolazione, inizializzata con metodo random. la popolazione è composta da 10 genomi.

Ogni genoma è un elenco 2D di sequenze di note organizzate in 8 bar, ciascuna contenente 16 note, tutte scelte casualmente dalla scala specificata.

Il codice che segue rappresenta il cuore di un algoritmo genetico utilizzato in quest'occasione, e in generale.

```
1
2      # Iterate for a maximum number of generations
3      for _ in range(MAX_GENERATIONS):
4
5          # Sort the population based on fitness
6          scores in descending order
7          population = sorted(population, key=
8                               fitnessFunction, reverse=True)
9
10         # Select the top 2 genomes (the fittest) to
11         carry over to the next generation
12         nextGeneration = population[:2]
13
14         # Generate the rest of the next generation
15         through crossover and mutation
16         for _ in range(len(population) // 2 - 1):
```

```

14         # Select two parent genomes based on
           their fitness
15     parentA, parentB = selectParents(
           population)
16     # Perform crossover to produce two child
           genomes
17     childA, childB = multipointCrossover(
           parentA, parentB)
18     # Mutate the child genomes and add them
           to the next generation
19     nextGeneration += [mutateGenome(childA,
           mutationRate, scale), mutateGenome(
           childB, mutationRate, scale)]
20
21     # Update the population with the new
           generation
22     population = nextGeneration

```

Per un numero massimo di generazioni prefissato, vengono create popolazioni di genomi.

Successivamente alla prima inizializzazione randomica, viene applicata la Fitness-Function, che assegna ad ogni genoma un peso/punteggio basandosi sui propri criteri di valutazione (vedremo in seguito).

Nella successiva popolazione vengono mantenuti i primi due genomi con il punteggio più alto, mentre per i restanti 8 vengono selezionati di volta in volta due genitori, da cui attraverso un (multipoint) crossover danno luogo a due figli, i quali passano attraverso un'ultima fase di mutazione.

I processi di crossover e mutazione sono definiti di "exploration", perché servono ad esplorare soluzioni che i due migliori genomi non sono riusciti ad ottenere.

Viceversa i due genomi mantenuti con il punteggio più alto, che si tramandano da una popolazione alla successiva, servono a garantire la migliore performance ad ogni generazione, qualora i genomi di exploration non diano risultati soddisfacenti.

Quando avviene che in una popolazione, i figli generati ottengano un punteggio superiore ai due migliori correnti, questi vengono dunque sostituiti.

Al termine del numero massimo di generazioni (prefissato, MAX-GENERATIONS), viene scritta su disco l'ultima popolazione costituita da 10 genomi.

Il migliore tra questi (con punteggio più alto) viene chiamato *best_{genome}.mid*.

4.0.2 fitness-function

La fitness function calcola il punteggio finale ponderando e sommando i punteggi dei componenti per fluidità, ritmo e armonia, restituendo il risultato. Vengono inizializzati i pesi e i punteggi.

```
1
2  # Weights for the different fitness components
3      smoothnessWeight = 15
4      restWeight = 5
5      harmonyWeight = 20
6
7  # Initialize fitness scores
8      smoothnessScore = 0
9      restScore = 0
10     harmonyScore = 0
```

L'harmonyscore riporta il punteggio di armonizzazione tra le note, ovvero come queste suonino "bene" tra loro.

Di seguito un dizionario in cui troviamo come chiave la distanza tra la nota corrente e la precedente, e come valore il punteggio assegnato a tale distanza armonica.

```
1
2  # Harmony intervals table for scoring harmony
3  harmonyIntervalsTable = {
4      0: -20, 1: 5, 2: 5, 3: 50, 4: 50, 5: 30, 6:
5      -10, 7: 50, 8: 10, 9: 40, 10: -2, 11: -2,
6      12: 10,
7      13: -5, 14: 5, 15: 5, 16: 50, 17: 50, 18:
8      30, 19: -10, 20: 50, 21: 10, 22: 40, 23:
9      -2, 24: -2, 25: 10
10 }
```

- Intervalli consonanti perfetti (0, 7, 12, 19, 24 semitoni): Questi intervalli, che corrispondono alle ottave e quinte perfette (e le loro ripetizioni all'ottava superiore), hanno punteggi molto alti (50 o 10), indicando che sono molto armoniosi.

- 0 (Unisono): -20 (interessante che l'unisono sia visto come negativo in questo sistema).

