UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

PROGETTO DI FONDAMENTI DI INTELLIGENZA ARTIFICIALE

AFAM3

Diteggiatura automatica per fisarmonica in terze minori

Naomi Punella

Matricola: 0512103623

`https://github.com/punella/AFAM3`

Indice

CAPITOLO 1

Definizione del problema

1.1 Motivazioni

La diteggiatura, ovvero la pratica di annotare lo spartito indicando tramite un numero da 1 a 5 il dito con cui conviene suonare ogni nota, è un passaggio fondamentale per ogni musicista classico che approcci lo studio di un pezzo nuovo. Richiede tempo e studio a sua volta, anche perché, seppure già presente sullo spartito, deve rispondere alle esigenze particolari di ogni musicista e viene quindi spesso riadattata. Si tratta di un problema che non può e non deve avere un'unica soluzione: esistono strade battute nel diteggiare certi passaggi, eppure capita che le soluzioni più creative e inusuali si rivelino anche le più comode.

Lo scopo di questo progetto è di trovare, tramite lo sviluppo di un'applicazione desktop con un modulo di intelligenza artificiale basato su un piccolo insieme di criteri ragionevoli, delle soluzioni che nel peggiore dei casi possano comunque fornire al musicista esperto un'alternativa da prendere in considerazione.

1.2 Descrizione del dominio applicativo

Il problema è stato già affrontato in progetti facilmente reperibili in rete, ma è perlopiù applicato a pianoforte e strumenti a corda. Questo progetto si propone di generare una diteggiatura automatica per fisarmonica a bottoni in terze minori.

Mettendo a confronto le caratteristiche di questo strumento con quelle, note ai più, di un pianoforte, si rileva subito una differenza sostanziale nel sistema a bottoni: le distanze tra le note sono molto diverse rispetto ai tasti e non esiste l'handicap delle alterazioni (i bottoni sono tutti uguali, mentre i tasti neri del pianoforte, per via delle dimensioni ridotte, sono notoriamente più scomodi per dita deboli come il quarto e il quinto dito). Inoltre il sistema delle terze minori sconvolge l'organizzazione delle note nella bottoniera, che non sono ordinate in fila ottava per ottava come in una tastiera. L'elemento più particolare è la presenza di due file supplementari [1] composte interamente da doppioni dei tasti delle prime due file. La loro ragion d'essere risiede proprio nel fornire un cammino alternativo che consenta una diteggiatura più agevole.

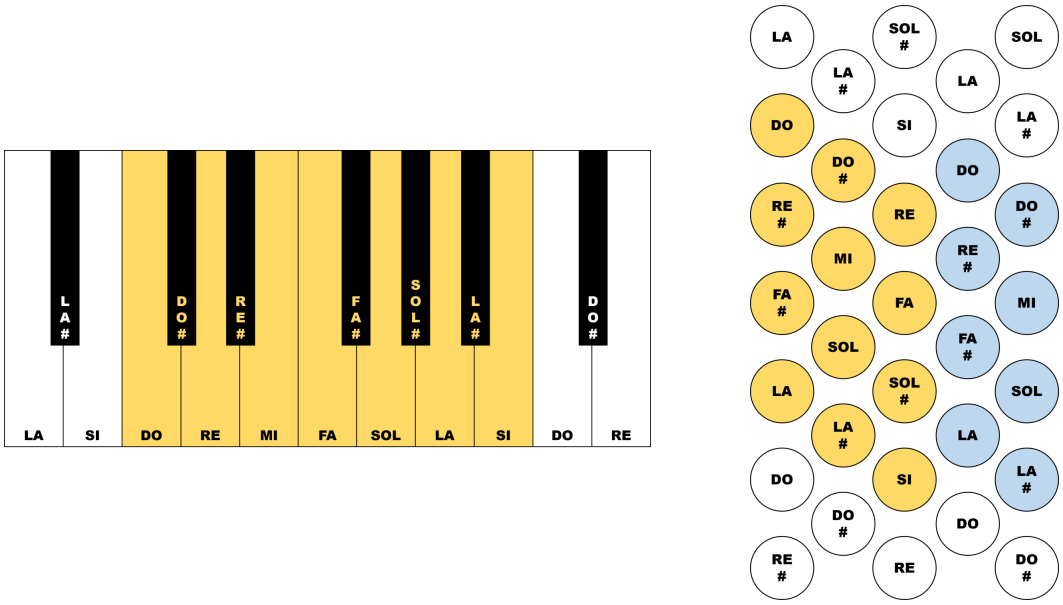

Figura 1.1: Confronto tra tastiera e bottoniera (l'ottava è in arancione, i doppioni dell'ottava evidenziata sono in azzurro).

