Outils informatiques d'analyse musicale

Benoit MATHIEU[1]

Rapport de stage de fin d'étude
DEA MIASH, ENST-Bretagne, dept. IASC
Responsable : Gérard ASSAYAG[2], Laboratoire d'accueil : IRCAM
Coordinateur : Gilles COPPIN[3]

24 juillet 2002

[1]benoit.mathieu@enst-bretagne.fr
[2]gerard.assayag@ircam.fr
[3]gilles.coppin@enst-bretagne.fr

Résumé

Ce rapport de stage concerne un stage effectué à l'IRCAM autour des outils informatiques d'analyse musicale. Il est divisé en deux parties. La première partie est consacrée à l'état de l'art et une réflexion sur les outils actuels, les limitations et les nouvelles perspectives que nous pourrions envisager. La deuxième partie est consacrée aux réalisations qui ont été menées au cours du stage. Nous y verrons notamment une application de classification pour des accords. Egalement, à partir d'une proposition d'une nouvelle représentation, nous verrons une réalisation d'un outil d'« imagerie musicale », l'exploitation de cette représentation pour l'analyse de motifs dans des accords, et l'étude de transitions entre accords.

Mots-clés analyse musicale, analyse de données, représentation de connaissances, découverte de connaissances, classification, cognition, musicologie.

Remerciements

Je tiens à remercier l'IRCAM pour son accueil et toute l'équipe représentation musicale pour son aide et les discussions qui m'ont fait avancer. Je remercie tout particulièrement Carlos Agon pour ses explications et sa patience à propos d'OPENMUSIC , Olivier Lartillot pour la relecture du rapport, Charlotte Truchet pour son aide à propos des contraintes, Benoit Meudic pour les discussions politiques, Dominique Eav pour les parties de Go, et bien sûr, Gérard Assayag pour m'avoir proposé ce stage, et m'avoir guidé durant cette période.

Table des matières

Introduction

L'analyse musicale est une discipline qui se consacre à l'étude de la musique. Son évolution, notamment au cours du XXème siècle[1], montre qu'elle cherche sa place en tant que science en tentant de formaliser et objectiver les méthodes mises en oeuvre.

D'un autre coté la constitution de grandes bases de données, l'intelligence artificielle et la fouille de données, sont venus bouleverser le monde de l'analyse de données en proposant des techniques de plus en plus pertinentes et efficaces.

Enfin, les changements radicaux qui s'opèrent dans notre société avec les nouveaux moyens d'accès et de diffusion de l'information, sont déjà en train de modifier nos rapports avec la musique[2].

Ces trois points poussent l'IRCAM à s'intéresser de plus près à l'analyse musicale, et notamment à réfléchir sur son impact sur la réception des oeuvres, les supports hypermédias et les outils informatiques d'analyse musicale.

Le stage s'est principalement centré sur les outils d'analyse de la musique au niveau symbolique, et particulièrement orientés vers l'analyse musicale assistée par ordinateur, c'est à dire des outils susceptibles d'aider le musicologue dans son analyse de la musique.

Dans une première partie, nous essaierons de faire un aperçu des outils informatiques d'analyse musicale. Cela nous mènera à quelques réflexions sur les perspectives envisageables pour ces outils. Dans une deuxième partie, nous verrons les études et réalisations auquelles ce stage a donné lieu. Par exemple, nous verrons comment mettre en oeuvre une méthode de classification pour des accords, mais aussi nous exploiterons la proposition d'une représentation particulière de la musique pour réalisation d'outils de visualisation, d'analyse de motifs et d'étude de transitions entre accords.

[1] La *systematic musicology* montre une tendance à vouloir objectiver la démarche de l'analyste. C'est à dire que l'analyse va chercher à replacer les propriétés dans leur contexte, en comparant ses résultats sur une oeuvre avec ceux obtenus sur un modèle aléatoire.

[2] Cela est particulièrement flagrant au niveau de la diffusion. Une fois connecté à Internet, les possibilités, légales ou illégales, d'accès à la musique sont nombreuses. Ceci est radicalement nouveau, par rapport au système de distribution supervisé par les majors.

Chapitre 1

Contexte

1.1 L'IRCAM

1.1.1 Présentation

Fondé en 1969 par Pierre Boulez, l'Institut de Recherche et Coordination Acoustique/Musique est associé au Centre Pompidou et dirigé de 1992 à 2002 par Laurent Bayle. L'IRCAM réunit autour de la musique, des scientifiques et des musiciens (compositeurs, interprètes).

L'IRCAM concentre son activité autour de trois missions :

Chercher L'IRCAM mène des recherches fondamentales sur les apports de l'informatique, de la physique et de l'acoustique à la problématique musicale. Elles ont pour vocation principale la mise au point d'outils logiciels qui viennent enrichir l'invention du compositeur et suscitent des échanges internationaux avec les grandes institutions universitaires ou de recherche.

Créer L'IRCAM invite dans ses studios de nombreux compositeurs. Chaque année 20 à 25 oeuvres sont réalisées, qui associent des interprètes classiques (instrumentistes et chanteurs) et des nouvelles techniques. Ces musiques sont ensuite présentées au public lors de manifestations organisées conjointement avec l'Ensemble Intercontemporain.

Transmettre L'IRCAM propose plusieurs programmes pédagogiques, dont un DEA, un cursus d'informatique musicale pour compositeurs, et de nombreux ateliers, conférences, débats. L'IRCAM diffuse également ses activités sous forme de livres et revues, de disques compacts et de CD-Roms.

Les activités de l'IRCAM ont jusqu'à lors été centrées autour de la création. Depuis 2002, le nouveau directeur, Bernard Stiegler, travaille au lancement d'un nouvel axe de recherche, celui de l'analyse.

1.1.2 L'analyse musicale

Le développement des télécommunications, notamment via les technologies numériques et la numérisation de la musique, sont venu bouleverser les modes de diffusions traditionnels de la musique. L'IRCAM souhaite prendre un rôle actif dans ce développement. C'est pourquoi Bernard Stiegler se tourne vers l'analyse musicale, discipline jusqu'alors restée l'apanage des musicologues. En effet, l'analyse musicale produit des artéfacts autour de la musique qui viennent modifier la réception[1] de l'oeuvre. Dans un cadre de diffusion, la production de documents multimédias, de représentations annotées ou autres, pourraient venir contribuer à une meilleure compréhension.

Dans une première étape, Bernard Stiegler a organisé des groupes de réflexion autour de l'analyse musicale, afin de permettre l'émergence de nouvelles idées qui seront développées dans une deuxième étape.

Le groupe « outils pour l'analyse »réunit des chercheurs en informatique et traitement du signal, ainsi que des musicologues. La réflexion porte sur les outils informatiques d'analyse, les besoins des musicologues, les outils existants, etc.

1.2 L'équipe Représentations Musicales

L'équipe Représentation Musicale travaille sur l'utilisation de l'informatique en musique, au niveau symbolique. Les membres permanents, Gérard Assayag et Carlos Agon, sont les auteurs de OPENMUSIC , le logiciel phare de l'équipe.

1.2.1 OPENMUSIC

OPENMUSIC est un logiciel de composition assistée par ordinateur, disponible sur Macintosh. Logiciel écrit en CLOS[2], OPENMUSIC a bénéficié de l'expérience acquise avec les logiciels précédents, notamment PATCHWORK. Il propose des fonctionnalités radicalement nouvelles par rapport à ce qui se fait dans le domaine.

OPENMUSIC fournit des outils pour créer et manipuler une représentation symbolique de la musique. Plusieurs librairies dressent également des ponts vers la synthèse sonore. Plus qu'un simple logiciel de CAO[3] , OPENMUSIC offre une interface de programmation visuelle permettant la définition et la manipulation de fonctions LISP et d'objets CLOS. Ceci fait de lui une véritable surcouche du langage CLOS. Les objets sont représentés par des icônes qui peuvent être manipulées et connectées directement à la souris.

[1] Il s'agit de la compréhension de l'oeuvre par le public.

[2] Common Lisp Object System est un langage orienté objets basé sur le langage Common Lisp.

[3] Composition Assistée par Ordinateur

1.2.2 L'analyse dans OPENMUSIC

Initialement conçu pour l'aide à la composition, OPENMUSIC n'est pas un outil spécialement dédié à l'analyse musicale. Cependant, on peut quand même l'utiliser pour certaines tâches. Par exemple, Gérard Assayag, Carlos Agon et Moreno Andreatta ont utilisé OPENMUSIC pour la reconstitution d'oeuvres de Xenakis : *Herma* et *Nomos Alpha*. Certaines librairies permettent aussi de réaliser des calculs qui relève de l'analyse.[4]

Etant plutot orienté composition, OPENMUSIC a participé à l'émergence d'un type d'analyse particulier, qui est celui de la reconstitution, partielle ou totale d'une oeuvre[5]. Ceci est possible à partir

- soit de formalisations explicites laissées par le compositeur, c'est le cas de Xenakis pour des pièces telles que *Herma*, *Nomos Alpha* ou *Acchorripsis* ;
- soit de traces informatiques[6] laissées par les compositeurs qui ont utilisé OPENMUSIC (ou un autre logiciel) pour composer leur oeuvre, comme dans le cas de Magnus Lindberg ou Gérard Grisey qui ont utilisé PATCHWORK.

1.2.3 Le stage

Dans ce contexte, mon stage au sein de l'équipe Représentation Musicale de l'IRCAM porte sur deux axes :

- la réalisation d'un état de l'art des outils informatiques d'analyse, qui sera présenté et servira au sein du groupe « outils pour l'analyse ».
- un travail de réflexion et de développement sur de nouveaux outils d'analyse.

[4]Une libraire implémentant les concepts de la set-theory, une theorie mathématique de la musique, développée par Allen Forte en 1973. Elle est basée sur la théorie des groupes, et cherche à regrouper les accords par classes, à l'aide de relations d'equivalence. La set-theory s'attache particulièrement à l'analyse harmonique. D'autre part, la librairie Esquisse qui fournit une fonction de calcul de fondamentale virtuelle, et des fonctions d'analyse spectrale permet aussi certaines analyses.

[5]On entends par reconstitution d'une oeuvre la proposition d'un processus formel, ou d'un programme informatique, capable de reconstruire la partition à partir de paramètres donnés. Les pionniers de ce type d'approches sont Marcel Mesnage et André Riotte[16, 15].

[6]Il peut s'agir de programmes entiers ou simplement de fonctions. Ces traces ne sont pas toujours explicites et sont souvent difficiles à comprendre, comme un programme non documenté est difficile à maintenir.

Chapitre 2

Etat de l'art des outils informatiques d'analyse musicale

2.1 En quoi consiste l'analyse musicale ?

2.1.1 Définition

L'analyse musicale n'est pas une discipline récente. Pour autant que l'on sache, elle est apparue vers 1750, mais ne s'est établie comme discipline autonome qu'à partir du XIXe siècle. Selon Ian Bent[3] :

> « *L'analyse musicale est la résolution d'une structure musicale en éléments constitutifs relativement plus simples, et la recherche des fonctions de ces éléments à l'intérieur de cette structure.* »

Plus généralement, l'analyse musicale est une discipline qui consiste à étudier une oeuvre musicale et produire un résultat qui peut aller de la note dans un programme de concert à une simulation informatique[1] en passant par une description de la structure[2].

Nous ne chercherons pas à entrer dans le débat de la place de l'analyse musicale dans la science. Nous considérerons l'analyse musicale comme une démarche, qui peut être celle d'une tentative de recherche de structures et qui nous amène à réfléchir autour de la musique et à l'envisager sous des points de vue différents.

L'analyse musicale peut avoir de multiples facettes. Il peut s'agir de l'étude d'une oeuvre particulière, de l'ensemble des oeuvres d'un compositeur, ou d'un corpus musical entier. Nous pouvons être amené à vouloir replacer une oeuvre

[1] G.Assayag, M.Andreatta, C.Agon, M.Malt, M.Solomos ont reconstruits des oeuvres de Xenakis dans OPENMUSIC.

[2] Lerdahl et Jackendorf[12] analysent la musique selon un modèle de grammaire formelle et extraient des arbres de dérivations qui représentent la structure de l'oeuvre.

singulière par rapport à son corpus, comparer des styles, etc. L'analyse musicale est également présente sous d'autres formes, dans des système de recherche de musique[3], dans des système d'alignement automatique partition/signal, etc.

Enfin, une analyse musicale est une démarche ou l'on part d'un support, on se base sur des hypothèses, que l'on utilise pour faire l'analyse proprement dite, et qui produit un résultat. Au cours du temps, nous avons vu apparaître diverses approches, toutes avec leurs hypothèses et leurs résultats. Récemment, le problème du support a pris une dimension particulière.

2.1.2 Les supports de l'analyse

Représentation sous-symbolique : le signal

La musique a ceci de particulier qu'il ne s'agit pas d'un objet tangible, saisissable. La musique est un phénomène que l'on écoute, l'analyse la plus légitime serait donc basée sur le signal.

Il y a encore 10 ans, il n'était pas envisageable de vouloir traiter, avec l'ordinateur, le signal sonore. Mais cela devient de plus en plus accessible. La puissance de calcul disponible aujourd'hui, nous permet d'analyser directement le signal sonore. Les possibilités sont certes modestes, mais réelles[4], et ces possibilités ne vont que s'accroître avec la puissance de calcul.

Le son étant le seul dénominateur commum de toutes les musiques, le choix du signal sonore comme support de l'analyse garantit que la méthode utilisée pourra être appliquée à toutes les musiques. Bien sûr, cela ne veut pas dire que toutes les méthodes d'analyse du son donnent des résultats satisfaisants sur toutes les musiques.

Outre l'universalité, le support sonore est celui qui permet une analyse avec le moins de connaissances. Ce qui permet à l'analyste de partir de la base, et de choisir lui-même les hypothèses qu'il souhaite faire[5].

Représentation symbolique : la partition

La partition a été le support principal de l'analyse. Elle a l'avantage de présenter une vue globale de l'oeuvre, à un niveau neutre[6]. Elle se prête facilement à l'analyse de la structure. Mais les changements qui s'opèrent dans la musique posent de nouveaux problèmes.

[3]Le domaine du *Musical Information Retrieval* suscite un intérêt grandissant, compte tenu de l'évolution spéctaculaire de l'échange de musique sous forme numérique.

[4]Voir les outils développé par Marc Leman pour une approche cognitive qui traite directement le signal[11]. Voir également, l'acousmographe, developpé à l'INA-GRM.

[5]L'utilisation de la partition entraîne des connaissances implicites, comme le mode d'organisation des évènements, dont l'analyste ne pourra pas faire abstraction

[6]voire la tripartition proposée par Molino : Poïétique/Neutre/Esthésique

La partition, même si elle est présente dans toute la musique occidentale, n'est pas universelle. Certaines musiques ethniques n'utilise parfois d'autres systèmes de représentation, voire pas de représentation du tout. Dans ce dernier cas, la perennité est assurée par l'apprentissage de générations en générations.

Outre le fait que la partition occidentale n'est pas la seule représentation possible, la musique contemporaine à imposé des changements radicaux du concept de partition. Notamment, la musique contemporaine s'émancipe vis à vis de la partition, en instaurant de nouvelles notations (clusters, glissandos plus ou moins précis, gestes musicaux particuliers, étouffements).

Mais le développement de la musique électroacoustique pose encore plus clairement le problème de la représentation. La partition est totalement impuissante face à cette musique qui n'a pas de représentation neutre. D'autres genres de problèmes peuvent apparaître, comme celui posé par la musique spectrale[7]. Certaines oeuvres de musique spectrale sont écrites sur des partitions. Le problème est que ces partitions ne rendent pas du tout compte de la structure que le compositeur a voulu exploiter dans sa musique[8]. L'utilisation de l'informatique induit aussi l'utilisation de nouvelles formes de partitions[1].

Notons que, même si une représentation symbolique implique un parti-pris, ne serait-ce que dans le choix et l'organisation des symboles, elle a l'énorme avantage de structurer les données d'entrée et de réduire considérablement la complexité.

2.1.3 Hypothèses et connaissances a priori

Chaque analyse comporte ses hypothèses, et le choix du support de l'analyse implique des hypothèses, et des connaissances à priori. Par exemple, les méthodes d'analyse se basant sur la partition sont impuissantes si la musique à analyser ne peut pas se transcrire sous forme de partition.

De plus, les méthodes d'analyses n'utilisent pas toute les mêmes connaissances a priori. Par exemple, une analyse Schenkerienne implique la connaissance des notions de tonalité, degré, alors qu'une analyse à la manière de Forte[8] ne nécessite aucune notion de tonalité ou autre, mais la connaissance de la PCSet-Theory.

Le choix du niveau de connaissance n'est pas anodin. On peut affirmer qu'il va diriger l'analyse dans un axe particulier. Choisir d'utiliser beaucoup de connaissances peut permettre d'aller plus loin dans l'oeuvre, mais ce sera une analyse pour un public averti, et restreint. De plus, la plupart des connaissances que l'on

[7]Notre écoute étant essentiellement basée sur une analyse fréquentielle, la musique spectrale, représentée par Gérard Grisey ou Hugues Dufourt tente de manipuler les spectres des instruments traditionnels et/ou électroniques afin de produire des combinaisons intéressantes.

[8]Comme les spectres ne sont pas du tout présents dans la partition, cette dernière ne met pas en évidence le travail particulier de l'auteur.

peut avoir sur la musique font référence à un style de musique particulier, comme la musique tonale, ou sérielle. Ainsi, si une analyse utilise des connaissances qui relèvent de la musique tonale, cette méthodologie ne sera pas applicable sur une autre musique.

Au contraire, on peut choisir d'utiliser peu de connaissances. C'est le cas de l'approche paradigmatique qui cherche à structure la musique sans connaissances a priori. Il s'agit d'une approche issue de la linguistique.

La segmentation

Parmi les connaissances que l'on peut utiliser pour analyser une oeuvre, il y a la segmentation. En effet, beaucoup d'analyses[9] utilisent une segmentation a priori de l'oeuvre. Or cette segmentation va entrainer des groupements particuliers, et le résultat de l'analyse peut varier d'une segmentation à l'autre[10]

La segmentation, faites a priori par l'analyste, est en général le résultat d'un raisonnement non formalisé, parfois irrationnel, et qui utilise souvent une culture musicale spécifique et des notions aussi subjectives que la phrase musicale.

Ce problème de la segmentation devient crucial lorsqu'on tente de faire une analyse avec l'ordinateur. L'ordinateur doit-il segmenter lui-même selon des critères objectifs ? Doit-on lui fournir la segmentation ?

2.1.4 La forme du résultat

Toute analyse produit un résultat, qui est différent selon les méthodes. Il se peut que la forme du résultat ait été normalisée, mais c'est rarement le cas. On peut par exemple obtenir des graphes illustrant la structure schenkerienne[11], ou des arbres décrivant la dérivation à partir d'une grammaire[12]. Plus souvent le résultat de l'analyse se présente sous forme d'un texte exposant les structures ou propriétés remarquables de l'oeuvre étudiée.

