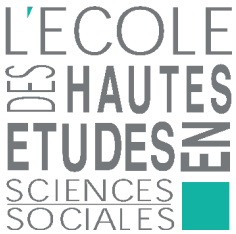
Master 2 ATIAM
Mémoire de stage de recherche

Intégrer l'harmonie dans un processus informatique d'improvisation musicale

---

Jérôme Nika, sous l'encadrement de Marc Chemillier.
15 mars 2011 - 15 août 2011



Remerciements

Je tiens tout d'abord à remercier Marc Chemillier qui m'a permis de me concentrer sur ces thématiques qui me sont chères, ainsi que pour ses conseils précieux et pour avoir toujours été à l'écoute de mes propositions.

Un grand merci à Carlos Agon pour sa bienveillance constante et son suivi des avancées de ce stage.

Merci aussi à Jean Bresson pour ses coups de pouce OpenMusic.

Merci à Benjamin Lévy et Gérard Assayag pour l'intérêt porté aux évolutions du projet, et pour avoir partagé leurs compétences (O)Maxiennes.

Merci également à Arshia Cont pour ses Antescofo sur mesure.

Un grand merci à toute l'équipe d'Uzeste : Bernard Lubat pour avoir donné de son temps pour inaugurer nos oracles, Fabrice Vieira pour son accueil et sa présence, et "JPax" pour son aide pendant les sessions.

Merci également à mes collègues ATIAM, en particulier à José pour l'enregistrement des morceaux du Realbook, ainsi qu'à Aymeric, Benjamin et leurs co-mezzanines Aurélie et Sandrine qui m'ont parfois accueilli à des heures tardives.

Je tiens enfin à adresser des remerciements tous particuliers à Sylvie Benoit, ma colocataire de bureau à la décoration sans cesse en évolution dont l'énergie est communicative même à distance (je passerai bientôt à 3 séries de 15).

Table des matières

Table des figures

Introduction

En retracant l'histoire encore brève de la la composition par informatique en 1988, Kemal Ebcioglu, l'auteur du système *CHORAL* [14] constate non sans ironie "*these are mostly avant-garde approaches ; computer-generated tonal music has somehow failed to be popular among computer musicians, perhaps because of its reactionary overtones.*"

Depuis ce projet d'harmonisation automatique dans le style des chorals de Bach souvent considéré comme l'acte fondateur de l'harmonisation d'une mélodie donnée en utilisant l'outil informatique, de nombreux systèmes ont vu le jour avec l'évolution des capacités matérielles et les développements ou apparitions de concepts comme la programmation par contraintes ou l'apprentissage automatique sur lesquels se basent une grande partie des dispositifs.

Les possibilités d'applications du champ informatique aux thématiques de la composition, de l'harmonisation, ou de l'arrangement musical étant vastes on peut référencer différentes catégories d'outils selon les entrées et sorties du système, ses connaissances a priori, son caractère interactif ou non... (voir la "taxonomie" établie dans [20]) : dans le cadre de ce projet, on se limitera aux systèmes d'harmonisation automatique, d'arrangement automatique (qui diffèrent des précédents par la connaissance d'une grille harmonique), et d'interaction mélodique (dispositifs faisant intervenir un ou plusieurs solistes virtuels).

L'étude de l'état de l'art permet de dégager des tendances générales associant la finalité du système et les moyens techniques et théoriques mis en oeuvre. Les trois grands domaines non-disjoints impliqués sont l'utilisation de **règles musicales formalisées**, d'un **corpus**, et le recours à **l'apprentissage automatique**. En partant du positionnement de son architecture relativement à ces différentes familles, les choix de conceptions et de réalisation du dispositif élaboré durant ce stage depuis les concepts théoriques au coeur de son fonctionnement jusqu'à l'élaboration d'un outil interactif sont détaillés.

Il consiste en un prototype permettant d'expérimenter l'intégration d'un cadre métrique donné (une pulsation) et d'une structure harmonique sous-jacente (une grille sous forme de labels d'accords) dans un contexte d'improvisation musicale.

Tout comme le logiciel d'improvisation OMax ([5, 4, 3, 18]), le système présenté repose sur l'Oracle introduit par Allauzen et al. ([1, 2]) pour tirer profit de ses propriétés particulièrement pertinentes et riches dans un contexte musical. Il consiste en un instrument d'improvisation destiné à être joué par un interprète pleinement intégré dans une formation et s'articule selon deux utilisations : l'interaction harmonique, consistant en un mécanisme d'harmonisation de mélodies et de génération d'improvisations informatiques harmonisées et arrangées en s'appuyant sur l'apprentissage d'un corpus pouvant être enrichi au cours même de la performance, ainsi que l'arrangement en temps réel d'une grille harmonique.

"L'art d'apprendre se réduit donc à imiter longtemps et à copier longtemps, comme le moindre musicien le sait, et le moindre peintre." Alain, *Propos sur l'éducation*

"Imiter quelqu'un, c'est dégager la part d'automatisme qu'il a laissée s'introduire dans sa personne." Henri Bergson, *Le rire*.

1. Situation par rapport à l'état de l'art

Ce chapitre propose une vision globale des caractéristiques principales de chacune des trois catégories énoncées précédemment accompagnée d'exemples de systèmes présentant des choix représentatifs des familles auxquelles ils appartiennent. L'ensemble de ceux-ci peut se résumer par différents parcours dans le schéma présenté en figure 1.1.