[1] Le "file" o "raws" (gergo tecnico del dominio applicativo) non vanno confuse con una rappresentazione matriciale della bottoniera. Il termine, infatti, indica ciò che nel dominio della soluzione chiameremmo "colonne".

Tutte queste peculiarità impattano sulla diteggiatura, che per uno stesso pezzo sarà necessariamente diversa a seconda dello strumento. Lo stesso vale per altri tipi di fisarmonica, come quelle a tastiera o per quinte.

Una volta individuato il dito con cui è più opportuno suonare la nota, è comune a tutti gli strumentisti indicare per entrambe le mani il pollice con il numero 1, l'indice con il 2, e così via fino al 5 del mignolo. La fisarmonica in terze minori si differenzia solo per la notazione speciale riservata alle due file supplementari: in questi casi, nella letteratura originale per lo strumento il numero viene normalmente cerchiato.

1.3 Descrizione dell'ambiente

Poiché la risoluzione del problema sarà affidata a un agente intelligente, occorre definire le caratteristiche dell'ambiente che questo percepisce e sul quale agisce.

1.3.1 Specifica PEAS

Performance. La misura delle prestazioni è data dalla confortevolezza della diteggiatura generata, dall'ammissibilità delle posizioni della mano e dalla minimizzazione della dispersività dei movimenti.

Environment. L'ambiente in cui opera l'agente intelligente è lo spartito musicale in formato MusicXML, con tutte le disposizioni possibili di note e diteggiatura.

Actuators. L'ambiente è manipolato tramite l'annotazione della diteggiatura sul file dello spartito.

Sensors. L'ambiente è percepito tramite il parsing del contenuto del file.

1.3.2 Proprietà dell'ambiente

L'ambiente operativo è

- **completamente osservabile**: è sempre possibile conoscere lo stato completo dell'istanza del problema tramite il contenuto del file MusicXML;

- **deterministico**: a partire dallo stato corrente, ogni nota diteggiata determina lo stato successivo;
- **sequenziale**: per ogni nota, la scelta della diteggiatura dipende da quella delle note precedenti;
- **statico**: l'istanza del problema non cambia mentre l'agente sceglie la diteggiatura;
- **discreto**: le azioni possibili dell'agente sono pari al numero di dita, mentre le percezioni si limitano all'estensione dello strumento (il numero di percezioni distinte è ridotto dall'esclusione della polifonia, per cui si veda la Sezione 1.4);
- **a singolo agente**.

1.4 Semplificazioni

Data la complessità di qualsiasi problema legato alla teoria musicale, la soluzione verrà progettata con una serie di limitazioni da intendersi come sviluppi futuri.

Come deducibile dalla Sezione 1.2, nella quale non si è fatto accenno alle diverse e complesse caratteristiche del manuale sinistro (per il quale bisognerebbe progettare un modello a parte), verrà presa in considerazione solo la mano destra. Per non complicare l'implementazione del parsing dello spartito e non accrescere il numero di vincoli del problema, si è deciso inoltre di escludere la polifonia.

Altre semplificazioni snelliscono esclusivamente l'implementazione della lettura dello spartito, senza toccare la modellazione del problema: verranno diteggiate solo le note in chiave di violino, le legature di valore verranno ignorate e non saranno processati spartiti che presentano alterazioni in chiave, tutte limitazioni a cui si rimedia facilmente ma con molto codice.

CAPITOLO 2

Descrizione della soluzione

2.1 Scelta dell'approccio

Un possibile soluzione basata sull'apprendimento automatico è stata scartata per mancanza di dati. Nonostante stia crescendo in popolarità, la fisarmonica è ancora uno strumento di nicchia nel mondo della musica classica e, per le caratteristiche esposte nella Sezione 1.2, non è possibile ricorrere ai dataset disponibili di diteggiatura per pianoforte.