- 7 (Quinta giusta): 50

- 12 (Ottava giusta): 10

- 19 (Unisono all'ottava superiore): -10

- 24 (Due ottave giuste): 10

- Intervalli consonanti maggiori (4, 9, 16, 21 semitoni): Questi includono la terza maggiore e la sesta maggiore, che sono generalmente considerati molto armoniosi.

- Intervalli consonanti minori (3, 8, 17, 22 semitoni): Questi includono la terza minore e la sesta minore.

- Intervalli dissonanti (1, 2, 6, 10, 11, 13, 23 semitoni): Questi includono la seconda minore, la settima minore e la settima maggiore, che sono considerati meno armoniosi o dissonanti.

- 1 (Seconda minore): 5

- 2 (Seconda maggiore): 5

- 6 (Tritono): -10

- 10 (Settima minore): -2

- 11 (Settima maggiore): -2

- 13 (Ottava aumentata): -5

- 23 (Nona aumentata): -2

Calcolo del punteggio di fluidità (smoothnessScore):

Se la differenza tra le note (noteDifference) è 0 (unisono), il punteggio di fluidità viene diviso per 10, penalizzando fortemente la ripetizione della stessa nota.

Se la differenza è minore o uguale a 2 semitoni, aumenta il punteggio di fluidità di 1, incentivando piccoli movimenti tra le note.

Se la differenza è di 11 semitoni (quasi un'ottava), il punteggio di fluidità viene diviso per 2, penalizzando questo ampio intervallo.

Per altri intervalli, il punteggio di fluidità aumenta di $1 / \text{noteDifference}$, favorendo movimenti più piccoli tra le note.

Se noteDifference è 0, si evita la divisione per zero assegnando 0.

Il codice cerca dunque di bilanciare l'armonia (basata sugli intervalli predefiniti) e la fluidità (basata sulla distanza tra note consecutive) in una sequenza musicale, con vari meccanismi per favorire o penalizzare certi tipi di movimenti tra le note.

```
1
2 if note is None and prevNote is None:
3     consecutiveRests += 1
```

Se entrambe note e prevNote sono None, significa che si è verificata una pausa consecutiva: Incrementa il contatore consecutiveRests.


```

1
2 if numRests * 10 <= len(flatten(genome)):
3     restScore += 10

```

Se il numero di pause (numRests) moltiplicato per 10 è inferiore o uguale alla lunghezza della sequenza musicale piatta (flatten(genome)), aumenta il punteggio ritmico (restScore) di 10.

Questo premia una quantità ragionevole di pause nel contesto della lunghezza totale del genoma musicale, suggerendo che alcune pause sono considerate musicalmente desiderabili.

```

1
2 if consecutiveRests:
3     restScore -= (consecutiveRests * 10)

```

Se ci sono pause consecutive (consecutiveRests > 0), penalizza il punteggio ritmico diminuendo restScore di consecutiveRests * 10.

Questo penalizza pesantemente le pause consecutive, suggerendo che una lunga serie di pause è considerata indesiderabile dal punto di vista ritmico.

```

1
2 fitness_weight_genome = (smoothnessScore *
    smoothnessWeight) + (restScore * restWeight) + (
    harmonyScore * harmonyWeight)

```

Il punteggio di fitness finale (fitness-weight-genome) è calcolato come la somma ponderata dei punteggi componenti.

Ogni punteggio componente (smoothnessScore, restScore, harmonyScore) viene moltiplicato per il rispettivo peso (smoothnessWeight, restWeight, harmonyWeight).

La somma dei prodotti risultanti rappresenta il punteggio di fitness finale del genoma.

I pesi (smoothnessWeight, restWeight, harmonyWeight) permettono di enfatizzare o de-enfatizzare specifici aspetti della valutazione a seconda degli obiettivi desiderati per la composizione musicale.

Ad esempio, aumentando il peso della fluidità, si darà maggiore importanza alle transizioni tra le note, mentre aumentando il peso dell'armonia, si enfatizzerà la qualità armonica della melodia.

Questo metodo di valutazione è utile in contesti come algoritmi genetici per la generazione di musica, dove è necessario valutare e selezionare sequenze musicali basate su criteri compositi.

4.0.3 risultati

Eseguiamo dunque il programma.

Sono riportati le opzioni scelte per una generazione musicale esemplificativa.

