

# IMprovEEsation: Intelligent Musical Evolutionary Entertainment

*Davide Berardi, Matteo Martelli, Marco Melletti, Federico Montori*

## Sommario

### 1 Introduzione

### 2 Stato dell'Arte

Questa sezione tratta del lavoro che è già stato compiuto nell'ambito della composizione e dell'improvvisazione della macchina. È da precisare che questo progetto non si è inizialmente fondato su un'idea esistente, poichè siamo voluti partire da un'idea che è scaturita da noi e l'abbiamo voluta sviluppare. Le pubblicazioni presenti in questa sezione sono da considerarsi spunti.

Una delle applicazioni più utilizzate da ciò che è chiamato "Machine Composition" è la possibilità di creare una vera e propria band di backup con la quale un improvvisatore può suonare. Perciò quello che interessa la maggior parte di questi software è la creazione di una melodia standard, non articolata, che segua una mappa di accordi fornita dall'utente e un certo pool di stili preimpostati (che possono essere molto numerosi), con l'obiettivo di dare all'utente la facoltà di suonare il proprio strumento con una band che esegue il background. L'improvvisazione è sempre il risultato finale, ma non è esattamente quello che vogliamo ottenere. Fra questi software citiamo Band-In-A-Box[3], che crea basi di accompagnamento in diverse divisioni, in tutte le possibili tonalità e accordi. Recentemente è in grado di esibirsi in assoli di chitarra e sassofono, che però si riferiscono sempre a standard compresi nel database dei pattern del programma. Questo software ottiene ottimi risultati nell'accompagnamento, ma non concede piena libertà ai suoi musicisti digitali, ovvero non considera che "tutto potrebbe accadere" come in un'improvvisazione reale.

Un altro aspetto molto affine con il nostro lavoro è una branca dell'IA chiamata Evolutionary Music, che è divenuta una vera e propria materia di studio a sè stante[4]. Sono stati effettuati molti tentativi di integrare l'improvvisazione e la composizione con gli algoritmi genetici, due esempi possono essere GenDash[5], che aiuta il musicista a comporre musica, e GenJam[6], il quale output è una jam session, che utilizza i risultati di Band-In-A-Box per poi mutarli a seconda del feeling che il brano sta assumendo, ad esempio, se il solista tende ad accelerare o a inserire un numero di note maggiore, anche la batteria e il basso addenseranno più colpi rendendo il ritmo più incalzante pur essendo la base la stessa.

L'Evolutionary Music si incrocia con la Computer Music in un ambito detto Machine Improvisation, che utilizza diversi sistemi per simulare l'improvvisazione di un solista su una base musicale parzialmente nota. Esistono i metodi statistici, basati su Catene di markov, HMM e processi stocastici[7]. Questi hanno dato origine a progetti come il Continuator[9], il quale fa uso di modellazioni di stili non-real time[8].

È da precisare che un programma in grado di produrre un brano completamente improvvisato non è mai stato realizzato, ma si è sempre cercato di partire da una base musicale definita (facendo improvvisare un solo solista) oppure musicisti virtuali che imparano da un musicista fisico in real time per poterlo accompagnare. L'intento del presente progetto è invece quello di creare ciò che pù si avvicina a una Jam session, ovvero strumentisti che si ritrovano in una sala prove e seguono una serie di direttive generiche per produrre un brano senza una base esistente.

### 3 Modello del Dominio

Questa sezione descriverà la composizione del nostro progetto alla luce di ciò che è già stato fatto e ciò che vogliamo introdurre. Il progetto ha come priorità l'esecuzione di un brano improvvisato da parte di più musicisti, i quali non hanno (almeno per il momento) alcuna coscienza della presenza di altri musicisti. Come mostrato in figura 1, la struttura del sistema consta di alcuni componenti differenti, che descriveremo nelle sezioni seguenti, organizzati esattamente come in un'orchestra reale (fatta eccezione per il player): il direttore d'orchestra trasferisce le proprie informazioni generali riguardo all'esecuzione a un numero arbitrario di musicisti, i quali eseguono una parte definita del brano e la passano a un player, che si occupa di tradurre in contemporanea ogni parte "scritta" da un musicista appunto in musica. È quasi come se i musicisti in questo caso, invece di produrre il suono loro stessi, producessero lo spartito misura per misura.

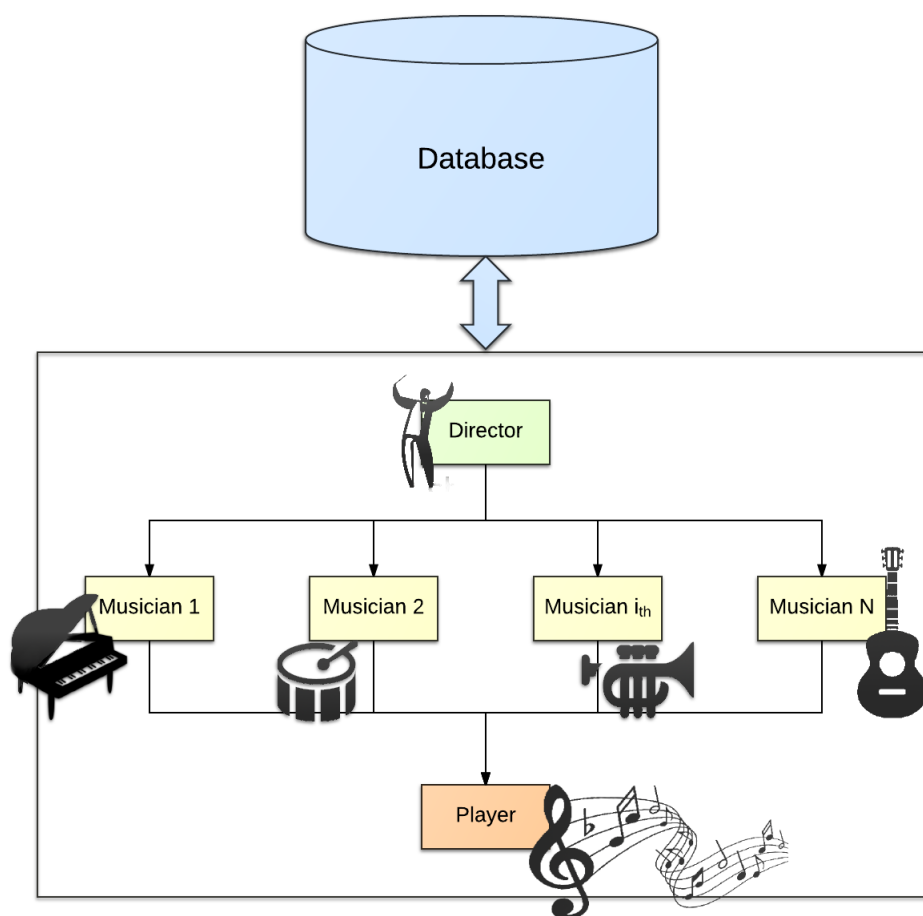


Fig. 1: Schema dello scenario di progetto

Nel nostro caso, sia il direttore sia i musicisti sono da considerarsi agenti intelligenti, mentre il player è un mero esecutore (come vedremo nelle descrizioni dettagliate nei prossimi capitoli).

Tali agenti intelligenti operano in un ambiente che potremmo descrivere con la notazione PEAS:

- Parzialmente osservabile, poichè i musicisti conoscono solo lo stato corrente del direttore, ovvero soltanto ciò che esso decide di far suonare più, ovviamente, il proprio stato, ma non conosce ciò che gli altri musicisti eseguono. Inoltre, il direttore non conosce lo stato interno dei musicisti, ma si limita a decidere i parametri della misura seguente mediante un proprio algoritmo interno. Questo aspetto è presente in parte per semplificare la struttura attuale, ma potrebbe essere in seguito modificato.

