Traitement de la polyphonie pour l'analyse informatique de partitions musicales

Nicolas Guiomard-Kagan

Thèse de Doctorat

Mention Informatique

présentée à ***l'École Doctorale en Sciences Technologie et Santé (ED 585)***

de l'Université de Picardie Jules Verne

par

Nicolas Guiomard-Kagan

pour obtenir le grade de Docteur de l'Université de Picardie Jules Verne

Traitement de la polyphonie pour l'analyse informatique de partitions musicales

Soutenue le 20 mars 2017 après avis des rapporteurs, devant le jury d'examen :

Myriam Desainte-Catherine, Professeur, LaBRI, Bordeaux	**Rapporteur**
Florent Jacquemard, Chargé de Recherche Inria, IRCAM, Paris	**Rapporteur**
Mathieu Giraud, Chargé de Recherche CNRS, CRIStAL, Lille	**Examinateur**
Richard Groult, Maître de Conférences, MIS, UPJV, Amiens	**Examinateur**
Florence Levé, Maître de Conférences, MIS, UPJV, Amiens	**Examinateur**
Olivier Pietquin, Professeur, CRIStAL, Lille et Google DeepMind	**Examinateur**
Vincent Villain, Professeur, MIS, UPJV, Amiens	**Examinateur**
Tillman Weyde, Senior Lecturer, City University of London	**Examinateur**

Aux femmes de ma vie, Fanny, Aurore, Kâline, Sanaha et Denise

Remerciements

Je tiens tout d'abord à remercier mes co-encadrants Mathieu Giraud, Richard Groult et Florence Levé pour leur disponibilité tout au long de ma thèse. À chaque fois que j'avais des questions, j'ai toujours trouvé un interlocuteur pour y répondre. Merci d'avoir développé pendant ces 3 ans et demi une relation de confiance, grâce à votre bonne humeur, et aux sorties pédagogiques qui ont permis de mieux nous connaitre, et ainsi tisser des rapports plus soutenus.

Merci à l'ensemble des membres de l'équipe Algomus pour les rencontres du lundi et les journées au vert qui m'ont permis de mener mes recherches dans un cadre de travail agréable et dynamique. Merci plus particulièrement à Manu et Louis pour votre relecture.

Myriam Desainte-Catherine et Florent Jacquemard ont accepté d'être les rapporteurs de cette thèse, et je les en remercie, de même que pour leur participation au Jury. Ce fut un vrai plaisir d'échanger avec eux.

Messieurs Olivier Pietquin, Vincent Villain et Tillman Weyde m'ont fait l'honneur de participer à mon Jury de soutenance, je les en remercie profondément. Merci encore pour vos questions et vos encouragements. J'espère pouvoir échanger de nouveau avec vous lors de prochain travaux.

Merci à l'équipe SDMA qui a financé une partie de mes déplacements, et plus particulièrement à son directeur Vincent qui m'a fait confiance et a soutenu mes projets, en me permettant de faire des échanges internationaux.

Je tiens aussi à mentionner le plaisir que j'ai eu à travailler au sein du laboratoire MIS, et j'en remercie ici tous les membres. Partager des bureaux s'est avéré assez facile grâce aux collègues qui malgré le travail sont toujours près à nous soutenir et à échanger (surtout les marques de thé et café).

Un très grand merci à toute ma famille pour leur présence et leur soutien. Et plus particulièrement à mes parents qui ont lu, relu et re-relu mon manuscrit de thèse quelque soit le jour de la semaine ou l'heure de la journée.

Merci à mes deux princesses (Kâline et Sanaha) qui par leurs rires et jeux m'ont permis de ne jamais baisser les bras

Et enfin, à ma moitié, ma femme Aurore sans qui je n'aurai jamais pu mener cette thèse (en tout cas cette partie sur les remerciements).

Je dédie cette dernière pensée à ma soeur...

Table des matières

« [...] La Machine n'est après tout qu'un outil, qui permet à l'humanité de progresser plus rapidement en la déchargeant d'une partie des besognes de calcul et d'interprétation. Le rôle du cerveau humain demeure ce qu'il a toujours été ; celui de découvrir les informations qu'il conviendra d'analyser et d'imaginer de nouveaux concepts pour procéder aux tests. »

Isaac Asimov, *Les robots*

Introduction

Le thème d'étude dans lequel se place cette thèse est l'*informatique musicale*. L'idée que l'on puisse combiner l'informatique avec la musique est présente depuis les débuts de l'informatique. En 1851, la première programmeuse Ada Lovelace écrit dans ses notes que *« [...] la machine pourrait composer de manière scientifique et élaborer des morceaux de musique de n'importe quelle longueur ou degré de complexité »* [42]. Depuis, l'informatique musicale a évolué et l'envie de l'utiliser pour interagir avec l'ordinateur en a fait un domaine à part entière. À l'intérieur de cette thématique, il existe deux approches complémentaires. La première utilise comme source de données *le signal audio* et la seconde utilise *l'information textuelle de la partition* (comme par exemple le format MIDI). Dans cette thèse, nous nous plaçons dans la seconde approche.

Lorsque nous écoutons de la musique, nous sommes capables de reconnaître des éléments structurants de la musique même sans connaissance préalable de notions musicales. Ceci se traduit, par exemple, par le fait que nous sommes capables de dire si une chanson est une reprise, l'auditeur est capable de reconnaître des motifs même si ceux-ci subissent des transformations. Dans le chapitre 1, nous reviendrons plus en détail sur ces notions. Plus généralement, *l'analyse musicale* consiste à étudier des œuvres dans leur ensemble, ce qui peut aller du contexte dans lequel la pièce a été écrite à l'analyse fine de la structure de la pièce. L'analyse musicale est ainsi pratiquée par des théoriciens de la musique, des musicologues, mais aussi, de manière plus ou moins explicite, par les interprètes et les auditeurs.

L'un des objectifs principaux de l'informatique musicale est de produire, à partir de partitions encodées pour l'ordinateur, des analyses pertinentes et automatiques se rapprochant de celles que pourrait effectuer un musicologue. Nous verrons dans le second chapitre que de nombreux travaux en informatique musicale considèrent des partitions *monophoniques* (une seule note est jouée à la fois) ou au moins, des partitions composées d'un ensemble de *voix* monophoniques.

Mais l'analyse automatique de partitions ne peut se restreindre aux données monophoniques. Tout d'abord, il arrive que les données musicales ne soient pas disponibles sous une forme monophonique. Par exemple, un enregistrement piano en MIDI ne sera pas forcément séparé en lignes monophoniques. Surtout, la musique n'est pas uniquement faite de lignes monophoniques distinctes. Par exemple des pièces écrites pour des instruments polyphoniques (comme la guitare ou le piano) peuvent utiliser des *accords*. Enfin, même dans le cas de musique séparable en lignes monophoniques, des ambiguïtés peuvent subsister.

Nous verrons dans le chapitre 2 que l'analyse de partitions *polyphoniques*, c'est-à-dire à plusieurs voix, est complexe. La figure 1 donne une première idée de cette complexité. Ici, deux partitions représentent le même extrait de musique. Sur la partition du haut, les deux lignes mélodiques sont placées sur deux portées indépendantes (1 et 2), l'accompagnement est séparé en deux autres portées (3 et 4), et la portée 3 contient des accords. Sur la partition du bas, l'ensemble des portées est fusionné. Nous voyons sur cet exemple qu'une observation

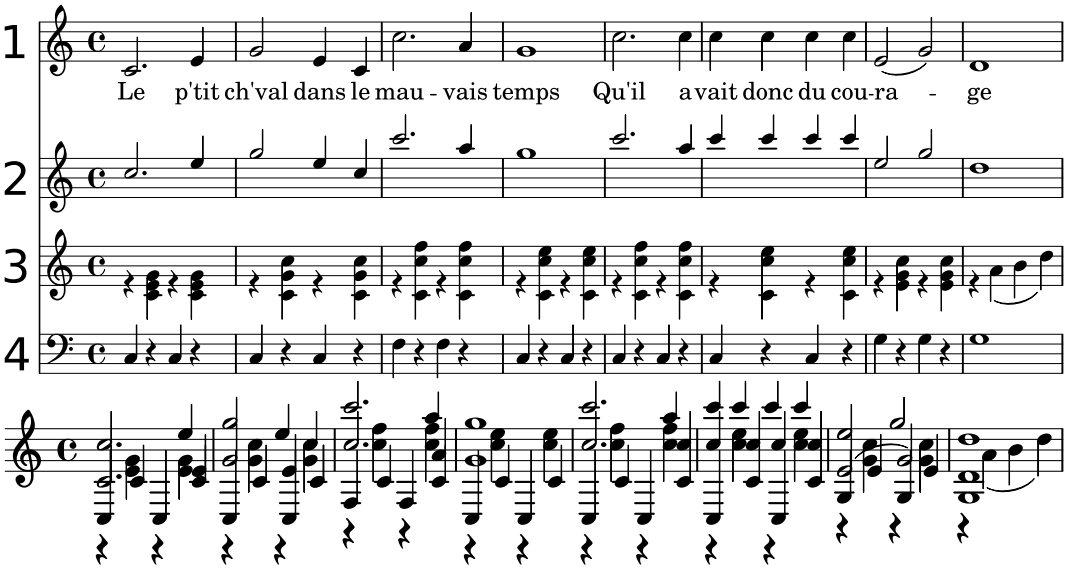


FIGURE 1 – Partitions représentant les mesures 7 à 14 d'une transcription de la chanson *Le petit cheval* de Georges Brassens (`http://leslaskar.fr/CD_Brassens/cadre%20alpha.htm`). La partition du haut représente une séparation possible de cette musique en quatre *streams* : les portées 1, 2 et 4 sont monophoniques, et la portée 3 est polyphonique. La partition du bas contient toutes les notes.

qui semblait évidente en monophonique (comme par exemple la mélodie reprise à l'octave sur la deuxième portée) est plus compliquée en polyphonique. *Comment organiser l'information polyphonique de manière à la rendre plus compréhensible ?*

Pour essayer de répondre à ce problème, je me suis intéressé à la *séparation des partitions polyphoniques en voix et en streams.* J'ai pour cela commencé par comparer trois algorithmes de séparation en voix et trois algorithmes de séparation en streams. Afin de mesurer équitablement les résultats de ces deux approches, la première phase de mes recherches a consisté à trouver une méthode d'évaluation. La méthode retenue est celle qui s'intéresse à la transition entre deux notes. Dans le chapitre 3, je montre que les séparations en voix et en streams sont assez proches. Les conclusions de cette étude mettent également en avant les qualités de l'algorithme de séparation en voix de Chew et Wu, et plus particulièrement la première étape de cet algorithme qui consiste à segmenter la partition en fonction du nombre de notes jouées simultanément dans ce qui est appelé « contigs ». C'est pourquoi, dans le chapitre 4, je m'intéresse à la seconde étape de l'algorithme de Chew et Wu. J'améliore cette étape en optimisant l'ordre de connexion des contigs ainsi que la manière de les connecter. La solution que je propose utilise plusieurs paramètres musicaux pour définir l'ordre et la manière de les assembler. Dans la majorité des études sur la séparation en voix ou en streams, le paramètre le plus utilisé compare la différence de hauteur entre deux notes. L'un des paramètres que j'utilise consiste à étendre cette caractéristique en prenant en compte la moyenne des hauteurs de *toutes les notes* d'un contig. Afin de définir les coefficients associés à chaque paramètre, j'utilise un algorithme génétique qui cherche les coefficients donnant la meilleure séparation en voix.

Je conclus cette thèse dans le chapitre 5 en proposant des perspectives pour combiner séparation en voix ou en streams et tâches d'analyse musicale.

Chapitre 1

Notions de musique, représentations musicales

Les recherches que j'ai effectuées tout au long de ma thèse se sont faites dans le cadre de l'informatique musicale. Plus précisément, elles ont porté sur la *séparation de musique polyphonique en voix et streams*. Je ne présenterai dans ce chapitre que les éléments qui permettront une compréhension de mon travail. Pour aller plus loin dans les connaissances de la théorie de la musique, je conseille l'ouvrage de Danhauser [16].

Cette thèse traitera uniquement de la musique dite *occidentale* (période moderne). Ce cadre restreint de la musique a été choisi car c'est une des musiques qui répond à des règles, par exemple de *consonance* qui définissent la cohérence d'un ensemble de sons entendus simultanément ou successivement. C'est également une musique qui contient des formes d'écritures assez documentées comme par exemple l'art de la fugue. Dans une première section, je parlerai des objets qui permettent de représenter la musique à l'écrit. Dans une seconde section, j'aborderai différentes organisations possibles de la musique. Dans la troisième section, je présenterai des méthodes couramment utilisées pour l'analyse d'une pièce musicale. Et dans la quatrième et dernière section de ce chapitre, je m'intéresserai aux différentes formes de représentation de la musique sur ordinateur.

1.1 Notes

La musique s'écoute mais elle peut également se lire. Dans cette section, nous allons nous intéresser à la transcription d'une partie de l'information musicale sur un support d'écriture : la *partition*. Elle contient des informations qui permettent d'interpréter la musique (voir la figure 1.1)

La musique se note sur une *portée*, de nos jours constituée d'un ensemble de cinq lignes horizontales (voir la figure 1.2). Les lignes se comptent du bas vers le haut. L'élément qui permet de représenter le son sur la partition est la *note*. La note représente la durée et la hauteur associée à un son. La forme de la note (*figure* de note) indique sa durée, comme par exemple la ronde, la blanche, la noire, la croche ou la double croche (voir la figure 1.3). La ronde représente la plus longue durée, et chacune des autres figures vaut la moitié de la figure qui la précède, et par conséquent le double de celle qui la suit (voir la figure 1.4).

L'unité de quantification de la durée musicale est le *temps*. Les partitions les plus régulières se découpent temporellement en sections appelées *mesures* (voir la figure 1.5). La durée musicale n'est attachée d'aucune manière que ce soit à une valeur absolue en temps.

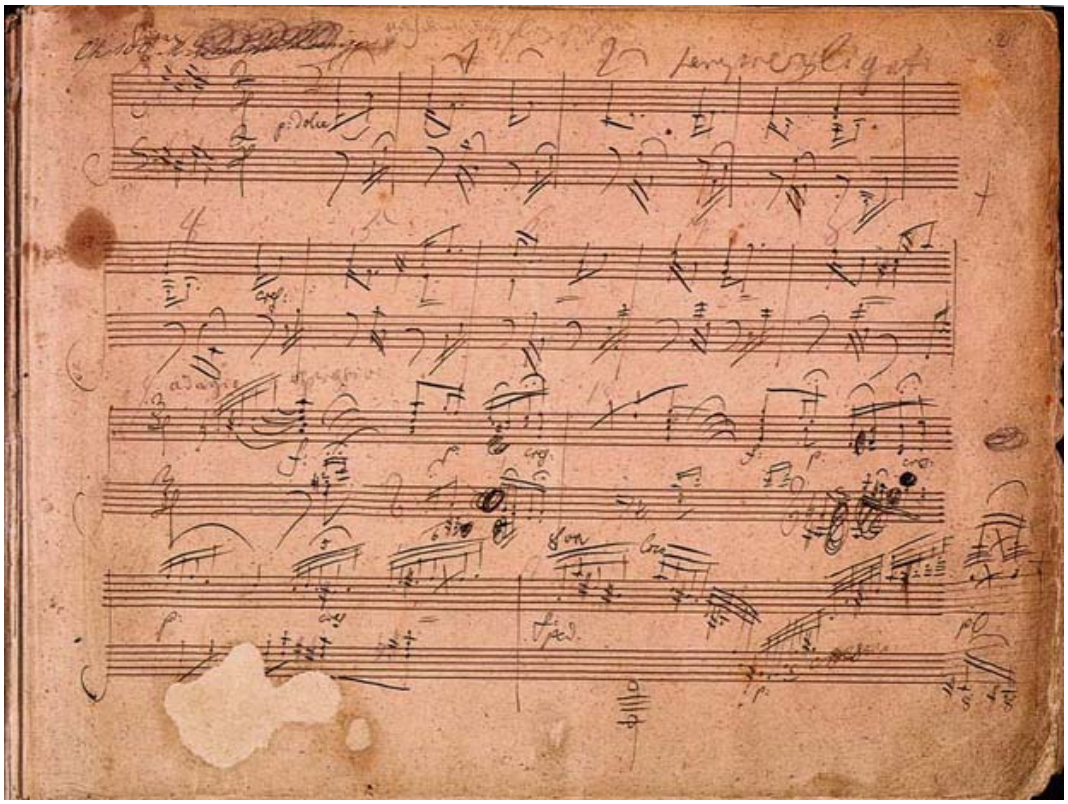


FIGURE 1.1 – Page manuscrite originale des treize premières mesures de la Sonate pour piano n° 30 de L. van Beethoven - `https://commons.wikimedia.org/wiki/File:Beethoven_Klaviersonate_Nr_30.jpg#filelinks`.

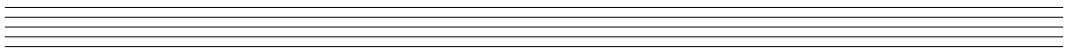


FIGURE 1.2 – Une portée est généralement faite de la réunion de cinq lignes horizontales.

FIGURE 1.3 – De gauche à droite : ronde, blanche, noire, croche, double croche.

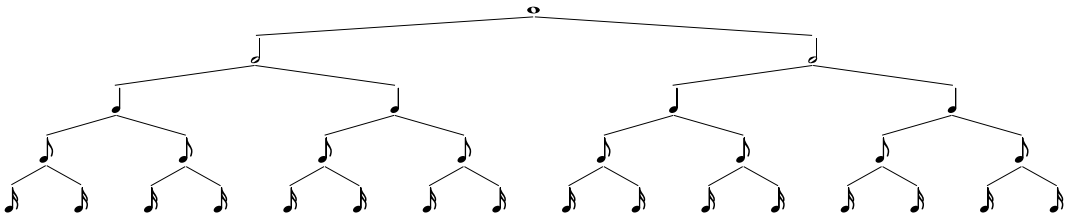


FIGURE 1.4 – Chaque figure vaut en durée, la moitié de la figure précédente, et le double de la figure suivante. Par exemple une ronde vaut quatre noires.

Le *tempo* d'une pièce ou d'un passage permet de donner sa vitesse d'exécution, en se référant à une *pulsation* (temps joués pendant une minute). Par exemple, une pulsation de « 120 à la noire » signifie que la durée d'une noire est égale à 1/2 seconde.

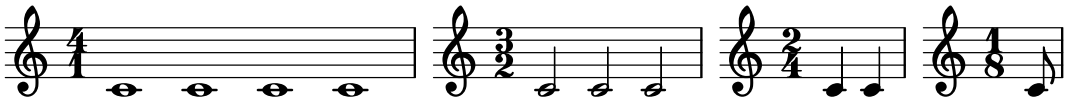


FIGURE 1.5 – Différentes mesures.

De la même manière qu'il existe des figures pour représenter les sons, il existe également des figures permettant de représenter les absences de sons. Il s'agit des *silences* (voir la figure 1.6). Chacune de ces figures possède son équivalent en durée parmi les figures représentant les sons. Par exemple, une pause a la même durée qu'une ronde et un soupir qu'une noire.

FIGURE 1.6 – De gauche à droite : pause, demi-pause, soupir, demi-soupir, quart de soupir.

La hauteur d'une note dépend de sa position verticale sur la portée. Plus une note sera placée haut et plus elle sera aigüe. Il existe sept noms de notes qui expriment tous les sons : do, ré, mi, fa, sol, la, si. La correspondance en notation anglo-saxonne est : C, D, E, F, G, A, B. Ces notes forment une *série* de sons allant du grave à l'aigu. La figure 1.8 montre une *gamme* de Do majeur. Une gamme est une suite de notes conjointes, où la dernière répète la première à l'octave. Pour savoir à quel nom correspond une note sur la portée, il faut regarder la *clé*. Les clés servent à fixer le nom des notes. Habituellement, on utilise trois clés : sol, ut, fa (voir la figure 1.7). Par exemple sur la figure 1.8, il s'agit d'une clé de sol cela signifie que la note placée sur la deuxième ligne est un sol.

FIGURE 1.7 – De gauche à droite : clé de sol, clé d'ut, clé de fa.

Le plus grand écart de hauteur entre deux notes voisines dans la gamme majeure est appelé *ton*. La figure 1.9 montre les distances séparant les notes de la gamme majeure.

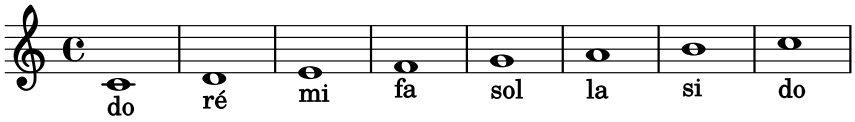


FIGURE 1.8 – Gamme de do majeur.

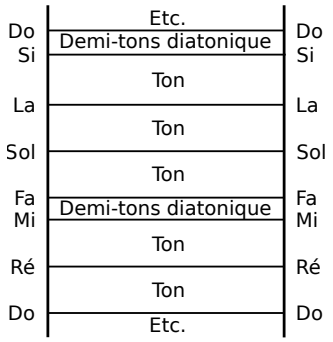


FIGURE 1.9 – Représentation verticale de la gamme majeure

La différence entre deux hauteurs de notes s'appelle l'*intervalle*, qui peut se mesurer en tons et demi-tons. Sur la figure 1.8, l'intervalle entre les deux do est de 12 demi-tons. Il existe également des symboles appelés *altérations* (voir la figure 1.10) permettant de modifier la hauteur d'une note. C'est le cas du *dièse* qui augmente une note d'un demi-ton, le *bémol* qui diminue une note d'un demi-ton, et le *bécarre* qui annule les effets du dièse et du bémol. Ces altérations se placent sur la portée à la même hauteur que la note modifiée. Elles peuvent se placer devant une seule note ou en début de portée après la clé, dans l'*armure*. Elles modifient toutes les notes de même nom, peu importe l'octave.

FIGURE 1.10 – Altérations. De gauche à droite : un dièse, un bémol et un bécarre

Exemple d'une partition. La figure 1.11 représente une partition écrite en clé de sol avec une armure contenant un bémol placé sur la ligne du si. Elle contient 12 mesures. La première mesure contient 7 notes et elle commence par un fa qui dure une croche.

1.2 Monophonie et polyphonie

Comment les notes sont-elles organisées dans une partition ? Si elles sont jouées les unes après les autres sans jamais être jouées simultanément, il s'agit d'une pièce *monophonique*. Dans la figure 1.11, nous pouvons voir qu'à aucun moment les notes ne sont jouées en même temps.

On parle de pièce *polyphonique* quand plusieurs sons peuvent être entendus au même moment. La musique polyphonique peut provenir de plusieurs instruments jouant en même temps ou d'un seul instrument harmonique (comme par exemple le piano ou la harpe). Il existe plusieurs formes d'écritures polyphoniques, comme par exemple la mélodie accompagnée ou le contrepoint.

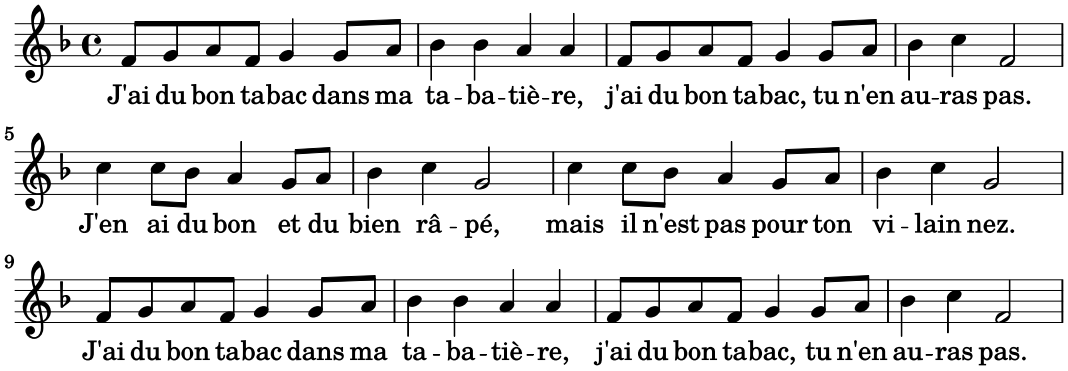


FIGURE 1.11 – Partition du premier couplet et du refrain de *J'ai du bon tabac* de l'abbé de l'Attaignant (1760)

Mélodie accompagnée. La notion de mélodie peut-être complexe à définir, la définition donnée par l'académie française indique qu'« [...] une mélodie est une suite ordonnée de sons de différentes hauteurs constituant une ligne musicale » [20].

La mélodie peut être mise en opposition avec l'accompagnement. La mélodie peut être vue comme plus variée dans le tempo et dans les intervalles alors que l'accompagnement serait défini comme une base faite d'accords pour soutenir la mélodie. Par exemple, dans la figure 1.12, la mélodie correspond à la ligne musicale chantée de la première portée et l'accompagnement correspond à la suite d'accords des deux portées du bas.

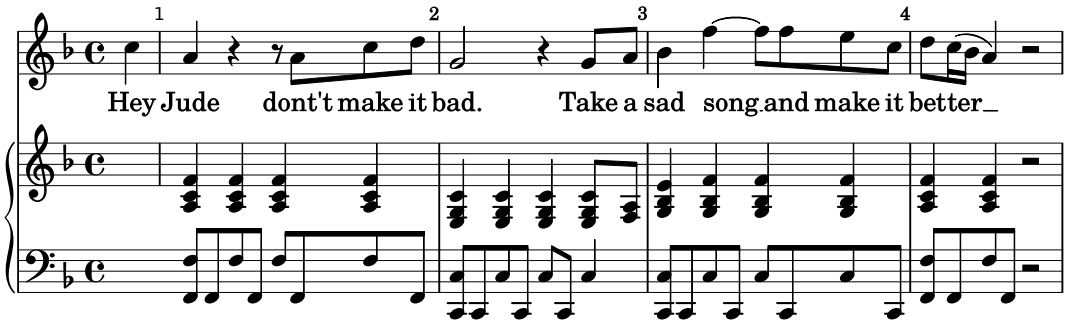


FIGURE 1.12 – Premières mesures de la partition *Hey Jude* de Paul McCartney et John Lennon. La première portée représente la mélodie chantée et les deux autres portées représentent l'accompagnement au piano. La mélodie et l'accompagnement sont joués simultanément – arrangement pour piano et voix (`http://www.musicroom.fr/fr-fr/se/id_no/018011/details.html`).

Contrepoint. Le contrepoint rigoureux est l'art de mêler des lignes mélodiques distinctes pour former une belle harmonie. Ces différentes lignes mélodiques peuvent être des *voix*. Les voix peuvent être classées en fonction de leur hauteur dans différents *registres*, le registre grave, le registre moyen, le registre aigu, et le registre suraigu. Dans la musique pour piano comme dans les voix chantées, ces voix peuvent prendre les noms suivant : basse, ténor, alto et soprano (de la plus grave à la plus aiguë).

Il existe de nombreux traités de contrepoint [5]. Dans la figure 1.13, nous avons un exemple de musique contrapuntique où trois voix sont jouées ensemble.

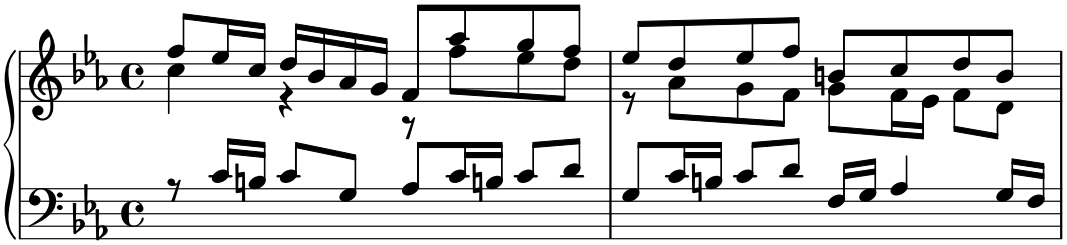


FIGURE 1.13 – Mesure 7 et 8 de la Fugue #2 du livre I du *Clavier bien tempéré* de J.-S. Bach (en Do mineur, BWV 847).

1.3 Analyse musicale

Les auditeurs de musique même sans connaissances théoriques sont capables d'un certain degré d'analyse. J'ai pu m'en rendre compte lors des présentations des travaux de l'équipe Algomus. Des animations ont permis de faire découvrir quelques notions d'analyse musicale à un jeune public (fête de la science 2015 et 2016 – `http://www.fetedelascience.fr`) en reconstruisant une mélodie à partir de fragments et en regroupant des extraits musicaux suivant leur ressemblance (`http://www.algomus.fr/pierres`). Lorsque l'on posait aux enfants la question : « Pouvez-vous nous dire quelle est la structure de la musique que vous entendez à la radio ? », ils répondaient le plus souvent « couplet/refrain ». L'organisation musicale est suffisamment variée entre le couplet et le refrain pour qu'un auditeur puisse entendre la différence.

Auditeurs, interprètes, musicologues historiens ou théoriciens de la musique, de nombreuses personnes font de l'analyse musicale. Elles étudient des œuvres dans leur ensemble, du contexte dans lequel la pièce a été écrit à l'analyse fine de la structure de la pièce. Nous pouvons ici faire l'analogie entre l'étude d'un texte et l'analyse d'une partition.

Une première analyse du texte pourra consister à le structurer en différentes parties. Par exemple, un texte contient une introduction, un développement et une conclusion. Ces parties peuvent être structurées en élément plus petits. L'introduction pourra par exemple commencer par une exposition des faits (où ?, quand ?, quoi ?) suivie d'un plan du document. Une analyse plus fine s'intéressera à la structure même des phrases. Par exemple, une phrase peut être composée d'un sujet, d'un verbe et d'un complément.

De la même manière, une analyse pourra consister à structurer une pièce en plusieurs parties. Par exemple, la *forme sonate* est généralement composée d'une *exposition*, d'un *développement* et d'une *réexposition* (voir la figure 1.14). Une analyse des parties permettra de définir leur structure. L'exposition d'une forme sonate commencera par un premier thème (ou zone thématique) suivi d'un second (voir la figure 1.14). L'analyse peut aussi être à une résolution plus fine, en s'intéressant à la phrase musicale.

Pour mieux comprendre la structuration possible d'un genre musical, prenons l'exemple des *fugues* de Bach, qui sont un modèle de musique contrapuntique, composées de deux à cinq voix. Chaque voix entre successivement en exposant le sujet, suivi dans la majorité des cas par le contre-sujet (voir la figure 1.15).

Ces motifs sont répétés tout au long de la pièce, soit dans leur forme initiale soit variées ou transposées. Une *transposition* consiste à décaler d'un même intervalle toutes les notes du motif. Par exemple dans la figure 1.16 pour passer de la partition du haut à celle du bas, il faut augmenter toutes les notes de deux tons et demi (quarte juste).

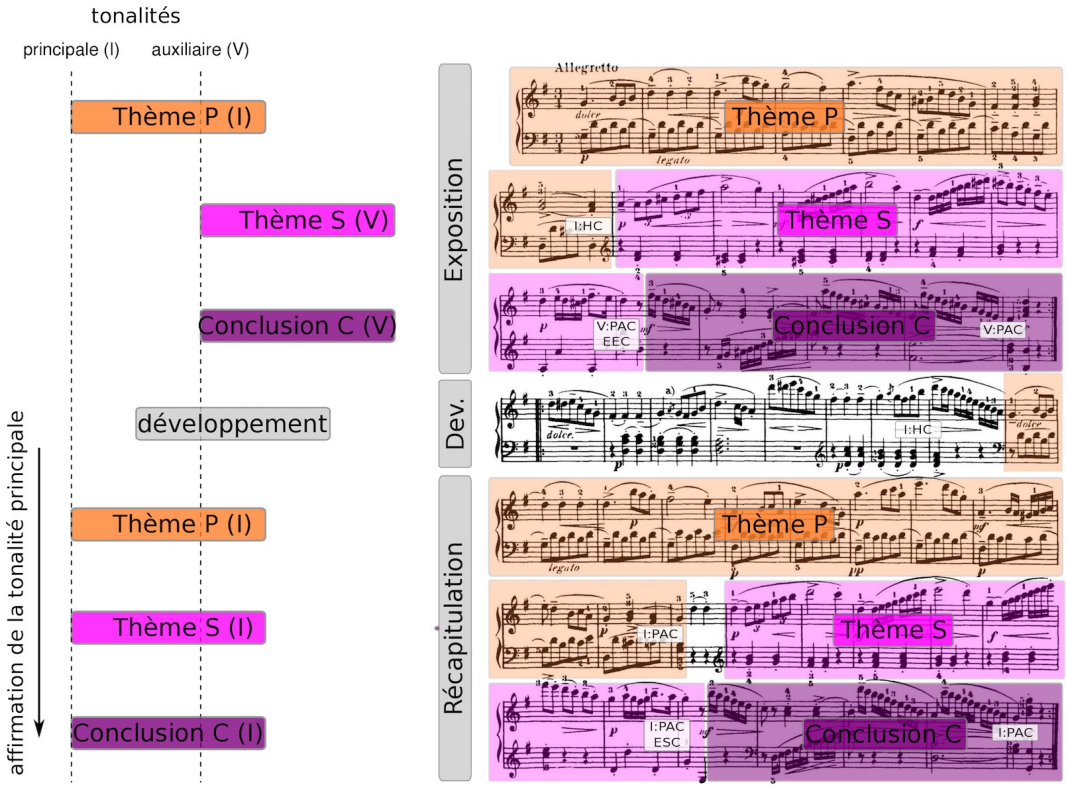


FIGURE 1.14 – Structure d'ensemble d'une forme sonate, schéma et notations tirés du livre de Hepokoski et Darcy [28, page 17] (figure extraite de [17, page 2]).