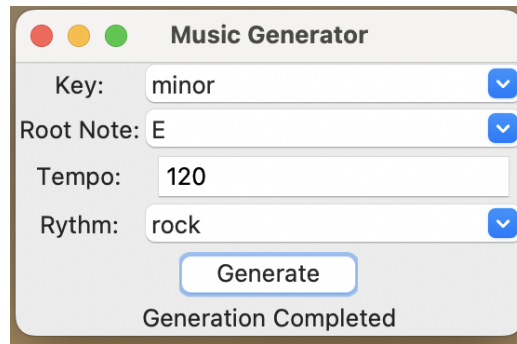


Figure 4.1: Descrizione dell'immagine

Al termine dell'esecuzione vengono stampati i risultati dei punteggi ottenuti dalla fitness function, per i genomi di ogni popolazione.

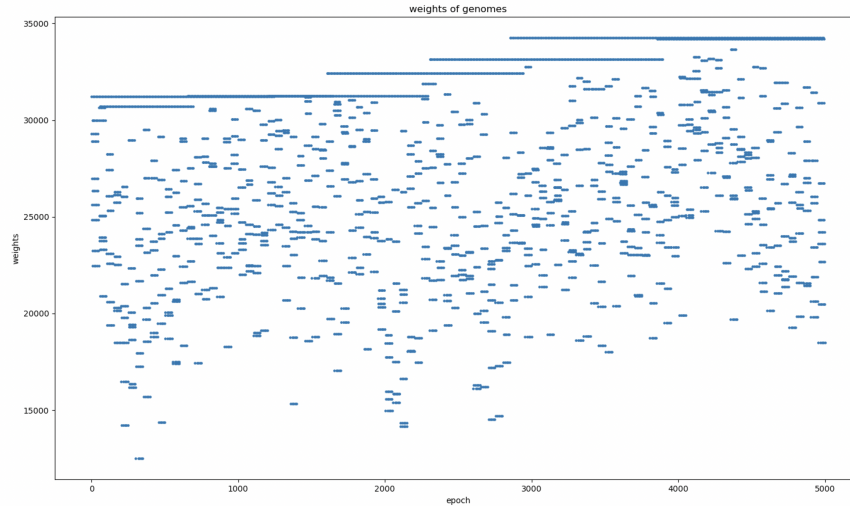


Figure 4.2: Descrizione dell'immagine

Le linee che si mantengono costanti sono i punteggi dei primi due genomi (con punteggio più alto ad ogni generazione) che nell'algoritmo scelto vengono tramandati senza alcuna alterazione (crossover, mutation).

Quando "nasce" un nuovo genoma che si verifica avere un punteggio più alto

dei primi due, questo fissa un nuovo livello di punteggio superiore ai precedenti, che viene mantenuto finché ciò non accade nuovamente.

Questa procedura garantisce un costante aumento del punteggio e un conseguente miglioramento tra le popolazioni.

La nuvola di punti che appaiono al di sotto, con punteggio inferiore, appartengono al set di genomi soggetti a crossover e mutazioni.

Al termine della procedura evolutiva, vengono stampati i 10 genomi appartenenti all'ultima popolazione.