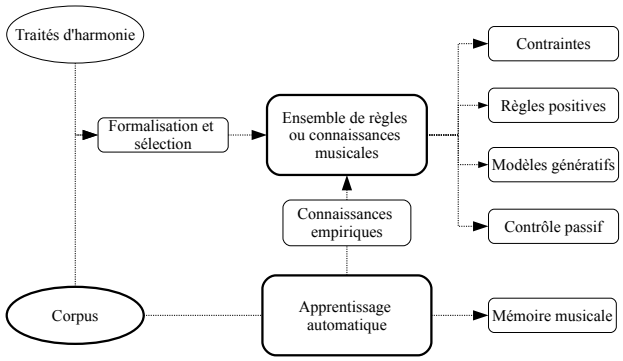


FIGURE 1.1 – Vue d'ensemble des différentes méthodes utilisées dans les systèmes d'harmonisation / arrangement automatique

1.1 Règles mélodiques et harmoniques *formalisées*

De manière générale, les systèmes employant des règles formalisées peuvent les utiliser selon deux modalités. Dans un premier cas, les règles sont activement impliquées dans la génération de l'harmonie. Dans le second, le matériau utilisé pour l'harmonie est généré par un autre procédé (recours à une mémoire musicale (technique dite de *fragments reusal*), modèles génératifs...), et les règles harmoniques n'interviennent qu'en fin de chaîne, en général pour assurer une cohérence globale.

Même s'il existe quelques initiatives intégralement déclaratives, la propagation de contraintes représente une partie majeure des systèmes utilisant exclusivement des règles formalisées. On peut tout de même citer le système de génération d'arrangements jazz de

N. Emura et al. ([15]) s'appuyant uniquement sur des règles d'arrangement jazz explicites et exclusivement locales à l'échelle d'une note de la mélodie, ainsi que **VirJa Session** ([16, 17]), un système d'arrangement interactif en temps réel déduisant du jeu d'un soliste des *paramètres d'intentions* (comme on en retrouvera fréquemment dans les systèmes basés sur une mémoire musicale présentés en 1.2)

1.1.1 Programmation par contraintes

Parmi les représentants de cette catégorie, on peut citer le système **Backtalk** de F. Pachet et P. Roy [23], qui ont consacré en 2001 un article [24] à l'étude des différentes applications de la programmation par contraintes pour la résolution de problèmes d'harmonisation. Ils divisent le procédé en deux étapes : le calcul de tous les accords possibles pour l'harmonisation de chaque note de la mélodie, puis le tracé de la séquence d'accord optimale le long de la mélodie.

Comme l'illustre le travail de Rafael Ramirez et Julio Peralta [27] se donnant pour objectif simple de créer un outil trouvant des harmonisations "correctes" en triades de mélodies populaires simples dans une tonalité fixée à la manière d'un accompagnement à la guitare, l'approche par contraintes suit le cheminement naturel de ce type d'entreprise : les "notes principales" à harmoniser dans la mélodies sont repérées et identifient les **variables**. Un *chord sequence generator* définit ensuite les 3 triades auxquelles peut appartenir chacune des notes (par exemple $\{C, Am, F\}$ pour un Do) : La suite d'ensembles d'accords constitue ainsi les **domaines** associés aux différentes variables. Les **contraintes**, enfin, sont constituées d'un ensemble de cadences courantes (I - IV - V - I, II - V - I...) sur lesquelles on veut calquer la future grille, ainsi que de quelques règles de substitution.

Pour chaque note, l'ensemble des accords possibles à chaque instant est stocké dans un *constraint store* qui est réduit à chaque propagation d'une contrainte. Lorsque les propagations de contraintes ne suffisent pas à assigner une unique valeur à une variable, un accord est arbitrairement choisi parmi les valeurs restantes.

1.1.2 Se spécialiser dans un style particulier en alliant contraintes et heuristiques

Le choix de construire un outil adapté à tous les styles en utilisant exclusivement des règles est un exercice difficile puisqu'il suppose de réduire l'ensemble de ses règles au dénominateur commun ou de prévoir une sélection de genre musical, d'école,...

Le choix de perdre une hypothétique généralité pour se concentrer sur une époque, un style, voire même un compositeur est donc le parti pris par les concepteurs de certains systèmes comme par exemple le projet *CHORAL* de Kemal Ebcioglu en 1988 [14], souvent considéré comme la première entreprise fructueuse d'harmonisation automatique, se concentrant donc sur les chorals de Bach.

CHORAL allie donc des contraintes provenant de traités d'harmonies, et des heuristiques issues de l'analyse des chorals.

En plus de l'harmonisation de chaque note de la mélodie, *CHORAL* considère des *sous-problèmes* complémentaires comme les relations entre des accords consécutifs, ou encore entre deux notes au sein d'une même voix, et compte pour ce faire un total de 350 contraintes.

1.1.3 Des règles formalisées en tant que contrôle passif : l'exemple du logiciel *Vielkang*

Le plug-in *Vielklang* développé par *zplane* est un parfait exemple de situation où des règles issues de traités d'harmonie jouent un rôle primordial, mais en aval de la génération elle-même du matériel harmonique.

Entièrement construit sur un ensemble de règles, ce plug-in effectue l'harmonisation homorythmique à quatre voix d'une piste audio selon des paramètres définis par l'utilisateur. Les différentes voix générées sont obtenues par transformations de la voix donnée en entrée.

Une des particularités du système décrit dans [32] réside dans son analyse perceptive préalable à l'harmonisation : une première étape du traitement cherche à extraire le contexte et la structure organisant la mélodie en s'appuyant sur la *Gestalttheorie* pour ainsi délimiter des phrases musicales.

