Università degli Studi di Salerno

Dipartimento di Informatica
Corso di Laurea Triennale in Informatica

Musimatica

Guitar Tablature Generation

Docente	**Studente**	**Matricola**
Prof. Roberto De Prisco	Tozza Gennaro Carmine	0512120382

Anno Accademico 2025-2026

Indice

Abstract

Il presente elaborato affronta il problema della trascrizione automatica per chitarra (Guitar Tablature Generation), una sfida classica nell'ambito della Computer Music a causa della natura "non biunivoca" della tastiera chitarristica, infatti, a differenza di strumenti come il pianoforte, dove esiste una corrispondenza univoca tra nota e tasto, la chitarra presenta un'ambiguità posizionale.

Il progetto propone lo sviluppo di un compositore automatico di tablature basato su Algoritmi Genetici (GA) implementati in linguaggio Python, con il supporto della libreria Music21 per la manipolazione simbolica delle strutture musicali. I risultati sperimentali sono presentati sia tramite visualizzazione grafica della tablatura, sia attraverso una funzionalità di riproduzione audio, che consente la verifica uditiva immediata della coerenza musicale della diteggiatura ottimizzata.

1 Introduzione

La chitarra moderna possiede sei corde. L'accordatura standard, procedendo dalla corda più grave (spessa) a quella più acuta (sottile), è definita come segue:

Numero Corda	**Nota (Notazione Angloassone)**	**Notazione MIDI**
6	E (Mi Basso)	40
5	A (La)	45
4	D (Re)	50
3	G (Sol)	55
2	B (Si)	59
1	e (Mi Cantino)	64

Tabella 1: Accordatura Standard della chitarra (E A D G B e).

Ogni tasto premuto sulla tastiera incrementa l'altezza della nota di un semitono (+1 nel valore MIDI). Questa struttura geometrica crea il fenomeno della **ridondanza**: ad esempio, la nota MIDI 64 (Mi Cantino a vuoto) può

essere prodotta anche premendo il 5° tasto della seconda corda, il 9° tasto della terza corda, e così via.

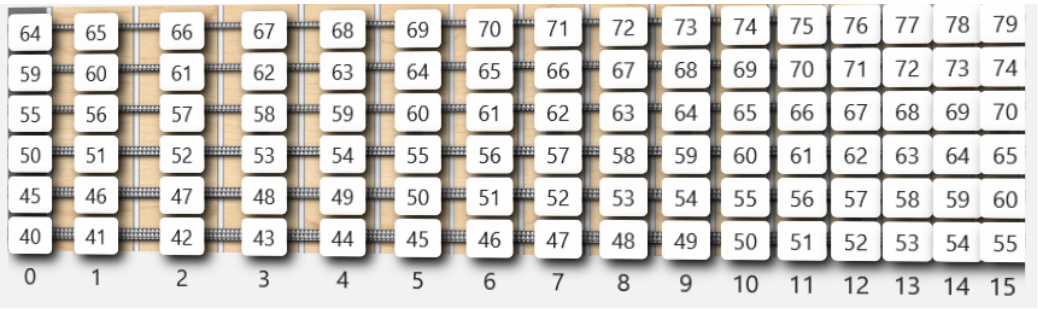

Figura 1: Codici MIDI per chitarra

La Figura 2 mostra la tastiera di una chitarra ed evidenzia quattro diverse posizioni in cui è possibile suonare lo stesso 'Mi' (E).