- Strategico, poichè lo stato successivo dell'ambiente non è determinato dalle mosse di un agente, ma deve tener conto anche delle mosse degli altri agenti, che, pur essendo cooperativi, potrebbero risultare imprevedibili. Il direttore decide le proprie mosse che influenzano la globalità del sistema, ma non ha controllo su ogni singolo agente.
- Episodico nel caso del musicista, poichè le azioni intraprese da quest ultimo non hanno ripercussioni future nè costituiscono un parametro di decisione nell'episodio seguente. La percezione del musicista è definita dalle informazioni pervenute dal direttore. Nel caso del direttore invece l'ambiente assume una connotazione sequenziale, poichè alcuni parametri dell'azione corrente determinano una probabilità di passaggio a differenti azioni successive possibili.
- Statico, poichè solo gli agenti coinvolti possono variare l'ambiente.
- Discreto poichè, pur operando in un'ottica real-time, le azioni degli agenti si basano su unità di tempo atomiche uguali per tutti, come, del resto, la teoria musicale impone.
- Multiagente, anche se le interazioni reali fra agenti sono relativamente scarse. Questo fa di un ambiente concettualmente cooperativo, in realtà un ambiente composto da unità che dagli altri agenti possono essere viste come stocastiche e imprevedibili.

## 4 Overview dei Componenti

Il progetto è composto principalmente di tre parti principali:

Un **direttore**, il quale svolge il compito di centralizzare l'improvvisazione, dettando regole e decidendo i parametri generali per ogni punto dell'improvvisazione; potrebbe in un certo senso essere visto come una sorta di *coscienza comune*, la quale amministerebbe silenziosamente i vari musicisti, vedremo più avanti come la natura centralizzata del direttore inoltre aiuti, ad esempio, a sincronizzare i vari musicisti.

Un **player**, il quale assieme al direttore compone l'architettura centralizzata del progetto; il compito del player è di ricostruire e di assemblare le varie improvvisazioni provenienti dai vari musicisti, ha inoltre un'interfaccia modulare per salvare o riprodurre l'improvvisazione, per analogia con il direttore, il quale è il punto d'ingresso del progetto, il player è il punto dove viene formato l'output utile del progetto.

Vari **musicisti**, i quali prendono decisioni in base alla loro configurazione e all'output del direttore, processandole secondo vari meccanismi euristici (vedremo ad esempio implementazioni di meccanismi randomici o basati su algoritmi evolutivi).

Al fianco di questi componenti fondamentali è presente un ambiente di supporto per facilitare l'operazione, quali ad esempio il **database** dell'applicazione, responsabile dell'immagazzinamento della conoscenza dei vari componenti, ad esempio la rappresentazione dei vari generi e dei loro pattern collegati.

## 5 Componenti del Sistema

Come indicato in precedenza, i principali componenti in stretto contatto tra loro sono delle seguenti tre categorie, è stata scelta un'implementazione a camere stagne tra di essi, in modo da rendere il sistema il più modulare possibile, potendo separare (anche fisicamente, distribuendoli su varie macchine) i vari componenti, utilizzando dei processi singoli per ogni istanza del singolo.

Si procede quindi con la descrizione dettagliata del comportamento di ogni elemento del sistema.

### 5.1 Direttore

Il direttore rappresenta la componente che lega ed indirizza l'intera esecuzione. Basa le proprie scelte su un database di generi e sottogeneri, descritti tramite pattern. Ogni pattern definisce la struttura base di un "giro" dello specifico sottogenere, ad esempio la struttura di un classico blues bebop, indicando la sequenza di accordi e di "intenzioni" consigliate per ogni battuta del giro. Sfruttando queste informazioni ed essendo a conoscenza di ciò che è già stato suonato, il direttore decide tramite meccanismi probabilistici, battuta in battuta, le caratteristiche della prossima misura di brano da eseguire, elegge un solista per questa sezione e comunica queste informazioni ai musicisti, per poi attendere che tutti

abbiano completato la propria esecuzione. Ad ogni nuova battuta, utilizzando metodi probabilistici, il direttore decide inoltre se cambiare sottogenere o genere e quale scala di riferimento deve essere usata dal musicista solista, tenendo in considerazione le possibili varianti compatibili dello stesso genere.

## 5.2 Musicista

Come il direttore, il musicista nel nostro software è un agente. Il suo scopo principale è quello di creare in tempo reale della musica. Della buona musica? Ci prova, infatti il processo musicista trascorre la sua esistenza suonando delle note che possano “andar bene” assieme alle note suonate dagli altri musicisti. Questi ultimi non vengono lasciati soli nelle decisioni prese durante un’improvvisazione ma il direttore li aiuta a prendere delle scelte che possano aver senso fra di loro e li aiuta a coordinarsi. Il direttore quindi, tramite un certo protocollo di comunicazione, invia determinati parametri globali a tutti i musicisti che a loro volta scandiscono il database per cercare delle note che possano avere senso nel loro attuale contesto. Ad ogni insieme di note che i musicisti ottengono ad ogni passo dell’esecuzione è correlato un set di probabilità, il quale viene utilizzato per filtrare le note scelte da utilizzare e ad introdurre il comportamento di improvvisazione.

## 5.3 Player

Il player è l’unico componente del progetto che non si comporta da agente intelligente, ma piuttosto da scheduler, poichè il suo compito è suonare in contemporanea le note che i musicisti hanno “scritto sul loro spartito”. Più in particolare, le informazioni che pervengono al player sono, da ogni musicista, un set di note e durate, secondo la struttura descritta nella sezione 6, della durata di una battuta. Il player resterà bloccato finchè non riceverà questo dato da ognuno dei musicisti (un numero conosciuto a priori), poi inizierà la sua esecuzione. Essa si basa sulla scansione dei dati in input secondo una base temporale atomica (che corrisponde alla durata della metà di una semicroma terzinata, ovvero un quarantottesimo di una misura intera) con la quale si possono rappresentare in base numerica intera tutte le note utilizzate. È da segnalare che non utilizziamo mai note di durata inferiore alla semicroma terzinata (quindi biscroma e semibiscroma, per i più avvezzi alla teoria musicale) per pura semplicità, poichè questo non pregiudica una buona dimostrazione del funzionamento del software.

Il player nasce come esecutore in tempo reale (o immediatamente successivo) rispetto alla creazione del brano, ma presenta una feature interessante, ovvero la scrittura su file midi del brano in creazione per esecuzioni future o studio dello spartito generato su tools come tuxguitar.

Questo componente utilizza una labeling standard degli strumenti musicali, coerente con quella utilizzata da TiMidity++<sup>1</sup>, il tool che abbiamo sfruttato per tradurre il nostro operato in qualcosa di udibile.

## 6 Interazione e Comunicazione

La comunicazione è uno dei punti cruciali del progetto, sia per la quantità di informazione scambiata, che per la sincronia dei messaggi.