Allora sono stati presi in esame gli algoritmi di ricerca e, considerata la natura del problema, la scelta più ovvia è stata quella di impiegare un algoritmo genetico. Si tratta infatti di un problema di ottimizzazione, in cui il numero di soluzioni possibili è molto grande e non è necessario esplorarle tutte, ma ci si può accontentare di un risultato sub-ottimo. Inoltre la diteggiatura è un problema che richiede continui compromessi tra obiettivi diversi: alla sua formulazione multi-obiettivo vengono incontro alcune implementazioni note di algoritmi genetici messe a disposizione dal framework jMetal.

2.2 Progettazione della soluzione

La soluzione sarà costituita da quattro moduli: uno per l'interfaccia che consentirà all'utente di scegliere il file dello spartito da diteggiare, uno per la logica che coordinerà i vari moduli e gestirà la lettura e la scrittura sullo spartito, uno per la modellazione del dominio applicativo che rappresenterà i bottoni della fisarmonica, uno per l'intelligenza artificiale che produrrà i risultati dell'algoritmo genetico.

Per la progettazione di quest'ultimo, sono stati fissati alcuni parametri chiave, mentre altri verranno scelti o cambiati in fase di implementazione o con i successivi test empirici.

Di seguito sono riportati i parametri scelti in fase di progettazione.

Codifica degli individui. Una soluzione sarà rappresentata da un array di interi di lunghezza pari al numero di note estratte dallo spartito. Gli interi possono assumere valori compresi tra 1 e 10: i valori tra 1 e 5 corrispondono a quelli di una normale diteggiatura, mentre i valori tra 6 e 10 rappresentano la variante "cerchiata" dei valori tra 1 e 5. Il raddoppiamento dei possibili valori generati introduce un vincolo.

Vincoli. In corrispondenza delle note che si trovano sulla terza fila, sprovviste di doppioni, i valori della soluzione devono essere compresi tra 1 e 5. Si ricorda, infatti, che la diteggiatura "cerchiata" indica il bottone doppione sulla quarta o quinta fila.

Funzione di fitness. Tra i molti obiettivi possibili, ne sono stati scelti quattro giudicati fondamentali, tutti da minimizzare.

- **Numero di ripetizioni**: è buona norma, in una diteggiatura, non assegnare mai lo stesso dito a due note consecutive, persino se si tratta dello stesso tasto o bottone (i cosiddetti *ribattuti*).
- **Numero di posizioni scomode**: verranno conteggiate come scomode le posizioni in cui le dita si incrociano, con l'eccezione del pollice, che fa volentieri da perno.
- **Numero di spostamenti della mano**: ogni volta che la distanza tra due bottoni eccede quella massima tra le dita, la mano deve necessariamente spostarsi

per raggiungere la nota, interrompendo la fluidità dei movimenti e quindi impattando sulla confortevolezza della diteggiatura.

- **Distanza totale tra i bottoni**: avendo a disposizione le due file supplementari, c'è la possibilità di scegliere un cammino più breve che disperda meno i movimenti.

Sebbene nessuno di questi obiettivi rappresenti una regola infrangibile della diteggiatura, l'ultimo è sicuramente meno stringente degli altri, perciò saranno preferite le soluzioni che a scapito della distanza totale minimizzino gli altri obiettivi, secondo un ordine stabilito successivamente.

2.3 Implementazione

2.3.1 Tecnologie utilizzate

L'applicazione è scritta interamente in linguaggio Java. L'interfaccia utente utilizza il framework Java Swing con il look and feel FlatLightLaf. Il modulo di intelligenza artificiale utilizza il framework jMetal. Le dipendenze e la build sono gestite da Maven.

2.3.2 GUI

L'interfaccia consiste in una finestra che chiede all'utente di scegliere lo spartito da diteggiare tra i file del suo PC, dopodiché il file può essere processato. All'avvio di questa operazione, un'altra finestra chiede di attenderne il completamento e al termine informa l'utente, dandogli la scelta di caricare un altro spartito o uscire dall'applicazione.

2.3.3 Logica

L'input dell'utente deve essere elaborato prima di trasformarsi nell'input dell'agente intelligente.