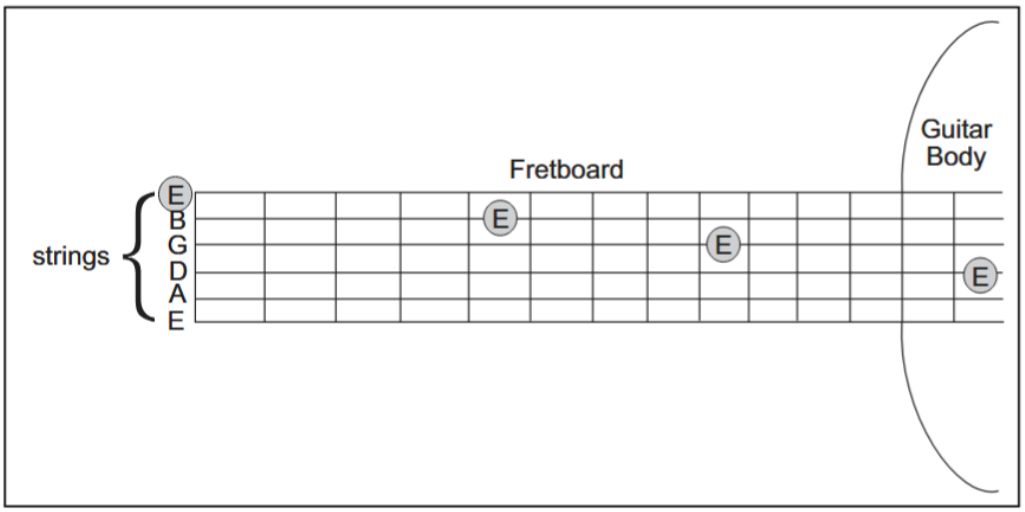

Figura 2

Gli strumenti a corda richiedono spesso una notevole esperienza e capacità decisionale da parte dell'esecutore. Per eseguire un brano musicale, l'esecutore deve stabilire una sequenza di posizioni sulla tastiera che minimizzi la

difficoltà meccanica dell'esecuzione, almeno fino a renderla fisicamente realizzabile.

Questo processo richiede molto tempo ed è particolarmente difficile per i musicisti principianti e di livello intermedio; di conseguenza, la capacità di leggere la musica direttamente dallo spartito, come farebbe un pianista, è limitata ai soli chitarristi molto esperti.

Per affrontare questo problema, si utilizza una notazione musicale nota come **tablatura**. La tablatura indica all'esecutore l'esatta modalità di esecuzione del brano, rappresentando graficamente le sei corde della chitarra e indicando, in sequenza, i tasti corrispondenti a ciascuna nota. [2]

1.1 La Tablatura

La tablatura è un sistema di notazione prescrittiva che indica all'esecutore *dove* mettere le dita, piuttosto che *quale* nota suonare (come fa il pentagramma).

Graficamente, una tablatura è composta da sei linee orizzontali che rappresentano le corde dello strumento. La linea più in basso rappresenta la corda più grave (6ª corda, Mi basso), mentre quella in alto rappresenta la corda più acuta (1ª corda, Mi cantino). I numeri posizionati sulle linee indicano il tasto da premere:

- **0**: Corda suonata a vuoto (senza premere tasti).
- **1, 2, 3...**: Il numero del tasto corrispondente.
- **Linee verticali**: Indicano la suddivisione temporale (battute).

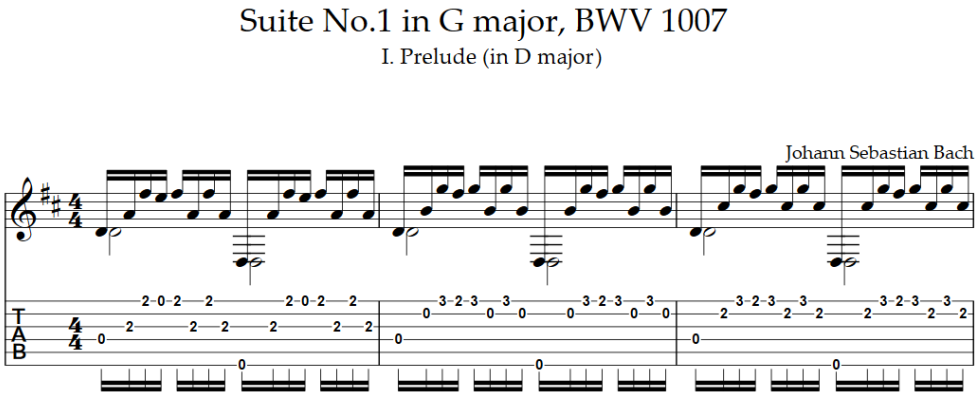

Figura 3: Esempio di confronto tra notazione su pentagramma (sopra) e tablatura (sotto).

2 Algoritmi Genetici

Un **algoritmo genetico** è un particolare tipo di algoritmo evolutivo in cui gli individui/soluzioni sono rappresentati da stringhe (di simboli o più genericamente di bit) e l'evoluzione è simulata con operazioni di ricombinazione (crossover) e mutazione.

- La ricombinazione (**crossover**) è un'operazione di evoluzione che genera un nuovo individuo a partire da 2 individui esistenti.
- La **mutazione**, invece, è un'operazione che genera un nuovo individuo modificando un individuo esistente.

La bontà di una soluzione/individuo viene misurata con una funzione detta di **fitness**; tale funzione fornisce una valutazione numerica.