Le contenu harmonique potentiel de chacune de ces phrases est ensuite calculé en utilisant des méthodes heuristiques et stochastiques, et la séquence des meilleures transitions entre les différents accords est déterminée à l'aide de règles classiques d'harmonie et de conduite des voix.

1.2 Apprentissage automatique sur un corpus et constitution d'une mémoire musicale

La **mémoire musicale** consiste en un repertoire de citations dans lequel le système cherche le fragment le plus approprié à chaque instant. Dans ce cas de figure, le corpus prend une part active à la génération de l'harmonie puisqu'il fournit les briques élémentaires. Le corpus peut-être constitué de standards, appris pendant un apprentissage "offline", et enrichi ou non à la volée, au fur et à mesure de la performance. La technique de *fragments reusal* revient donc à aller chercher au sein de cette mémoire le fragment d'harmonie le plus adapté au fragment mélodique donné en entrée. Le choix de l'unité de segmentation est un question primordiale dans ce types de systèmes puisqu'il s'agit de trouver le bon équilibre entre des tranches assez grandes pour assurer la continuité et un grain assez fin pour éviter de recopier les éléments du corpus de manière trop identifiable.

Selon les systèmes, on fait appel à la mémoire musicale avec plus ou moins d'initiatives : les citations peuvent être effectuées sans transformations, avec des transformations mineures relevant simplement d'une adaptation au contexte (transposition, transposition élargie à la notion de mode [20]...), et dans certains cas couplées avec des fragments générés à partir de règles empiriques apprises du corpus.

Dans ce cas, l'apprentissage consiste le plus souvent en une classification, et est donc **discriminatif** : les techniques régulièrement employées sont différentes approches de *clustering*. Différentes classes sont donc définies et l'appartenance d'un fragment mélodique à l'une d'entre elle lui associe un fragment d'accompagnement. Dans ce type de système, les règles harmoniques et mélodiques (si toutefois elles sont présentes) n'interviennent qu'en fin de chaîne afin d'assurer des mouvements cadenciels ou une certaine cohérence globale quand plusieurs choix sont possibles.

Plusieurs systèmes supposent que l'agent a des **intentions de jeu** et intègrent donc des caractéristiques plus abstraites en calculant des propriétés musicales ou **"primitives"** générales et globales indexées par des données de contour mélodique, dissonance, densité,

style rythmique, répétitions...

1.2.1 *Exemple* : Système de D. Martín (2009)

Ce système d'arrangement automatique ([20]) prend en entrée des données MIDI et recherche dans une base de données le fragment d'accompagnement adapté au fragment de mélodie le plus proche de chaque mesure de la mélodie donnée en entrée.

Il repose sur une **utilisation conjointe de connaissances musicales et de techniques d'apprentissage automatique** : la mélodie donnée en entrée est fragmentée et chaque fragment est rapproché d'un fragment de la base de donnée à l'aide des informations qui en sont extraites. Les techniques d'apprentissage interviennent dans la création du modèle établissant la relation entre mélodie et accompagnement.

Une fois le fragment d'accompagnement adapté trouvé, celui-ci est transformé pour correspondre au contexte local du fragment mélodique d'entrée ("transposition" dans le mode local). La base de donnée est constituée de 7 standards de jazz pour lesquels la mélodie et l'accompagnement ont été enregistrés par des musiciens (respectivement saxophone et piano MIDI).

Segmentation et extraction de données statistiques

L'unité de segmentation choisie est ici la **mesure**, et la représentation de chaque fragment se base sur **l'extraction de données statistiques** :

Pour la mélodie : durée moyenne des notes, pitch moyen, nombre de notes, *tension rate* (proportion des notes étrangères à l'accords pondérée par la durée).

Pour l'accompagnement : durée moyenne des notes, pitch moyen, vélocité moyenne, nombre d'attaques de notes distinctes dans l'accord, nombre moyen de notes distinctes par attaque.

1.2.2 *Exemple* : ImPact, G. Ramalho (1999)

Ce modèle temps réel (décrit dans [26, 25]) se concentre sur le comportement d'une basse au sein d'un jazz band et se fonde, comme le système précédent [20], sur la réutilisation de fragments mélodiques issus d'une "mémoire musicale" constitué d'une partie du répertoire de Ron Carter, et dont le choix est déterminé par le contexte. L'accent est mis sur l'importance d'associer des connaissances musicales théoriques pour permettre le choix des fragments les plus pertinents : l'architecture générale du système associe *"Case-based reasoning"* et *"Rule-based reasoning"*. Il est implémenté dans le langage *Smalltalk*-80 en utilisant le system *MusES* de F. Pachet ([22]), et se décompose en 3 blocs :

Listener collecte les données de l'environnement qui est modélisé à chaque instant par un *scénario* qui évolue au cours du temps. Celui-ci est donné en entrée au même titre que la grille et est constitué d'informations sur le reste du groupe comme par exemple "le pianiste joue en mode dorien", "les percussions sont de plus en plus en plus fortes", ou encore "le soliste fait des arpèges sur les notes de l'accord actuel de la grille".

Executor fait simplement office de planificateur et joue le segment n pendant que l'improvisateur calcule le segment $n+1$. Il est aussi en charge de modifier les dernières notes d'un fragment si nécessaire pour assurer la continuité du contour mélodique et éviter les sauts mélodiques.

Reasoner au coeur du système, décide du fragment à jouer lors du segment suivant en fonction des paramètres que sont **la grille** (avec une prise en compte du segment suivant), les différentes informations provenant de **l'environnement** (*scénario*), et **les sorties du bloc *executor***.