Aussi, la forme du résultat revêt une importante toute particulière car elle est le support de l'interprétation du musicologue (ou de toute autre personne accédant à ce résultat). Le choix d'une forme pour le résultat d'une analyse n'est donc pas du tout anodin, mais conditionne le résultat de l'analyse. Ainsi, une forme ou structure cible qui permet la prise en compte d'un aspect sémantique est fortement recommandée. Il est donc particulièrement important d'étudier les diverses formes ou structures cibles possibles, comme les graphes conceptuels, les hypergraphes, les dendogrammes, ou autres.

[9] Analyse schenkerienne, Set Theory

[10] C'est un des point que l'on reproche souvent à Forte dans ses analyses.

[11] Voir quelques exemples dans *l'analyse musicale* de Ian Bent[3].

[12] On fait référence ici aux travaux de Lerdahl et Jackendorf[12]

2.2 Les outils existants

2.2.1 Humdrum

Présentation

HUMDRUM est un logiciel d'aide à la recherche en musique, crée par David Huron en 1994. Les capacités de HUMDRUM sont assez abstraites, il est donc difficile de caractériser ce qu'il fait précisement. HUMDRUM peut manipuler un nombre de représentations illimité, et peut transformer, classer, rechercher, restructurer, comparer.

HUMDRUM est un outil destiné à la musicologie systématique, il permet de tester et vérifier des hypothèses formalisables à propos de musique (ou tout type de chose qui peut être représenté dans le format HUMDRUM).

Architecture

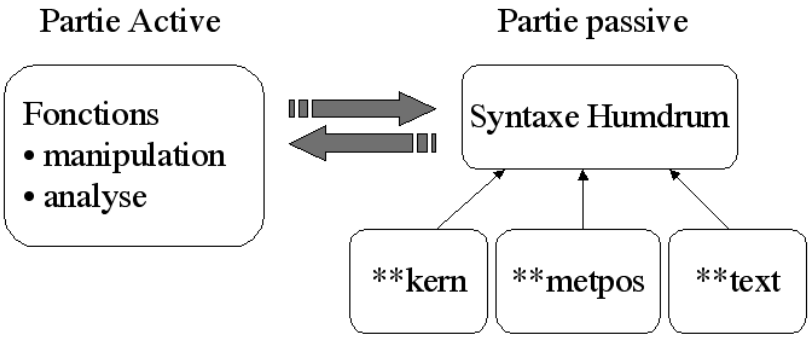


FIG. 2.1 – Architecture de HUMDRUM

HUMDRUM est composé de deux parties distinctes (cf fig 2.1) :

la partie passive : la spécification de la syntaxe des données. Il s'agit d'une syntaxe propre à HUMDRUM qui permet de structurer les données de façon à pouvoir y appliquer les fonctions. Une description de cette syntaxe est faite en annexe A.2.

la partie active : la librairie de fonctions. Il s'agit de fonctions permettant d'agir sur les données conformes à la syntaxe précédente. Ces fonctions permettent de manipuler, changer de représentation, traiter les données. Une description des fonctions disponibles se trouve en annexe A.3.

C'est dans cette distinction que HUMDRUM tire sa puissance. En effet, la plupart des fonctions sont tout à fait génériques (en particulier toutes les fonctions

de manipulation, de recherche de motifs et similarité), et s'appliquent à n'importe quelle représentation conforme à la syntaxe spécifiée. De plus, beaucoup de fonctions retournent un résultat qui est lui-même conforme à la syntaxe (fonction de calcul de métrique, analyse de *pitch class set*, etc.), ce qui ouvre un vaste champ de possibilités en terme d'interconnection des fonctions.

Ergonomie

L'interface de HUMDRUM est rudimentaire. En effet, aucune interface graphique n'est disponible, tout se passe dans un terminal Unix, donc en mode texte. L'installation n'est pas aisée, il y est d'ailleurs spécifié dans le manuel que toute personne souhaitant utiliser HUMDRUM est invité à prendre contact avec un utilisateur chevronné (ou un des auteurs).

Ainsi il s'est formé une communautée d'utilisateurs d'HUMDRUM autour des programmeurs. Au sein de cette communauté, des musicologues ont utilisé ce logiciel dans des analyses, publiées par la suite.

Fonctionnalités

Les fonctions[13] peuvent être des fonctions de manipulation des données, de *pattern matching*, d'analyse tonale, sérielle, ou autres. On remarque que ces fonctionnalités sont toutes de type traitement direct, aucune méthode heuristique, inductive ou par apprentissage. La fonction la plus souple, est celle de détection de motifs qui permet d'utiliser des expressions régulières. Peut-être pouvons-nous penser que la simplicité des fonctions compense en partie la déficience de l'interface[14].

2.2.2 Musicoscope

Présentation

Le MUSICOSCOPE est un logiciel d'analyse musicale écrit en Macintosh Common Lisp (ver 4.2 et au-delà) par Marcel Mesnage. Il résulte de l'intégration des anciens programmes MUSINOTE et MORPHOSCOPE. MUSINOTE permet à l'utilisateur de créer et modifier une partition via une interface graphique intuitive. Ensuite, MORPHOSCOPE peut récupérer le fichier au format de MUSINOTE et l'utilise pour construire sa structure de données interne (muage) qui peut être interrogée par l'utilisateur. Lors d'une analyse, l'utilisateur peut utiliser les paramètres préprogrammés dans le MUSICOSCOPE, ou définir les siens.

[13] cf. annexe A.3

[14] En effet, les fonctionnalités type IA nécessite un *feed-back* important au niveau de l'interface, afin de faciliter la compréhension par l'utilisateur. En revanche, des fonctions simples sont comprises plus rapidement et facilement par l'utilisateur.

La structure de données : muage

Muage est un néologisme provenant de musique et nuage. Il s'agit simplement d'une liste de n-tuples comprise comme une relation n-aire au sens des banques de données relationnelles. C'est une représentation évènementielle, ou chaque évènement est représenté par un n-tuple constitué de la valeur prise par les n composantes primaires.

Les composantes primaires sont celles qui sont extraites directement de la partition, comme la date, la voix, la hauteur, la durée, l'intensité, etc. Ensuite, une série de composantes dérivées peuvent être calculées à partir des composantes primaires, comme la densité, un accord, l'écart à la basse, etc.

Les composantes sont les paramètres que l'utilisateur peut choisir d'utiliser pour ses analyses. Par ailleurs, il est également possible de définir soi-même de nouvelles composantes[15].

Ergonomie

Le MUSICOSCOPE comporte une interface graphique qui permet à l'utilisateur de visualiser plus facilement la partition, de choisir les fonctionnalités qu'il souhaite utiliser pour son analyse. Les résultats des analyses sont sous forme de graphes en mode texte, qui offrent une meilleure lisibilité qu'un résultat numérique.

Fonctionnalités

Le MUSICOSCOPE est construit autour des notions clés de l'analyse :

segmentation C'est à dire le découpage de l'oeuvre en segments, souvent faite a priori par le musicologue.

composantes C'est à dire les paramètres sur lesquels vont porter l'analyse.

graphes C'est le résultat de l'analyse.

Une fois la segmentation faite[16], l'utilisateur peut choisir les composantes qu'il souhaite utiliser, et ensuite choisir un type de graphe. Il existe de nombreux graphes différents[17], présentant tous des particularités, et mettant en valeur certaines propriétés. Ces graphes peuvent être utilisés avec n'importe quelle composante, ou ensemble de composantes[18].

[15] Par exemple, l'ambitus d'un accord, ou toute composante relevant d'une classification arbitraire, comme la séparation des hauteurs en registres. Il est toutefois à noter que certaines composantes ne sont pas aisées à définir, il est parfois nécessaire d'avoir quelques connaissances en programmation LISP.

[16] Dans le MUSICOSCOPE, la segmentation peut se faire de manière automatique, en précisant un critère, ou de manière manuelle, en la construisant explicitement à la souris. voir annexe B.2.1

[17] voir la description des graphes, annexe B.3

[18] voir la liste des composantes, annexe B.2.2

Les graphes, utilisables d'une manière tout à fait flexible, confèrent au MUSICOSCOPE un statut de logiciel de visualisation. En effet, il permet à l'utilisateur de visualiser la musique sous de multiples angles, dans le but de faire émerger des propriétés non visibles directement dans la partition.

Pour ce qui est des fonctions du MUSICOSCOPE, elles sont principalement de type traitement direct, comme HUMDRUM. On remarque la présence d'une fonctionnalité d'extraction de règles, qui reste rudimentaire puisqu'elle ne permet l'extraction que des règles exactes, ce qui fait qu'on peut assez facilement se retrouver avec autant de règles que de cas.

2.2.3 Cypher

Présentation

CYPHER est un logiciel développé par Robert Rowe. Il s'agit un logiciel interactif dédié à la composition et l'exécution. Le principe est de créer un logiciel qui soit capable d'interagir avec un musicien, via MIDI, dans le but d'improviser ou composer de la musique. Son architecture s'inspire des travaux de M. Minsky, décrits dans *The Society of Mind* (Minsky 1986). Ainsi ce programme va être composé de plusieurs agents, connectés entre eux.

CYPHER comprend deux composants principaux :

le *listener* se charge d'écouter les événements et de les analyser, afin de transmettre au *player* les directives à suivre.

le *player* s'occupe de communiquer avec les autres modules MIDI.

Dans notre contexte, nous allons nous intéresser au *listener* qui seul comporte quelques fonctionnalités d'analyse.

L'analyse harmonique dans CYPHER

L'analyse harmonique dans CYPHER est faite à base d'agents. Chaque agent offre une expertise particulière (registre, dynamique, densite, vitesse, etc.), la combinaison de ces expertises conduit à la recherche de la tonalité[19].

L'analyse dans CYPHER a un statut tout à fait différent, puisqu'elle n'est pas finalisée par la restitution d'un résultat pour le musicologue, mais par le choix d'une tonalité qui va venir paramétrer le jeu. Ainsi l'analyse n'a, ici, pas de soucis d'explicitation, mais seulement celui de décision.

[19] Pour plus de précision, voir annexe C.2.

2.2.4 Autres outils

Dans le cadre du groupe de réflexion sur les outils d'analyse à l'IRCAM , d'autres outils ont été présentés.

Un outils d'analyse du signal : AUDIOSCULPT

L'équipe Analyse/Synthèse de l'IRCAM travaille sur des logiciels d'analyse du signal, comme AUDIOSCULPT. Ce logiciel permet de litéralement sculpter un son, et propose à l'utilisateur toute une panoplie d'analyses comme la FFT, fréquence fondamentale, hauteur virtuelle, formants, et autres.

AUDIOSCULPT permet de sélectionner des descripteurs parmi ceux proposés et de les visualiser à côté du signal ou du spectre de ce signal.

Kanthume, analyse par induction

Dans le cadre de sa thèse, Olivier Lartillot se concentre sur un système d'analyse suivant un modèle cognitif d'induction[10]. Il s'agit d'un logiciel d'analyse motivique, telle que Reti l'avait proposée, développé dans OPENMUSIC .

A partir d'une partition, donc une représentation symbolique, le logiciel devrait, en l'examinant dans l'ordre chronologique, découvrir des motifs musicaux et tisser des liens entre ces motifs. Cette analyse dans l'ordre chronologique modéliserait une écoute. La combinatoire est telle que sans limitations, la découverte de motifs musicaux exploserait. C'est pourquoi il est intéressant d'appliquer des contraintes cognitives, telles que la taille de la mémoire à court terme, la sélection des motifs, et un mécanisme d'induction permettant de reconnaître et de faire des liens avec les motifs entendus.

La place de l'induction On peut se demander quelle est la place de l'induction dans un logiciel d'analyse musicale ? En effet, un logiciel inductif entraine un rapport différent avec l'utilisateur, car ce dernier, pour interpréter correctement les résultats, doit avoir conscience que la machine a elle-même pris des décisions de sélection et d'induction sur les connaissances.

Dans ce cas, le rôle de la machine est beaucoup plus prospectif, au fur et à mesure du déroulement, la machine va induire des connaissances (ici des motifs) afin de construire une analyse de la pièce. Le résultat sera donc différent de celui d'un outil habituel car l'utilisateur, à moins d'être le programmeur, n'aura généralement pas une idée précise du résultat qu'il va obtenir. C'est pourquoi il est nécessaire de bien informer l'utilisateur sur les mécanismes mis en oeuvre.

2.3 Synthèse

Les remarques à propos des logiciels d'analyse musicale peuvent se faire à plusieurs niveaux. La première remarque, est qu'on en est qu'aux balbutiemment de tels logiciels, chacun produit son logiciel dans son coin pour ses propres fins. Les utilisateurs sont peu nombreux. Afin de tenter de clarifier, nous proposerons les remarques suivantes et quelques conséquences. Nous rappelons que nous ne parlons que des outils informatiques d'analyse, et pas de l'analyse musicale elle-même.

2.3.1 Usages et explicitation des connaissances

Analyse appliquée

Les logiciels autour de la musique peuvent être amené à mettre en oeuvre des fonctionnalités d'analyse musicale. C'est le cas de CYPHER, mais c'est également le cas de logiciels tels que les séquenceurs, les outils de gestion de plate-forme de partition (comme WEDELMUSIC)[20].

Ces logiciels ont ceci de particulier que l'analyse musicale intervient en tant qu'élément de décision et pas en tant que résultat à expliciter à l'utilisateur. Dans ce cas, on pourra (même si ce n'est ni une obligation, ni une recommandation) utiliser des modèles type « boite noire »comme des réseaux neuronaux. La seule contrainte est la qualité du résultat, et pas son explicitation. Ces logiciels, qui sont amenés à utiliser des fonctions d'analyse musicale, ont un large public, qui peut aller du musicien averti au simple mélomane.

Imagerie musicale

On entendra par imagerie musicale des logiciels qui offre la possibilité à l'utilisateur de manipuler et visualiser la musique selon des points de vues différents. C'est le cas du MUSICOSCOPE, qui n'est qu'un embryon, et en quelque sorte aussi celui de HUMDRUM, excepté que ce dernier n'est pas doté de visualisation graphique.

Lors de discussions au sein du groupe de réflexion « outils pour l'analyse », il est ressorti que le changement de représentation, et la visualisation de la musique est une demande forte des musicologues. M.Malt affirme que cela constitue « une puissante aide à la pensée ». En effet, le changement de représentation, et l'appropriation par l'utilisateur de ce nouvel espace de réflexion lui permet de dégager de nouvelles propriétés.

[20] Les outils cités ne font que du traitement symbolique. Nous pourrions considèrer la panoplie d'outils d'analyse orientés signal.

Contrairement à l'analyse appliquée, les modèles type « boite noire »sont à proscrire, car c'est justement l'explicitation de propriétés cachées qui intéresse les musicologues. On privilégiera donc des représentation qui permettent à l'utilisateur d'interpréter[21]. Ces outils sont plutôt des outils d'Analyse Assisté par Ordinateur (AAO) destinés aux musicologues, mais pourquoi pas aussi aux compositeurs.

2.3.2 Interactivité et prospection

Dans le cadre des outils d'AAO, la notion de prospection est fondamentale. Le musicologue n'utilise un logiciel d'AAO qu'à titre prospectif, c'est à dire pour réaliser des calculs, des changements de représentation qu'il ne pourrait pas raisonnablement faire à la main. Or la prospection, implique toujours un comportement du type essai-erreur. Il y a donc potentiellement une forte interactivité entre l'utilisateur et l'ordinateur.

Dans les logiciels présentés jusqu'ici, la seule prise en compte de l'interactivité se situe au niveau de la flexibilité. C'est à dire qu'il est considéré que pour faire des outils prospectifs, il suffit de proposer un maximum de fonctionnalités. Il est clair que c'est fondamental, mais il existe aussi d'autres possibilités.

Ainsi, pourquoi ne pas faire intervenir l'utilisateur pour des fonctions qui posent des problèmes. Par exemple, lors de la détection de motifs similaires, si rigide aujourd'hui, pourquoi ne pas mettre en place des fonctions d'apprentissage supervisé. L'ordinateur pourrait proposer des catégories que l'utilisateur corrige, et que l'ordinateur réintègre pour corriger sa méthode de sélection. Il existe des méthodes IA permettraient ceci (voir ce qui suit).

2.3.3 Fonctionnalités

Comme il a été dit plus haut, la plupart des logiciels d'AAO ne proposent que des fonctions de calcul brut. On ne voit pas émerger de fonctionnalité type IA c'est à dire des fonctionnalités d'apprentissage, de classification. Essayons d'imaginer quelques applications ou ces fonctionnalités seraient déterminantes.

Apprentissage et induction

Comme évoqué ci-dessus, on pourrait mettre à profit des fonctions d'apprentissage pour la détection de similarité. L'apprentissage couplé à l'interactivité peut permettre d'améliorer la reconnaissance de motifs similaires.

L'apprentissage a été utilisé brillamment par David Cope[6, 7] pour la conception du logiciel EMI. Ce logiciel se propose d'apprendre le style d'un compositeur,

[21] Voir des tentatives dans ce sens, partie 3.3.

en vue de composer une oeuvre dans le même style.

interaction homme-machine et epistémologie La mise en place de fonctionnalités d'apprentissage et d'induction peut poser problème dans le cas d'outils d'AAO. En effet, le musicologue, comme tout utilisateur, va s'approprier l'outil, se construire sa propre représentation mentale du fonctionnement de la machine, et va tirer ses conclusions.

Le problème est que si les mécanismes d'apprentissage et d'induction ne permettent pas d'expliciter la préférence d'une solution par rapport à une autre. Cette préférence résulte du processus d'induction et des données auquelles le système a été confronté auparavent. Ainsi, le résultat peut varier, par exemple en préférant certains motifs par rapport à d'autres. Dans ce cadre, l'interprétation du résultat obtenu est difficile. On obtiendra une analyse selon certaines connaissances apprises par le système, et pas une analyse systématique ni exhaustive qui permettrait une justification scientifique.

Cependant, il ne faut pas confondre les outils d'*Analyse Assistée par Ordinateur* avec des outils d'*analyse automatique*. Les outils d'AAO n'ont pour vocation que de servir le musicologue et pas de réaliser l'analyse à sa place. L'interprétation du résultat restera de toutes façons à la charge du musicologue. Il convient juste de connaître le système et ses limites.

Classification

Dans beaucoup d'analyses, une des premières tâches de l'analyste est d'expliciter les objets sur lesquels il travaille et de les caractériser. Il s'agit de définir les *unités*. Certains musicologues vont jusqu'à penser que la définition de ces unités est la part la plus important de l'analyse.

Ces objets peuvent être des accords, des rythmes, des segments musicaux. Pour le cas, peut-être particulier, des accords, l'analyste utilise toujours une classification des accords a priori. Cette classification peut-être celle issue de la théorie tonale (accord majeur, mineur, septieme, etc.), ou par exemple de la set-theory. Les ouvrages de théorie musicale propose aussi à un moment ou à un autre, une classification des objets musicaux[2, 20].

