UNIVERSITÀ DI BOLOGNA

Scuola di Scienze

Corso di Laurea in Matematica

Analisi topologica per la classificazione musicale

Tesi di Laurea Magistrale
in Topologia Algebrica

Relatore
Prof. Massimo Ferri

Candidato
Matteo Pennesi

Correlatore
Prof. Moreno Andreatta

Anno Accademico 2017/2018

Indice

Indice **2**

1 Concetti introduttivi **7**

1.1 Alcune nozioni di teoria musicale 7

1.1.1 Intervalli musicali 7

1.1.2 Lo spazio delle classi di altezza 8

1.1.3 Teoria neo-Riemanniana 10

1.2 Modelli geometrici 12

1.2.1 Simplessi e complessi simpliciali 12

1.2.2 Omologia simpliciale 14

1.3 Visualizzazione e analisi delle strutture musicali 17

1.3.1 Tonnetz 17

1.3.2 Rappresentazione simpliciale di un insieme di accordi 19

2 Analisi topologica **22**

2.1 Omologia persistente 22

2.1.1 Distanza bottleneck 25

2.2 Analisi sul tonnetz 27

2.2.1 Il Tonnetz deformato 27

3 Risultati **31**

3.1 Caratteristiche dei diagrammi di persistenza 31

3.1.1 Diagrammi di 0-persistenza 32

3.1.2 Diagrammi di 1-persistenza 41

3.2 Clustering gerarchico 49

4 Persistenza su serie temporali **52**

4.1 Algoritmo DTW 52

4.1.1 Qualche risultato 56

Appendices **63**

A Codice Python **64**

Bibliografia **79**

Introduzione

Il rapporto tra musica e matematica è stato oggetto di interesse sin dai tempi dei Pitagorici. Uno degli sviluppi fatti negli ultimi anni in questo campo interdisciplinare riguarda la realizzazione di modelli formali per l'analisi e la classificazione musicale. La creazione di un modello di questo tipo risulta sin da subito complicata in quanto non è così semplice definire un'analisi oggettiva per un'estetica musicale.
In questo elaborato investigheremo la connessione tra una composizione musicale e la forma di uno spazio associato a questa attraverso metodi topologici.

La tesi è strutturata come segue: il primo capitolo introduce i concetti necessari per comprendere l'analisi musicale in termini scientifici attraverso il linguaggio algebrico. Vengono inoltre presentate alcune nozioni fondamentali relative ai complessi simpliciali e all'omologia simpliciale. Nella seconda parte del capitolo viene introdotto il *Tonnetz*, un diagramma usato per la rappresentazione dello spazio tonale e per la visualizzazione delle principali relazioni armoniche della tradizione occidentale. Nell'ultima parte del capitolo viene data una descrizione formale per la rappresentazione simpliciale di un insieme di accordi, cioè la creazione di un complesso simpliciale associato ad una composizione musicale. Il capitolo si conclude con la definizione di una generalizzazione del Tonnetz a tutte le combinazioni di classi d'altezza.
Il secondo capitolo si concentra sull'analisi topologica e, in particolare, sulla persistenza. Vengono richiamate le nozioni di omologia persistente e di distanza bottleneck. La seconda parte del capitolo presenta l'analisi topologica sul Tonnetz deformato tramite persistenza come presentata da Bergomi [1].
Il terzo capitolo mostra qualche risultato ottenuto dall'analisi dei diagrammi di 0 e $1-$persistenza per gli spazi associati ad alcune composizioni presenti nel corpus di *Music21*. Vengono evidenziati dei descrittori di forma comparabili a descrittori di

stile compositivo e, nell'ultima parte del capitolo, vengono messi in evidenza i limiti e le potenzialità del metodo utilizzato.
Il quarto capitolo presenta l'analisi della persistenza su serie temporali e qualche risultato utile all'analisi musicologica computazionale.
In appendice è riportato il codice scritto in Python.

Contesto

Il lavoro presentato in questa tesi è stato iniziato durante il tirocinio formativo all'interno del progetto SMIR (Structural Music Information Research)[1] presso l'IRMA (Institut de Recherche Mathématique Avancée) dell'Università di Strasburgo.
Questo gruppo di ricerca formato da matematici, informatici e musicologi lavora seguendo un approccio strutturale multidisciplinare alla musicologia computazionale che si differenzia dai metodi statistici e di trattamento di segnale utilizzati attualmente all'interno dello scenario di ricerca del MIR (Music Information Retrieval).
Nel mio caso, ho focalizzato gli studi partendo dai lavori di Bergomi [1] e Bigo [2], in particolare ho provato a studiare un approccio all'omologia persistente di uno spazio di accordi generalizzato a un numero qualsiasi di classi di altezza.

[1]Il progetto, coordinato da Moreno Andreatta, è consultabile all'indirizzo: `epmus.ircam.fr/moreno/smir`

Capitolo 1

Concetti introduttivi

1.1 Alcune nozioni di teoria musicale

La teoria musicale occidentale pone le sue basi su due pilastri apparentemente indipendenti: l'armonia ed il contrappunto. Per *armonia* si intende la combinazione simultanea di più suoni, e quindi la formazione, le relazioni e il concatenamento degli accordi. Il *contrappunto* (o *voice-leading*) è la tecnica di combinazione di più linee melodiche generate dalla connessione delle singole note in una serie di accordi.
Possiamo pensare all'armonia come l'elemento compositivo che si occupa dell'aspetto verticale della musica mentre il contrappunto si sviluppa orizzontalmente.

1.1.1 Intervalli musicali

Il suono è un fenomeno fisico generato dalle vibrazioni in uno spazio di propagazione. Oltre ad avere caratteristiche puramente fisiche, il suono ha anche caratteristiche percettive. Gli attributi fisici come la frequenza e l'ampiezza sono diversi rispetto a quelli percettivi come, ad esempio, la consonanza.
La maggior parte delle persone non è in grado di distinguere le frequenze assolute ma è capace di riconoscere i rapporti tra queste frequenze. Questo ci suggerisce che la nozione di *distanza* nello spazio delle frequenze sia definita in termini di rapporti tra frequenze fondamentali.
Esiste un modello, chiamato *legge di Fechner*, in cui la altezze percepite dall'orecchio umano sono proporzionali al logaritmo della loro frequenza. Il rapporto delle

frequenze è misurato in *cents* e l'intervallo tra due frequenze f_1 e f_0 è dato da

$$\frac{1200}{ln(2)} ln(f_1/f_0)$$

L'ottava è l'intervallo tra due frequenze che hanno il rapporto uguale a 2 quindi un'ottava equivale a 1200 cents. L' ottava è percepita come l'intervallo più consonante tra due frequenze diverse ed è l'intervallo da cui vengono generati i diversi tipi di temperamenti musicali. Il temperamento è il sistema di intonazione proposto per ridurre le differenze che risaltano tra gli intervalli principali. Gli intervalli diversi da quelli giusti sono detti "temperati". La formalizzazione algebrica delle strutture musicali considera il temperamento equabile, ottenuto dividendo l'ottava in dodici parti uguali.

1.1.2 Lo spazio delle classi di altezza

Il ripetersi del nome delle note suggerisce che il concetto di ottava stabilisca una relazione privilegiata tra le altezze che sta alla base della nostra cultura musicale. Questa relazione ci permette di classificare le altezze e gli intervalli. In termini matematici andiamo a quozientare l'insieme $\mathbb{R}$ di tutte le altezze. Infatti, date due altezze a e b in $\mathbb{R}$, è nota la relazione di equivalenza:

$$a \sim b \Leftrightarrow a - b \in \mathbb{Z}$$

Otteniamo quindi una nota come un punto in $\mathbb{R}/\mathbb{Z} \simeq S^1$.
La maggior parte degli strumenti musicali, come ad esempio il pianoforte, può produrre un numero finito di altezze. Questo è uno dei motivi che ci porta a discretizzare lo spettro dell'udibile in un numero finito di note $N \subset S^1$ dove la cardinalità del sottoinsieme N dipende dal temperamento scelto. Il temperamento che utilizzeremo in questo elaborato è quello equabile ed è dato dal sottoinsieme delle dodici note della scala cromatica $(C, C\sharp, D, D\sharp, E, F, F\sharp, G, G\sharp, A, A\sharp, B)$ che corrispondono a dodici punti equidistanti di S^1. Questa discretizzazione induce un isomorfismo tra N e $\mathbb{Z}/12\mathbb{Z}$. Chiameremo questo spazio *Pitch-class space o Spazio delle classi d'altezza*. Questa costruzione si può generalizzare a temperamenti equabili di n note suddividendo l'ottava in n parti uguali.

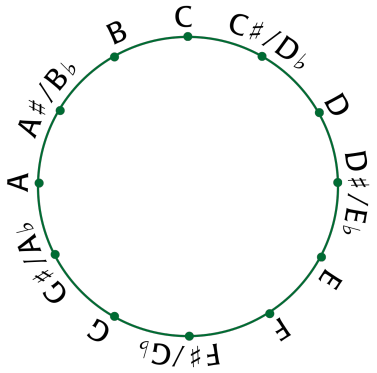


Figura 1.1: Spazio delle classi d'altezza

Notazione italiana	*do*	*do*♯	*re*	*re*♯	*mi*	*fa*	*fa*♯	*sol*	*sol*♯	*la*	*la*♯	*si*
Notazione inglese	C	$C\sharp$	D	$D\sharp$	E	F	$F\sharp$	G	$G\sharp$	A	$A\sharp$	B
$\mathbb{Z}_{12}$	0	1	2	3	4	5	6	7	8	9	10	11

Tabella 1.1: Tabella con le classi d'altezza. Le prime due corispondono alla notazione italiana (solfeggio) e quella inglese. La terza riga corrisponde ai valori di $\mathbb{Z}_{12}$ associati alle classi di altezza dall'isomorfismo.

Allo stesso modo possiamo raggruppare gli intervalli tra altezze per classi d'equivalenza e associarli a dei valori numerici come rappresentato in tabella 1.2.

P1	m2	M2	m3	M3	P4	TT	P5	m6	M6	m7	M7
0	1	2	3	4	5	6	7	8	9	10	11

Tabella 1.2: Tabella con le dodici classi intervallari della scala cromatica che vanno dall'unisono ($P1$) alla settima maggiore ($M7$). La seconda riga indica la distanza in semitoni associata a ciascuna classe intervallare.

L'insieme delle classi di intervalli munito dell'operazione somma $mod12$ forma un gruppo in senso algebrico. Questo gruppo è isomorfo a $(\mathbb{Z}_{12}, +_{mod12})$. Ad esempio l'intervallo di quinta $P5$ si può ottenere componendo una terza maggiore e una terza minore, cioè: $m3 + M3 = P5$. La trasposizione musicale, che consiste nello spostare un'altezza di un certo intervallo, si può quindi vedere come un'azione del gruppo degli intervalli sull'insieme delle classi d'altezza.

La *scala* è una serie di classi di altezza ordinate. Quest'ordine definisce una misura di distanza musicale diversa rispetto a quella generale data dai semitoni della scala cromatica. La *scala diatonica* è una scala eptatonica che sta alla base della tradizione musicale occidentale formata da cinque intervalli di un tono e due di un semitono. I *modi* della scala diatonica vengono classificati in base alla posizione degli intervalli di un semitono all'interno della scala.
Le sette classi d'altezza di una scala diatonica si possono ottenere tramite una successione di quinte. Ad esempio, per ottenere le sette classi d'altezza che formano la scala diatonica di Do, possiamo calcolare le quinte a partire dal Fa:

Fa - Do - Sol - Re - La - Mi - Si

Inoltre, è possibile formalizzare gli accordi come insiemi di classi di altezze. Ad esempio l'accordo di Do maggiore formato dalle note Do, Mi, Sol corrisponde alla collezione $\{0, 4, 7\}$.

1.1.3 Teoria neo-Riemanniana

La teoria neo-Riemanniana è un ramo dell'analisi musicale nato dalle idee presentate da musicologi dell'ultimo secolo tra cui David Lewin[3], Richard Cohn[14] e altri. Ciò che lega queste idee è lo studio che mette in stretta relazione l'armonia con il contrappunto. Lo studio si basa principalmente sul gruppo PLR, il cui nome è determinato dalle iniziali dei nomi inglesi dei suoi tre suoi elementi fondamentali P (Parallel), L (Leading Tone Exchange), R (Relative). Questo gruppo agisce sullo spazio delle triadi generato dalla combinazione delle azioni P,L e R così definite:

P Trasforma una triade nella sua parallela. Converte una triade maggiore in una minore (Do maggiore → Do minore) e viceversa spostando la terza.

R Trasforma una triade nel suo relativo. Converte una triade maggiore spostando la quinta (Do maggiore → La minore) o una triade minore spostando la tonica (La minore → Do maggiore)

L Trasforma una triade per il *Leading-Tone Exchange*. Converte una triade maggiore spostando la tonica (Do maggiore → Mi minore) o una triade minore spostando la quinta (Mi minore → Do maggiore)

Cohn [4] descrive così alcune caratteristiche di queste trasformazioni:
"The three transformations share several properties: each is an involution; each is mode-altering (it maps a triad to its pitch-class inversion); and each transformation preserves two common tones, replacing the third with a pitch-class a semitone away (in the case of P and L) or a whole tone away (in the case of R)."[1]
Sempre secondo Cohn, la teoria si mostra particolarmente efficiente per l'analisi delle strutture armoniche del tardo romanticismo, caratterizzate da un alto livello di cromatismi e voice-leading parsimoniosi.
La connessione tra operatori neo-Riemanniani e voice-leading è stata anche oggetto di critiche, ad esempio Tymoczko [5] afferma che questo tipo di approccio sia solamente un'approssimazione di un'analisi molto più complessa.
É stato osservato che nel corso del diciannovesimo secolo le scale cromatiche hanno gradualmente soppiantato quelle diatoniche [12]. Nel primo periodo, ciò che non apparteneva alla scala diatonica veniva tradotto come passaggio ad un'altra scala o come "sporcatura" della struttura diatonica principale.
A partire dall'inizio del ventesimo secolo, la scala diatonica ha cominciato ad essere vista come una particolare configurazione di sette note prese dalla collezione cromatica fondamentale. Non più dipendenti dalla scala diatonica per la loro funzione, le note della scala cromatica hanno cominciato a diventare entità a sé stanti. In breve, i compositori si approcciarono a questo nuovo contesto cromatico in due modi: il primo, associato a compositori come Wagner, Strauss o il primo Schoenberg, intraprese un processo di "de-enfatizzazione" delle scale a favore di quella cromatica. Le progressioni di accordi non furono più limitate alla regione diatonica ma si svilupparono all'interno dello spazio cromatico. Anche l'attività melodica diventò sempre più cromatica e meno riconducibile ad una particolare configurazione diatonica. Questo approccio puntò ad un appiattimento dello spazio musicale dato dalla forte influenza cromatica sulle scelte armoniche e melodiche.

[1] "Le tre trasformazioni hanno diverse proprietà in comune: sono involuzioni; alterano i modi; e mantengono due altezze in comune, sostituendo la terza altezza con un'altra ad un semitono di distanza (P o L) o ad un tono di distanza (R)."

Il secondo approccio, riconducibile a compositori come Rimsky-Korsakov, Debussy e Ravel, preservò una relazione più convenzionale tra accordi e scale ma espandendo in modo significativo il proprio vocabolario musicale. Le nuove scale diedero accesso a nuovi accordi, mentre nuovi accordi suggerirono nuove scale. Questo approccio propose una concezione dello spazio musicale meno radicale rispetto ai primi ma più strutturato gerarchicamente, con l'importante ruolo di mediazione tra la tradizione musicale e il totale cromatismo.
Questo fatto ci suggerisce che negli ultimi due secoli i rapporti tra le progressioni armoniche standard nella scala diatonica e quelle generate all'interno della scala cromatica abbiano assunto un valore diverso rispetto a quello della tradizione musicale occidentale, arrivando ad essere, come nel caso della musica atonale, equivalenti da un punto di vista di stile compositivo.

1.2 Modelli geometrici

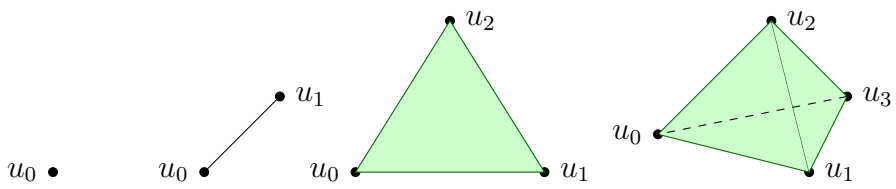
1.2.1 Simplessi e complessi simpliciali

Sia $V = \{v_0, v_1, ..., v_n\}$ una collezione di punti in $\mathbb{R}^n$. Una *combinazione affine* di V è la somma

$$\sum_{i=0}^{n} \lambda_i v_i$$

con $\sum_{i=0}^{n} \lambda_i = 1$.
Una *combinazione convessa* di V è una combinazione affine tale che $\lambda_i \geq 0 \ \forall i$.
L' *inviluppo convesso* di V è la collezione di tutte le combinazioni convesse dei punti in V.

Definizione 1.1. Un *k-simplesso geometrico* è un inviluppo convesso di $k+1$ punti affinemente indipendenti.



I 0,1, 2 e 3-simplessi sono chiamati rispettivamente *vertici, spigoli, triangoli e tetraedri.*

Definizione 1.2. Sia σ un simplesso generato da una collezione di punti affinemente indipendenti V, chiamiamo *faccia* τ di σ ($\tau \prec \sigma$) l'inviluppo convesso di un sottoinsieme non vuoto $S \subset V$.

Un insieme di $k+1$ punti ha $2^{(k+1)}$ sottoinsiemi (includendo l'insieme vuoto $\emptyset$) quindi σ ha $2^{(k+1)} - 1$ facce.

Definizione 1.3. Un *complesso simpliciale* è una collezione finita di simplessi K che soddisfa le seguenti condizioni:

1. $\sigma \in K$ e $\tau \prec \sigma$ allora $\tau \in K$
2. $\sigma_1, \sigma_2 \in K$ allora $\sigma_1 \cap \sigma_2 = \emptyset$ o $\sigma_1 \cap \sigma_2 \prec \sigma_1$ e $\sigma_1 \cap \sigma_2 \prec \sigma_2$

La *dimensione* di K è la dimensione massima tra i suoi simplessi.
Un *sottocomplesso* di K è un complesso simpliciale $L \subset K$. Un sottocomplesso particolare è il *j-scheletro* che contiene tutti i simplessi di dimensione non superiore a j, $K^{(j)} = \{\sigma \in K \mid dim\sigma \leq j\}$. Lo 0-scheletro è anche chiamato insieme dei vertici $VertK = K^{(0)}$
Una *stella* di un simplesso σ è la collezione di tutte le *cofacce* di τ, cioè $\mathrm{St}(\tau) = \{\sigma \in K \mid \tau \leq \sigma\}$. Aggiungendo le facce mancanti possiamo ottenere un complesso simpliciale, cioè la *stella chiusa* $\bar{St}(\tau)$ che è il più piccolo sottocomplesso di σ che contiene $\mathrm{St}(\tau)$.
É possibile costruire complessi simpliciali in maniera astratta senza doversi preoccupare di rappresentarli geometricamente.