La classe che si occupa del parsing del file MusicXML applica i controlli su tutte le semplificazioni di cui si è discusso nella Sezione 1.4.

Dallo spartito viene estratto un numero minimale di informazioni per il funzionamento dell'applicazione: il tono e l'ottava delle note. Queste informazioni, che nella notazione musicale anglosassone vengono rappresentate da un carattere e un intero (p.e. "C4" per indicare il do centrale), vengono sintetizzate in un unico valore intero al fine di identificare più efficientemente la posizione della nota sulla bottoniera. Il mapping del valore è stato pensato in modo che le note sulla terza fila, sprovvista di doppioni, siano divisibili per 3. I valori seguono la scala cromatica, perciò per le alterazioni bastano semplici incrementi e decrementi di 1. Le ottave sono ricalcolate sull'estensione reale dello strumento e possono essere dedotte, a partire dal valore di sintesi, tramite una semplice divisione.

La scelta implementativa più importante è stata quella di gestire lo spartito come un insieme di frasi musicali. Dal momento che in genere le pause danno al musicista la possibilità di riposizionare la mano, per ogni pausa lo spartito viene spezzato e suddiviso in input più piccoli. In questo modo, ogni spartito potrebbe richiedere più esecuzioni dell'algoritmo genetico, che in compenso dovrà gestire soluzioni più corte.

Questa classe perciò restituisce una lista di liste di interi che verrà passata al runner dell'algoritmo genetico. Al termine della computazione, la classe si occupa anche di scrivere la diteggiatura sul file dello spartito creando l'apposito elemento previsto dal formato MusicXML. Il formato non ha però uno standard per la notazione speciale della diteggiatura cerchiata: per semplicità in questi casi si è deciso di scrivere la diteggiatura tra parentesi tonde.

2.3.4 Modellazione

Piuttosto che rappresentare intuitivamente la bottoniera come una matrice, si è deciso di migliorare la complessità spaziale e temporale calcolando dinamicamente la posizione dei bottoni: la fila (col) si può ricavare da un controllo sul gene della soluzione e un'operazione di modulo sulla nota, mentre l'altezza (row) si può ricavare dall'ottava. Conservando questi due parametri per ogni bottone che la soluzione va a toccare, possiamo calcolarne la distanza in centimetri rispetto a ogni altro bottone sulla bottoniera, che quindi non necessita di una sua rappresentazione. Il calcolo

della distanza tra due bottoni segue delle regole studiate ad hoc sullo strumento reale.

Questo modulo è essenziale per il calcolo della fitness, che ha bisogno di conoscere la posizione dei bottoni coinvolti nella soluzione.

2.3.5 AI

Il modulo di intelligenza artificiale consiste in due classi: una per la rappresentazione del problema, che definisce la codifica degli individui e ne valuta la fitness, e una per il runner dell'algoritmo genetico, che imposta tutti gli altri parametri e restituisce la soluzione migliore tra i risultati dell'esecuzione tramite preference sorting.

Sebbene la prima classe fosse stata in gran parte delineata in fase di progettazione, durante l'implementazione sono stati fatti degli aggiustamenti. Di seguito si riportano perciò tutti i parametri dell'algoritmo genetico al termine della prima implementazione.

Inizializzazione

Codifica degli individui. La codifica è rimasta invariata rispetto a quanto progettato nella Sezione 2.2.

Procedura di inizializzazione. Assecondando l'implementazione proposta da jMetal, l'inizializzazione avviene in modo casuale. Per ogni gene viene stabilito in partenza il range di valori che questo potrà assumere, perciò attraverso un controllo sulla nota i-esima dello spartito ci si assicura che, in caso questa si trovi sulla terza fila, il range di valori del gene i-esimo sia ridotto. Poiché gli algoritmi di mutazione fanno riferimento agli stessi bound, così facendo il vincolo risulta sempre rispettato.

Size popolazione. La dimensione della popolazione è fissata a 100. Se necessario, verranno testate dimensioni diverse più avanti.

Size mating pool. Se non specificato diversamente, jMetal imposta come dimensione del mating pool la stessa della popolazione.

Operatori genetici

Selezione. Tra le implementazioni fornite da jMetal per la selezione, ne verrà testato un sottoinsieme: Binary Tournament Selection, al momento scelta per la prima implementazione, e Ranking And Crowding Selection.