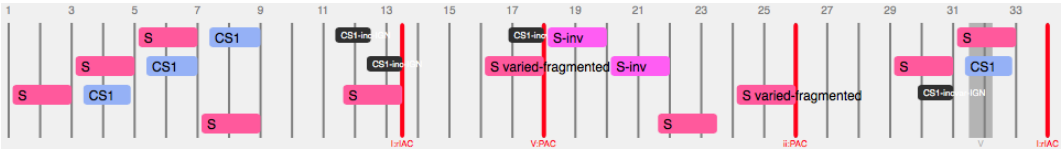


FIGURE 1.15 – Analyse de référence de la Fugue #23 du livre I de J.S. Bach (en Si majeur, BWV 868) - `http://www.algomus.fr/fugues/bach/index.html`
Les voix entrent successivement mesure 1, 3 et 5 en exposant le sujet puis en jouant le contre-sujet.

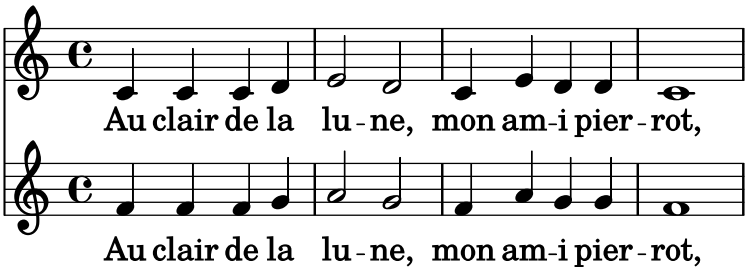


FIGURE 1.16 – Partition du haut : quatre première mesures de *Au clair de la lune* - partition du bas : transposition à la quarte.

Sur la figure 1.17, nous voyons que l'alto commence par exposer le sujet. À la fin de cette exposition, la soprano joue à son tour le sujet pendant que l'alto expose le contre sujet. Ce

contre-sujet est repris mesure 7 par la soprano. Au même moment, la ténor entre et expose à son tour le sujet.

Malgré cette structure bien connue des fugues de Bach, il existe des analyses différentes de musicologues [54, 6, 64, 36, 8]. Certains points d'analyse peuvent faire consensus, d'autre être sujets à débat...

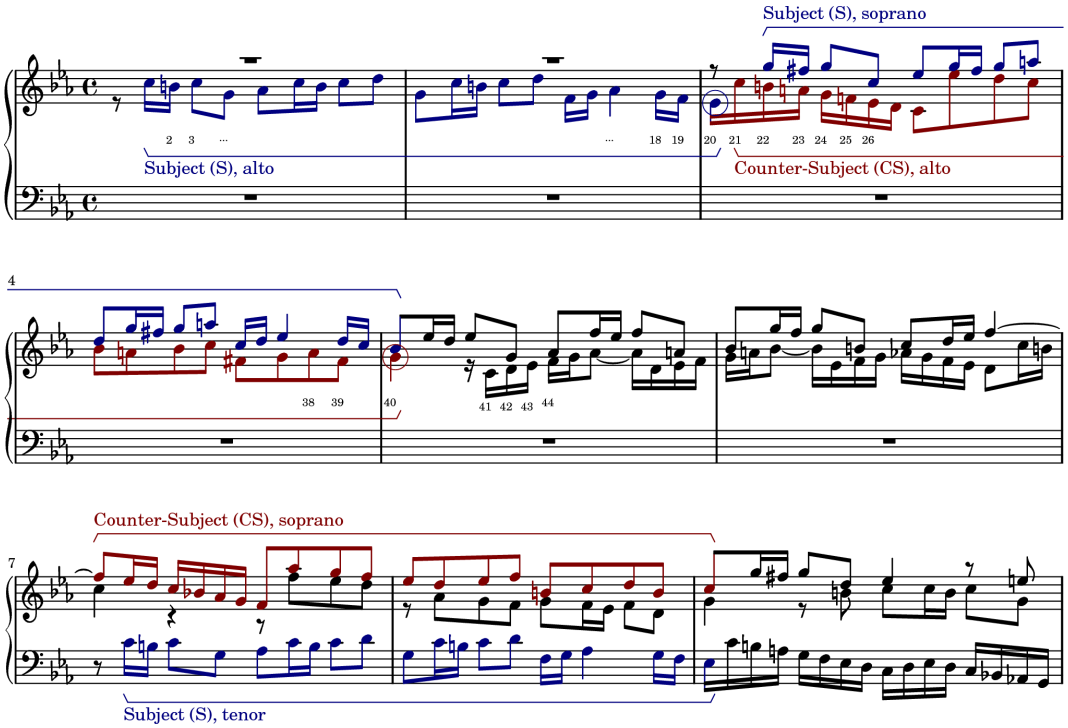


FIGURE 1.17 – Début de la Fugue #2 du livre I de J.S. Bach (en Do mineur, BWV 847).

1.4 Représentation informatique

Nous avons vu dans la section 1.1 que la partition retranscrit un ensemble d'informations musicales. Dans cette section, je présenterai différentes représentations de la musique par l'ordinateur. J'illustrerai chaque visualisation par les mesures 7 à 14 de la chanson *Le petit cheval* de Georges Brassens (voir la figure 1.18).

Dans l'ensemble du travail de ma thèse, je n'ai travaillé que sur des données symboliques (ou encore *textuelles*). Je fais ici un aparté autour de la représentation de la musique sous la forme d'un *signal audio*. Cela consiste à représenter physiquement le son d'une pièce musicale à l'aide de sa « forme d'onde ». Nous pouvons voir en comparant les deux signaux de la figure 1.19 qu'à partir d'une même partition nous obtenons différentes interprétations. Une grande partie des travaux faits en *recherche d'informations musicales* se fait en utilisant des données audio. Ces recherches peuvent avoir par exemple pour objectif la séparation de sources audio [52, 41] ou la transcription automatique de musique[53, 2].

Dans le cadre de l'analyse automatique de partition, nous nous limiterons à la représentation de trois paramètres d'une note : sa hauteur, son début et sa durée. L'information minimale nécessaire peut-être visualisée sur un *piano-roll*. La musique est représentée sur

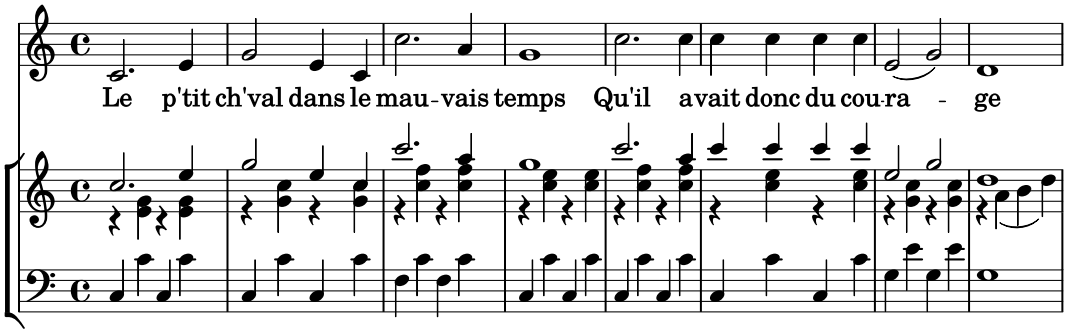


FIGURE 1.18 – Mesures 7 à 14 de la chanson *Le petit cheval* de Georges Brassens - `http://leslaskar.fr/CD_Brassens/cadre%20alpha.htm`.

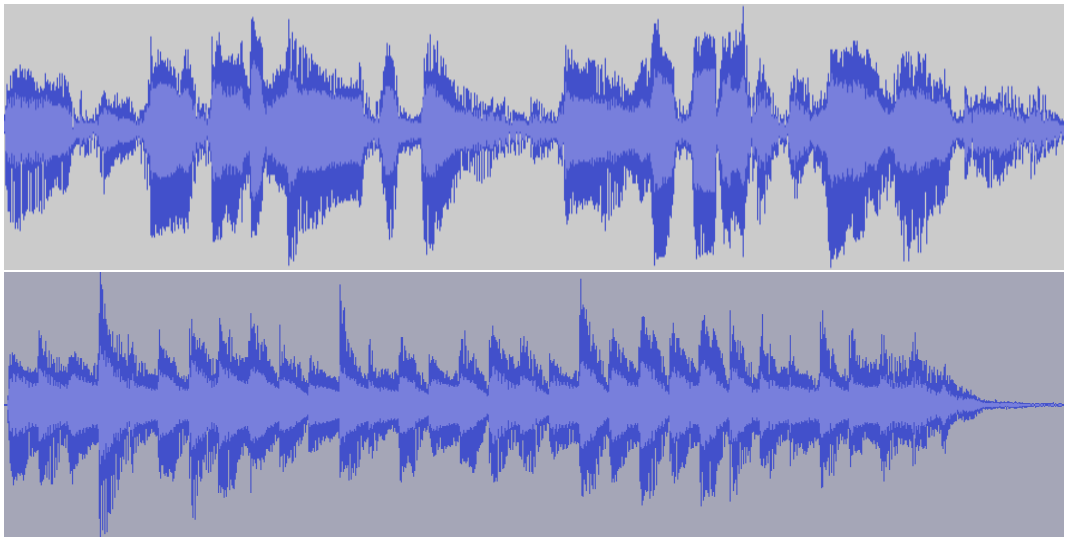


FIGURE 1.19 – Deux signaux audio correspondant aux mesures 7 à 14 de la chanson *Le petit cheval* de Georges Brassens. Le signal du haut est issu de l'enregistrement de 1952 par Georges Brassens dans le studio Chopin-Pleyel. Celui du bas provient de la génération du fichier MIDI de la figure 1.18 par un synthétiseur logiciel.

un plan orthonormé, où l'axe des abscisses correspond au temps et l'axe des ordonnées à la hauteur des notes. Dans cette représentation, chaque note est représentée par un segment. La figure 1.20 représente la vue piano-roll d'un extrait de la chanson le *petit cheval* de George Brassens.

Plusieurs formats permettent de représenter ces données, comme par exemple le protocole de communication *Musical Instrument Digital Interface* (MIDI – [50]) qui permet de pouvoir enregistrer directement des instruments de musique sur ordinateur. De plus des logiciels d'édition de partitions musicales proposent des sorties MIDI. Un avantage des fichiers MIDI est qu'ils sont assez répandus. Cependant ces fichiers provenant souvent d'enregistrements, il peut exister une différence entre la durée de la note jouée par un musicien et celle de la partition. Cela aura comme conséquence qu'il pourra être assez difficile de retrouver exactement l'information de la partition jouée.

Il existe également des formats qui décrivent plus précisément le contenu de la partition. Nous pouvons citer par exemple les formats ***kern* [30], *MusicXml* [24] ou *MEI* [58, 12].

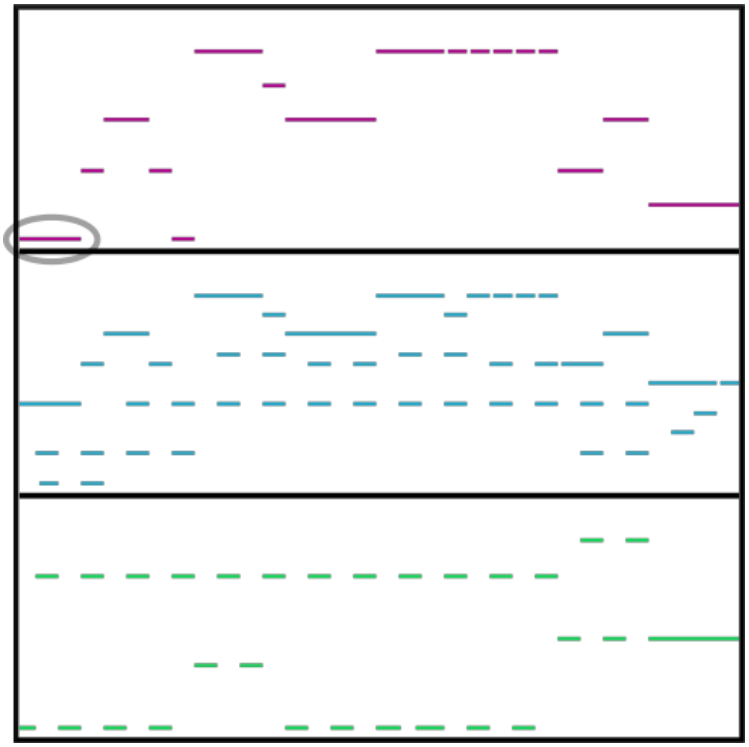


FIGURE 1.20 – Chaque cadre correspond à la vue piano-roll d'une portée des mesures 7 à 14 de la chanson *Le petit cheval* de Georges Brassens (voir la figure 1.18). Chaque segment correspond à une note de la partition. Par exemple, le segment entouré en haut à gauche équivaut au do de la mesure 7 de la première portée de la partition.

Le format **kern (`http://www.music-cog.ohio-state.edu/Humdrum/representations/kern.html`) permet de récupérer toute l'information de la partition. Sur la figure 1.21, nous voyons que chaque colonne correspond à une ligne musicale (ici quatre lignes musicales). Dans ce langage, les mesures sont explicitement représentées et les notes sont notées en utilisant la notation anglo-saxonne. Le point positif des fichiers **kern est de représenter exactement la partition sans ambiguïté. Par exemple la figure 1.21 correspond exactement à la partition de la figure 1.18.

Les formats MusicXml (`http://www.musicxml.com`) et MEI (`http://music-encoding.org`) sont tous les deux basés sur XML. Ces formats hiérarchisés à l'aide de balises ont comme point positif d'être facilement traités par des programmes de notation. Le point négatif de ces formats est qu'à l'inverse du **kern, il sont difficilement lisibles par un humain. Par exemple, la figure 1.22 ne représente que le début de la septième mesure de la chanson *Le petit cheval*.

```
1       !!!COM: Brassens, George
        !!!CNT: French
        !!!OTL: Le petit cheval
        **kern  **kern  ** kern ** kern
5       *staff1 *staff2 *staff2 *staff3
        *clefG4 *clefG4 *clefG4 *clefF3
        *k[]    *k[]    *k[]    *k[]
        *a:     *a:     *a:     *a:
        *M4/4   *M4/4   *M4/4   *M4/4
10      *MM120  *MM120  *MM120  *MM120
        =7      =7      =7      =7
        2.c     2.cc    4r      4C
        .       .       4e 4g   4c
        .       .       4r      4C
15      4e      4ee     4e 4g   4c
        =8      =8      =8      =8
        2g      2gg     4r      4C
        .       .       4g 4c   4c
        4e      4ee     4r      4C
20      4c              4r      4g 4c   4c
        =9      =9      =9      =9
        2.cc    2.ccc   4r      4C
        .       .       4cc 4ff 4c
        .       .       4r      4C
25      4a              4aa     4cc 4ff 4c
        =10     =10     =10     =10
        1g      1gg     4r      4G
        .       .       4cc 4ee 4c
        .       .       4r      4G
30      .       .       4cc 4ee 4c
        =11     =11     =11     =11
        2.cc    2.ccc   4r      4C
        .       .       4cc 4ff 4c
        .       .       4r      4C
35      4cc     4aa     4cc 4ff 4c
        =12     =12     =12     =12
        4cc     4ccc    4r      4C
        4cc     4ccc    4cc 4ee 4c
        4cc     4ccc    4r      4C
40      4cc     4ccc    4cc 4ee 4c
        =13     =13     =13     =13
        [2e     2ee     4r      4A
        .       .       4g 4cc  4e
        2g]     2gg     4r      4A
45      .       .       4g 4cc  4e
        =14     =14     =14     =14
        1d      1dd     4r      1G
        .       .       4a      .
        .       .       4b      .
50      .       .       4d      .
        ==:|!   ==:|!   ==:|!   ==:|!
        *-      *-      *-      *-
```

FIGURE 1.21 – Notation **kern des mesures 7 à 14 de la chanson *Le petit cheval* de Georges Brassens. La première colonne correspond à la mélodie de la première portée de la partition. Les trois premières lignes correspondent aux meta-données. Les sept suivantes donnent des informations relatives à la partition (clé, armure, tempo...). Les lignes commençants par un signe égal donnent le début de chaque mesure, par exemple la ligne 11 indique le début de la septième mesure. Les lignes placées entre deux mesures permettent d'écrire les notes, le chiffre donne la durée et la lettre le nom de la note, par exemple le 2.c de la première colonne à la ligne 12 correspond au do de la mesure 7 de la première portée de la partition.

```
<measure number="7" width="159">
        <note default-x="16">
                <rest/>
                <duration>4</duration>
                <voice>1</voice>
                <type>quarter</type>
        </note>
        <note default-x="52">
                <pitch>
                        <step>C</step>
                        <octave>4</octave>
        </pitch>
                <duration>4</duration>
                <voice>1</voice>
                <type>quarter</type>
                <stem default-y="5">up</stem>
        </note>
        <note default-x="52">
                <chord/>
                <pitch>
                        <step>E</step>
                        <octave>4</octave>
                </pitch>
                <duration>4</duration>
                <voice>1</voice>
                <type>quarter</type>
                <stem>up</stem>
        </note>
        <note default-x="52">
                <chord/>
                <pitch>
                        <step>G</step>
                        <octave>4</octave>
                </pitch>
                <duration>4</duration>
                <voice>1</voice>
                <type>quarter</type>
                <stem>up</stem>
        </note>
        <note default-x="88">
                <rest/>
                <duration>4</duration>
                <voice>1</voice>
                <type>quarter</type>
        </note>
        <note default-x="123">
                <pitch>
                        <step>C</step>
                        <octave>4</octave>
                </pitch>
                <duration>4</duration>
                <voice>1</voice>
                <type>quarter</type>
                <stem default-y="5">up</stem>
        </note>
        <note default-x="123">
                <chord/>
                <pitch>
                        <step>E</step>
                        <octave>4</octave>
                </pitch>
                <duration>4</duration>
                <voice>1</voice>
                <type>quarter</type>
                <stem>up</stem>
        </note>
        <note default-x="123">
                <chord/>
                <pitch>
                        <step>G</step>
                        <octave>4</octave>
                </pitch>
                <duration>4</duration>
                <voice>1</voice>
                <type>quarter</type>
                <stem>up</stem>
        </note>
</measure>
```

FIGURE 1.22 – Notation MusicXML des premières notes de la mesure 7 de la chanson *Le petit cheval* de Georges Brassens.

Chapitre 2

Analyses monophoniques et polyphoniques : un état de l'art

Nous avons vu qu'il existait des spécialistes de la musique dont l'un des buts est l'*analyse musicale*. J'ai également présenté (voir la section 1.4) différentes méthodes de représentation de l'information contenue dans la partition sur un ordinateur. Un des objectifs principaux de l'*informatique musicale* est de produire, à partir de ces représentations, des analyses pertinentes se rapprochant de celles que pourrait effectuer un *musicologue*. Mon travail de recherche a consisté à proposer des solutions permettant de *mieux comprendre des partitions polyphoniques*. Pour ce faire, je me suis intéressé à la séparation de ces données en *voix* et en *streams*.

L'un des moyens d'effectuer une *analyse automatique* d'une partition est de chercher les *répétitions de motifs*. Nous avons vu dans la section 1.3 que dans les fugues, le sujet pouvait être répété plusieurs fois à l'identique ou modifié. Je commencerai par aborder ce problème en m'intéressant à la recherche de motifs dans des données monophoniques. La recherche et l'inférence de motifs n'étant pas l'objectif principal de ma thèse, je ne présenterai ici que les articles permettant de comprendre ce domaine sans entrer dans les détails. J'exposerai ensuite pourquoi il est difficile de retrouver des motifs dans des données polyphoniques et je présenterai également dans cette partie différents algorithmes pour la recherche de motifs dans de telles données.

Aborder sans pré-traitement des données la question de la recherche de motifs dans des partitions polyphoniques est un problème difficile. Je présenterai donc dans la section 2.2 une solution alternative permettant de contourner le problème en *séparant les données polyphoniques en streams ou en voix*. Cette séparation de partitions polyphoniques en streams et en voix a constitué l'essentiel de mes travaux de recherche et j'expliquerai en détail la différence entre ces deux séparations. La section 2.3 et la section 2.4 seront consacrées à la présentation des algorithmes de séparation en voix puis en streams.

2.1 Recherche et inférence de motifs

Dans le cadre de l'analyse automatique, les données utilisées peuvent être monophoniques ou polyphoniques. Nous verrons dans cette section un aspect de l'analyse automatique, l'inférence et la recherche de motifs. Nous présentons d'abord les méthodes classiques sur les données monophoniques, puis nous nous intéresserons aux difficultés rencontrées pour ap-

pliquer ces mêmes approches dans des partitions polyphoniques et les différentes techniques permettant de contourner ces difficultés.

2.1.1 Données monophoniques

Un des objectifs de l'analyse automatique est de pouvoir retrouver les *motifs pertinents* d'une pièce, par exemple pour en déduire sa structure. La figure 2.1 représente une musique décomposée en lignes monophoniques (où les notes sont jouées les unes à la suite des autres) : nous pouvons voir sur le piano-roll de la partie basse qu'à aucun moment les segments ne se chevauchent.

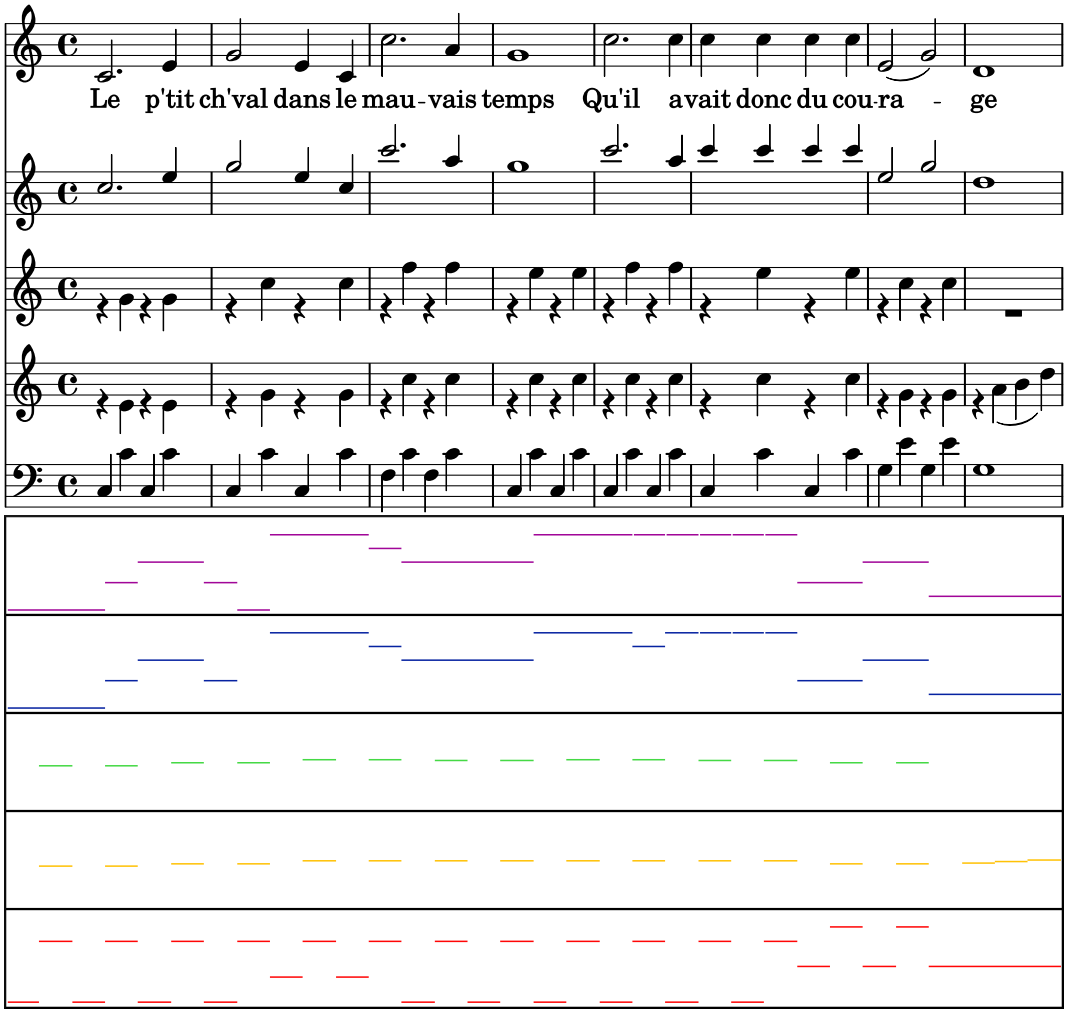


FIGURE 2.1 – Partition et piano-roll des mesures 7 à 14 séparées en lignes monophoniques de la chanson *Le petit cheval* de Georges Brassens.

Sur cet exemple, une analyse peut consister à identifier la seconde ligne monophonique comme une copie de la première une octave au dessus, avec une note modifiée dans la cinquième mesure. Sur la figure 2.2, une analyse automatique consiste par exemple à retrouver le premier motif (les quatre notes de la première mesure) de manière exacte mesure 3 et de manière approchée mesure 6 (motif transposé). Le motif de la seconde mesure (quatre

notes) se retrouve également de manière approchée mesure 4 (transposition et modification partielle du sens du mouvement) et mesure 7 (transposition et changement de rythme).

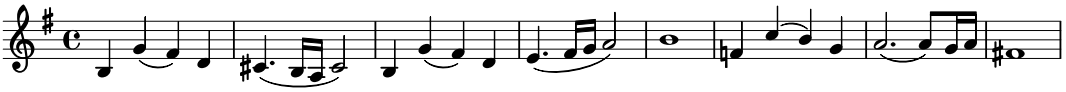


FIGURE 2.2 – Mélodie d'un thème de *Dans les plaines d'Hyrule* de Zelda

Dans [3], Janssen *et al.* dressent un état de l'art de la recherche de motifs dans des partitions monophoniques et polyphoniques. Les motifs pertinents d'une pièce sont le plus souvent ceux qui vont se répéter plusieurs fois dans une pièce. Pour trouver ces motifs et leurs occurrences la plupart des algorithmes utilisent une représentation de la musique sous forme d'une chaîne de caractères. Ces méthodes peuvent être classées en deux groupes : celles qui font de l'indexation (par exemple sous forme d'arbre) et celles qui n'en font pas.

Dans le premier groupe, nous pouvons citer [11] qui utilise un arbre de séquences pour représenter l'espace de recherche des motifs. Cet arbre est élagué en se basant sur la fréquence d'apparition des motifs. Dans [40], Lartillot utilise un arbre de motifs pour modéliser des pièces musicales, sa structure permet de trouver l'ensemble des motifs dans un temps raisonnable en utilisant des motifs courts se répétant comme brique de construction de motifs plus grands.

Pour le deuxième groupe, la première idée qui vient à l'esprit est de faire glisser tous les motifs potentiels le long de la partition et de conserver ceux qui ont le plus d'occurrences. C'est ce qui est fait dans [51]. Dans [59], Rolland utilise l'algorithme de Knuth, Morris et Pratt [39] pour éviter certaines comparaisons, en étant sûr de ne pas manquer des occurrences de motifs. De manière à trouver l'ensemble des motifs répétés maximaux, Cambouropoulos dans [7] utilise l'algorithme de Crochemore [13] qui permet de segmenter une mélodie en un ensemble d'unités répétées. Dans [14], les auteurs adaptent les algorithmes de Tuned-Boyer-Moore [29], SKIP-SEARCH [9] et MAXIMAL-SHIFT [61] pour faire de la recherche approchée de motifs.

On peut s'appuyer sur les algorithmes de recherche de motifs pour retrouver la structure d'une pièce musicale. L'équipe Algomus a travaillé sur l'analyse automatique de fugue basée sur l'inférence et la recherche de motifs [21].

La première étape consiste à utiliser les connaissances de la forme d'écriture en fugue pour aider à retrouver les éléments structurants. Par exemple, le sujet de la fugue est toujours exposé au début et se termine entre -8 et +7 notes de l'entrée de la voix suivante (voir la figure 2.3). Les motifs répondant à ce critère ont une probabilité d'être le sujet. La deuxième étape consiste à rechercher les différents motifs inférés de manière à déterminer lequel a le plus de probabilité d'être le bon.

2.1.2 Données monophoniques – Comparaison par programmation dynamique

De manière à trouver les occurrences d'un motif, la majorité des auteurs des articles cités précédemment utilisent des techniques qui permettent de calculer la similarité entre deux suites de notes.

Ces techniques ont été introduites par Mongeau et Sankoff dans [49]. Ils ont eu l'idée d'utiliser des règles classiques de calcul de similarité d'algorithmes du texte (opérations de

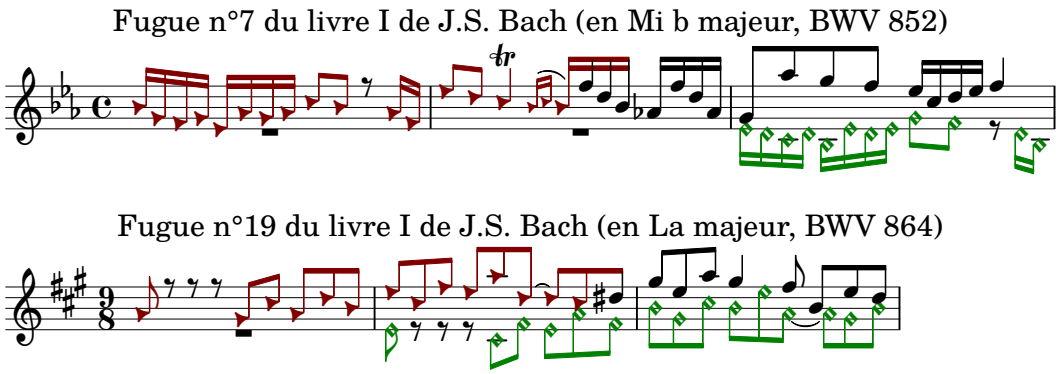


FIGURE 2.3 – Le sujet peut se terminer avant ou après le début d'une nouvelle voix. Les notes en forme de triangle représentent les sujets jouées par la soprano. Les notes jouées par l'alto sont en forme de losange.

suppression, *insertion*, *remplacement de caractères*) augmentées de règles propres à la musique (*fragmentation*, *consolidation*). Le but est de transformer une chaîne C en une chaîne C' en *minimisant le coût des opérations de transformation* (voir la figure 2.4). Mongeau et Sankoff ont mis au point un algorithme de *programmation dynamique* qui permet de déterminer la suite d'opérations à effectuer.