D'une manière générale, l'analyste cherche à réduire la diversité des objets musicaux[8], en les réunissant dans des catégories. Dans ce cadre, les méthodes de classification peuvent proposer des classes différentes, qui peuvent avoir leurs intérêts pour le musicologue.

2.3.4 La segmentation

La segmentation est une étape importante de l'analyse car elle influe considérablement sur le résultat. Elle est souvent choisie a priori par l'analyste. Les

outils d'analyse sont totalement impuissants face à au problème de segmentation automatique[14], car il s'agit souvent d'un processus subjectif et impliquant une culture musicale.

Dans le cadre de sa thèse, Benoit Meudic a développé des méthodes d'extraction de métrique[17], ou de rythme[18]. Les travaux E.Cambouropoulos vont également dans le sens d'une segmentation automatique[5].

La meilleure alternative pour le moment, est celle adoptée par le MUSICOSCOPE qui propose soit une segmentation automatique selon un test sur des composantes, soit de laisser l'utilisateur libre de définir lui-même sa segmentation.

2.3.5 Intégration sous-symbolique $\leftrightarrow$ symbolique

Nous pouvons remarquer que HUMDRUM, qui n'effectue que l'analyse symbolique, prévoit des champs de données, comme le spectre, qui résulte d'une analyse au niveau signal. L'intérêt de récupérer ces informations au niveau symbolique, est que l'information est réduite, et les méthodes d'analyse beaucoup plus efficaces.

Cependant, nous ne pouvons pas réduire le problème de l'intégration sous-symbolique/symbolique à une transposition sous-symbolique $\rightarrow$ symbolique. On peut extraire certaines informations du signal qui ne vont pas se transposer facilement au domaine symbolique. Par exemple, si on s'attache à un descripteur de type instantané, c'est à dire un descripteur qui évolue avec le signal, le passage de continu à discret pose des problèmes. On voit surgir ici le problème de la segmentation automatique.

2.3.6 Une organisation flexible

> « *Il me semble que c'est une tendance générale de la musique (aujoud'hui de l'électoacoustique à Lachenmann et Grisey) de composer des paramètres nouveaux et plus audacieux que rythme, durée, nuance, hauteur de la fondamentale* » Fabien Levy

La flexibilité d'un logiciel d'AAO est fondamentale. Afin que l'outil puisse jouer son rôle prospectif, il faut nécessairement qu'il offre à l'utilisateur divers outils et combinaisons. Si les compositeurs utilisent de nouveaux « paramètres », il faut que l'analyste puisse aussi les avoir à sa disposition pour analyser. HUMDRUM et MUSICOSCOPE se sont tous les deux dirigés dans cette voie.

Afin d'obtenir une bonne flexibilité, il faut réussir à bien séparer les composantes qu'on analyse, les fonctions d'analyse, et les outils de visualisation. Ceci dans le but de permettre un maximum de combinaisons possibles entre ces trois entités. Il est également intéressant d'avoir une architecture ouverte, permettant l'extensions des composantes et des fonctions, voire des outils de visualisation.

Pour permettre une telle séparation au sein du logiciel, il est nécessaire de spécifier des structures des données qui font le lien entre ces parties. C'est exactement ce que fait HUMDRUM.

2.4 Conclusions et perspectives

Les différents points abordés dans la partie précédente montrent quelques questions qu'il faut se poser lors de la réalisation d'un outil d'analyse. Dans le cas particuliers des outils d'analyse assistée par ordinateur, les pistes qui semblent s'imposer sont :

- bien définir la structure cible de l'analyse.
- prendre en compte l'interactivité homme-machine pour construire un système plus complet.
- intégrer les aspects symboliques et sous-symbolique.

Par la suite, le stage a été l'occasion de faire quelques expérimentations. Ainsi nous allons maintenant voir comment nous pouvons appliquer des méthodes de classification en analyse musicale, puis nous verrons comment mettre en oeuvre un outil d'imagerie musicale.

Chapitre 3

Réalisations

3.1 Contexte et support

3.1.1 OPENMUSIC

Présentation

OPENMUSIC est un logiciel de composition assistée par ordinateur, disponible sur Macintosh. Il a été développé au sein de l'IRCAM par Gérard Assayag et Carlos Agon. Logiciel écrit en CLOS[1], OPENMUSIC a bénéficié de l'expérience acquise avec les logiciels précédents, notamment PATCHWORK. Il propose des fonctionnalités radicalement nouvelles par rapport à ce qui se fait dans le domaine.

Un environnement visuel

Plus qu'un simple logiciel de CAO, OPENMUSIC offre une interface de programmation visuelle permettant la définition et la manipulation de fonctions LISP et d'objets CLOS. Ceci fait de lui une véritable surcouche du langage CLOS. Les classes et le méthodes sont représentées par des icônes qui peuvent être manipulées et connectées directement à la souris.

OPENMUSIC offre à son utilisateur une large panoplie d'objets et de fonctions destinés à la composition musicale. Il comporte notamment un noyau d'objets musicaux permettant de manipuler une représentation d'une partition. De ce fait, le programmeur a à sa disposition tous les composants nécessaires pour traiter de musique.

Toutes les réalisations de ce stage ont été faites dans OPENMUSIC , certaines ont été finalisées et ont été regroupés dans une librairie[2], qui sera à la disposition

[1] *Common Lisp Object System* est un langage orienté objets basé sur le langage LISP.

[2] Il s'agit des fonctions d'imagerie musicale, présentés dans la partie 3.3, et regroupées au sein de la librairie OMiel.

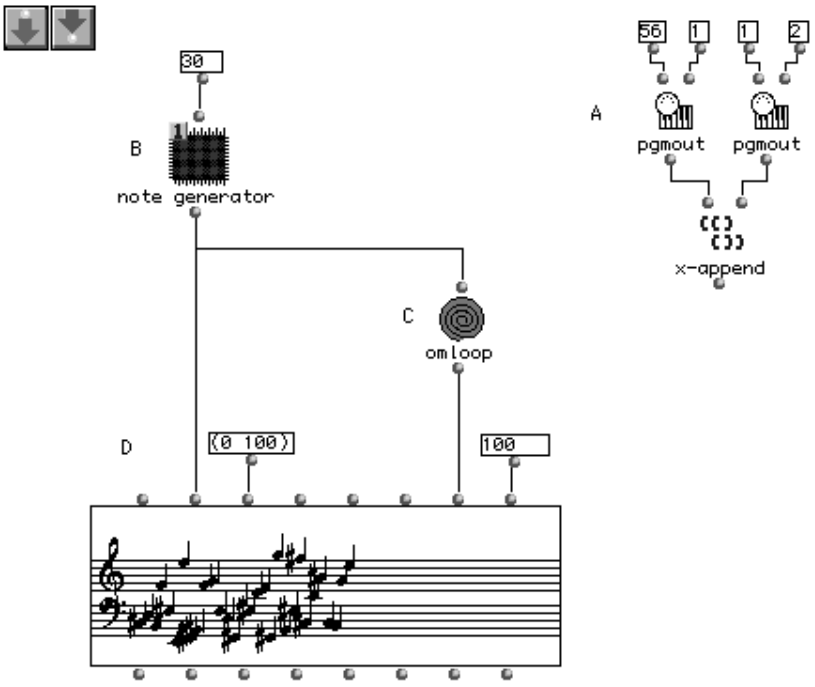


FIG. 3.1 – Manipulation et connection de fonctions LISP

des utilisateurs d'OPENMUSIC .

3.1.2 Hermeto Pascoal

Lors d'une des réunions du groupe de réflexion sur les outils d'analyse, un musicologue, Jean-Pierre Cholleton est venu présenter la musique d'Hermeto Pascoal. Ce compositeur brésilien produit une musique qui influence beaucoup de jeunes compositeurs au Brésil, et qui pose des problèmes à l'analyste.

Il s'agit d'une musique à caractère polytonal, ou l'harmonie, complexe, tient une place prédominante. Or, si certaines phrases musicales s'expliquent par des théories tonales courantes, d'autres ne s'y prêtent pas du tout.

C'est sur des exemples tirés d'Hermeto Pascoal que les outils présentés dans ce qui suit ont été élaborés et testés.

3.2 Classification d'accords

3.2.1 Objectif

Comme il a déjà été énoncé plus haut, une des premières étapes de l'analyse musicale est d'identifier les objets constituant l'oeuvre, et de les organiser en classes ou catégories. C'est dans cette optique que nous allons essayer de mettre en oeuvre des méthodes de classification.

Les harmonies développées par Hermeto Pascoal posent des problèmes à-l'analyste. Nous allons essayer dans un premier temps, même si cela est insuffisant, d'identifier les accords utilisés par Pascoal. L'objectif est de proposer des classifications de ces accords au musicologue. Diverses distances seront testées, avec les algorithmes de classification ascendante hiérarchique.

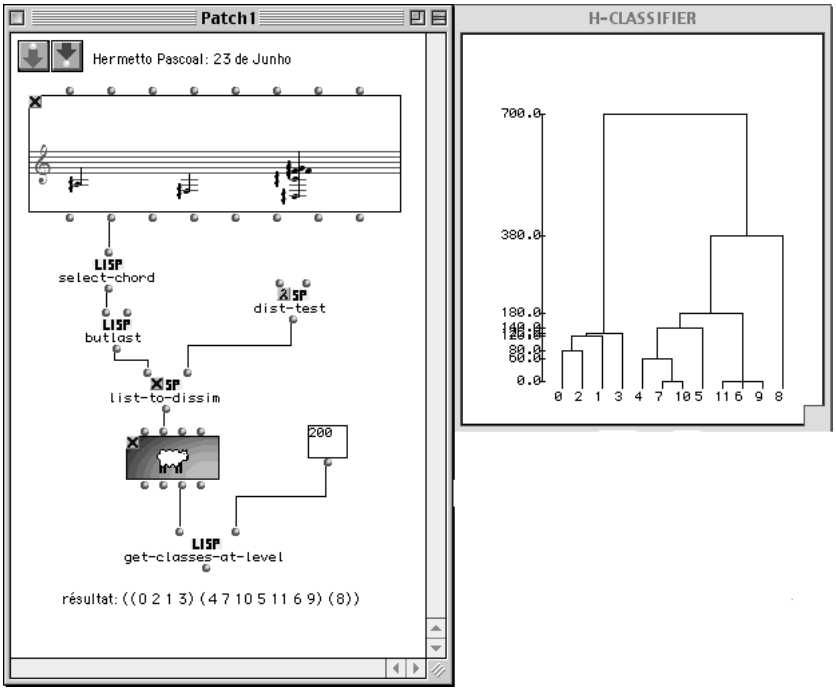


FIG. 3.2 – Classification dans OPENMUSIC

3.2.2 Des outils de classification pour OPENMUSIC

Afin de réaliser ces classifications, il a fallu créer les méthodes et classes nécessaires. Ainsi, les composants suivants ont été conçus :

– une méthode, *list-to-dissim*, qui, à partir d'une liste d'objets et d'une fonction de dissimilarité entre ces objets, construit la matrice de dissimilarité

des objets.
- une classe, *h-classifier*, qui, à partir d'une matrice de dissimilarité, applique l'algorithme de classification ascendante hiérarchique. L'utilisateur a la liberté de choisir la méthode de lien utilisée parmi le lien simple, le lien moyen et le lien complet. Cette classe est munie d'une représentation graphique qui permet de visualiser le résultat.
- une méthode, *get-classes-at-level* qui permet d'obtenir les classes qui correspondent à une coupe du dendogramme à un niveau donné.

Un exemple d'utilisation de ces composants est présenté à la figure 3.2.

3.2.3 Distance entre accords

Après avoir réalisé les outils précédents, il ne reste plus qu'à définir une distance entre accords pour réaliser la classification. C'est évidemment là que les problèmes les plus intéressants se posent.

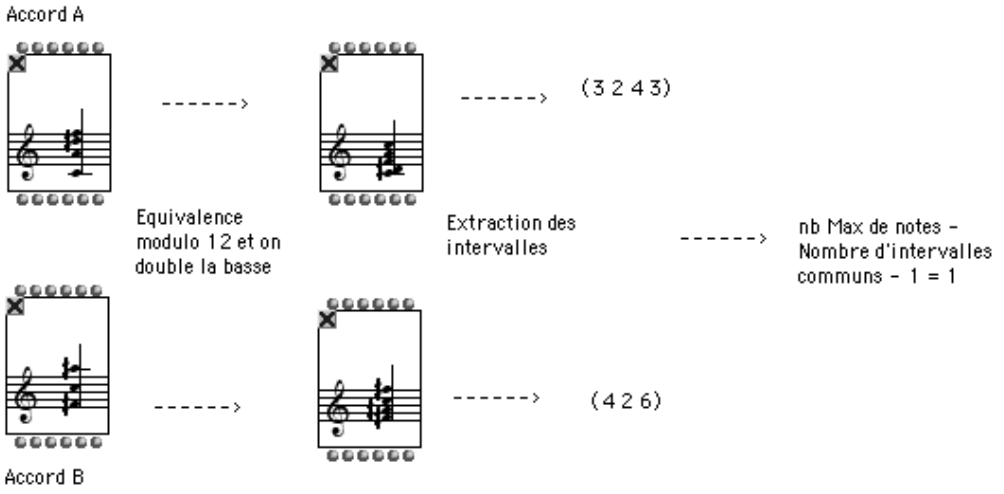


FIG. 3.3 – Calcul de la distance d'Estrada

Distance Estrada

La distance Estrada a été proposée par le musicologue Julio Estrada. Il s'agit d'une distance entre accords qui semble donner de bon résultats lors d'expérience de psycho-acoustique à propos de simulation de style. Cette distance consiste à ramener l'accord donné à l'accord le plus compact en considérant les notes distantes d'un octave comme équivalentes, et ensuite à soustraire au nombre maximal moins 1 de notes le nombre d'intervalles communs aux deux accords. (voir figure 3.3).

Pour deux accords identiques – ils auront donc la même réduction – la distance sera nulle. En revanche, deux accords qui ne sont pas identiques mais qui auraient la même réduction auront une distance nulle (il s'agit donc d'une pseudo-distance). Et, de manière générale, les accords seront d'autant plus distants qu'il n'auront pas les mêmes intervalles internes.

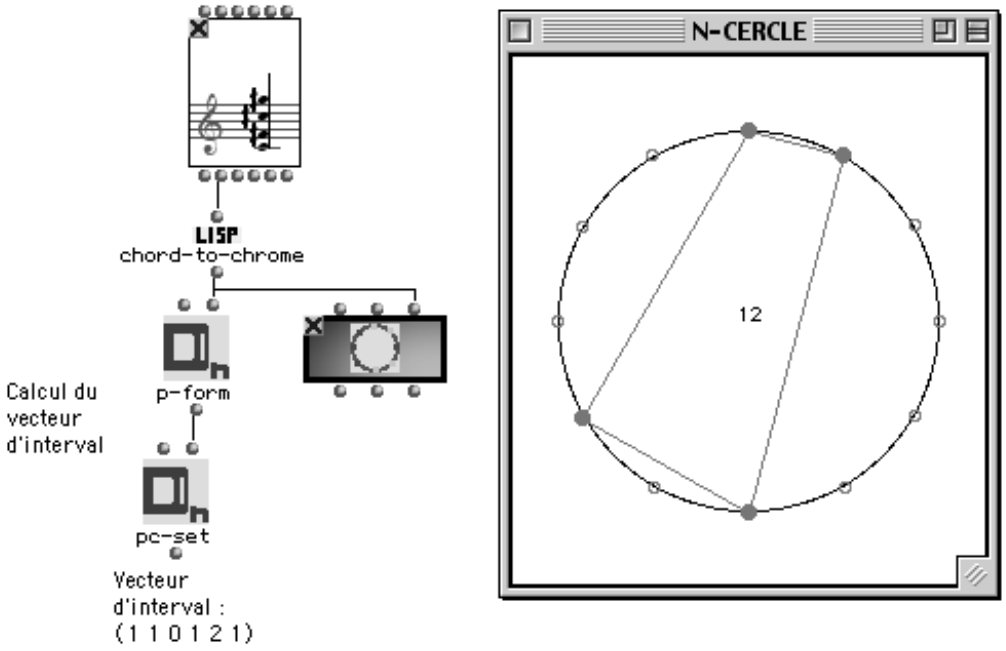


FIG. 3.4 – Le cercle chromatique et le vecteur d'intervalles

Distance selon le vecteur d'intervalles

Le vecteur d'intervalles est une notion développé dans la *PCSet Théory* par Allen Forte. Comme pour la distance d'Estrada, il faut réduire l'accord en considérant l'équivalence à l'octave, mais cette fois, on ne s'occupe pas de la basse. Une bonne représentation de l'objet manipulé est l'inscription dans le cercle chromatique (voir figure 3.4).

Enfin, le vecteur d'intervalles consiste à compter le nombre d'intervalles de longueur 1 à 6 sur le cercle. Ainsi, dans l'exemple de la figure 3.4, le vecteur d'intervalles est $< 110121 >$, ce qui signifie que l'accord réduit comporte 1 seconde mineure (intervalle de 1 demi-ton), 1 seconde majeure (2 demi-tons), 0 tierce mineure (3 demi-tons), 1 tierce majeure (4 demi-tons), 2 quartes (5 demi-tons) et 1 triton (6 demi-tons).

Distance entre deux vecteurs d'intervalles L'idée consiste à sommer la différence de nombre d'intervalles de chaque longueur. Ainsi, les vecteurs d'intervalles correspondant aux accords utilisés dans l'exemple de la figure 3.3 sont $< 012111 >$ et $< 010101 >$. La distance entre ces deux accords sera donc 3.

3.2.4 Résultats

Pour ce qui est des méthodes de liens, la méthode de lien simple est inefficace car les distances utilisées donnent beaucoup de 0. Ainsi, une méthode de lien simple donne une seule classe qui contient tous les éléments. De ce fait, la méthode du lien complet a été privilégiée.

Les données utilisées sont les accords de la partie Piano de deux pièces d'Hermeto Pascoal pour Piano et Flute : *23 de junho* et *musica das nuvens e do chao-red*. Les diverses transpositions d'un même accord ont été éliminées. Au final, il reste 51 accords, de 3 à 7 sons, tous distincts.

Les dendogrammes des classifications selon la distance Estrada et selon le vecteur d'intervalles sont présentés dans les figures 3.5 et 3.6.