Definizione 1.4. Un *complesso simpliciale astratto* A è una collezione finita di insiemi che soddisfano la condizione $\alpha \in A$ e $\beta \subseteq \alpha$ allora $\beta \in A$

Gli elementi α in A sono i *simplessi.* La *dimensione* di un simplesso è uguale a $card(\alpha) - 1$ e la dimensione del complesso simpliciale astratto è la dimensione del simplesso con dimensione massima. Una faccia di α è un sottoinsieme non vuoto $\beta \subseteq \alpha$. L'*insieme dei vertici* è l'unione di tutti i simplessi, $VertA = \bigcup A$. B si dice *sottocomplesso* se è un complesso simpliciale astratto e $B \subseteq A$. Due complessi

simpliciali astratti sono *isomorfi* se esiste una biiezione $b : VertA \to VertB$ tale che $a \in A$ se e solo se $b(a) \in B$. Il più grande complesso simpliciale astratto con un insieme di vertici di cardinalità $n+1$ è un n-simplesso con $2^{(n+1)} - 1$ facce.

1.2.2 Omologia simpliciale

Possiamo considerare l'omologia come il compromesso tra l'intuizione geometrica di un oggetto e la necessità di avere qualcosa che sia computabile. L' $i-$esimo gruppo di omologia di uno spazio X, $H_i(X)$, è un gruppo abeliano i cui elementi non nulli sono oggetti che danno informazioni sui "buchi" $i-$dimensionali di X.

Introduciamo i concetti di *p-catena*, *p-ciclo* e *p-bordo* che ci serviranno per definire i gruppi di omologia. Una *p-catena* è una combinazione lineare formale finita

$$\sum_{i=1}^{n} a_i \sigma_i$$

con σ_i $p-$simplessi e $a_i \in \mathbb{Z}_2$. L'insieme delle $p-$catene dotato dell'operazione somma forma il *gruppo delle p-catene* $(C_p, +)$ (o più semplicemente C_p). L'elemento neutro è la catena nulla e l'associatività del $+$ è ereditata dalla somma in $\mathbb{Z}_2$. Inoltre $\partial_p : C_p \to C_{p-1}$ è l'operatore bordo definito nel seguente modo: sia $\sigma =< u_0, ..., u_p >$ un $p-$simplesso, allora il bordo di σ:

$$\partial_p(\sigma) = \sum_{i=0}^{p} < u_0, ..., \widehat{u_i}, ..., u_p >$$

dove $\widehat{u_i}$ viene cancellato ottenendo una catena di $p-1$-simplessi.
Dal *lemma fondamentale dell'omologia* abbiamo il seguente risultato: $\partial^2 = 0$.
Un *p-ciclo* è una $p-$catena c tale che il suo bordo sia dato dalla catena nulla cioè $\partial(c) = 0$. Si indica con $Z_p(K)$ l'insieme dei p-cicli di K.
Un *p-bordo* è una p-catena b per la quale esiste una $p+1$-catena c tale che $b = \partial_{p+1} c$
L'insieme dei p-bordi di K si indica con $B_p(K)$.
I tre insiemi si strutturano come gruppi e spazi vettoriali e si dimostra che:

$$B_p \subseteq Z_p \subseteq C_p$$

L'operatore bordo definisce un complesso di catene

$$... \xrightarrow{\partial_{p+2}} C_{p+1} \xrightarrow{\partial_{p+1}} C_p \xrightarrow{\partial_p} C_{p-1} \xrightarrow{\partial_{p-1}} ...$$

che ha senso per $0 < p \leqslant dimK$. Posso riscrivere i cicli e i bordi in termini dell'operatore ∂:

$$Z_p = Ker\partial_p \quad B_p = Im\partial_{p+1}$$

Definiamo il *p-esimo gruppo di omologia simpliciale di K* come il gruppo quoziente

$$H_p(K) = Z_p(K)/B_p(K)$$

Chiamiamo $\beta_p = dimH_p$ il *p-esimo numero di Betti* definito per ogni $p \geqslant 0$. Questo numero è un invariante topologico e rappresenta in termini intuitivi il numero di buchi p-dimensionali presenti in uno spazio. Ad esempio, β_0 rappresenta il numero delle componenti connesse del complesso simpliciale, β_1 ci dà il numero dei buchi delimitati da 1-cicli (curve chiuse) del complesso simpliciale e così via. Il numero di Betti ha senso da β_0 a β_d con d dimensione dello spazio.
Essendo ∂ un omomorfismo tra spazi vettoriali, possiamo associare una matrice A_n rispetto alle basi C_n e C_{n-1} date dagli n e (n-1)-simplessi definita nel seguente modo:

$$A_n(i,j) = \begin{cases} 1 & \text{se } \sigma_i \text{ è nel bordo di } \sigma_j \\ 0 & \text{altrimenti} \end{cases} \tag{1.1}$$

Con $\sigma_i = i$-esimo (n-1)-simplesso e $\sigma_j = j$-esimo n-simplesso

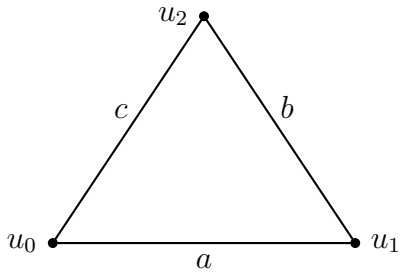


Figura 1.2: Complesso simpliciale K formato da tre vertici e tre edges

Esempio 1.2.1. *Calcoliamo i gruppi di omologia del complesso simpliciale K in figura 1.2.*

$$A_1 = \begin{array}{c} \\ u_0 \\ u_1 \\ u_2 \end{array} \begin{array}{c} \begin{array}{ccc} a & b & c \end{array} \\ \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \end{array}$$

$dim\,\beta_0 = b_0 = dim\,Im\,\partial_1 = rank\,A_1$ cioè $b_{i-1} = rank\,A_i$. Riduciamo la matrice nella sua forma normale di Smith, cioè in una matrice della forma $\left(\begin{smallmatrix} I & 0 \\ 0 & 0 \end{smallmatrix}\right)$ attraverso operazioni lecite di somma sulle righe e le colonne:

$$A_1 = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Quindi $rank\,A_1 = b_0 = 2$. Inoltre sappiamo che b_1 e b_2 sono uguali a zero in quanto non esistono $2-$simplessi da inserire nella matrice. I numeri di Betti si calcolano attraverso la formula $\beta_p = n_p - b_p - b_{p-1}$ dove n_p è il numero di simplessi di dimensione p, e b_p è il rango della matrice A_{p+1}. In questo caso abbiamo $\beta_0 = 1$ e $\beta_1 = 1$ che corrispondono rispettivamente ad una componente connessa e ad un $1-$ciclo che delimita un buco.

1.3 Visualizzazione e analisi delle strutture musicali

La letteratura scientifica ha proposto diversi modelli per visualizzare ed analizzare le strutture musicali: oltre al classico pentagramma possiamo trovare modelli più o meno esotici come il *Tonnetz* creato da Eulero che rappresenta gli intervalli musicali disposti in una griglia [6], il diamante delle tonalità di Harry Partch dove due assi rappresentano gli intervalli musicali più importanti [7], la spirale di Chew [8] e molti altri.
La domanda che ci possiamo porre è: come si possono definire delle regole affinché i modelli riflettano la realtà musicale?

1.3.1 Tonnetz

Il Tonnetz è un diagramma che compare per la prima volta nel *"Tentamen novae theoriae musicae"* scritto da Eulero nel 1739. Lo scopo di questo diagramma è analizzare gli intervalli di consonanza massimale: la quinta giusta e la terza maggiore. Il diagramma generato da questi due intervalli consente di trattare le dodici altezze della scala cromatica enarmonicamente[2]. Il Tonnetz quindi nasce per rappresentare i rapporti intervallari e fornire un supporto ad un problema di tipo acustico.

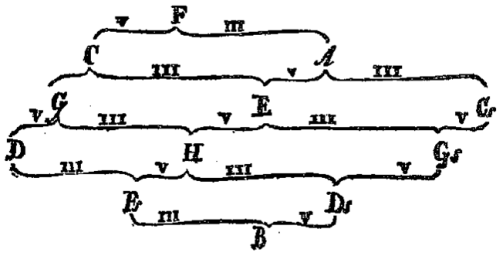


Figura 1.3: Tonnetz di Eulero

Qualche secolo dopo, nella seconda metà del XIX secolo, Arthur von Oettingen e Hugo Riemann ridefiniscono il Tonnetz come griglia tonale, ossia un piano generato da due assi dove quello orizzontale genera l'intervallo di quinta tra due altezze mentre l'asse verticale genera la terza (maggiore e minore). Il Tonnetz descritto da Oettingen e Riemann si sviluppa su un piano infinito delle altezze con temperamento

[2]Per enarmonia si intende il rapporto tra due note che hanno nomi diversi ma lo stesso suono grazie al temperamento equabile (per es. do diesis, re bemolle).

giusto sebbene Riemann stesso suggerisca l'identificazione enarmonica delle altezze nella scala cromatica al fine di semplificare l'accordatura degli strumenti musicali. Per maggiori dettagli sulla storia del Tonnetz, vi suggeriamo di consultare [9] [10] [11].
Il Tonnetz diventa il modello d' adozione per la teorie neo-riemanniane in quanto compaiono tutti gli elementi sui quali si fondano queste teorie [3]. Infatti, sul Tonnetz è possibile identificare con molta chiarezza le trasformazioni *P, L* e *R* come mostrato in figura 1.4.

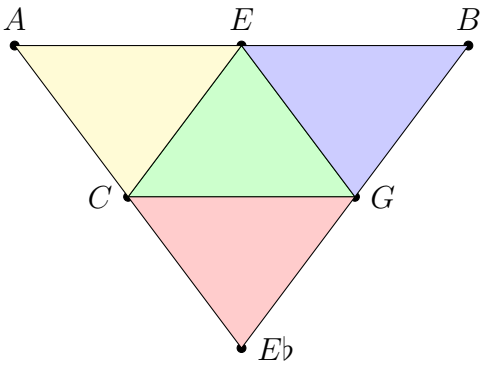


Figura 1.4: L'accordo iniziale è il triangolino verde (C,E,G) che rappresenta l'accordo di Do maggiore. La trasformazione R è il triangolino giallo (La minore), la trasformazione L è il triangolino viola (Mi minore) e il triangolino rosa rappresenta la trasformazione P (Do minore).

Molti studi hanno investigato diverse derivazioni del Tonnetz, spesso identificato come il *Tonnetz generalizzato*. Ad esempio il Tonnetz tridimensionale introdotto da Gollin [13] corrisponde ad un estensione dell'originale con un ulteriore asse intervallare che include gli accordi di settima dominante e rappresentati da tetraedri. Strutture simili a questa possono essere costruite in vari modi, ad esempio associando intervalli diatonici anzichè cromatici. In questa particolare configurazione i vertici rappresentano solo note appartenenti ad una sola scala diatonica [14].

La rappresentazione contemporanea del Tonnetz è proprio quella neo-Remanniana, cioè quella di un grafo semplice, 6-regolare, formato da 12 vertici e costruito nel seguente modo: ad ogni vertice viene assegnata una delle dodici classi di altezze e due altezze sono adiacenti secondo gli intervalli di quinta, terza minore e terza

maggiore (e dei rispettivi rivolti di quarta, sesta sesta minore e sesta maggiore).

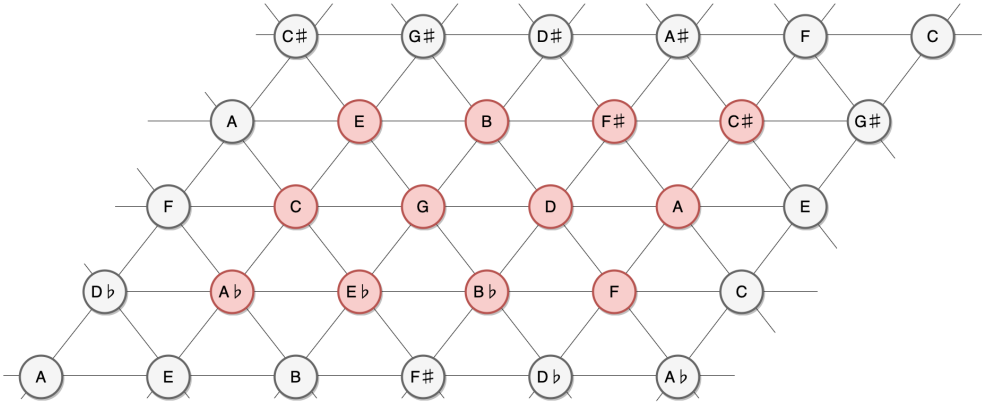


Figura 1.5: Rappresentazione del Tonnetz (area evidenziata)

1.3.2 Rappresentazione simpliciale di un insieme di accordi

Abbiamo visto precedentemente che è possibile formalizzare gli accordi come collezioni di classi di altezze. Uno dei vantaggi di questa rappresentazione sta nel fatto che ogni sottocollezione corrisponde ad un sottoaccordo dell'accordo rappresentato dalla collezione. Questo risultato ci permette di studiare progressioni armoniche complesse con accordi di 5 o 6 note che possono essere visti come una combinazione di triadi [2].
Ci sembra importante conservare questa proprietà nella rappresentazione simpliciale degli accordi. La struttura di un complesso simpliciale impone proprio che ogni $d-$simplesso, associato ad un accordo di $d+1$ note, abbia le $p-$facce $(p < d)$ appartenenti al complesso e associate agli accordi di $p+1$ note.
Un accordo C si può rappresentare come collezione simpliciale A definita nel seguente modo:

1 $|A|$ è un simplesso di dimensione uguale a $card\ (C) - 1$.

2 Ogni vertice di $|A|$ corrisponde ad un'unica classe di altezze.

3 Ogni combinazione di p note σ in $|A|$ $(0 < p \leq d)$ corrisponde al simplesso $A(\sigma) = \{A(\tau) | \tau \prec \sigma \wedge dim(\tau) = 0\}$

Posso quindi considerare un insieme di n classi di altezze e rappresentarlo con un $(n-1)$-simplesso. Chiamiamo $S(A)$ il simplesso che rappresenta un insieme di classi di altezze A. I simplessi che costituiscono la chiusura di $S(A)$ rappresentano le parti dell'accordo A (*subchords*).
Questo metodo ci assicura che ogni accordo venga rappresentato in un solo modo all'interno del complesso simpliciale. Un accordo di tre note si può rappresentare come un 2-simplesso sia in forma geometrica che astratta.

Un risultato interessante per quanto riguarda la rappresentazione simpliciale del Tonnetz descritto precedentemente è il seguente:

Proposizione 1.3.1. *Il Tonnetz si può realizzare su un toro.*

Dimostrazione. Disponiamo i 12 vertici sul toro T^2 e colleghiamo i punti con degli archi. Chiamiamo facce le regioni determinate dagli archi che possono essere rettificati. La caratteristica di Eulero del toro è $\chi(T^2) = 0$. Quindi $\chi(T^2) = v - e + f = 0$. Andando a sostituire il numero di vertici e di spigoli otteniamo che $f = 24$. Considerando che ogni faccia ha 3 spigoli ho: $2e \geq 3f$.
Andando a sostituire il numero di spigoli e facce otteniamo lo stesso valore da entrambe le parti. □

Quindi il Tonnetz è immergibile in un toro e ammette 24 facce triangolari delimitate tutte da 3-cicli[3] Seguendo il cammino dato dagli intervalli di quinta otteniamo un intero circolo delle quinte (un 12-ciclo), mentre il cammino delle terze minori e maggiori danno rispettivamente quattro 3-cicli disgiunti e tre 4-cicli disgiunti.
Possiamo notare che il diagramma così definito sia un' approssimazione di un modello per le relazioni di contrappunto limitato ad alcuni accordi [17]. Infatti, il diagramma contiene una nozione di distanza musicale molto diversa rispetto a quella descritta inizialmente nello spazio delle classi d'altezza. Ad esempio, le note A e A♭ sono a un semitono di distanza nello spazio quoziente (cioè la più piccola distanza possibile tra due note diverse) mentre sul Tonnetz non sono affatto vicine. Questo fatto mette in luce alcuni comportamenti fondamentali del voice-leading affinché possa funzionare all'interno del Tonnetz. Gli accordi privilegiati sono quelli che Tymoczko chiama

[3]Per m-ciclo si intende una catena di lunghezza m del tipo $(v_0, ..., v_{m-1}, v_0)$. Questa definizione è diversa rispetto a quella del ciclo topologico data precedentemente.

even chords che corrispondono alle sonorità più familiari della musica occidentale. Ad esempio, un accordo di tre note maggiore o minore divide l'ottava in maniera relativamente equa: un accordo maggiore corrisponde alla sequenza nel pitch class space (0,4,8) che sono distribuiti equamente sullo spazio rispetto, ad esempio, ad un cluster cromatico $(C, C\sharp, D)$ corrispondente alla sequenza (0,1,2).

Consideriamo ora un nuovo Tonnetz che contiene tutte le possibili combinazioni di accordi all'interno dell'insieme delle dodici classi di altezza della scala a temperamento equabile. In particolare, assemblando correttamente i vari simplessi, otteniamo un complesso simpliciale che è un $11-$simplesso e contiene tutti le possibili combinazioni dei vertici che corrispondono agli elementi del *pitch class set*. D'ora in poi considereremo il complesso simpliciale solamente nella sua configurazione astratta. Infatti, questo $11-$simplesso (con tutte le sue facce) è rappresentabile in $\mathbb{R}^{11}$, ben lontano dalle 3 dimensioni che abbiamo a disposizione per visualizzare gli oggetti geometrici[4]
A differenza del Tonnetz descritto precedentemente, questa nuova versione è un grafo semplice 12-regolare completo. I vertici rimangono 12 come le classi delle altezza mentre aumenta la cardinalità dei simplessi di dimensione maggiore. Ad esempio il numero di spigoli e le facce diventa:

- Spigoli/Bicordi $e = \binom{12}{2} = 66$
- Triangoli/Accordi $f = \binom{12}{3} = 220$

Giustificheremo la scelta di questa generalizzazione brutale del Tonnetz in un secondo momento. Possiamo da subito notare che da una parte abbiamo un modo per inserire all'interno del grafo gli accordi di più di tre note e quelli non "even", ma dall'altra abbiamo messo tutte le possibili combinazioni di accordi sullo stesso piano, andando ad annientare qualsiasi argomentazione sulla "compattezza armonica" del Tonnetz[18].

[4]Per il *teorema di realizzazione geometrica*, sappiamo che un qualunque complesso di dimensione n è rappresentabile in $\mathbb{R}^{2n+1}$, ma il caso specifico di un $n-$simplesso con tutte le sue facce è rappresentabile direttamente in $\mathbb{R}^n$.