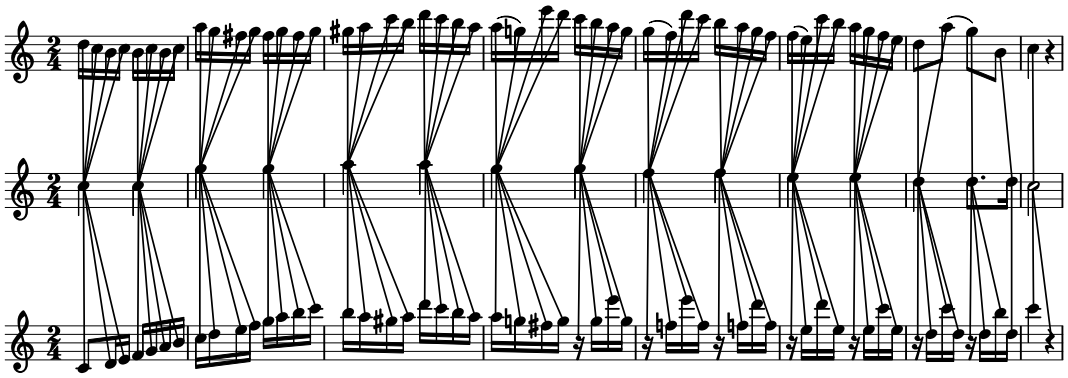


FIGURE 2.4 – Thème et variation de *Ah ! Vous dirai-je Maman* de W. A. Mozart (KV 265 – `https://imslp.org/wiki/Special:ImagefromIndex/56885`). Les traits reliant les notes entre les portées représentent les opérations de transformations effectuées pour obtenir un coût de similarité.

Remplacement. Cette opération consiste à remplacer une note A de C par une note B de C'. Pour calculer le poids de cette transformation, Mongeau et Sankoff utilisent les différences de hauteur et de durée entre les notes. Plus les notes sont consonantes et ont la même durée, plus le poids est faible. Mongeau et Sankoff proposent une échelle qui définit le poids à donner à chaque intervalle. La fonction $Intervalle(A, B)$ renvoie le poids associé à la différence de hauteur d'intervalle entre les notes A et B. Cette échelle a été définie en fonction de la consonance habituelle de la musique occidentale (un intervalle de quinte est plus consonant qu'un intervalle de quarte). Un coefficient pondérateur k est appliqué à la différence de durée des notes pour ne pas gommer l'importance de l'intervalle. Dans leur article, Mongeau et Sankoff choisissent un k égal à $0,348$. Les durées des notes sont données en rapport à la durée d'une ronde, par exemple la durée d'une noire est égale à $1/4$ de la durée d'une ronde. Sur la figure 2.4, le si de l'avant-dernière mesure de la première portée est remplacé par le ré

de l'avant dernière mesure de la seconde portée, le coût de transformation de cette opération est égale à $Intervalle(ré, si) + |(k * (1/8 - (3/16))| = 0,2 + | - 0,02| = 0,22$.

Suppression. Si une note est présente dans C mais pas dans C', cette opération permet de la supprimer. Le poids de cette transformation portera uniquement sur la durée de la note multipliée par le coefficient de pondération.

Insertion. À l'inverse de l'opération précédente, si une note est présente dans C' mais pas dans C, elle sera ajoutée à C. Le poids se calcule de la même manière que pour la suppression.

Enfin, les deux opérations suivantes considèrent particulièrement la durée des notes pour transformer une chaîne C en une chaîne C'.

Fragmentation. Elle consiste à subdiviser une note longue en plusieurs notes plus courtes. Le poids de cette opération correspond à la somme des intervalles d'une note C fragmentée en notes C', à laquelle on ajoute la différence de durée entre C et C' multipliée par un coefficient pondérateur. Sur la figure 2.4, le premier do de la seconde portée est fragmenté en trois notes (do, ré, mi) dans la première mesure de la troisième portée, ce qui donne un coût de transformation égale à $(Intervalle(do, do) + Intervalle(do, ré) + Intervalle(do, mi)) + |k * (1/4 - (1/8 + 1/16 + 1/16))| = (0 + 1 + 2) + |0,348 * 0| = 3$.

Consolidation. À l'inverse, cette méthode permet de regrouper plusieurs notes en une note plus longue. Le calcul du poids est identique à l'opération de fragmentation.

En utilisant les opérations définies ci-dessus et en prenant $k = 0,348$, nous pouvons calculer les coûts de similarités entre les différentes variations de la figure 2.4. Par exemple, le coût de similarité entre les deux premières portées (variation I et thème) est de $19,67$ et entre la deuxième et la troisième portée (thème et variation VII) est de $16,75$. La variation VII est ici évaluée comme plus similaire au thème que ne l'est la variation I.

La complexité en temps de l'algorithme dynamique qui permet de calculer la similarité entre deux chaînes est $O(n * m * k)$, où n et m sont les tailles des chaînes et k est le nombre maximum de notes autorisé pour les opérations de fragmentation et consolidation.

2.1.3 Données polyphoniques

L'analyse automatique de partition ne peut se restreindre aux données monophoniques pour plusieurs raisons. La première est qu'il arrive que les données musicales ne soient pas disponibles sous une forme monophonique. Par exemple, un enregistrement piano en MIDI ne sera pas forcément séparé en lignes monophoniques. La seconde raison est que la musique n'est pas uniquement faite de lignes monophoniques distinctes. Par exemple des pièces écrites pour des instruments polyphoniques (comme la guitare ou le piano) peuvent utiliser des *accords*. Enfin, même dans le cas de musique séparable en lignes monophoniques, des ambiguïtés peuvent subsister. Par exemple dans la Fugue #2 du livre I du Clavier bien tempéré de J.-S. Bach (en Do mineur, BWV 847), le contre-sujet passe de la première ligne monophonique à la seconde mesure 27 (voir la figure 2.5).

Nous avons vu précédemment qu'il existe des algorithmes qui peuvent retrouver des motifs dans des données monophoniques. La question est de savoir si l'on peut appliquer ces méthodes à des données polyphoniques et obtenir un résultat pertinent dans un temps raisonnable.

Pour se donner une idée de la difficulté de ce problème, reprenons l'exemple de la musique *Le petit cheval* de Georges Brassens. La figure 2.1 montre la partition et le piano-roll

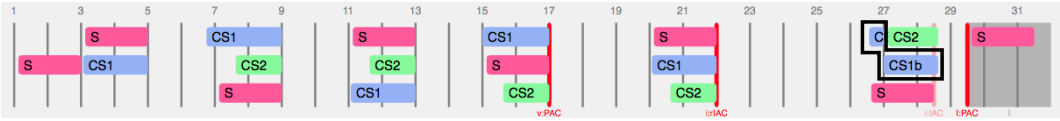


FIGURE 2.5 – Analyse de référence de la Fugue #2 du livre I du Clavier bien tempéré de J.-S. Bach (en Do mineur, BWV 847). Le contre sujet (noté CS1 et entouré) passe de la soprano à l'alto mesure 27 – `http://www.algomus.fr/fugues/bach/index.html`

associé, séparés en lignes monophoniques. Nous voyons sur le piano-roll que la seconde ligne monophonique est une copie de la première (à une variation près de la neuvième note). Pour bien voir maintenant la difficulté dans des données polyphoniques, intéressons-nous au même passage en plaçant l'ensemble des notes sur la même portée (voir la figure 2.6). Il devient très difficile de distinguer les deux lignes monophoniques identiques distinctes précédemment.

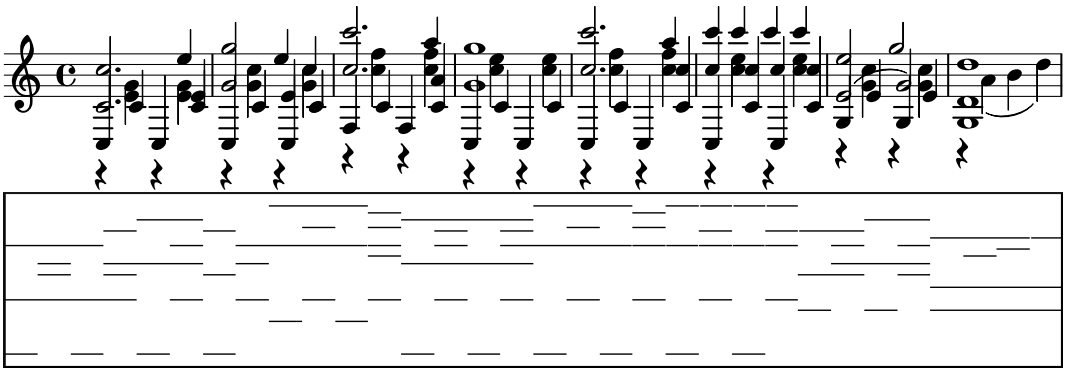


FIGURE 2.6 – Vue de l'ensemble de l'information sur une partition et un piano-roll des mesures 7 à 14 de la chanson *Le petit cheval* de Georges Brassens.

En ayant une approche plus mathématique, si nous souhaitons retrouver toutes les occurrences d'un motif constitué de deux éléments dans un ensemble de trois voix jouant trois notes, cela demandera déjà de faire 18 comparaisons (voir la figure 2.7). Néanmoins des chercheurs essayent de trouver une solution à ce problème complexe.

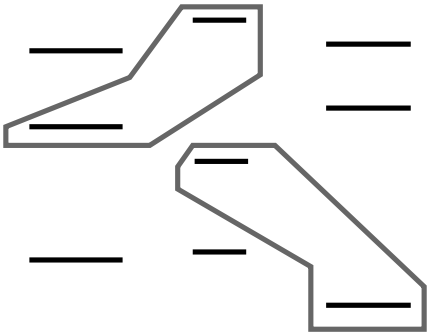


FIGURE 2.7 – Les notes entourées représentent 2 des 18 comparaisons possibles pour un motif de deux éléments

Plusieurs approches permettent de faire de la recherche de motifs dans des partitions polyphoniques (voir [3] pour un état de l'art de ces approches). L'une des approches consiste à utiliser des techniques basées sur les chaînes de caractères [60]. La plupart essayent de

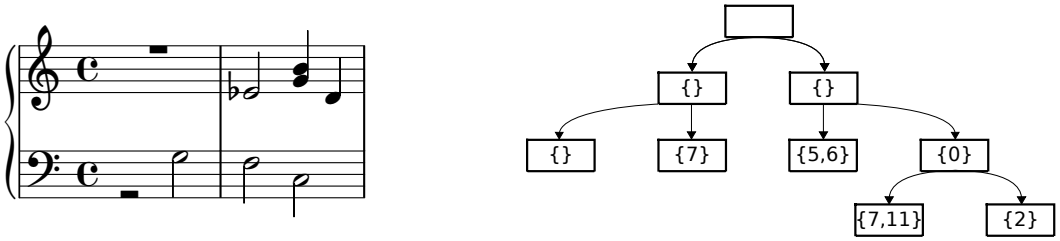


FIGURE 2.8 – La partition polyphonique de gauche est représentée par l'arbre de droite.

transposer des techniques utilisées dans des données monophoniques. Par exemple dans [60], Doraisamy et Rüger utilisent les *n-grammes* pour indexer, chercher et retrouver un ensemble de séquences dans des données polyphoniques. Un n-gramme est une sous-séquence de n éléments, l'ensemble s'obtient en faisant glisser une fenêtre de taille n sur la séquence à subdiviser. Dans [57], Rizo présente une adaptation d'une *structure arborescente* aux données polyphoniques (voir la figure 2.8). Son objectif est de retrouver des occurrences de motifs en comparant des arbres. Il définit la distance entre deux arbres comme le coût minimal d'opérations à effectuer pour transformer un arbre en un autre.

Une autre approche se base sur une *représentation géométrique* de la musique telle que le piano-roll [48]. Nous avons vu précédemment (voir la section 1.4) que nous pouvons utiliser un plan orthonormé pour représenter la musique. L'objectif principal des algorithmes *géométriques* [48, 66, 65, 63, 46] est de réussir à retrouver un motif M dans une partition P, en translatant M dans P. Pour la représentation piano-roll, un vecteur de translation est décrit par un couple (x, y) représentant le mouvement sur l'axe des abscisses et sur l'axe des ordonnés. Dans le cas d'une recherche exacte de motifs, il faut réussir à translater M dans P en appliquant le même vecteur à tout point de M. Dans la figure 2.9, une translation par le vecteur $(10, 52)$ à l'ensemble des points du motif permet de trouver une occurrence du motif dans l'ensemble de données.

Même si nous pouvions passer outre le problème de la complexité de la recherche de motif dans des données polyphoniques , la question principale est de savoir *comment définir la pertinence et le choix des motifs*. Pour apporter une réponse à cette question, il faut trouver un moyen d'organiser les données entre elles de manière à regrouper ensemble les notes selon des critères de perception musicale. Nous verrons une approche permettant d'apporter une réponse à cette question dans la section 2.2.

Étant donné la nature de la musique, il est important de pouvoir analyser des partitions polyphoniques, mais il est difficile de travailler avec ces données de manière pertinente et rapide. C'est pourquoi je présenterai dans la section suivante un moyen de contourner le problème en utilisant plusieurs approches pour *séparer les données polyphoniques en plusieurs sous-ensembles de notes*.

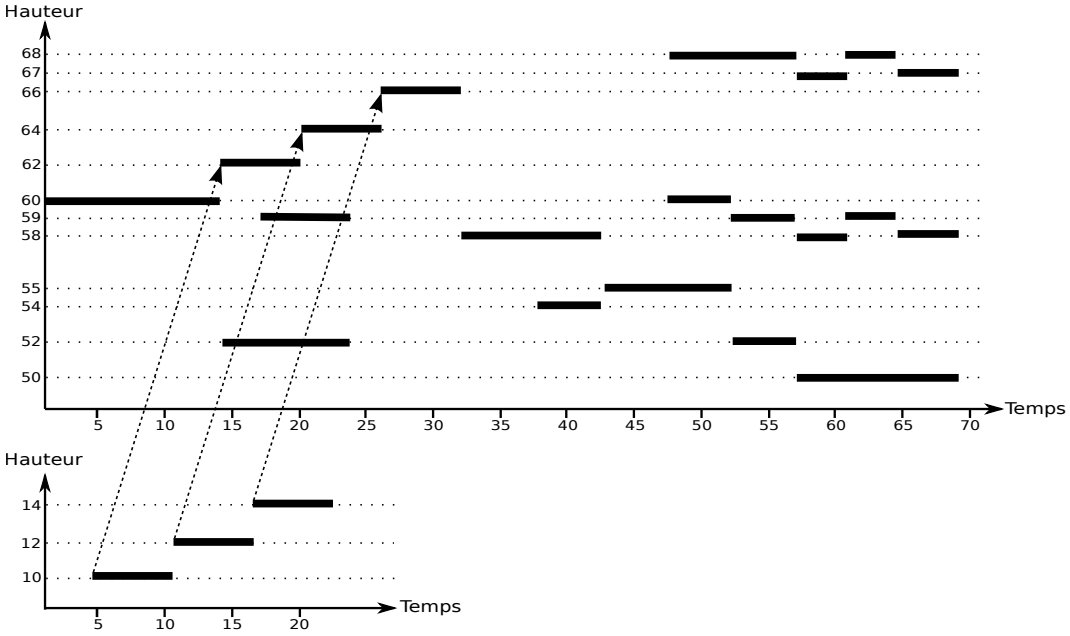


FIGURE 2.9 – Le motif représenté par le piano-roll du bas peut être retrouvé dans le piano-roll du haut si l'on applique le vecteur (10, 52) à l'ensemble des segments du motif.

2.2 Principes généraux de la séparation en voix et en streams

La polyphonie, en opposition à la monophonie, est la musique dans laquelle des notes peuvent être jouées en même temps. Cette polyphonie peut provenir de plusieurs instruments de musique jouant ensemble ou d'un seul instrument polyphonique comme par exemple le piano ou la guitare.

Nous avons vu précédemment que la recherche de motif dans des partitions polyphoniques est un problème compliqué. Une solution qui permettrait d'organiser l'information polyphonique serait de trouver un moyen de regrouper les notes selon des critères de perception musicale. Ces regroupements peuvent se faire dans des *voix* ou dans des *streams*. Séparer une partition polyphonique en ensemble de voix monophoniques ou en streams monophoniques/polyphoniques, nous permet d'apporter un élément de réponse à deux problèmes de polyphonie : la complexité de la recherche de motifs et la modélisation de l'information musicale. En effet, les regroupements de notes dans des voix ou dans des streams respectent des règles musicales qui permettent d'aider à l'analyse musicale.

La notion de *voix* peut prêter à de multiples interprétations. Nous avons vu dans la section 1.2 plusieurs types de polyphonies. Dans le cas d'une écriture polyphonique contrapuntique, la séparation en voix consiste à retrouver les différentes lignes mélodiques monophoniques. Dans la figure 2.10, nous pouvons décomposer cet ensemble de notes en trois voix : soprano (notes en forme de losange), alto (notes en forme de triangle), ténor (notes en forme de croix). Mais dans le cas d'une écriture en mélodie/accompagnement, il peut exister plusieurs manières de définir les voix. Certains chercheurs considèrent l'ensemble de l'accompagnement comme une seule « voix » même si plusieurs notes y sont jouées en parallèle.

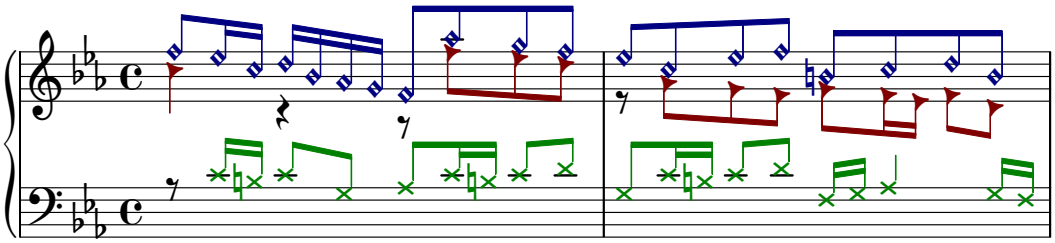


FIGURE 2.10 – Mesure 7 et 8 de la Fugue #2 du livre I du Clavier bien tempéré de J.-S. Bach (en Do mineur, BWV 847).

Séparer une partition polyphonique revient à la décomposer en plusieurs ensembles de notes. L'objectif est de séparer les voix en ensembles de notes telles que deux notes ne puissent être jouées simultanément dans la même voix. Si les ensembles de notes regroupés répondent aux deux critères suivants, nous parlerons de séparation en voix :

— les voix prédites doivent être monophoniques ;
— le nombre de voix prédit doit correspondre au nombre maximal de notes jouées simultanément.

Les séparations ne respectant pas ces deux contraintes seront vues comme des séparations en *streams*. Un stream est défini comme un ensemble cohérent de notes (hauteur des notes, durée, éloignement temporel...). La taille des streams est très variable, généralement de quelques notes à quelques mesures. Le nombre de streams peut également beaucoup varier, en étant toutefois quasiment toujours supérieur ou égal à celui des voix pour une même pièce.

Dans les figures 2.11 et 2.12, nous pouvons voir deux séparations possibles d'une même pièce polyphonique. Sur ces figures, pour savoir si deux notes appartiennent à la même voix ou stream, il suffit de regarder si elles ont la même forme (et la même couleur). La figure 2.11 présente une séparation en voix monophonique, chaque voix contenant au maximum une seule note jouée à chaque instant. Dans cet exemple, nous avons une séparation en 8 voix. La figure 2.12 montre une séparation en streams, le découpage en stream peut varier en fonction de la définition donnée au stream par les auteurs. Ici, nous avons choisi la définition donnée au stream par Rafailidis *et al.* dans [56]. Dans cet exemple, la pièce est séparée en quatre streams polyphoniques.

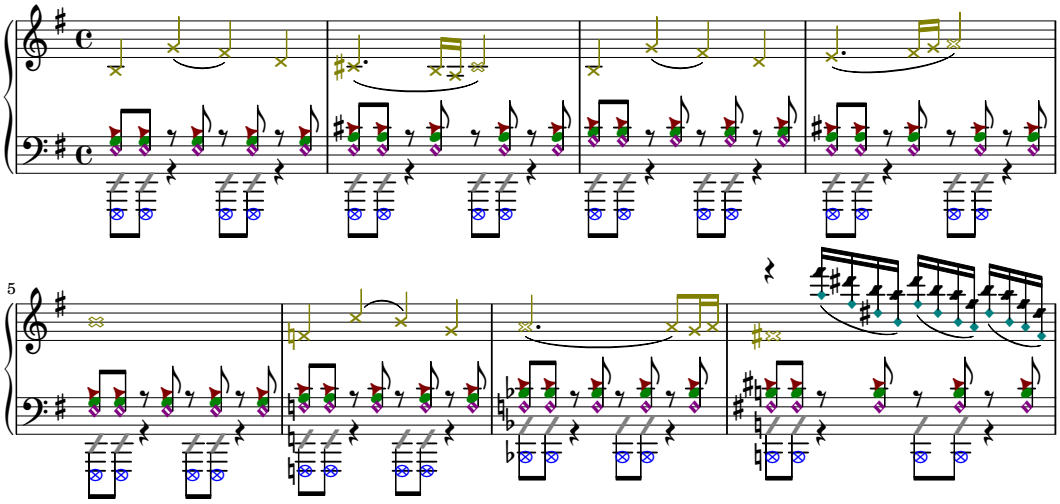


FIGURE 2.11 – Séparation en huit voix monophoniques d'un thème de la pièce *Dans les plaines d'Hyrule* de Zelda.

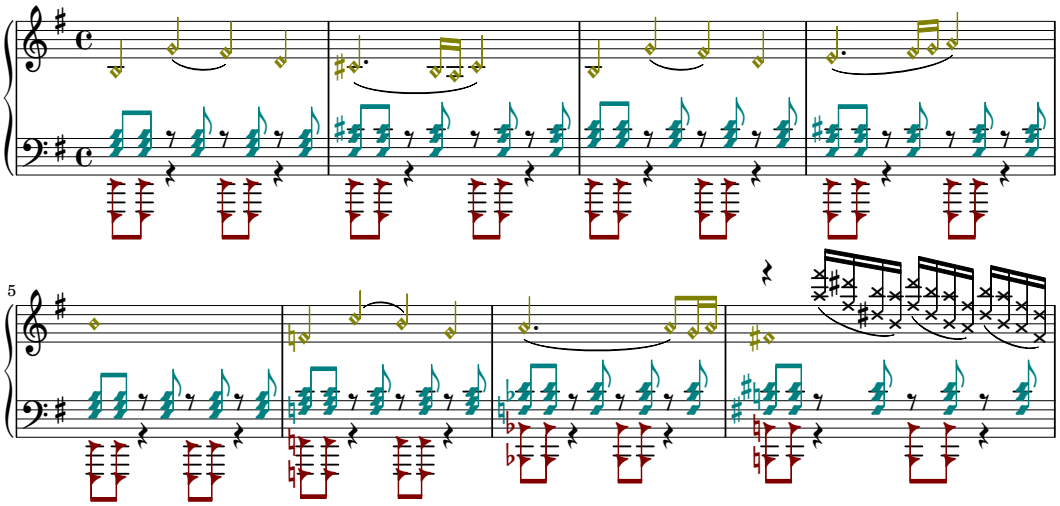


FIGURE 2.12 – Séparation possible en quatre streams d'un thème de la pièce *Dans les plaines d'Hyrule* de Zelda. La séparation est ici plus pertinente que celle de la figure 2.11. Le stream de la première portée correspond à la mélodie. Les deux streams de la seconde portée forment l'accompagnement. Dans la dernière mesure, le nouveau stream de la portée du haut est une *marche harmonique* (motif répété au moins deux fois à des degrés différents).

Dans la section suivante, je présenterai des algorithmes de séparation en voix et dans la dernière section, je présenterai des algorithmes de séparation en streams. Mon travail de thèse a consisté à améliorer ces algorithmes.

2.3 Séparation en voix

La séparation en voix consiste à obtenir un ensemble de lignes monophoniques en partant d'un fichier polyphonique. Pour réussir à séparer correctement les voix, la majorité des algorithmes utilise des principes de perception musicale décrits par David Huron [31] ou par Diana Deutsch [19]. Le principe le plus utilisé demande à minimiser la différence de hauteur entre deux notes consécutives d'une même voix.

2.3.1 Algorithme trivial

Comme vu dans le chapitre précédent, les voix chantées ou instrumentales peuvent être classées en fonction de leur registre, c'est-à-dire l'étendue des hauteurs de leurs notes. Les voix chantées sont traditionnellement réparties en voix allant de la soprano (voix la plus aigüe) à la basse (voix la plus grave). Un algorithme trivial de séparation de voix, mentionné par [37], assigne une hauteur de référence à chaque voix, puis assigne chaque note à la voix qui a la hauteur de référence le plus proche de la sienne. La première étape de cet algorithme consiste à détecter l'ensemble des débuts de notes où toutes les voix sont jouées en au même instant de manière à calculer une moyenne de hauteur de notes pour chaque voix (de la plus aigüe à la plus grave). Par exemple, dans la figure 2.13, pour obtenir la hauteur moyenne de la voix 1 (la plus aigüe), nous faisons la moyenne des hauteurs des notes les plus aiguës des débuts de notes où l'ensemble des voix sont jouées en parallèle (débuts de notes : 17, 20, 47, 52, 57, 61 et 64), ce qui donne une hauteur MIDI moyenne de $(62 + 64 + 68 + 68 + 67 + 68 + 67) \div 7 = 66,28$.

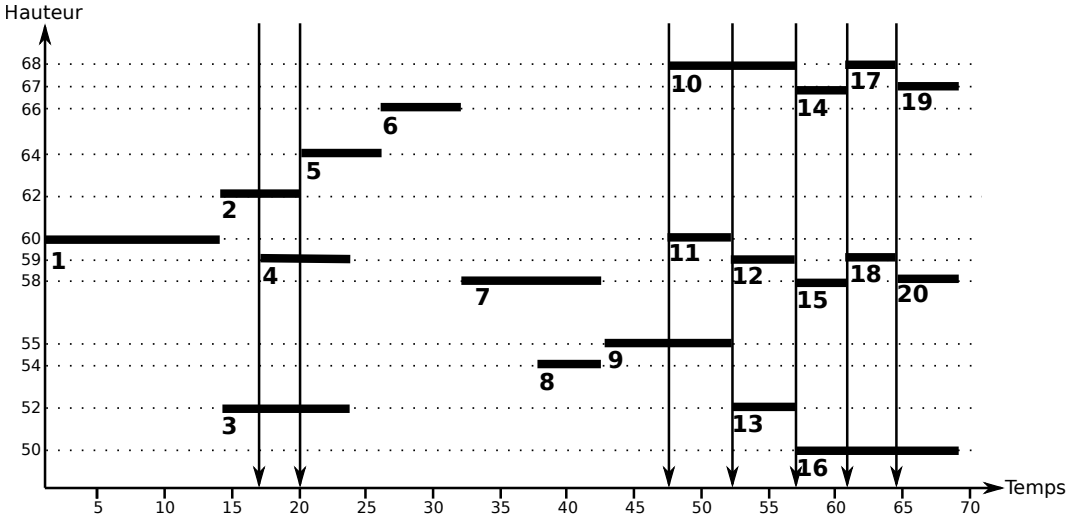


FIGURE 2.13 – Les flèches indiquent les instants où les trois voix sont jouées simultanément.

La deuxième étape consiste à assigner chaque note à la voix la plus proche en terme de hauteur de note. Un problème de respect de la monophonie peut apparaître si plusieurs notes jouées en parallèle sont toutes plus proches de la même voix. Il faut dans ce cas forcer l'annotation pour qu'il ne puisse pas y avoir deux notes jouées en parallèle dans la même

voix. Dans la figure 2.14, les notes 2 et 4 sont jouées ensemble du temps 17 au temps 20. Ces deux notes sont plus proches de la hauteur moyenne de la voix 2 et devraient donc être placées toutes les deux dans la voix 2. Pour ne pas avoir de voix polyphoniques dans la prédiction, il faut donc prédire la note 2 dans la voix 1.

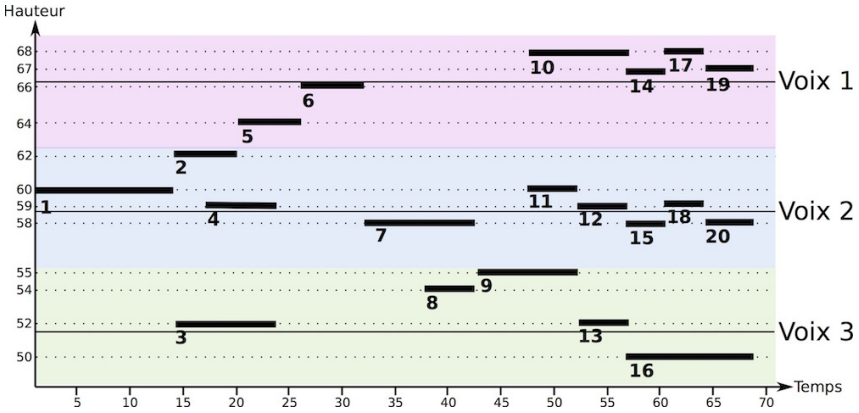


FIGURE 2.14 – Les zones de couleurs représentent les zones d'influence des différentes voix, la zone du haut correspond à la voix 1, la zone du milieu à la voix 2 et la zone du bas à la voix 3.

2.3.2 Approche par contigs : Algorithme CW

Pour séparer en voix, Chew et Wu [10] utilisent des principes définis par David Huron [31] pour expliquer l'écriture polyphonique. Ces principes sont les suivants :

— *p1* : Les voix sont monophoniques ;
— *p2* : À un moment, l'ensemble des voix doit être joué simultanément ;
— *p3* : Les intervalles sont minimisés entre les notes successives dans la même voix (proximité de hauteur de notes) ;
— *p4* : Les voix ne doivent généralement pas se croiser.

L'algorithme de Chew et Wu (noté CW) [10] se fait en deux étapes. La première étape consiste à découper la pièce en fonction du nombre de voix jouées en parallèle, la seconde étape consiste à connecter les fragments de voix en utilisant les principes présentés ci-dessus.

Contigs. La première étape découpe les données d'entrées en *blocs*. Ces blocs sont construits pour qu'à l'intérieur le nombre de notes jouées soit constant. Dans la figure 2.15, la pièce a été découpée en 7 blocs (B0, B1, B2, B3, B4, B5, B6), tous les blocs contiennent un nombre constant de notes jouées. Par exemple, le troisième bloc (B2) contient toujours trois notes jouées en même temps.

De plus, si une même note est jouée dans au moins deux blocs, il faut re-découper les blocs aux instants où la note commence et où elle termine. Dans la figure 2.15, la note 5 est présente dans les blocs B2 et B3, il faut donc couper B2 et B3 au niveau du début et de la fin de la note 5. Ces blocs ainsi obtenus sont appelés *contigs* (voir la figure 2.16).

Par construction, le nombre de notes jouées dans un contig est constant. À l'intérieur de ces contigs, les notes sont regroupées de la hauteur la plus haute à la hauteur la plus basse dans des *fragments de voix* (voir la figure 2.17).

Méthodes de connexions. L'objectif de la deuxième étape est de connecter ensemble les fragments provenant de contigs distincts. Les contigs contenant le nombre maximal de voix sont appelés *contigs maximaux*, sur la figure 2.16 les contigs C2, C3, C9, C10 et C11. Chew et Wu décident de commencer les connections par les contigs maximaux. L'argument utilisé

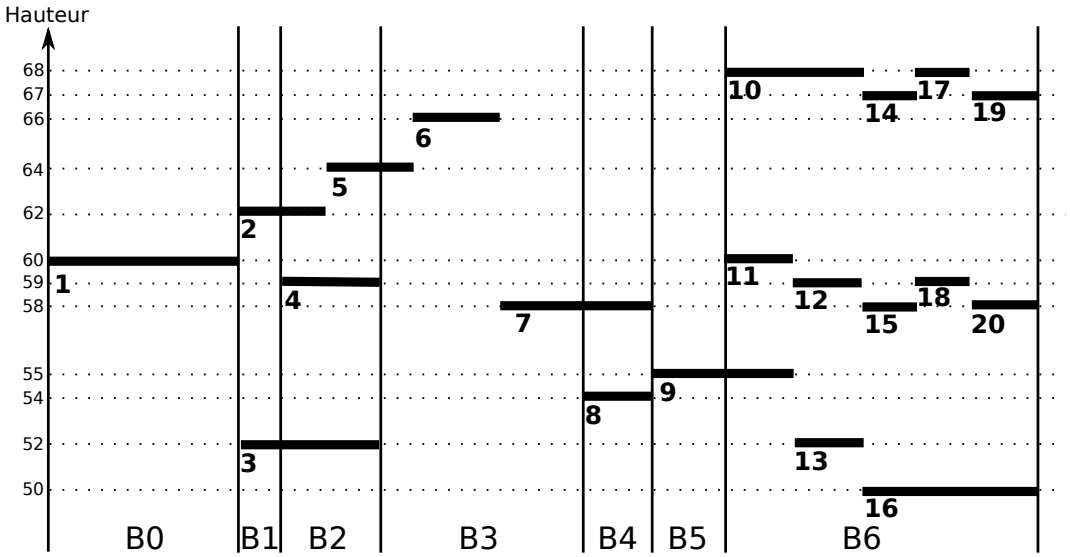


FIGURE 2.15 – La pièce est fragmentée en 7 blocs : B0, B1, B2, B3, B4, B5, B6. Le nombre de voix jouées en parallèle est constant à l'intérieur de chaque bloc.