Les auteurs ont opté pour une segmentation de la grille en cadences en estimant qu'un contexte trop réduit (comme par exemple un accord dans le logiciel **Band in a Box** [21]) ne permettait pas d'assurer une cohérence musicale globale. Ainsi toute grille donnée en entrée est segmentée sur la base des cadences présentes dans le corpus.

Intentions de jeu et PACT

Le modèle suppose que l'agent a des **intentions de jeu**, comme par exemple "jouer de plus en plus fort jusqu'à la fin du chorus". La structure de PACT (*Potential ACTion*) permet d'intégrer ce types de caractéristiques plus abstraites : elle contient la description concrète d'un segment en tant que séquence de notes associée à un fragment de grille (en pratique durée, pitch...) ainsi que des propriétés musicales plus générales et globales (indexées par des données de contour mélodique, dissonance, densité, style rythmique, répétitions...). Les auteurs considèrent en effet que la description d'une phrase musicale dans ces termes se rapproche plus de la démarche d'un musicien plutôt que la justification d'un choix note par note.

C'est donc sous forme de PACTs que sont stockés les segments de la "mémoire musicale", c'est-à-dire des couples contenant les informations issues du contexte et les segments qui seront effectivement utilisés. On distingue :

- *Standard PACT* : décrivant de manière déclarative un fragment mélodique.
- *Transforming PACT* : décrivant des transformations applicables à un autre PACT source, par exemple "jouer cette séquence transposée à la quarte".

Récupération et transformation des fragments

Les segments sont recherchés par une méthode de plus proche voisins à une transposition près et contiennent donc parmi leurs descripteurs un coefficient d'adaptabilité pour éviter, par exemple, de dépasser la tessiture de l'instrument.

Des exemples musicaux ont été mis en ligne par l'auteur [1]. Ils présentent des lignes de basses générées sur la grille d'*Autumn leaves* à l'aide de modèles de plus en plus complexes, en partant de la simple application de règles et en ajoutant progressivement la récupération de fragments dans une mémoire musicale, les PACTs, et enfin leur activation par les situations et scénarios.

Alors que l'application de règles harmoniques et harmoniques donne un résultat relativement pauvre rythmiquement et harmoniquement, les derniers exemples présentent une ligne intéressante qui se complexifie au cours du temps et qui ne se répète pas à l'identique lors de deux expositions du thème.

1. `http://www.cin.ufpe.br/~glr/Thesis/examples.html`

Les auteurs voient comme limite première de leur modèle son incapacité d'auto-évaluation et d'apprentissage. En effet, lorsqu'il joue un fragment retrouvé et transformé, il n'a pas les moyens de juger de sa pertinence, ni ensuite de l'enregistrer dans sa mémoire musicale afin de l'enrichir.

1.3 Extraction de règles empiriques pour la création de modèles génératifs

Le corpus constitué en amont et/ou au fil de la performance ne constitue pas nécessairement une base de citations. Il peut être utilisé de manière indirecte dans le processus de génération en permettant la création de modèles à travers son apprentissage.

Dans ce cas, le corpus n'est utilisé qu'en amont : un apprentissage est effectué sur celui-ci pour créer des modèles génératifs qui seront ensuite utilisés pour produire l'harmonie qui sera associée à n'importe quelle nouvelle mélodie donnée en entrée. Ce type de système a souvent pour objectif d'être un outil tout-terrain à utiliser sur un style bien spécifique en ayant pris soin d'effectuer la phase d'apprentissage sur un corpus cohérent. Il s'agit en quelque sorte d'une généralisation de la spécification évoquée dans une partie précédente (1.1.2) avec *CHORAL*, si ce n'est que l'analyse du style n'est plus formalisée par le concepteur mais est effectuée empiriquement par l'apprentissage.

Les techniques d'apprentissage automatique utilisées sont ici axées vers la création de modèles génératifs (modèles de Markov,...) qui sont ensuite parcourus lors d'une phase de génération.

1.3.1 Décrire la mélodie

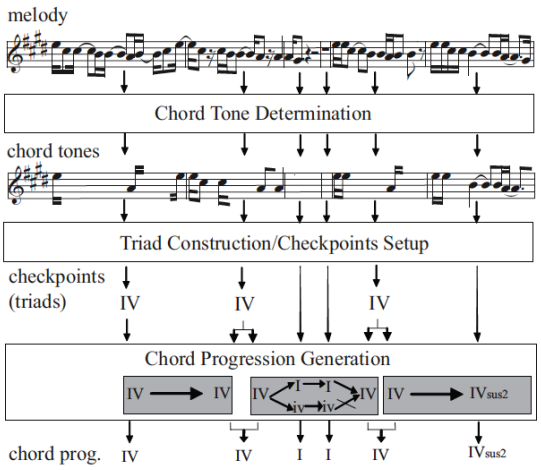


FIGURE 1.2 – Procédé d'harmonisation proposé dans [12]

La phase d'apprentissage revient dans ce type de systèmes à créer des modèles associant la mélodie à l'harmonie (ou en général à la dimension à produire en sortie quand il ne s'agit pas d'harmonie à proprement parler). Tout comme la question de fond des systèmes à mémoire musicale était le choix de l'unité de segmentation du corpus, un des critères déterminants est ici le **choix du mode de description de la mélodie** aussi bien pour

la phase d'apprentissage sur le corpus, que pour les mélodies données en entrées pour la génération.

Dans le cas du système d'harmonisation de C. Chuan et E. Chew ([12]), par exemple, la mélodie est réduite à un noyau de notes à harmoniser dans chaque mesure (voir figure 1.2) . La phase d'entraînement a alors pour but d'apprendre les associations entre ces notes cibles et la grille d'accompagnement dans les pièces du corpus afin de créer le modèle sous forme de chaînes de Markov.