Crossover. Tra le implementazioni fornite da jMetal per il crossover, viene utilizzata quella dell'N-Point Crossover. Per la prima implementazione, il numero di punti è stato impostato a 1 (si tratta perciò di un Single Point Crossover). Più avanti verrà testata con un numero di punti diverso.

Probabilità di crossover. La probabilità di crossover è per ora fissata a 0,8.

Mutazione. In jMetal è implementato un solo tipo di mutazione per le soluzioni intere: si tratta dell'Integer Polynomial Mutation, utilizzata al momento dall'algoritmo. È stato quindi implementato ad hoc un semplice algoritmo di Random Resetting per permettere un confronto in fase di analisi dei risultati.

Probabilità di mutazione. La probabilità di mutazione è al momento fissata a 0,01.

Valutazione

Funzione di fitness. Rispetto a quanto progettato nella Sezione 2.2, gli obiettivi del problema sono stati ridotti in numero, accorpandone due in un obiettivo solo: ci si è accorti che la presenza di ripetizioni in una soluzione veniva doppiamente penalizzata, dal momento che una ripetizione implica uno spostamento della mano (la distanza tra due bottoni non sarà mai minore di quella tra un dito e se stesso). D'altra parte, una volta che si è reso indispensabile uno spostamento della mano, non importa che si impieghi o meno lo stesso dito. I due obiettivi erano perciò dipendenti e si è deciso di gestirli insieme, aggiungendo un controllo sul caso particolare dei ribattuti (per sfruttare lo stesso codice, verranno conteggiati come spostamenti della mano anche se è forse concettualmente improprio). Il preference sorting ordinerà le soluzioni preferendo prima quelle con il minor numero di posizioni scomode e poi, a parità di risultati, quelle con il minor numero di spostamenti della mano.

Stopping condition. Il framework jMetal permette di specificare un numero massimo di valutazioni come budget di ricerca. Al momento il numero è stato impostato a 10.000.

Metaeuristica

Il framework jMetal presenta una selezione molto ampia di algoritmi per i problemi multi-obiettivo. Si è deciso di limitare la scelta agli algoritmi di tipo NSGA, perciò verranno esplorate le differenze nelle prestazioni di NSGA-II e NSGA-III, partendo dal primo.

2.4 Analisi dei risultati ottenuti

Per via dell'inaccessibilità a un dataset sufficientemente ampio di file MusicXML che rispettino i requisiti richiesti, i test sono stati effettuati su 7 spartiti con caratteristiche abbastanza diverse. Il numero di frasi, e quindi di esecuzioni dell'algoritmo, per gli spartiti testati varia da 1 a 42.

I risultati sono stati valutati a partire dalla metaeuristica. Visto il comportamento molto buono di entrambi gli algoritmi, la sperimentazione empirica degli altri parametri è stata portata avanti su tutti e due fino alla scelta finale.

Confronto tra metaeuristiche

L'algoritmo NSGA-II, fin con la prima configurazione di parametri, dà risultati ottimi per i primi due obiettivi sulla maggior parte degli input testati. Gli unici casi in cui non riesce a raggiungere l'ottimo sono quelli di frasi musicali particolarmente lunghe (individui con più di 100 geni).

Nell'esempio è riportato l'output di un problema con due soluzioni di lunghezza superiore a 100.

```
========== INPUT ==========
Executions: 22 Shortest phrase size: 1 Longest phrase size: 127
=== ALGORITHM PARAMETERS ===
Algorithm: NSGAII
```

```
Population size: 100 Stopping condition: 10000 evaluations
Selection: BinaryTournamentSelection
Crossover: NPointCrossover con N=1 e con prob. 0.8
Mutation: IntegerPolynomialMutation con prob. 0.01
========= RESULTS =========
Total uncomfortable positions: 0.0
Total hand shiftings/repetitions: 6.0
Total distance: 1683.75
Total computing time: 5326
```

L'algoritmo NSGA-III, con la stessa configurazione di parametri, su tutti gli input testati restituisce sempre un risultato ottimo per i primi due obiettivi, nonché risultati generalmente migliori per il terzo.