---

<sup>1</sup> <http://timidity.sourceforge.net/>

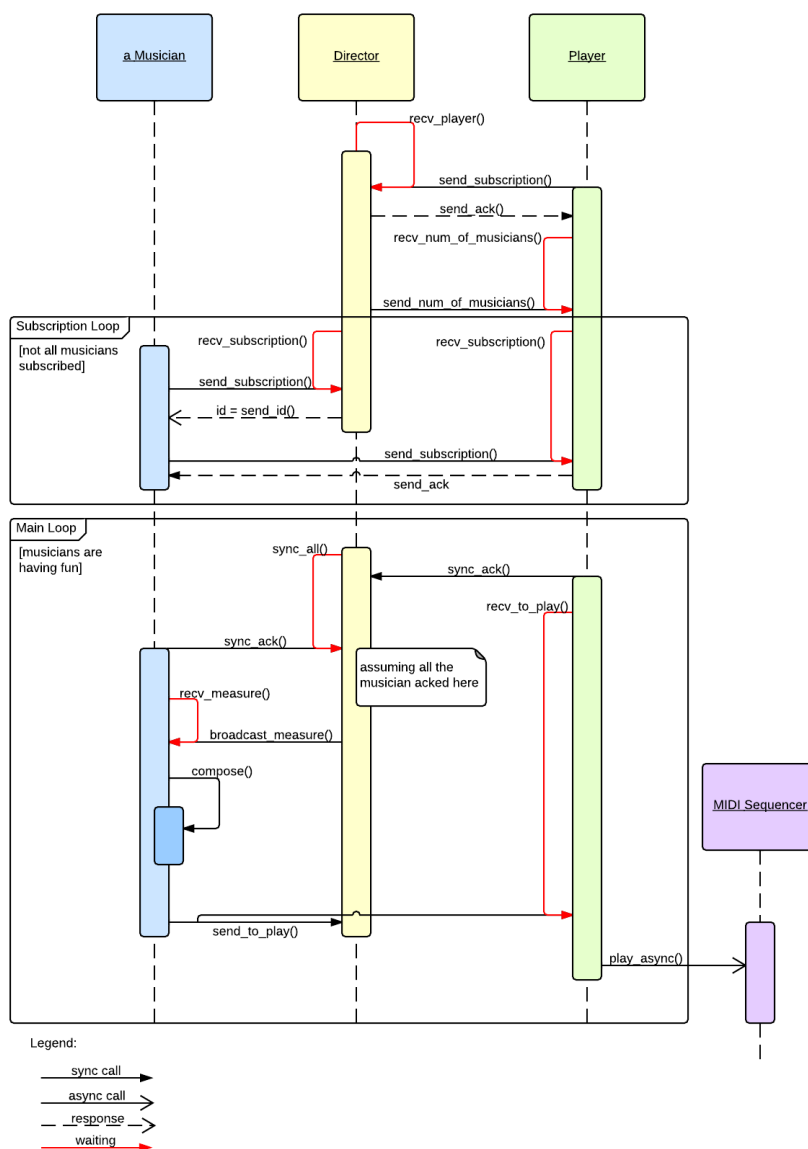


Fig. 2: Diagramma del protocollo di comunicazione

Il protocollo è strutturato principalmente in tre fasi distinte.

## 6.1 Inizializzazione

Inizialmente il direttore riceve una sottoscrizione da parte del player.

Alla sottoscrizione (opportunamente segnalata tramite *ack*), susseguirà un pacchetto contenente il numero di musicisti <sup>2</sup>. A questo punto l'inizializzazione dei componenti principali (direttore e player) è completa, e la loro comunicazione, come vedremo si limiterà a messaggi di sincronizzazione.

## 6.2 Ciclo di Sottoscrizione

A questo punto, ogni musicista si preoccuperà di inviare la sua sottoscrizione all'improvvisazione sia al direttore che al player, in questo modo le due componenti principali avranno una chiara visione dell'improvvisazione, pur rimanendo completamente indipendenti.

<sup>2</sup> In modo da specificare solamente al direttore il numero di istanze di musicisti.

## 6.3 Ciclo principale

Finita l'inizializzazione inizia il ciclo vero e proprio d'improvvisazione; una volta sincronizzati tutti i musicisti (e il player) a barriera, vengono inviate da parte del direttore le informazioni di improvvisazione, come vedremo più avanti questi pacchetti contengono tutte le informazioni sullo stato dell'improvvisazione, in modo da mantenere una certa coerenza tra tutti i musicisti.

Una volta composto, i musicisti inviano quindi la loro creazione<sup>3</sup> al player, il quale si occuperà di suonare la composizione nell'ordine corretto; Mentre il direttore si occuperà della sincronizzazione, proseguendo con un nuovo passo del ciclo.

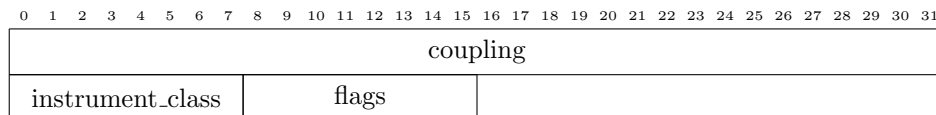
## 6.4 Libreria di comunicazione

L'implementazione della libreria di comunicazione è presente nei file *communication.cpp* e *struct.h*, se ne descrivono di seguito i dettagli.

I principali pacchetti scambiati tra le varie istanze sono di 3 tipi: *Sottoscrizioni*, *Misure* e *Play Measures*.

### 6.4.1 Sottoscrizioni

Sono i pacchetti scambiati per la registrazione presso il Direttore o il Player



Essi contengono:

#### Coupling

Un indicazione (per il player) sul fatto che il musicista sia in realtà una composizione di più musicisti<sup>4</sup>

#### Instrument Class

Il tipo di strumento, utilizzato per la ricerca nel database e per l'assegnamento del corretto strumento *MIDI* in uscita.

#### Flags

I flag disponibili per la sottoscrizione sono:

- 0x0: Nessun flag rilevante.
- 0x1: Il musicista è un solista.
- 0x2: Il musicista utilizza pratiche di machine learning genetiche.

Da qui in poi sono riservati per utilizzi futuri.

La risposta a questi pacchetti non si limita al semplice ack, ma bensì ad un pacchetto contenente l'id (univoco per la sessione) del musicista.

È necessario indicare che la registrazione del il player presso il direttore avviene inviando questo pacchetto con un coupling pari ad un valore costante (-1).

<sup>3</sup> Si rimanda alle opportune sezioni per approfondire come queste decisioni vengano prese.

<sup>4</sup> Immaginiamo ad esempio la mano sinistra e la mano destra dello stesso pianista, le quali devono essere assegnate allo stesso canale di output.

### 6.4.2 Misure

Sono i pacchetti contenenti l'informazione sullo stato dell'improvvisazione e il prossimo passo d'improvvisazione, sono inviati in broadcast dal direttore a tutti i musicisti.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
bpm																																	
soloist_id																																	
tempo.upper								tempo.lower																									
prioargs																																} 9	
note																scale																} tempo.upper	
chord																mode																} tempo.upper	
tags length																																	
tags (variable)																																	

#### BPM

Un indicazione sui bpm dell'improvvisazione corrente.

#### Soloist ID

L'ID del musicista che improvvisa correntemente in modalità solista.

#### Tempo (upper e lower)

Indica la *Time Signature* dell'improvvisazione corrente.

#### Prioargs

Sono 9 campi costanti il quale compito é specificare una scala di priorità con la quale effettuare le scelte di improvvisazione.<sup>5</sup>

#### Note e scale

Sono *tempo.upper* campi contenti la successione dei centri tonali della misura (rilevanti per il solista).

#### Chord e mode

Sono *tempo.upper* campi contenti la successione di accordi della misura (rilevanti per gli accompagnatori).

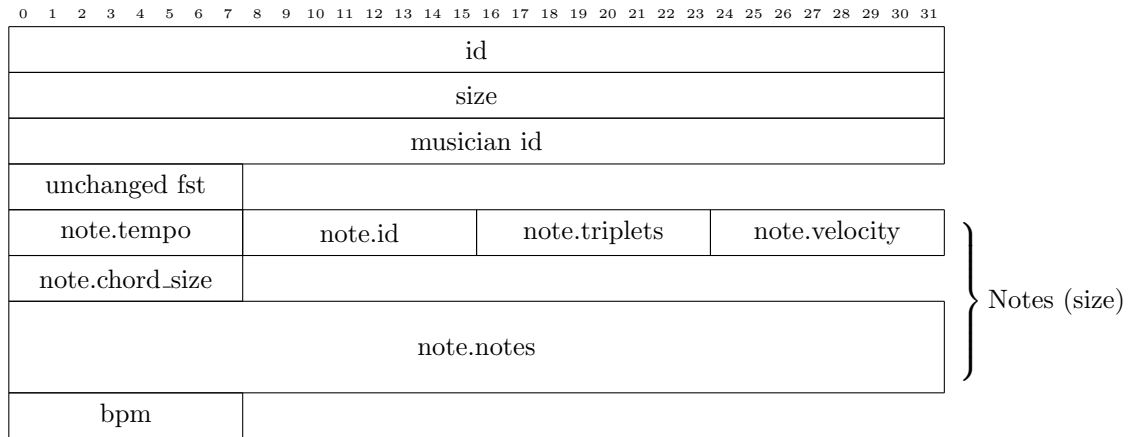
#### Tags

É un campo testuale utilizzato per indicare attributi del pezzo da improvvisare (è una terna che specifica genere, dinamiche ed intenzione).