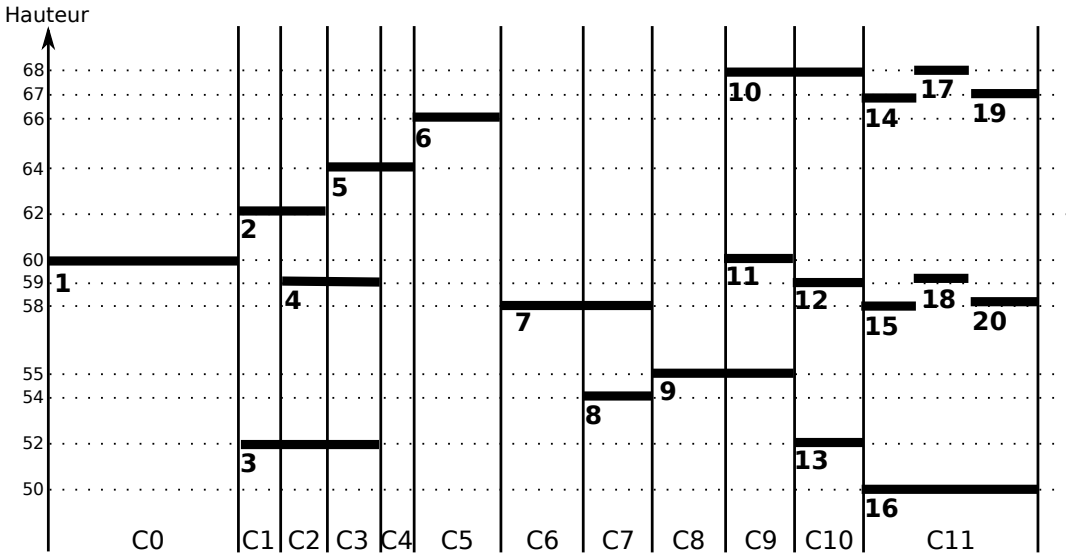


FIGURE 2.16 – La pièce est fragmentée en onze contigs de C0 à C10.

par Chew et Wu est qu'en respect du principe *p4*, les voix ne sont pas censées se croiser. Il y a donc une forte probabilité que l'ordre des fragments dans ces contigs corresponde aux bonnes voix.

Pour connecter les fragments, Chew et Wu choisissent d'appliquer le principe *p3* qui veut que les intervalles soient minimisés entre les notes successives dans la même voix (proximité de hauteur de notes). Lorsqu'on prend deux fragments de contigs voisins, Chew et Wu définissent un *poids de connexion* dépendant de la dernière note du fragment de gauche (noté n_1) et de la première note du contig de droite (noté n_2). Si n_1 et n_2 correspondent à la même note, ce poids vaut $-K$, où K est un très grand entier, sinon le poids est égal à

la différence absolue des hauteurs entre ces deux notes. Les fragments connectés entre deux contigs sont ceux qui minimisent le poids total de connexion (voir la figure 2.18).

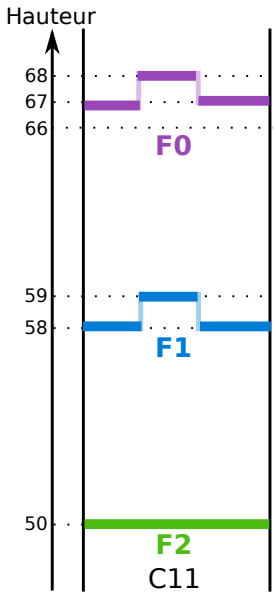


FIGURE 2.17 – Le contig C11 contient trois fragments de voix : F0, F1 et F2.

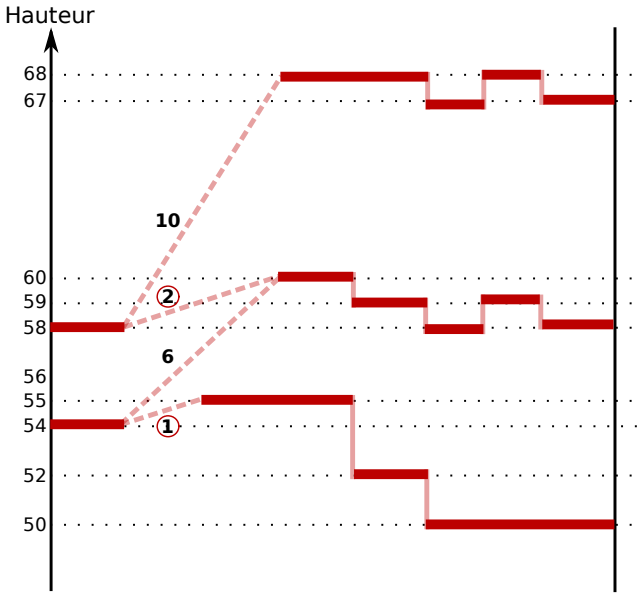


FIGURE 2.18 – La connexion entre contigs : les liens sélectionnés sont ceux minimisant le poids total ($2 + 1 = 3$).

Les figures 2.19a, 2.19b et 2.19c montrent un exemple des différentes étapes de connexions de l'ensemble de la pièce.

Chew et Wu ont utilisé un corpus d'œuvres de J.S. Bach : les deux livres du *Clavier bien tempéré* ainsi que les Inventions et Sinfonies. Ils obtiennent de bons résultats en réussissant à prédire dans la bonne voix plus de 88% des notes de ces pièces.

2.3.3 Approche par contigs : IMS

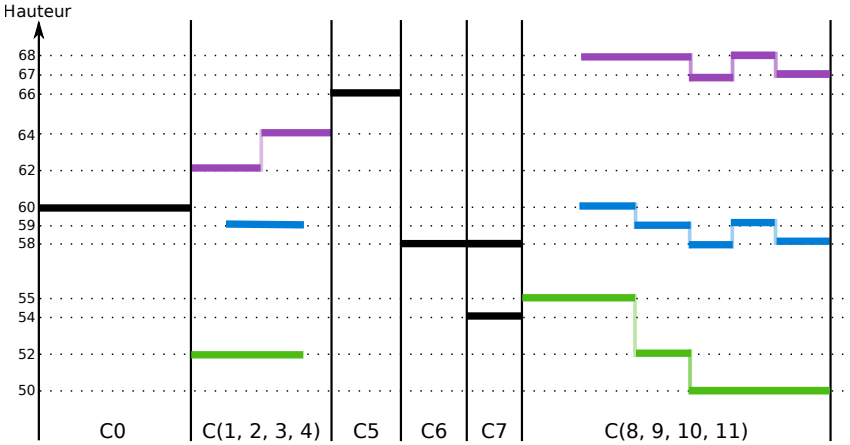
Ishigaki, Matsubara et Saito [33] proposent une modification de l'étape de connexion entre les contigs de l'algorithme de Chew et Wu pour améliorer leurs résultats. Nous noterons leur algorithme IMS. Leur motivation est que lorsqu'*une voix entre*, il n'y a pas souvent d'ambiguïté, contrairement à la sortie d'une voix qui peut-être plus difficile à déterminer. Au lieu de commencer par les contigs maximaux, ils choisissent de commencer uniquement par les contigs adjacents ayant un nombre de voix *croissant*. Par exemple, dans la figure 2.16, la procédure commence en connectant le C0 avec C1, C6 avec C7 et C8 avec C9. Le choix de connexion des fragments est identique à la méthode décrite dans l'algorithme CW. Après la connexion de tous les fragments provenant de deux contigs adjacents c_1 et c_2, nous obtenons un nouveau contig contenant le même nombre de voix que dans c_2 (voir la figure 2.20a). Le nom de ces voix est déterminé par le contig qui contient le plus de voix. Par exemple dans la figure 2.20b, la note la plus basse de C(6, 7) est prédite dans la deuxième voix alors que dans C(6, 7, 8, 9) cette même note est prédite dans la troisième voix.

La procédure décrite ci-dessus est réitérée tant que deux contigs adjacents ont un nombre croissant de voix. Si à la fin de cette procédure, il reste plus qu'un contig, ils sont connectés en utilisant la méthode de connexion originale de CW.

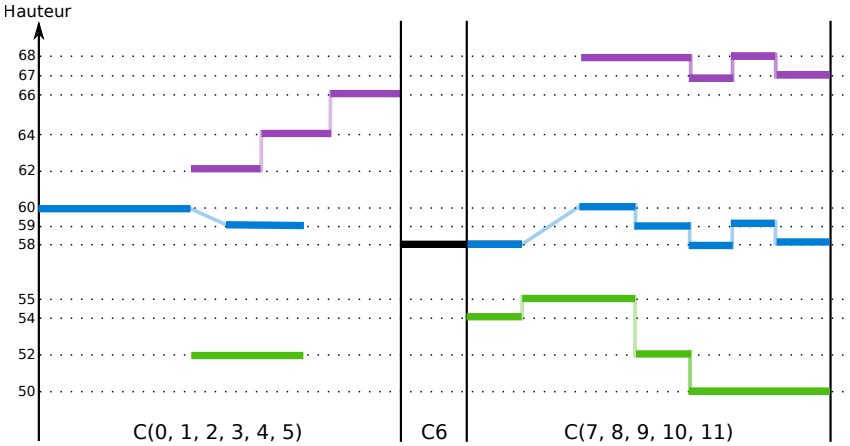
Sur la figure 2.20a, trois paires de contigs ont un nombre croissant de voix : C0 avec C1 (une vers deux voix), C6 avec C7 (une vers deux voix) et C8 avec C9 (une vers trois voix). Une fois ces combinaisons effectuées, le processus continue en combinant C(0, 1) avec C2 et C(6, 7) avec C(8, 9) (voir la figure 2.20b). Après la connexion de ces contigs, il n'y a plus de contigs ayant un nombre croissant de voix à connecter, l'algorithme continue alors en appliquant l'ordre de connexion de CW. Les contigs maximaux C(0, 1, 2), C3, C(6, 7, 8, 9), C10 et C11 sont alors connectés à leurs voisins (voir la figure 2.21a). Le processus termine en connectant C(0, 1, 2, 3, 4) avec C(5, 6, 7, 8, 9, 10, 11) (voir la figure 2.21b).

Nous voyons sur les figures 2.19c et 2.21b que l'ordre de connexion des contigs influence la séparation de voix. Par exemple, avec l'algorithme CW la première note est prédite dans la voix 2 (voir la figure 2.19c) et dans l'algorithme IMS cette même note est prédite dans la voix 1 (voir la figure 2.21b). Nous reviendrons plus en détail sur la manière de choisir l'ordre de connexion dans le chapitre 3.

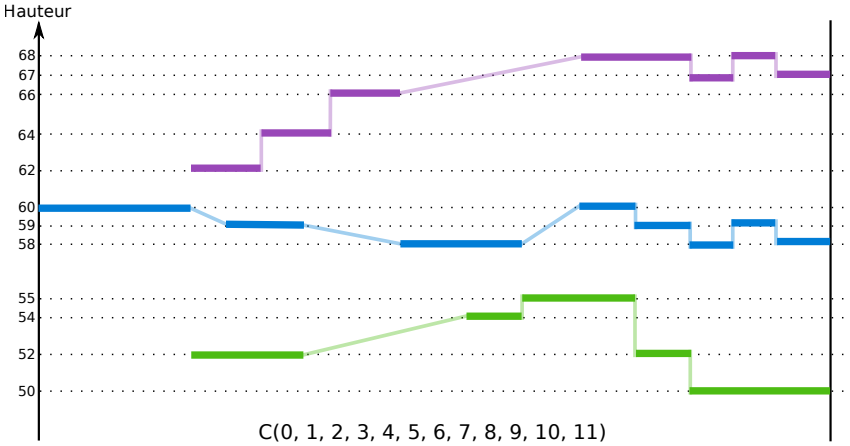
Ishigaki *et al.* utilisent le même corpus pour leur évaluation que Chew et Wu (le *Clavier bien tempéré* et les Inventions et Sinfonies de J.-S. Bach). Ils annoncent que leur algorithme obtient de meilleurs résultats que Chew et Wu en donnant une prédiction des notes dans les bonnes voix à plus de 92,1%.



(a) Les contigs maximaux C2 C3 C9 C10 et C11 sont connectés avec leur contigs voisins.

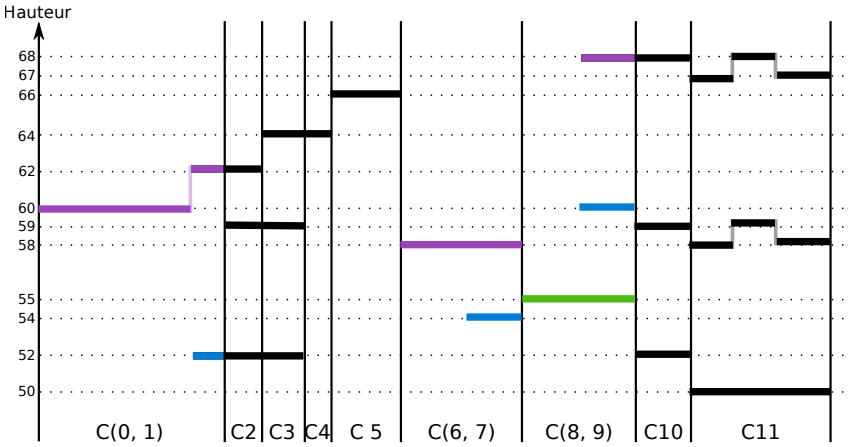


(b) C(1, 2, 3, 4) est connecté à C0 et C5, C(8, 9, 10, 11) est connecté à C7

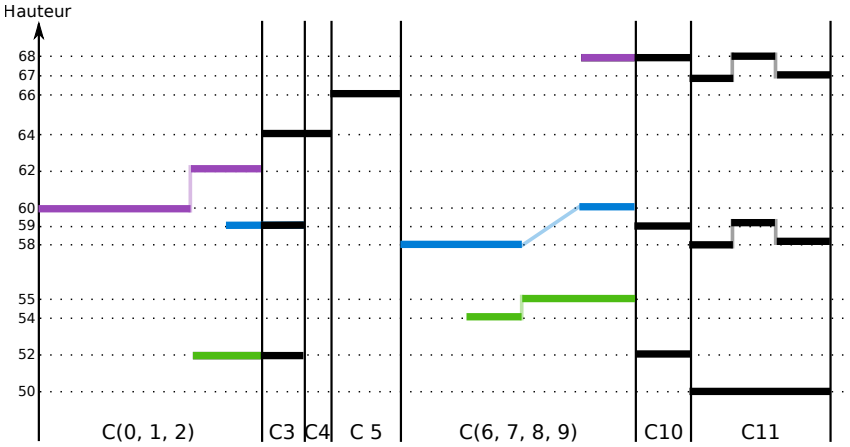


(c) Après la connexion de C(0, 1, 2, 3, 4, 5) avec C6 et C(7, 8, 9, 10, 11) avec C6, la séparation est terminée.

FIGURE 2.19 – Étapes de connexion des contigs qui respectent les règles de l'algorithme CW.

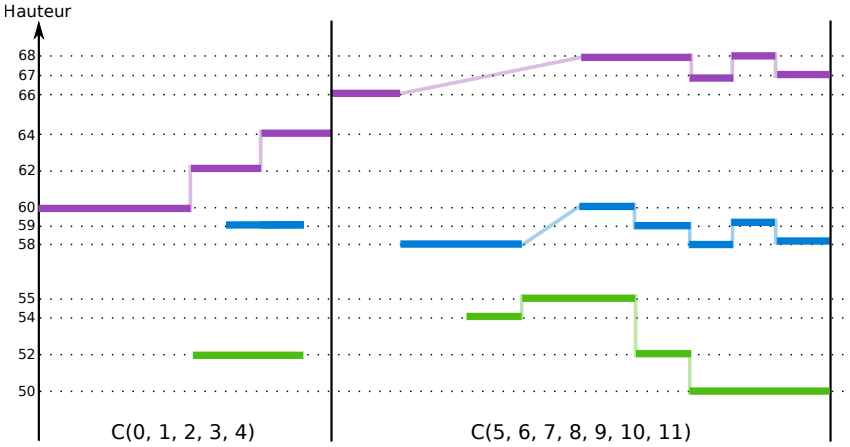


(a) C0 est combiné avec C1, C6 est combiné avec C7 et C8 est combiné avec C9.

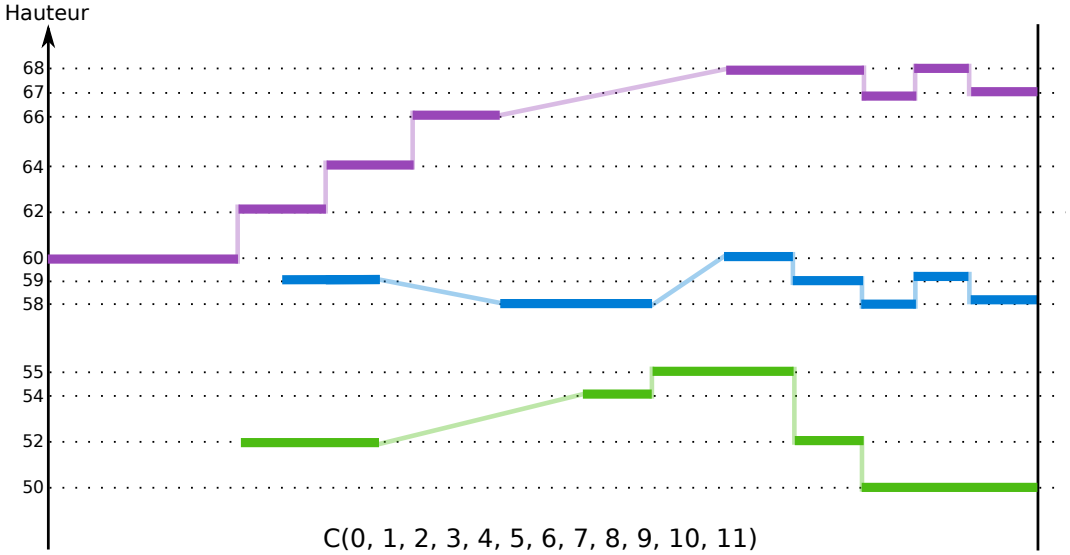


(b) C(0,1) est combiné avec C2 et C(6, 7) est combiné avec C(8, 9).

FIGURE 2.20 – Connexion des contigs en utilisant la règle de l'algorithme IMS : connecter les contigs adjacents ayant un nombre croissant de voix.



(a) C(0, 1, 2) est combiné avec C3, C3 est combiné avec C4, C(6, 7, 8, 9) est combiné à gauche avec C5 et à droite avec C10 et C10 est combiné avec C11.



(b) C(0, 1, 2, 3, 4) est combiné avec C(5, 6, 7, 8, 9, 10, 11).

FIGURE 2.21 – Il n'y a plus de contigs adjacents avec un nombre de voix croissant. L'algorithme IMS poursuit les connections des contigs en utilisant les principes de l'algorithme CW.

2.3.4 Approches par apprentissage

L'apprentissage consiste à adapter les choix de l'algorithme en fonction de situations connues. L'algorithme donné par Jordanous dans [34] utilise une *analyse statistique* de la structure des voix d'un corpus pour faire de la séparation de voix. Les deux premières étapes découpent la pièce en plusieurs blocs : l'algorithme commence par créer des zones contenant des moments où l'ensemble des voix est joué simultanément ; puis à l'intérieur de chacune de ces zones, l'algorithme conserve uniquement le moment qui contient la plus grande différence de hauteur entre sa note la plus aiguë et sa note la plus basse ; pour terminer les blocs sont centrés sur les moments conservés dans les zones définies précédemment (voir la figure 2.22 - ces blocs sont inspirés des contigs de l'algorithme CW). En se basant sur un corpus d'apprentissage, l'algorithme examine comment les hauteurs de notes apparaissent dans chaque voix et également comment les transitions entre ces hauteurs sont réparties. À partir de ces données, Jordanous définit la probabilité que deux notes appartiennent à la même voix. Son algorithme commence par connecter les notes à l'intérieur des blocs puis étend la connexion à l'ensemble des notes.

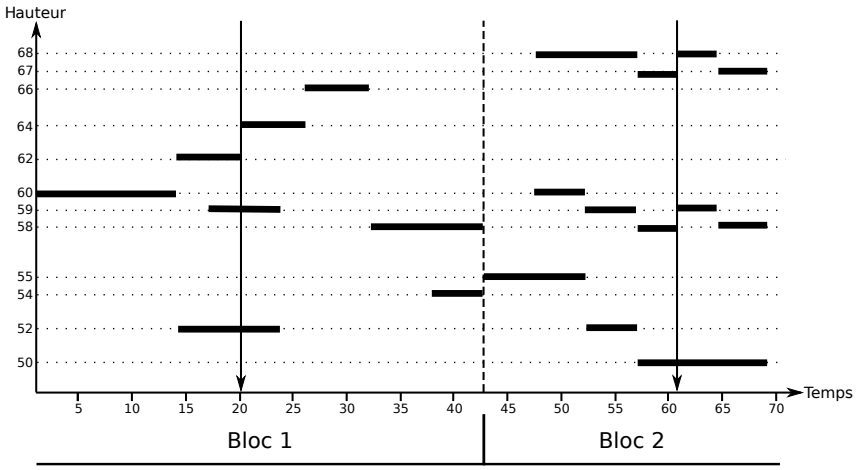


FIGURE 2.22 – Segmentation en blocs d'une pièce par l'algorithme de Jordanous *et al.*. Ce passage est découpé en deux blocs. Les flèches correspondent aux moments conservés par l'algorithme lorsque l'ensemble des notes sont jouées simultanément. La frontière entre les deux blocs se trouve au début de la note qui est la plus proche du milieu entre les deux flèches.

Dans [38], les auteurs utilisent quinze paramètres répartis en deux groupes pour définir la probabilité que deux notes soient prédites dans la même voix. Le premier groupe (cinq paramètres) s'intéresse à l'intervalle entre deux notes. Le second groupe (dix paramètres) prend en compte le temps pour savoir si deux notes proviennent de la même voix, par exemple plusieurs paramètres de ce groupe analysent la distance temporelle entre ces deux notes.

D'autres travaux [47, 18] proposent d'utiliser des *modèles de Markov cachés* pour faire de la séparation de voix. Dans le modèle proposé par McLeod *et al.* dans [47], les états représentent des voix monophoniques et la fonction d'émission associée à chaque état est une suite de notes. Une transition entre deux états est possible si toutes les notes du second état sont présentes dans le premier état ou dans la fonction d'émission du premier état. Pour calculer la probabilité de la transition entre deux états, McLeod *et al.* utilisent deux caractéristiques. La première s'intéresse à la différence de hauteur des notes entre les deux états. Sa particularité est qu'elle utilise une hauteur de référence associée à chaque voix. Cette référence est une moyenne des hauteurs des notes d'une voix qui donne plus d'importance aux

hauteurs des notes les plus récentes. La seconde s'intéresse à la distance temporelle entre les deux états. L'algorithme de Viterbi [67] est utilisé pour prédire la succession d'états (donc les voix monophoniques) qui correspond aux notes observées. Dans leur algorithme, les auteurs autorisent dans certains cas des notes qui se chevauchent à être placées dans la même voix, et cela de manière à pouvoir utiliser des données provenant d'enregistrement. En effet, lors de la capture d'une performance, il peut arriver qu'il y ait une différence entre la durée de la note jouée et la note sur la partition.

De Valk *et al.* [18] utilisent aussi un modèle de Markov caché. Les données d'entrée sont ici des suites d'accords et les états représentent les affectations de voix. Les accords sont représentés uniquement par leur hauteur en MIDI, et chaque affectation de voix est représentée par un vecteur de taille 4, leur corpus présentant au maximum quatre voix distinctes. La probabilité de transition entre deux états se base uniquement sur les hauteurs des notes. Là encore, la séquence optimale d'affectation des voix est calculée en utilisant l'algorithme de Viterbi [67].

De Valk *et al.* [18] proposent également un algorithme d'apprentissage basé sur un *réseau de neurones*. Leur objectif est d'associer une ou plusieurs voix à chaque note. Les neurones d'entrée représentent 32 caractéristiques qui permettent de décrire précisément chacune des notes d'une pièce. Ces caractéristiques peuvent se classer en quatre sous-ensembles. Le premier sert à décrire une note (hauteur, durée...), le second à décrire l'accord dans lequel se trouve la note (nombre de notes au-dessus, différence de hauteur de notes...), le troisième donne des informations relatives aux notes voisines (durée de différence avec la note précédente, différence de hauteur avec la note suivante...) et le dernier sert à savoir si la note est déjà associée à une voix. Les neurones de sortie représentent chacun une des 4 voix possibles d'affectation.

2.4 Séparation en streams

Toutes les musiques polyphoniques ne sont pas composées de voix. Par exemple beaucoup de musiques pour instruments polyphoniques incluent des *accords* ayant un nombre de notes variables. C'est pourquoi, dans cette partie, nous étudierons des algorithmes qui séparent des partitions polyphoniques en *streams*. Un stream est défini comme un ensemble cohérent de notes (hauteur des notes, durée, éloignement temporel...). La taille des streams est très variable, généralement de quelques notes à quelques mesures. Le nombre de streams peut également beaucoup varier, en étant toutefois quasiment toujours supérieur ou égal à celui des voix pour une même pièce.

2.4.1 Temperley

L'algorithme proposé par Temperley [62, p. 85 à p. 114] extrait des streams d'une partition polyphonique en respectant plusieurs contraintes. La première est la proximité de hauteur : deux notes contiguës ayant des hauteurs proches sont placées dans le même stream (ce principe est équivalent au principe *p3*). La seconde est temporelle : quand il y a une longue pause entre deux notes, la deuxième note est placée dans un nouveau stream. La troisième autorise la duplication de notes dans deux streams (pour ne pas qu'il y ait de croisement dans les streams, *p4*).

La figure 2.23 montre un résultat de séparation en streams en suivant les règles de Temperley. Le numéro indiqué sous chaque note correspond à son numéro d'apparition dans la pièce. Les notes reliées entre elles par un segment gris sont placées dans le même stream, nous avons ici cinq streams.

La prédiction de la note 1 peut être assez ambiguë. En effet, elle est plus proche au niveau temporel de la note 2 que de la note 4, mais plus proche dans l'intervalle de hauteur avec la note 4 qu'avec la note 2. C'est pourquoi en respect de la troisième contrainte, elle est prédite dans les streams 1 et 3. Les notes 6 et 10 sont éloignées par une longue pause, ce qui a pour conséquence de placer la note 10 dans un nouveau stream. De même pour les streams 2 et 4.

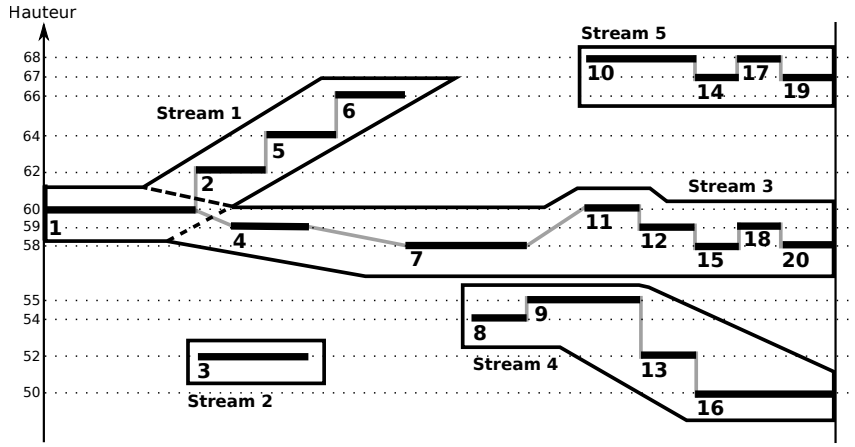


FIGURE 2.23 – Résultat d'une séparation en streams suivant les principes de Temperley.

2.4.2 Madsen et Widmer

Pour être sûr d'obtenir des streams monophoniques, les auteurs de [44] formulent deux règles : chaque note est assignée à un unique stream et un stream ne peut pas contenir de notes qui se chevauchent. Pour assigner les notes aux streams, leurs algorithmes respectent trois principes : minimiser les différences de hauteurs entre les notes d'un même stream (principe de proximité de hauteur de notes, *p3*), minimiser le nombre de stream, minimiser le nombre de pauses dans un stream (créer un nouveau stream plutôt qu'avoir une longue pause dans un stream). Ces principes sont assez proches de ceux de l'algorithme de Temperley présenté ci-dessus. La plus grande différence vient du fait que Madsen et Wilmer autorisent le croisement de voix.

2.4.3 Segments de Streams

Dans [55], les auteurs définissent une nouvelle notion : *le segment de stream*. Un segment de stream est un ensemble cohérent de notes relativement court, leur algorithme (que j'appellerai *Stream Segment*) permet de définir la taille souhaitée des segments de streams. Il groupe les notes en se basant sur les regroupements des k-plus-proches-voisins. L'algorithme commence par calculer une matrice de distance, qui indique pour chaque possible paire de notes si elles doivent être placées dans le même stream. La distance entre deux notes est calculée en prenant en compte la synchronisation (deux notes commençant au même moment et ayant la même durée), la proximité de hauteur et de temps (*p3* entre autres critères) ; puis pour chaque note, la liste de ses k-plus-proches-voisins est établie.

Dans l'exemple de la figure 2.24, l'algorithme Stream Segment découpe la pièce en huit fragments de streams. Les notes 14, 15, 17, 18, 19 et 20 sont prédites dans le même stream. En effet, l'ensemble des notes 14, 17 et 19 peux être vu comme une transposition des notes 15, 18 et 20, ce qui forme une succession d'accords.

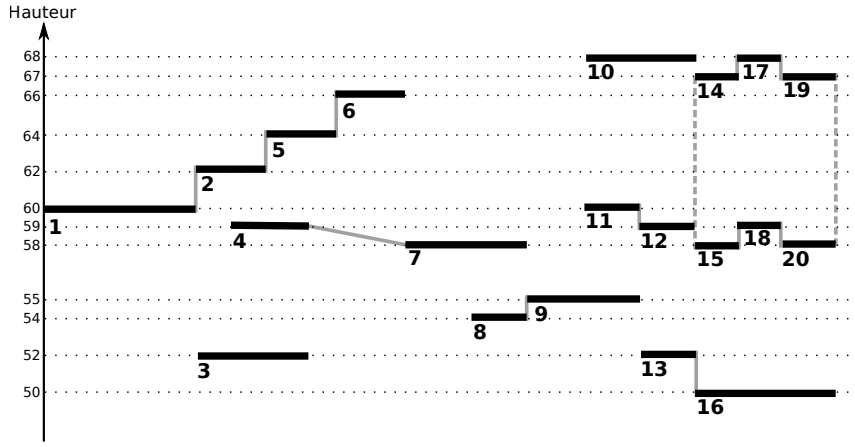


FIGURE 2.24 – Stream Segment prédit huit fragments de streams.

2.4.4 Rafailidis *et al.*

Dans [56], les auteurs proposent une amélioration de l'algorithme *Voice Integration/Segregation Algorithm* (VISA) proposé par Karydis *et al.* [35]. Les règles de séparation sont identiques à celles de l'algorithme Stream Segment (proximité horizontale et verticale) mais au lieu de

séparer la pièce en de courts segments de streams, ils choisissent d'obtenir des streams qui peuvent durer du début à la fin de la pièce.

Pour cela, la pièce est d'abord découpée en *clusters*. La définition du cluster est assez proche de celle du bloc de l'algorithme CW, dans un cluster le nombre de notes jouées doit être constant et toutes les notes doivent commencer à l'intérieur du cluster (voir la figure 2.25).