Pour le logiciel **Songsmith** ([29, 28]) qui calcule un accompagnement à partir d'un flux audio en entrée, aucune note n'est laissée de côté : l'apprentissage consiste en la création de modèles de Markov cachés pour associer les séquences d'accords avec les mélodies. Pours chaque type d'accord et chaque fondamentale, on somme pour chaque morceau du corpus la durée de chaque note de la mélodie qui lui est simultanée pour obtenir après normalisation des probabilités de présence d'un accord sous une note. La matrice ainsi constituée est couplée lors de la phase de génération à une matrice de transition qui, avec une construction similaire, recense les probabilités pour chaque type d'accord d'être suivi par un accord d'un type donné.

1.3.2 *Exemple* "Band Out of a box"

Band Out of a Box (BoB) ([30, 31]) développé par B. Thom est un parfait exemple de système n'utilisant aucune règle harmonique ou mélodique et, fait plus rare, un nombre très restreint de connaissances musicales. *BoB* consiste en un "compagnon pour l'improvisation" échangeant sous forme d'interaction mélodique avec un musicien soliste dans une logique "trade fours" (chaque soliste prend la main le temps de 4 mesures) et ne se préoccupe donc pas d'harmonisation ni d'arrangement. Néanmoins, les constats au départ de ce projet, certains points de l'implémentation pratique, ainsi surtout que le type d'interaction impliqué sont assez proches de la philosophie du dispositif présenté dans la suite du document.

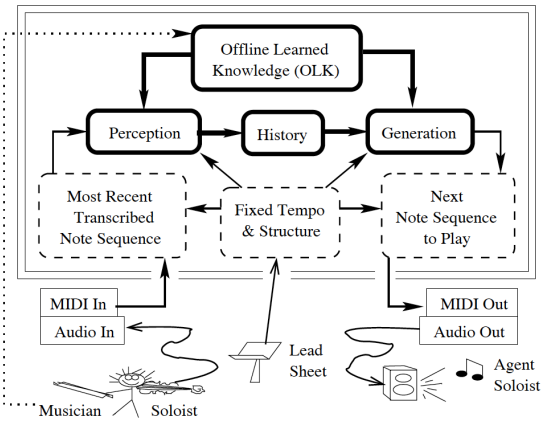


FIGURE 1.3 – Architecture de BoB [31]

BoB apprend comment un musicien improvise lors d'une phase d'apprentissage, et crée un modèle pour pouvoir ensuite dialoguer avec lui. Comme schématisé en figure 1.3, B.

Thom distingue l'apprentissage "offline" et l'apprentissage "online" : afin que l'improvisation ne soit pas trop pauvre au début de la performance, il est en effet nécessaire que la mémoire du système ne soit pas vide. Il s'agit donc d'effectuer au préalable un apprentissage "offline" pouvant utiliser comme matériau les "échauffements des musiciens" ([30]) ou encore un corpus préalablement appris (Charlie Parker et Stéphane Grappelli dans [31])

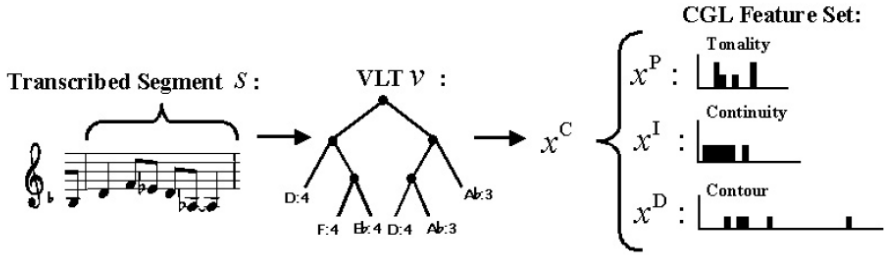


FIGURE 1.4 – Extraction et indexation des données d'une mesure : arbre et histogrammes [31]

Les exemples pour l'étape d'apprentissage "offline" ainsi que les entrées pendant la performance subissent le même traitement : le solo est découpé en mesures, et chacune d'entre elles est représentée par un arbre (figure 1.4) dont les feuilles contiennent la hauteur des notes.

Des données sont calculées pour chaque mesure sous forme d'histogrammes : **hauteur/-tonalité**, **intervalles/continuité**, **direction mélodique/contour**. Toutes les mesures sont attribuées à des classes sur les critères indexés par ces histogrammes (*playing modes*) en suivant un modèle appris des histogrammes par une méthode *Expectation-Maximisation.*

Le classement des mesures s'accompagne de la définition d'un "**coefficient de surprise**" qui quantifie la distance d'une mesure au centre du cluster auquel elle a été associée.

Ce coefficient est ensuite utilisé dans la génération du prochain solo de *BoB* : en effet, en suivant la logique "trade fours", la mesure y_i ($i \in [1,4]$) qui sera jouée par *BoB* est construite à partir d'un matériau provenant de la même classe que la mesure x_i précédemment jouée par le musicien et présentant un coefficient de surprise comparable.

Le matériau choisi pour générer y_i, provenant donc de l'apprentissage "offline" ou "online" subit ensuite 2 traitements. Premièrement une transformation rythmique, c'est-à-dire un enrichissement ou une simplification du squelette de l'arbre de la mesure, puis la génération d'une suite de hauteurs à l'aide du modèle appris que l'on attribue ensuite aux feuilles de l'arbre de rythme obtenu par transformation de l'entrée lors de l'étape précédente (figure 1.5).