### 6.4.3 Play Measures

Sono i pacchetti contenenti l'informazione dettagliata sulle note suonate è la composizione che, in output dal musicista, viene passata al player pronta per essere allineata con le altre battute ed essere suonata.

<sup>5</sup> Per una spiegazione più dettagliata si rimanda ai capitoli sul come operano il Direttore e il Musicista.



#### Id

Un numero progressivo (per musicista) indicante la posizione della battuta da suonare.

#### Size

Il numero di note presenti nella misura corrente.

#### Musician id

L'identificativo univoco del musicista che ha generato la misura in questione.

#### Unchanged fst

Un booleano indicante il fatto che la prima nota sia cambiata o meno. <sup>6</sup>

#### Note

Sono quindi presenti *size* strutture contenenti le seguenti informazioni

##### Tempo

La durata della nota (in sedicesimi).

##### ID

Un numero progressivo (per misura) indicante la posizione nella misura.

##### Triples

Un indicazione sul fatto che la nota debba essere scandita con tempo terzinato.

##### Velocity

La velocity *MIDI*, è un indicazione sul volume della nota in output.

##### Chord\_size

Il numero di note presenti (nel caso sia un accordo, altrimenti la nota sarà singola)

##### Notes

Il vettore di note (in notazione *MIDI*).

#### BPM

Un indicazione sui bpm da utilizzare nella riproduzione dell'improvvisazione.

## 6.5 Meccanismi di sincronizzazione

L'ultimo compito della libreria di comunicazione è provvedere ad alcuni meccanismi per la sincronizzazione dei vari componenti.

Questo obiettivo è raggiunto utilizzando le chiamate di sistema di linux basate sugli eventi quali *epoll*: vengono radunati su di una barriera (in qualsiasi ordine) tutti i musicisti, dopo di che viene notificata, tramite *ack*, la possibilità di continuare.

<sup>6</sup> Questa feature è necessaria nel per effettuare dei legati o dei continui dalla misura precedente.



## 7 Rappresentazione della Conoscenza

Generalmente la conoscenza musicale di ogni musicista, cantante, compositore o direttore d'orchestra è formata da tre componenti fondamentali:

1. esecuzioni passate dell'artista stesso
2. esecuzioni altrui ascoltate precedentemente
3. regole provenienti dalla teoria musicale

In IMprovEesation queste informazioni sono alla base del modello della conoscenza degli agenti del sistema. Le componenti 1 e 2 costituiscono i pattern musicali e le relative note associate ad essi. Inoltre nella memoria degli agenti sono presenti informazioni aggiuntive, come ad esempio scale, modo degli accordi, etc. Quest'ultime vengono messe in relazione con i pattern e le note formando così la terza componente, ovvero l'insieme di regole teoriche possedute dall'agente.

Inoltre la conoscenza degli agenti viene rappresentata come se fosse immagazzinata in un'unica memoria collettiva, che viene acceduta però in regioni diverse in base al ruolo dell'agente. Ad esempio il direttore ha accesso alla regione della memoria dove sono salvate le informazioni riguardo all'andamento complessivo di un'improvvisazione che comunicherà durante quest'ultima ai musicisti. Un agente musicista ha accesso alla regione di memoria dove salvate le informazioni necessarie per comporre in tempo reale delle note che siano coerenti in qualche modo con le informazioni fornite dal direttore.

### 7.1 Pattern e Regole

Definiamo i pattern come sequenze di misure di accordi. Il direttore ha la conoscenza di una collezione di diversi pattern che a loro volta possono ammettere delle varianti di dinamica e stile. Il musicista, nella memoria complessiva, ha accesso alle collezioni di informazioni riguardo note singole. Si è scelto di rappresentare una singola nota come una semicroma essendo  $1/16$  la suddivisione temporale più piccola che prendiamo in considerazione. Inoltre ad ogni semicroma non è associato un solo valore tonale, ma un vettore di probabilità di dimensione  $n$  pari a 13. L'indice  $i$ -esimo di ogni elemento corrisponde alla distanza tonale dalla tonalità decisa dal direttore, oppure una pausa. Ogni elemento  $i$ -esimo del vettore corrisponde al valore di probabilità  $p_i$  che la nota di distanza  $i$  dalla tonalità corrente venga selezionata ad un certo istante di tempo  $t$ . Per ogni semicroma è associato anche un valore di probabilità  $p_c$  utilizzato per decidere se quella semicroma e il suo vettore di probabilità debba essere considerato ad un certo istante di tempo  $t$ . Nella memoria del musicista sono presenti anche i quarti che raggruppano fino a 4 semicrome ognuno. Ai quarti sono correlate ulteriori informazioni, utili a comprendere il contesto di appartenenza di un certo quarto, come:

- posizione del quarto in una misura
- l'accordo associato
- il modo dell'accordo
- strumento associato
- dinamica del quarto
- mood (stile) del quarto
- se lo strumento associato è solista o meno

Mettendo in relazione queste informazioni con quelle fornite dal direttore durante un'improvvisazione, un musicista cerca di scegliere delle note che siano il più possibile inerenti al contesto. Come ciò viene fatto verrà spiegato nella sezione 8.

## 7.2 Database Relazionale

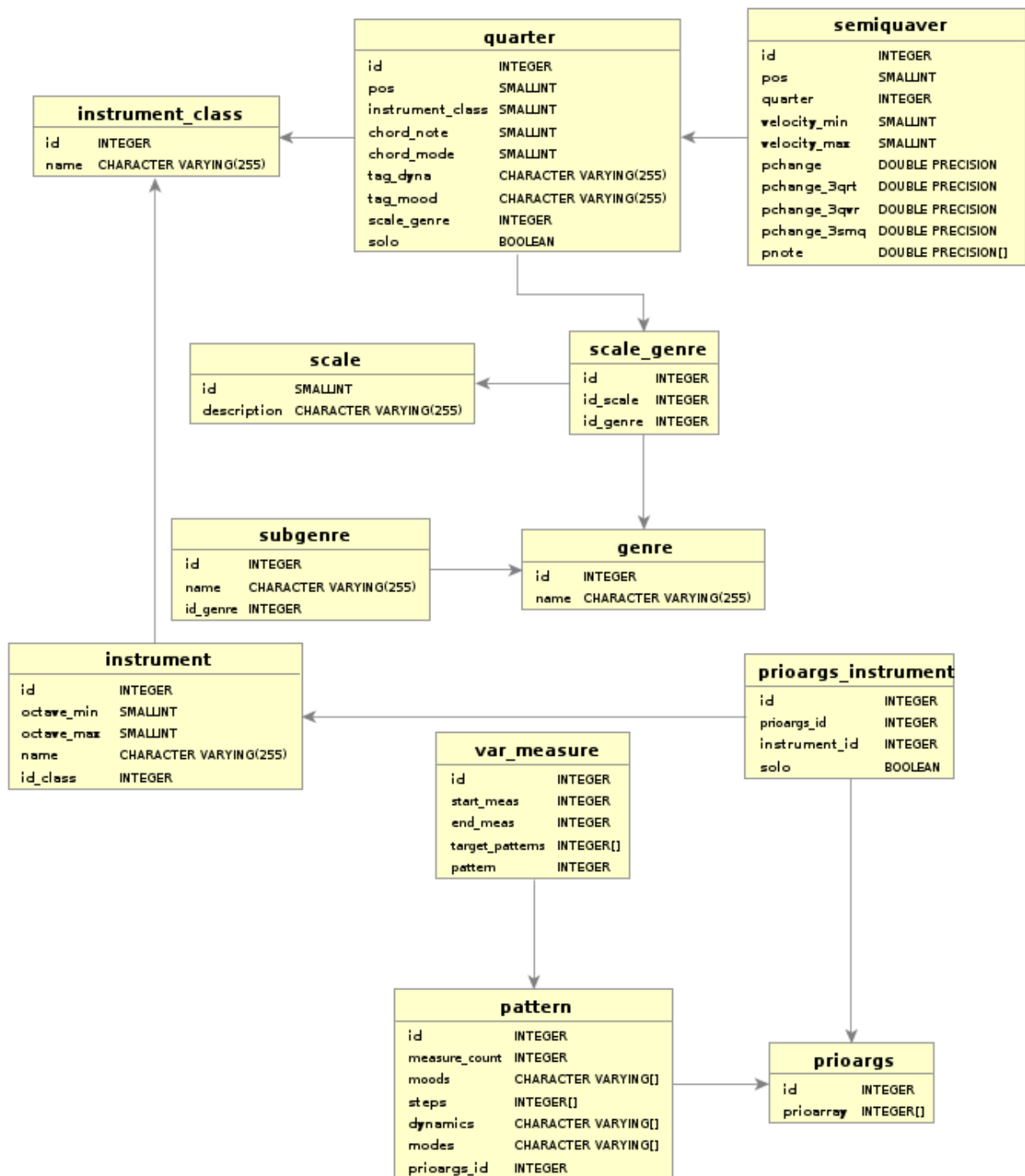


Fig. 3: Schema E-R del DataBase

La struttura di conoscenza spiegata nel paragrafo precedente è stata implementata tramite un DataBase relazionale il cui schema Entità Relazione è mostrato nella figura 3.