Capitolo 2

Analisi topologica

2.1 Omologia persistente

I problemi di classificazione e riconoscimento di forme vengono spesso risolti attraverso trasformazioni geometriche (euclidee, affini, proiettive). In particolare, associando ad ogni oggetto una matrice di misure geometriche, è possibile confrontare più oggetti utilizzando queste matrici dette descrittori di forma. Questo approccio funziona bene quando si lavora con oggetti rigidi ma è sconveniente quando, ad esempio, bisogna riconoscere la somiglianza tra un'immagine di un uomo seduto e di uno in piedi. In quest'ultimo caso è più pratico utilizzare la topologia definendo un omeomorfismo che associ le due immagini invece di lavorare con le trasformazioni matriciali.

D'altra parte, se la geometria risulta troppo rigida, la topologia permette di costruire omeomorfismi tra oggetti che sono parecchio diversi tra loro (il classico esempio che, secondo un topologo, una tazza col manico è omeomorfa a una ciambella ne è la prova). Il concetto di forma nella topologia persistente è qualcosa di diverso: non è più associato ad un solo spazio topologico X ma ad una coppia (X, f) dove f è una funzione continua detta *funzione filtrante* definita su X. La topologia persistente (e in particolare il settore di omologia persistente) ci consente di lavorare con delle forme a livello topologico offrendo dei descrittori che preservano alcune caratteristiche geometriche degli oggetti.

L'omologia persistente è stata introdotta da Patrizio Frosini e altri collaboratori con il nome di *Size Theory*, un formalismo focalizzato al problema del riconoscimento

delle forme [28]. Questa teoria è stata indipendentemente sviluppata da Edelsbrunner e collaboratori[29]. In questo elaborato considereremo solo il caso dell'omologia persistente su complessi simpliciali anche se la teoria si riferisce al caso più generale di spazi topologici.

Sia K un complesso simpliciale finito. Definiamo *filtrazione* una successione ordinata di sottocomplessi di K tale che $K_i \subseteq K_{i+1}$. Sia $f : K \to \mathbb{R}$ una funzione reale non decrescente rispetto alle successioni crescenti di facce di K. Cioè: siano $\sigma, \tau \in K$ e $\tau \prec \sigma$, allora $f(\tau) \leqslant f(\sigma)$.

Per ogni $a \in \mathbb{R}$, il *sottolivello* $K(a) = f^{-1}((-\infty, a])$ è un sottocomplesso di K. Le ipotesi su f assicurano che l'ordine dei simplessi dato dai valori di f induca una filtrazione:

$$\emptyset = K_0 \subseteq K_1 \subseteq ... \subseteq K_p = K$$

Per ogni coppia i, j tale che $0 \leq i \leq j \leq p$, l'inclusione $K_i \hookrightarrow K_j$ induce un omomorfismo sui gruppi di omologia simpliciale per ogni dimensione n:

$$f_{n\sharp}^{i,j} : \mathbb{H}_n(K_i) \to \mathbb{H}_n(K_j)$$

Per ogni $n \in 0, ..., p$ i *gruppi di n-omologia persistente* sono le immagini di questi omomorfismi e il *numero di Betti di n-persistenza* è il rango dei gruppi:

$$\beta_n^{i,j} = dim(Imf_{n\sharp}^{i,j})$$

I numeri di Betti di n-persistenza contano quante classi di omologia di dimensione n sopravvivono nel passaggio da K_i a K_j. Diremo che la classe di omologia $[\alpha] \in \mathbb{H}_n(K_i)$ nasce entrando in K_i se α non era nel sottocomplesso precedente, cioè $\alpha \notin Imf_n^{i-1,i}$. In maniera analoga, se α nasce in K_i, muore entrando in K_j se l'immagine della mappa indotta dall'inclusione $K_{i-1} \subseteq K_{j-1}$ non contiene l'immagine di α ma l'immagine della mappa indotta da $K_{i-1} \subseteq K_j$ la contiene. In questo caso la persistenza di α è $j - i$.

La persistenza può essere calcolata in modo efficiente attraverso un algoritmo di riduzione matriciale. Per calcolarlo, andiamo ad ordinare i simplessi, cioè definiamo una successione $\sigma_1, \sigma_2, \sigma_3, ...$ tale che $f(\sigma_i) < f(\sigma_j)$ implica $i < j$ cioè σ_i è faccia di σ_j. Quest'ordinamento esiste perché f è una funzione monotona. Notiamo che ogni sottosuccessione dei simplessi forma un sottocomplesso di K.

Consideriamo ora la matrice bordo descritta in (1.1) e la riduciamo in una nuova $0-1$ matrice R. Sia *low(j)* l'indice della riga più bassa nella colonna j. Se l'intera colonna è zero allora *low(j)* è indeterminato. Chiamiamo R *ridotto* se *low(j)* $\neq$ *low(j_0)*. L'algoritmo riduce la matrice aggiungendo colonne da sinistra a destra ed è consultabile in [26].
Il diagramma di persistenza si costruisce considerando i valori di nascita u e morte v delle classi di omologia come punti $(u, v) \in \mathbb{R}^2$. Più precisamente, sia $\Delta = \{(u, v) \in \mathbb{R}^2 | \, u = v\}$ la diagonale del piano euclideo, $\Delta^+ = \{(u, v) \in \mathbb{R}^2 | \, u < v\}$ il semipiano aperto sopra la diagonale e $\Delta^* = \Delta^+ \bigcup \{(u, \infty) | \, u \in \mathbb{R}^2\}$ la chiusura del semipiano che include i punti all'infinito. Possiamo quindi introdurre la definizione di diagramma di persistenza.

Definizione 2.1. Il *diagramma di k-persistenza* $D_k(f)$ è una collezione di punti in Δ^* uniti ai punti della diagonale Δ con molteplicità infinita.

In questo diagramma i punti (u, v) sono detti *cornerpoints* mentre le classi che nascono in u e non muoiono nella filtrazione vengono rappresentate con delle semirette (u, ∞) parallele all'asse delle ordinate e vengono chimate *cornerlines*.

Esempio 2.1.1. *Mostriamo un esempio di diagramma di* $0-$*persistenza associato ad una filtrazione del complesso simpliciale* K*. Definiamo la filtrazione come la sequenza dei seguenti complessi simpliciali:*

$K_0 = \{\emptyset\}$ $\qquad$ $K_3 = \{\{u_0, u_1\}, u_0, u_1, u_2, \emptyset\}$

$K_1 = \{u_0, \emptyset\}$ $\qquad$ $K_4 = \{\{u_1, u_2\}, \{u_0, u_1\}, u_0, u_1, u_2, \emptyset\}$

$K_2 = \{u_0, u_1, u_2, \emptyset\}$ $\qquad$ $K_5 = \{\{u_0, u_2\}, \{u_1, u_2\}, \{u_0, u_1\}, u_0, u_1, u_2, \emptyset\}$

Disegnamo il diagramma di persistenza associato alla filtrazione:

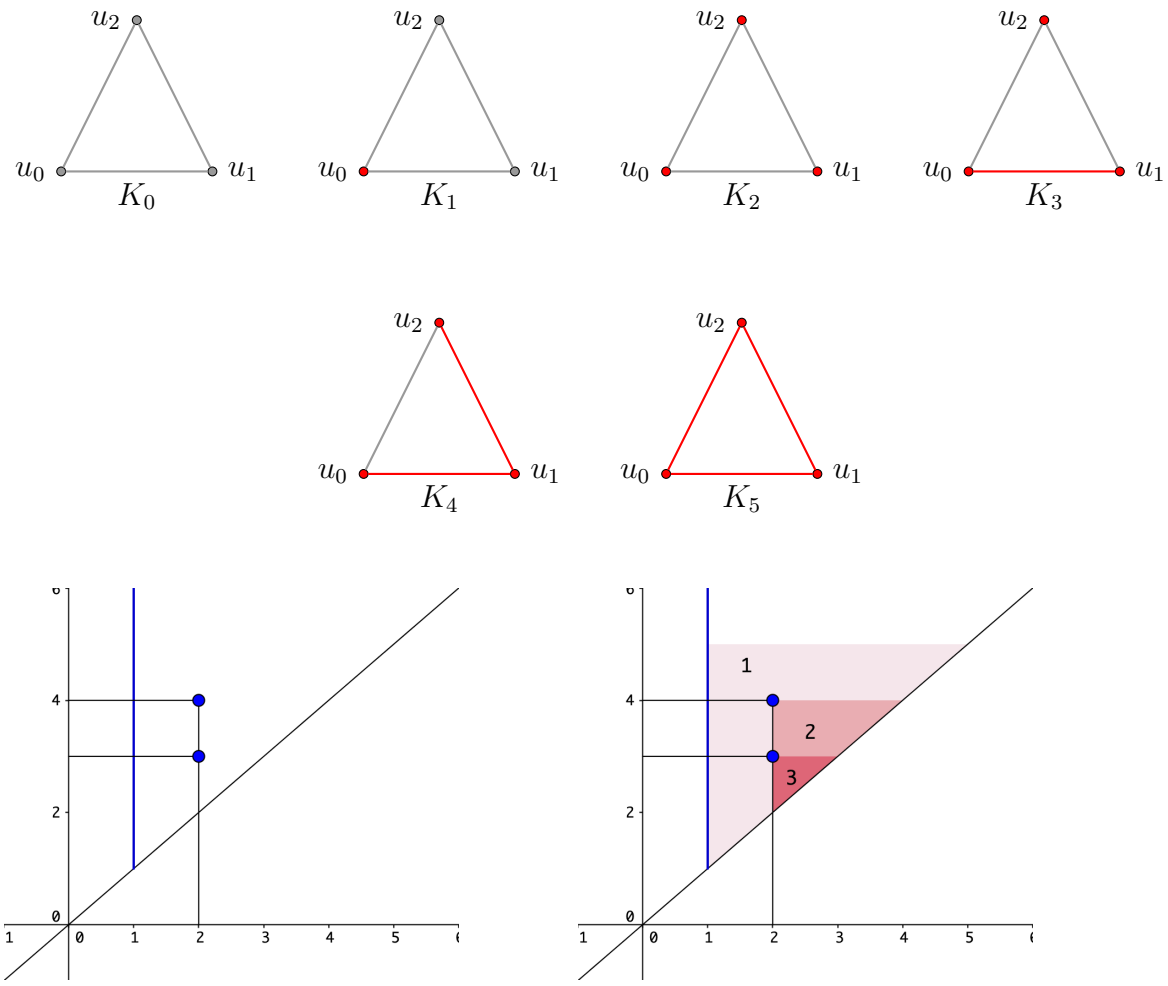


Figura 2.1: Diagramma di 0−persistenza relativo alla filtrazione su K. Nel diagramma a destra sono evidenziati i numeri di Betti β_0 relativi ai livelli della filtrazione.

2.1.1 Distanza bottleneck

I diagrammi di persistenza sono più facili da manipolare rispetto agli spazi che rappresentano. La *distanza bottleneck o distanza di matching* è una delle distanze che ci permettono di mettere a confronto due diagrammi.

Definizione 2.2. Sia X uno spazio topologico triangolabile e $f, g : X \to \mathbb{R}$ due funzioni di taglia. La distanza bottleneck tra i diagrammi di k-persistenza $D_k(f)$ e $D_k(g)$ è:

$$d_B(D_k(f), D_k(g)) = \inf_{\gamma} \sup_{p \in D_k(f)} ||p - \gamma(p)||_\infty$$

dove $\gamma : D_k(f) \to D_k(g)$ è una biiezione tra i diagrammi di persistenza.
In pratica, si considerano tutti i possibili accoppiamenti (matching) tra i cornerpoints di $D_k(f)$ e $D_k(g)$. Il valore di ogni accoppiamento è dato dalla distanza massima tra le coppie di punti. La distanza bottleneck tra i due diagrammi è il minimo tra tutti i possibili accoppiamenti.

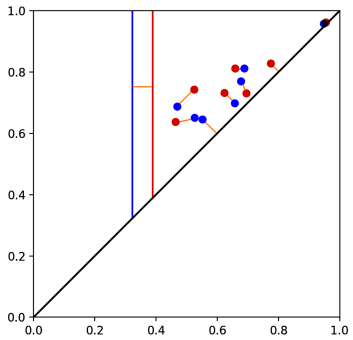


Figura 2.2: Matching tra i punti di due diagrammi di persistenza (blu e rossi). La distanza bottleneck è il valore massimo della migliore corrispondenza tra i cornerpoints dei due diagrammi.

Utilizzeremo questa distanza per confrontare i diagrammi di persistenza delle composizioni. Prima di procedere, notiamo che la bottleneck non è l'unica distanza che possiamo definire sui diagrammi di persistenza. Potremmo scegliere di applicare una distanza diversa come quella di Wasserstein, più sensibile ai dettagli del diagramma ma meno stabile rispetto al rumore.

2.2 Analisi sul tonnetz

2.2.1 Il Tonnetz deformato

Consideriamo il Tonnetz come definito in Bergomi [1], cioè come il grafo formato da tutte le classi di altezza e assumiamo di poter dare un peso ai vertici in base al numero di volte che una classe di altezze compare all'interno di un brano musicale. Cioè definiamo una funzione sull'insieme dei vertici $h : V \to \mathbb{R}$ che associa ad ogni vertice un valore corrispondente al numero di volte che la classe di altezze associata al vertice compare nel brano. La funzione h corrisponde alla composizione:

$$h : V \xrightarrow{\text{l}} L \xrightarrow{\text{c}} \mathbb{R}$$

dove l associa i vertici alle classi di altezze e c conta la frequenza delle classi. Andando ad applicare questa funzione sui vertici di un grafo ben strutturato come il Tonnetz, otterremo una versione deformata del Tonnetz immerso in $\mathbb{R}^3$ dove i vertici rimangono collegati dagli edge e hanno altezza uguale a valore della funzione $h(v)$. Il Tonnetz è già stato usato per classificare generi musicali in [18] attraverso l'analisi della "compattezza" delle strutture simpliciali che rappresentano la traccia di un brano musicale in diversi tipi di Tonnetz planari.
La deformazione del Tonnetz attraverso la funzione h ci permette di avere informazioni sugli aspetti armonici dei brani analizzati. Otteniamo quindi un Tonnetz deformato da una funzione che considera i vertici e modifica le altezze di questi in base al numero di apparizioni nel brano musicale. I ruolo degli edges, dei triangoli e di tutti gli altri simplessi è quindi quello di "tenere uniti" i punti dello spazio deformato.

Proviamo ora a dare un significato ad una funzione "altezza" che valga per tutti gli accordi che compaiono all'interno di un brano musicale. L'idea più semplice è proprio quella di associare un accordo ad un simplesso che appartiene al complesso simpliciale che contiene tutti i possibili simplessi generati dalle classi di altezza definito nel primo capitolo. Ogni volta che un accordo compare nel brano analizzato, deformiamo il complesso simpliciale in favore di quell'accordo e di tutti i suoi sotto-accordi. In termini matematici, definiamo K come lo spazio di tutti i possibili accordi di p note ($p \leq 12$) e $h : K \to \mathbb{R}$ funzione filtrante. Andiamo a definire la filtrazione.

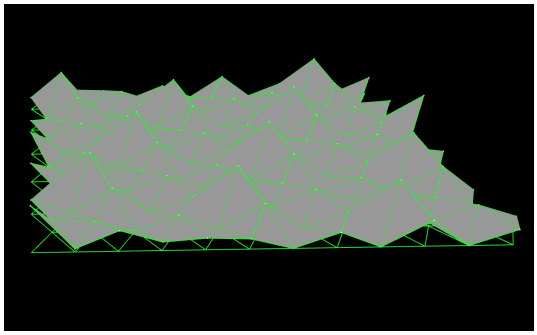


Figura 2.3: Screenshot di un Tonnetz deformato preso dall'applicazione web sviluppata da Bergomi consultabile al sito `http://nami-lab.com/tonnetz/examples/deformed_tonnetz_int_sound_pers.html`

Costruzione della funzione filtrante

Dalla definizione di funzione filtrante, vogliamo che h sia non decrescente rispetto alle catene di facce crescenti, cioè $h(\tau) \leq h(\sigma)$ se τ è faccia di σ. Questa definizione ci porta a fare qualche considerazione sulla rappresentazione di un insieme di accordi come complesso simpliciale. Infatti, come abbiamo detto in precedenza, dato un accordo C di d note, la sua rappresentazione simpliciale A consiste nel simplesso generato dai vertici che corrispondono alle classi di altezza e in ogni combinazione di p vertici $(0 < p \leq d)$ corrispondente al simplesso $A(\sigma) = \{A(\tau) | \tau \prec \sigma \wedge dim(\tau) = 0\}$. Ad esempio, l'accordo di Do Maggiore (*Do, Mi, Sol*) è rappresentato da un $2-$simplesso $\{v_0, v_1, v_2\}$ e da tutte le sue facce $\{v_0, v_1\}$,$\{v_1, v_2\}$,$\{v_0, v_2\}$,$\{v_0\}, \{v_1\}, \{v_2\}$. Secondo questa costruzione la funzione altezza $h(\tau) \geq h(\sigma)$ con $\tau \prec \sigma$ che è proprio il contrario di quello che si dovrebbe avere per definire h funzione filtrante.
Per risolvere il problema dobbiamo considerare la funzione filtrante rovesciata, cioè definiamo una funzione che consideri quanto un accordo venga suonato all'interno della composizione e prendiamo il suo complementare rispetto alla durata complessiva del brano. Quindi la nostra funzione filtrante ci dice quanto un accordo non viene suonato all'interno del brano. Riassumiamo l'algoritmo per calcolare la filtrazione in 2.4.
Gli accordi con valori più bassi saranno quelli che vengono suonati più volte, ai primi posti ci saranno i vertici quindi le classi di altezza perché compaiono in ogni accordo che li contiene. Gli ultimi elementi sono quelli con valori $\simeq 1$ e corrispondono agli abbellimenti o accordi suonati di meno. In questo modo possiamo considerare una

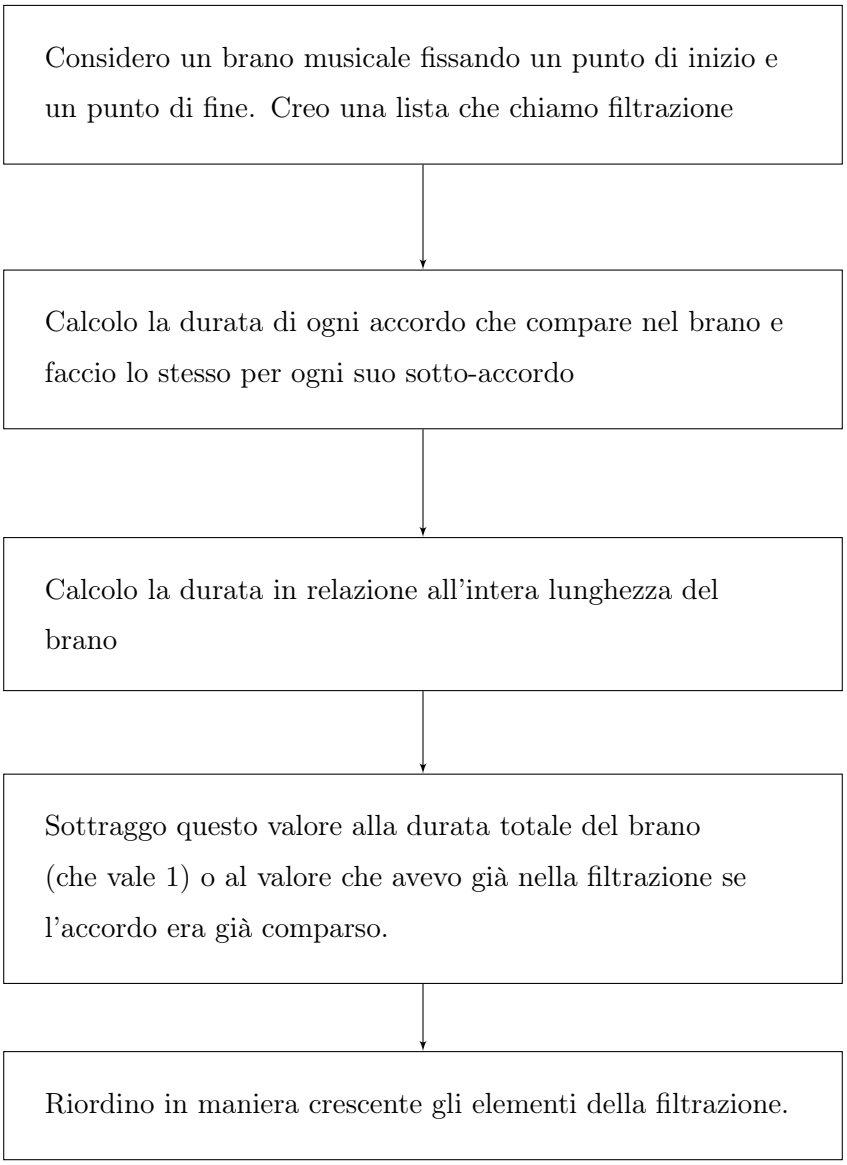


Figura 2.4: Algoritmo utilizzato per la creazione della filtrazione

funzione filtrante che è monotona non decrescente e vale la condizione $h(\tau) \leq h(\sigma)$ con $\tau \prec \sigma$.