1.4 Positionnement du prototype réalisé

En 2008, N.Emura et al. justifient leur choix de concevoir un système exclusivement basé sur l'application de règles d'arrangements explicites ([15]) en considérant que la plupart des systèmes utilisant une concaténation d'arrangements pré-existants sont condamnés à proposer une très faible variété de solutions, voire à proposer "le même arrangement pour n'importe quelle mélodie donnée en entrée". De plus, ils déplorent le fait que ces

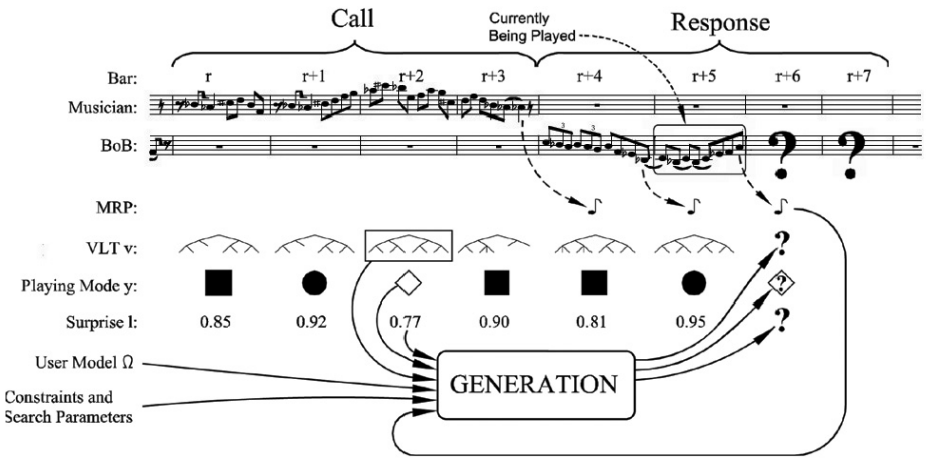


FIGURE 1.5 – Génération d'une mesure jouée par BoB [31]

systèmes ne permettent aucune coloration personnelle propre aux musiciens qui les utilisent puisqu'ils ne prennent en compte aucune instruction de l'utilisateur et que leurs "mémoires musicales" sont imperméables à toute nouveauté. Ces remarques font écho à celle de B.Thom dans [31] qui remarquait que rares étaient les systèmes, qu'ils utilisent des règles ou une "mémoire musicale", qui développaient une esthétique réellement dépendante du musicien qui l'utilise.

C'est pour éviter ce type d'écueils que le dispositif présenté dans la suite de ce document s'est construit sur un prototype antérieur à l'actuelle version du logiciel d'interaction musicale pour l'improvisation OMax ([5, 4, 3, 18]) : **OMax 2.0** datant de 2004 tournant sous Common Lisp dans Open Music 4.9. Ce modèle fonctionnant à l'époque de manière indépendante comme une librairie Open Music sans interaction temps réel avec Max/MSP.

Il bénéficie ainsi de l'Oracle de Allauzen et al. (voir 2.1.1) et donc de la capacité d'apprendre le style d'un improvisateur humain ([6]). De plus l'atout de la structure d'oracle au coeur du système est ici de pouvoir assurer une cohérence du langage musical tout en ayant une finesse de calcul à l'échelle de la pulsation. Son utilisation permet ainsi dans une certaine mesure de s'affranchir du dilemme du choix entre continuité et innovation par rapport aux phrases du corpus auquel se confrontent plusieurs systèmes utilisant une mémoire musicale.

Conçu à partir de l'apprentissage sur un corpus, on verra par la suite que l'utilisation faite de l'oracle dans le cadre de notre projet diffère de celle faite dans OMax. La séparation en étapes distinctes des processus d'harmonisation et d'arrangement lui confère une place hybride au sein des systèmes basés sur un corpus, puisque celui-ci est utilisé simultanément pour la création de modèles génératifs, et comme mémoire musicale.

Le prototype réalisé se veut interactif et apte à être intégré à terme au sein d'une formation au cours de sessions d'improvisation. A travers une interface développée sous Max/MSP, il prend en entrée des canaux MIDI captant le jeu d'un ou plusieurs musiciens, et enregistre ces données en effectuant en direct le formatage nécessaire à leur traitement. Par l'intermédiaire de cette même interface, l'utilisateur peut séléctionner une grille har-

monique connue qui pourra être changée au cours de la performance et qui constitue la colonne vertébrale de l'improvisation. Il a ensuite accès à des paramètres de contrôle plus ou moins fins pour piloter le calcul de phrases musicales de natures diverses (arrangement de la grille, mélodies harmonisées, nouvelles improvisations harmonisées ou non) dont le lancement s'effectue une fois encore depuis le patch Max/MSP au sein duquel se situent les sorties MIDI.

Les consignes de calculs et les paramètres sont transmis via le protocole OSC à une bibliothèque OpenMusic construite sur la base de la bibliothèque *OMax 2.0*[2] contenant l'implémentation informatique de la structure d'oracle en "mode *beat*" utilisée lors du processus d'harmonisation et d'arrangement.

2. Un tutoriel sur les classes utilisées en "mode *beat*" est disponible en ligne [11]

2. La pulsation comme élément musical et structurant

Ce chapitre présente le cadre théorique dans lequel se place l'utilisation de la structure d'oracle, puis l'application musicale qui en est faite centrée autour de la notion de pulsation. Enfin, l'utilisation du patch **Antescofo** et de l'objet **BeatTracker** pour la labellisation des entrées par pulsation et la restitution des phrases au tempo variable donnée par les musiciens est détaillée.