La struttura si può pensare come divisa essenzialmente in due parti principali.

La prima si concentra intorno alla tabella *quarter*, attraverso la quale il musicista riesce a capire quali

*semiquaver* considerare in base ad un certo contesto definito dal direttore come già spiegato nella sezione precedente. È importante notare che nonostante la memoria sia collettiva a tutti gli agenti, tramite le informazioni legate ai quarti un musicista è in grado di filtrare le informazioni che meglio si adattano a se stesso. Oltre al contesto definito dal direttore infatti, il musicista può anche filtrare le informazioni che riguardano la sua natura, come ad esempio il tipo di strumento da lui suonato o se si sta comportando o meno da solista.

La seconda parte fondamentale del DataBase è concentrata intorno alla tabella *pattern*. Dalla figura 3 potrebbe sembrare che l'entità *pattern* non abbia poi così tante relazioni come l'entità *quarter* per considerare quest'ultima un parte fondamentale del DB. Questo deriva dal fatto che si è deciso di semplificare un po' il modello E-R per l'entità *pattern* utilizzando dei vettori per ogni record di quest'ultima anziché creare delle nuove entità-relazioni. Questa scelta è giustificata dal fatto che viene inizialmente considerato un numero relativamente basso di pattern e che il modello è facilmente estendibile in quanto le due parti fondamentali del DataBase hanno due schemi abbastanza indipendenti tra loro.

Nello specifico la tabella *pattern* è caratterizzata fondamentalmente dai seguenti campi vettoriali

- steps
- modes
- dynamics
- moods

tramite i quali un direttore riesce a scandire l'andamento dell'improvvisazione misura per misura, definendo a quali accordi riferirsi e suggerendo ai musicisti con quale dinamica e sentimento eseguire le loro performance. Queste direttive possono venire suggerite ai musicisti in modo stretto o lato.

A tal proposito ad ogni pattern viene associato un vettore di priorità dalla tabella *prioargs* che definisce con quale priorità le direttive del direttore riguardanti quel pattern debbano essere considerate dai musicisti, anche rispetto alle altre informazioni sulla loro stessa natura. È anche possibile che un musicista non consideri affatto le priorità legate al pattern ma utilizzi delle priorità ad-hoc, definite rispetto ad altri parametri.

Ad esempio un musicista solista potrebbe considerare un vettore di priorità ad-hoc che non sia legato a nessun pattern in quanto potrebbe non essere interessato alla successione di accordi tra le misure, informazione molto forte per i pattern, ma solamente alla tonalità corrente, alle dinamiche e ai mood delle misure.

C'è poi il caso particolare del batterista che non ha alcun interesse riguardo le informazioni inerenti alla linea melodica e armonica di un'improvvisazione, ma è interessato solo alla linea ritmica. Inoltre per il batterista viene utilizzato un modello diverso di vettore di probabilità di ogni *semiquaver*, quindi è fondamentale che la sua priorità più alta sia il tipo di strumento, altrimenti, incorrendo in vettori di priorità associati a quarti di altri strumenti, potrebbe interpretare in maniera completamente errata tali informazioni.

Dal lato applicazione le informazioni residenti nel DB vengono recuperate tramite una libreria dinamica dedicata. Quest'ultima si occupa di effettuare le apposite query per recuperare le informazioni, con i dovuti vincoli, necessarie ai processi musicali e direttori per improvvisare.

## 8 Ragionamento Automatico

Quando un musicista o un gruppo di musicisti eseguono un'improvvisazione, sono molteplici i fattori che incidono sulle loro azioni. Tra questi sicuramente hanno molta influenza le emozioni individuali e reciproche dei musicisti, oppure il feeling che c'è tra di loro.

Per un software che genera delle improvvisazioni musicali, il fattore emotivo non può essere escluso. Anche se le macchine non sono in grado di provare delle emozioni reali, possono sempre simulare dei comportamenti che sembrano, ai sensi di un osservatore esterno, essere influenzati da esse, e forse possono anche scaturire emozioni a chi interagisce con tali macchine. Quest'ultimo è un obiettivo arduo ma sicuramente interessante per software di questo genere. I puristi della musica e dell'arte potrebbero sollevare una critica dicendo che un musicista che non prova emozioni non è in grado di sollevare emozioni verso

chi li ascolta. Lo studio emotivo nell'intelligenza artificiale è un campo aperto e osservato con interesse dal mondo scientifico, risulta quindi stimolante proporre un'applicazione di studio.

Inoltre, anche se non è l'unico fattore in gioco, non può mancare un certo tipo di ragionamento nella scelte effettuate da ogni musicista, che sa in qualche modo come agire per produrre un risultato che sia apprezzabile per se stesso e per gli altri.

In *IMprovEEsation* il ragionamento dei musicisti è fortemente influenzato dal ragionamento del direttore che potrebbe essere visto come una sorta di sintonia tra i vari musicisti improvvisatori, che in un caso reale non avrebbero un coordinatore centrale.

In questa sezione vedremo meglio come ragionano il direttore e i musicisti durante una sessione di improvvisazione.

## 8.1 Mente del Direttore

Il compito del direttore è quello di controllare il flusso dell'improvvisazione nella sua globalità, istruendo i musicisti su cosa e come “sarebbe corretto” suonare. Per ogni battuta, la mente del gruppo stabilisce in una serie di step successivi:

- il genere (e sottogenere/variante)
- gli accordi di riferimento per l'accompagnamento
- i centri tonali per l'improvvisazione
- il musicista solista

Inoltre vengono settati in maniera statica (ma è prevista la possibilità di deciderli dinamicamente):

- velocità
- tempo
- dinamiche ed intenzione
- priorità degli argomenti delle query per il musicista (vedi ??)

Questi valori sono letti dal file di configurazione e dal database, nello specifico i pattern dei generi dichiarano dinamiche ed intenzione per ogni battuta ed un set di priorità di argomenti da scartare.

Per lasciare massima libertà di sperimentazione, tutte le soglie variabili degli algoritmi probabilistici sono lette dal file di configurazione durante la fase di inizializzazione del direttore. Un file di configurazione di esempio è fornito insieme al programma, i valori qui contenuti sono quelli discussi nei prossimi paragrafi e rappresentano ciò che si è ritenuti più “sensato”, sfruttando anche la consulenza di musicisti esperti.

### 8.1.1 Scelta del genere

Per prima cosa è necessario stabilire, per ogni battuta, se è possibile cambiare l'intento globale dell'improvvisazione, la maniera di ottenere questo cambiamento drastico è scegliere un genere o sottogenere differente da quello attualmente in uso.

Il concetto di sottogenere identifica una serie di varianti dello stesso genere musicale, non necessariamente intercambiabili l'una con l'altra, ma certamente appartenenti alla stessa macrocategoria, per esempio sono stati definiti pattern per blues base, che è il classico blues in dodici battute, il blues bebop ed il blues Coltrane che si basa sulla famosa progressione blues inventata dal noto sassofonista.

Prima di tutto vanno considerate due situazioni differenti, in base alla battuta che si andrà a creare:

1. la battuta è la prima di un “giro”;
2. la battuta è una battuta qualunque all'interno di un “giro” (e.g. non la prima).