Possiamo quindi considerare i sottolivelli $K(t) = h^{-1}((-\infty, t])$ che sono dei sotto-complessi di K che contengono i simplessi corrispondenti agli accordi suonati per una durata maggiore o uguale a $1 - t$.

Questo tipo di funzione h ci permette, nel contesto di analisi musicale, di lavorare in tonalità relative invece di averne una assoluta. Cioè, questa funzione ci garantisce l'invarianza per trasposizioni di un brano musicale al costo di perdere importanti informazioni sulla natura armonica del brano stesso.

Questo fatto è molto importante perché mette in luce uno dei difetti principali di questo modello. Ad esempio, la progressioni armoniche 2.1a e 2.1b risulteranno identiche come complessi simpliciali.

$$(C, E, G) \to (E, G, B) \to (D\sharp, F\sharp, B) \tag{2.1a}$$

$$(C, E, G) \to (E, G, B\flat) \to (A\flat, A, B\flat) \tag{2.1b}$$

Questo fatto può risultare un problema nel processo di classificazione ed è la diretta conseguenza della generalizzazione degli accordi in simplessi.
Fortunatamente, la gran parte del corpus che considereremo, così come la gran parte della musica occidentale, si basa su progressioni di accordi appartenenti ad una famiglia ristretta di rapporti intervallari quindi è piuttosto raro trovare una progressione armonica come (2.1b) soprattutto per quanto riguarda il corpus di composizioni fatte fino alla fine del XIX secolo, cioè fino all'avvento della musica post-tonale in tutte le sue declinazioni.

Capitolo 3

Risultati

3.1 Caratteristiche dei diagrammi di persistenza

Nel lavoro di Bergomi [1] sono stati analizzati i diagrammi di 0 e 1−persistenza associati ad alcune composizioni che rispecchiavano diversi stili musicali. Sintetizziamo le caratteristiche di questi diagrammi.
La 0−persistenza ci dà informazioni sul numero di componenti connesse dello spazio. Nel caso del Tonnetz planare deformato, essendo un connesso omeomorfo ad un toro $\mathbb{T}^2$, ci si aspetta che ci sia sempre una cornerline e dei cornerpoints disposti nel diagramma in maniera da far intuire dei caratteri stilistici per ogni brano.
Il livello della nascita della cornerline gioca un ruolo fondamentale nell'interpretazione musicale di questi diagrammi. Più la cornerline si trova vicina all'origine e più la composizione è basata su scelte tonali o modali. Se la cornerline invece si trova molto distante dall'origine, ogni classe di altezza è apparsa nella composizione per un periodo significativo. Questa configurazione corrisponde a stili più contemporanei identificati come musica post-tonale.
Per quanto riguarda la 1−persistenza ci sono due cornerlines che corrispondono ai generatori del toro. In termini musicali, il valore dei livelli di nascita delle cornerlines e la distanza tra esse ci danno informazioni sul carattere stilistico della composizione.
Abbiamo scelto di analizzare diagrammi delle stesse dimensioni per rilevare le caratteristiche dei cornerpoints e cornerlines che possono essere interpretati come descrittori di forma e possibilmente di stili compositivi che caratterizzano i brani analizzati.

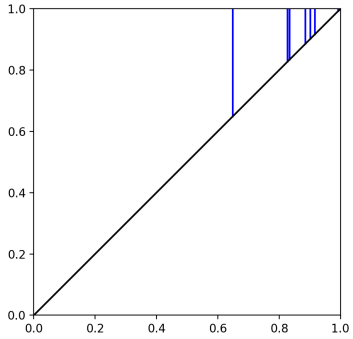
3.1.1 Diagrammi di 0-persistenza

La struttura usata per calcolare il diagramma di 0-persistenza è più semplice rispetto a quella per calcolare la $p-$persistenza con $p > 0$. Infatti, questo diagramma dipende solamente dai vertici e dagli edges di K e dalla loro sequenza ordinata. Un vertice u non ha bordo e la sua comparsa corrisponde sempre alla nascita di una nuova componente. Un edge σ_j ha due vertici come bordo $\partial(\sigma_j) = u_1 + u_2$. Andando ad osservare la filtrazione, quando ci troviamo al livello di σ_j ci riduciamo a due casi possibili: come primo caso σ_j fa nascere un $1-$ciclo se u_1 e u_2 si trovano sulla stessa componente connessa in K_{j-1}, cioè il complesso dato dalla filtrazione prima di aggiungere σ_j. Come secondo caso abbiamo che σ_j fa morire una componente connessa se u_1 e u_2 appartenevano a due diverse componenti connesse di K_{j-1}.

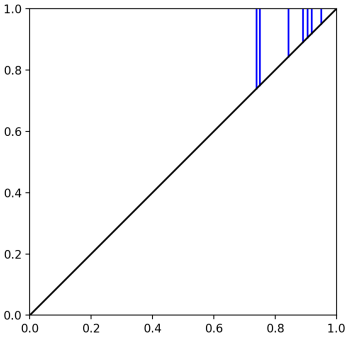
La configurazione del dominio fondamentale del Tonnetz precedente, ossia quello generato dagli intervalli di quinta e di terza maggiore e minore, ci permette di trovare un massimo di tre componenti connesse. Questa configurazione particolare si ottiene suonando un cluster cromatico, ad esempio $C, C\sharp, D$. Andando a considerare il Tonnetz generato da tutti gli intervalli possibili tra le classi di altezza, otteniamo uno spazio che consente un numero massimo di dodici componenti connesse. Questa configurazione specifica si ottiene suonando tutte e sole le classi di altezza come ad esempio una scala cromatica. Questo dato mette in luce una caratteristica del diagramma di $0-$persistenza per il nuovo Tonnetz: Oltre alla posizione, anche la cardinalità dell'insieme delle cornerlines e cornerpoints dà informazioni sulle scelte stilistiche presenti nella composizione. Mostriamo e commentiamo in seguito alcuni diagrammi di $0-$persistenza di composizioni scelte dal corpus di *Music21* [32].

Ryan's Mammoth Collection

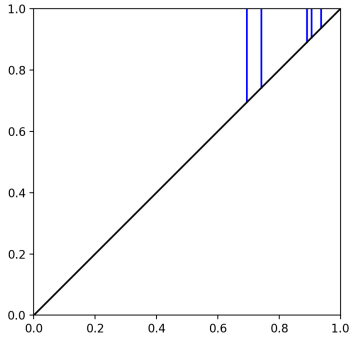
La Ryan's Mammoth Collection è una raccolta pubblicata nel 1883 ed è la più importante collezione di musica tradizionale irlandese.
Il repertorio consiste principalmente in monodie, cioè melodie ad una voce, per strumenti a fiato ed archi in cui prevalgono i modi che si sviluppano sulla scala diatonica.



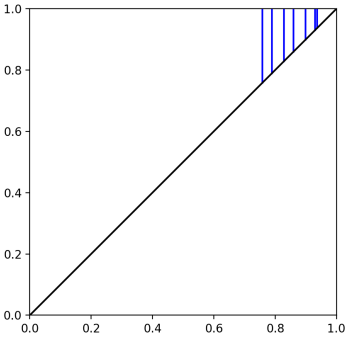
(a) 42 D Highland Regiment Strathspey



(b) Alhambra Reel



(c) All The Way To Galway Reel



(d) American Hornpipe

Figura 3.1: Diagrammi di 0-persistenza dal corpus Ryan's Mammoth Collection di Music21.

Ci aspettiamo quindi un numero massimo di sette componenti connesse che corrispondono al numero di classi di altezza della scala diatonica.
Nei brani analizzati appaiono solamente note singole che corrispondono, nel complesso simpliciale associato, a dei vertici che non si uniscono tra di loro. Quindi nel diagramma nasce un numero di cornerlines che equivale al numero di note suonate. Ricordiamo che il punto di nascita della cornerline equivale alla frequenza di non apparizione della classe d'altezza più presente all'interno del brano. In questo caso, le cornerlines che nascono in prossimità della fine del diagramma corrispondono alle note che appaiono raramente nel brano, come ad esempio gli abbellimenti[1].

[1]In notazione musicale, un abbellimento è una nota o un gruppo di note inserite nella linea melodica con funzione non strutturale, ma decorativa o espressiva.

Musica del Trecento italiano

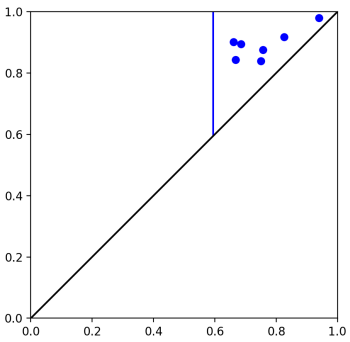
La musica del Trecento presenta alcune caratteristiche importanti: le consonanze erano basate principalmente sui rapporti di quinta e di ottava, l'intervallo di terza era trattato come dissonante e non utilizzato, specialmente nel primo periodo. Le forme principali di composizioni musicali corrispondono ai madrigali a due voci e ballate monodiche, l'uso di una terza voce si sviluppò principalmente nell'ultima parte del secolo. I versi del madrigale iniziano e finiscono con dei melismi[2].
In figura 3.2 possiamo visualizzare i diagrammi di due madrigali a tre voci (Pmfc 01, 04) e due madrigali a due voci (Pmfc 13.01, 12.21). Il diagramma di $0-$persistenza associato alle composizioni risulta piuttosto ricco. Nella maggior parte delle composizioni la prima componente connessa nasce tra 0.5 e 0.6 mentre i cornerpoints rilevanti sono circa 6, quindi durante la filtrazione appaiono sette componenti che corrispondono alle note della scala diatonica utilizzata nei madrigali. Gli altri cornerpoints che compaiono in prossimità della fine della filtrazione possono essere associati agli abbellimenti nei melismi. A differenza del corpus della Ryan's Mammoth Collection analizzato precedentemente, possiamo notare che c'è solo una cornerline. Infatti, essendo composizioni per più voci, arriveremo ad un punto della filtrazione in cui i simplessi corrispondenti agli accordi di più classi di altezza "uniscono" i vertici andando ad uccidere le componenti connesse.
La posizione della cornerline ci dà un'idea di quanto una composizione dipenda dalla tonica[3].

Messe di Palestrina

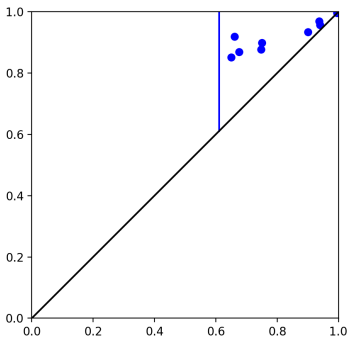
Uno degli eventi più importanti della Chiesa del XVI secolo fu la riforma protestante di Lutero. Anche l'ambiente musicale risentì gli effetti di questo cambiamento, infatti Lutero stesso incentivò la scrittura di canti monodici chiamati corali e caratterizzati da grande semplicità e regolarità metrica.
Alla riforma Luterana, la Chiesa romana rispose col Conciliio di Trento e la sua

[2]Il melisma è un tipo di ornamentazione melodica che consiste nel caricare su di una sola sillaba testuale un gruppo di note ad altezze diverse. La vocale della sillaba viene distribuita su più note e quindi cantata modulando l'intonazione.

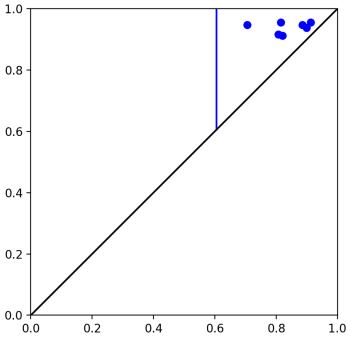
[3]La tonica è il primo grado di una scala diatonica ed è la nota che dà il nome alla scala corrispondente.



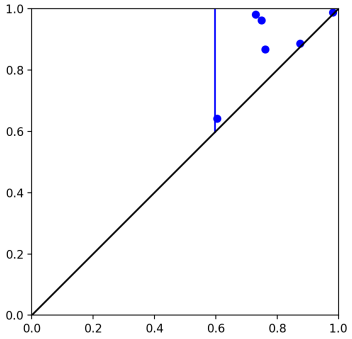
(a) Pmfc 01 - Virtutibus Laudabilis



(b) Pmfc 04 - Si Dolce Non Sono



(c) Pmfc 13.01 Kyrie Summe Clementissime



(d) Pmfc 12.21 - Benedicamus

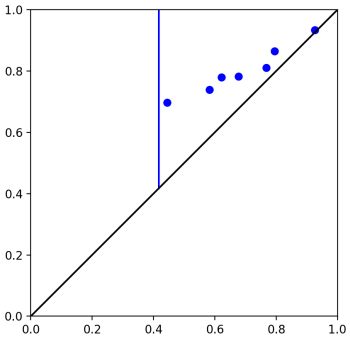
Figura 3.2: Diagrammi di 0-persistenza dal corpus di musica del Trecento di Music21. Pmfc 01 e Pmfc 04 sono madrigali a tre voci mentre Pmfc 13.01 e Pmfc 12.21 sono madrigali a due voci.

Controriforma. Anche in questo caso la musica venne *depurata da tutti gli artifici* al fine di poter sottolineare il valore del testo [33].
Giovanni Pierluigi da Palestrina (1525-1594) fu una figura primaria all'interno dell'ambiente musicale controriformista. La sua opera sarà alla base del repertorio seicentesco, che ispirerà i lavori di Bach.
Analizzando quattro *Agnus* presenti nelle messe del corpus di Palestrina in Music21 otteniamo i diagrammi in figura 3.3. I diagrammi presentano una cornerline che nasce tra 0.3 e 0.4 e sei cornerpoints, corrispondenti alle classi d'altezza della scala diatonica.

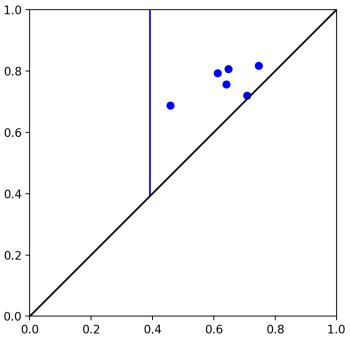
Corali di Bach

Il corale, nato all'interno della nuova Chiesa Luterana, raggiunge con Bach (1685-1750) il suo più alto livello di valorizzazione. Il repertorio di Bach rappresenta la sintesi perfetta dell'estetica musicale del periodo ed è ispirato ai criteri di Palestrina per ciò che riguarda l'andamento delle parti.
I corali bachiani sono caratterizzati da un'armonia a quattro voci che corrispondono a collezioni di 3 o 4 note. Questo ci permette di ipotizzare una filtrazione in cui le componenti connesse determinate dalle singole classi d'altezza si uniranno molto prima rispetto a quelle dei brani del Trecento e si comporteranno in maniera simile al repertorio di Palestrina.
Inoltre la tonica assume un peso maggiore rispetto alle composizioni viste precedentemente in quanto svolge un ruolo fondamentale nel processo contrappuntistico dei corali. La principale caratteristica della musica tonale è proprio questo fatto, cioè la forte spinta verso la risoluzione cadenzale[4] Possiamo visualizzare qualche diagramma di $0-$persistenza corrispondente ai corali di Bach in figura 3.4. Tutti i diagrammi mostrano una cornerline che nasce tra 0.2 e 0.4, che testimonia il carattere tonale delle composizioni, mentre i cornerpoints che non hanno "vita breve" sono pochi.

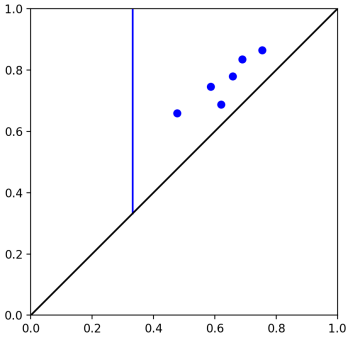
[4]La musica tonale si impose in Occidente tra l'inizio del XVII secolo e la fine del XIX e si riferisce alle composizioni organizzate attorno a un suono centrale o tonica. Bach fu il massimo rappresentante di questa evoluzione.