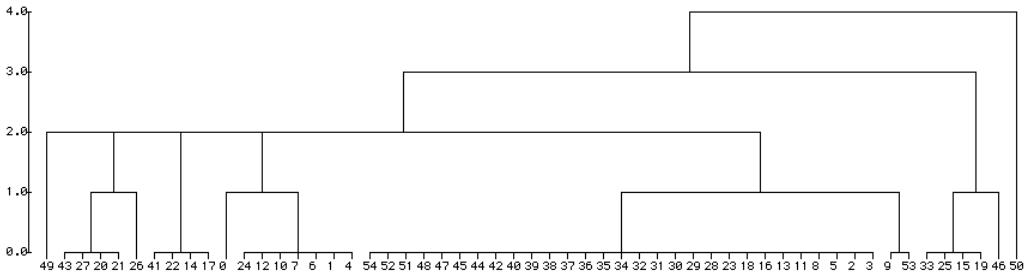


FIG. 3.5 – Classification selon la distance d'Estrada

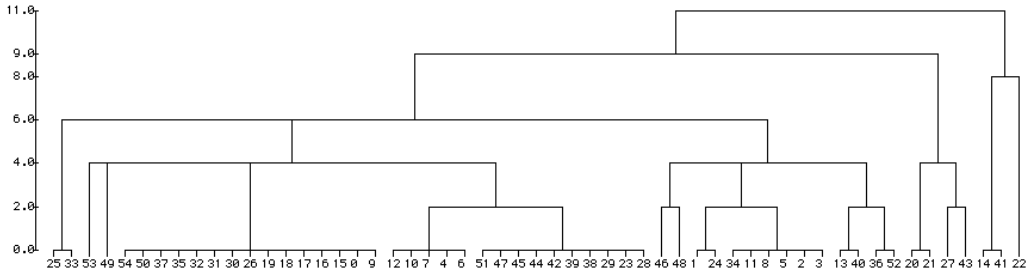


FIG. 3.6 – Classification selon le vecteur d'intervalle

3.2.5 Interprétation

Les deux distances sont basées sur la structure intervallaire de l'accord. Ainsi on trouve quelques similitudes entre les deux classifications. Notamment, si on coupe la classification selon Estrada au niveau 1.5 et celle selon le vecteur d'intervalles au niveau 5, on retrouve dans les deux cas les classes $\{43, 27, 20, 21, 26\}$ et $\{33, 25\}$.

Cependant des grosses différences apparaissent entre les deux classifications. Par exemple, les classes $\{41, 22, 14, 17\}$ et $\{24, 12, 10, 7, 6, 1, 4\}$ de la classification Estrada sont plutot dispersées dans la classification selon le vecteur d'intervalles.

Après vérification, on se rend compte que la distance selon le vecteur d'intervalles est plus dépendante du nombre de notes que celle d'Estrada. La distance selon le vecteur d'intervalles aura tendance à regrouper les accords de même nombre de notes.

Limitations Ce type de méthode pour classer les accords a l'inconvénient majeur de ne s'occuper que de l'aspect statique de l'accord, et pas de son rôle ou de sa fonction dans la démarche harmonique. Une analyse plus fonctionnelle serait plus satisfaisante.

Remarques Il est important de noter à propos de cette approche que cet outil permet au musicologue d'aborder la classification à un autre niveau. Au lieu de s'attaquer courageusement aux accords, un par un, et de mettre en place un processus de classification personnel, qui sera discutable, le musicologue peut travailler directement au niveau de la distance. Il pourra ainsi mettre en évidence d'autres propriétés.

On pourrait également envisager l'utilisation de ce genre de méthode pour proposer des modèles d'écoute. Simplement en comparant les classifications obtenues, avec les classifications proposées par des sujets.

3.3 Imagerie musicale

Comme cela a été décrit dans la partie état de l'art, l'imagerie musicale consiste à offrir au musicologue un moyen de visualiser la musique, sous des angles différents. Dans ce cadre, le musicologue Jean-Marc Chouvel a proposé une représentation sur un damier hexagonal qui a des propriétés intéressantes. Afin de la tester, cette représentation a été mise en oeuvre dans des outils de visualisation.

3.3.1 La représentation hexagonale

Il s'agit d'une représentation instantanée de la partition, donc on ne considère que les notes présentes à cet instant. La représentation pourra ensuite être animée de façon à décrire le flux musical.

On se base sur le système tempéré, les notes sont prises en compte indépendamment de l'octave (il en reste donc 12), et sont placées sur un damier hexagonal, d'une façon particulière, décrite à la figure 3.7. Cette figure montre les 12 sons répétés 4 fois. Ceci pour mettre en évidence qu'il s'agit d'un tore, c'est à dire que la représentation est cyclique verticalement et horizontalement.

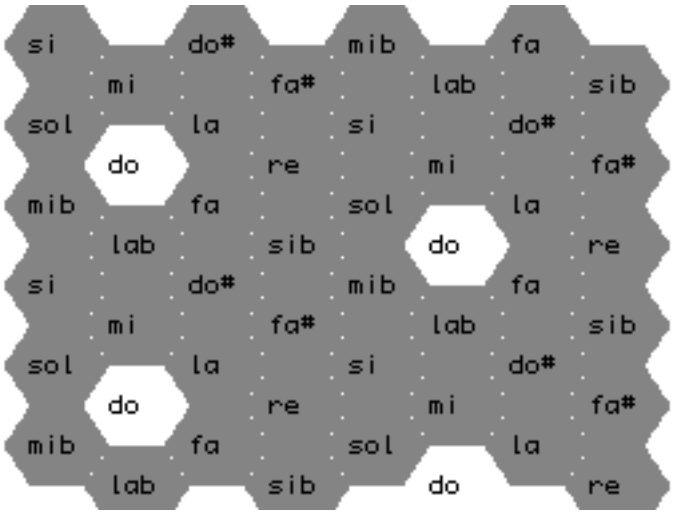


FIG. 3.7 – Représentation hexagonale.

Les propriétés de cette représentation sont les suivantes :

- Il s'agit d'un tore.
- L'axe ↑ voit s'empiler les tierces majeures. Cycle de longueur 3. L'axe vertical représente l'accord de quinte augmenté.
- L'axe ↗ constitue le cycle des tierce mineure, c'est l'accord de septième diminuée.
- L'axe ↖ représente le cycle des quintes, qui lui passe par toutes les notes, et s'enroule le long du tore.
- Les notes voisines sur la représentation entretiennent une relation harmonique forte (tierce ou quinte).
- Les notes proches chromatiquement (1 demi-ton et 1 ton) sont celles qui sont séparée par une arête. Par exemple : Do-Si, Do-Sib, Do-Re, Do-Do#.
- Par rapport à une note, la seule autre note qui n'est ni dans l'entourage direct, ni a distance d'une arête, est son triton, c'est à dire la note la plus éloignée harmoniquement.

– Globalement, la distance sur la représentation correspond à une distance harmonique.
– Si deux formes[3] sont identiques alors il s'agit du même accord ou d'un renversement.

Un des intérêt de cette représentation est qu'elle est très visuelle, ce qui nous permet de reconnaitre très facilement et rapidement les accords et sous-accords, via les formes qu'ils produisent sur la représentation.

L'exemple de la figure 3.8 montre l'accord parfait de Do majeur. Il s'agit d'un « triangle »tourné à gauche. Le symétrique, c'est à dire le triangle tourné à droite correspond à l'accord parfait mineur relatif à Do, c'est à dire La mineur.

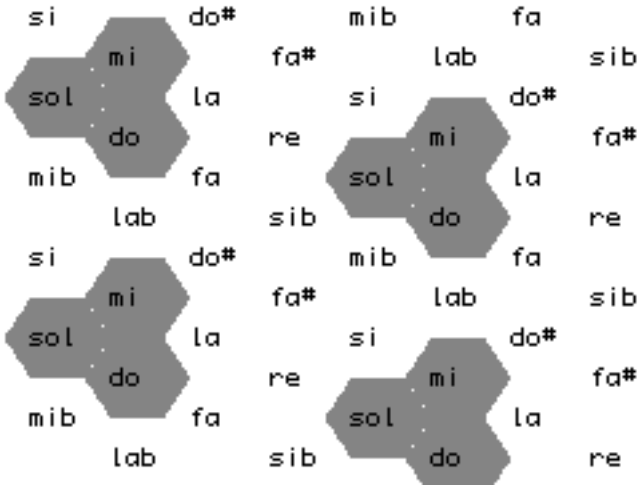


FIG. 3.8 – Accord parfait majeur sur le damier hexgonal.

3.3.2 Un outil de visualisation

Afin d'utiliser cette représentation, un outil de visualisation, le *chord-seq-hex*, a été développé. Il reprend l'interface d'un objet commun d'OPENMUSIC , le *Chord-seq*[4], et l'enrichit en offrant les fonctionnalités suivantes (voir figure 3.9) :

– Visualisation sur le damier hexagonal, avec un curseur indiquant sur la partition l'accord correspondant.
– Possibilité d'avance et reculer dans le morceau.
– Possibilité d'afficher en grisé l'accord précédant, afin de permettre un lecture facile des notes communes, et des déplacements de notes.
– Possibilité d'entourer la basse réelle de l'accord.

[3]On appellera *forme* un ensemble de notes connexes sur la représentation.

[4]Le *Chord-seq* est une séquence d'accord, les accords pouvant n'avoir qu'un seul son. Il s'agit de l'objet le plus simple pouvant contenir un morceau de musique.

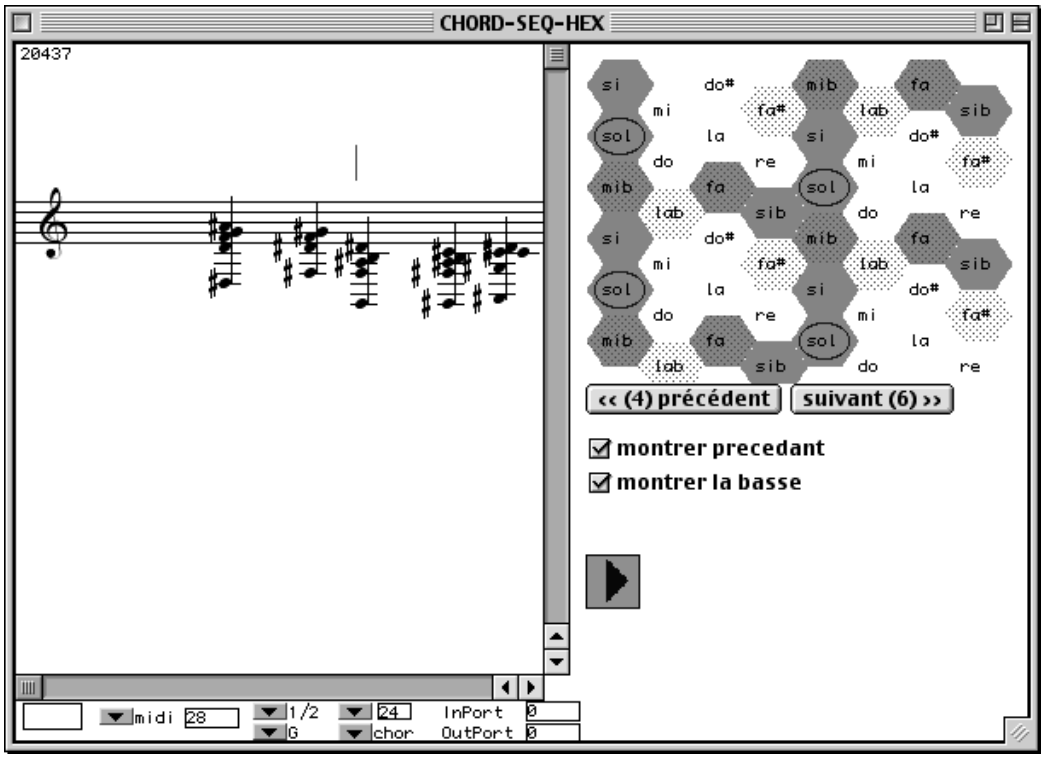


FIG. 3.9 – L'interface de l'objet *chord-seq-hex*

3.3.3 Un outil d'analyse de sous-motifs

Il est amusant de remarquer que les formes qui apparaissent sur le damier hexagonal éveillent notre curiosité. De cette constatation part l'idée d'analyser les accords en termes de sous-motifs. Nous appelons sans distinction motif et forme un ensemble de notes connexes sur la représentation, et nous considérerons comme égaux les motifs obtenus pas translation sur la représentation. Un sous-motif est un motif inclu dans un autre. Dans la suite, nous parlerons de motifs pour désigner la forme obtenue par représentation d'un accord sur le damier hexagonal, et de sous-motifs pour désigner une partie de ce motif.

L'idée consiste à extraire tous les sous-motifs, avec des contraintes de taille, d'une séquence d'accord, et de repérer quels sont les sous-motifs prépondérants, les sous-motifs rares, ceux qui sont discriminants, etc.

Extraction des sous-motifs

La première étape est d'extraire les sous-motifs présents dans un motif. Bien sûr, il existe un grand nombre de sous-motifs dans un motif, on permet donc à l'utilisateur de fixer une contrainte de taille. Par exemple il peut ne considérer que

les sous-motifs de 3 ou 4 sons.

Une fois cette contrainte fixée, on extrait tous les sous-motifs des accords considérés, on apparie les sous-motifs équivalents (ie à une translation près), et on compte le nombre d'occurrences de chaque sous-motif. On peut ensuite déterminer le taux d'apparition d'un sous-motif qui sera le rapport du nombre d'occurences du sous-motif au nombre d'occurences de sous-motifs de même taille.

Sélection de sous-motifs discriminants

La sélection de sous-motifs discriminants consiste à trouver un ensemble minimal de sous-motifs tel que pour tout couple d'accords différents sur le damier hexagonal, il existe un sous-motif de l'ensemble qui soit présent dans un des accords et pas dans l'autre. Cet ensemble peut nous apporter deux indications. Il nous indique des sous-motifs qui jouent un rôle particulier, mais, via la taille de l'ensemble, nous donne aussi un indice de la complexité des accords.

Ce problème de sélection des sous-motifs discriminants est un problème de recouvrement minimal, dont on sait qu'il est NP-complet, mais dont on connait des heuristiques qui donnent des résultats satisfaisants. Nous avons choisi d'utiliser une heuristique présentée dans le domaine de l'analyse logique de données[4]. Bien sûr, l'ensemble sélectionné n'est pas unique.

L'outil *analyse-hex-patterns*

Un deuxième outil a donc été réalisé, l'objet *analyse-hex-patterns*. Son interface est présentée à la figure 3.10. Il présente, en plus des celles du *chord-seq-hex*, les fonctionnalités suivantes :

- Extraction des sous motifs[5] dont la liste est présentée en bas à droite, avec pour chaque motif son taux d'apparition.
- Possibilité de « projeter »les accords sur des motifs sélectionnés. C'est à dire que pour chaque motif sélectionné on crée une nouvelle portée, et pour chaque accord, si un motif sélectionné apparaît au sein de l'accord, on place sur la portée le sous-accord qui correspond au motif.
- La possibilité de sélectionner un ensemble de motifs discriminants.

Résultats

Cette analyse de sous-motifs a permis de dégager deux remarques à propos de musiques d'Hermeto Pascoal.Tout d'abord, la sélection de sous-motifs discriminants sur la pièce *23 de junho* propose comme résultat 5 sous-motifs dont les ac-

[5]La taille des sous-motifs peut être fixée manuellement à l'initialisation de l'objet. Par défaut, l'objet extrait les motifs de taille 3 et 4.

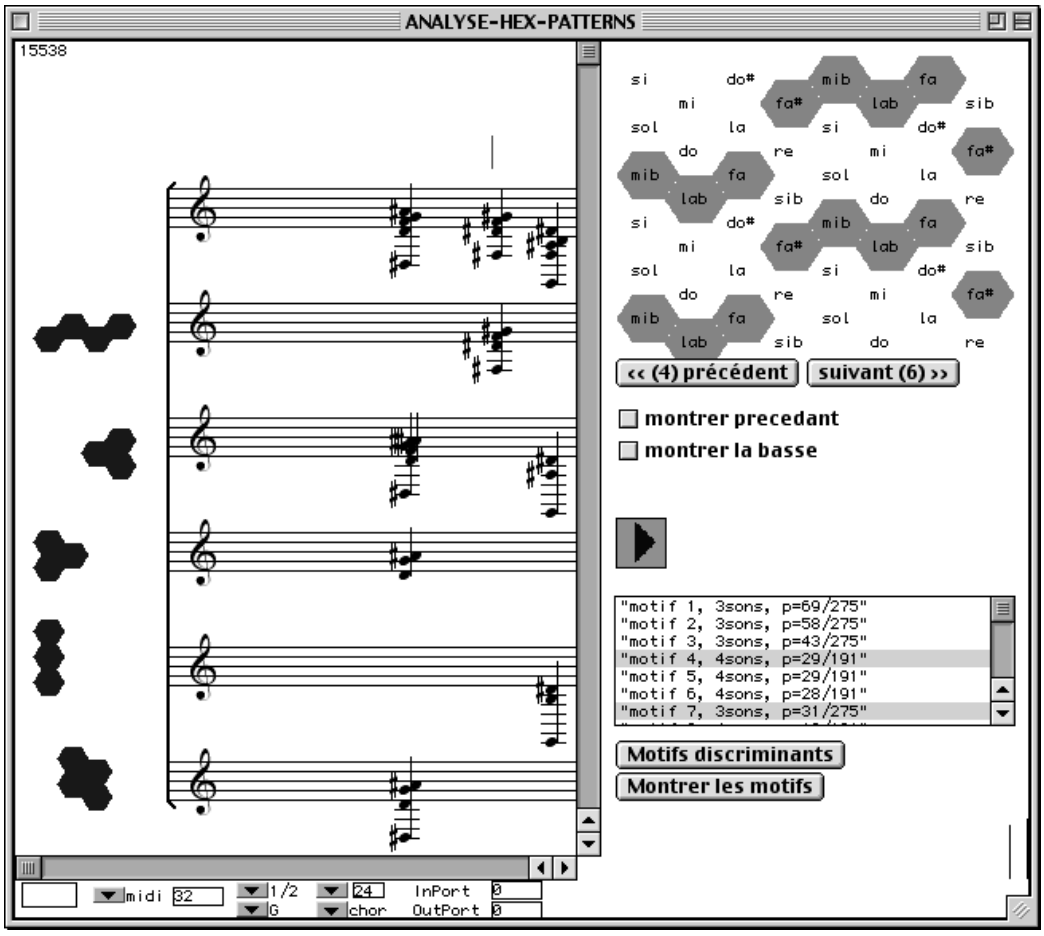


FIG. 3.10 – L'interface de l'objet *analyse-hex-patterns*

cords parfait majeur et mineur, ainsi que l'accord de quinte augmenté. La sélection de sous-motifs discriminants a été menée sur une autre pièce *musica das nuvens e do chao-red* qui se compose de deux harmonisation successive de la même mélodie, la deuxième étant beaucoup plus riche que la première. La sélection de sous-motifs discriminants propose 5 motifs pour la première harmonisation, et 12 pour la deuxième. Ceci met bien en évidence la complexité harmonique de chaque harmonisation. On peut émettre une réserve sur ce résultat, puisque la deuxième harmonisation comporte deux fois plus d'accords que la première.

3.3.4 Transitions

Dans un troisième temps, on se propose d'étudier la transition d'un accord à un autre, selon la représentation hexagonale. Un des courant de l'analyse musicale, appellé néo-riemannien, consiste à considérer toute évolution comme une

combinaison de transformations élémentaires[13].

Pour reprendre cette idée, on se propose de définir des transformations élémentaires sur la représentation hexagonale, et de rechercher le minimum d'opérations pour passer d'un accord à un autre.