Il comportamento implementato è quello di NON cambiare mai genere se non sulla prima battuta di un giro, in modo da caratterizzare ogni parte dell'improvvisazione in maniera sufficientemente chiara e plausibile.

In ultimo viene deciso, attraverso una soglia percentuale, se è il caso di cambiare sottogenere oppure passare proprio ad un genere differente. Poiché il dataset creato non contiene altri generi propriamente implementati all'infuori del blues, questa soglia è fissa sul cambio di sottogenere; in ogni modo un valore plausibile è il 75% di probabilità di passare ad un sottogenere, invece di cambiare il genere in toto.

### 8.1.2 Accordi di riferimento

Una volta stabilito il genere è necessario decidere gli accordi che andranno suonati dai musicisti di accompagnamento. Il primo riferimento considerato è quello dichiarato dal pattern del genere: ogni pattern specifica la progressione di accordi tipica di un giro di quello specifico sottogenere (ad esempio il blues base sarà descritto dalla sequenza di accordi maggiori con la settima minore costruiti sui gradi I-I-I-I-IV-IV-I-I-V-V-I-I). Basandosi su questa informazione ed essendo a conoscenza di quale battuta si sta per suonare all'interno della progressione, il direttore stabilisce se è plausibile cambiare l'accordo di riferimento rispetto a quello dichiarato dal pattern: si considerano accordi sostituibili quelli che coprono l'intera battuta oppure sono una cadenza<sup>7</sup>.

Il secondo nodo della decisione è guidato dalla posizione della battuta all'interno del "giro", analogamente a quanto accade per la sostituzione del genere:

- se la battuta è la prima di un giro, l'accordo non dovrebbe essere mai sostituito (tendenzialmente la prima battuta è quella che dichiara la tonalità)
- se la battuta non è la prima, con bassa probabilità si potrà cambiare l'accordo.

Se, a questo punto, il direttore ha deciso di introdurre una variazione, utilizzerà una scala di probabilità per decidere il criterio secondo cui modificare l'accordo:

- 30% Sostituzione di tritono
- 30% Cadenza relativa all'accordo originale
- 30% Sostituzione per zone tonali<sup>8</sup>
- 10% Accordi casuali

### 8.1.3 Centri tonali

Un altro fattore che influenza sostanzialmente la qualità dell'improvvisazione è la scala di riferimento utilizzata dai musicisti solisti. La teoria musicale propone svariate regole estremamente rigide per quanto riguarda le scale utilizzabili in determinate circostanze, ma queste regole portano molto spesso a risultati banali o prevedibili, che sono desiderabili in alcune circostanze ma alla lunga renderebbero l'improvvisazione noiosa.

Per non limitarsi ad una scelta meccanica della scala di riferimento, viene dunque utilizzato un sistema di selezione non del tutto banale:

1. viene stabilito il centro tonale;
2. vengono controllate le scale disponibili;
3. viene selezionata la scala definitiva.

La parte cruciale è la selezione del centro tonale, che viene effettuata attraverso un meccanismo iterativo di valutazione dei centri tonali più plausibili:

<sup>7</sup> La cadenza è la successione all'interno della stessa battuta di un accordo minore di settima minore sul secondo grado e un accordo maggiore di settima minore sul quinto grado; questa circostanza è armonicamente assimilabile ad un accordo sul primo grado

<sup>8</sup> La sostituzione per zone tonali offre due possibilità: VI grado relativo minore con settima minore, oppure III grado relativo minore (o diminuito). Le due possibilità hanno uguale probabilità di venire scelte.

---

**Algorithm 1**

---

```
1: function DECIDEIMPROSCALE(currentMeasure)
2:   for range  $\leftarrow$  totalMeasuresCount; range < 0; range -- do
3:     for i  $\leftarrow$  0; i < range; i ++ do
4:       meas  $\leftarrow$  pattern[(currentMeasure - (range/2) + i)%12]
5:       if meas.chordsNum = 1 or isCadenza(meas) then
6:         found[meas.chord] ++
7:       else
8:         for j  $\leftarrow$  0; j < meas.chordsNum; j ++ do
9:           found[meas.chords[j]] ++
10:        end for
11:      end if
12:    end for
13:    max  $\leftarrow$  getMaxIndex(found)
14:    tonalZones[max] ++
15:  end for
16:  r  $\leftarrow$  rand()%sizeof(tonalZones)
17:  for i  $\leftarrow$  0; i < 12; i ++ do
18:    if tonalZones[i] > 0 then
19:      if r < tonalZones[i] then
20:        note  $\leftarrow$  i
21:        break
22:      else
23:        r  $\leftarrow$  r - tonalZones[i]
24:      end if
25:    end if
26:  end for
27: end function
```

---

Il primo blocco del metodo effettua una scansione del pattern attuale per contare il numero di occorrenze dei centri tonali considerati battuta per battuta e salva quello più gettonato, continua successivamente la scansione restringendo il range di ricerca e mantenendolo centrato sulla battuta attuale fino ad avere la sola misura che deve essere suonata.

Con questo sistema è possibile creare un vettore dei centri tonali considerabili validi per la battuta attuale, viene dunque utilizzato questo vettore come partizionamento di probabilità per scegliere il centro tonale definitivo.

Una volta deciso il centro tonale, vengono confrontate tutte le scale disponibili nel database, nella tonalità decisa dal centro tonale, con l'accordo che verrà suonato dall'accompagnamento. Se l'accordo è interamente contenuto nella scala presa in esame, questa viene aggiunta ad una lista di scale accettabili<sup>9</sup> ed ne viene selezionato un elemento casuale.

Se nessuna scala è ritenuta accettabile a questo punto, viene scelta quella che ha il maggior numero di gradi contenuti nell'accordo di riferimento.

Questo sistema è quello implementato nella versione attuale del software, ma è già pronto il supporto per le varianti di sottogenere. Si tratta di un'ulteriore variazione alla scelta della scala su cui improvvisare: ogni sottogenere specifica una lista di altri sottogeneri per cui tutto il pattern o alcune sue parti sono plausibili varianti improvvisative.

Le varianti improvvisative sono un'indicazione utile per il solista, poiché gli offrono una maggiore gamma di intenzioni, oltre ad un ovvio incremento della varietà improvvisativa in genere. Nella pratica, un musicista esperto sa che, considerando i centri tonali di un pattern come, ad esempio, il blues di Coltrane, può suonare una assolo decisamente atipico ma estremamente coerente su un accompagnamento che suona un blues basilare.

Il direttore che volesse sfruttare questa occasione potrebbe quindi decidere, prima dell'analisi del pattern corrente alla ricerca del centro tonale, di sostituire parti del pattern con quelle provenienti da sottogeneri compatibili, in modo da ottenere una maggiore varietà di sonorità senza ricadere nella casualità più totale.

#### 8.1.4 Scelta del solista

La scelta del solista viene effettuata in base ad un principio generale derivato dalla composizione dal vivo<sup>??</sup>: la band ha un leader, il quale improvviserà mediamente il doppio degli altri musicisti solisti.

Seguendo questo principio, all'inizio dell'improvvisazione viene eletto un leader, che sarà il primo strumento ad improvvisare. In secondo luogo, tenendo conto del numero totale di solisti e della durata dell'improvvisazione prevista, si stabilisce la durata degli *slot di improvvisazione* disponibili per ogni solista, in modo da lasciare due slot al leader ed uno slot vuoto per eventuali variazioni della dimensione degli altri slot oppure per un giro senza improvvisazione prima del finale.

Prima di cominciare l'improvvisazione viene dunque deciso l'ordine di improvvisazione dei musicisti, se la durata prevista dell'improvvisazione è meno di tre giri completi viene riservato spazio per improvvisare soltanto al leader della band, altrimenti lui avrà il primo e l'ultimo giro dell'improvvisazione, gli altri sono scelti in maniera casuale distribuendo i giri tra i vari strumenti fino a che ognuno non ha esaurito il proprio slot di improvvisazione.

Durante la creazione di una nuova battuta, viene quindi impostato come solista quello che, secondo la procedura effettuate in fase di inizializzazione, ha il diritto di improvvisare durante il giro corrente.