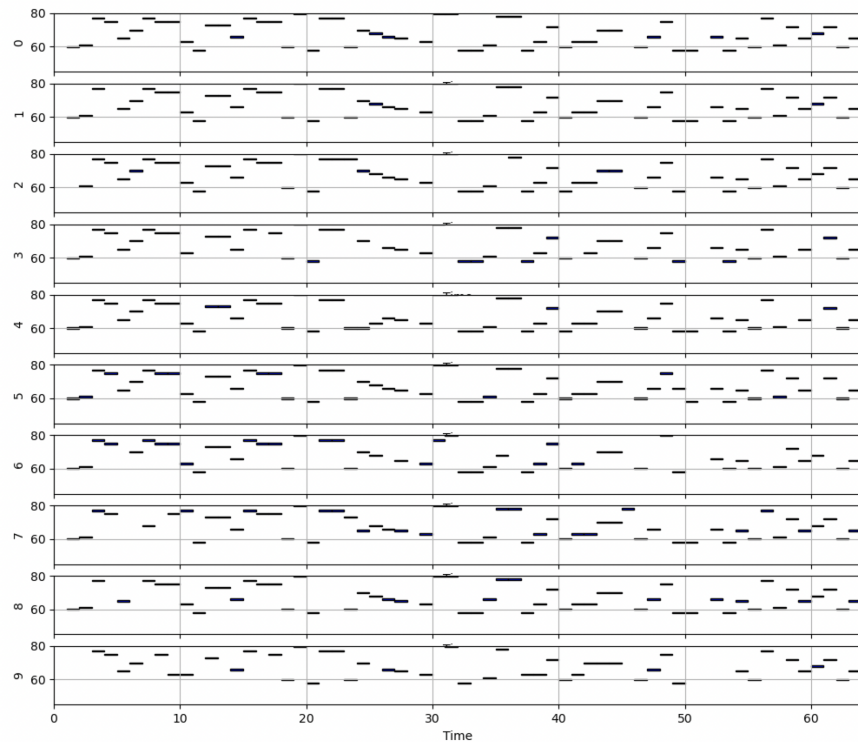


Figure 4.3: Descrizione dell'immagine

La stampa rappresenta 10 file midi, dal migliore (numero 0) al "peggiore" (numero 9) secondo la nostra fitness-function.

Le note sono rappresentate dalle linee orizzontali che appaiono a diverse altezze di pitch, indicati nell'asse delle ordinate.

Le note sono sequenziali, appaiono dunque in momenti consecutivi, con diversa durata (dettata dal ritmo scelto dall'utente).

Si ricorda che l'algoritmo genera una sequenza monofonica, infatti non appare più di una nota contemporaneamente.

Le sequenze vengono salvate su disco.

Il genoma migliore prende il nome di `best-genome.mid`, mentre ai restanti viene assegnato un numero in ordine decrescente in punteggio (`genome-0.mid`, ..., `genome-8.mid`).

Ad un primo ascolto di `best-genome.mid`, percepiamo un movimento coerente nella teoria, ordinato nel ritmo, ma non regolare nel tempo.

Non viene percepita infatti alcuna vera struttura compositiva, questo perché la fitness function si è preoccupata di fare confronti solo tra una nota e la precedente, e non tra tutte le note della sequenza.

Non esiste dunque una vera struttura a lungo termine, una struttura narrativa, avente inizio svolgimento e fine, propria invece di qualsiasi genere musicale.

Non è presente nemmeno il classico schema delle canzoni pop: verso-ritornello-verso-ritornello-bridge-ritornello.

Per ovviare a questo difetto si procede con un'operazione naive ma efficace:

Si decide di generare una nuova sequenza, chiamata `repeated-melody.mid`, la quale ripete per 4 volte le prime 3 bar prese dal `best-genome.mid`.

Il risultato risulta essere "ripetitivo" e dunque intrinsecamente più regolare, oltre a essere per il nostro orecchio più confortevole.

Abbiamo dunque ottenuto un risultato teorico *accettabile e coerente*.

Possiamo ottenere di più nella pratica?

Sì.

Importando il file midi (`repeated-melody.mid`) in una DAW (Digital Audio Workstation, in questo caso è stato utilizzato Ableton) possiamo apportare ulteriori modifiche.

Viene assegnata alla traccia lo strumento Grand Piano.

Su questa vengono applicate le seguenti opzioni

"MinorTim" permette di generare più note, ad una certa distanza in semitoni, partendo dalla nota corrente.

Nel caso specifico un accordo minore.

Tuttavia un primo ascolto risulta essere poco piacevole, perché le note generate non considerano la scala di riferimento.

È necessario dunque usare lo strumento "Minor" per ancorare ogni nota in una posizione coerente.

Il risultato è decisamente più armonico, seppur tuttavia ancor troppo "pi-

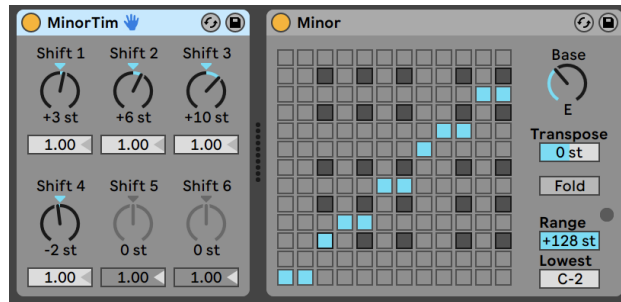


Figure 4.4: Descrizione dell'immagine

atto” con nessuna variazione nell’esecuzione.

Per far fronte a questa ”freddezza” ed aumentare la dinamicità è possibile randomizzare, in modo automatico, i valori di Velocity nella sequenza per ciascuna nota.