Les transformations élémentaires

Elle sont au nombre de 4 :

1. La disparition d'une note.
2. L'apparition d'une note dans le voisinage d'une autre note.
3. La substitution d'une note par une autre note distante d'au maximum 1 ton (proximité chromatique).
4. Le déplacement de toutes les notes dans une des six directions du damier (c'est à dire une transposition dans un ton voisin).

Trouver une transition entre deux accords

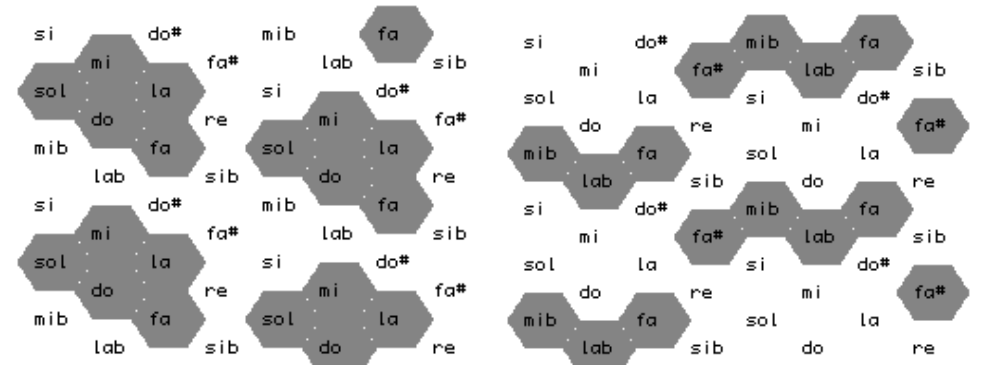
L'objectif est de trouver la suite de transformations élémentaires la plus courte permettant de transformer un accord en un autre accord, sous la contrainte que l'on ne passe jamais par un accord ayant moins (resp. plus) de sons que celui des deux accords données ayant le moins (resp. plus) de sons.

Il s'agit d'un problème de plus court chemin. Pour le résoudre, nous allons donc utiliser l'algorithme de Dijkstra, avec un coût égal à 1 pour chaque transformation élémentaire. Il se trouve que le nombre de combinaisons possibles fait largement exploser le calcul. En effet, pour un accord de 4 sons, les possibilités sont 4 disparitions, au moins 6 apparitions, 4 déplacements ou 6 transpositions. Un total de 20 possibilités à chaque étape. Parmi ces 20 possibilités, il nous faut définir quelles sont les meilleures à l'aide d'une heuristique particulière.

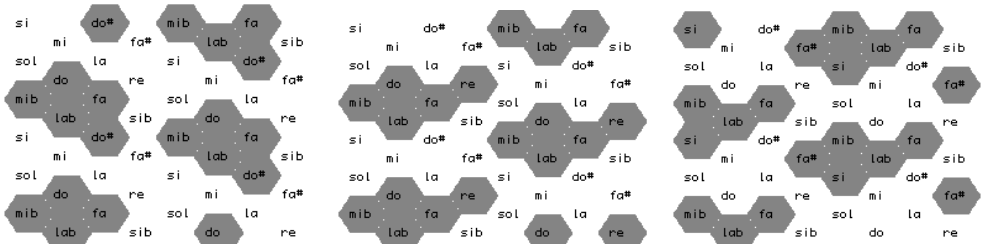
Heuristique

Etant donné un accord cible, nous considérerons, pour un accord donné, l'heuristique suivante :

- On ajoute le nombre de notes que les deux accords n'ont pas en commun. Cela correspond au nombre maximal d'opérations élémentaires qu'il faut faire pour transformer l'un en autre, si on oublie la contrainte du nombre de note.
- On retranche la taille du plus grand motif commun, moins 1, si ce motif fait au moins 3 notes. Cela représente le gain potentiel d'une transposition de l'accord, sachant que, compte-tenu de la taille du damier et de sa topologie de tore, dans au moins la moitié des cas, les motifs communs peuvent se superposer en une seule transposition.



(a) Accord de départ (b) Accord cible



(c) Etape 1 (d) Etape 2 (e) Etape 3

FIG. 3.11 – Décomposition d'une transition

Pour récapituler, si A et B sont deux accords :

$$\begin{aligned} distance\_heuristique(A,B) &= Card(A \cup B) - 2Card(A \cap B) \\ &\quad -Card(plus\_grand\_motif\_commun) - 1 \end{aligned}$$

Ce qui a posé problème lors de la conception de cette heuristique, c'est principalement le moyen de prendre en compte la transposition. Si on élimine la prise en compte des motifs communs, les solutions proposées oublient les transpositions. Les résultats obtenus avec l'heuristique présentée ci-dessus sont beaucoup plus satisfaisant.

Exemple de transition

La figure 3.11 présente la transition calculée entre l'accord de départ 3.11(a) et l'accord cible 3.11(b). Les étapes sont, successivement :

1. Transposition à 4 demi-tons inférieurs (3.11(a) à 3.11(c))
2. Substitution $Do\# \rightarrow Re$ (3.11(c) à 3.11(d))

3. Transposition à 3 demi-tons supérieurs (3.11(d) à 3.11(e))
4. Disparition de la note Si (3.11(e) à 3.11(b))

Une fois la transition déterminée sur le damier hexagonal, le travail n'est pas fini car cette transition ne nous donne pas les accords réels qui correspondent.

Calcul des accords

Après le calcul précédent, nous avons déterminé, à partir de l'accord de départ et de l'accord cible réels, les notes que doivent contenir les différents accords intermédiaires. Il reste maintenant, à « concrétiser »ces accords, en choisissant des notes réelles.

L'objectif étant de réaliser une transition fluide ou continue entre les accords donnés, le problème revient minimiser une distance[6] sous la contrainte de voir apparaître les accords choisis précédemment. OPENMUSIC possède une librairie de backtracking, permettant de résoudre des problèmes sous contraintes, mais pas de faire l'optimisation[7]. Il est possible de fixer un seuil de distance à ne pas dépasser. Cela permet de trouver des solutions. Mais ce n'est pas satisfaisant pour deux raisons : on est obliger de faire plusieurs essais pour trouver le minimum et le nombre de variables pose des problèmes de temps de calcul.

La méthode choisie La méthode choisie, discutable, consiste à déterminer des voix à partir de la représentation afin de les traiter indépendamment. Une voix sera une succession de notes appartenant à chaque accords. Pour former les voix, on se base sur les principes suivants :

- Dans le cas d'une transposition, toutes les voix suivent la transposition
- Dans le cas d'une substitution, les notes immobiles restent dans la même voix, et la note qui change également.
- Dans le cas d'une disparition, une voix disparaît, les autres ne changent pas.
- Dans le cas d'une apparition, une voix apparaît, les autres ne changent pas.

La définitions de telle règles nous garantit qu'en prenant comme point de départ une voix par note du premier accord, nous obtiendront bien toutes les notes de chaque accord intermédiaire ainsi que celles du dernier accord.

Les voix issues de l'exemple de la figure 3.11 sont présentées dans le tableau 3.1.

Cette décomposition en voix est un choix. Il y a d'autres manières de procéder. Nous pourrions envisager de considérer les notes immobiles dans une même voix, même dans le cas des transpositions.

[6] Nous faisons allusion ici à une distance entre accords, qui modéliserait la proximité à l'écoute, ceci afin d'obtenir une suite d'accords « fluide ».

[7] Une librairie de recherche adaptative est en cours de développement par Charlotte Truchet. Elle viendra enrichir largement les possibilités d'OPENMUSIC en matière de programmation par contrainte.

	départ	étape 1	étape 2	étape 3	cible
Voix 1	Do	Lab	Lab	Si	.
Voix 2	Fa	Do#	Re	Fa	Fa
Voix 3	La	Fa	Fa	Lab	Lab
Voix 4	Mi	Do	Do	Mib	Mib
Voix 5	Sol	Mib	Mib	Fa#	Fa#

TAB. 3.1 – Décomposition de la transition selon 5 voix.

Une fois les voix déterminées, il ne nous reste plus qu'à appliquer de nouveau l'algorithme de Dijkstra pour minimiser la distance sur les notes réelles correspondant aux notes calculées sur la représentation. Ceci est fait indépendamment pour chaque voix. Nous rappellons que les notes réelles du premier accord et du dernier accord sont connues.

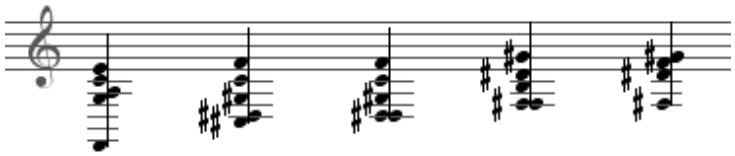


FIG. 3.12 – Accords calculés pour la transition étudiée.

Résultats Nous obtenons une séquence d'accords réels. L'impression de fluidité est présente dans la plupart des cas. Mais nous remarquons un défaut important qui vient du choix des voix. Ce choix est fait sur le damier hexagonal, c'est à dire que l'on a aucune idée de la place de la note réelle dans l'accord réel. De ce fait, il peut très bien se trouver que l'on construise une voix qui commence sur la basse du premier accord, et qui se termine sur la note la plus haute du dernier accord. Avoir des voix qui se croisent semble inévitable, mais ce type de croisement est particulièrement troublant à l'écoute.

Remarques Il serait intéressant de comparer le résultat obtenu avec d'autres manières de résoudre le problème. On peut aussi noter que le problème auquel nous avons été confronté, c'est à dire recalculer des accords réels à partir de notes, est un problème générique. Le même type de problème apparaît lors de la synthèse d'un sons après avoir déterminé les formants.

Chapitre 4

Conclusion

Les outils d'analyse musicale assistée par ordinateur A travers ce rapport, nous constatons que le champs des logiciels d'analyse musicale est ouvert, et qu'il met en oeuvre des axes de recherche aussi variés que l'analyse de données, la musicologie, la cognition ou l'intelligence artificielle. Négliger un de ces axes serait réduire largement la portée des outils dont on souhaite voir le jour.

Deux points, parmi d'autres, semblent importants :

1. La mise à la disposition des musicologues d'outils permettant de manipuler des représentations différentes leur offre un support de réflexion tout à fait riche et nouveau, qui peut leur permettre de mettre en place des raisonnements différents ou de faire apparaître des propriétés non explicitées jusque là. Il peut s'agir de mettre en place une représentation différente de la musique, au niveau symbolique, ce qui a été fait lors de ce stage, ou également d'offrir des outils de visualisation de données dans un sens plus large, les données pouvant être des descripteurs musicaux.
2. Les techniques actuelles en intelligence artificielle permettent de faire évoluer la machine au delà du stade de simple outil de calcul. La prise en compte de méthodes d'apprentissage, d'induction et de classification peuvent être une nouvelle source de réflexion pour les musicologues. Egalement, il serait intéressant d'essayer de tirer profit de l'interactivité que l'on peut mettre en place entre l'homme et la machine.

Bilan personnel Ce stage a été extrêmement positif sur plusieurs points. Tout d'abord, il a été pour moi une première expérience avec la recherche, dans un milieu à la fois particulier et ouvert, celui de l'informatique musicale. J'ai été amené à travailler avec des gens passionnés, et d'origine variée : chercheurs et thésards en informatique et en traitement du signal, musicologues, compositeurs, ce qui garantit la richesse et la qualité des échanges.

A l'image des activités variées de l'IRCAM , ce stage m'a permis de mettre

en oeuvre des compétences diverses acquises durant ma formation à l'ENST-Bretagne, aussi bien au niveau informatique et intelligence artificielle, qu'au niveau sciences cognitives, ou encore communication. Mes compétences musicales m'ont permis de donner une orientation particulière à ce stage de fin d'étude, et donc d'appliquer à une problématique originale les enseignements reçus.

Enfin, ce stage a été pour moi l'occasion de redécouvrir la musique et d'être sensibilisé à des problèmes tels que la formalisation dans l'art, la diffusion de la musique contemporaine, la place de l'outil informatique dans la création et l'analyse musicale.

Annexe A

HUMDRUM

A.1 Présentation

HUMDRUM est un logiciel d'aide à la recherche en musique, crée par David Huron en 1994. Les capacités de HUMDRUM sont assez abstraites, il est donc difficile de caractériser ce qu'il fait précisement. HUMDRUM peut manipuler un nombre de représentations illimité, et peut transformer, classer, rechercher, restructurer, comparer.

HUMDRUM est un outil destiné à la musicologie systématique , il permet de tester et vérifier des hypothèses formalisables à propos de musique (ou tout type de chose qui peut être représenté dans le format HUMDRUM).

HUMDRUM est composé de deux parties distinctes :

la partie passive : la spécification de la syntaxe des données.

la partie active : la librairie de fonctions destinées à manipuler les données.

A.2 La syntaxe

A.2.1 Description

La base de HUMDRUM réside dans la spécification de la grammaire utilisée pour représenter l'information. C'est cette grammaire que nous appelons syntaxe. Cette syntaxe est tout à fait générique et il est possible de construires plusieurs représentations différentes conformes à cette syntaxe, HUMDRUM en comporte une vingtaine, et l'utilisateur peut définir la sienne pour répondre à des besoins particuliers.

La syntaxe de HUMDRUM permet la représentation de tout type de données symboliques séquentielles, comme un spectre fréquentiel, des données MIDI, des pas de danse, des graphes de Schenker.

Un fichier de données pour HUMDRUM est un tableau à deux dimensions. Les colonnes sont utilisées pour représenter n'importe quel type de données, tandis que les lignes ont un sens défini : elle représentent les moments successifs du temps.

	**Pitch	**Duration	**Valve Combination
1st note	C4	quarter	0
2nd note	B3	eighth	2
3rd note	G4	eighth	0
4th note	F4	eighth	1
5th note	G4	eighth	0
6th note	A4	quarter	1-2
7th note	G4	eighth	0
8th note	Ab4	quarter	2-3

TAB. A.1 – Exemple de fichier HUMDRUM

Dans le tableau A.1, on trouve trois colonnes. La première, ***pitch*, représente la hauteur des notes jouées, la seconde, ***duration*, représente la durée de ces notes, et ***valve combination* représente les pistons de la trompette à enfoncer pour jouer la note.

Les caractères qui apparaissent dans le tableau n'ont pas d'importance, à partir du moment ou ils ont une signification pour l'utilisateur. Il s'agit seulement de symboles. La différence entre le format de donnée HUMDRUM et un tableau réside dans le fait que les "colonnes" des données de HUMDRUM peuvent de diviser, se rejoindre, apparaitre et disparaitre au cours du fichier. C'est pourquoi nous les appelerons *flux*[1].

A.2.2 Différentes représentations

Les fichiers les plus courament utilisés par HUMDRUM représentent de la musique, les flux représentent des voix différentes, les cellules représentent des notes. Il est important de noter que chaque flux peut utiliser une représentation différente, qui, en fait, correspond au type de données que le flux va représenter.

Par exemple, si le flux est destiné à représenter une voix d'une partition, on pourra utiliser la représentation **kern (la plus utilisée pour HUMDRUM), si le flux est destiné à représenter le texte d'un choeur, on utilisera la représentation **text.[2]

[1] *spines* dans le manuel

[2] Voir HUMDRUM *representations reference* :
http ://www.music-cog.ohio-state.edu/Humdrum/representations.toc.html

Il est possible de trouver diverses représentations pour le même phénomène, ainsi **kern n'est pas la seule manière de représenter une partition, on pourra utiliser **midi, **semits (où on conserve juste une valeur absolue représentant chaque demi-ton). Mais ces représentations ne renferment pas toujours la même information. Ainsi, certaines fonctions font des conversions, quand c'est possible. L'utilisateur peut créer sa propre représentation si il a des données particulières à traiter, des gestes d'instrumentistes, une notation particulière, etc. Certaines fonctions sont génériques (notament la recherche de motifs) et peuvent s'appliquer à n'importe quel type de données (compatible avec la syntaxe HUMDRUM).

A.2.3 La représentation **kern

La représentation **kern est la représentation la plus courament utilisée dans HUMDRUM. Ainsi, au moins 30000 oeuvres complètes ont été encodées dans cette représentation. Pour des raisons de droits, ces fichiers ne sont pas disponibles, on en trouve seulement quelques uns [3].

**kern permet le codage de la hauteur, la durée, les altérations, les ornementations, les liaisons, les phrasés, les glissandos, les reprises, les coups d'archets, les directions des queues des notes.[4]

A.3 Les fonctions

A.3.1 Organisation et interface

HUMDRUM n'est pas un programme que l'on lance, comme c'est le cas du MUSICOSCOPE, d'un traitement de texte, ou autre. HUMDRUM est une librairie de fonctions qui sont fournies à l'utilisateur pour lui permettre de manipuler les fichiers de la bonne syntaxe. Ces fonctions sont directement accessibles depuis un terminal UNIX.

Le choix de cette organisation permet une grande flexibilité d'interconnection des fonctions entre elles, mais aussi avec des autres programmes. En effet, les fonctions prises indépendemment réalisent des opérations plus ou moins simples, et c'est réellement les possibilités d'interconnections de ces fonctions qui font de HUMDRUM un logiciel puissant.

Le revers de la médaille est l'interface. Tout se fait à partir d'un terminal UNIX par des lignes de commandes, les résultats sont au format texte, à l'écran ou dans des fichiers. L'utilisation de HUMDRUM passe par l'accomodation à ce type d'interactions homme-machine. La puissance que l'utilisateur tirera de HUMDRUM

[3] voir *Kern Scores* : http ://kern.humdrum.net/

[4] Pour en savoir plus sur la représentation **kern :
http ://www.music-cog.ohio-state.edu/Humdrum/representations/kern.html

dépendra essentiellement de son aptitude à utiliser et connecter correctement les bonnes fonctions.

Exemple de fonctions HUMDRUM fournit une librairie de plus de 70 fonctions. Parmis ces fonctions on en distingue au moins deux types : les fonctions de manipulation, et les fonctions d'analyse.

Les fonctions de manipulation permettent à l'utilisateur d'assembler, extraire, visualiser, convertir, transposer, vérifier les fichiers de données afin de parvenir à l'information utilisable. A noter la présence de fonctions de conversion de fichier MIDI et Finale en fichier HUMDRUM (**kern), et vice-versa.

Dans le cadre de notre réflexion sur les outils d'analyse, nous allons nous attacher plus particulièrement aux fonctions d'analyse.

A.3.2 Recherche de similarités

patt* et *pattern Recherche des motifs donnés par l'utilisateur. L'utilisateur enregistre des motifs dans des fichiers annexes, puis peut, à l'aide de la commande *patt* rechercher ces motifs dans une oeuvre. *patt* peut créer un nouveau flux ou il placera des marqueurs, ou simplement insérer des commentaires. Un motif peut être définit à l'aide d'expressions régulières, ce qui permet de définir des motifs plus abstrait d'un simple motif mélodique. De plus, *patt* permet la définition de motifs multi-flux, ceci aussi avec des expressions régulières.

correl Mesure la corrélation entre deux flux de valeurs numériques. Plus précisement, *correl* calcule le coefficient de Pearson, c'est à dire le degré de linéarité entre les deux flux. On obtient un résultat de 1 si les deux flux sont en relation linéraire positive, -1 s'ils sont en relation linéaire négative, entre -1 et 1 sinon.