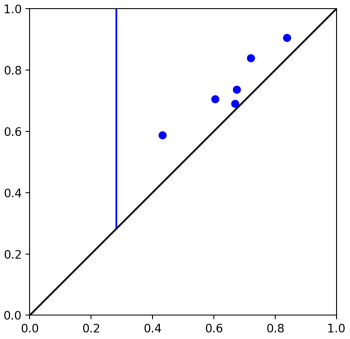
(a) Agnus 00



(b) Agnus 01

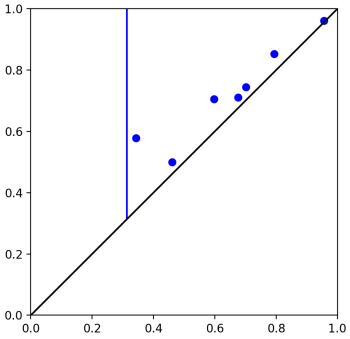


(c) Agnus 02

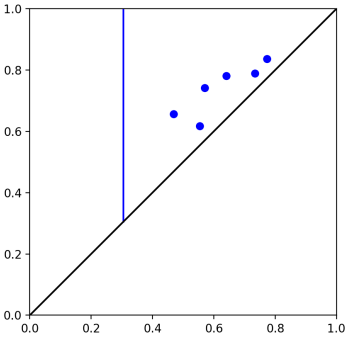


(d) Agnus 03

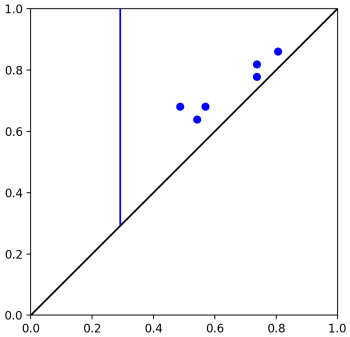
Figura 3.3: Diagrammi di 0-persistenza dal corpus di Palestrina di Music21.



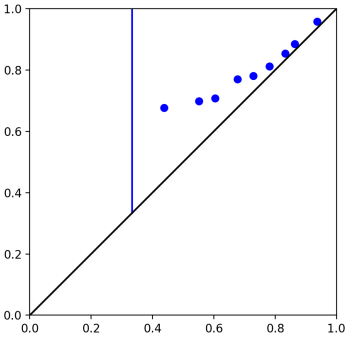
(a) Bwv 70.7



(b) Bwv 78.7



(c) Bwv 84.5



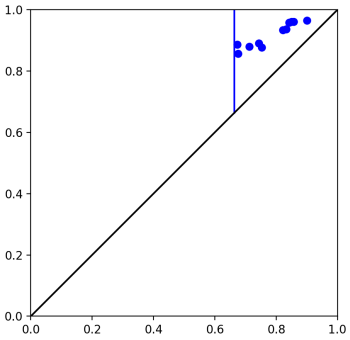
(d) Bwv 90.5

Figura 3.4: Diagrammi di 0-persistenza dal corpus di Bach di Music21.

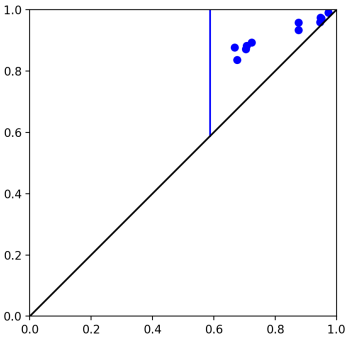
Musica post-tonale

Per musica post-tonale si intende quel ramo di produzione musicale sviluppatosi a partire dalla fine del XIX secolo che non segue le convenzioni tradizionali dell'armonia tonale. Ciò non significa che questa musica non graviti attorno ad una tonica centrale, ma che abbia un' attrazione da quest'ultima diversa rispetto alla musica tonale. Riportiamo un passo da *The Technique of my musical language* di Olivier Messiaen che mette in luce il comportamento cadenzale nella musica post tonale [34]:
"With the advent of Claude Debussy, one spoke of appoggiaturas without resolution, of passing notes with no issue, etc. In fact, one found them in his first works. In Pelléas et Mélisande, les Estampes, les Préludes, les Images for the piano, it is a question of foreign notes, with neither preparation nor resolution, without particular expressive accent, which tranquilly make a part of the chord, changing its color, giving it a spice, a new perfume. These notes keep a character of intrusion, of supplement: the bee in the flower! They have, nevertheless, a certain citizenship in the chord, either because they have the same sonority as some classified appoggiatura, or because they issue from the resonance of the fundamental. They are added notes."

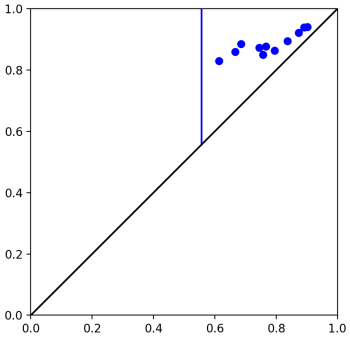
All'interno del corpus di Music21 non ci sono composizioni post-tonali da analizzare, a parte due pezzi per pianoforte dall'opera 19 di Schoenberg. Abbiamo comunque ritenuto opportuno analizzare dei file MIDI di alcune composizioni di Debussy e Ravel per estrarre i caratteri stilistici dai diagrammi di persistenza. In figura 3.5 possiamo osservare i diagrammi corrispondenti alle composizioni: "Jeux d'eau" di Ravel (1901), l'Arabesque n.1 di Debussy (1891) e "Menuet" e "Passepied" dalle Suite Bergamasche dello stesso autore (1905). Si può notare come la prima componente connessa nasca piuttosto tardi in tutte le composizioni. Questo fatto non ci sorprende, anzi mette in luce quanto il valore della nascita della cornerline sia un elemento per riconoscere il carattere stilistico che rappresenta l'intensità della spinta verso la risoluzione cadenzale. Il numero dei cornerpoints è maggiore rispetto a quello nei grafici precedenti, inoltre si può notare che un numero significativo di questi si trovi lontano dalla diagonale e vicino alla cornerline.



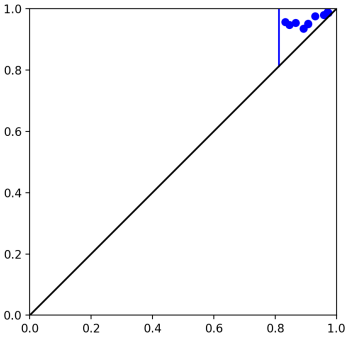
(a) Ravel - Jeux d'eau



(b) Debussy - Arabesque n.1



(c) Debussy - Menuet



(d) Debussy - Passepied

Figura 3.5: Diagrammi di 0-persistenza di alcune composizioni post-tonali.

3.1.2 Diagrammi di 1-persistenza

Se i diagrammi di 0−persistenza danno informazioni riguardanti l'attrazione tonale come spinta ad una risoluzione della cadenza attraverso il calcolo delle componenti connesse, lo studio degli 1−diagrammi di persistenza risulta meno intuitivo rispetto al precedente. Andremo a visualizzare i diagrammi rispettivi agli stessi brani analizzati in dimensione 0 e confronteremo i risultati in modo da mettere in luce i caratteri stilistici delle composizioni. Una considerazione che possiamo fare sin da subito è che più le composizioni sono ricche di diverse progressioni armoniche a più voci e più i diagrammi corrispondenti saranno ricchi di classi di omologia che non nascono in prossimità della fine della filtrazione.
Nella nostra rappresentazione del brano musicale come complesso simpliciale, un 1−ciclo corrisponde ad una successione di accordi che forma un "buco". In alcuni casi, più precisamente nel caso della musica del Trecento, vedremo che alcuni diagrammi avranno un numero diverso di cornerlines tra loro. Questo fatto si traduce in un problema più sottile da risolvere. Infatti, due grafici che hanno un numero diverso di cornerlines corrispondono a due spazi non omeomorfi e non ha molto senso definire una distanza come la bottleneck che confronti i diagrammi di persistenza se non esiste un omeomorfismo tra gli spazi. Per bypassare questo problema abbiamo pensato di trasformare le cornerlines in cornerpoints con ordinata uguale al valore massimo della funzione filtrante.

Ryan's Mammoth Collection

I diagrammi di 1−persistenza per i brani della Ryan's Mammoth Collection non hanno punti. Infatti, come abbiamo visto dai diagrammi di 0−persistenza, i brani sono realizzabili in maniera simpliciale come un insieme finito di punti in cui non esistono 1−cicli, quindi il diagramma di 1−persistenza non ha nessuna classe che nasce o muore.

Musica del Trecento

Possiamo notare che in tutti i diagrammi in 3.6 e 3.7 le classi di 1−omologia nascono molto tardi, in prossimità della fine della filtrazione. Inoltre si può osservare un fatto: i primi due diagrammi corrispondenti a Pmfc 01 e Pmfc 04 hanno cornerpoints

mentre gli altri due corrispondenti a Pmfc 13.01 e Pmfc 12.21 hanno solo cornerlines. Questo fatto è motivato dal tipo di composizione analizzata. Infatti i primi due corrispondono a madrigali a tre voci, il che significa che all'interno della composizione ci sono momenti di polifonia a tre voci che "tappano" i buchi formati nel processo di filtrazione. Le altre due composizioni sono madrigali a due voci che sono associate a complessi simpliciali in cui gli 1−cicli non muoiono mai, perché per uccidere un ciclo bisogna "riempirlo" con una catena che ha come bordo il ciclo stesso.

Messe di Palestrina

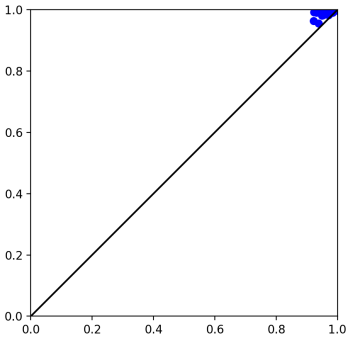
Le classi di 1-omologia corrispondenti alle composizioni analizzate nascono intorno al livello 0.8 della filtrazione. Nei diagrammi in figura 3.8 possiamo notare circa 10 cornerpoints, di cui molti distruibuiti in prossimità della diagonale e alcuni con un valore dell'ordinata vicino al valore massimo.

Corali di Bach

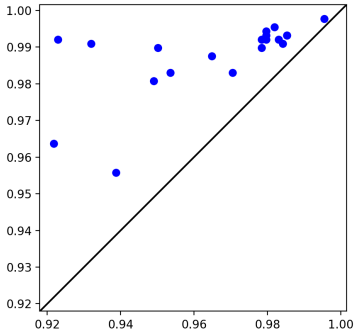
Quello che ci si può aspettare dai corali a quattro voci di Bach è che alcune classi di 1−omologia corrispondenti agli accordi più suonati nascano molto prima di altre. Possiamo visualizzare i diagrammi in figura 3.9. Le classi che si trovano a ridosso della diagonale sono le classi che hanno un periodo tra la nascita e la morte molto breve. Questo caso particolare è una conseguenza del fatto che gli 1-cicli nascono da dei sotto-accordi degli accordi più grandi di tre o quattro note che "tappano" in un tempo breve il ciclo generato. Le classi più interessanti da studiare sono quelle rappresentate dai punti più distanti dalla diagonale che sono dei cicli che muoiono in tempi più lunghi.

Musica Post-Tonale

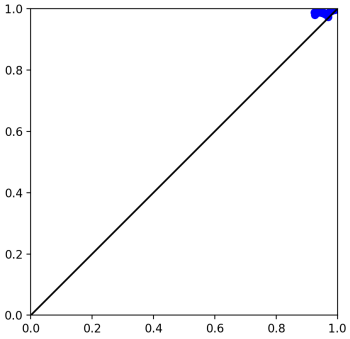
Come nella 0−persistenza, la classi di 1−omologia associate ai brani del corpus post-tonale nascono e muoiono a ridosso della fine della filtrazione. Possiamo notare che il numero di cornerpoints è più alto rispetto a quello negli altri esempi e si aggira tra i 40 e 50 punti. Riportiamo nelle figure 3.10 e 3.11 i diagrammi di 1−persistenza.



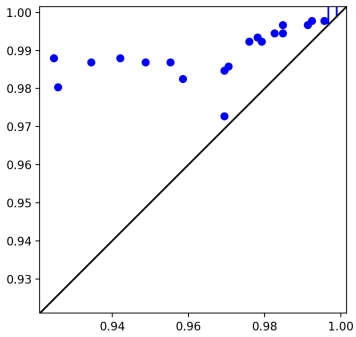
(a) Pmfc 01 - Virtutibus Laudabilis



(b) Pmfc 01 - zoom

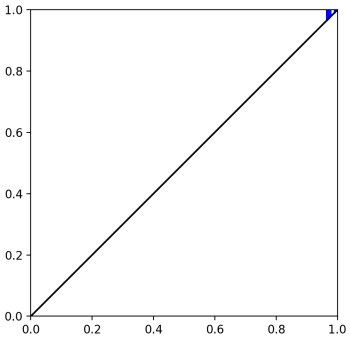


(c) Pmfc 04 - Si Dolce Non Sono

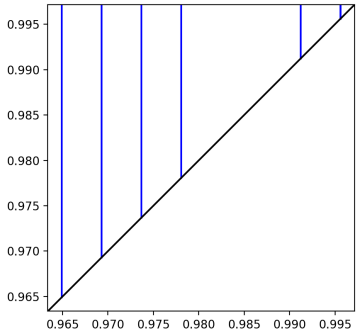


(d) Pmfc 04 - zoom

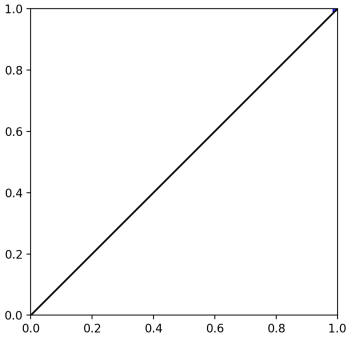
Figura 3.6: Diagrammi di 1-persistenza dal corpus di musica del Trecento (a sinistra) e zoom (a destra). Pmfc 01 e Pmfc 04 sono madrigali a tre voci.



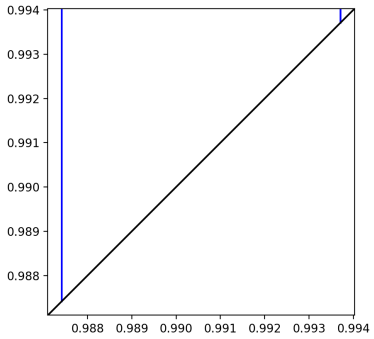
(a) Pmfc 13.01 Kyrie Summe Clementissime



(b) Pmfc 13.01 - zoom

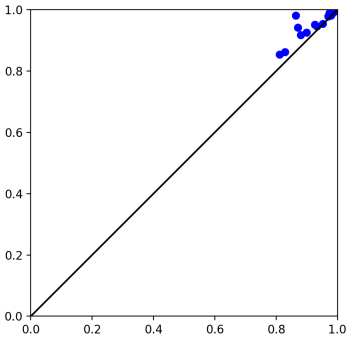


(c) Pmfc 12.21 - Benedicamus

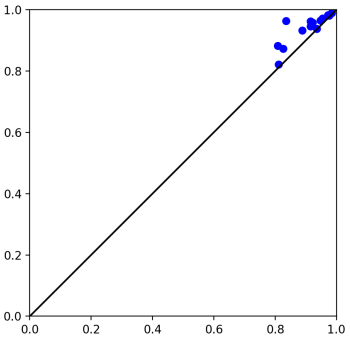


(d) Pmfc 12.21 - zoom

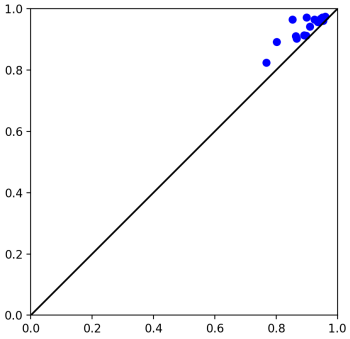
Figura 3.7: Diagrammi di 1-persistenza dal corpus di musica del Trecento (a sinistra) e zoom (a destra). Pmfc 13.01 e Pmfc 12.21 sono madrigali a due voci.



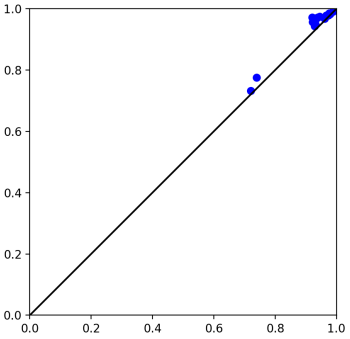
(a) Agnus 00



(b) Agnus 01

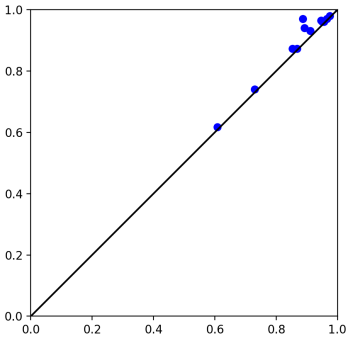


(c) Agnus 02

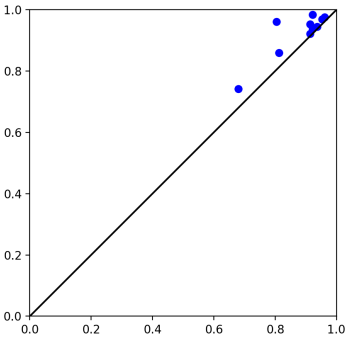


(d) Agnus 03

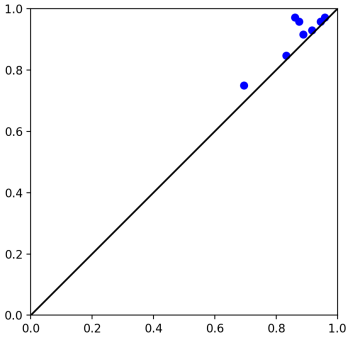
Figura 3.8: Diagrammi di 1-persistenza dal corpus di Palestrina di Music21.