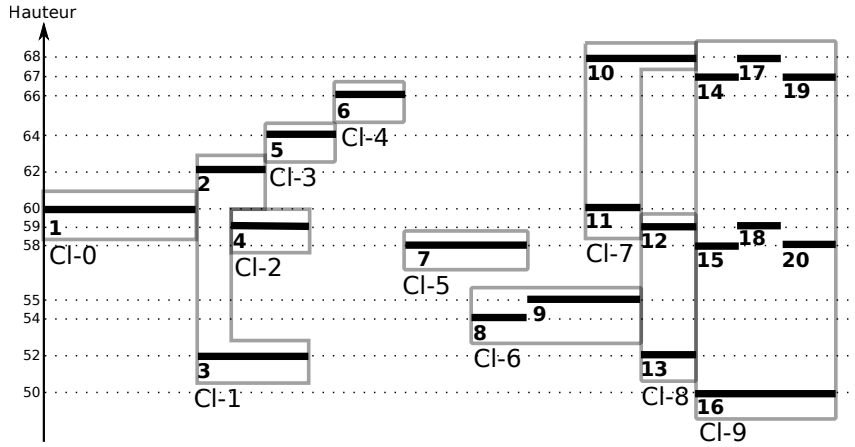


FIGURE 2.25 – La pièce est découpée en dix clusters, notés de Cl-0 à Cl-9.

Puis à l'intérieur des clusters les notes sont regroupées en fragments de streams selon les principes de proximité horizontale : deux notes seront placées dans le même stream si elles commencent au même moment et ont la même durée (voir la figure 2.26).

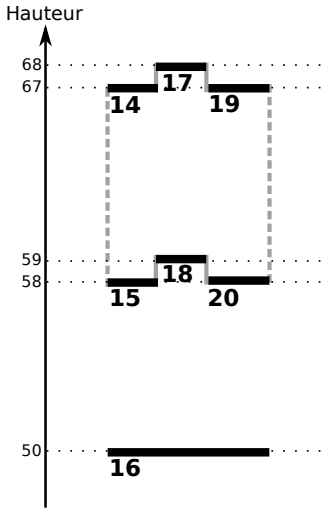


FIGURE 2.26 – Le cluster Cl-9 contient deux streams, le premier contient les notes 14, 15, 17, 18, 19, 20 et le second la note 16.

Les streams des différents clusters sont enfin regroupés selon des règles de préférences prenant en compte pour 25% l'homogénéité des streams, 50% la différence de hauteur et 25% la différence de temps entre le début du premier stream et le second. Deux streams contenant le même nombre de notes, une faible distance d'intervalle et joués consécutivement auront plus de chance d'être fusionnés ensemble que deux streams ne contenant pas le même nombre

de notes, avec un intervalle très important et éloignées dans le temps. En suivant ces règles de préférences, sur la figure 2.27, le stream 1 (S1) est fusionné au stream 4 (S4).

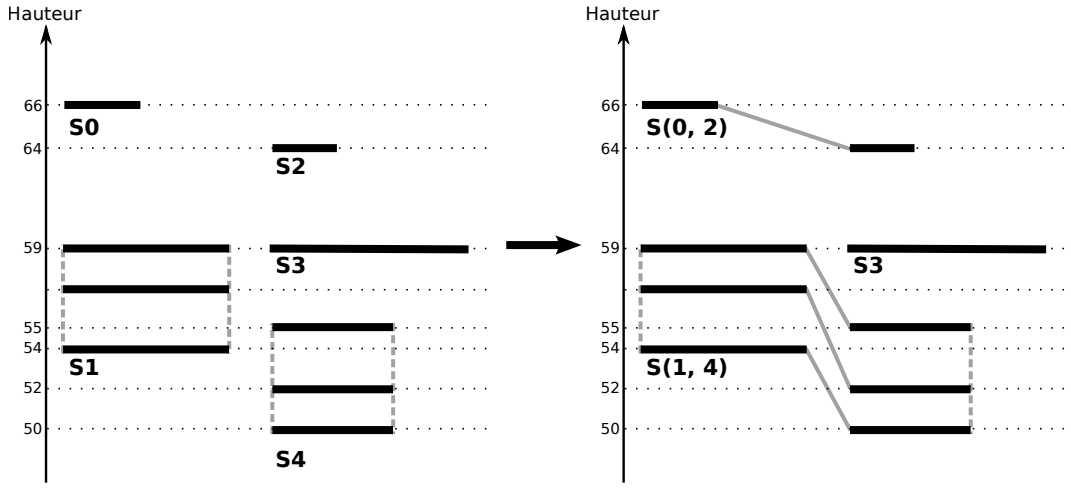


FIGURE 2.27 – Les streams sont connectés en respectant les règles de préférences de l'algorithme VISA.

2.4.5 Apprentissage

Dans [25] les auteurs ont utilisé un système à réseau de neurones pour faire de la séparation en streams. La première étape consiste à regrouper ensemble les notes commençant au même moment (voir la figure 2.28) dans ce qui est appelé par les auteurs une *suite d'accords*. La seconde étape consiste à prédire le stream de provenance de chaque note constituant les accords : une note est placée dans un stream si elle maximise pour ce stream un score composé de six caractéristiques (probabilité qu'un stream commence, proximité de hauteur, continuité temporelle, pseudo-polyphonie, éloignement tonal et probabilité que la note compose un accord).

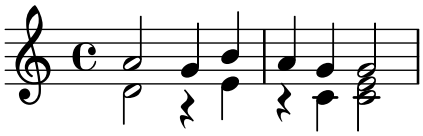


FIGURE 2.28 – Ici la pièce est segmentée en une suite de six d'accords : (1,2), (3), (4,5), (6), (7,8) et (9,10,11).

2.4.6 CW-Contigs

Dans l'article de Chew et Wu, la première étape consiste à découper une pièce en contigs. Pour rappel dans chaque contig, le nombre de notes jouées en parallèle est constant (voir la figure 2.16). À l'intérieur de ces contigs, les notes sont regroupées de la hauteur la plus haute à la hauteur la plus basse dans des fragments de voix. Ces fragments forment des groupes de notes cohérents, que nous pouvons considérer comme des streams. La première étape de l'algorithme CW peut donc être assimilée à un algorithme de séparation en streams que j'appellerai CW-Contigs. Nous pouvons voir sur la figure 2.29 que l'algorithme CW-Contigs produit beaucoup de streams, dans cet exemple 16, notés de S0 à S15 (voir la figure 2.29).

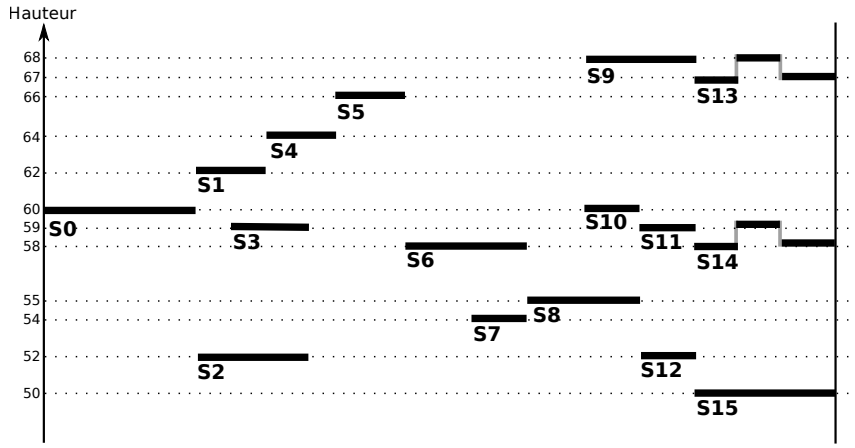


FIGURE 2.29 – Les fragments des contigs de l'algorithme CW peuvent être considérés comme des streams.

Chapitre 3

Évaluer la séparation en voix et en streams

Dans les sections 2.3 et 2.4, j'ai présenté deux manières différentes d'appréhender la séparation de partitions polyphoniques en un ensemble d'objets monophoniques ou polyphoniques. Il s'agit de la *séparation en voix* et de la *séparation en streams*. Nous avons pu voir que même si les objectifs de cette séparation sont différents, l'ensemble de ces algorithmes partagent comme critère principal de regroupement celui de la *proximité des hauteurs*.

L'objectif de ce chapitre est de montrer que les streams et les voix sont les deux facettes d'un même problème et que nous pouvons utiliser des métriques identiques pour les comparer.

Dans la section 3.1, je présente trois méthodes de mesures de qualité de la séparation en voix et en streams.

Dans la section 3.2, je propose une évaluation des méthodes de mesure de qualité proposées. Je commence par présenter le corpus utilisé pour réaliser cette étude. Puis, je continue en expliquant les différences qui existent entre les algorithmes originaux et mes implémentations. Par la suite, je discute des résultats des mesures d'évaluation sur ces algorithmes. Je termine en montrant la proximité entre les algorithmes de séparation en streams et les algorithmes de séparation en voix.

Le travail présenté dans ce chapitre a fait l'objet de l'article [26], présenté à la conférence internationale ISMIR 2015.

3.1 Évaluer des algorithmes de séparation en voix et en streams

Dans cette section je souhaite trouver un moyen d'évaluer les algorithmes de séparation en voix et les algorithmes de séparation en streams sur un pied d'égalité. Je suppose dans cette partie que je dispose de *fichiers de référence* donnant la bonne séparation en voix. Je reparlerai de ce problème dans la section 3.2.1 en présentant mon corpus d'évaluation.

Rappelons d'abord les définitions des vrais/faux positifs/négatifs permettant de mesurer la qualité des résultats d'évaluations (voir la figure 3.1) :

— Les *vrais positifs* sont les éléments qui appartiennent à l'ensemble des éléments prédits et à l'ensemble des éléments du fichier de référence ;

— Les *faux positifs* sont les éléments qui appartiennent à l'ensemble des éléments prédits mais pas à l'ensemble des éléments du fichier de référence ;

— Les *faux négatifs* sont les éléments qui appartiennent à l'ensemble des éléments du fichier de référence mais pas à l'ensemble des éléments prédits ;
— Les *vrais négatifs* sont les éléments qui n'appartiennent ni à l'ensemble des éléments du fichier de référence ni à l'ensemble des éléments prédits.

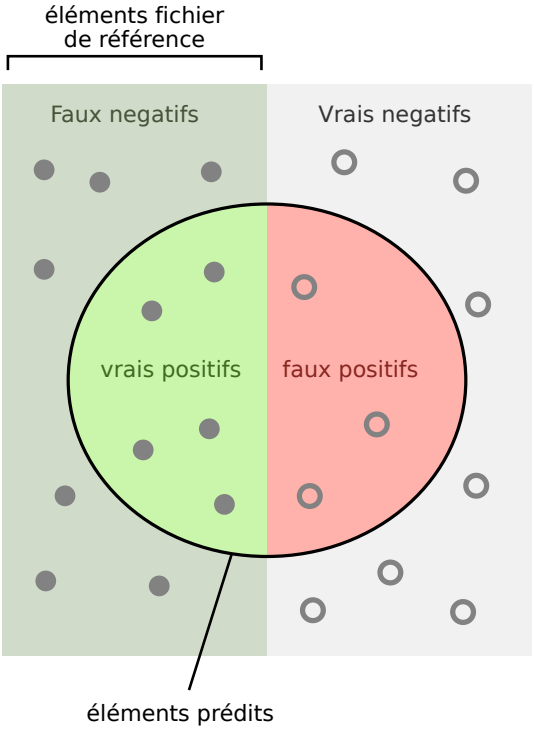


FIGURE 3.1 – Vrais positifs (True Positives en anglais, TP), faux positifs (False Positives en anglais, FP), vrais négatifs (True Négatives en anglais, TN) et faux négatifs (False Negatives en anglais, FN) - Par Walber (Own work) [CC BY-SA 4.0 (`http://creativecommons.org/licenses/by-sa/4.0`)], via Wikimedia Commons

3.1.1 Évaluation note-à-note

Considérons tout d'abord la séparation en voix. La première question que l'on peut se poser lorsque l'on souhaite évaluer des algorithmes de séparation de partitions polyphoniques est de se demander si nous avons été capable de prédire correctement chaque note dans la bonne voix. Nous pouvons comparer le numéro de voix que porte une note dans le fichier de référence avec le numéro qu'elle porte dans la prédiction (voir la figure 3.2).

Pour cela je définis une métrique appelée *précision note-à-note* (en anglais, *note precision*, NPR) qui est le ratio entre le nombre de notes correctement prédites (dans la bonne voix) et le nombre total de notes. Autrement dit, NPR = (Vrais Positifs) / (Vrais Positifs + Faux Négatifs). Sur la figure 3.2, les notes étiquetées A1, B2 ou C3 sont des vrais positifs et les notes étiquetées B3 sont des faux positifs.

Dans l'article [10], les auteurs définissent la mesure AVC (Average Voice Consistency) calculée de la même manière que NPR à la différence que pour une pièce ou un corpus, leurs résultats sont des moyennes par voix et non pas un ratio sur le nombre total de notes d'une pièce ou d'un corpus. Nous pouvons voir les différences entre les deux calculs dans la figure 3.3. Dans cet exemple, deux voix (A et B) avec peu de notes (3) sont prédites correctement et une voix (C) contenant beaucoup de notes (18) n'est pas prédite correctement. La

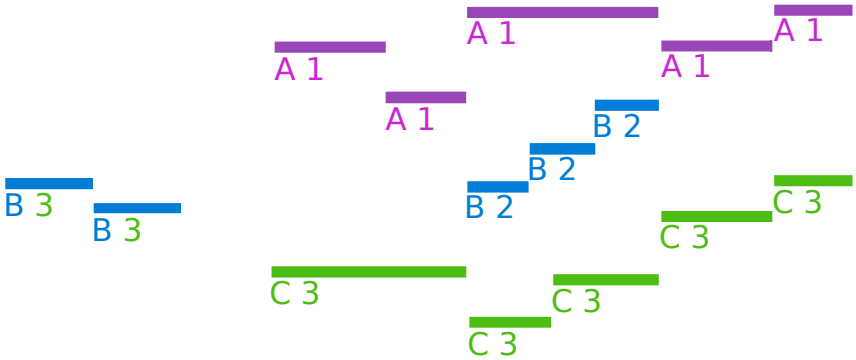


FIGURE 3.2 – La lettre sous la note donne la voix à laquelle elle est associée dans le fichier de référence et le chiffre sous la note correspond à la voix dans la prédiction. Une note est prédite correctement si le chiffre de la prédiction correspond à la lettre (dans l'ordre lexicographique). Par exemple la note A1 est prédite correctement mais pas la note B3. Ici la polyphonie se décompose en 3 voix dans l'analyse de référence (A, B et C) et en 3 autres voix par un algorithme de séparation de voix (1, 2 et 3). NPR = 13/15 = 87%.

différence entre les calculs de AVC et de NPR dépasse les 50%. Cette si grande différence vient du fait que, pour le calcul par moyenne, toutes les voix sont équivalentes quel que soit le nombre de notes qu'elles contiennent. Ainsi, en prenant l'ensemble des notes pour le calcul du ratio, nous ne gommons pas l'importance de la taille d'une voix ou d'une pièce.

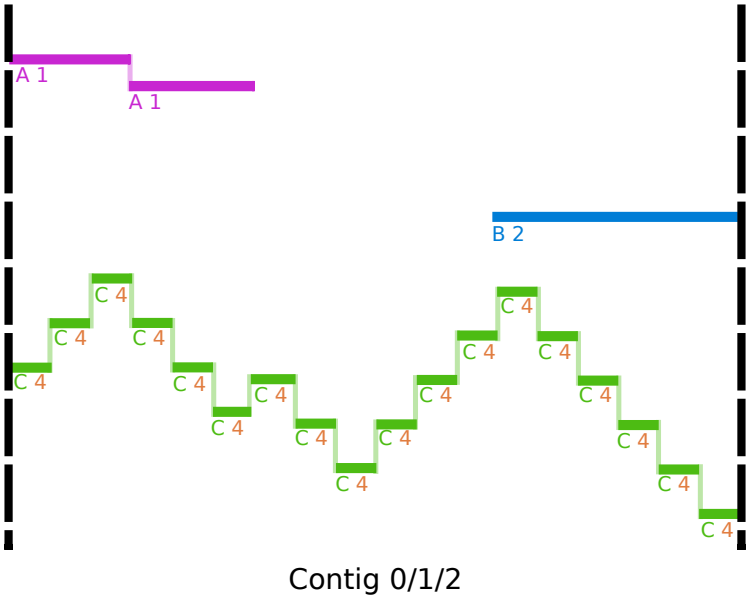


FIGURE 3.3 – Les notes des voix A et B ont été prédites correctement (3 Vrais Positifs), l'ensemble des notes de la voix C ont été prédites dans la voix 4 au lieu de la voix 3 (18 Faux Négatifs). AVC $= (100\% + 100\% + 0\%)/3 = 66\%$; NPR $= 3/21 = 14\%$.

Calculer NPR requiert de savoir *à quelle voix de la prédiction correspond une voix du fichier de référence*. Si dans les algorithmes de séparation de voix, le nombre de voix prédit correspond à celui du fichier de référence, cela n'est pas le cas pour les algorithmes de séparation en streams qui prédisent plus de streams qu'il n'y a de voix (voir la figure 3.4).

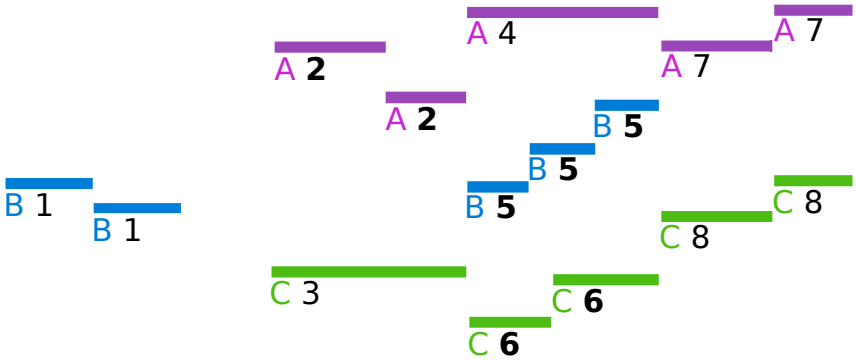


FIGURE 3.4 – Prédiction de 8 streams. Pour l'évaluation note-à-note, une voix est équivalente à un seul stream, c'est pourquoi seuls les streams ayant le plus de notes en commun sont conservés (2, 5, 6) : NPR = 7 / 15 = 47%.

C'est pourquoi une solution consiste à comparer chaque voix (ou stream) prédite par l'algorithme à *la voix la plus similaire du fichier de référence* (celle avec laquelle elle partage le plus de notes).

C'est ce que l'on peut voir sur la figure 3.4. Nous avons trois voix (A, B et C) dans le fichier de référence et huit streams (1, 2, 3, 4, 5, 6, 7 et 8) prédits, nous associons à chaque voix le stream avec lequel elle partage le plus de notes. Le stream 2 est associé à la voix A, le stream 5 est associé à la voix B et le stream 6 est associé à la voix C. Les autres streams (qui représentent plus de la moitié des notes) ne sont associés à aucune voix. C'est pourquoi la métrique NPR ne donne un résultat que de $7/15 = 47\%$.

L'idée de vouloir comparer directement chaque note prédite à une voix du fichier de référence ne permet donc pas de comparer équitablement les algorithmes de séparation en voix et les algorithmes de séparation en streams.

L'évaluation de la précision note-à-note tend à pénaliser de la même manière des séparations qui n'ont pas le même sens musical. Dans la figure 3.5, une note sur deux n'est pas prédite dans la bonne voix, aucune cohérence ne ressort de cette séparation.

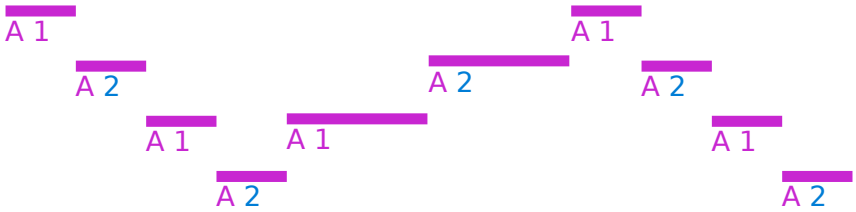


FIGURE 3.5 – Dans ce passage, une note sur deux n'est pas prédite dans la bonne voix. NPR = 5/10 = 50% (TR-sen = 0/9 = 0%, TR-prec = 0% - voir la section 3.1.2).

Dans la figure 3.6, comme précédemment, la moitié des notes n'a pas été prédite dans la bonne voix, mais il apparaît néanmoins deux groupes de notes cohérents. Ces deux séparations obtiennent pourtant le même NPR de $5/10 = 50\%$. Cette métrique ne permet pas de comparer correctement ces deux séparations.

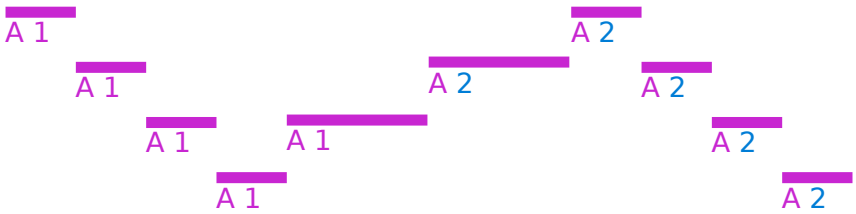


FIGURE 3.6 – Les notes de la première moitié de ce passage sont prédites dans une première voix et la seconde moitié dans une deuxième voix. NPR = 5/10 = 50%, TR-prec = 8/8 = 100%, TR-sen = 8/9 = 89%.

3.1.2 Évaluation des transitions

Ainsi prendre les notes indépendamment les unes des autres ne permet pas de comparer des algorithmes de séparation en voix et des algorithmes de séparation en streams de manière équitable. Afin de mieux prendre en compte le contexte, nous avons étudié les *transitions* entre deux notes. En effet, on peut voir le résultat des méthodes de séparation en voix ou un même streams comme un ensemble de paires de notes successives dans une même voix ou stream de la prédiction.

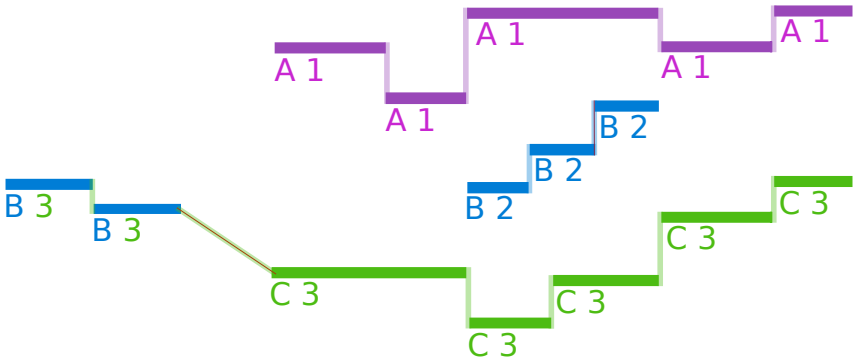


FIGURE 3.7 – La transition entre les notes étiquetées B3 et C3 est un faux positif, toutes les autres transitions sont des vrais positifs.

Je compare ces transitions aux transitions des *fichiers de référence*, et je calcule les ratios de *précision* et de *sensibilité*. La transition entre deux notes est considérée comme un vrai positif (voir la figure 3.7) si ces deux notes sont dans la même voix du fichier de référence (c'est-à-dire que la transition est présente dans l'analyse de référence). Je définis la métrique *précision des transitions* (*transition precision* en anglais, TR-prec) comme le ratio du nombre de transitions correctement assignées par rapport au nombre de transitions dans les voix prédites (dans la figure 3.8, TR-prec = 7/7 = 100%).

Je définis également la *sensibilité* des transitions (*transition sensibility*, TR-sen) comme le ratio du nombre de transitions correctement assignées par rapport au nombre de transitions du fichier de référence (dans la figure 3.8, TR-sen = 7/12 = 58%).

Dans [10], il n'existe pas de mesure directement comparable à TR-prec et TR-sen, par contre deux mesures s'en rapprochent : *Correct Fragment Connection* qui donne le pourcentage des connexions entre fragments correctement prédits par rapport au fichier de référence et *Average Fragment Consistency* qui donne le ratio de notes correctement prédites à l'intérieur des fragments par rapport au fichier de référence.

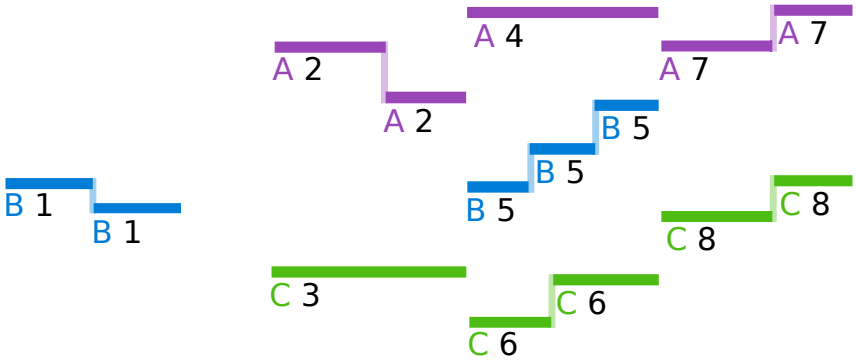


FIGURE 3.8 – Les liaisons entre les notes correspondent aux transitions entre deux notes d'un même stream.

Comme pour la mesure NPR, je n'utilise pas les moyennes des résultats sur chaque voix pour donner les résultats des TR-prec et TR-sen d'une pièce. Je calcule l'ensemble des transitions correctes, l'ensemble des transitions prédites pour une pièce ou pour un ensemble de pièces et ensuite je calcule les TR-prec et TR-sen correspondants. Lorsque le nombre de voix dans le fichier de référence est égal au nombre de voix prédites (ce qui est le cas pour les algorithmes de séparation en voix), les ratios TR-prec et TR-sen sont égaux aussi. Ceci est dû au fait que dans ce cas le nombre de transitions prédites est égal au nombre de transitions dans le fichier de référence : nous appellerons cette mesure *TR*. Contrairement à l'évaluation précision note-à-note (NPR), la manière dont les erreurs sont commises aura une influence sur la mesure TR. Dans les figures 3.5 et 3.6, la moitié des notes est attribuée à la mauvaise voix mais dans la première figure TR-sen est égal à $0/9 = 0\%$ alors que pour la seconde TR-sen est égal à $8/9 = 89\%$. Cette différence s'explique par le fait que dans le premier cas deux ensembles de notes cohérents se dégagent alors que dans le second cas une erreur est commise toutes les deux notes.

À travers les exemples présentés ci-dessus, nous avons pu voir qu'avec cette mesure nous pouvons analyser la séparation prédite aussi bien pour les voix que pour les streams.

3.1.3 Évaluation basée sur l'information mutuelle

Je propose enfin d'adapter des techniques d'évaluation de la segmentation musicale, où chaque note peut-être assignée à une étiquette [1, 43]. Lukashevich définit deux scores S_o et S_u, basés sur l'*entropie* normalisée, montrant comment un algorithme peut trop segmenter (*over segmentation* en anglais, S_o) ou pas assez segmenter (*under segmentation* en anglais, S_u) une pièce par rapport au fichier de référence . Ces scores reflètent assez bien l'information contenue dans la prédiction de l'algorithme par rapport au fichier de référence (S_o) et inversement (S_u). Ici, j'utilise la même métrique pour la séparation en voix et en streams : le fichier de référence et les prédictions de tous ces algorithmes peuvent être considérés comme une assignation d'étiquettes sur toutes les notes. Connaissant la probabilité de distribution de ces étiquettes, je peux ensuite calculer les entropies $H(\text{prédiction}|\text{référence})$ et $H(\text{référence}|\text{prédiction})$, qui deviennent S_o et S_u après normalisation [43]. Comme ces scores sont basés sur les notes plutôt que sur les transitions, ils permettent de mesurer si les regroupements de notes sont cohérents, même dans la situation où deux voix jouées simultanément sont prédites dans un même stream (donnant un mauvais TR). Dans la figure 3.9, l'ensemble des notes des deux voix a été placé dans le même stream de la prédiction, ce qui donne une mesure TR égale à 0%. Cela ne reflète pas correctement le résultat de cette annotation, alors que les mesures $S_o = 1$ et $S_u = 0$ en donnent une meilleure vue. En effet,

la pièce n'a pas été assez segmentée mais toutes les notes de la voix A ont été prédites dans un même stream ainsi que celles de la voix B. Dans la figure 3.10, nous observons l'inverse de l'exemple précédent. Ici toutes les notes ont été prédites dans des streams différents, ce qui donne comme résultat : $S_o = 0$ et $S_u = 1$. Le fichier a été trop segmenté.

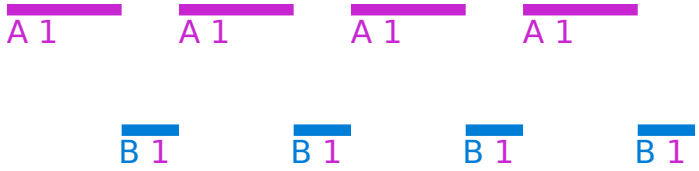


FIGURE 3.9 – Les deux voix A et B du fichier de référence sont prédites dans la voix 1.

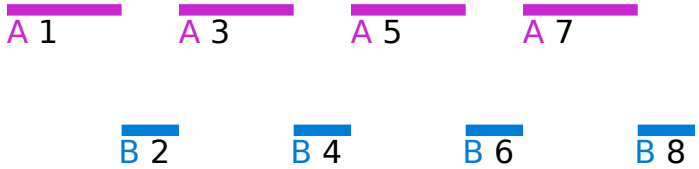


FIGURE 3.10 – Toutes les notes sont prédites dans des streams différents (8 streams).

3.2 Comparer des algorithmes de séparation

Dans la section 3.1, j'ai présenté trois *mesures d'évaluation* d'algorithmes de séparation de pièces polyphoniques. Sur ces trois mesures, deux permettent de comparer de manière équitable les algorithmes de séparation en streams et ceux en voix, il s'agit des mesures TR-prec/TR-sen et S_o/S_u. Je vais confirmer ces intuitions par une évaluation de ces mesures sur plusieurs corpus.

3.2.1 Corpus d'évaluation et fichiers de référence

Une problématique récurrente en informatique musicale est de pouvoir comparer le résultat des algorithmes à des fichiers de référence fiables. Dans mon étude, il fallait des fichiers de départ correctement séparés en un ensemble de lignes monophoniques. C'est pourquoi je travaille en grande majorité avec des fichiers **kern (cf `http://www.humdrum.org/Humdrum/guide.toc.html`) qui décrivent explicitement les voix (voir la figure 3.11).

```
**kern  **kern  **kern
*staff2 *staff1 *staff1
*clefF4 *clefG2 *clefG2
*M4/4   *M4/4   *M4/4
*c:     *c:     *c:
*MM72   *MM72   *MM72
=1      =1      =1
1r      8r      1r
.       16cc    .
.       16bn    .
.       8cc     .
.       8g      .
.       8a-     .
.       16cc    .
.       16b     .
.       8cc     .
.       8dd     .
=2      =2      =2
1r      8g      1r
.       16cc    .
.       16bn    .
.       8cc     .
.       8dd     .
.       16f     .
.       16g     .
.       4a-     .
.       16g     .
.       16f     .
=3      =3      =3
1r      16e-    8r
.       16cc    .
.       16bn    16gg
.       16an    16ff#
.       16g     8gg
.       16fn    .
.       16e-    8cc
.       16d     .
.       8c      8ee-
.       8ee-    16gg
.       .       16ff#
.       8dd     8gg
.       8cc     8aan
```

FIGURE 3.11 – Les trois premières mesures de la Fugue #2 du premier livre du *Clavier bien tempéré* de J.-S. Bach (en Do mineur, BWV 847) en format **kern. Chaque colonne représente une voix, une note est représentée par sa durée suivie de sa hauteur (par exemple un sol croche est noté `8g`)

Comme une fugue est constituée de plusieurs voix, cela constitue naturellement une bonne référence pour évaluer les algorithmes de séparation de voix [10, 62, 18, 33, 34, 35, 44, 55]. J'ai donc évalué ces algorithmes sur les 48 fugues des deux livres du *Clavier bien tempéré* de J.-S. Bach (corpus `wtc-i` et `wtc-ii`, voir la table 3.1, fichiers `.krn` téléchargés à partir de `kern.ccarh.org` [32]).