Si les deux flux ne sont pas de la même longueur, *correl* donne le coefficient de corrélation pour chaque alignement possible.

simil Mesure la distance d'édition entre deux flux. Il s'agit du plus petit nombre de manipulations élémentaires (substitution, insertion et déletion) nécessaires pour passer du premier flux au second. *simil* peut traiter aussi bien des données numériques que symboliques.

infot Calcule des indices issues de la théorie de l'information. Par exemple, on peut obtenir les informations suivantes sur un flux :

```
Total number of unique states in message: 4
Total information of message (in bits): 20.75
Total possible information for message: 24
Info per state for equi-prob distrib: 2
```

```
Average information conveyed per state: 1.73
Percent redundancy evident in message: 13.52
```

Ce résultat nous apprend que le flux ne comporte que 4 états (4 symboles différents), l'information totale du message, au sens de Shannon, est de 20.7549 bits. Un message de même longueur peut avoir comme information maximale 24 bits (atteint lorsque la distribution des symboles est équiprobable). Le nombre de bits nécessaires pour coder 4 symboles est 2, alors que dans notre message, chaque symbole renferme 1.72957 bits d'information. Enfin, la redondance dans le message est de 13.5%.

Cette fonction permet, en autres, de dresser des statistiques d'information sur un répertoire d'oeuvres, on peut obtenir les symboles utilisés ainsi que l'information moyenne qu'ils transportent. Ces statistiques peuvent ensuite servir à positionner une oeuvre particulière au sein du répertoire, et voir si telle oeuvre est exceptionnelle du point de vue informationnel.

A.3.3 Analyse tonale

hint, mint *hint* détermine les intervalles harmoniques entre des flux parallèles, *mint* les intervalles mélodiques de deux notes consécutives. Le résultat utilise des abréviations standards (m2 seconde mineure, P4 quarte juste, A6 sixte augmentée, etc.)

key Estime la tonalité d'un passage musical en utilisant une méthode proposée par Krumhansl basée sur des expériences en psychologie musicale.*key* n'est pas capable de distinguer les enharmonies.

key fournit trois résultats : la tonalité estimée, le coefficient de corrélation de Pearson avec un exemple majeur ou mineur, et un coefficient de confiance suivant qu'il y a d'autres bonnes tonalités candidates ou non.

A.3.4 Analyse sérielle

HUMDRUM fournit des fonctions et des représentations destinées à l'analyse sérielle :

pcset identifie les ensembles de classes de hauteurs et nous les donne dans la notation de Forte

iv calcule le vecteur d'intervalles d'un ensemble de classes de hauteurs

reihe permet d'appliquer des transformations sur une suite de classes de hauteurs (transposition, inversions, retrogradation, et inversion rétrograde).

nf détermine la forme normale d'un ensemble de classe de hauteurs (forme la plus compacte)

pf détermine la forme primitive d'un ensemble de classe de hauteurs (forme la plus compacte transposée pour commencer sur le degré 0)

pcset et *iv* peuvent permettre de générer le type de résultats du tableau A.2

**pc	**pcset	**name	**iv
0	1-1	tone	< 000000 >
0 2	2-2	major second	< 010000 >
0 3 7	3-11	minor triad	< 001110 >
0 4 7	3-11	major triad	< 001110 >
0 4 7 10	4-27	dominant seventh	< 012111 >
1 5 8 11	4-27	dominant seventh	< 012111 >
-	*-	*-	*-
...			

TAB. A.2 – Exemple de résultats des commandes *pcset* et *iv*

La commande *reihe* peut, par exemple, être utilisée pour construire une matrice (liste de série) à partir d'une série. Cette matrice peut ensuite servir de base à une recherche de série dans une oeuvre en la combinant avec les commandes *patt* et *pattern*.

A.3.5 Analyse rythmique

synco Mesure le "niveau de syncope" à chaque instant du flux. *synco* requiert deux flux en entrée, dont un doit être du type **metpos (représentation de la métrique). Cette fonction implémente un algorithme inspiré du travail de Lee et Longuet-Higgins.

urrhythm Analyse un passage musical en termes de prototype rythmiques de Johnson-Laird. *urrhythm* attribue à chaque temps une des trois fonctions suivantes : Notes, Syncope ou Autres. Cette fonction nécessite une métrique constante et ne traite seulement que celles dont le numérateur est 2, 3, 4, 6, 9 ou 12. Ces métriques sont divisées en 3 ou quatre temps.

A.3.6 Autres fonctions

melac Calcule les accents mélodiques d'un flux. *melac* implémente le modèle d'accents mélodiques de Joseph Thomassen. Pour chaque note du flux, melac attribut un accent représenté par un réel entre 0 et 1.

diss Calcule le degré de dissonance pour des spectres successifs. Cette fonction utilise un algorithme proposé par Kameoka et Kuriyagawa. *diss* necessite une entrée des flux au format **spect (représentation discrète d'un spectre de fréquences).

**pitch	**metpos	**synco
M2/4	*M2/4	*
tb8	*tb8	*tb8
=1	=1	=1
r	1	0
.	3	0¡
r	2	0
A4	3	0¡
=2	=2	=2
G4	1	0¡
.	3	0
B4	2	0¡
r	3	0
=3	=3	=3¡
C5	1	0
C5	3	0¡
.	2	0.41
B4	3	0¡
=4	=4	=4
-	*-	*-

TAB. A.3 – Exemple de résultat de la fonction *synco*

xdelta, ydelta *xdelta* calcule la différence entre deux données numériques successives, *ydelta* calcule la différence entre deux données numériques simultanées de deux flux parallèles.

A.4 Exemple d'utilisation

HUMDRUM a été utilisé par W. Forde Thompson et M. Stainton pour tester la validité du principe d'implication réalisation de Narmour[19].

A.4.1 Principe implication-réalisation de Narmour

Narmour développe le concept d'implication proposé par Meyer, et affirme qu'une formation musicale donnée contient certaines implications qui aspire à une réalisation. Cette réalisation peut se produire, il y aura cloture, ou ne pas se produire. Les cinq principes qu'exposent Narmour et auquels les auteurs vont s'attacher sont les suivants :

Direction du registre : Si l'intervalle d'implication est plus petit que la quarte

juste, alors la réalisation se fera dans la même direction. Sinon, il y aura un changement de direction

Intervalle : Si l'intervalle d'implication est plus petit que la quarte juste, alors la réalisation aura sensiblement le même intervalle, et si l'intevalle d'implication est plus grand que la quarte juste, alors l'intervalle de réalisation sera plus petit.

Retour au registre : L'intervalle de réalisation se termine sur une hauteur proche (unisson ou seconde) de la première note de l'intervalle d'implication.

Proximite : la réalisation se termine sur une note distante de moins d'une quarte juste de la seconde note (note de fin de l'implication et début de la réalisation)

Cloture : Un intervalle d'implication implique une réalisation.

A.4.2 Analyse réalisée

Les auteurs ont choisi de travailler sur 50 chorales de Bach et 16 mélodies de Schubert.

Identifier les intervalles d'implication Les critères retenus pour caractériser un intervalle d'implication sont les suivants :

- Seconde note aussi stable au niveau de la tonalité, ou moins, que la première
- Seconde note sur un temps plus faible que la première.
- Seconde note de durée aussi longue ou moins que la première.

Utilisation de HUMDRUM HUMDRUM a été utilisé pour préparer les données (commandes *extract* et *assemble*). Il a permit la transposition des notes sur l'échelle des degrés (commandes *deg*). La commande metpos génère automatiquement la métrique, ce qui va permettre de repérer facilement les temps forts et les temps plus faibles. Les commandes *semits* et *xdelta* vont permettre la mesure des intervalles.

Un script en language awk va permettre de quantifier la stabilitée tonale (selon une étude de Krumhansl et Kessler, 1982) et de repérer les intervalles d'implication. Un second script va se charger de classer et compter.

Résultats Parmis les intervalles identifiés comme intervalles d'implication par les critères précédents, les taux de vérification des principes de Narmour sont les suivants :

Les résultats du tableau A.4 montrent que les principes ne sont pas tous vérifiés avec la même concordance. Cependant, les pourcentages seuls ne veulent rien dire. Pour pouvoir tirer des conclusions il faudrait comparer avec des modèles.

	Bach sop.	*Bach bas.*	*Schubert*
Implication < quarte juste	96,6%	92,9%	
Direction du registre	52,9%	51,2%	47,1%
Intervalle	95,5%	92,8%	85%
Retour au registre	52,9%	41,5%	31,4%
Proximite	99,5%	92,3%	96,4%
Cloture	48,8%	45,8%	55,7%

Bach sop. la voie soprano d'un chorale de Bach
Bach bas. la voie de basse d'un chorale de Bach
Schubert une mélodie de Schubert

TAB. A.4 – Résultat obtenus par Forde Thompson et Stainton[19]

Par exemple, dans le cas d'une mélodie aléatoire, le taux qu'on obtiendrait pour la direction du registre serait 50%.

Les auteurs ont ensuite effectué une analyse multinomiale qui vise à voir si un ensemble d'observation Y peut être prédit par une somme pondérée de quelques prédicteurs. En choisissant comme prédicteurs les principes énoncés par Narmour, on peut se demander quels principes suffisent pour prédire les observations.

Cette analyse a montré que tous les principes de Narmour avait une valeurs prédictive significative, c'est à dire que laisser de coté un des principes conduit à des résultats beaucoup moins bons, sauf pour le principe intervalle (intervalle de taille inférieur ou équivalente) dans le cas des mélodies de Schubert.

Annexe B

MUSICOSCOPE

B.1 Présentation

Le MUSICOSCOPE est un logiciel d'analyse musicale écrit en Macintosh Common Lisp (ver 4.2 et au-delà) par Marcel Mesnage. Il résulte de l'intégration des anciens programmes MUSINOTE et MORPHOSCOPE. MUSINOTE permet à l'utilisateur de créer et modifier une partition via une interface graphique intuitive. Ensuite, MORPHOSCOPE peut récupèrer le fichier au format de MUSINOTE et l'utilise pour construire sa structure de données interne (muage) qui peut être interrogée par l'utilisateur. Lors d'une analyse, l'utilisateur peut utiliser les paramètres préprogrammés dans le MUSICOSCOPE, ou définir les siens.

Saisie et édition de partitions Le MUSICOSCOPE présente une interface utilisateur permettant de saisir et d'éditer une partition, avec des indications de nuances, de liaisons, la possibilité de créer plusieurs voix, etc.

B.2 Paramètres de l'analyse

B.2.1 La segmentation

Lors de son analyse, le MUSICOSCOPE a besoin structurer la partition. Le MUSICOSCOPE permet de découper la partition de deux manières, par voix et par segments.

Le découpage par voix dépend de la manière dont a été entrée la partition, et ne peut être modifié. Cependant le MUSICOSCOPE peut sélectionner les voix à analyser, ou décider de considérer toutes les voix comme une voix unique.

Le MUSICOSCOPE peut aussi découper la partition en segments, séparés par des séparateurs. Certaines fonctions d'analyse utilisent ce découpage, ce qui nous donne accès à une information plus synthétique Par exemple, on peut chercher

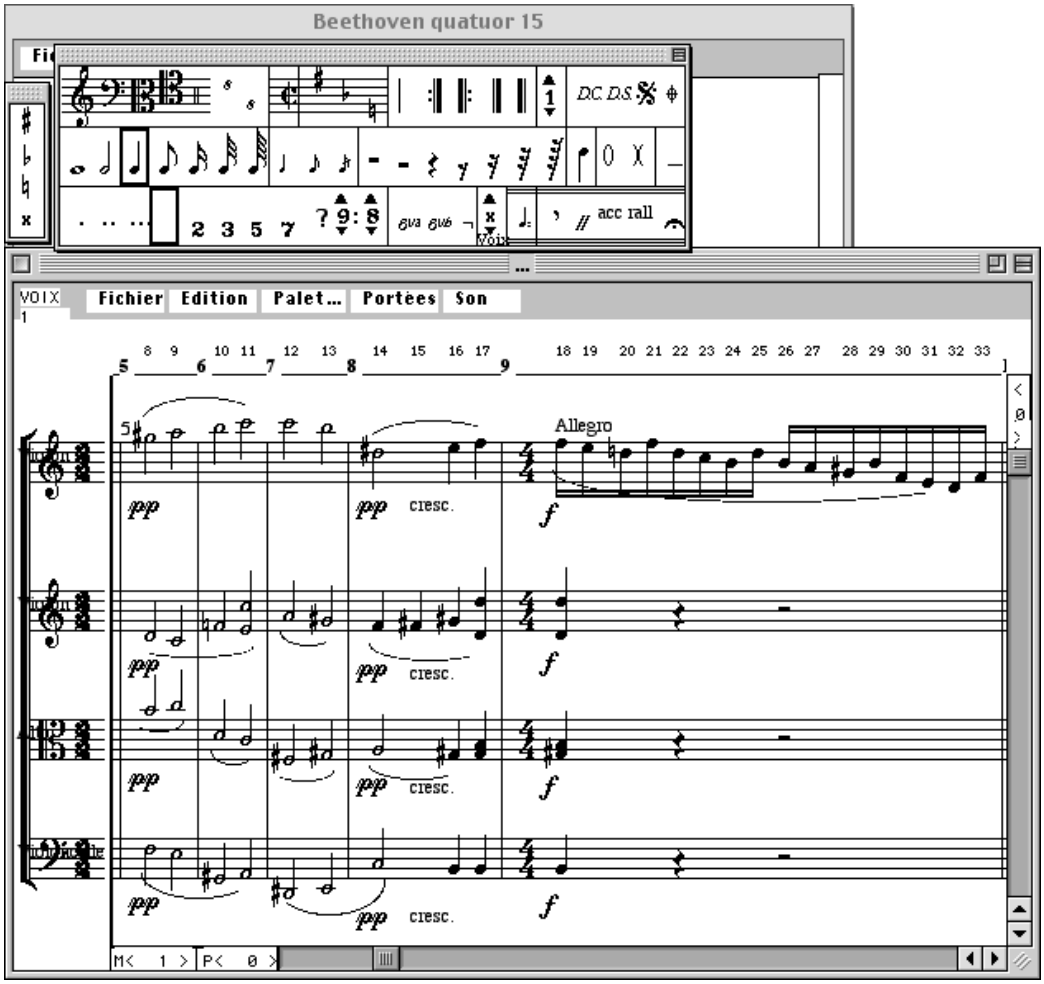


FIG. B.1 – Edition d'une partition dans MUSICOSCOPE

l'ensemble des notes utilisées au cours de chaque segment. Les séparateurs par défaut sont les mesures, mais l'utilisateur peut en définir de nouveaux de diverses manières :

graphique : l'utilisateur place les séparateurs directement sur la partition

crible ou résiduel : l'utilisateur peut donner la liste des dates qui définiront les séparateurs, ou regrouper les dates par groupe de N dates.

comparaison : en utilisant une composante (par exemple la hauteur d'une note), l'utilisateur peut, par exemple, décider de placer un séparateur avant chaque intervalle ascendant.

Par la suite, l'utilisateur peut décider d'analyser seulement quelques segments de l'oeuvre.

B.2.2 Les composantes

Une composante est un paramètre ponctuel (comme la hauteur d'une note, la densité d'un accord) qui servira de support à l'analyse. Le MUSICOSCOPE possède sa propre structure de données. Cette structure de donnée définit les composantes de base qui sont utilisées pour décrire la partition. Plusieurs autres composantes sont ensuite définies à partir des composantes de base. L'utilisateur peut aussi définir lui-même ses propres composantes (à partir des composantes de base) qui lui semblent plus pertinentes pour son analyse.

Les composantes simples sont :

date : durée cumulée du début de la pièce au début de l'évènement

durée : durée de l'évènement

hauteur : hauteur de la note, sous la forme de son nom (sol3, re#2) ou sa hauteur MIDI (34, 67)

intensité : correspond aux indications de type piano, forte, etc. classe de hauteurs : nom de la note indépendament de l'octave dans laquelle elle se situe, et en distanguant les enharmoniques (re b, do#, si)

chrome : dégré chromatique, (0 = do, 3= re# ou mi b)

On trouve aussi des composantes qui sont le résultat d'un calcul sur plusieurs évènements :

intervalles : l'intervalle mélodique représente l'intervalle entre deux notes succesives d'une même voix. Si on a affaire à des accords, le MUSICOSCOPE ne tient compte que de la note la plus haute. L'intervalle harmonique représente le(s) intervalle(s) présent(s) au sein d'un accord. On trouve aussi l'intervalle de date, durée entre les débuts de deux évènements consécutifs.

temps local : durée cumulée du début du segment au début de l'évènement accords : on trouve les accords de hauteurs, accords de classes de hauteur contour de hauteur : forme de l'accord décrite par la liste des intervalles entre deux notes superposées de l'accord réduit.

conjonction : représente les éléments communs entre deux évènements. Par exemple la conjonction de hauteur, la conjonction de classes de hauteurs, et la conjonction de contours. Par exemple : la conjonction de classe de hauteurs à un instant t sera la liste des classes de hauteurs présentes à l'instant t et à l'instant t+1.

densité : notament la densité de notes, ou la densité de classe de hauteurs représente le nombre de notes ou de classe de hauteurs à chaque date. Ajouter de nouvelles composantes

Le MUSICOSCOPE permet l'ajout de nouvelles composantes par l'utilisateur. L'utilisateur peut, sans notions particulière en programmation, aisément créer une

composante qui soit le résultat de comparaison ou opération simple ($+ - * /$) sur les composantes existantes. Par exemple, il est possible de définir facilement une composante 'Registre' qui donne 'haut' si la note est supérieure à DO5, 'moyen' si la note est entre DO3 et DO5 et 'bas' si la note est inférieure à DO3.

Cependant la création de certaine composantes, qui peuvent sembler basique au musicologue, comme l'ambitus d'un accord ou groupe de note, requiert quelques notions de programmation LISP et un peu de lecture de code afin de trouver les fonctions appropriées.

B.3 Les différente analyses et les graphes

Le MUSICOSCOPE offre plusieurs types d'analyse dont le résultat prendra la forme d'un graphe (en mode texte). Chaque type d'analyse peut être appliquée sur n'importe quelle composante, et avec n'importe quel découpage. Nous allons examiner les possibilités qu'offre le MUSICOSCOPE.