## 8.2 Mente del Musicista

Come già introdotto nelle sezioni precedenti, il compito principale del musicista è quello di decidere le note da suonare basandosi sulle informazioni ricevute dal direttore che le manda in broadcast in rete per ogni misura da improvvisare. L'algoritmo 2 mostra come un musicista compone una misura da improvvisare basandosi sulle informazioni ricevute dal direttore su di essa,  $data_{MSR}$ , e sulle informazioni inerenti alla sua natura,  $data_{MCN}$ .

All'inizio il musicista cerca nel DB dei *quarter* che possano avere un riscontro con i due set di informazioni. Se non trova nessun quarto che fa match con tutte le informazioni ripete la ricerca scartando

---

<sup>9</sup> La scala blues segue delle regole differenti, poiché si tratta di una scala speciale che non farebbe match con un accordo completo e d'altro canto può essere proficuamente utilizzata in svariate situazioni in cui la sovrapposizione con l'accordo di riferimento è scarsa.

---

**Algorithm 2** Musician measure improvisation algorithm

---

```
1: function COMPOSEMEASURE( $data_{MSR}$ ,  $data_{MCN}$ )
2:   for  $q \leftarrow 0$  to  $nq$  do       $\triangleright nq$ , the number of quarters in a measure, is time signature dependent
3:      $n, i \leftarrow 0$ 
4:     while  $n = 0$  do
5:        $(n, quarters) \leftarrow \text{searchQuarters}(db, data_{MSR}, data_{MCN}, prioargs, i++)$ 
6:     end while
7:      $quarter \leftarrow \text{random}(quarters)$ 
8:      $(n, sqs) \leftarrow \text{getSemiquavers}(db, quarter)$ 
9:      $s \leftarrow 0$ 
10:    for  $i \leftarrow 0$  to 4 do           $\triangleright$  For the all quarter subdivisions
11:       $sq \leftarrow sqs[s]$ 
12:      if  $s = n$  or  $sq_{pos} \neq i$  then
13:         $\text{extendPrevNote}()$ 
14:        continue
15:      end if
16:      if  $sq_{pc} < \text{rand}()$  then       $\triangleright$  Probabilty of note change
17:         $\text{extendPrevNote}()$ 
18:      else
19:         $\text{decideNote}(sq_{pnote})$ 
20:      end if
21:       $s++$ 
22:    end for
23:  end for
24: end function
```

---

gli argomenti con priorità più bassa, fino a quando non ne trova almeno uno. Tra i quarti trovati, se più di 1, ne viene scelto uno random <sup>10</sup> di cui il processo ottiene tutte i record *semiquaver* associate.

## 9 Appendimento

In questa sezione si vedrà l'agente musicista agire come un agente in grado di apprendere, che costituisce una base sulla quale esso tenderà sperabilmente a migliorarsi sempre più. È noto, come spiegato nelle sezioni precedenti, che il musicista non è un agente basato su un obiettivo, difatti si può dire che un vero obiettivo non esista. Vogliamo che esso agisca producendo una "bella parte musicale", ma non abbiamo una parte musicale perfetta da assegnare, poichè, essendo un'improvvisazione, si perderebbe il concetto stesso di improvvisazione, inoltre non abbiamo un modo perfetto per assegnare a questa melodia un voto, poichè ci sarebbero da considerare un numero spropositato di variabili che in ogni caso non potrebbero valutare il prodotto in toto perchè questo dovrebbe considerare l'intera esecuzione di tutti gli strumentisti assieme. Questo ovviamente non ha senso nel dare un voto al musicista singolo perchè esso non ha coscienza degli altri musicisti e perciò potrebbe produrre un brano che suoni bene con certi musicisti e suoni male con altri. Come fare allora per produrre un brano che non sia perfetto (perchè non sarebbe improvvisato), ma che si ispiri a un brano esistente per produrre qualcosa sul genere e che possa migliorare se stesso?

La nostra risposta è stata: con un algoritmo evolutivistico. Esso ha la peculiarità, una volta fornito un ideale e una buona funzione di fitness, di produrre un brano musicale "vicino" a quello fornito come ideale senza mai essere identico.

### 9.1 Algoritmo Evolutivistico

Il nostro algoritmo evolutivistico può venire applicato, nella pratica, attraverso un flag in input al programma musicista al momento del lancio, questo fa sì che alcuni musicisti possano essere "genetici"

---

<sup>10</sup> In una futura estensione, un quarto potrebbe venir scelto con una politica migliore, ad esempio facendo un confronto ulteriore tra quelli scelti e i precedenti.



e altri no. Assieme al flag deve essere fornito il path al file che contiene il pattern ideale in formato .gme (caratteristico del progetto e spiegato nella sezione seguente). In questo modo, il musicista scriverà contemporaneamente due “spartiti” (nella pratica, due array di strutture che denotano le note suonate): lo spartito classico, improvvisato con la tecnica descritta nella sezione 8.2 e immagazzinato in una struttura apposita, invece di essere passato al player, e lo spartito ideale, estratto misura per misura dal file .gme utilizzando i parametri forniti dal direttore e immagazzinato in una struttura analoga alla precedente. Entrambe queste strutture, che chiameremo rispettivamente “iniziale” e “ideale”, vengono passate all’algoritmo evoluzionistico il quale, prima di processarle, si assicurerà che ambedue constino dello stesso numero di note, allungando per ripetizione la più corta.

L’algoritmo evoluzionistico, prima di entrare in loop, si costruisce un numero fissato (ora 512) copie della struttura iniziale in quello che è il pool genetico. Il loop seguente è costituito da un numero prefissato (ora 1500) di iterazioni che constano di quattro passaggi fondamentali[2]:

- Il primo passaggio è la **Point Mutation**, che introduce la mutazione nella generazione e agisce separatamente su tutti i membri del pool genetico. Nel nostro caso, si limita a variare il pitch e/o la durata di  $X$  note random dove  $X$  è pari a circa un ventesimo nel numero di note totali. Anche la variazione dei valori ha base randomica, chiaramente è molto più facile che il valore sia in qualche modo vicino al valore esistente piuttosto che molto lontano (è difficile che una semibreve diventi una semibreve oppure che una nota molto acuta divenga una nota molto grave).
- Il secondo passaggio è il **Sorting**, che si limita a ordinare i membri del pool genetico in base alla funzione di fitness in modo che siano disposti per similitudine decrescente con la struttura ideale. La funzione di fitness verrà spiegata in una sezione separata in seguito. Per l’ordinamento è stato scelto l’algoritmo mergesort, che ha complessità uniforme in casi ottimi e pessimi. L’algoritmo quicksort è stato scartato perchè è facile che, a ogni iterazione, il pool genetico sia vicino all’ordinamento, caso che rende il quicksort vicino al suo caso pessimo.
- Il terzo passaggio è la *Recombination*, che, come in molti algoritmi evoluzionistici esistenti, opera su coppie di membri del pool genetico. Nel nostro caso, esso opera su coppie adiacenti nell’ordinamento, solo nel top 25% del pool genetico. Esso sceglie, per ogni coppia, una posizione random sulla struttura, detta crossover, che taglia entrambi i membri della coppia in due parti, poi scambia le due seconde parti all’interno della coppia. Questo set risultante di un quarto del pool genetico (nel nostro caso 128 elementi) viene sostituito al 25% inferiore nell’ordinamento, in modo che il top 24% non venga toccato, ma soltanto copiato e siagisca per ricombinazione sulla copia.
- Il quarto passaggio, proprio solo di alcuni algoritmi evoluzionistici, è la **Transposon Propagation**. Questa tecnica opera singolarmente solo sui membri del pool genetico che hanno subito la Recombination. Essa sfrutta il fatto che nella musica siano frequenti le ripetizioni: determina due punti di crossover all’interno della struttura in modo che essi denotino un frammento di quest’ultima. Tale frammento viene copiato e sovrascritto su un’altra sezione della struttura in una posizione anch’essa casuale.