Dans la section 4.1.6, j'utiliserai également un *corpus* constitué de 17 premiers mouvements de quatuor à cordes de musique classique et romantique (corpus `quatuor`, Haydn op. 33-1 to 33-6, op. 54-3, op. 64-4, Mozart K80, K155, K156, K157 et K387, Beethoven op. 18-2, Brahms op. 51-1 et Schubert op. 125-1) ainsi que les 12 premières fugues de l'opus 87 de D. Shostakovitch (corpus `Opus 87`).

J'ai également voulu faire l'évaluation sur d'autres écritures polyphoniques. Le problème est d'avoir les fichiers de référence correspondant à la bonne séparation de voix (ou streams). À partir d'un ensemble de 2290 fichiers MIDI de musique populaire, j'ai formé un corpus utilisable pour l'évaluation de ces algorithmes. Les fichiers MIDI que j'utilise sont de type SMF (Standard Midi File) 1, les pistes sont indépendantes les unes des autres. Je me suis intéressé aux pistes MIDI (et non aux canaux MIDI). J'ai gardé uniquement les pistes « monophoniques » (où au maximum une note est jouée à la fois) d'une taille suffisante (au moins 20% de la taille de la piste la plus longue). J'ai supprimé les pistes correspondant à la batterie. J'ai considéré chaque voix restante comme une voix indépendante. Finalement, j'ai conservé 97 fichiers MIDI avec au moins deux voix, composés en moyenne de 3.0 voix (corpus `pop`, voir la table 3.1).

corpus	wtc-i	wtc-ii	opus 87	quatuor	pop
fichiers	24	24	24	17	97
voix	3.5	3.4	3.5	3.9	3.0
notes	1041	1071	1378	2238	874

Table 3.1 – Description des cinq corpus utilisés pour évaluer la séparation de voix et de streams. J'indique pour chaque corpus le nombre de pièces ainsi que les nombres moyens de voix et de notes.

3.2.2 Implémentation

J'ai implémenté les algorithmes CW-Contigs (voir la section 2.4.6), CW (voir la section 2.3.2), IMS (voir la section 2.3.3) et Stream Segment (voir la section 2.4.3), en utilisant un framework développé en Python basé sur music21 [15]. Je noterai CW-Prioritized la version modifiée de l'algorithme IMS que j'ai implémentée.

Le programme exécutant l'algorithme de Streamer (voir la section 2.4.1) a été téléchargé à partir de `www.link.cs.cmu.edu/melisma`. Pour l'exécution, j'ai utilisé les paramètres donnés par défaut.

Modification de l'algorithme Streamer : Quantification. Le programme développé par Temperley prend en entrée des fichiers MIDI et donne en sortie des fichiers texte qui indiquent à quel stream appartient chaque note. Les fichiers MIDI peuvent être obtenus de différentes manières. Ils peuvent provenir par exemple d'éditeurs de partitions ou d'enregistrements de performances. Dans le cas d'enregistrement de performances, il peut exister une différence entre la durée de la note jouée par un musicien et celle de la partition. C'est pourquoi une étape de quantification est souvent nécessaire : il s'agit de modifier le temps de début et la durée des notes pour faire la correspondance entre la durée réelle des notes et leur valeur théorique (voir la figure 3.12). Même si mes fichiers MIDI proviennent d'éditeurs de partitions, l'étape de pré-traitement du programme de Streamer consiste en une quantification des données. À cause de cette étape, l'onset et la durée des notes sont légèrement modifiés

en sortie par rapport aux fichiers originaux : j'ai donc du ré-associer les notes obtenues en sortie du programme de Streamer à celles des fichiers de référence, de manière à pouvoir mesurer la qualité de séparation.

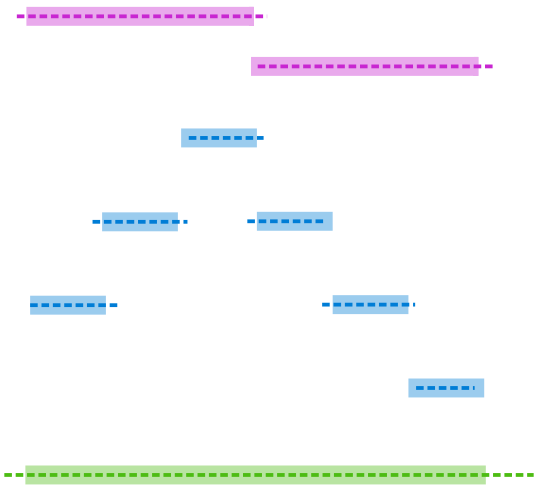


FIGURE 3.12 – Étape de quantification effectuée par Streamer. Les pointillés représentent les notes avant l'étape de quantification et les segments continus les notes après l'étape de quantification.

Modification de l'algorithme CW : Débuts multiples. Dans l'algorithme CW, l'étape de connexion des contigs commence simultanément de l'ensemble des *contigs maximaux* (contig contenant le nombre maximal de notes jouées en parallèle). Dans l'exemple de la figure 3.13, il y a deux contigs maximaux, le 0 et le 3. Dans la figure 3.14, la règle de proximité des hauteurs permet de connecter les fragments des contigs 0 et 1, ainsi que des contigs 2 et 3. L'algorithme CW interdit les croisements de voix : sur la figure 3.15, nous sommes dans l'impossibilité de déterminer comment les contigs 0/1 et 2/3 peuvent se connecter. En effet, en respectant la règle de proximité des hauteurs, le fragment 2 du contig 0/1 doit se connecter au fragment 1 du contig 2/3. Cette connexion est impossible car nous ne pouvons déterminer si le fragment de voix obtenu après connexion appartient à la voix 1 ou à la voix 2. Pour résoudre ce problème, j'ai choisi dans mon implémentation de partir d'un unique contig maximal à partir duquel j'étends les connexions.

Modification de l'algorithme CW : Notes partitionnées. Dans l'algorithme de CW, il arrive que dans l'étape de création des contigs, des notes chevauchant plusieurs contigs soient dupliquées dans plusieurs fragments. Dans ce cas, il se peut que lors de l'étape de connexion des contigs, des fragments contenant les mêmes notes ne soient pas connectés ensemble (voir la figure 3.16), ce qui conduit à avoir plus de notes dans le fichier de prédiction que dans le fichier de référence. Ce cas n'est pas détaillé dans l'étude [10]. Pour ne pas avoir de notes artificiellement dupliquées en sortie de l'algorithme, je connecte ensemble les fragments qui partagent une même note même si pour cela les voix doivent se croiser (figure 3.16).

Modification algorithme IMS : Poids des liens. Pour favoriser la connexion entre les fragments qui partagent une note, je donne un poids très faible de -10000 ($-\infty$ dans CW). La somme des liens ayant le poids le plus faible est retenue. Dans le même cas de figure IMS donne un poids de -1 : une même note pourra ainsi se retrouver dans deux voix différentes (voir la figure 3.16).

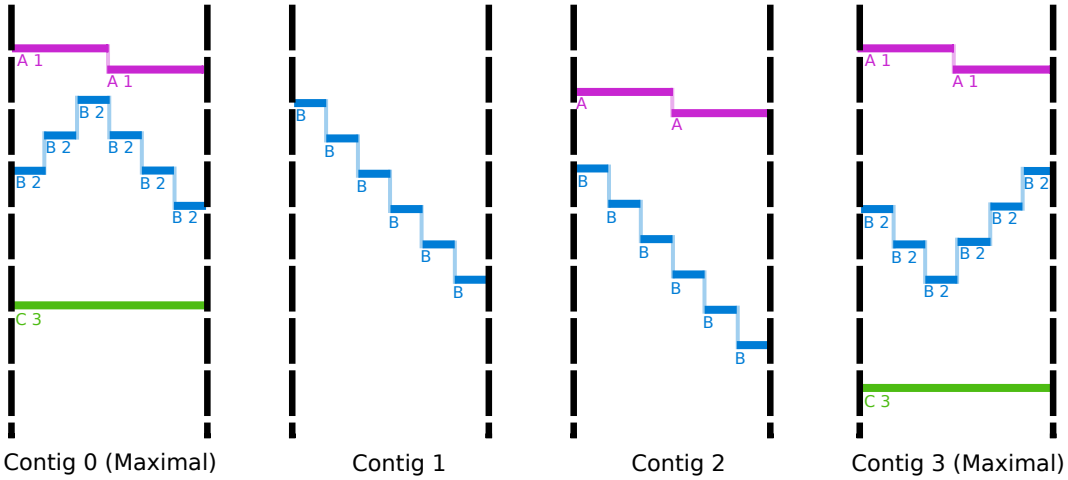


FIGURE 3.13 – L'objectif est de connecter les fragments des contigs 0 à 3. Les contigs 0 et 3 sont maximaux.

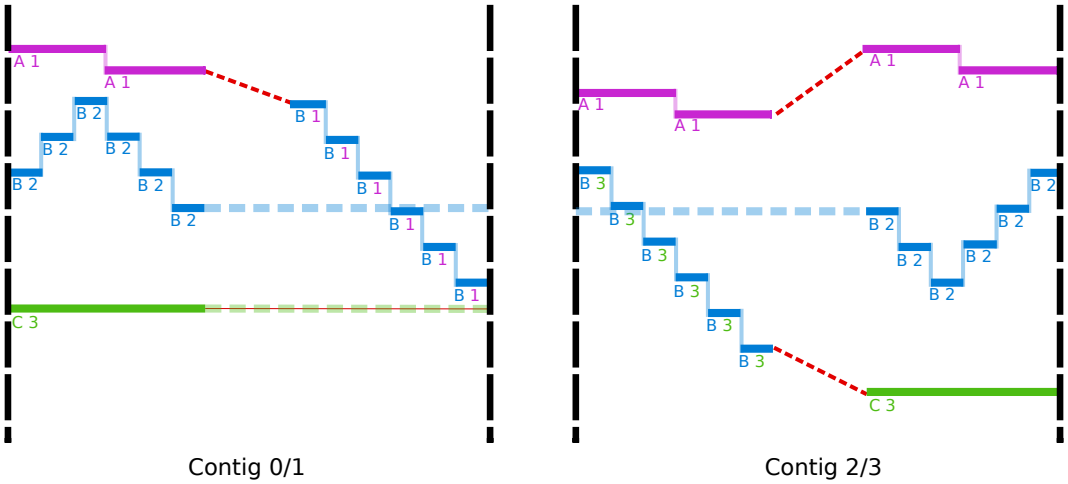


FIGURE 3.14 – L'algorithme CW commence en partant des contigs maximaux 0 et 3 simultanément.

3.2.3 Résultats et discussions

J'ai évalué les six algorithmes (algorithme trivial, CW, IMS, Streamer, Stream Segment, CW-Contigs) sur les 48 fugues du *Clavier bien tempéré* de J.-S. Bach. Les algorithmes de séparation en voix ont également été évalués sur les 97 fichiers de musique populaire. La table 3.2 détaille les résultats.

Résultats

Évaluation note-à-note et évaluation des transitions. Les résultats de la table 3.2 confirment que la métrique NPR n'est vraiment pas pertinente. L'algorithme trivial présenté à la section 2.3.1 a un bon NPR et sur le corpus de musique pop, cette mesure est même supérieure à celles obtenues par les algorithmes CW et CW-Prioritized. L'algorithme trivial à l'inverse des deux autres algorithmes de séparation en voix a une mesure NPR plus élevée que sa

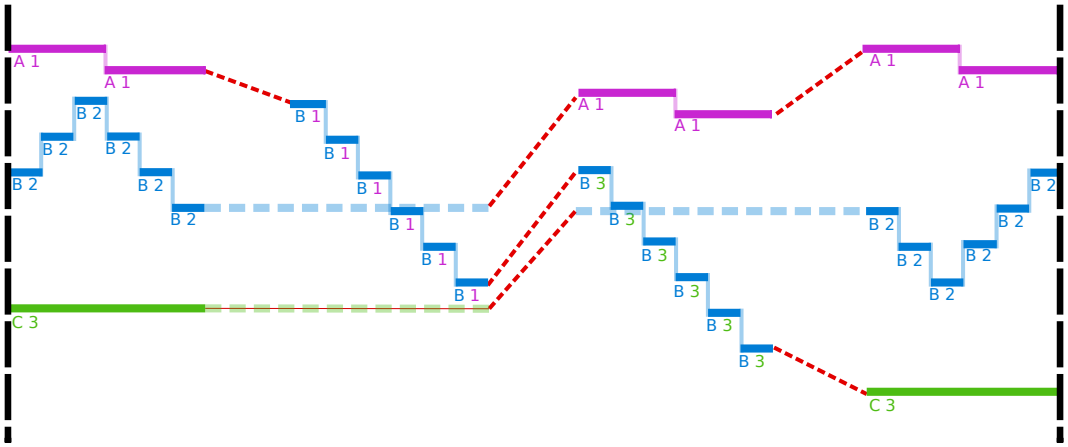


FIGURE 3.15 – L'algorithme CW interdit les croisements de voix. Nous arrivons dans une situation indéterminée où nous ne pouvons pas connecter les fragments des contigs 0/1 avec les fragments des contigs 2/3.

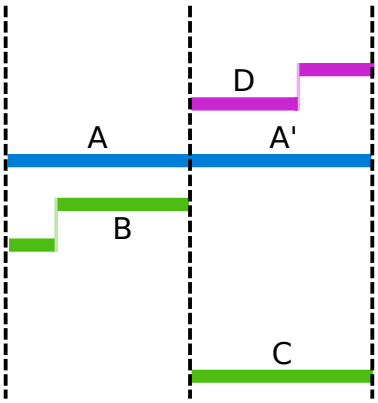


FIGURE 3.16 – La note AA' est présente dans deux fragments. CW et CW-Prioritized lient les fragments $(A + A')$, $(B + C)$, gardant A dans la même voix. L'implémentation originale de IMS connecte les fragments $(A + D)$, $(B + A')$, dupliquant la longue note $A + A'$.

mesure TR. Cela vient du fait qu'en moyenne les groupes de notes appartenant à la même voix sont plus courts que ceux des deux autres algorithmes (voir la figure 3.17).

Pour CW-Prioritized, mes résultats diffèrent de ceux obtenus dans [33] : leurs AVC étaient meilleurs comparés à CW. Dans mon implémentation, le ratio NPR est plus bas pour CW-Prioritized que pour CW. Mon implémentation de CW-Prioritized donne de bons résultats en considérant TR sur les corpus de fugues et de pop. Comme attendu, la mesure du ratio de transition (TR) permet de déterminer la capacité des algorithmes à regrouper les notes dans la bonne voix : tous les algorithmes ont un meilleur TR que l'algorithme trivial. Les trois algorithmes de segmentation en streams prédisent plus de streams que le nombre de voix dans le fichier de référence, d'où un ratio TR-sen relativement bas. À l'intérieur des streams, le taux de prédictions correctes est assez élevé, en particulier pour l'algorithme CW-Contigs qui a un TR-prec proche de 100%.

Évaluation basée sur l'information mutuelle. Un cas extrême est la prédiction parfaite, où les mesures NPR et TR sont égales à 100% et les mesures S_o et S_u valent 1. Ceci arrive

Corpus	Algorithme	avg.	NPR	TR-prec	TR-sen	S_o	S_u
wtc-i	Baseline	3,5	71,4 %	63,7 %		0,48	0,48
	CW		**83 %**	95,9 %		**0,73**	0,73
	CW-Prioritized		82,5 %	**97,4 %**		0,72	0,72
	Streamer	16	-	75,6 %	68,3 %	0,46	0,62
	Stream Segment	191,1	-	76,5 %	68,3 %	0,23	0,79
	CW-Contigs	226,2	-	**99,4 %**	**86,7 %**	0,34	**0,98**
wtc-ii	Baseline	3,4	71,9 %	62,6 %		0,45	0,45
	CW		**87,8 %**	95,6 %		0,73	0,73
	CW-Prioritized		86,5 %	**97,1 %**		**0,74**	0,74
	Streamer	16	-	75,6 %	65,2 %	0,42	0,57
	Stream Segment	191,1	-	77,4 %	61,9 %	0,21	0,79
	CW-Contigs	226,2	-	**99,4 %**	**86,8 %**	0,34	**0,98**
pop	Baseline	3	**89,5** %	87,1 %		**0,77**	0,75
	CW		84,6 %	88,7 %		0,76	**0,76**
	CW-Prioritized		64,8 %	**89,4 %**		0,51	0,5

TABLE 3.2 – Résultats sur les corpus de fugues et de pop. La colonne *avg* donne le nombre moyen de voix/streams prédit.

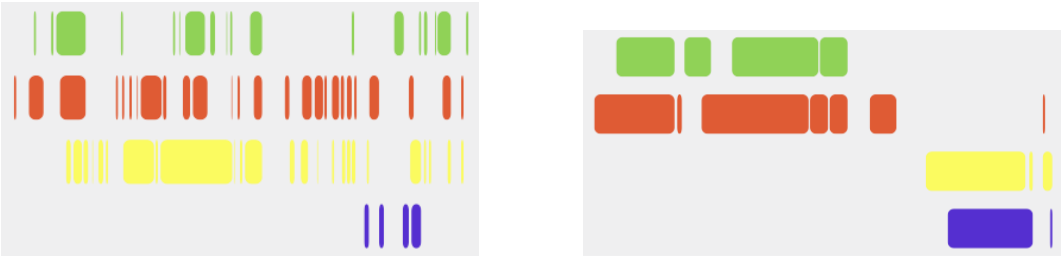


FIGURE 3.17 – L'algorithme trivial (à gauche) prédit beaucoup plus de fragments de voix que l'algorithme CW (à droite) sur la Fugue #2 du livre II du *Clavier bien tempéré* de J.-S. Bach (en Do mineur, BWV 871).

en général pour les algorithmes de séparation en voix CW et CW-Prioritized quand il n'y a que deux voix à prédire, comme par exemple dans la Fugue #10 du livre I du *Clavier bien tempéré* de Bach (en mi mineur, BWV 855). Dans une chanson pop (allsaints-bootiecall) deux voix jouant quasiment les mêmes notes sont prédites dans la même voix, ce qui donne un NPR et un TR proches de 50%, mais S_o est proche de 1 et S_u est proche de 0. Comme attendu dans les algorithmes de séparation en streams, la mesure S_u est supérieure à la mesure S_o du fait que ces algorithmes prédisent beaucoup plus de streams qu'il n'y a de voix, ils sur-segmentent les fichiers. Il est à noter que Stream Segment n'a pas le ratio TR-prec le plus haut (il arrive qu'il connecte des notes qui proviennent de voix différentes), mais il garde un plutôt bon score S_u comparativement à tous les algorithmes du fait que lorsqu'il connecte des notes provenant de différentes voix, il a tendance à placer dans le même stream toutes les notes qui sont dans des voix proches. Les meilleurs scores S_u sont obtenus par CW-Contigs, confirmant le fait que l'étape de création des contigs est une très bonne méthode qui ne fait presque pas d'erreurs.

Étude d'un cas particulièrement complexe

La figure 3.18 montre les résultats des algorithmes sur un extrait de la Fugue #16 du livre I du *Clavier bien tempéré* de Bach (en Sol mineur, BWV 861). Ce passage est un challenge pour la séparation en voix : les quatre voix entrent successivement et, au début du sujet (les notes 2 à 6 de la mesure 12 sur la première portée), il y a un intervalle important (une sixte, entre le sol et le si sur la première mesure) qui rapproche les voix. Dans la dernière mesure de l'extrait, il y a même un croisement de voix quand la soprano joue l'intervalle de sixte du sujet. Les algorithmes se comportent différemment sur ce passage mais aucun ne l'analyse parfaitement.

L'algorithme CW-Prioritized est le seul à prédire correctement les trois premières mesures, en particulier le début de l'alto sur les deux premiers temps de la mesure 12. CW commet une erreur sur le troisième temps de la mesure 14, qui implique de mauvaises prédictions dans les mesures 12/13/14 (mais un ratio TR tout de même élevé). Hormis des erreurs dans la mesure 12 et le croisement de voix qu'il ne détecte pas dans la dernière mesure, Streamer a des résultats corrects. Stream Segment crée beaucoup de streams et, comme attendu, il assigne dans un même stream des notes qui se chevauchent, comme on peut voir sur le premier temps de la mesure 12.

Aucun des algorithmes n'a été capable de réussir la segmentation du croisement de voix dans la mesure 15. CW-Contigs fait ici son unique erreur de prédiction (mais, il a quand même un excellent TR-prec) : il lie le *ré* de la soprano avec le *sol* qui suit dans l'alto. Comme on s'y attendait, cette erreur est présente dans CW et CW-Prioritized mais également dans Streamer qui place dans des streams différents des notes contiguës ayant des différences de hauteur importantes. À cet endroit, Stream Segment crée des streams qui contiennent les deux voix.

Pour traiter correctement ce passage, il est nécessaire d'avoir une connaissance des motifs (incluant la tête du sujet avec l'intervalle de sixte) et pour conserver ces motifs dans les bonnes voix, il faut autoriser les croisements de voix.

3.3 Bilan : Voix ou stream ?

Les algorithmes de séparations en voix et en streams proposent des méthodes pour regrouper les notes à partir de partitions polyphoniques. Les méthodes d'évaluation que j'ai choisies ont permis de montrer que les algorithmes de séparation en streams donnaient de bons résultats de séparations en voix, comme on a pu voir avec les ratio TR et les scores S_u. L'algorithme de Streamer est très proche d'une séparation en voix complète, prédisant de longs streams monophoniques.

L'algorithme de Stream Segment permet d'obtenir en sortie des streams polyphoniques. Dans les pièces où la polyphonie ne peut pas se décomposer en un ensemble de lignes monophoniques, cette approche permet de mettre en relation des groupes de notes cohérents comme par exemple des accords.

Sur le problème de la séparation en voix, l'approche par contig, initialement proposée par [10], est une excellente approche – très peu d'erreurs sont faites dans les transitions à l'intérieur des contigs, comme le montrent les résultats de l'algorithme CW-Contigs.

Le challenge est de réussir à connecter correctement les contigs. En fait, il s'agit de répondre à une double question : *Quels* contigs doivent être connectés ? *Comment* les connecter ? Dans nos expériences, je n'ai observé que peu de différences dans les résultats entre notre implémentation de CW-Prioritized et de CW. Néanmoins, la réflexion initiée par IMS a la qualité de montrer que le choix de l'ordre de connexion des contigs a une importance.

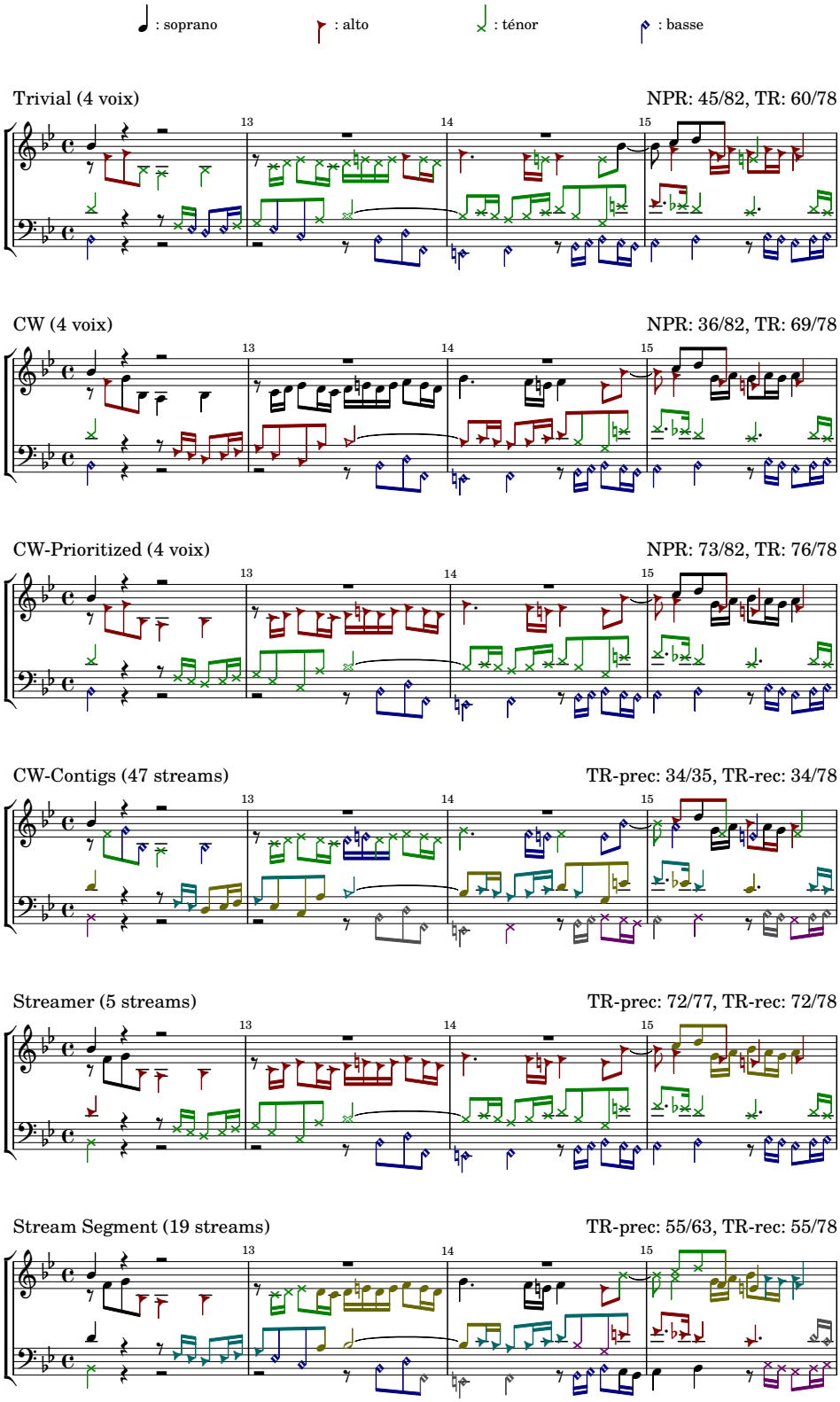


FIGURE 3.18 – Sorties des cinq algorithmes des mesures 12 à 15 de la Fugue #16 du livre I du *Clavier bien tempéré* de J.-S. Bach (en Sol mineur, BWV 861). Après un accord initial contenant presque toutes les voix, elles entrent successivement (l'alto et la ténor : mesure 12, la basse : mesure 13, la soprano : mesure 15).

Chapitre 4

Améliorer la séparation en voix et en streams

Dans le chapitre 3, j'ai mis en évidence la proximité entre les algorithmes de *séparation* en voix et les algorithmes de *séparation en streams*. J'ai également montré que l'algorithme CW donne de bons résultats – particulièrement lors de son étape de création des contigs. Les résultats de l'algorithme CW-Prioritized montrent que l'ordre de connexion des contigs influence la qualité de la prédiction.

L'objectif de ce chapitre est de montrer l'influence du choix de l'ordre de connexion des contigs sur la capacité à séparer des voix et déterminer les caractéristiques musicales permettant de prévoir au mieux l'ordre de connexion.

Dans ce chapitre, je réponds à deux questions : *Quels* contigs doivent être connectés ? *Comment* les connecter ? Dans les études précédentes, ces questions étaient traitées séparément. Dans la section 4.1, j'explique comment il est possible de répondre à ces questions simultanément en utilisant des caractéristiques musicales qui nous permettent de déterminer un score de fiabilité de connexion entre deux groupes de notes. Je présente ces caractéristiques, puis j'explique comment j'ai déterminé les meilleures valeurs de coefficients pour ces caractéristiques en utilisant un algorithme génétique. Et enfin, je détaille les résultats obtenus.

Les conclusions du chapitre 3 et les résultats encourageants de la section 4.1 m'ont amené à me demander s'il ne valait pas mieux stopper la connexion des contigs avant d'avoir une séparation complète en voix. Cette question est étudiée dans la section 4.2.

Le travail présenté dans ce chapitre a fait l'objet de l'article [27], présenté à ISMIR 2016.

4.1 Améliorer l'étape de connexion des contigs

Les précédentes méthodes commençaient par déterminer quels contigs devaient être connectés puis définissaient les connexions entre fragments. Je soutiens que *pour déterminer l'ordre de connexion des contigs le plus sûr, il faut également prendre en compte à cette étape la connexion entre les fragments.* Pour construire une voix cohérente à travers toute la pièce, mon approche est donc de commencer par connecter les contigs ayant les connexions entre leurs fragments les plus « sûres ». Si les premières connexions sont correctes, nous augmentons l'information pertinente au sein du nouveau contig créé.

4.1.1 Une nouvelle manière de penser la séparation en voix basée sur les contigs

Pour deux contigs successifs i et $i+1$ et une manière C de les connecter (c'est-à-dire un choix des paires de fragments à connecter), je définis un *score de connexion* $S(i, C)$, calculé comme une somme pondérée de *caractéristiques musicales* (détaillées dans la section suivante), qui mesure la qualité de cette connexion : plus le *score de connexion* est haut, plus la connexion est sûre. Le score de connexion étend celui utilisé par CW et IMS. La relation entre les deux choix de connexions (contigs et fragments) n'avait alors pas été explorée systématiquement. Cette approche ressemble par certains aspects aux méthodes de séparation en voix par apprentissage (voir la section 2.3.4). En effet, dans les deux cas, l'objectif est de définir les meilleures valeurs à attribuer à des caractéristiques. La différence vient du fait que, dans mon algorithme, ces valeurs sont apprises sur un corpus d'entraînement et ne sont pas modifiées pour chaque corpus d'évaluation. Dans la majorité des approches par apprentissage, le principal critère utilisé pour définir l'appartenance d'une note à une voix est celui de la différence de hauteur. Je propose ici des caractéristiques musicales plus variées et qui utilisent le contexte local pour définir la meilleure séparation en voix.

À chaque étape de l'algorithme, le couple (i, C) maximisant S est sélectionné, ce qui donne la « meilleure paire de contigs » à connecter ainsi que la « meilleure manière » de les connecter. Une fois que cette connexion est faite, les scores de connexion entre le nouveau contig créé et ses voisins de gauche et de droite doivent être recalculés.

Définitions. Soit n le nombre maximal de notes jouées simultanément dans la pièce ; soit n_i (respectivement n_{i+1}) le nombre maximal de voix d'un contig i (respectivement $i+1$) ; après que deux contigs ont été connectés, le contig résultant peut avoir un nombre différent de notes jouées simultanément à ses extrémités, mais les voix suspendues sont « projetées » aux extrémités (voir la figure 4.1).

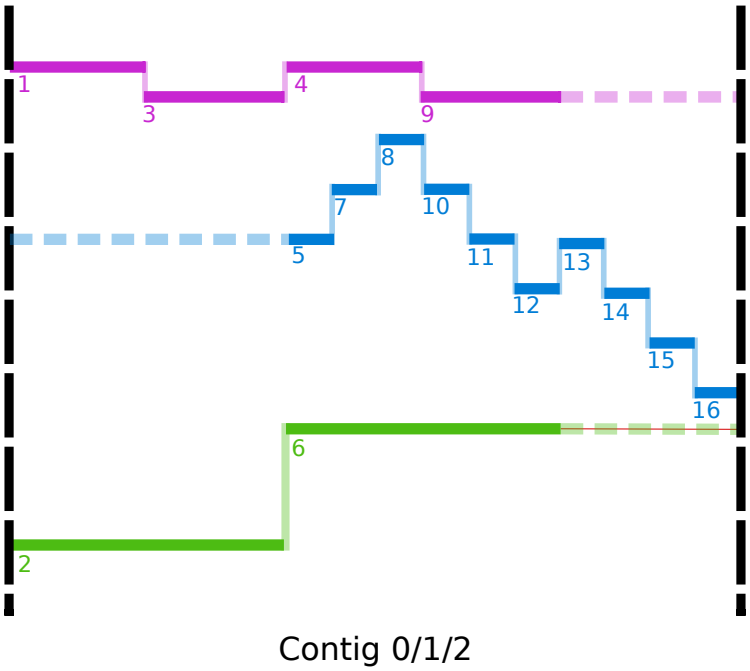


FIGURE 4.1 – Après les connexions des contigs 0, 1 et 2, les notes 5, 6 et 9 sont projetées aux extrémités du contig nouvellement créé.

Pour deux contigs successifs i et $i+1$, soit C un ensemble de paires (ℓ, r), où ℓ est un fragment du contig de gauche i et r un fragment du contig de droite $i+1$, chaque fragment apparaît au plus une fois dans C (voir la figure 4.2).