Il processo evolutivo umano si basa sul trasferimento alla successiva generazione di informazioni relative al funzionamento e alla strutture delle cellule. Tali informazioni sono contenute nel DNA che è formato da segmenti detti **cromosomi**. Ogni cromosoma è costitutito da **geni** che sono le unità funzionali dell'eriditarietà, cioè sono i "mattoncini" che determinano il corredo eriditario di ogni individuo.

In analogia a tale processo evolutivo, nel gergo degli algoritmi genetici, si usa il termine cromosoma per indicare un individuo (o più precisamente la sua rappresentazione) e il termine gene per indicare i "pezzettini" che compongono il cromosoma. [1]

2.1 Caratteristiche dell'Algoritmo Genetico

2.1.1 Individuo

L'individuo (o cromosoma) rappresenta una potenziale soluzione al problema, ovvero una sequenza melodica completa di lunghezza fissa L. Nel codice sviluppato, ogni individuo è codificato come una lista di L geni (con $L = 30$ nel caso di studio).

Ogni gene non rappresenta semplicemente una nota musicale, ma una specifica coordinata sulla tastiera, definita da una coppia di valori interi (s, f). Dove:

- $s \in [0, 5]$ rappresenta l'indice della corda (dove 0 è la corda più grave, Mi basso, e 5 la più acuta).
- $f \in [0, 15]$ rappresenta il tasto premuto, dove 0 indica la corda a vuoto.

Questa rappresentazione diretta "Corda-Tasto" permette all'algoritmo di gestire implicitamente la ridondanza della chitarra, evolvendo non solo le note ma le specifiche diteggiature.

```
def generate_random_gene():
"""Generates a random position (String Index, Fret Number)."""
s = random.randint(0, NUM_STRINGS - 1)
f = random.randint(0, MAX_FRET)
return (s, f)

def generate_individual():
"""Creates a complete random melody (a list of genes)."""
return [generate_random_gene() for _ in range(MELODY_LENGTH)]
```

2.1.2 Fitness Function

La funzione di fitness $F(x)$ è il cuore dell'algoritmo e determina la qualità di un individuo. Il punteggio è calcolato sommando premi (valori positivi) e

penalità (valori negativi) basati su tre criteri principali: armonia, biomeccanica ed euristica melodica.

Sia n_i il valore MIDI della nota generata dal gene i-esimo. La valutazione avviene come segue:

1. Criterio Armonico (Musicalità) L'algoritmo verifica se la nota appartiene alla scala di riferimento (nel codice, Do Maggiore: C, D, E, F, G, A, B).

- Intonazione: Se n_i appartiene alla scala, viene assegnato un premio di $+10$. Altrimenti, viene applicata una penalità severa di -50.
- Ruoli Tonali: Se la nota è la Tonica (Do) o la Dominante (Sol), viene assegnato un bonus extra di $+5$ per rafforzare il centro tonale.
- Risoluzione: Se l'ultima nota della melodia è una Tonica, viene assegnato un forte bonus conclusivo di $+30$.

2. Criterio Biomeccanico (Suonabilità) Per garantire che la melodia sia eseguibile, viene calcolata la "distanza fisica" tra la nota corrente g_i e la precedente g_{i-1}. La penalità è calcolata come:

$$Cost = (2 \cdot \Delta_{fret}) + (5 \cdot \Delta_{string}) \tag{1}$$

Dove Δ_{fret} è la distanza in tasti e Δ_{string} è la distanza tra le corde. È stato assegnato un peso maggiore al salto di corda (5) rispetto al movimento lungo il manico (2), poiché i salti di corda sono tecnicamente più onerosi per la mano destra e sinistra.

3. Euristica Melodica

- Intervalli: Se la distanza tonale tra due note consecutive supera un'ottava (12 semitoni), viene applicata una penalità di -20.
- Ripetitività: Se la stessa nota viene ripetuta consecutivamente ($interval = 0$), viene applicata una leggera penalità di -5 per incentivare il movimento melodico.

2.1.3 Crossover

L'operatore di crossover (o ricombinazione) combina le informazioni genetiche di due genitori per produrre una nuova soluzione. L'algoritmo utilizza un Single Point Crossover.

Viene scelto un punto di taglio casuale k lungo la lunghezza del cromosoma (tra 1 e $L-1$). Il nuovo individuo (figlio) viene creato prendendo i geni di P_1 fino al punto di taglio e concatenandoli con i geni di P_2 dal punto di taglio in poi.

```
def crossover(p1, p2):
"""Single-point crossover: combines two parents to make a child."""
split = random.randint(1, len(p1) - 1)
return p1[:split] + p2[split:]
```

2.1.4 Mutazione

L'operatore di mutazione introduce variabilità genetica per evitare che l'algoritmo converga prematuramente su minimi locali. Nel sistema implementato, la mutazione avviene con una probabilità $P_m = 0.1$ (10%).