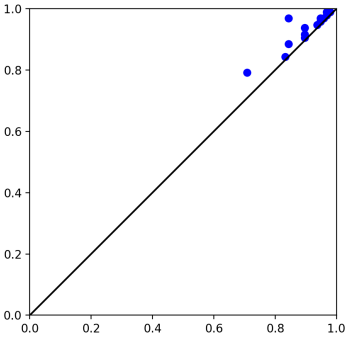
(a) Bwv 70.7



(b) Bwv 78.7

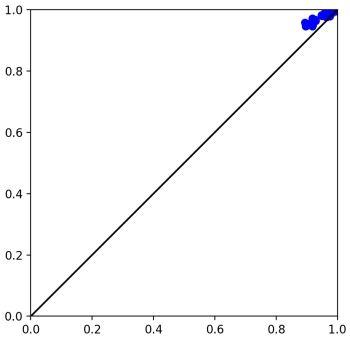


(c) Bwv 84.5

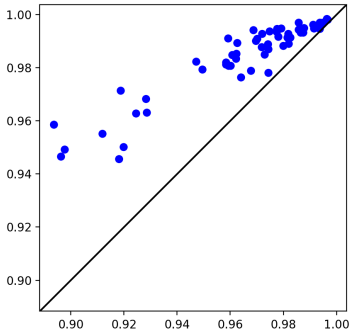


(d) Bwv 90.5

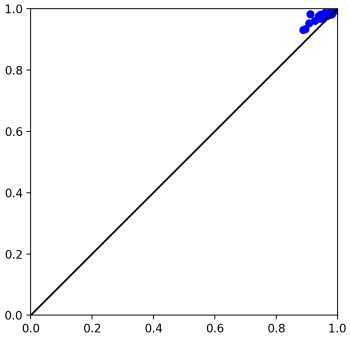
Figura 3.9: Diagrammi di 1-persistenza dal corpus di Bach di Music21



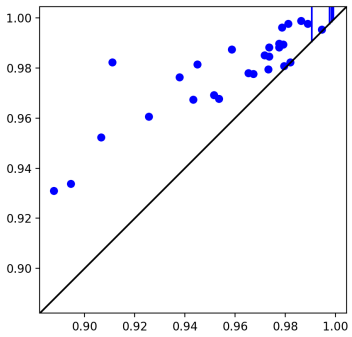
(a) Ravel - Jeux d'eau



(b) Ravel - Jeux d'eau (zoom)

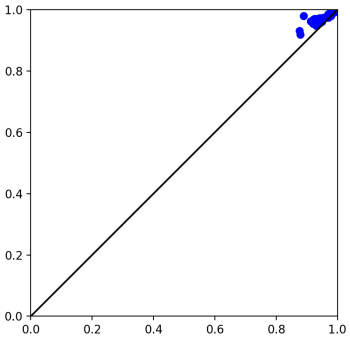


(c) Debussy - Arabesque n.1

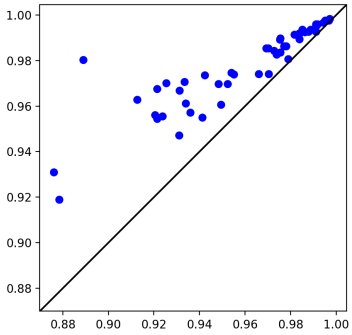


(d) Debussy - Arabesque n.1 (zoom)

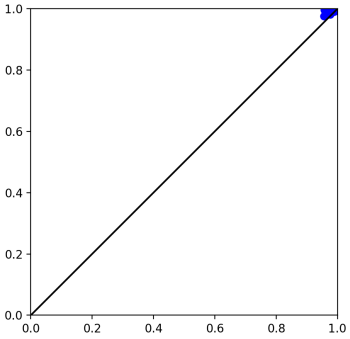
Figura 3.10: Diagrammi di 1-persistenza delle composizioni post-tonali (a sinistra) e zoom (a destra).



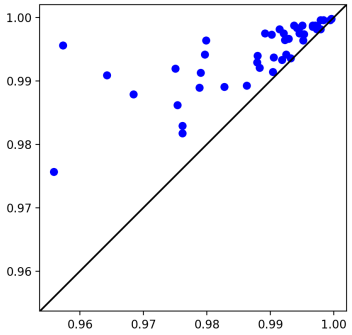
(a) Debussy - Menuet



(b) Debussy - Menuet (zoom)



(c) Debussy - Passepied



(d) Debussy - Passepied (zoom)

Figura 3.11: Diagrammi di 1-persistenza delle composizioni post-tonali (a sinistra) e zoom (a destra)

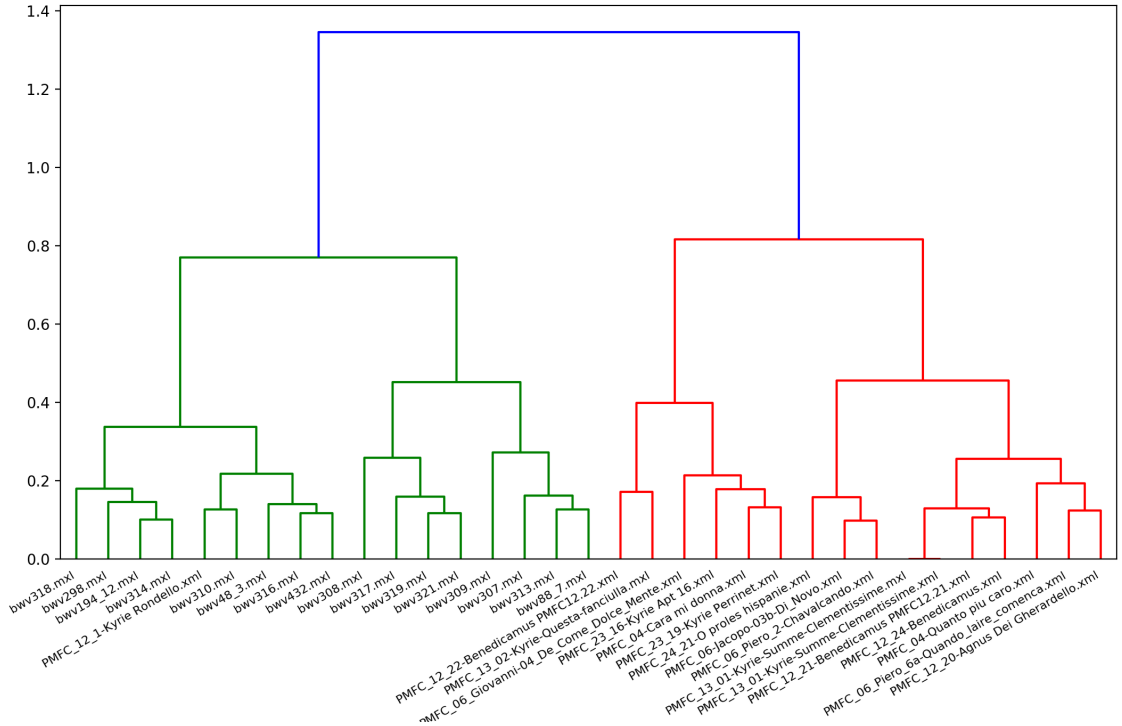
3.2 Clustering gerarchico

In questo capitolo abbiamo presentato i diagrammi di persistenza di alcune composizioni che si possono confrontare tramite una distanza, nel nostro caso la distanza presa in considerazione è la bottleneck definita in 2.2.
Pensiamo ora ad un insieme di n composizioni come una nuvola di n punti in uno spazio metrico in cui la distanza tra due punti è la bottleneck dei rispettivi diagrammi di persistenza. In questo modo possiamo descrivere la disposizione dei punti attraverso l'analisi di clustering gerarchico [39]. Questa analisi ci dà una rappresentazione di tutte le possibili disposizioni dei punti sotto forma di dendrogramma.
Il dendrogramma è un diagramma ad albero che fornisce una rappresentazione intuitiva del clustering gerarchico di dati (maggiori informazioni sono reperibili in [40]). Ogni cluster è visualizzato attraverso unioni tra clusters più piccoli. La linea orizzontale rappresenta l'unione tra due clusters, mentre la lunghezza delle linee verticali dell'unione rappresenta la distanza tra i clusters più piccoli. I colori rappresentano i clusters che si formano all'interno dello spazio metrico.
In base al metodo di calcolo della distanza tra un cluster e ogni elemento possiamo ottenere configurazioni diverse[5].
Riportiamo in seguito un esempio di due dendrogrammi utili per fare qualche considerazione sul metodo di classificazione utilizzato in questo elaborato.

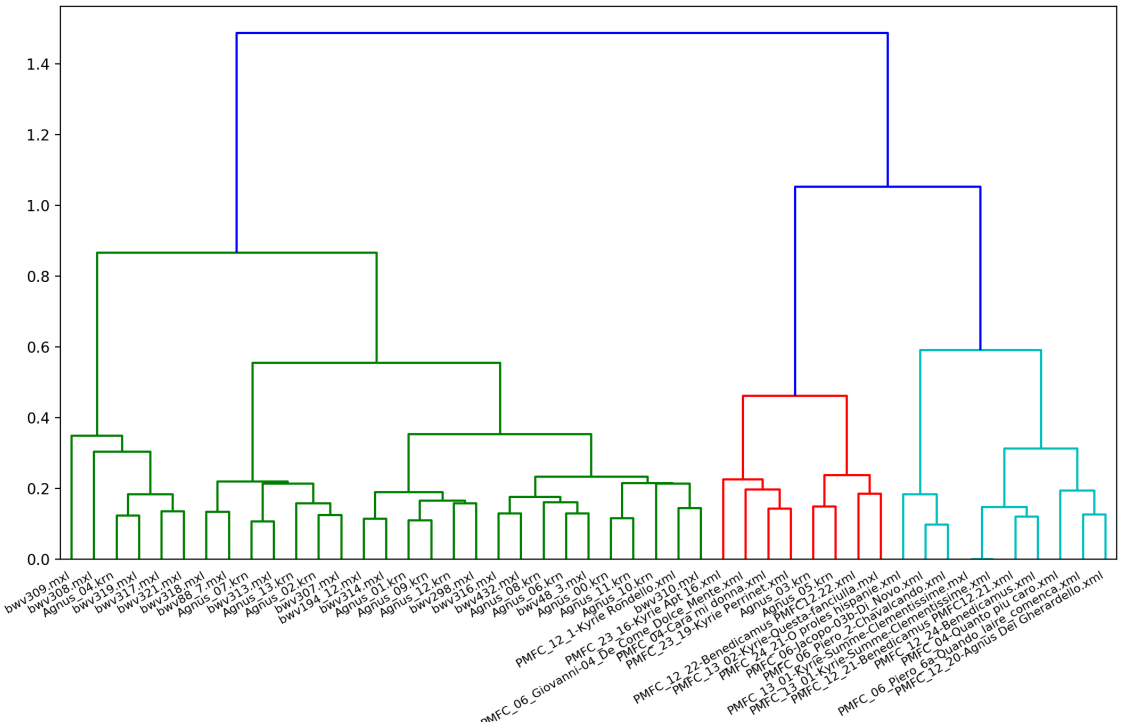
Nel primo dendrogramma 3.12a abbiamo utilizzato un dataset di 32 composizioni divise in maniera equa tra corali di Bach e composizioni del trecento. Applicando la bottleneck su tutti i diagrammi del corpus otteniamo una divisione in due cluster più o meno coerente con i dati analizzati (l'unica composizione che si trova nel cluster sbagliato è Kyrie Rondello).
Andando ad inserire nel dataset alcune composizioni di Palestrina, ci troviamo con una configurazione del cluster 3.12b lontana dalle aspettative in quanto le composizioni bachiane e quelle di Palestrina risultano essere nello stesso cluster. Questo fatto mostra due limiti del metodo utilizzato fino ad ora: la distanza bottleneck non risulta essere la scelta migliore per una configurazione del genere in quanto scarta molte informazioni riguardanti i cornerpoints che sono utili per il riconoscimento

[5]I metodi disponibili nel pacchetto Scipy di Python per il linkage sono i seguenti: single, complete, average, weighted, centroid, median e ward e sono tutti descritti in [40]

delle caratteristiche stilistiche delle composizioni. L'altro limite è dato dal fatto che il metodo descritto finora non considera l'aspetto orizzontale della musica, cioè l'evoluzione di una composizione nel tempo. Nel prossimo capitolo ridefiniremo il metodo considerando le varie configurazioni dello spazio rispetto al tempo.



(a) Cluster Bach - Trecento



(b) Cluster Bach - Palestrina - Trecento

Capitolo 4

Persistenza su serie temporali

Consideriamo ora l'aspetto orizzontale della composizione, cioè l'evoluzione temporale della musica analizzando le varie configurazioni dello spazio rispetto al tempo. Intuitivamente, data una funzione continua $f : X \times [0,1] \to \mathbb{R}$ è possibile rappresentare la sua evoluzione su $[0,1]$ con una collezione di cammini. Questa collezione viene chiamata *vineyard* e ogni cammino è detto *vine*, rimandiamo a [36] per approfondimenti.
In questo elaborato utilizzeremo un metodo diverso rispetto a quello dei vineyards, già sviluppato in [1]. Sia X spazio topologico, $f : X \times [0,T] \to \mathbb{R}$ una funzione lineare a tratti e $t = \{t_0, ..., t_n\}$ una partizione di $[0,T]$ diviso equamente in n parti. Per ogni istante t_i posso definire un diagramma di $k-$persistenza $D_{f_i,k}$. La collezione di questi *snapshot di k-persistenza* è una *serie di k-persistenza* $D_{f,n} = \{D_{f_i,k}\}_{i=0}^{n}$.
Siano $t_n = \{t_0, ..., t_n\}$ e $t_m = \{t_0, ..., t_m\}$ due partizioni di $[0, T_f]$ e $[0, T_g]$. Siano:

$$f : X \times [0, T_f] \to \mathbb{R}$$

$$g : X \times [0, T_g] \to \mathbb{R}$$

due funzioni tali che f_{ti} e g_{tj} sono funzioni con un numero finito di valori critici in X per ogni $i \in \{1, ..., n\}$ e $j \in \{1, ..., m\}$. Esiste un algoritmo, chiamato Dynamic Time Warping (DTW), che ci permette di confrontare due serie di lunghezze diverse.

4.1 Algoritmo DTW

L'algoritmo di Dynamic Time Warping (DTW) è un algoritmo introdotto negli anni 60 ed utilizzato in maniera estensiva a partire dagli anni 70 con applicazioni in ri-

conoscimento vocale, matching di scrittura, riconoscimento del linguaggio dei segni, allineamento di sequenze di proteine, clustering di serie temporali e molto altro. L'algoritmo è diventato popolare per via della sua efficienza computazionale. Date due serie temporali $X = (x_1, x_2, ..., x_n)$ e $Y = (y_1, y_2, ..., y_m)$ con $n, m \in \mathbb{N}$, l'algoritmo DTW trova una soluzione ottimale in un tempo $O(mn)$. Se le serie contengono valori dello spazio Φ allora la distanza tra le due serie è definita dalla funzione

$$d : \Phi \times \Phi \to \mathbb{R} \geq 0$$

Consideriamo questa funzione distanza come una funzione costo, considerando il problema dell' allineamento della serie un problema di minimizzazione della funzione costo.

L'algoritmo inizia con la costruzione della matrice dei costi $C \in \mathbb{R}^{m \times n}$ che rappresenta le distanze tra tutti gli elementi di X e Y. Una volta costruita la matrice, l'algoritmo trova il percorso di allineamento ottimale che passa per le aree con costi più bassi. Il percorso, chiamato *warping path* definisce la corrispondenza tra un elemento $x_i \in X$ e $y_j \in Y$, rispettando le condizioni al bordo che assegnano rispettivamente il primo e l'ultimo elemento di X al primo e l'ultimo di Y. In linguaggio formale, il cammino costruito attraverso il DTW è la sequenza di punti $p = (p_1, p_2, ..., p_K)$ con $p_l = (x_i, y_j) \in [1 : N] \times [1 : M]$ per $l \in [1 : K]$ tale che soddisfi le seguenti condizioni:

- **Condizioni al bordo**: $p_1 = (1, 1)$ e $p_K = (M, N)$. Il punto iniziale e quello finale del warping path devono corrispondere col rispettivo punto finale e iniziale delle sequenze allineate.

- **Condizione di monotonia**: Siano p_i e p_j due punti del warping path tali che $i \leq j$, allora $x_i \leq x_j$ e $y_i \leq y_j$.

- **Condizione di lunghezza del passo**: Il criterio limita il warping path ad un passo arbitrario. La condizione che verrà utilizzata in questo elaborato è quella di passo unitario, cioè $p_{l+1} - p_l \in \{(1, 1), (1, 0), (0, 1)\}$.

La funzione di costo associata al warping path calcolato rispetto alla matrice di costi locali (che rappresenta tutte le distanze tra la serie temporale di X e Y) sarà:

$$c_p(X, Y) = \sum_{l=1}^{K} c(x_{n_l}, y_{n_l})$$

Il warping path di costo minimo si chiamerà *warping path ottimale* ed è definito come:

$$DTW(X,Y) := min\{c_p(X,Y)\}$$

Osservazione 1. Il cammino ottimale esiste sempre perché l'insieme dei percorsi è finito.

Dalla definizione, per trovare un percorso ottimale si dovrebbero testare tutti i possibili percorsi tra X e Y. Questo approccio si mostra da subito sconveniente a livello computazionale vista la crescita esponenziale del numero dei percorsi rispetto alla crescita lineare delle lunghezze di X e Y.
Per risolvere questo problema, il DTW utilizza una parte di programmazione dinamica con complessità di $O(MN)$. Infatti si costruisce la matrice dei costi accumulati o matrice dei costi totali D definita nel seguente modo:

1 Prima riga: $D(1,j) = \sum_{k=1}^{j} c(x_1, y_k), j \in (1, ..., M)$

2 Prima colonna: $D(i,1) = \sum_{k=1}^{i} c(x_i, y_1), i \in (1, ..., N)$

3 Altri elementi: $D(i,j) = min\{D(i-1, j-1), D(i-1, j), D(i, j-1)\} + c(x_i, y_j), i \in (1, ..., N), j \in (1, ..., M)$.

Il costo computazionale della costruzione della matrice è $O(NM)$. Una volta costruita, è facile trovare il warping path partendo dall'ultimo punto $p_{end} = (M, N)$ fino al primo $p_{start} = (1, 1)$ seguendo la strategia descritta nell'algoritmo *Optimal-WarpingPath*.
Un parametro ulteriore che abbiamo pensato di aggiungere nel calcolo delle matrici di costi è lo *slidingWindow*. Questo parametro ci permette di scegliere la porzione di brano s che vogliamo analizzare: quando $s = 0$, la filtrazione si applica su tutto il brano dall'inizio al punto t_i, quando $s > 0$ la filtrazione si applica dal punto $t_i - s$ a t_i.

Algorithm 1 OptimalWarpingPath

```
1: path[] ← new array
2: i ← rows(dtw)
3: j ← columns(dtw)
4: while (i > 1&j > 1) do :
5:     if i == 1 then
6:         j = j − 1
7:     else if j == 1 then
8:         i = i − 1
9:     else
10:        if dtw(i−1, j) == min{dtw(i−1, j), dtw(i, j−1), dtw(i−1, j−1)} then
11:            i = i − 1
12:        else if dtw(i, j−1) == min{dtw(i−1, j), dtw(i, j−1), dtw(i−1, j−1)}
   then
13:            j = j − 1
14:        else i = i − 1; j = j − 1
15:    path.add((i, j))
   return path
```

4.1.1 Qualche risultato

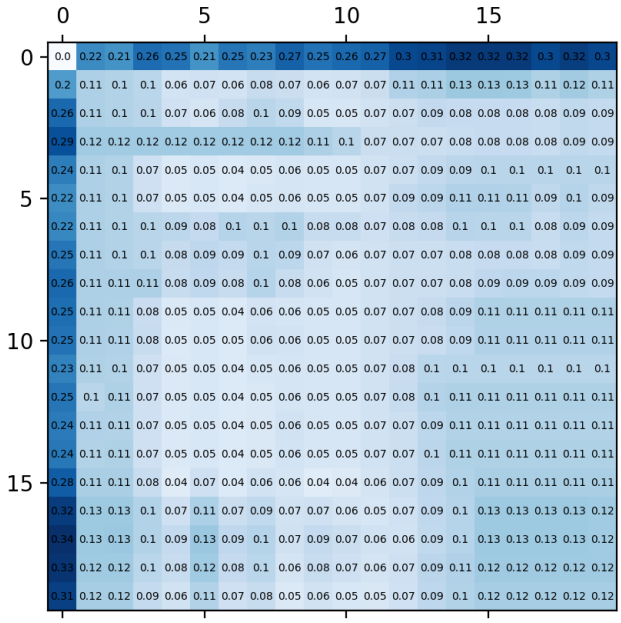
Mostriamo le matrici di costi accumulati e parziali calcolati tramite bottleneck tra i diagrammi di $0-$persistenza degli snapshots di due corali di Bach (figura 4.1) e tra una corale di Bach e l'Arabesque di Debussy (figura 4.2). Notiamo che, pur sviluppandosi in diagonale in entrambi i casi, il warping path ottimale vale 1.7 nel primo caso e 6.7 nel secondo.