Un nuovo ascolto ci ricorda un qualche compositore jazz, immerso nel suo fiume di sentimenti, vissuti istante per istante, ma che vuole comunque mantenere, per piacere della clientela, un ritmo e melodia regolari.

Per rendere il risultato ancor più gradevole, e musicalmente completo, possiamo aggiungere un elemento di percussioni, come una batteria.

Per il risultato ottenuto è stato utilizzato il plug-in Magenta sviluppato da Google e attualmente disponibile gratuitamente.

In particolare è stata utilizzata la funzione ”Drumify”, che data una sequenza di note, genera un secondo file MIDI in cui, similmente per il piano, vengono indicate le parti della batteria da percuotere (piatto, charleston, gran cassa, rullante).

In particolare, nel risultato finale, quest’ultima sequenza MIDI è stata processata dal Plugin SuperiorDrummer3, una libreria sofisticata che raccoglie suoni di batteria accurati.

Si vuole far notare, che l’algoritmo implementato in Drumify, si basa su una neural network, non è di tipo genetico come nel caso del nostro esempio, in grado dunque di cogliere e sviluppare una certa consequenzialità tra parti.

Infine, per le basse frequenze è stata aggiunta una semplice linea di basso, trasponendo la melodia due ottave inferiori, eliminando a mano, secondo il proprio gusto, le note ridondanti, lasciando invece quelle fondamentali e di maggiore durata.

Il risultato finale, nello specifico, è *quasi accettabile*, considerando che per generarlo sono state necessarie poco più di 500 linee di codice (commenti inclusi) e qualche plugin dal mondo di internet.

Il tratto fondamentale risiede nel fatto che per ottenere questo file mp3, non abbiamo mai: né dovuto suonare un pianoforte, pizzicare un basso o ritmare una batteria a tempo.

Si possono sperimentare ulteriori modifiche, cambiando i pesi (smoothnessWeight, restWeight, harmonyWeight) o implementando un sistema nell'assegnazione della durata e velocity nelle note, ora affidate ad array con valori prefissati (rock, jazz, dance, bossa nova) e ad una scelta random rispettivamente.

Chapter 5

Conclusioni

Negli ultimi decenni, la ricerca sulla generazione musicale ha compiuto enormi progressi nella generazione di aspetti ben definiti della musica, come la melodia, l'armonia, il ritmo e il timbro.

Modelli statistici all'avanguardia, tecniche di ottimizzazione avanzate, database digitali più grandi su cui addestrare i modelli e l'aumento della potenza di calcolo hanno portato alla produzione di sistemi migliori.

Perché allora non utilizziamo sistemi di generazione musicale nella nostra vita quotidiana?

L'indagine di cui sopra mostra che rimane un'importante sfida generale: quella di creare musica con una struttura a lungo termine.

La struttura a lungo termine, che spesso assume la forma di temi, motivi e schemi ricorrenti, è una parte essenziale di qualsiasi esperienza di ascolto musicale.

Per rendere i sistemi musicali generati dal computer parte della nostra vita quotidiana, c'è bisogno di sistemi più "narrativi", in grado cioè di creare un inizio, sviluppo e conclusione nella musica.

Sebbene ci siano recenti tentativi di generare musica con tensione [Farbood et al. 2007; Herremans e Chew 2016a], che sia applicata a colonne per videogiochi o che accompagnano film, c'è ancora margine di miglioramento per comprendere meglio la connessione tra musica ed emozione.

Sebbene le tecniche di apprendimento automatico possano essere estremamente efficaci, di solito richiedono grandi quantità di dati.

Esiste un potenziale reale affinché il lavoro futuro si sposti verso sistemi intelligenti che non richiedano abbondanti quantità di tracce audio, o sequenze musicali, capaci cioè di un ragionamento innato, rispecchiando meglio il funzionamento della mente umana.

Ciò risolverebbe anche la continua sfida di trovare un equilibrio tra la rigenerazione della musica esistente, o frammenti di questa senza ricadere nel plagio.

Appendix A

Appendice A

A.1 Dettagli aggiuntivi

Appendix B

Appendice B

B.1 Altro materiale supplementare

Come descritto da Lamport [2], e come discusso da Doe e Smith [1], i metodi utilizzati sono basati su...

Bibliography

- [1] John Doe and Jane Smith. An example article. *Journal of Examples*, 42(1):1–10, 2023.
- [2] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, 1994.