2.1 La structure d'oracle

2.1.1 L'oracle des facteurs

Tout comme dans le cas du logiciel *OMax*, l'**Oracle des facteurs** de C. Allauzen, M. Crochemore et M. Raffinot ([1, 2]) est au coeur du système présenté dans ce document. La structure et la construction de cet automate trouvant entres autres des applications dans le domaine de la génétique lui confèrent la capacité de reconnaître les motifs présents dans une séquence : son utilisation première est de calculer de manière optimale les facteurs présents dans une chaîne de caractères dont les éléments constituent l'alphabet des étiquettes des transitions. Enfin, en ce qui concerne son implémentation informatique, sa construction incrémentale linéaire en temps et en espace (voir paragraphe 2.1.2) présente une efficacité particulièrement adaptée à l'interaction temps réel.

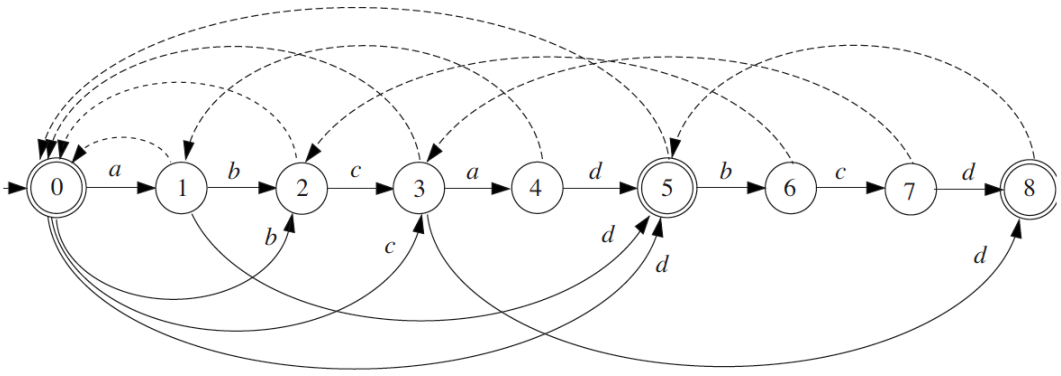


FIGURE 2.1 – Oracle des suffixes du mot *abcadbcd* ([10])

Comme l'illustre la figure 2.1, l'oracle des facteurs est constitué de deux catégories de liens :

Les transitions : d'un état **i** à un **j** avec **i<j**, liens en avant représentés en traits pleins, notés par la suite **j=t(i,x)** avec **x** l'étiquette de la transition.

Les liens suffixiels : depuis d'un état **i** jusqu'à un état **j** avec **i>j**, liens arrières représentés en pointillés, notés par la suite **j=s(i)**.

On appelle "**écorché**" du mot le sous-graphe constitué des états et des transitions vers leurs successeurs directs, il représente la séquence d'entrée à partir de laquelle l'apprentissage a été effectué pour construire l'oracle. Les parcours de l'oracle employant ces transitions en plus de celle présentes dans l'écorché permettent de générer tous les facteurs présents dans la séquence d'origine (l'oracle reconnaît en réalité un peu plus que les facteurs du mot original, voir [19]).

On remarque sur la figure 2.1 que les dernières lettres lues avant de quitter un état par un lien suffixiel sont identiques à celles qui précèdent l'état d'arrivée du lien. Ces lettres constituent une partie commune entre les motifs lus avant et après le lien suffixiel. Ainsi les liens suffixiels permettent d'enchaîner des motifs qui se chevauchent avec une partie commune.

Ceci provient en effet de la propriété fondamentale de la structure d'oracle qui permettra d'assurer la cohérence et la continuité dans la création de phrases musicales (voir partie 3.3.1) :

Pour un oracle construit sur un mot **M**, si **i** est l'état d'arrivée d'un préfixe p_i de **M**, le lien suffixiel **s(i)** pointe sur le plus long suffixe de p_i répété à gauche

Sur la figure 2.1, par exemple, l'état **7** est l'état d'arrivée du préfixe *abcadbc* de l'écorché, et le lien suffixiel **s(7)** pointe sur l'état **3** qui correspond bien à l'arrivée du plus long suffixe répété à gauche de *abcadbc* : *bc*.

Remarque : Les états terminaux différencient l'**oracle des facteurs** et **oracle des suffixes** : dans le premier cas tous les états sont terminaux, dans le second (figure 2.1) les états terminaux sont ceux du chemin suffixiel partant du dernier état.

2.1.2 Algorithme de Crochemore : construction d'un oracle

Les propriétés de l'oracle des facteurs sont assurées par son algorithme de construction dont on peut trouver un exemple en figure 2.2.

Créer un lien suffixiel **s(1)=0** de l'état 1 à l'état 0

Pour un oracle construit jusqu'à l'état **n**, à la lecture d'une nouvelle étiquette **e** :

Créer le nouvel état **n+1** tel que **n+1=t(n,e)**
Remonter le lien suffixiel partant de **n** pour arriver en $\mathbf{n^s}$=**s(n)**
1) *Si* $\boldsymbol{t(n^s,e)}$ *n'existe pas*
Créer la transition $\mathbf{t(n^s,e)}$=**n+1**
1a) *Si* $\boldsymbol{n^s=0}$
Créer un lien suffixiel **s(n+1)=0**.
Etat n+1 construit.
1b) *Si* $\boldsymbol{n^s\neq 0}$
Reprendre en remontant les liens suffixiels : $\mathbf{n^s \leftarrow s(n^s)}$
2) *Si* $\boldsymbol{t(n^s,e)=n^s_{e\rightarrow}}$ *existe*
Créer un lien suffixiel depuis le nouvel état, $\mathbf{s(n+1)=n^s_{e\rightarrow}}$
Etat n+1 construit.