Flux de valeurs Représente la valeur de la composante, prise par chaque voix, date par date. Exemple : flux des valeurs de hauteur d'une partition à 4 voix :

```
        0       1       2       3       4       5       6       7       8      10      11      12      13      14
        1                                               2                               3
-------------------------------------------------------------------------------------------------------------------
 1     ré4 ___ré4 ___ré4 ___ré4 ___ré4 ___ré4                                       ré4 ___ré4 ___ré4 ___ré4
 2    sol0 ___sol0 ___sol0 ___sol0 ___sol0 ___sol0    sol0 ___sol0 ___sol0 ___sol0    sol0 ___sol0 ___sol0 ___sol0
 3            ré3     mi3     si3     ré4     mi4                                            ré3     mi3     si3
 4            ré2     mi2     si2     ré3     mi3             ré1     sol0     ré1            ré2     mi2     si2
```

FIG. B.2 – Graphe de type flux de valeurs

Déploiement Il s'agit d'un graphe avec en ordonnée les valeurs de la composante étudiée (dans l'ordre si un ordre à été défini) et en abscisse les dates successives. Chaque point représente la présence d'une valeur de la composante à une date donnée.

Champ segments Même type de représentation que le déploiement, le champ de segment indique les valeurs prises par la composante au sein de chaque segment (et non plus à chaque date). L'exemple de la figure B.4 montre les différents chromes présents à chaque mesure de la partition analysée.

Condensation On considère ici la suite des valeurs prises par la composante au sein d'un segment. Le graphe obtenu représente le déploiement des segments tout au long de l'oeuvre. Ce graphe permet de détecter facilement les segments identiques selon certaines composantes. Notons, que nous pouvons effectuer l'analyse

FIG. B.3 – Graphe de type déploiement

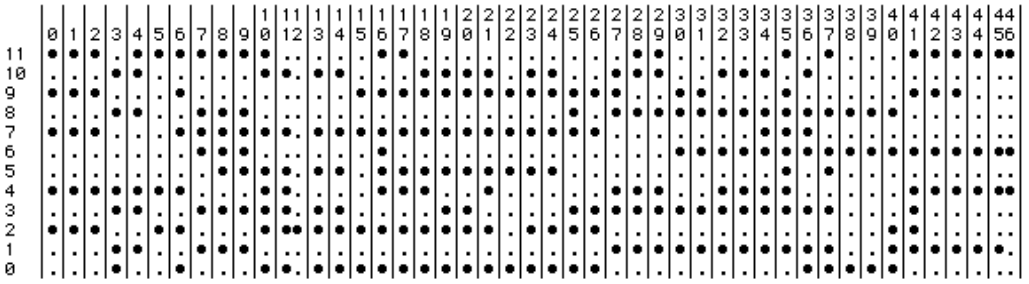


FIG. B.4 – Graphe type champ des segments

sur plusieurs composantes simultanément. Bien sûr, ce graphe ne sera efficace que si deux fragment sont rigoureusement identiques selon la ou les composantes sélectionnées.

Condensation champs Même principe que le graphe précédent, mais en ne considérant pas la suite des valeurs prises au sein d'un segment, mais seulement l'ensemble des valeurs prises.

Le graphe de la figure B.5 est une analyse de type condensation-champ sur les classes de hauteurs d'une oeuvre. Ce graphe nous montre que les 3 premiers segments utilisent les mêmes classes de hauteurs. On remarque aussi que les segements 28 et 29 utilisent les même classes de hauteurs que le segment 4.

Relations Le graphe de type relation dresse un tableau qui représente la valeur d'une composante en fonction de la valeur de l'autre. Le tableau indique pour

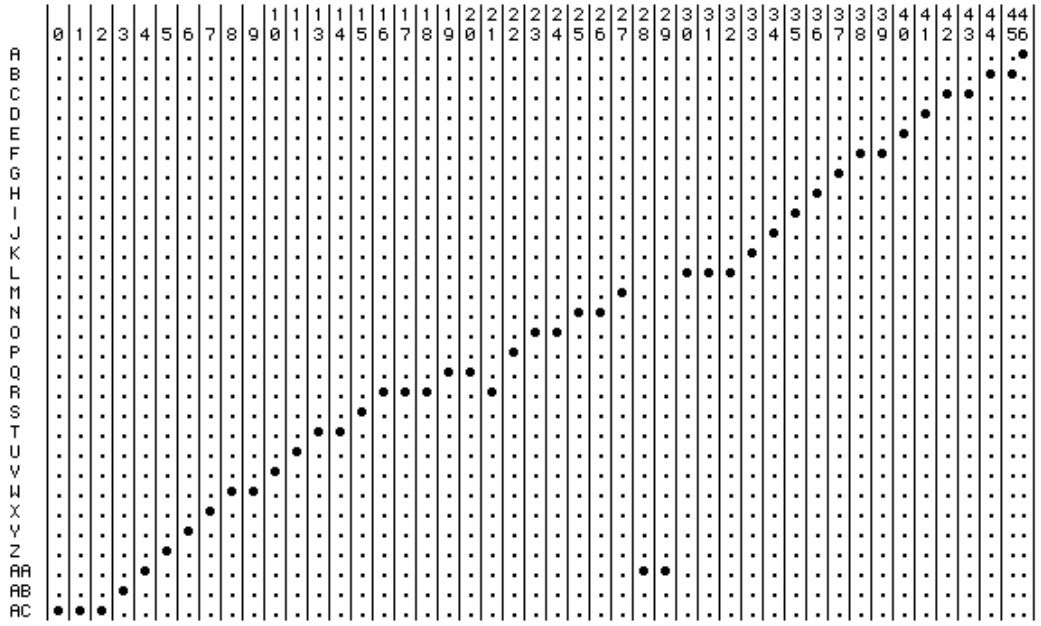


FIG. B.5 – Graphe type condensation des champs

chaque couple valeur des deux composantes sélectionnées si ces deux valeurs apparaissent une fois sur un même évènement. Par exemple, l'étude de la relation entre hauteur et durée peut nous apprendre que certaines hauteurs apparaissent régulièrement sous forme de blanches et rondes, alors que d'autres apparaissent uniquement sous forme de croches.

Comptage Il s'agit de compter le nombre d'occurence de chaque valeurs d'une ou des composante(s) sélectionnée(s). Il est possible de compter sur toute l'oeuvre, ou de compter segment par segment.

Gradient Il est possible de calculer le gradient d'apparition ou le gradient de disparition d'une composante. Par exemple, dans le cas du gradient d'apparition, cette analyse nous donne la liste des valeurs prises par une composante dans l'ordre d'apparition, avec la date d'apparition et une représentation graphique de l'intervalle entre les apparitions.

Transitions Ce type d'analyse compte les transitions d'une valeurs à une autre d'une composante et présente le résultat sous forme de tableau. Par exemple, la figure B.7 donne un graphe de transition de classe de hauteurs.

Le graphe de transition peut être effectué sur toute l'oeuvre comme sur chaque segment de l'oeuvre.

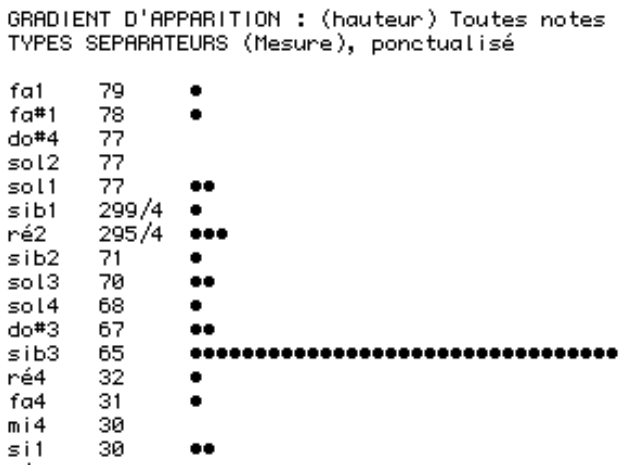
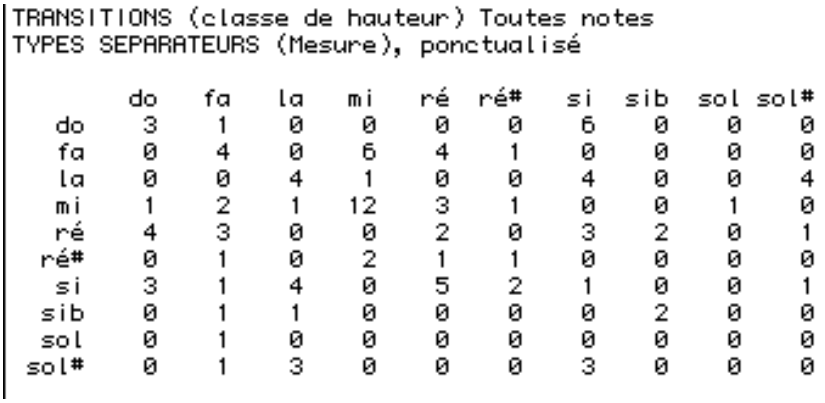


FIG. B.6 – Graphe type gradient d'apparition



	do	fa	la	mi	ré	ré#	si	sib	sol	sol#
do	3	1	0	0	0	0	6	0	0	0
fa	0	4	0	6	4	1	0	0	0	0
la	0	0	4	1	0	0	4	0	0	4
mi	1	2	1	12	3	1	0	0	1	0
ré	4	3	0	0	2	0	3	2	0	1
ré#	0	1	0	2	1	1	0	0	0	0
si	3	1	4	0	5	2	1	0	0	1
sib	0	1	1	0	0	0	0	2	0	0
sol	0	1	0	0	0	0	0	0	0	0
sol#	0	1	3	0	0	0	3	0	0	0

FIG. B.7 – Graphe type transitions

Règles de transition Dresse la liste des règles nécessaires pour décrire le flux des valeurs prises par la composante sélectionnée. Cette analyse trouve pour chaque valeur prise par la composante, les suites de prédécesseurs les plus courtes permettant de déterminer la valeur suivante.

On peut noter, dans la figure B.8 que le MUSICOSCOPE indique le nombre d'occurence de chaque valeur et le nombre de règles, ce qui nous permet de savoir si on a trouvé des règles génériques ou non. L'exemple ci-dessus nous montre que pour 21 occurences de mi on a besoin de 21 règles pour déterminer la valeur suivante, donc ces règles ne sont pas génériques. Dans le cas où on a des règles qui fonctionnent plus d'une fois, il serait intéressant de savoir combien de cas elles permettent de traiter, ce qui n'est pas indiqué actuellement.

```
REGLES DE TRANSITIONS (classe de hauteur) Toutes notes
TYPES SEPARATEURS (Mesure), ponctualisé
Longueur du flux 109 Nombre de règles 98

valeur : mi, nbre occurrences 21, nbre règles 21
  -> la, 1 préfixes: (ré# fa mi mi)
  -> sol, 1 préfixes: (la)
  -> fa, 2 préfixes: (ré# ré#)(sol# fa mi)
  -> mi, 12 préfixes: (ré fa)(sol# fa)(mi fa)(ré# fa)(ré fa mi)(ré# fa mi)(ré fa mi mi)(fa mi mi mi)(fa mi mi mi mi)(
fa mi mi mi mi)(fa mi mi mi mi mi mi)(fa mi mi mi mi mi mi mi)
  -> ré#, 1 préfixes: (mi mi mi mi mi mi mi mi)
  -> ré, 3 préfixes: (fa fa)(si fa)(mi fa mi)
  -> do, 1 préfixes: (si ré#)
valeur : si, nbre occurrences 17, nbre règles 15
  -> si, 1 préfixes: (ré# ré do)
  -> la, 3 préfixes: (si do)(si ré)(si)
  -> sol#, 1 préfixes: (mi do)
  -> fa, 1 préfixes: (fa fa mi ré fa ré do si ré si la sol#)
  -> ré#, 2 préfixes: (sol# la la)(do do)
  -> ré, 4 préfixes: (fa ré do)(mi ré)(si fa mi ré fa ré do si ré si la sol#)(ré sol#)
```

FIG. B.8 – Graphe type règles de transitions

Contours L'analyse de contours nous donne, pour chaque date, ou chaque segment, une représentation de l'harmonie dans le cercle chromatique.

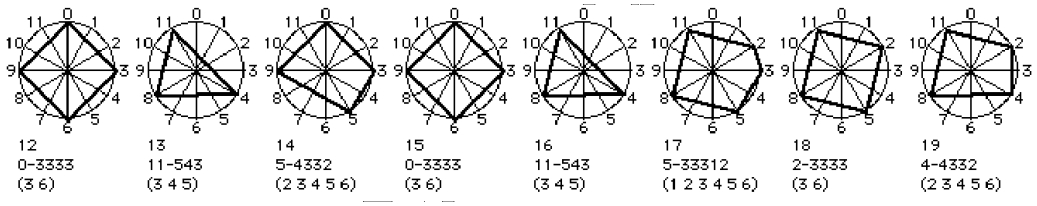


FIG. B.9 – Graphes type contour

Détection de répétitions Le MUSICOSCOPE offre des fonctions de détections de répétitions, transpositions, inversions d'un motif donné. L'utilisateur doit donc indiquer quel motif il souhaite rechercher. Le motif doit être complètement inclu dans un segment, et le MUSICOSCOPE ne retrouvera que les occurrences du motifs qui sont, de même, complètement incluses dans un autre segment. Il est possible de rechercher un motif dans toutes les voix, comme il est aussi possible de considérer toutes les voix comme une seule voix et rechercher la répétition d'un morceau de l'oeuvre.

B.4 Exemple d'utilisation

Didier Guigue, dans son analyse de *la cathédrale engloutie* de Debussy[9], a utilisé le logiciel PATCHWORK (qui a maintenant évolué en OPENMUSIC). Didier Guigue a utilisé PATCHWORK pour calculer l'évolution de l'espace dans l'oeuvre de Debussy. Nous allons voir brièvement, dans ce texte, comment le MUSICOSCOPE aurait pu être utilisé afin de retrouver les mêmes résultats.

B.4.1 Création des composantes utiles

D.Guigue a choisi, pour son analyse, de s'intéresser à l'utilisation de l'espace. Pour caractériser cette utilisation de l'espace, il a utilisé deux notions qui sont l'ambitus et le registre.

Ambitus et taux d'occupation

L'ambitus est l'écart entre la note la plus grave et la note la plus aigue d'un accord ou d'un fragment musical. Cet écart est mesuré en demi tons. Il s'agit d'un aspect quantitatif de l'utilisation de l'espace. A partir de l'ambitus, D.Guigue définit le taux d'occupation qui représente le rapport de l'ambitus à l'ambitus maximal du piano.

La création de la composante `AMBITUS` dans le MUSICOSCOPE demande quelques connaissances du language LISP ainsi qu'un peu de curiosité envers le code source. Voici comment nous pouvons définir cette composante :

```
(t (-
  (eval (cons 'max (mapcar
                     'midi-from-hauteur
     !h-accord)))
  (eval (cons 'min (mapcar
     'midi-from-hauteur
     !h-accord)))
))
```

Une fois la composante AMBITUS définie, nous pouvons définir la composante `TAUX_OCCUPATION`. Ce taux d'occupation nous donnera le pourcentage d'espace utilisé par rapport à l'espace maximal du piano (88 notes donc 87 demi-tons). On définit cette composante de la manière suivante :

```
(t (truncate (* 100 !AMBITUS) 87))
```

Le deploiement du taux d'occupation au sein de l'oeuvre, en suivant la fragmentation effectuée par D.Guigue est donné dans la figure B.10. Notons que l'échelle des ordonnées n'est pas continue. On retrouve bien les constatations de D.Guigue. : L'occupation est large au début, diminue jusqu'au climax (fragments 35 à 40) puis s'élargit à la fin.

Registre

D.Guigue a partitionné l'espace en 7 registres `(-3 -2 -1 0 1 2 3)` ou `0` est le registre moyen DO3 à DO5. Il a utilisé ce partitionnement pour rendre compte d'un aspect qualitatif de l'utilisation de l'espace, en attribuant à chaque registre un score de qualité qui permet ensuite d'évaluer la qualité d'un fragment

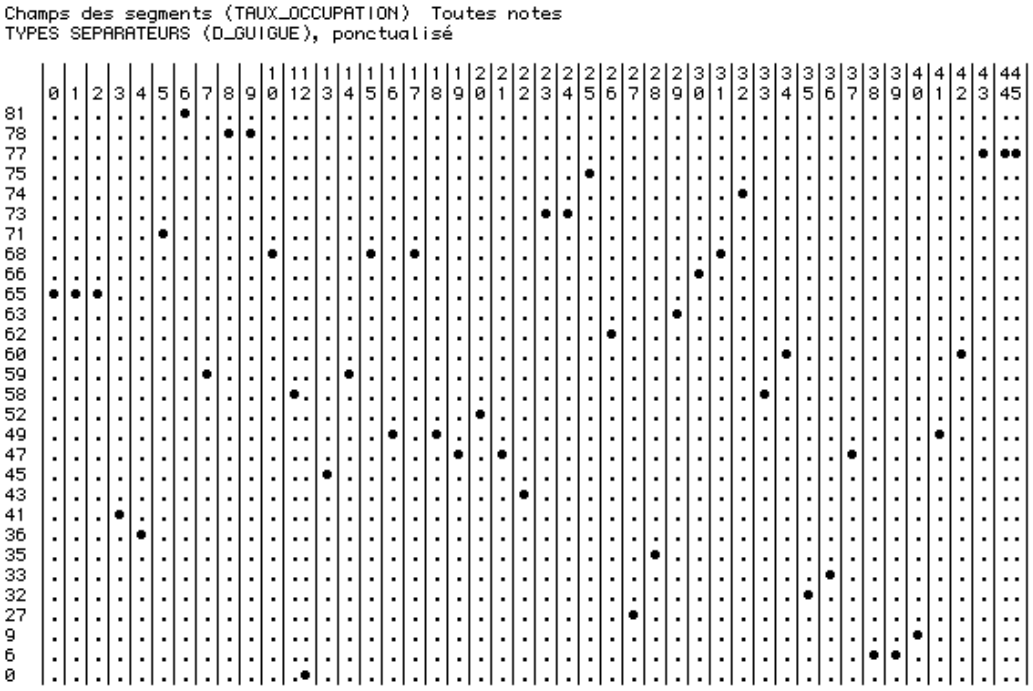


FIG. B.10 – Graphe du taux d'occupation

musical en terme d'utilisation de chaque registre. Un fragment musical recevra comme score de qualité de registre la somme des scores des registres qu'il

Voici comment définir la composante `QUALITE_REGISTRE` dans le MUSICOSCOPE :

```
(t
   (loop for (register weight) in '((-3 25) (-2 17)
                                    (-1 8) (1 8)
                                    (2 17) (3 25))
      when (member register
                   (loop for note in (mapcar 'midi-from-hauteur
                                             !h-accord)
                     when (> note 96)
                     collect 3
                     else when (> note 86)
                     collect 2
                     else when (> note 77)
                     collect 1
                     else when (> note 52)
                     collect 0
```

```
                    else when (> note 43)
                    collect -1
                    else when (> note 34)
                    collect -2
                    else  collect -3
                  ))
        sum weight
    )
)
```

Cette composante donne des notes qui vont de 0 pour le seul registre moyen à 100 pour l'utilisation de tous les registres.