Possiamo utilizzare questo risultato per calcolare un nuovo dendrogramma per alcune composizioni. Prendendo in esame alcuni elementi che si confondevano in un unico cluster che comprendeva i corali di Bach e le messe di Palestrina, notiamo un leggero miglioramento della configurazione del diagramma 4.3 rispetto alla classificazione precedente che non teneva conto dello sviluppo temporale.

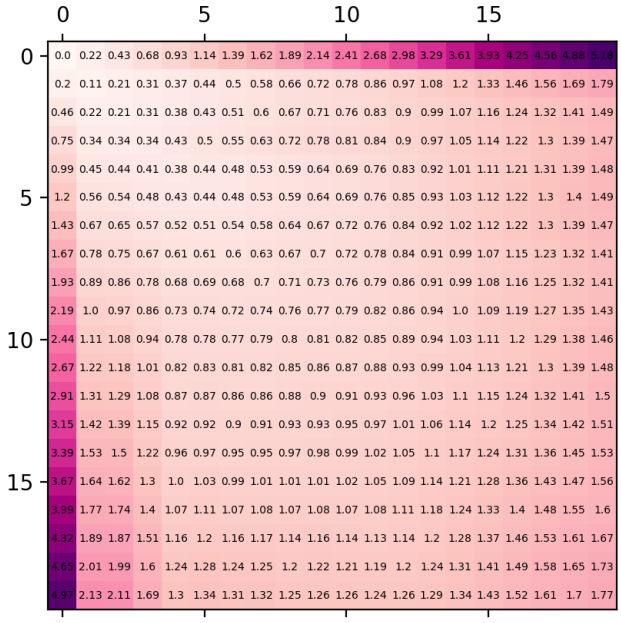
Un altro risultato interessante può essere il confronto di una composizione con sé stessa per l'analisi di caratteristiche che risulterebbero complicate a prima vista. Ad esempio, si possono analizzare delle composizioni per studiare l'evoluzione dell'uso della tonalità all'interno di queste. Mostriamo due matrici di costi parziali con diversi sliding windows in figura 4.4 corrispondenti al quartetto per archi op.74 di L.van Beethoven. Quest'opera è formata da 1114 battute e cambia tonalità sei volte. Possiamo visualizzare, soprattutto nella seconda matrice, che alcune righe e colonne[1] mostrano un costo elevato e alcune di queste corrispondono ai cambiamenti di tonalità presenti nella composizione.

Infine, uno ulteriore sviluppo del metodo potrebbe essere l'analisi della somiglianza per il riconoscimento dei plagi. Ad esempio in figura 4.5 mostriamo un esperimento fatto durante il tirocinio coi diagrammi di $0, 1$ e 2-persistenza per due versioni diverse dello stesso brano.

[1] La matrice in questo caso è simmetrica

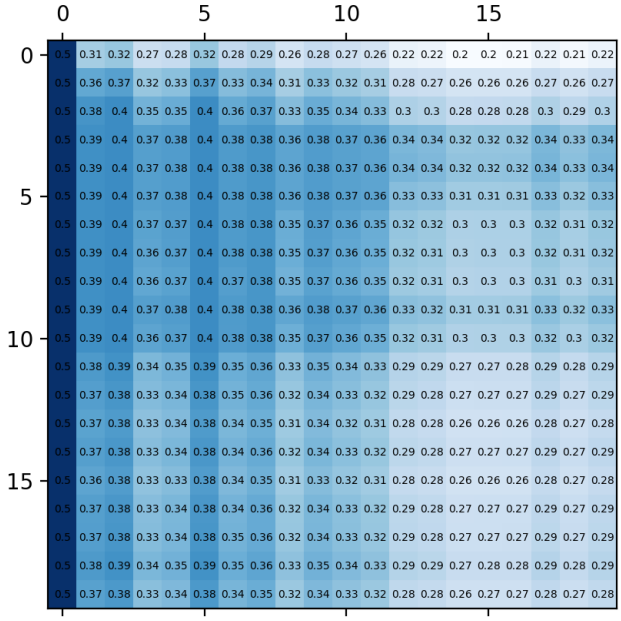


(a) Matrice dei costi parziali

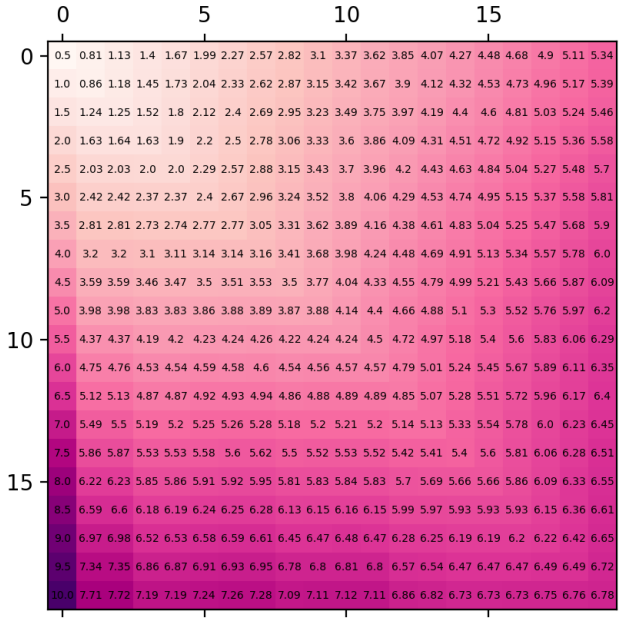


(b) Matrice dei costi accumulati

Figura 4.1: DTW tra i corali di Bach Bwv 70.7 e Bwv 78.7

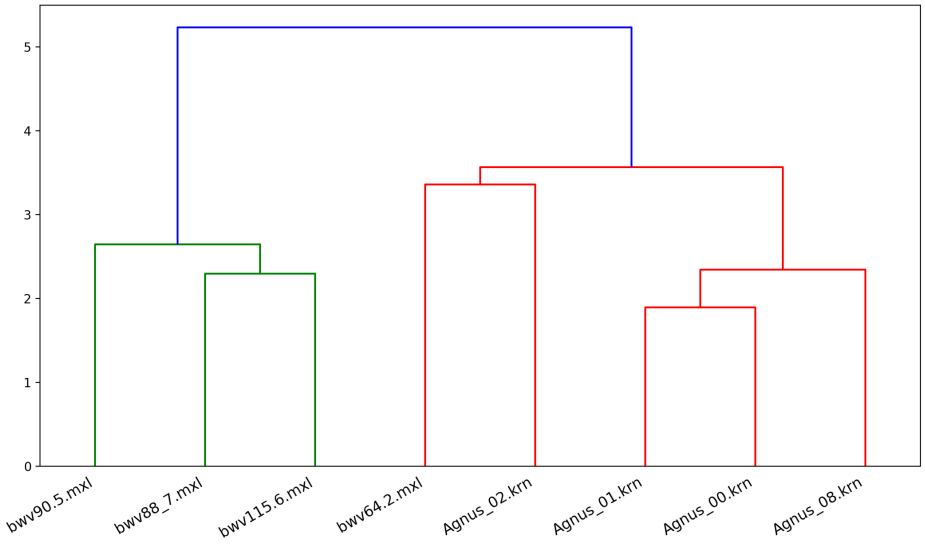


(a) Matrice dei costi parziali

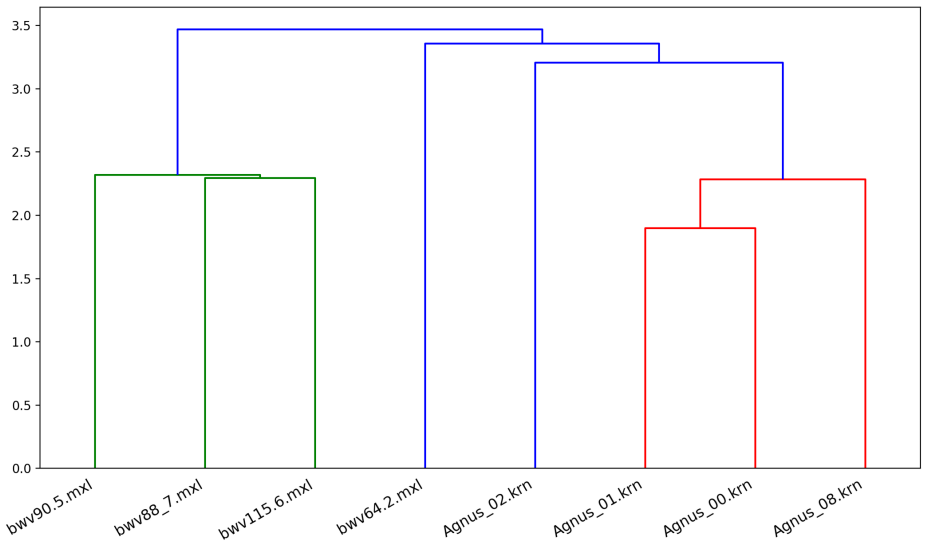


(b) Matrice dei costi accumulati

Figura 4.2: DTW tra il corale Bwv 78.7 e Arabesque no.1 di Debussy

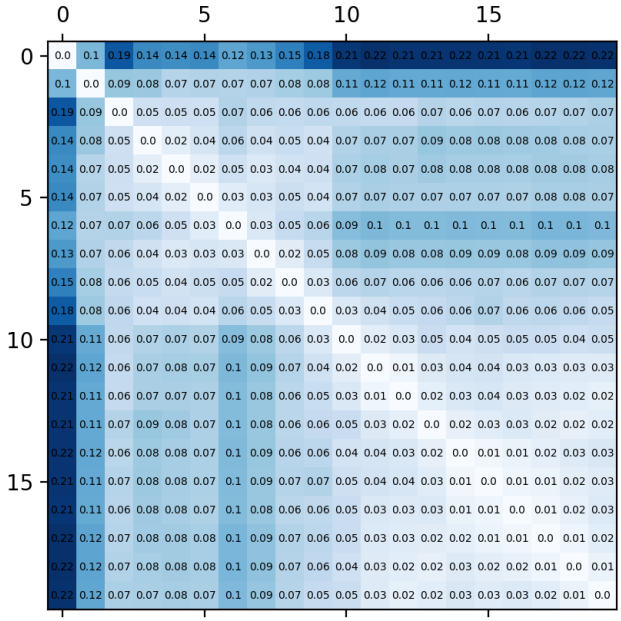


(a) Dendrogramma con distanza *average*

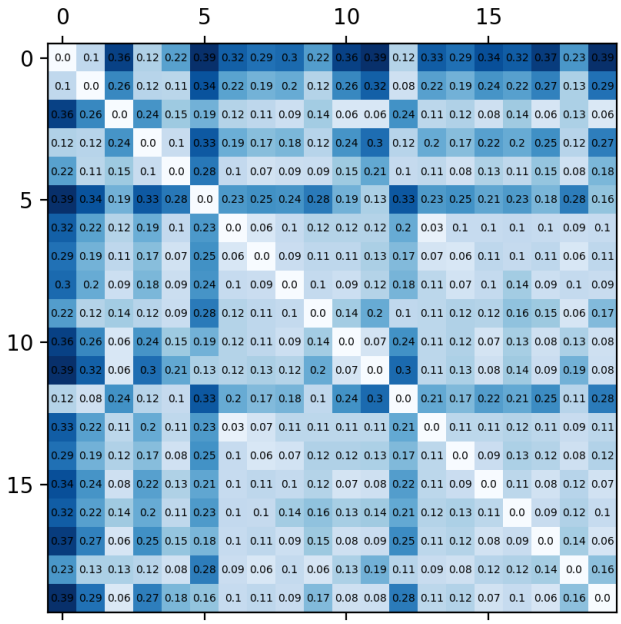


(b) Dendrogramma con distanza *single*

Figura 4.3

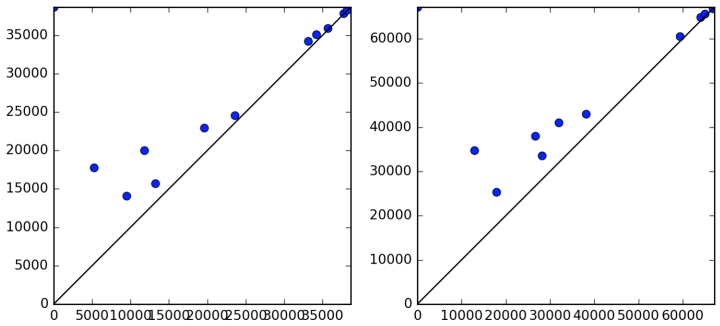


(a) Sliding window = 0

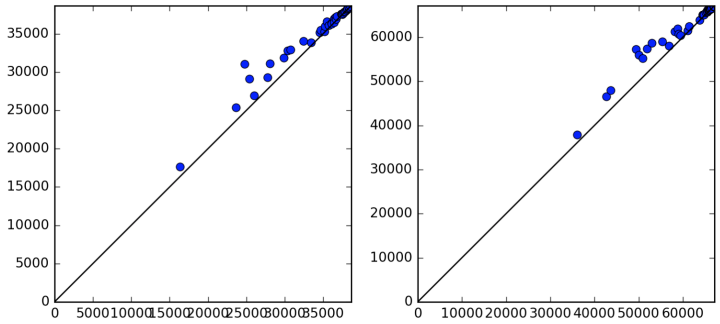


(b) Sliding window = 50

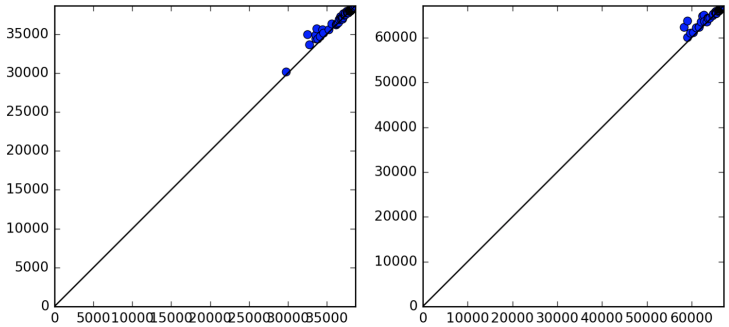
Figura 4.4: L.v.Beethoven - op.74. La scelta della sliding window può mettere a fuoco i momenti all'interno della composizione relativi a un cambio di tonalità



Diagrammi di 0-persistenza



Diagrammi di 1-persistenza



Diagrammi di 2-persistenza

Figura 4.5: Confronto dei diagrammi di persistenza di due diverse versioni del brano *"Impressioni di Settembre"*

Conclusioni

Con questo progetto di tesi abbiamo indagato una nuova configurazione per la rappresentazione simpliciale di un insieme di accordi partendo dai lavori di Bergomi [1] e Bigo [2]. Il motivo di questa scelta è dare un senso alla filtrazione per accordi di più note. Questo ci ha permesso di avere una rappresentazione dei diagrammi di persistenza più ricca rispetto alla struttura del Tonnetz di partenza. Applicando poi la persistenza per serie temporali e il confronto di queste attraverso il DTW, abbiamo raggiunto qualche risultato a livello classificatorio.
In particolare, abbiamo definito alcune caratteristiche dei diagrammi di persistenza corrispondenti a dei tratti stilistici presenti nelle composizioni.
D'altra parte, abbiamo notato più volte nel corso dell'elaborato i limiti del metodo presentato. Uno dei problemi principali risulta essere la scelta della distanza bottleneck definita sui diagrammi. Infatti il compromesso della bottleneck è quello di fornire uno strumento stabile al rumore, al costo di perdere molte informazioni fornite da tutti i cornerpoints dei diagrammi. Quindi, in una configurazione di questo tipo, dove il numero e la disposizione dei cornerpoints è rilevante per determinare le caratteristiche stilistiche delle composizioni, quella della bottleneck non risulta essere la scelta migliore. Si potrebbe tentare un approccio simile utilizzando la distanza di Wasserstein o attraverso la rappresentazione dei diagrammi come l'insieme di radici complesse di un polinomio [41] in modo da dare un peso a tutti i punti presenti nel diagramma.
Uno sviluppo futuro potrebbe riguardare l'estensione del lavoro all'analisi audio. Ad esempio, attraverso un cromagramma si possono estrarre informazioni sulle classi di altezza e la durata di esecuzione, oppure attraverso nuove funzioni filtranti che tengono conto di altri parametri come, ad esempio, i coefficienti spettrali Mel o MFCCs per l'analisi del timbro.

Appendices

Appendice A

Codice Python

Riportiamo in seguito il codice Python (versione 3.7) utilizzato per le analisi sul corpus musicale. In tabella A.1 sono riportati tutti i pacchetti Python utilizzati. Parte del codice è stato sviluppato in collaborazione con Corentin Guichaoua del gruppo SMIR.