Quando lavorano su individui con un numero contenuto di geni, i due algoritmi forniscono esattamente le stesse prestazioni, con la differenza che i tempi di computazione di NSGA-III sono notevolmente più lunghi. Empiricamente si può dire che questo impieghi il doppio del tempo rispetto a NSGA-II. Perciò proveremo a modificare alcuni parametri affinché migliorino le prestazioni del primo algoritmo o il tempo di computazione del secondo.

Tutti i test successivi verranno effettuati sullo spartito più discriminante, costituito da 22 frasi di lunghezza variabile (da 1 a 127). Perciò per ogni test si fa riferimento ai risultati di 22 esecuzioni.

Confronto tra operatori genetici

Nonostante i buoni risultati ottenuti con la prima configurazione, sono stati testati altri operatori genetici per avere un termine di paragone.

Ranking And Crowding Selection e Random Resetting Mutation hanno dato risultati molto deludenti con entrambe le opzioni di metaeuristica. Con questi algoritmi i tempi di computazione si riducono molto, il che fa presumere che la qualità della soluzioni sia dovuta a una convergenza prematura. Sono stati confermati, perciò, la Binary Tournament Selection e la Integer Polynomial Mutation. È stato inoltre osservato che, rispetto al valore iniziale di 0,01, aumentando anche di poco la probabilità di mutazione aumenta il tempo di computazione e peggiorano i risultati.

Variando il numero di punti di crossover, si sono osservate leggere differenze nelle prestazioni. In particolare, con un numero di punti di crossover uguale alla dimensione della soluzione entrambe le metaeuristiche hanno registrato soluzioni migliori rispetto al terzo obiettivo. Considerando che però l'Uniform Crossover peggiora i tempi di computazione, è stato stabilito che lo scarto nei risultati non è sufficientemente rilevante per giustificare la scelta di quest'algoritmo. Il Two-Point Crossover è leggermente più lento del Single Point e non presenta differenze rilevanti. Dal momento che le istanze del problema possono avere un numero di geni pari a 1, la scelta è ricaduta sul Single Point così da poter utilizzare lo stesso algoritmo per ogni istanza, senza ulteriori controlli sulla lunghezza della soluzione.

Dimensione della popolazione e stopping condition

Raddoppiando il numero massimo di valutazioni, anche NSGA-II raggiunge l'ottimo per i primi due obiettivi. I tempi di computazione lievitano, rimanendo però al di sotto di quelli di NSGA-III.

Di contro, modificando i parametri fino a dimezzare il numero della popolazione e il numero massimo di valutazioni, NSGA-III si comporta sempre allo stesso modo, inclusi i tempi di esecuzione.

È stato infine scelto NSGA-II con un numero massimo di valutazioni fissato a 17000. Questo rappresenta un buon compromesso: restituisce sempre o un risultato ottimo per i primi due obiettivi o un risultato sub-ottimo per il secondo obiettivo ma migliore rispetto al terzo, mentre i tempi di esecuzione rimangono più vicini a quelli della sua configurazione iniziale che a quelli di NSGA-III. Sebbene, conoscendo le applicazioni del progetto, il tempo di computazione sia sacrificabile in favore di soluzioni migliori, è probabile che gli input siano molto più grandi di quelli qui testati e che quindi il tempo diventi un fattore importante, perciò è sembrato opportuno dargli la giusta considerazione.

CAPITOLO 3

Conclusioni

Le soluzioni generate, benché ammissibili, sono piuttosto anticonvenzionali. Considerando che l'applicazione si rivolge a musicisti esperti, potrebbe essere un'occasione per riflettere sui motivi per cui spesso si sceglie una soluzione a scapito di una matematicamente migliore.

Immaginando invece gli sviluppi futuri, la sola introduzione della dimensione del tempo, qui esclusa per semplicità, renderebbe il problema e le sue soluzioni infinitamente più interessanti e complessi, e le scelte di algoritmo e parametri a quel punto sarebbero cruciali.

Ma al di là delle limitazioni di cui si è discusso, il progetto potrebbe essere notevolmente raffinato rimodellando e aggiungendo obiettivi di fitness e criteri di preferenza sulla base di una conoscenza più approfondita delle buone pratiche maturate da diversi musicisti.