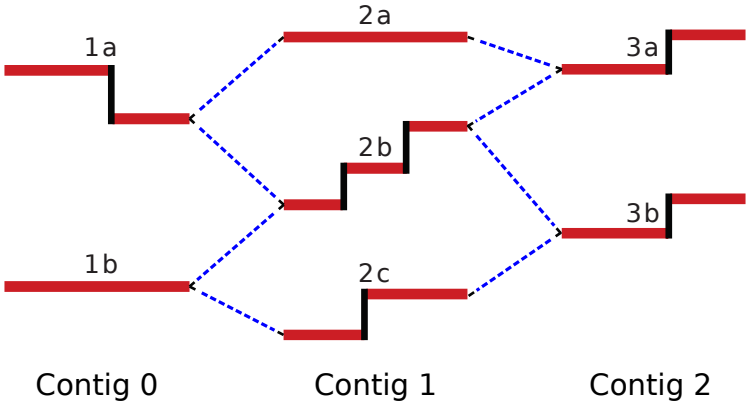


FIGURE 4.2 – Pour déterminer la connexion la plus sûre, il faut se demander quels contigs doivent être connectés (par exemple, Contig 1 et Contig 2) et comment faire leur connexion. Ici, il y a trois connexions possibles (sans croisement de voix) entre les contigs 1 et 2 : $C_1 = \{(1a, 2a), (1b, 2b)\}$, $C_2 = \{(1a, 2a), (1b, 2c)\}$, et $C_3 = \{(1a, 2b), (1b, 2c)\}$.

C a donc au plus $m = \min(n_i, n_{i+1})$ éléments et dans ce qui suit je considère uniquement les ensembles avec m éléments, ce qui est le plus grand nombre de connexions possibles. Notons $M = \max(n_i, n_{i+1})$. Il y a $M!/(M-m)!$ différentes combinaisons possibles pour C, et uniquement $\binom{M}{m}$, si nous restreignons aux combinaisons qui ne contiennent pas de croisements de voix. Nous considérons que nous avons N caractéristiques $f_1(i, C), f_2(i, C) \ldots f_N(i, C)$ singularisant certaines propriétés musicales de la connexion C entre les contigs i et $i+1$. Chaque caractéristique a une valeur comprise entre 0 et 1. Finalement, prenons N coefficients $\alpha_1, \alpha_2, \ldots, \alpha_N$, tels que $\sum_{k=1}^{N} \alpha_k = 1$. Je définis le score de connexion comme une *combinaison linéaire* des caractéristiques $S(i, C) = \sum_{k=1}^{N} \alpha_k f_k(i, C)$.

Dans les deux paragraphes qui suivent, je propose différentes caractéristiques $f_k(i, C)$ dépendant des propriétés musicales des contigs et des fragments. La valeur des coefficients α_k sera discutée dans la section 4.1.4.

4.1.2 Caractéristiques dépendant des contigs

Pour commencer, nous considérons des caractéristiques qui dépendent uniquement des contigs et plus précisément du nombre maximal de voix dans chaque contig :

— *maximal_voices(i)* $= \max(n_i, n_{i+1})/n$. Plus le nombre de voix est proche du nombre maximal de voix, plus le score de connexion est élevé.
— *minimal_voices(i)* $= (n + 1 - \min(n_i, n_{i+1}))/n$. Moins un des deux contigs contient de voix, plus le score de connexion est élevé.

On peut en particulier favoriser la connexion des contigs en se basant sur la différence du nombre de voix entre le contig de droite et celui de gauche :

— *difference_nb_voices(i)* $= 1 - (|n_i - n_{i+1}|/(n-1))$. Plus la différence du nombre de voix entre le contig de gauche et celui de droite est faible, plus le score de connexion est élevé.

Les caractéristiques binaires suivantes vaudront 0 si la condition n'est pas respectée :

— *increase(i)* $= 1$ si et seulement si $n_i < n_{i+1}$;

— *increase_one(i)* $= 1$ si et seulement si $n_i + 1 = n_{i+1}$;
— *increase_equal(i)* $= 1$ si et seulement si $n_i \leq n_{i+1}$;
— *decrease(i)* $= 1$ si et seulement si $n_i > n_{i+1}$;
— *decrease_one(i)* $= 1$ si et seulement si $n_i - 1 = n_{i+1}$;
— *decrease_equal(i)* $= 1$ si et seulement si $n_i \geq n_{i+1}$.

Ces caractéristiques sont inspirées des méthodes de connexions des algorithmes existants. La caractéristique *maximal_voices(i)* fait référence à l'idée utilisée dans l'algorithme CW : il est plus sûr de commencer par connecter les contigs ayant un nombre de voix important.

L'idée inverse, telle que mesurée par *minimal_voices(i)*, a été proposée avec *increase(i)* pour se rapprocher de l'idée de l'algorithme IMS, à savoir favoriser les connexions entre contigs ayant un nombre croissant de voix. L'idée est que le départ (local) d'une nouvelle voix est un événement plus prévisible que sa fin (locale). Ceci est un événement tout à fait remarquable dans la musique contrapuntique telle que les fugues : dans ce type de musique, chaque voix qui *entre* reprend un motif thématique (sujet, contre-sujet).

Je propose d'aller plus loin en utilisant la caractéristique *increase_one(i)* qui permet de mieux identifier l'entrée d'*une* nouvelle voix. Inversement, j'évalue aussi l'idée opposée (*decrease(i)*, *decrease_one(i)*, *decrease_equal(i)*).

Finalement, une caractéristique favorise la connexion des contigs ayant des notes en commun :

— *maximal_sim_notes(i)* $= n^{=} / \min(n_i, n_{i+1})$, où $n^{=}$ est le nombre de notes avec la même hauteur et le même temps de début (i.e. note coupée en deux) à l'extrémité des contigs i et $i+1$. Plus les contigs partagent des notes communes, plus le score de connexion est élevé.

Cette caractéristique vient de l'implémentation originale de CW, où un score très important était attribué lorsque deux fragments partageant une même note étaient connectés.

4.1.3 Caractéristiques dépendant des fragments

Maintenant, je vais présenter les caractéristiques considérant les connexions de fragments (ℓ, r) composant C.

Hauteurs. Comment peut-on mesurer la qualité de connexion d'un fragment ℓ avec un fragment r ? Le critère principal des algorithmes CW et IMS était de suivre le principe de proximité de hauteur, favorisant les connexions de fragments ayant un petit intervalle de hauteur. Étant donnés C et $(\ell, r) \in C$, on note *last_pitch*(ℓ) et *first_pitch*(r) les hauteurs des notes extrêmes du fragment de gauche ℓ et du fragment de droite r. Étant donné ν un facteur de normalisation, explicité ci-dessous, je définis :

— *extreme_pitch(C)* $= 1 - \sum_{(\ell,r)\in C} |\mathit{last\_pitch}(\ell) - \mathit{first\_pitch}(r)| / max(1, \nu)$. Plus les notes connectées ont une hauteur proche, plus le score de connexion est élevé.

Le facteur de normalisation $\nu = 60 \cdot |C|$ demi-tons a été choisi de manière à donner à la caractéristique une valeur comprise entre 0 (5 octaves de différence entre les notes connectées, ce qui correspond à la différence de hauteur maximale entre deux notes de notre corpus) et 1 (hauteurs égales).

De plus, ce score *extreme_pitch(C)* prend uniquement la hauteur d'une unique note de chaque fragment. Je propose d'étendre cette caractéristique en prenant en compte la moyenne des hauteurs (*average_pitch*) de *toutes les notes* d'un ou des deux fragments.

Ce score peut se rapprocher de l'idée de McLeod [47] de faire une moyenne des hauteurs des notes d'une voix en donnant plus d'importance aux dernières notes jouées. Effectivement,

les voix tendent à avoir une même échelle de hauteur à travers toute la pièce, et plus encore à travers les fragments (voir la figure 4.3) :

— $avg\_pitch\_right(C) = 1 - \sum_{(\ell,r)\in C} |last\_pitch(\ell) - average\_pitch(r)|/\nu$;
— $avg\_pitch\_left(C) = 1 - \sum_{(\ell,r)\in C} |average\_pitch(\ell) - last\_pitch(r)|/\nu$;
— $avg\_pitch(C) = 1 - \sum_{(\ell,r)\in C} |average\_pitch(\ell) - average\_pitch(r)|/\nu$.

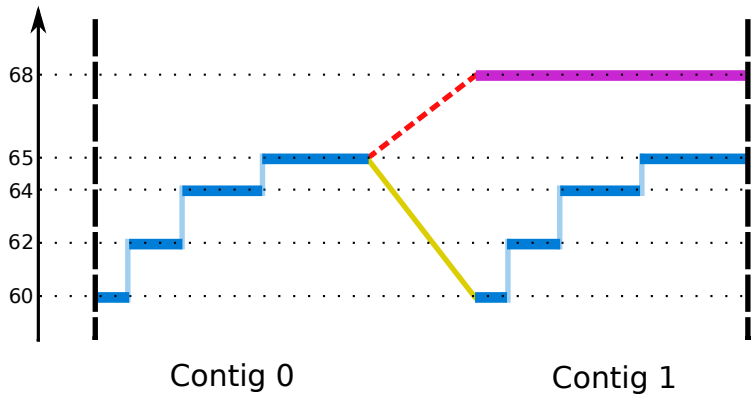


FIGURE 4.3 – Arpèges accompagnant le début d'une mélodie. Les précédentes méthodes de connexion commettaient une erreur (en pointillé), due à l'unique utilisation des hauteurs des dernières et premières notes des fragments à connecter. En utilisant la moyenne des hauteurs de l'ensemble des notes des fragments, nous trouvons la bonne connexion à faire (trait plein).

Certains algorithmes de séparation en voix assignent à chaque note la voix qui a la hauteur moyenne la plus proche [37]. Ces algorithmes sont assez efficaces, et la caractéristique *avg_pitch(C)* reproduit cette idée à une échelle locale : étant donné un fragment de quelques notes, même si nous ne pouvons pas savoir de quelle voix (globale) ces notes proviennent, nous pouvons connaître une partie de son *ambitus*, c'est-à-dire de l'étendue des hauteurs qu'elle utilise.

Durée. De même, je peux mesurer la différence de durée pour favoriser la connexion de fragments contigus ayant un même rythme. En effet, l'écriture musicale tend à avoir une cohérence rythmique. Par exemple, une voix contenant des rondes et une autre contenant des croches seront souvent entendues comme deux voix séparées même si elles ont des hauteurs très proches. Étant donnés C et $(\ell, r) \in C$, on note $last\_dur(\ell)$ et $first\_dur(r)$ les durées prises dans une échelle logarithmique (de manière à limiter l'influence d'une note longue parmi un ensemble de notes courtes) des notes extrêmes du fragment de gauche ℓ et du fragment de droite r. Étant donné λ un facteur de normalisation, explicité ci-dessous, je définis :

— $extreme\_dur(C) = 1 - (\sum_{(\ell,r)\in C} |last\_dur(\ell) - first\_dur(r)|/max(1, \lambda))$. Plus les durées des notes connectées sont proches, plus le score de connexion est élevé.

Le facteur de normalisation $\lambda = 6 \cdot |C|$ représente ici la différence maximale (dans une échelle logarithmique) entre des rondes et des quadruples croches, les notes les plus longues et les plus courtes de notre corpus.

Ici aussi la caractéristique peut être étendue pour prendre en compte une durée logarithmique moyenne (*average_dur*) d'un ou des deux fragments (voir la figure 4.4) :

— $avg\_dur\_right(C) = 1 - \sum_{(\ell,r)\in C} |last\_dur(\ell) - average\_dur(r)|/\lambda$;
— $avg\_dur\_left(C) = 1 - \sum_{(\ell,r)\in C} |average\_dur(\ell) - last\_dur(r)|/\lambda$;
— $avg\_dur(C) = 1 - \sum_{(\ell,r)\in C} |average\_dur(\ell) - average\_dur(r)|/\lambda$.

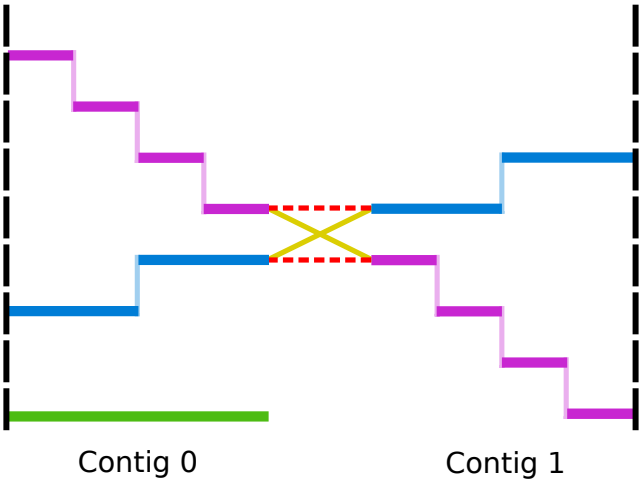


FIGURE 4.4 – Croisement de voix. Les précédentes méthodes de connexion commettaient une erreur (en pointillé), en utilisant uniquement les valeurs des hauteurs pour connecter les fragments. En utilisant la moyenne des durées de l'ensemble des notes des fragments nous trouvons la bonne connexion à faire (trait plein).

Ces caractéristiques mesurent comment un fragment peut-être constitué « plutôt de notes courtes » ou « plutôt de notes longues », même si ces fragments contiennent des durées différentes.

Croisements de voix. Pour finir, deux caractéristiques contrôlent le croisement de voix. D'un côté, les croisements de voix peuvent exister, mais d'un autre côté, ils sont difficiles à prévoir. Habituellement, les algorithmes de séparation en voix (comme CW et IMS) les interdisent.

— *crossed_voices(C)* $= 1$ si C contient un croisement de voix, et 0 sinon ;
— *no_crossed_voices(C)* $= 1$ si C ne contient pas de croisement de voix, et 0 sinon.

4.1.4 Apprentissage des coefficients par *algorithme génétique*

La sélection des *coefficients* des caractéristiques $(\alpha_1, \alpha_2, \ldots \alpha_N)$ a été faite en utilisant un algorithme génétique avec des opérations de *mutation* et de *croisement* [4] (voir la figure 4.5).

Pour des raisons d'efficacité de calculs, une *génération* est un ensemble de 60 instances appelées *solutions*. Chaque solution est constituée d'un ensemble de coefficients dont la somme totale fait 1. La première *génération* G_0 est un ensemble de solutions initialisé avec des valeurs aléatoires. Les générations suivantes sont construites en faisant des *mutations* et des *croisements*.

Mutation. Étant donnée une génération G_t, chaque solution est mutée 4 fois, donnant 4×60 solutions mutées. Chaque mutation consiste en un transfert aléatoire d'une partie de la valeur d'un coefficient (compris entre 0 et la valeur du coefficient) choisi aléatoirement vers un autre coefficient. Un nouvel ensemble de 40 solutions est sélectionné à partir des solutions originales et des solutions mutées, en prenant parmi eux les 30 meilleures solutions et 10 autres choisies aléatoirement. Les meilleures solutions sont celles obtenant les séparations en voix les plus proches des fichiers de références.

Croisement. Les solutions de cet ensemble sont celles utilisées pour générer les 20 solutions enfants en prenant aléatoirement un couple de parents. Chaque parent est sélectionné une seule et unique fois, et une solution enfant est la moyenne des coefficients des solutions

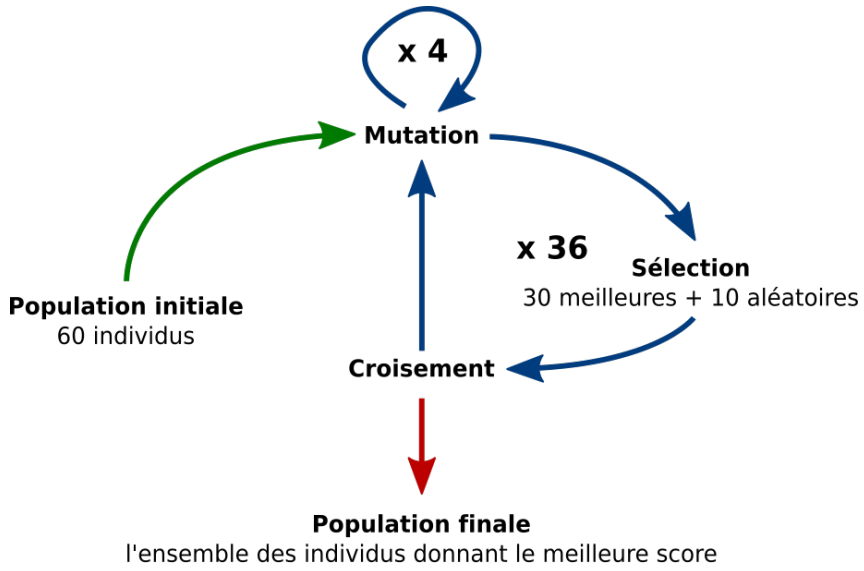


FIGURE 4.5 – Étapes principales de notre algorithme génétique.

parents. La nouvelle génération G_{t+1} est formée par les solutions des 40 parents et des 20 enfants.

4.1.5 Implémentation

J'ai implémenté cet algorithme génétique décrit précédemment et je l'ai lancé sur un calculateur appelé MeCS (`https://www.mecs.u-picardie.fr/`) hébergé par l'université Picardie Jules Verne. Il dispose de 288 coeurs CPU et d'une baie de stockage d'une capacité d'environ 100 To. De manière à profiter au mieux des capacités de ce calculateur, j'ai parallélisé le code de mon algorithme génétique pour permettre le lancement des 60 solutions.

4.1.6 Résultats

J'ai entraîné les valeurs des coefficients des caractéristiques avec l'algorithme génétique décrit ci-dessus sur les 24 fugues du premier livre du *Clavier bien tempéré* par J.-S. Bach (corpus `wtc-i`). Au bout de 36 générations, correspondant à environ 4608 heures de temps CPU, le processus s'est stabilisé sur les coefficients que je nomme GA1. Ensuite, j'ai évalué ces coefficients GA1 et d'autres méthodes de connexion sur les 24 fugues du livre deux du *Clavier bien tempéré* (corpus `wtc-ii`), sur 17 premiers mouvements de quatuor à cordes de musique classique et romantique (corpus `quatuor` – voir la section 3.2.1) et sur les 12 premières fugues de l'opus 87 de D. Shostakovitch (corpus `Opus 87`).

Coefficients sélectionnés

La colonne GA1 de la table 4.1 montre les coefficients appris qui donnent la meilleure solution. La valeur importante du coefficient *no_crossed_voices(C)* confirme que les croisements de voix devraient être évités jusqu'à ce que des algorithmes spécifiques puissent traiter ces cas. Je fais deux autres observations :

— La hauteur est la caractéristique la plus importante (les quatre coefficients de *pitch* totalisent 0.271). Toutefois, le coefficient de *avg_pitch_right(C)* est plus haut que le coefficient de *extreme_pitch(C)* – et la somme des coefficients de *avg_pitch_left(C)*, *avg_pitch_right(C)* et *avg_pitch(C)* donne 0.181, soit deux fois le coefficient de *extreme_pitch(C)*. Ceci confirme que l'utilisation d'une échelle de hauteur cohérente est plus fiable que d'utiliser uniquement la hauteur d'une seule note ;

— Les durées sont aussi des caractéristiques importantes, particulièrement quand on prend les durées moyennes (*avg_dur(C)* ou *avg_dur_right(C)*, totalisant 0.121). Le coefficient de *extreme_dur(C)* est très bas, confirmant l'idée que même si les durées individuelles changent, les rythmes ou les motifs à petite échelle sont conservés à l'intérieur des fragments de voix.

Finalement, la caractéristique *increase_equal(i)* suggérée par IMS est haute, mais, de manière surprenante, la caractéristique *decrease_equal(i)* est aussi haute. Ces deux caractéristiques combinées semblent souligner le fait que la connexion est plus sûre quand les contigs ont le même nombre de voix.

Dans la table 4.1, je présente également les différents jeux de valeurs données aux coefficients des caractéristiques pour reproduire le comportement des algorithmes précédents. Pour la simulation de l'algorithme CW (SimCW), je donne une valeur importante (0.5) au coefficient de la caractéristique *maximal_voices(i)* pour respecter la contrainte de CW de commencer par les contigs maximaux. Le reste du poids est réparti entre les deux coefficients des caractéristiques *no_crossed_voices(C)* et *extreme_pitch(C)* qui permettent de limiter au maximum les croisements de voix et de connecter les fragments les plus proches en termes de différence de hauteurs. En ce qui concerne la simulation de l'algorithme IMS (SimIMS), je donne une valeur identique de 0.25 aux coefficients des caractéristiques *increase_equal(i)* et *minimal_voices(i)* pour commencer par connecter les contigs ayant un nombre croissant de voix. Comme la connexion entre fragments est identique à celle de CW, les coefficients des caractéristiques *no_crossed_voices(C)* et *extreme_pitch(C)* reçoivent chacun une valeur de 0.25 pour les mêmes raisons que SimCW.

Qualité des méthodes de connexions

Essayons maintenant de répondre aux deux questions principales de ce chapitre : *Quels* contigs doivent être connectés ? *Comment* les connecter ? Pour pouvoir mesurer la qualité de réponse à ces questions, j'utilise différentes mesures : d'une part, les mesures TR-sen/TR-prec et S_o/S_u, expliquées dans la section 3.1, qui permettent de voir l'impact des caractéristiques sur la qualité globale de la séparation en voix ; d'autre part, une nouvelle mesure (CC) qui nous donne le ratio du nombre de connexions correctes sur le nombre total de connexions.

Meilleur que l'original ?

La table 4.2 détaille les résultats des différentes méthodes d'évaluation sur le corpus d'entraînement ainsi que sur les autres corpus en fonction des différents jeux de valeurs données aux coefficients des caractéristiques (GA1, SimCW et SimIMS, voir la table 4.1).

Les résultats de CW et de IMS sont légèrement différents des résultats de la section 3.2. Ceci est dû au fait que je n'ai pas utilisé la même implémentation. En effet, l'objectif de la section précédente était de montrer le rapport entre voix et stream, alors que dans cette section je cherche à montrer les différences de résultats en fonction de plusieurs techniques pour le choix de connexion des contigs (et de leurs fragments). C'est pourquoi j'ai choisi ici d'évaluer les différentes méthodes de connexions au sein de la même implémentation.

Caractéristiques	GA1	SimCW	SimIMS
increase(i)	0.004	0	0
increase_one(i)	0.004	0	0
increase_equal(i)	**0.137**	0	0.250
decrease(i)	0.013	0	0
decrease_one(i)	0.019	0	0
decrease_equal(i)	**0.112**	0	0
difference_nb_voices(i)	0.009	0	0
maximal_voices(i)	0.026	0.500	0
minimal_voices(i)	0.007	0	0.250
maximal_sim_notes(i)	0.007	0	0
crossed_voices(C)	0.009	0	0
no_crossed_voices(C)	**0.248**	0.250	0.250
extreme_pitch(C)	**0.090**	0.250	0.250
avg_pitch_right(C)	**0.117**	0	0
avg_pitch_left(C)	0.023	0	0
avg_pitch(C)	0.041	0	0
extreme_dur(C)	0.007	0	0
avg_dur_right(C)	0.048	0	0
avg_dur_left(C)	0.006	0	0
avg_dur(C)	**0.073**	0	0

TABLE 4.1 – Coefficients de pondération des caractéristiques musicales utilisées pour mesurer la qualité de la connexion, avec les meilleures coefficients appris en utilisant le corpus `wtc-i` (GA1) et les coefficients simulant les méthodes de connexion de CW et IMS.

Sur l'ensemble des corpus, les coefficients GA1 obtiennent de meilleures résultats sur TR-prec/TR-sen/CC que les coefficients SimCW et SimIMS. En effet, les coefficients GA1 font de meilleures connexions, plus de 87% de connexions sont correctes sur le corpus test `wtc-ii`. Nous voyons également qu'en considérant la première étape de CW comme un algorithme de séparation en streams (voir la section 2.4.6), les résultats sont excellents sur TR-prec (supérieur à 99% pour les deux corpus de fugues – lignes « sans-connexion » dans la table 4.2) mais moins bon sur TR-sen. Cette différence est due au fait que la pièce est extrêmement segmentée (en fragments) mais qu'à l'intérieur de ces fragments très peu d'erreurs sont commises. La principale source d'amélioration des résultats avec les coefficients GA1 vient du fait que les nouvelles caractéristiques prennent en considération une hauteur et/ou une longueur moyenne, c'est ce que l'on peut voir sur la figure 4.6.

4.2 Stopper les connexions au bon moment

Nous avons vu dans la partie précédente qu'en modifiant l'ordre de connexion des contigs ainsi que le choix des fragments à connecter, nous pouvions améliorer la qualité de séparation en voix. Dans la section 3.2, nous avons montré que les algorithmes de séparation en streams sont similaires à ceux de séparation en voix. Afin d'améliorer les résultats, ne pourrait-on pas terminer la connexion des contigs avant la fin d'un algorithme de séparation de voix, et donc terminer par des streams non connectés sur toute la pièce ?

4.2.1 Pourquoi stopper la connexion ?

L'objectif initial pour lequel je me suis intéressé à la segmentation était de faciliter la recherche et l'inférence de motifs dans des partitions polyphoniques. Nous avons vu dans la section 4.1 que la première étape de l'algorithme CW ne fait que très peu d'erreurs (moins de

Corpus	Type de connexion	CC	TR-sen	TR-prec	S_o	S_u
wtc-i (corpus d'entraînement)	Sans Connexion	–	86.78%	**99.32%**	**0.98**	0.34
	GA1-75%	**92.61%**	93.45%	98.54%	0.91	0.42
	GA1	**89.30%**	**97.84%**		**0.72**	**0.72**
	pire	16.93%	85.25%		0.06	0.09
	SimCW	81.26%	96.58%		0.65	0.64
	SimIMS	80.62%	96.55%		0.68	0.69
wtc-ii	Sans Connexion	–	86.66%	**99.29%**	**0.98**	0.35
	GA1-75%	**92.54%**	92.53%	98.36%	0.91	0.40
	GA1	**87.50%**	**97.14%**		**0.71**	**0.71**
	pire	25.06%	84.22%		0.05	0.07
	SimCW	83.27%	96.22%		0.69	0.68
	SimIMS	81.61%	96.07%		0.69	0.68
Opus 87	Sans Connexion	–	84.09	99.06	0.99	0.3
	GA1-75%	97.04	87.27	98.46	0.9	0.31
	GA1	94.35	96.89%		0.47	0.45
	pire	56.92	73.4		0.06	0.1
	SimCW	93.59	95.79%		0.5	0.49
	SimIMS	90.98	95.09%		0.38	0.36
quatuor	Sans Connexion	–	82.61%	**97.00%**	**0.94**	0.29
	GA1-75%	**85.30%**	87.06%	94.80%	0.83	0.32
	GA1	**78.44%**	**92.59%**		0.44	0.44
	pire	31.88%	80.59%		0.12	0.13
	SimCW	75.99%	92.29%		0.39	0.38
	SimIMS	74.53%	91.79%		**0.62**	**0.61**

TABLE 4.2 – Évaluation de la qualité des différentes méthodes de connexions. Notons que les deux premières méthodes (Sans Connexion, GA1-75%) ne connectent pas l'ensemble des contigs de la pièce : leurs mesures de TR-prec/ et S_o sont très hautes mais, étant donné qu'elles prédisent un nombre de voix supérieur à celui des fichiers de référence, leurs mesures de TR-sen et S_u sont plus faibles.

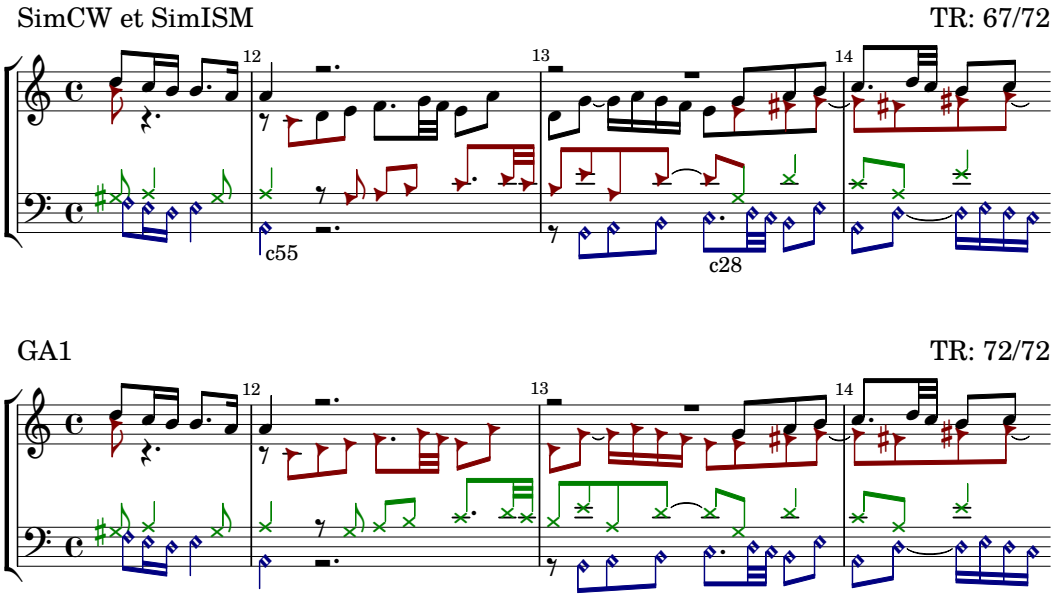


FIGURE 4.6 – Un extrait de la Fugue #1 du livre I du *Clavier bien tempéré* de Bach (en do majeur, BWV 846). (En haut.) La méthode de connexion des précédents algorithmes échouait à la connexion c28 à cause de la quinte entre le ré et le sol de la ténor. Cette erreur conduit à une mauvaise connexion c55 à une étape ultérieure de l'algorithme. (En bas.) Grâce aux valeurs des coefficients GA1 concernant la caractéristique *avg_pitch(C)* et les caractéristiques en relation avec celle-ci, les connexions sont correctes ici.

1%). Nous avons également vu qu'en utilisant des caractéristiques musicales nous pouvions améliorer les connexions entre contigs et entre fragments. Avec les coefficients SimIMS et encore plus avec les coefficients GA1, les premières connexions sont en générales sûres, les erreurs ont tendance à se faire dans les dernières connexions des contigs (voir la figure 4.7). C'est pourquoi j'ai décidé de rechercher le moment donnant un bon compromis entre la longueur des fragments et la qualité de segmentation.

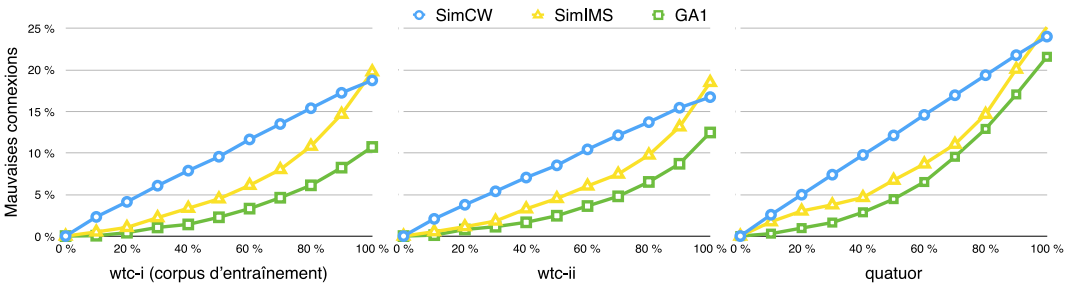


FIGURE 4.7 – Pourcentage de mauvaises connexions en fonction du nombre total des connexions. Plus la courbe est basse, meilleurs sont les résultats. Coefficients SimCW (en bleu) : les erreurs se produisent en continu, de manière constante. Coefficients SimIMS (en jaune) : les premières connexions sont plus fiables. Coefficients GA1 (en vert) : les premières connexions sont encore plus fiables, mettant bien en évidence l'intérêt d'arrêter l'algorithme avant que toutes les connexions soient faites. Le grand nombre de mauvaises connexions pour les quatuor à cordes (comparées aux fugues) est sans doute dû à une écriture polyphonique moins régulière, avec en particulier des différences stylistiques conduisant à des intervalles plus grands.