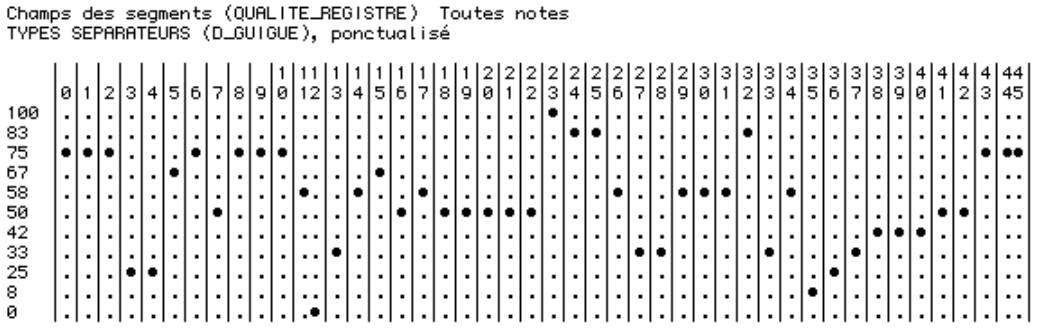


FIG. B.11 – Graphe de la qualité du registre

Le déploiement de cette composante au sein de l'oeuvre est donné à la figure B.11. On retrouve une évolution similaire à celle du taux d'occupation de l'espace, en diminution jusqu'au climax, et rétablissement sur la fin.

Espace

D.Guigue fait ensuite la synthèse des deux composantes en une composante unique, en faisant une simple moyenne du taux d'occupation et de la qualité du registre. La composante ESPACE est donc définie de la manière suivante dans le MUSICOSCOPE :

```
(t (truncate (+ !AMBITUS !QUALITE_REGISTRE) 2))
```

B.4.2 Analyse de la corrélation

D.Guigue remarque que le taux d'occupation et la qualité du registre évoluent linéairement au sein de l'oeuvre. La figure B.12 représente le graphe de relation entre ces deux composantes. Ce graphe rend compte d'une ébauche de relation

linéaire qui lierait les composantes QUALITE_REGISTRE et TAUX_OCCUPATION. On remarque que cette relation est plus accentuée pour les valeurs élevées, alors qu'elle devient plus contestable pour des valeurs faibles. Les valeurs élevées apparaissent au début et à la fin de l'oeuvre, alors que le valeurs faibles apparaissent au climax. On constate le même phénomène dans le graphe de D.Guigue fait sous PATCHWORK (voir figure B.13)

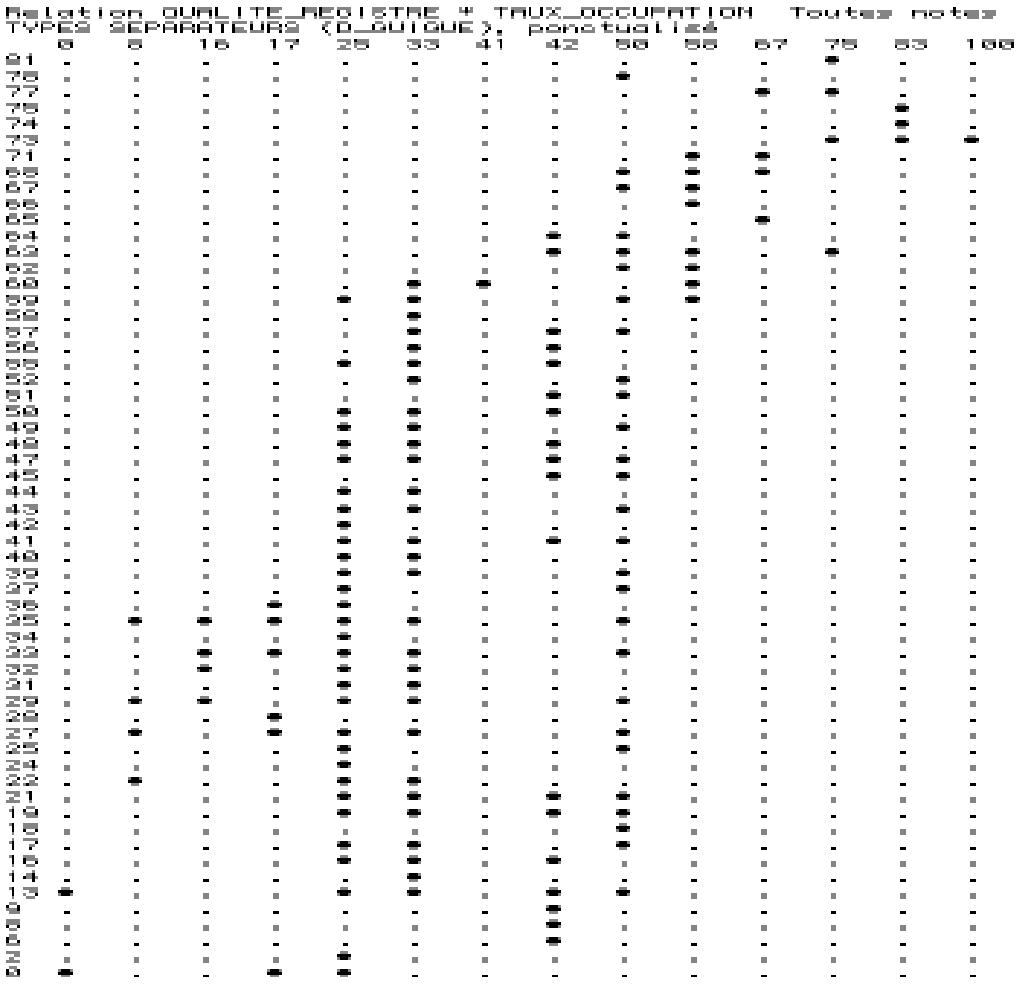


FIG. B.12 – Graphe de correlation entre le taux d'occupation et la qualité du registre

B.4.3 Remarques

A travers cette petite étude nous voyons un exemple d'utilisation du MUSICOSCOPE. Nous pouvons toutefois noter que la définition de certaines composantes

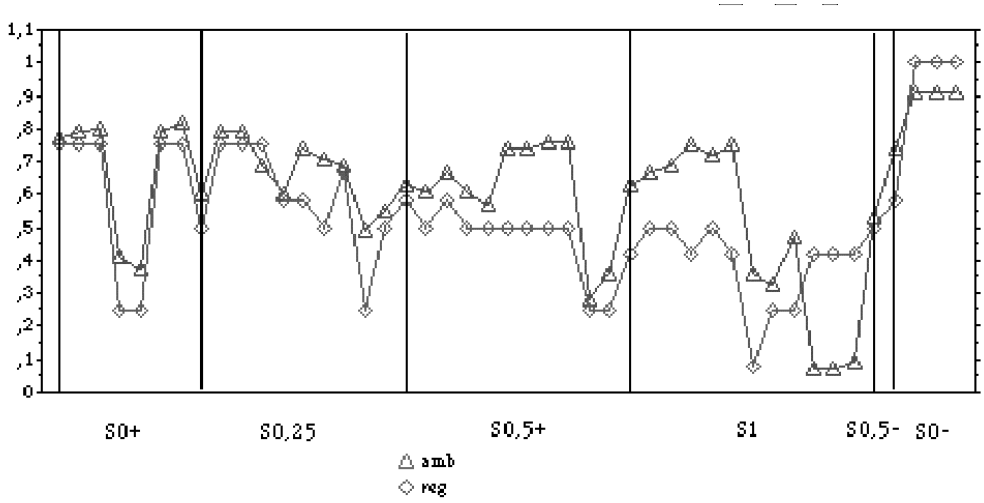


Fig.5 : Courbes représentant la séquence des taux d'occupation de l'espace et les registres investis.

FIG. B.13 – Graphe de correlation entre le taux d'occupation et la qualité du registre, selon D.Guigue

(comme l'ambitus ou la qualité du registre) n'est pas forcément chose facile. On remarque également que le mode de restitution des résultats a été travaillé mais reste rudimentaire.

Notons que cette petite étude n'a pas la prétention d'être représentative de toutes les possibilités du MUSICOSCOPE.

Annexe C

Cypher

C.1 Présentation

C.1.1 Introduction

Cypher est un logiciel développé par Robert Rowe. Il s'agit un logiciel intéractif dédié à la composition et l'éxécution. Le principe est de créer un logiciel qui soit capable d'interagir avec un musicien, via MIDI, dans le but d'improviser ou composer de la musique. Son architecture s'inspire des travaux de M. Minsky, décrits dans *The Society of Mind* (Minsky 1986). Ainsi ce programme va être composé de plusieurs agents, connectés entre eux.

Cypher comprend deux composants principaux : un *listener* et un *player*. Comme leurs noms l'indiquent, le *listener* se charge d'écouter les événements et de les analyser, afin de transmettre au *player* les directives à suivre. Dans notre contexte, nous allons nous intéresser au *listener* qui seul comporte quelques fonctionnalités d'analyse.

C.1.2 L'analyse dans Cypher

L'intégration d'un module d'analyse dans Cypher provient du soucis de la cohérence du discours musical. Cypher à pour vocation d'être un outil intéractif, qui doit improviser et composer avec d'autres musiciens, ou d'autres Cypher. Il est donc nécessaire que sa contribution reste en relation avec le contexte musical, et que cette relation puisse apparaître à un auditeur humain. Cependant, il n'y a pas de théorie généralement admise de l'écoute musicale. Et comme l'écoute de la musique n'est pas la même suivant la culture, il est nécessaire de faire des choix. Le choix de la norme MIDI impose déjà un axe, qui, par exemple, limite les possibilités au niveau des timbres. Finalement, Robert Rowe a choisi d'essayer de construire une écoute "occidentale".

Cette analyse a donc été orientée dans le but de doter Cypher d'une capacité d'écoute allant dans le même sens que celle d'un humain. C'est pour celà que Robert Rowe, sous l'influence des travaux de M. Minsky, a décider d'adopter une approche multi-agent et connectionniste.

C.1.3 Une organisation hiérarchique

Cypher est un système piloté par l'éxécution. Il ne s'appuie pas sur des représentations mémorisées de partitions musicales pour guider son intéraction avec les éxécutants humains.

L'analyse musicale a vu, avec Schenker (1933), Lerdahl et Jackendoff[12] (1983), et Narmour (1977), un tendance vigoureuse à la hiérarchisation. C'est dans cette optique que Cypher est organisé hiérarchiquement.

Les niveaux de hiérarchie de Cypher se distinguent de trois manières :

- Les niveaux supérieurs se réfèrent à un groupe d'objets traités au niveau inférieur. Par exemple, en ce qui concerne l'écoute, le niveau le plus bas étudie les événements individuels, tandis que le niveau immédiatement au-dessus étudie le comprtement à l'intérieur d'un groupe comprenant de tels événements.
- Les niveaux supérieurs utilisent des abstractions produites par les niveaux inférieur pour leur traitement. Par exemple, les agents de niveau inférieur classifie les données afin d'en extraire une information plus synthétique et abstraite, qui sera utilisée par les niveau supérieurs.
- Du fait de la nature temporelle de la musique, les niveaux supérieurs représentent des structures qui occupent des durées plus longues.

Au sein du *listener*, deux niveaux d'analyses coxistent : le niveau inférieur décrit des événements individuels d'accord ou de note, tandis que le niveau haut décrit la manière dont ces éléments changent dans le temps.

C.1.4 Les agents

Au premier niveau, on trouve des agents chargés de classer les événements selon certain critères. Chaque agent va se charger d'un aspect de l'événement, et ceci indépendement des autres. Voyons le rôle de quelques agents.

Register L'agent *Register* classe la plage de hauteurs dans laquelle est située un événement. Il les classe dans deux registres : graves et aiguës, selon que la hauteur est plus haute ou moins haute que le mi median. Cette classification peut paraître simpliste, mais il faut se rendre compte que la puissance du système va se trouver dans les intéractions entre les agents, et pas dans la précision d'un agent.

Dynamic L'agent *Dynamic* classe l'amplitude des évènements. Compte tenu que le calibrage de la dynamique est complètement différent d'un instrument MIDI à l'autre, il est nécéssaire de régler cet agent à la main en fonction de l'instrument MIDI qu'il écoute.

Density L'agent *Density* caractérise la densité verticale. Cette dernière dépend du nombre et de l'espacement des événements simultanés. La classification de cet agent se fait sur deux type d'informations : un événement est jugé comme étant une note seule ou un accord, et ensuite, s'il s'agit d'un accord, l'agent évalue son ambitus.

Speed L'agent *Speed* classe le décalage temporel entre l'événement couramment analysé et l'événement le précédent dans le temps. Il s'agit de l'intervalle inter-onset. Cet agent classe les événements entre quatre catégories correspondant à des vitesse différentes.

Duration L'agent *Duration* classe la longuer des événements. Il s'agit de l'intervalle entre le debut et la fin de l'événement. Un problème se pose lors de l'évaluation de la durée : pour évaluer la durée d'une événement, il faut attendre qu'il soit finit. Or, la plupart des analyse de premier niveau on lieu dès le début de l'événement. Il a donc été choisi de considérer, dans une approximation grossière que la durée d'un événement serait la même que celle de l'événement précédent (que l'on a elle évalué concrètement).

C.2 Analyse harmonique

Le sens harmonique implémenté dans Cypher est une version simple de la tonalité occidentale. Cypher utilise une approche connectionniste pour résoudre le problème de l'analyse harmonique.

C.2.1 Identification d'accord

Cypher se limite à la reconnaissance des accords parfaits majeurs et mineurs. Ceci non pas dans le but de limiter le vocabulaire harmonique, mais pour se concentrer sur la détection d'une fondamentale et d'un mode.

L'identification d'un accord se fait à l'aide d'un réseau à deux couches. Chaque noeud de la couche d'entrée représente une note, et ceux de la couche de sortie représentent les accord parfaits.

Les notes sont reliées aux accords auquels elles appartiennent, de façon que lorsqu'une note est activée, par propagation dans le réseau, les accords correspondants seront activés (voir figure C.1). Ainsi si les trois notes d'un accord sont activés, plusieurs accords seront activés, mais l'accord correspondant aux trois notes, et seulement lui, sera activé trois fois.

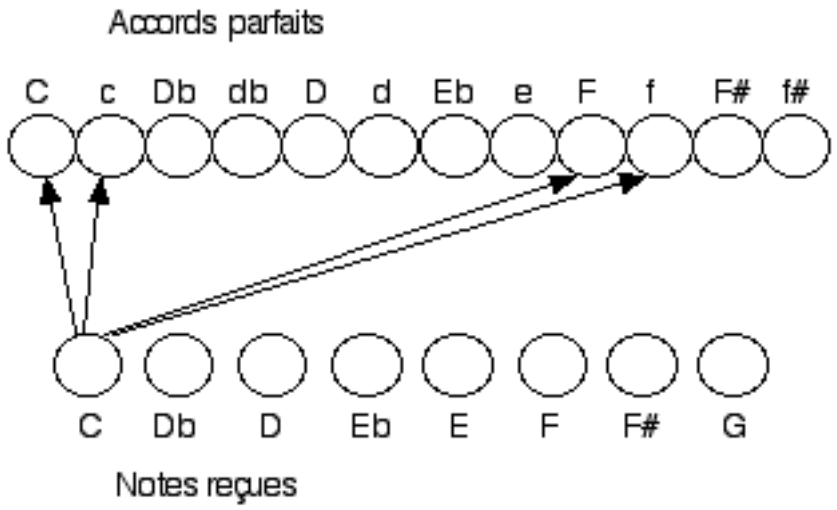


FIG. C.1 – Structure du réseau d'analyse harmonique

C.2.2 Améliorations

Pour améliorer le système, on attribut un coefficient à chaque liaison. Ceci pour traduire le fait que chaque note d'un accord joue un rôle différent. Ainsi la tonique d'un accord a plus d'importance que la quinte qui est à son tour plus importante que la tierce. Et si nous entendons DO et MI, nous allons en premier penser à DO majeur, et pas à LA mineur. Ainsi, le coefficient 5 est affecté aux liaisons entre un accord et sa tonique, 3 pour la quinte et 1 pour la tierce.

Afin d'obtenir des informations plus précises sur le contexte harmonique, le processus d'identification de tonalité est présent au sein d'une plus grande agence. Ainsi, il se retrouve connecté avec des agents tels que *Densité*, *Dynamique*, *Durée*.

Annexe D

Bibliographie

[1] G. ASSAYAG, *Vers la partition potentielle*, Les cahiers de l'IRCAM, vol. 3 (1993), pp. 13–41.

[2] P. BARBAUD, *Vademecum de l'ingénieur en musique*, Springer-Verlag, Paris, 1993.

[3] I. BENT, *L'analyse musicale*, Macmillan Press Ltd, 1987.

[4] E. BOROS, P. L.HAMMER, T. IBARAKI, A. KOGAN, E. MAYORAZ, AND I. MUCHNIK, *An implementation of logical analysis of data*, Séminaire Analyse Logique de Données, (2001).

[5] E. CAMBOUROPOULOS, *Towards a General Computational Theory of Musical Structure*, PhD thesis, University of Edinburgh, 1998.

[6] D. COPE, *Computers and musical style*, Oxford University Press, Oxford, 1991.

[7] ——, *Techniques of the contemporary composer*, Schirmer Books, New York, London, Singapore, 1997.

[8] A. FORTE, *The harmonic organization of the rite of spring*, Yale University Press, 1978.

[9] D. GUIGUE, *Sonorité, espace et forme dans la cathédrale engloutie de debussy*, (1994). `http://liaa.ch.ufpb.fr/~gmt/DiGuigue/espcath.htm`.

[10] O. LARTILLOT, *Kanthume : un projet d'analyse analogique suivant un modèle cognitif d'induction*, in Acte du 2ème colloque international d'épistémologie musicale, L'Harmattan, ed., 2002.

[11] M. LEMAN, M. LESAFFRE, AND K. TANGHE, *Introduction to the ipem toolbox for perception-based music analysis*, (2000). `http://farben.latrobe.edu.au/mikropol/volume7/leman_m.html`.

[12] F. LERDAHL AND R. JACKENDOFF, *A Generative Theory of Tonal Music*, MIT Press, Cambridge, 1985.

[13] D. LEWIN, *Musical Form and Tranformation*, Yale University Press, 1993.

[14] M. MESNAGE, *Techniques de segmentation automatique en analyse musicale*, Musurgia, (1994).

[15] M. MESNAGE AND A. RIOTTE, *Un modèle informatique d'une pièce de stravinsky*, Analyse Musicale, vol. 10 (1988), pp. 51–67.

[16] ——, *Modélisation informatique de partitions, analyse et composition assistées*, Les cahiers de l'IRCAM, vol. 3 (1993), pp. 43–59.

[17] B. MEUDIC, *Automatic meter extraction from midi files*, (2002).

[18] B. MEUDIC AND F. GOUYON, *Towards rhythmic content processing of musical signals : Complementary approaches*, (2002).

[19] W. F. THOMPSON AND M.STAINTON, *Using humdrum to analyse melodic structure : an assessment of narmour's implication-realization model*, Computing in musicology, vol. 10 (1995-1996).

[20] I. XENAKIS, *Musiques formelles*, Paris : Massé, 1963.