### 9.1.1 La Funzione di Fitness

L’ordinamento viene eseguito utilizzando una funzione di fitness apposita che determina la similitudine della struttura con la struttura ideale di partenza. Tale similitudine è una combinazione lineare di una serie di grandezze di seguito elencate in realzione a dei pesi predeterminati:

- La similitudine punto a punto, che sarà più alta qualora in una stessa posizione sarà presente lo stesso pitch e/o la stessa durata della nota.
- La Unigram Pitch Similarity è calcolata come segue:

$$\frac{\sum_{i=0}^U \frac{1}{\sqrt{1+(train(i)-test(i))^2}}}{U}$$

, dove  $U$  è il numero di Pitch Unigrams, ovvero di note singole distinte presenti nella struttura in esame e le funzioni  $train(i)$  e  $test(i)$  restituiscono il numero di occorrenze della nota di indice  $i$

all'interno dei Pitch Unigrams in esame in input rispettivamente all'interno della struttura ideale e del membro del genetic pool in esame. Questa similitudine restituisce un numero tra 0 e 1 e denota la similitudine dei pitch ovvero quanto il membro del pool genetico corrente utilizza note utilizzate anche dalla struttura ideale.

- La Unigram Tempo Similarity è calcolata esattamente come la precedente, con la differenza che fa riferimento alle durate invece che ai pitch delle note.
- La Bigram Pitch Similarity è calcolata come segue:

$$\frac{\sum_{i=0}^B \frac{1}{\sqrt{1+(train(i)-test(i))^2}}}{B}$$

, dove  $B$  è il numero di Pitch Bigrams, ovvero di coppie ordinate distinte di note presenti nella struttura in esame e le funzioni  $train(i)$  e  $test(i)$  restituiscono il numero di occorrenze del Bigram di indice  $i$  all'interno dei Pitch Bigrams in esame in input rispettivamente all'interno della struttura ideale e del membro del genetic pool in esame. Questa similitudine è analoga alla Unigram Pitch Similarity, ma è più forte in quanto analizza la co-occorrenza di coppie ordinate di note, e perciò avrà un peso maggiore.

- La Bigram Tempo Similarity è calcolata esattamente come la precedente, con la differenza che fa riferimento alle durate invece che ai pitch delle note.

Queste similitudini sono tratte dalla categorizzazione Bayesiana di testi in un tipico algoritmo di machine learning all'interno dell'NLP.

## 9.2 Il Pattern Ideale

I file di pattern ideale ospitano una serie di pattern codificati in CSV e sono semplicemente elencati come righe all'interno dei file .gme. Le caratteristiche listate in un pattern sono: dyna (come ad esempio il groove o il fill), Unchanged fst (che determina se la prima nota è un continuo di una precedente), una serie di triple <pitch, tempo, triplet>, dove triplet indica con un numero booleano, se la presente nota fa parte di una terzina o meno. Queste caratteristiche consentono al musicista di estrarre il corretto pattern conforme alle regole imposte dal direttore misura per misura. Possono essere ovviamente presenti più pattern con le stesse caratteristiche, in tal caso ne verrà scelto uno a caso, in tal modo si introdurrà un fattore casuale anche nella formazione del brano ideale.

## 10 Risultati Sperimentali

Dopo tanto sbatto funziona tutto random!

## 11 Conclusioni

Ci vuole un DB supermegagigante!!!

## 12 Sviluppi Futuri

### 12.1 Miglioramenti del Direttore

Come già dichiarato al paragrafo 8.1.3, un primo miglioramento che richiederebbe poco sforzo è l'implementazione delle varianti di improvvisazione durante la scelta della scala di riferimento per i solisti. Questa miglgoria non è stata ancora completamente implementata perchè richiederebbe un refactoring di parte del codice decisionale del direttore, ma non si tratta di una ristrutturazione eccessivamente onerosa, dato che il resto del sistema prevede già la presenza di questa feature.

Un altro passo avanti potrebbe essere la modifica dinamica dei parametri attualmente statici di ogni battuta generata. Questo è un argomento delicato, poiché la maggior parte di questi parametri hanno carattere estremamente istintivo, non esistono regole che definiscano come modificarli in maniera

corretta o sbagliata. La maniera più sensata di affrontare un simile problema potrebbe essere quella di implementare un meccanismo di machine learning basato su training manuale che sfrutti l'opinione di un numero sostanzioso di ascoltatori eterogenei e valuti di conseguenza quali modifiche sono più o meno apprezzabili.

Simili meccanismi di machine learning potrebbero portare anche a risultati interessanti per la scelta di tutti gli altri fattori determinati in maniera probabilistica, nonché una possibile evoluzione delle strutture dati contenute nel database, aggiornando i pattern con le modifiche introdotte da scelte pseudocasuali di accordi e le relazioni tra sottogeneri compatibili.

## 12.2 Sviluppo della Conoscenza

Meritano una menzione a parte gli sviluppi futuri dedicati alla parte di machine learning. In questo momento, l'algoritmo evoluzionistico fa in modo che un musicista "impari" a improvvisare data una sequenza ideale come traccia. Come rendere questo apprendimento permanente?

La nostra proposta per un futuro sviluppo è incrociare l'algoritmo evoluzionistico con l'approccio sulle probabilità. Idealmente, un utente darebbe in pasto all'algoritmo evoluzionistico il musicista con il brano ideale finché ciò che viene prodotto non sarà buono. Avvenuto questo, il prodotto dell'algoritmo dovrebbe venire trasformato in un set di probabilità (meglio ancora se si considerano più output di tale algoritmo, per facilitare la trasformazione in percentuali) che potrà aggiornare il database dei pattern, ovvero le probabilità già presenti sul database alle quali si fa riferimento durante l'approccio probabilistico.

Una seconda miglione, affine con le tecniche esistenti di machine composition e machine improvisation, sarebbe quella di considerare gli altri musicisti e le loro esecuzioni. Ciò che l'algoritmo evoluzionistico potrebbe fare in questo caso è l'alterazione della funzione che computa la similitudine in modo che vengano introdotte grandezze che non confrontano più il pool genetico con il pattern ideale, ma anche con le scelte degli altri strumentisti.

Abbiamo notato che l'algoritmo evoluzionistico tende a girare attorno a un punto fisso, dopo un repentino miglioramento iniziale, dipendente dalla lunghezza del brano. Una terza miglione consiste nel migliorarlo per aumentarne le prestazioni, soprattutto su brani di lunghezza consistente, per poter spingere quel punto fisso a percentuali di similitudine più alta.

## Riferimenti bibliografici

- [1] Homer J. Simpson. *Mmmmm...donuts*. Evergreen Terrace Printing Co., Springfield, SomewhereUSA, 1998
- [2] Murali S. N., *Music: Markov Chains, Genetic Algorithms and Scale Transformation*. on Youtube at <https://www.youtube.com/watch?v=FvrcvXpcfVw>, 2013
- [3] Gannon, P., *Band-in-a-Box*. PG Music, 1990
- [4] Miranda E. R., Biles J. A., *Evolutionary Computer Music*, Springer Ed., 2007
- [5] WASCHKA II, R. O. D. N. E. Y. Composing with Genetic Algorithms: GenDash. *Evolutionary Computer Music*. Springer London, 2007. 117-136.
- [6] Biles, John. GenJam: A genetic algorithm for generating jazz solos. *Proceedings of the International Computer Music Conference*. INTERNATIONAL COMPUTER MUSIC ASSOCIATION, 1994.
- [7] Jan Pavelka; Gerard Tel; Miroslav Bartosek, eds. (1999). Factor oracle: a new structure for pattern matching; *Proceedings of SOFSEM'99; Theory and Practice of Informatics*. Springer-Verlag, Berlin. pp. 291-306. ISBN 3-540-66694-X. Retrieved 4 December 2013. *Lecture Notes in Computer Science* 1725
- [8] S. Dubnov, G. Assayag, O. Lartillot, G. Bejerano, Using Machine-Learning Methods for Musical Style Modeling, *IEEE Computers*, 36 (10), pp. 73-80, Oct. 2003.

- [9] Pachet, Francois The Continuator: Musical Interaction with Style. In ICMA, editor, Proceedings of ICMC, pages 211-218, Göteborg, Sweden, September 2002 ICMA. best paper award