Pacchetto	Versione
Dionysus	2.0.6
Music21	5.5.0
NumPy	1.15.4
SciPy	1.2.0
Matplotlib	3.0.2
Posixpath	Nativo

Tabella A.1: Elenco dei pacchetti Python usati

Dionysus fixes

```
"""
this is an edit of the Dionysus Package
"""
# -- ==imports== --
import dionysus as dio
```

```
# -- ==plot_size== --
def plot_size(diag,plotBounds=(None,None)):
    #"""Helper to set the plot's size automatically"""
    min_plot,max_plot = plotBounds
    if min_plot != None and max_plot != None:
        #We already have our bounds
        return plotBounds
    else:
        min_birth = min(p.birth for p in diag)
        cornerpoints = list(filter(lambda p: p.death !=
            float('inf'), diag))
        #Use maximum birth as fallback if there are only
            cornerlines
        max_death = max(p.death for p in cornerpoints) if
            cornerpoints else max(p.birth for p in diag)

        if min_birth != max_death: # Add 5% on both sides of
            the plot
            autoMin = min_birth - 0.05*(max_death-min_birth)
            autoMax = max_death + 0.05*(max_death-min_birth)
        else: # Zoom out a little
            autoMin = min_birth - abs(0.05*min_birth)
            autoMax = max_death + abs(0.05*max_death)

        return min_plot if min_plot != None else autoMin,
            max_plot if max_plot != None else autoMax


# -- ==plot_bar== --
def plot_bar(diag,min_plot=None,max_plot=None, show=True):
    import matplotlib.pyplot as plt
```

```
    bounds = plot_size(diag,(min_plot,max_plot))

    for (i,p) in enumerate(diag):
        #We'll also handle bars with no death. The line is
           not really infinite though, so we need to be
           careful if resizing the plot
        plt.plot((p.birth,p.death if p.death != float('inf')
            else bounds[1]),(i,i),'b')
    plt.xlim(bounds)
    plt.ylim(-0.5,len(diag))
    if show:
        plt.show()
    return plt


# -- ==plot_diagram== --
def plot_diagram(diag, min_plot=None,max_plot=None, plt =
   None, show = True, animate = False):
    if plt == None:
        import matplotlib.pyplot as plt

    plt.axes(aspect='equal')
    bounds = plot_size(diag,(min_plot,max_plot))

    for p in diag:
        if p.death == float('inf'): # p is a cornerline
            plt.plot([p.birth,p.birth], [p.birth, bounds
               [1]],'b')
        else: # p is a cornerpoint
            plt.plot(p.birth,p.death,'ob') # Cornerpoints

    plt.plot(bounds, bounds,'black') # Diagonal
```

```
    ## clip the view
    plt.xlim(bounds)
    plt.ylim(bounds)

    if show:
        plt.show()
    return plt.gcf()


# -- ==uninfinite_diagram== --
def uninfinite_diagram(diagram,big_number=1e21):
    # Not using sys.float_info.max because it turns back
       into infinity when turned into a diagram
    return dio.Diagram([(d.birth, d.death if d.death !=
       float('inf') else big_number) for d in diagram])


# -- ==bottleneck== --
def bottleneck_distance(diag1,diag2,delta=0):
    return dio.bottleneck_distance(uninfinite_diagram(diag1
       ,1.0),uninfinite_diagram(diag2,1.0),delta)


# -- ==mini== --
def mini(x, y, z):
    if (x < y):
        return x if (x < z) else z
    else:
        return y if (y < z) else z
```

Music21 Parser

```
"""
    authors: Corentin Guichaoua and Matteo Pennesi
```

```
"""


# -- ==imports== --
import music21
import dionysus as dio
import sys
sys.path.insert(0, '../dionysus')
import dionysus_fixes as dio_fixes
import numpy as np
import math
from matplotlib import pyplot as plt


# -- ==loaders== --
def load_from_corpus(identifier):
    return music21.corpus.parse(identifier)

def load_from_file(filename):
    return music21.converter.parse(filename)


# -- ==preprocessing== --
def preprocess(piece, warn=True):
    if warn : print('Preprocessing the piece. This could
       take some time, please be patient.')
    return piece.chordify().flat


# -- ==make_filtration== --
def make_filtration(piece, start=0, end=None):
    # Count occurrence frequencies
    filterLevels = dict()
```

```
    if end==None :
        end = piece.quarterLength
    for chord in piece.getElementsByOffset(start,end,
       classList=[music21.chord.Chord]):
        for subchord in nonEmptyPowerset(chord.
           orderedPitchClasses):
            # How much of the chord is actually within the
               analysis scope
            scopeLength = min(end, chord.offset+chord.
               quarterLength) - max(start,chord.offset)
            if subchord in filterLevels:
                # -= since we are counting the complement to
                    the total length
                filterLevels[subchord]-= scopeLength / (end-
                   start)
            else:
                # substract from the total duration
                filterLevels[subchord]= 1 - scopeLength / (
                   end-start)
    # Build a filtration with these levels
    filtration = dio.Filtration([(simplex,filterLevels[
       simplex])
                                 for simplex in filterLevels
                                    ])
    filtration.sort()
    print(*filtration, sep = ", ")
    return filtration


# -- ==show_diagrams== --
def show_diagrams(filtration, start=None, end=None):
    # Compute the homology
    h = dio.homology_persistence(filtration)
```

```
    diagrams = dio.init_diagrams(h,filtration)
    # Display the results
    for i,diag in enumerate(diagrams):
        print(i)
        print(list(diag))
        # Zoomed out version
        if start!=None or end != None : dio_fixes.
            plot_diagram(diag, start, end, show = True)
        # Zoomed in version
        dio_fixes.plot_diagram(diag, show = True)
        dio_fixes.plot_bar(diag)
    return diagrams


# -- ==return_diagram== --
def return_diagram(filtration, degree, start=None, end=None)
    :
    from pathlib import Path
    # Compute the homology
    h = dio.homology_persistence(filtration)
    diagrams = dio.init_diagrams(h,filtration)
    diag = diagrams[degree]
    #Display the results
    print(list(diag))
    print (diag)
    dio_fixes.plot_diagram(diag, start, end, show = True) #
        Plot diagram
    dio_fixes.plot_diagram(diag, show= True) #Plot zoomed
        diagram
    return diag
# -- ==return_diagrams== --

# -- ==homology_analysis== --
```

```
def homology_analysis(path, degree, source='corpus', start =
    0, end = None, exclude = 'ask'):
    if source == 'corpus':
        piece = load_from_corpus(path)
    elif source == 'file' :
            piece = load_from_file(path)
    else:
        raise 'Unknown source: '+source
    piece =preprocess(piece)
    if end == None : end = piece.duration.quarterLength
    filtration = make_filtration(piece, start, end)
    print('Risultati del brano:', path)
    return return_diagram(filtration, degree, 0, 1)


# -- ==corpus_distance== --
def corpus_distance(path, degree, source='corpus', start =
   0, end = None):
    pieces = music21.corpus.getWork(path)
    diagram = []
    distance = []
    i=0
    j=0
    while i < len(pieces):
        item = pieces[i]
        diagram.append(homology_analysis(item,degree,exclude
            = ' '))
        i = i+1
    i=0
    while i < len(pieces):
        item1 = diagram[i]
        while j < len(pieces):
            item2 = diagram [j]
```

```
                distance.append(dio_fixes.bottleneck_distance(
                    item1,item2))    #Calculate the bottleneck
                    distance
                j = j+1
            j = 0
            i = i+1
        return distance


# -- ==powersets== --
import itertools as iter
def powerset(iterable):
    s = list(iterable)
    return iter.chain.from_iterable(iter.combinations(s, r)
        for r in range(len(s)+1))

def nonEmptyPowerset(iterable):
    power = powerset(iterable)
    next(power)
    return power


# -- ==homologyTimeSeries== --
def homology_time_series(path, degree, start=0, end=None,
    datapoints=20, slidingWindow=0):
    # If end is not specified, we run on the whole piece
    piece = load_from_corpus(path)
    piece = preprocess(piece)
    if end == None : end = piece.duration.quarterLength
    result = []
    for i in np.linspace(start, end, datapoints+1):
        if start == i : continue
        filtration = make_filtration(piece,(start if
```

```
            slidingWindow==0 else max(start,i-slidingWindow))
            ,i)
        h = dio.homology_persistence(filtration)
        diagrams = dio.init_diagrams(h,filtration)
        result.append((diagrams[degree]))
    return result


# -- ==dynamic_warping== --
def dynamic_warping(path1, path2, dim):
    fig, ax = plt.subplots()
    piece1 = homology_time_series(path1,dim)
    piece2 = homology_time_series(path2,dim)
    distance = []
    i=0
    j=0
    while i < len(piece1):
        item1 = piece1[i]
        while j < len(piece2):
            item2 = piece2[j]
            distance.append(dio_fixes.bottleneck_distance(
                item1,item2))
            j=j+1
        j=0
        i=i+1
    intersection_matrix = np.array(distance)
    intersection_matrix.resize(int(len(piece2)),int(len(
        piece1)))
    min_val, max_val = 0, 1,
    ax.matshow(intersection_matrix, cmap=plt.cm.Blues)
    for i in range(len(piece1)):
        for j in range(len(piece2)):
            c = intersection_matrix[j,i]
```

```
            cr = round(c,2)
            ax.text(i, j, str(cr), va='center', ha='center',
                fontsize = 5)
    plt.show()
    dtw = accumulated_cost(intersection_matrix,len(
        intersection_matrix),len(intersection_matrix[0]))
    return dijkstra(intersection_matrix,len(
        intersection_matrix)-1,len(intersection_matrix[0])-1,
        dtw)


# -- ==accumulated_cost== --
def accumulated_cost(matrix,m,n):
    dtw = np.zeros((m,n))
    dtw[0,0]=0
    for i in range(n):
        dtw[i,0]=dtw[i-1,0]+matrix[i,0]
    for j in range(m):
        dtw[0,j] = dtw[0,j-1]+matrix[0,j]
    i=1
    j=1
    while i<n:
        while j<m:
            dtw[i,j]=matrix[i,j]+dio_fixes.mini(dtw[i-1,j],
                dtw[i,j-1],dtw[i-1,j-1])
            j=j+1
        j=1
        i=i+1
    fig, ax = plt.subplots()
    ax.matshow(dtw, cmap=plt.cm.RdPu)
    for i in range(n):
        for j in range(m):
            c = dtw[j,i]
```

```
            cr = round(c,2)
            ax.text(i, j, str(cr), va='center', ha='center',
                fontsize = 5)
    plt.show()
    return dtw


# -- ==dijkstra== --
def dijkstra(matrix,m,n,dtw):
    path = []
    i=m
    j=n
    path.append(matrix[i,j])
    while(i>0)or(j>0):
        if (i==0):
            j = j-1
        elif (j==0):
            i = i-1
        else:
            if dtw[i-1,j] == dio_fixes.mini(dtw[i-1,j],dtw[i
                ,j-1],dtw[i-1,j-1]):
                i=i-1
            elif dtw[i,j-1] == dio_fixes.mini(dtw[i-1,j],dtw
                [i,j-1],dtw[i-1,j-1]):
                j=j-1
            else:
                i=i-1
                j=j-1
        path.append(matrix[i,j])
    return sum(path)


# -- ==corpus_dtw== --
```

```
def corpus_dtw(path, degree, source='corpus'):
    pieces = music21.corpus.getWork(path)
    diagram = []
    i=0
    j=0
    while i < len(pieces):
        item1 = pieces[i]
        while j < len(pieces):
            item2 = pieces[j]
            if (i==j):
                diagram.append(0)
            else:
                diagram.append(dynamic_warping(item1,item2,
                    degree))
            j = j+1
        j=0
        i = i+1
    return diagram
```

Visualizer

```
"""
    author: Matteo Pennesi
"""
import dionysus as dio
import dionysus_fixes as diofixes
import music21parser as m21p
import music21
import math
import numpy as np
import matplotlib.pyplot as plt
import os
import posixpath
```

```
# -- ==heatmap== --
def heatmap(path, degree, source='corpus', start = 0, end =
    None):
    fig, ax = plt.subplots()
    pieces = music21.corpus.getWork(path)
    intersection_matrix = np.array(m21p.corpus_distance(path
        ,degree))
    intersection_matrix.resize(int(len(pieces)),int(len(
        pieces)))
    min_val, max_val = 0, 1,
    ax.matshow(intersection_matrix, cmap=plt.cm.Blues)
    for i in range(len(pieces)):
        for j in range(len(pieces)):
            c = intersection_matrix[j,i]
            c = round(c,3)
            ax.text(i, j, str(c), va='center', ha='center',
                fontsize = 10)
    pezzi = []
    i=0
    plt.show()


# -- ==dendrogram== --
def show_dendrogram(path, degree, source='corpus', start =
    0, end = None):
    from scipy.cluster.hierarchy import dendrogram, linkage
    pieces = music21.corpus.getWork(path)
    intersection_matrix = np.array(m21p.corpus_distance(path
        ,degree))
    intersection_matrix.resize(int(len(pieces)),int(len(
        pieces)))
    min_val, max_val = 0, 1,
```

```
    Z = linkage(intersection_matrix, 'average',
        optimal_ordering = True)
    fig = plt.figure(figsize=(10,10))
    nomi = []
    for piece in pieces:
        nomi.append(posixpath.basename(piece))
    dn = dendrogram(Z, labels=nomi)
    fig.autofmt_xdate()
    plt.show()
    return intersection_matrix


# -- ==dendrogram_dtw== --
def dendrogram_dtw(path, degree, source='corpus', start = 0,
    end = None):
    from scipy.cluster.hierarchy import dendrogram, linkage
    from scipy.spatial.distance import pdist
    pieces = music21.corpus.getWork(path)
    dtw_matrix=np.array(m21p.corpus_dtw(path,degree))
    dtw_matrix.resize(int(len(pieces)),int(len(pieces)))
    min_val, max_val = 0, 1,
    Z = linkage(pdist(dtw_matrix), method='single',
        optimal_ordering = True)
    fig = plt.figure(figsize=(10,10))
    nomi = []
    for piece in pieces:
        nomi.append(posixpath.basename(piece))
    dn = dendrogram(Z, labels=nomi)
    fig.autofmt_xdate()
    plt.show()
    return dtw_matrix
```

Bibliografia

[1] M. Bergomi, *Dynamical and Topological Tools for (Modern) Music Analysis*, Tesi di dottorato, Université Pierre et Marie Curie-Paris VI. (2015)

[2] L. Bigo, *Spatial Computing for Symbolic Musical Representations*, Tesi di dottorato (2013)

[3] D. Lewin, *Generalized musical intervals and transformations.* Oxford University Press. (1987)

[4] R. Cohn, *Maximally Smooth Cycles, Hexatonic Systems, and the Analysis of Late-Romantic Triadic Progressions.*, Music Analysis 15/1: 9–40 (1996)

[5] D. Tymoczko, *Three Conceptions of Musical Distance.* Mathematics and Computation in Music (2009)

[6] R. Cohn, *Neo-Riemannian operations, parsimonious trichords, and their "Tonnetz" representations*, Journal of Music Theory, 41(1):1–66. (1997)

[7] H. Partch, *Genesis of a Music*,Da Capo Press, (1974)

[8] E. Chew, *The spiral array: an algorithm for determining key boundaries.* In: *Music and Artificial Intelligence - Second International Conference*, ICMAI. (2002)

[9] G. Albini, *Modelli matematici per l'analisi e la composizione musicale: uno studio sul Tonnetz e sulle teorie neo-Riemanniane*, Tesi di laurea specialistica, Università di Pavia. (2007)

[10] L. Eulero, *Tentamen Novae Theoriae Musicae, Petropoli*, ex Typographia Academiae Scientia, (1739)

[11] H. Riemann, *Ideas for a Study "On the Imagination of Tone"*, 1914-1915, english translation, *Journal of Music Theory*, Vol. 36, No. 1. pp. 81-117. (1992)

[12] D. Tymoczko, *Scale Networks and Debussy.* Journal of Music Theory, 48 (2): 219–294 (2004)

[13] E. Gollin, *Some aspects of three-dimensional "Tonnetze"*, Journal of Music Theory (1998)

[14] R. Cohn *Audacious Euphony: Chromatic Harmony and the Triad's Second Nature*, Oxford University Press (2012)

[15] D. Tymoczko, *The geometry of musical chords*, Science 313, pp. 72–74. (2006)

[16] D. Tymoczko, *A geometry of music: harmony and counterpoint in the extended common practice*, Oxford University Press.(2011)

[17] D. Tymoczko, *The Generalized Tonnetz*, In: *Journal of Music Theory*.56(1):1–52. (2012)

[18] L. Bigo,M. Andreatta *Topological Structures in Computer-Aided Music Analysis*. In: *Computational Music Analysis*. Springer (2016)

[19] C. Callender, I. Quinn, D. Tymoczko, *Generalized Voice-Leading Spaces*,In: *Science* 320, pp. 346–348. (2008)

[20] M. Andreatta, *Méthodes algébriques en musique et musicologie du XXe siecle: aspects théoriques, analytiques et compositionnels*, Tesi di dottorato, École des Hautes Etudes en Sciences Sociales (2003)

[21] L. Bigo, M. Andreatta, J.-L. Giavitto,O. Michel, A. Spicher, *Computation and visualization of musical structures in chord-based simplicial complexes.* In *Mathematics and Computation in Music*, pag 38–51. Springer (2013)

[22] W.A. Sethares, R. Budney, *Topology of Musical Data*, In: *Journal of Mathematics and Music*, Vol. 00, , 1–33 (2013)

[23] F. Jedrzejewski, *Mathematical Theory of Music*, Collections Musique/Sciences. Editions DELATOUR FRANCE/Ircam-Centre Pompidou (2006)

[24] G. Genuys, *Non-commutative homometric musical structures and chord distances in geometric pitch spaces*, Tesi di dottorato, UPMC-IRCAM (2017)

[25] L. Bigo, J.-L. Giavitto,O. Michel, A. Spicher, *Building Topological Spaces for Musical Objects.* In *Mathematics and Computation in Music* (2011).

[26] H. Edelsbrunner, J. Harer, *Persistent homology - A survey*, Contemporary mathematics, 453:257–282.

[27] M. Ferri, P. Frosini, C. Landi, *Stable shape comparison by persistent homology.* Atti del Seminario Matematico e Fisico dell' Università di Modena e Reggio Emilia. (2011)

[28] P. Frosini, *Measuring shapes by size functions.* In *Intelligent Robots and Computer Vision X: Algorithms and Techniques*, pages 122–133. International Society for Optics and Photonics (1992)

[29] H. Edelsbrunner, D. Letscher, A. Zomorodian, *Topological persistence and simplification.* In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 454–463, (2000)

[30] L. Tu, *An Introduction to Manifolds* 10.1007/978-0-387-48101-2 (2008)

[31] R. Ghrist, *Barcodes: the persistent topology of data. Bulletin of the American Mathematical Society.* 45(1):61–75. (2008)

[32] M.S. Cuthbert, C. Ariza, *music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data.*, 11th International Society for Music Information Retrieval Conference (ISMIR), pp. 637-642. (2010)

[33] M.Paolizzi, *Analisi della complessità del voice leading di musica polifonica sacra tramite matrici di permutazione parziale*, Tesi di laurea, Università di Bologna (2018)

[34] O. Messiaen, *The technique of my musical language*, translated by Satterfield J., *Alphonse Leduc* (1966).

[35] S. Kostka, *Materials and Techniques of Post-Tonal Music*, Fourth edition, Routledge (2016)

[36] D. Cohen-Steiner,H. Edelsbrunner, D. Morozov, *Vines and Vineyards by Updating Persistence in Linear Time*, Conference paper, Proceedings of the 22nd ACM Symposium on Computational Geometry, (2006)

[37] P. Senin, *Dynamic Time Warping Algorithm Review* (2008)

[38] R.N. Shepard, *Geometrical Approximations to the Structure of Musical Pitch.* In *Psychological Review* Vol.89 n.4. (1982)

[39] A.k. Jain, R. C. Dubes *Algorithms for Clustering Data.* Prentice Hall. (1988)

[40] `https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html#module-scipy.cluster.hierarchy`

[41] A. Angeli, M. Ferri, I. Tomba, *Symmetric functions for fast image retrieval with persistent homology*, Math Meth Appl Sci. (2018)