4.2.2 Quand stopper les connexions ?

En observant les courbes de GA1 (voir la figure 4.7), nous pouvons voir que les erreurs arrivent surtout dans les dernières connexions des contigs. Plus précisément, plus de 50% des erreurs totales proviennent du dernier tiers des connexions. Pour garder une cohérence dans les streams, en ayant un bon ratio entre précision et sensibilité, je décide de stopper les connexions lorsque 75% d'entre elles ont été effectuées. La ligne « GA1-75% » de la table 4.2 montre que l'on gagne ainsi plus de 6,5% sur TR-sen pour une perte de seulement 0,78% sur TR-prec. On a prédit 1902 streams pour 185 voix (soit en moyenne 10 streams pour 1 voix), alors qu'en allant jusqu'au bout, on fait 135 erreurs dans les 237 connections restantes.

4.3 Bilan

J'ai réussi à améliorer les résultats de l'algorithme CW en modifiant l'ordre de connexion des contigs ainsi que la manière de connecter les fragments à l'aide de caractéristiques musicales. Les valeurs des coefficients données à ces caractéristiques ont été déterminées à l'aide d'un algorithme génétique et ces valeurs sont identiques quels que soient les contigs à connecter. Un moyen qui pourrait permettre d'améliorer encore la connexion entre contigs serait de modifier les valeurs des coefficients des caractéristiques en fonction des contigs (nombre de fragments, durée, différence de hauteur...). Par exemple, faire une moyenne des hauteurs n'a pas beaucoup de sens pour de trop petits ou de trop long fragments (voir la figure 4.8).

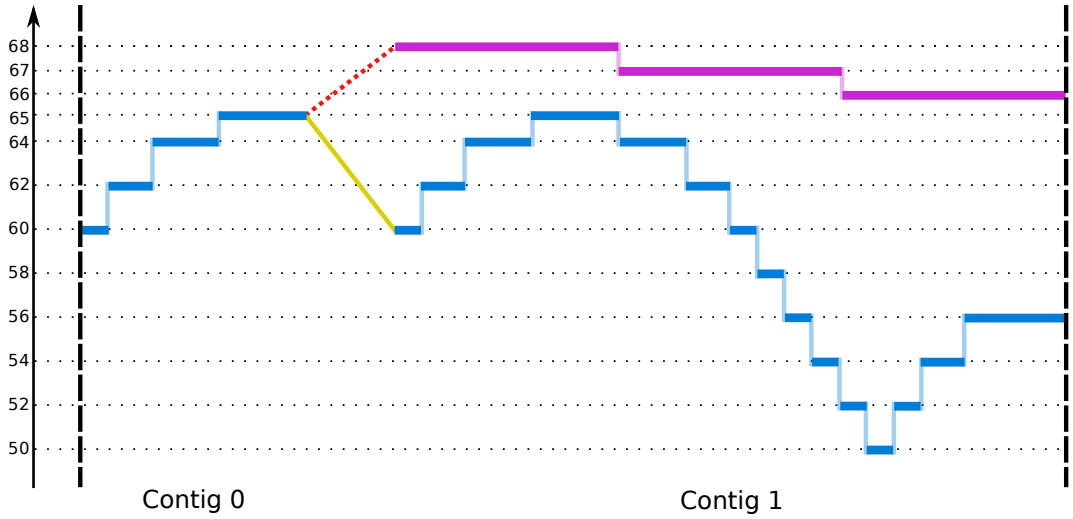


FIGURE 4.8 – L'erreur de connexion entre les contigs 0 et 1 (en pointillé) est faite en prenant la moyenne des hauteurs de l'ensemble des notes de chaque fragments. Pour réussir à correctement connecter ces deux contigs (trait plein), il faut limiter à la taille du fragment pour faire la moyenne des hauteurs (ici cela correspond aux sept premières notes du second fragment du contig 1).

Même si j'obtiens de bons résultats en terme de séparation en voix, il reste des cas difficiles à traiter, comme par exemple les croisements de voix, les fortes variations de la hauteur des notes ou les changements rythmiques. Dans la figure 3.18, aucun algorithme n'a été capable de détecter le croisement de voix dans le sujet. Une solution serait de donner à notre algorithme une liste de motifs (voir la section 5.2) qui permettrait de connecter ensemble des fragments ne répondant pas aux critères habituels de connexions.

Dans la dernière partie de ce chapitre, j'ai montré l'intérêt d'arrêter la connexion des contigs avant la fin pour avoir un bon rapport entre la qualité et le nombre des streams. Pour l'instant l'unique critère pour stopper les connexions est celui du pourcentage d'erreurs commises. Diminuer le nombre de streams évite d'avoir des streams ne contenant que quelques notes. Il pourrait donc s'avérer intéressant d'avoir également une condition d'arrêt sur la taille des fragments (nombre de notes ou durée du fragment). Une autre perspective serait d'adapter l'arrêt de la connexion à chaque pièce et non pas de se baser sur la moyenne des erreurs d'un corpus.

Dans la suite de mon travail, je combine la séparation de voix avec des problématiques d'analyse musicale se basant sur la recherche de motifs.

Chapitre 5

Conclusions et perspectives

L'objectif de cette thèse était de faciliter le traitement informatique de la polyphonie. Je me suis intéressé à la séparation en streams et en voix.

Ma première contribution a montré la *similarité entre la séparation en voix et la séparation en streams*. Pour cela, j'ai utilisé comme mesure d'évaluation la transition entre deux notes successives. Les résultats donnés par cette mesure ont permis de comparer de manière équitable les deux approches et de souligner les performances des approches basées par contigs, dont l'algorithme de Chew et Wu (CW).

Ma seconde contribution a proposé un *ensemble de caractéristiques musicales* pour mieux connecter les contigs en quantifiant la probabilité d'appartenance à la même voix de deux fragments de notes contigus. Par exemple, l'une des caractéristiques compare la moyenne des hauteurs de deux groupes de notes. Dans la majorité des études sur la séparation en voix et en streams, la différence de hauteur entre deux notes est l'un des critères les plus importants. S'intéresser à la différence des hauteurs entre deux *groupes de notes* permet à l'algorithme de mieux connaître l'environnement dans lequel se fait la connexion. À l'aide de l'ensemble des caractéristiques j'ai pu déterminer un ordre de connexion des contigs et une manière de connecter leurs fragments qui donne de meilleurs résultats que ceux de CW. Il reste néanmoins des cas difficiles à traiter, comme par exemple les croisements de voix, les fortes variations de la hauteur des notes ou les changements rythmiques.

En conclusion de cette thèse, j'aimerais proposer quelques perspectives pour rapprocher le problème de séparation en voix et streams de problématiques d'analyse musicale (ici de recherche de motifs dans les fugues) : peut-on évaluer la séparation polyphonique par des résultats d'analyse musicale ? Réciproquement, peut-on utiliser certains éléments d'analyse pour encore améliorer la séparation polyphonique ?

5.1 Faire une analyse musicale basée sur une séparation en voix

Une première direction pour coupler analyse musicale et traitement de données polyphoniques est de proposer une nouvelle évaluation de qualité. Les algorithmes de séparation en voix et en streams permettent-ils d'effectuer correctement certaines tâches d'analyse musicale ?

J'ai ainsi commencé à évaluer les résultats d'un algorithme de recherche du *sujet des fugues* développé dans l'équipe [22]. J'applique cet algorithme sur des fichiers polyphoniques séparés en voix par les quatre algorithmes présentés dans les sections 2.3 et 4.1 : trivial, SimCW, SimIMS et GA1.

Le tableau 5.1 présente ces résultats préliminaires. C'est l'algorithme SimCW qui donne les meilleurs résultats dans le corpus des fugues de Bach lorsqu'on regarde les prédictions absolues des voix (64% de sujets). Cependant, dès qu'on évalue la présence de sujets dans n'importe quelle voix (voir figure 5.1), c'est notre algorithme GA1 qui est le meilleur (80% des sujets des fugues du livre I du *Clavier bien tempéré* de J. S. Bach, contre 73% pour SimCW). Dans la Fugue #18 du livre I du *Clavier bien tempéré* de J.-S. Bach, suite à une erreur dans la séparation de voix, les algorithmes SimCW et SimIMS sont dans l'incapacité de trouver la première occurrence du sujet jouée par la basse. À l'inverse l'algorithme GA1, en utilisant la caractéristique de moyenne des hauteurs, arrive à séparer correctement les voix et à retrouver cette occurrence (voir la figure 5.2).

Ces résultats préliminaires confirment les idées proposées dans le chapitre précédent : avoir un ensemble de permet de relier des contigs avec plus de sens musical, et donc finalement ici d'identifier plus de motifs pertinents pour l'analyse.

Algorithme	Corpus	Vrai Positif	Faux Positif	Faux Négatif	Sensibilité	Précision
Trivial	wtc-i	31	72	260	10,65%	30,1%
SimCW		**185**	71	**106**	**63,57%**	72,26%
SimIMS		171	100	120	58,76%	63,1%
GA1		163	105	120	56,01%	60,82%
Monophoniques		266	29	25	91,41%	90,17%
Trivial	Shosta	3	31	171	1,72%	8,82%
SimCW		43	24	131	24,71%	64,18%
SimIMS		39	36	135	22,41%	52%
GA1		**44**	37	**130**	**25,29%**	54,32%
Monophoniques		94	12	80	54,02%	88,68%
Algorithme	Corpus	Vrai Positif	Faux Positif	Faux Négatif	Sensibilité	Précision
Trivial	wtc-i	47	56	244	16,15%	45,63%
SimCW		213	46	78	73,2%	82,2%
SimIMS		217	44	74	74,57%	83,14%
GA1		**230**	38	**61**	**79,04%**	85,82%
Monophoniques		266	29	25	91,41%	90,17%
Trivial	Shosta	14	20	160	8,05%	41,18%
SimCW		69	4	105	39,66%	94,52%
SimIMS		70	2	104	40,21%	97,22%
GA1		**77**	4	**97**	**44,25%**	95,06%
Monophoniques		94	12	80	54,02%	88,68%

Table 5.1 – Résultats de l'algorithme de recherche et d'inférence du sujet de fugue [22] sur les 24 fugues du livre I de J. S. Bach et sur les 12 premières fugues de l'opus 87 de D. Shostakovitch après séparation des données polyphoniques en voix par les algorithmes : trivial, SimCW, SimIMS, GA1. L'algorithme monophonique donne les résultats de la recherche en utilisant en entrée des données monophoniques donnant la « bonne » séparation en voix (utilisation de ficher **Kern).

(En haut.) Un sujet prédit dans une voix différente de celle de la vérité est considéré comme un faux positif. L'algorithme SimCW commençant par connecter les fragments provenant des contigs maximaux, il a l'information du nombre de voix de la pièce dès ces premières connexions. À l'inverse, les algorithmes SimIMS et GA1 n'ont cette information qu'à la fin de l'étape de connexion, ce qui peut induire des erreurs dans l'ordre des voix prédites.

(En bas.) Dans cette évaluation, la prédiction se trouvant dans l'une des voix de l'analyse de référence sera considérée comme un vrai positif.

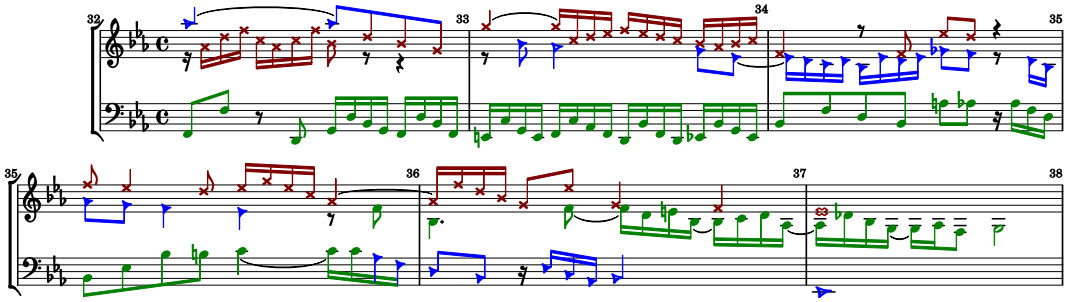


FIGURE 5.1 – Les notes en forme de croix correspondent à la voix soprano, celles en forme rondes correspondent à l'alto et celles en forme de triangles à la ténor. L'algorithme GA1 commet une erreur une croche avant la fin de la mesure 35 en prédisant le do de la ténor dans l'alto. L'erreur se propage aux mesures précédentes et a comme conséquence de prédire la majorité des notes de la ténor dans l'alto. Malgré cette erreur, l'ensemble du sujet est bien prédit dans la même voix (la ténor) à la mesure 34.

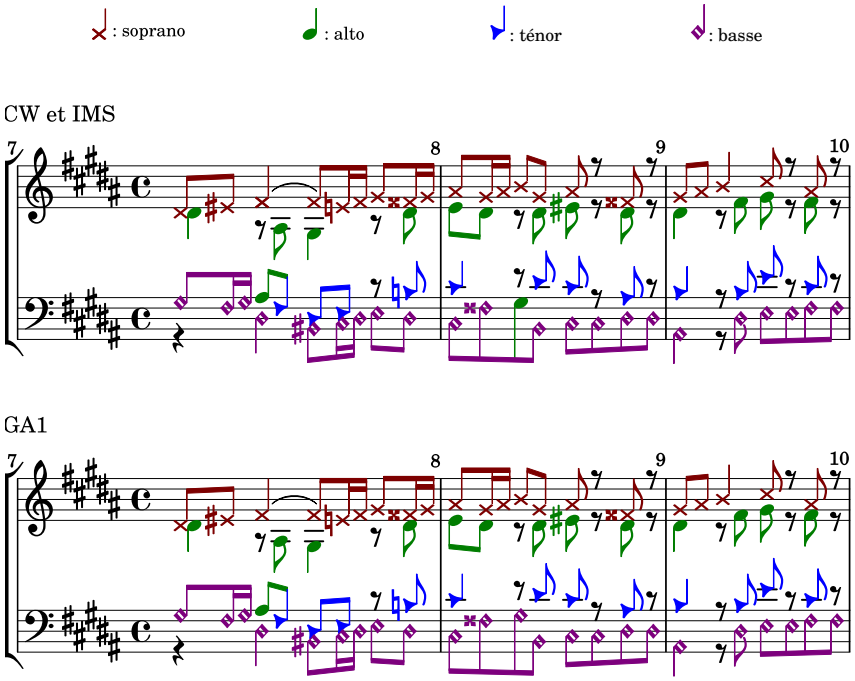


FIGURE 5.2 – L'erreur commise sur le deuxième temps de la basse mesure 8 par les algorithmes SimCW et SimIMS rend impossible l'identification du sujet. En donnant un coefficient suffisamment important à la caractéristique de moyenne des hauteurs, l'algorithme GA1 arrive à placer l'ensemble du sujet dans le même fragment - Fugue #18 du livre I du *Clavier bien tempéré* de J.-S. Bach

5.2 Améliorer la séparation polyphonique par la recherche de motifs

Enfin, ne serait-ce pas possible d'utiliser des notions d'analyse musicale pour améliorer la séparation de voix et de streams ? Quelque part, ne cherchons-nous pas à créer des contigs qui seraient des motifs pertinents pour l'analyse musicale ?

Motifs et connexions. Pour déterminer l'ordre de connexion des contigs et les choix de connexion des fragments, on pourrait donner à un algorithme une liste de motifs qui ferait une nouvelle caractéristique, *pattern_matching(C)*. En effet, en encourageant la connexion de deux fragments issus d'un même motif, on pourrait traiter des cas difficiles en acceptant par exemple de faire un croisement de voix si le fragment créé contient l'un des motifs de la liste, comme dans le cas de la Fugue #16 du livre I du *Clavier bien tempéré* de J.-S. Bach présentée dans la section 3.2.3 (voir la figure 3.18).

Pour que ce paramètre soit pertinent, les motifs doivent être significatifs – pour une fugue, on peut prendre encore un motif thématique comme le sujet. Une simple heuristique peut permettre d'identifier la partie monophonique du début des pièces (voir la figure 5.3).

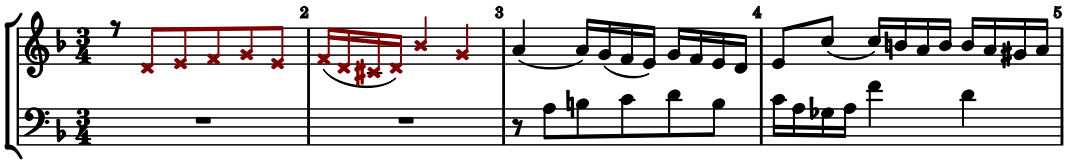


FIGURE 5.3 – Les notes en croix forment le motif qui précède l'arrivée d'une nouvelle voix dans la pièce - Fugue numéro #6 du livre I du *Clavier bien tempéré* de J. S. Bach (en Fa majeur, BWV 851). Peut-on se servir de ce motif pour améliorer la séparation en voix dans le reste de la pièce ?

Les premières expériences sur cette idée ne sont pas concluantes. En choisissant manuellement la valeur du paramètre *pattern_matching(C)* afin qu'il prime sur les autres, le nombre de sujets détectés dans le corpus wtc-i avec ce nouveau paramètre est de 223 alors qu'il était de 230 précédemment et qu'il est identique pour le corpus Shosta. De nombreux questions seraient à explorer : comment évaluer le motif ? Doit-il figurer entier, en partie, approché ? Là encore, on pourrait peut-être optimiser les coefficients non pas sur la séparation des voix mais sur le résultat en termes d'analyse musicale.

Arrêt des connexions. Une autre idée dans cette direction serait de travailler à l'*arrêt des connexions*. J'ai montré l'intérêt qu'il y aurait à arrêter la connexion des contigs avant la fin afin d'avoir un bon rapport entre la qualité et le nombre des streams (voir la section 4.2). Néanmoins pour l'instant l'unique critère pour stopper les connexions est celui du pourcentage d'erreurs commises. Avec cette méthode d'arrêt, nous ne nous assurons pas d'avoir des streams contenant plus de quelques notes. Il pourrait donc s'avérer intéressant d'avoir également une condition d'arrêt sur la *taille des fragments* (nombre de notes ou durée du fragment) pour que les streams formés soient le plus proche possible de motifs pertinents.

5.3 Bilan

Les recherches effectuées dans cette thèse auront permis d'apporter une aide à l'analyse automatique de partitions en proposant des techniques qui permettent d'organiser l'information polyphonique, en permettant de mieux séparer une polyphonie en voix ou en streams.

Les caractéristiques introduites dans le chapitre 4 pourront être réutilisées dans d'autres défis d'informatique musicale sur de la musique polyphonique, notamment la caractéristique permettant d'attribuer une hauteur moyenne à un ensemble de notes. Nous avons vu précédemment (voir la section 2.4) que la notion de stream pouvait varier en fonction des auteurs. Par exemple, pour Makris *et al.* [45], la définition de stream se rapproche de celle de la texture [23]. Les nouvelles caractéristiques que je propose pourraient aider à trouver des éléments relevant de la *texture* comme par exemple la mélodie ou l'accompagnement.

Ces caractéristiques pourraient être également utilisées dans le cadre de l'interprétation en temps réel d'une pièce pour faire de la détection de voix. Enfin, un autre cas d'utilisation serait celui du calcul de la proximité entre deux pièces, par exemple pour faire de la recherche de motifs qui pourrait aider à la détection de plagiat.

Bibliographie

[1] Samer Abdallah, Katy Noland, Mark Sandler, and Mark S. Theory and evaluation of a bayesian music structure extractor. In *International Conference on Music Information Retrieval (ISMIR 2005)*, pages 420–425, 2005.

[2] Juan Pablo Bello, Giuliano Monti, and Mark B. Sandler. Techniques for automatic music transcription. 2000.

[3] Janssen Berit, Bas de Haas W., Volk Anja, and van Kranenburg Peter. Finding repeated patterns in music : State of knowledge, challenges, perspectives. In *International Symposium on Computer Music and Multidisciplinary Research (CMMR 2013)*, pages 277–297, 2013.

[4] Albert Donally Bethke. Genetic algorithms as function optimizers. In *ACM Computer Science Conference*, 1978.

[5] Marcel Bitsch and Noël Gallon. *Traité de contrepoint.* Durand, 1964.

[6] Siglind Bruhn. *J. S. Bach's Well-Tempered Clavier. In-depth Analysis and Interpretation.* Mainer International, 1993.

[7] Emilios Cambouropoulos. Musical parallelism and melodic segmentation. *Journal of New Music Research*, 23(3) :249–268, 2006.

[8] Claude Charlier. *Pour une lecture alternative du Clavier bien tempéré.* Jacquart, 2009.

[9] Christian Charras, Thierry Lecroq, and Joseph Daniel Pehoushek. A very fast string matching algorithm for small alphabets and long patterns. In *Annual Symposium on Combinatorial Pattern Matching*, pages 55–64, 1998.

[10] Elaine Chew and Xiaodan Wu. Separating voices in polyphonic music : A contig mapping approach. In *International Symposium on Computer Music Modeling and Retrieval (CMMR 2005)*, pages 1–20, 2005.

[11] Darrell Conklin and Christina Anagnostopoulou. Comparative pattern analysis of cretan folk songs. *Journal of New Music Research*, 40(2) :119–125, 2011.

[12] Tim Crawford and Richard Lewis. Musicxml for notation and analysis. *Journal of the American Musicological Society*, 69(1) :273–285, 2016.

[13] Max Crochemore. An optimal algorithm for computing the repetitions in a word. *Information Processing Letters*, 12(5) :244–250, 1981.

[14] Maxime Crochemore, Costas S. Ilioupoulos, Thierry Lecroq, and Yoan J. Pinzon. Approximate string matching in musical sequences. In *The Prague Stringology Conference (PSC 2001)*, pages 26–36, 2001.

[15] Michael Scott Cuthbert and Christopher Ariza. music21 : A toolkit for computer-aided musicology and symbolic music data. In *International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 637–642, 2010.

[16] Adolphe Danhauser. *Théorie de la musique.* Lemoine, 1872.

[17] Laurent David, Mathieu Giraud, Richard Groult, Corentin Louboutin, and Florence Levé. Vers une analyse automatique des formes sonates. In *Journées d'Informatique Musicale (JIM 2014)*, 2014.

[18] Reinier de Valk, Tillman Weyde, and Emmanouil Benetos. A machine learning approach to voice separation in lute tablature. In *International Society for Music Information Retrieval Conference (ISMIR 2013)*, pages 555–560, 2013.

[19] Diana Deutsch. Grouping mechanisms in music. *The psychology of music*, 28 :299–348, 1999.

[20] Académie française. *Dictionnaire de l'Académie française. Tome 3, Maq-Quo.* Paris : Imprimerie nationale éd. : Fayard, 9e éditions edition, 2011.

[21] Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. Computational fugue analysis. *Computer Music Journal*, 39(2), 2015.

[22] Mathieu Giraud, Richard Groult, and Florence Levé. Subject and counter-subject detection for analysis of the Well-Tempered Clavier fugues. In *International Symposium on Computer Music Modeling and Retrieval (CMMR 2012)*, pages 661–673, 2012.

[23] Mathieu Giraud, Florence Levé, Florent Mercier, Marc Rigaudière, and Donatien Thorez. Modeling texture in symbolic data. In *International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014.

[24] Michael Good. Musicxml for notation and analysis. *The virtual score : representation, retrieval, restoration*, 12 :113–124, 2001.

[25] Patrick Gray and Razvan Bunescu. A neural greedy model for voice separation in symbolic music. In *International Society for Music Information Retrieval Conference (ISMIR 2016)*, pages 782–788, 2016.

[26] Nicolas Guiomard-Kagan, Mathieu Giraud, Richard Groult, and Florence Levé. Comparing voice and stream segmentation algorithms. In *International Society for Music Information Retrieval Conference (ISMIR 2015)*, pages 493–499, 2015.

[27] Nicolas Guiomard-Kagan, Mathieu Giraud, Richard Groult, and Florence Levé. Improving voice separation by better connecting contigs. In *International Society for Music Information Retrieval Conference (ISMIR 2016)*, pages 192–198, 2016.

[28] James Hepokoski and Warren Darcy. *Elements of Sonata Theory : Norms, Types, and Deformations in the Late-Eighteenth-Century Sonata.* Oxford University Press, 2006.

[29] Andrew Hume and Daniel Sunday. Fast string searching. *Software : Practice and Experience*, 21(11) :1221–1248, 1991.

[30] David Huron. *Humdrum and Kern : Selective feature encoding*, pages 375–401. MIT Press, 1997.

[31] David Huron. Tone and voice : A derivation of the rules of voice-leading from perceptual principles. *Music Perception*, 19(1) :1–64, 2001.

[32] David Huron. Music information processing using the Humdrum toolkit : Concepts, examples, and lessons. *Computer Music Journal*, 26(2) :11–26, 2002.

[33] Asako Ishigaki, Masaki Matsubara, and Hiroaki Saito. Prioritized contig combining to segragate voices in polyphonic music. In *Sound and Music Computing Conference (SMC 2011)*, volume 119, 2011.

[34] Anna Jordanous. Voice separation in polyphonic music : A data-driven approach. In *International Computer Music Conference (ICMC 2008)*, 2008.

[35] Ioannis Karydis, Alexandros Nanopoulos, Apostolos Papadopoulos, Emilios Cambouropoulos, and Yannis Manolopoulos. Horizontal and vertical integration/segregation in auditory streaming : a voice separation algorithm for symbolic musical data. In *Sound and Music Computing Conference (SMC 2007)*, pages 299–306, 2007.

[36] Hermann Keller. *Das Wohltemperierte Klavier von Johann Sebastiann Bach.* Bärenreiter, 1965.

[37] Jürgen Kilian and Holger H Hoos. Voice separation – a local optimization approach. In *International Conference on Music Information Retrieval (ISMIR 2002)*, 2002.

[38] Phillip B Kirlin and Paul E Utgoff. Voise : Learning to segregate voices in explicit and implicit polyphony. In *International Conference on Music Information Retrieval (ISMIR 2005)*, pages 552–557, 2005.

[39] Donald Knuth, James Morris, and Vaughan Pratt. Fast pattern matching in strings. *SIAM journal on computing*, 6(2) :323–325, 1977.

[40] Olivier Lartillot. Multi-dimensional motivic pattern extraction founded on adaptive redundancy filtering. *Journal of New Music Research*, 34(4) :375–393, 2005.

[41] Yipeng Li and DeLiang Wang. Separation of singing voice from music accompaniment for monaural recordings. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(4) :1475–1487, 2007.

[42] Ada Lovelace. A sketch of the analytical engine, with notes by the translator. *Scientific Memoirs*, 3 :666–731, 1843.

[43] Hanna M Lukashevich. Towards quantitative measures of evaluating song segmentation. In *International Conference on Music Information Retrieval (ISMIR 2008)*, pages 375–380, 2008.

[44] Søren Tjagvad Madsen and Gerhard Widmer. Separating voices in midi. In *International Conference on Music Information Retrieval (ISMIR 2006)*, pages 57–60, 2006.

[45] Dimos Makris, Ioannis Karydis, and Emilios Cambouropoulos. Visa3 : Refining the voice intergration/segregation algorithm. 2016.

[46] Szeto Wai Man and Wong Man Hon. A graph-theoretical approach for pattern matching in post-tonal music analysis. *Journal of New Music Research*, 35(4) :307–321, 2006.

[47] Andrew McLeod and Mark Steedman. HMM-based voice separation of MIDI performance. *Journal of New Music Research*, 45(1) :17–26, 2016.

[48] David Meredith, Kjell Lemström, and Geraint A Wiggins. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, 31(4) :321–345, 2003.

[49] Marcel Mongeau and David Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24(3) :161–175, 1990.

[50] Robert Moog. Midi : musical instrument digital interface. *Journal of the Audio Engineering Society*, 34(5) :394–404, 1986.

[51] Oriol Nieto and Morwaread Mary Farbood. Perceptual evaluation of automatically extracted musical motives. In *International Conference on Music Perception and Cognition (ICMPC 2012)*, pages 723–727, 2012.

[52] Alexey Ozerov, Pierrick Philippe, Frédéric Bimbot, and Rémi Gribonval. Adaptation of bayesian models for single-channel source separation and its application to voice/music separation in popular songs. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(5) :1564–1578, 2007.

[53] Martin Piszczalski and Bernard A. Galler. Automatic music transcription. *Computer Music Journal*, 1(4) :24–31, 1977.

[54] Ebenezer Prout. *Analysis of J. S. Bach's forty-eight fugues (Das Wohltemperierte Clavier)*. E. Ashdown, 1910.

[55] Dimitrios Rafailidis, Alexandros Nanopoulos, Yannis Manolopoulos, and Emilios Cambouropoulos. Detection of stream segments in symbolic musical data. In *International Conference on Music Information Retrieval (ISMIR 2008)*, pages 83–88, 2008.

[56] Dimitris Rafailidis, Emilios Cambouropoulos, and Yannis Manolopoulos. Musical voice integration/segregation : Visa revisited. In *Sound and Music Computing Conference (SMC 2009)*, pages 42–47, 2009.

[57] David Rizo. *Symbolic music comparison with tree data structures*. PhD thesis, Univ. de Alicant, 2011.

[58] Perry Rolland. The music encoding initiative (mei). In *International Conference on Musical Applications Using XML*, pages 55–59, 2002.

[59] Pierre-Yves Rolland. Discovering patterns in musical sequences. *Journal of New Music Research*, 28(4) :334–350, 1999.

[60] Doraisamy Shyamala and Rüger Stefan. Robust polyphonic music retrieval with n-grams. *Journal of Intelligent Information Systems*, 21(1) :53–70, 2003.

[61] Daniel Sunday. A very fast substring search algorithm. *Communications of the ACM*, 33(8) :132–142, 1990.

[62] David Temperley. *The Cognition of Basic Musical Structures.* The MIT Press, 2001.

[63] Collins Tom, Jeremy Thurlow, and Robin Laney. A comparative evaluation of algorithms for discovering translational patterns in baroque keyboard works. In *International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 3–8, 2010.

[64] Donald Tovey, editor. *Forty-Eight Preludes and Fugues, J.-S. Bach.* Associated Board of the Royal Schools of Music, 1924.

[65] Rainer Typke. *Music retrieval based on melodic similarity.* PhD thesis, Univ. Utrecht, 2007.

[66] Esko Ukkonen, Kjell Lemström, and Veli Mäkinen. Geometric algorithms for transposition invariant content based music retrieval. pages 193–199, 2003.

[67] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2) :260–269, 1967.

Table des figures

Liste des tableaux

Index

Résumé

La musique peut être *monophonique* – une seule note est jouée à chaque instant – ou *polyphonique* – plusieurs notes sont jouées simultanément, formant des harmonies. Comprendre la musique polyphonique peut être très complexe. L'objectif de cette thèse en informatique musicale est de simplifier l'analyse de partitions polyphoniques en les décomposant en *voix* monophoniques ou en *streams* (ensembles cohérents de notes).

Ces deux approches n'ayant jamais été confrontées, mes premiers travaux consistent à comparer trois algorithmes de séparation en voix et trois algorithmes de séparation en streams. Je propose pour cela des méthodes d'évaluation équitables pour ces deux approches. Les tests réalisés sur un corpus de musique classique et de musique pop ont mis en avant les qualités de l'algorithme de séparation en voix de Chew et Wu. La première étape de cet algorithme, qui segmente la partition en « contigs » avec un nombre de voix constant, est particulièrement robuste.

La suite des travaux de cette thèse porte sur la seconde étape de l'algorithme de Chew et Wu, qui définit *l'ordre de connexions* des contigs et *la manière de les connecter*. J'améliore ces connexions en utilisant des paramètres musicaux comme la différence des moyennes des hauteurs des notes entre contigs voisins. La thèse se conclut en évaluant conjointement la séparation en voix et la recherche de motifs pour l'analyse musicale de fugues.

Abstract

Music can be either *monophonic* (a single note sounds at each time) or *polyphonic* (several notes sound simultaneously, building harmonies). Understanding polyphonic music can be very complex. The goal of this thesis in computer music is to ease the analysis of polyphonic scores by splitting them in either monophonic voices or streams (coherent sets of notes).

Research in this thesis first consists in comparing three voices separation algorithms and three streams separation algorithms. I propose an evaluation method to fairly compare these two approaches. This study shows the qualities of the Chew and Wu algorithm. The first step of this algorithm, which segments the score into "contigs" having a constant number of voices, is particularly robust.

Further work of this thesis focuses on the second stage of the Chew and Wu algorithm that defines *what contigs to connect* and *how to connect them*. I improve these connections by using musical parameters such as the average pitch difference between neighbor contigs. The thesis concludes by evaluating simultaneously voice separation and pattern matching for the music analysis of fugues.