Quando la mutazione viene attivata su un individuo viene selezionato casualmente un indice k all'interno della sequenza melodica. Il gene in posizione k viene sostituito con una nuova coppia (s, f) generata casualmente.

```
def mutate(chromosome):
"""Mutation: Randomly changes one note in the melody."""
mutant = list(chromosome)
if random.random() < MUTATION_RATE:
    idx = random.randint(0, len(mutant) - 1)
    mutant[idx] = generate_random_gene()
return mutant
```

3 Risultati Sperimentali

3.1 Configurazione del Test

Per validare l'efficacia dell'algoritmo proposto, è stata eseguita una simulazione con i seguenti parametri:

Parametro	**Valore**
Dimensione della Popolazione	100 individui
Numero di Generazioni	200
Tasso di Mutazione (P_m)	0.1 (10%)
Lunghezza della Melodia	30 note
Tonalità di Riferimento	Do Maggiore (C Major)

Tabella 2: Parametri di configurazione dell'Algoritmo Genetico.

3.2 Analisi della Convergenza

Durante l'esecuzione, è stato monitorato l'andamento del valore di fitness del miglior individuo per ogni generazione. Come mostrato nei log di esecuzione, l'algoritmo parte da soluzioni casuali con punteggi spesso negativi (dovuti alle forti penalità armoniche e biomeccaniche) e converge rapidamente verso soluzioni positive.

```
Evolutionary composition in progress...
Gen 0: Best Score = -781
Gen 50: Best Score = 103
Gen 100: Best Score = 197
Gen 150: Best Score = 220
```

3.3 Output Generato

Di seguito viene riportato un esempio di tablatura generata dall'algoritmo. La rappresentazione è testuale (ASCII Tab), come prodotto dal software implementato.

```
--- GENERATED MELODY (Tablature) ---
e|-------------------------------------------------------------
B|-------------------------------------------------------------
G|-----------------------------------------------5---4---5--
D|-10--9---10-----------------10--7-------7---5--------------
A|-------------10--8---8---10----------7---------------------
E|-------------------------------------------------------------

-------------------------------------------------------------
---------13---------------------------------------------------
-7---12------14--12--7---5---7---5---7---12-----------------
-------------------------------------------------------------
-----------------------------------------15--14--15-----
-------------------------------------------------------15-
```

Inoltre, il codice salva in automatico il file MIDI della tablatura generata con il nome di `generated_melody.mid`. In questo modo, tramite opportuni tool (come MuseScore), è possibile visualizzare la partitura corrispondente ed eventualmente risentirla.

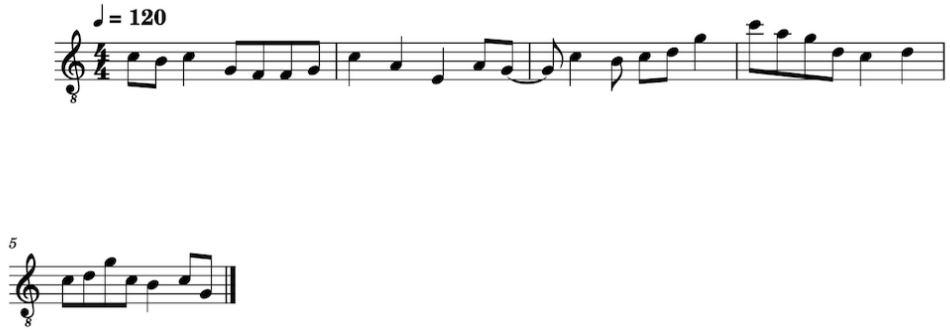

Figura 4: Partitura corrispondente alla tablatura generata

4 Conclusioni

Il lavoro svolto ha dimostrato come gli Algoritmi Genetici possano essere applicati con successo alla generazione di tablature per chitarra, risolvendo efficacemente il problema dell'ambiguità posizionale tipico dello strumento.

Il sistema sviluppato è in grado di:

- Generare linee melodiche armonicamente corrette in una tonalità data.
- Ottimizzare la diteggiatura per garantire la suonabilità fisica.
- Fornire un output immediatamente leggibile dal musicista (tablatura) e ascoltabile.

5 Riferimenti

[1] Roberto De Prisco. *Composizione Musicale Algoritmica*.

[2] D.R. Tuohy e W.D. Potter. "A genetic algorithm for the automatic generation of playable guitare tablature". In: (2005).