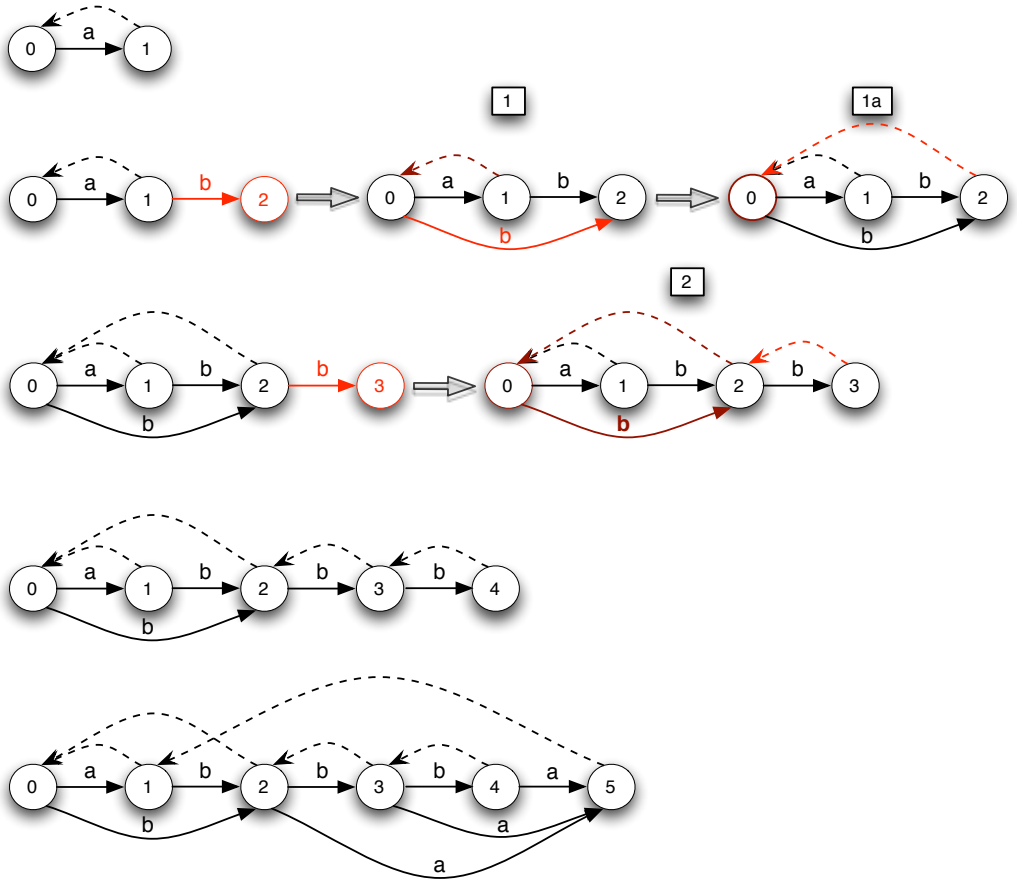


FIGURE 2.2 – Illustration de l'algorithme de Crochemore, construction incrémentale de l'oracle associé à la chaîne *abbba*

2.2 La segmentation par pulsation

L'application de l'oracle vise ici à repérer des motifs dans une séquence musicale, puis dans un second temps à rechercher les occurences des motifs d'une séquences au sein d'une autre séquence de référence (voir partie 3.3.1). Sa construction s'effectue donc non plus sur des caractères mais sur des évènements musicaux codés sous format MIDI dont on introduit ici le formalisme.

2.2.1 Les classes beat et melobeat

Dans un contexte de tempo régulier, les classes `Beat` et `Melobeat` définissent des objets contenant des tranches d'évènements MIDI correspondant à la durée d'une pulsation ou éventuellement d'un nombre entier de pulsations avec un label harmonique associé.

Toutes les séquences MIDI manipulées par le programme, qu'elles proviennent d'une acquisition du jeu d'un musicien sur une entrée ou d'un calcul, sont segmentées par pulsation et annotées par un label d'accord. Un objet issu de la classe `Beat` contient donc le fragment mélodique en lui-même sous la forme d'une suite d'évènements MIDI, le label harmonique qui lui est associée, ainsi entre autre que la durée de la pulsation en ms. `Melobeat` enrichit cette représentation de différentes caractérisations du fragment mélodique comme par exemple la **signature mélodique** listant de manière non ordonnée toutes les hauteurs MIDI présentes dans la pulsation.

2.2.2 Oracles en *mode beat*

Tout comme dans la bibliothèque **OMax 2.0** sur la base de laquelle s'est construit ce projet, ce système utilise des oracles en *mode beat* : chaque état d'un oracle correspond aux évènements se produisant pendant le temps d'une pulsation. Toutes les entrées sont donc fragmentées et labelisées par pulsation ou *beat* afin d'avoir le format adéquat pour leurs traitements qui consistent la majeure partie du temps en la construction d'oracles ou la création d'un chemin pour contraindre un parcours dans un oracle lors d'une phase de génération.

Par l'attention apportée à la généricité de l'implémentation de ses méthodes, la classe `Oracle` peut-être instanciée sur des séquences de natures très diverses allant du texte aux données MIDI (la version actuelle d'OMax, **OMax 4.0** sait tout aussi bien traiter de l'audio que de la vidéo). Les 3 oracles au coeur de ce système, l'**oracle *live* courant**, l'**oracle *d'harmonisation* courant**, et l'**oracle *d'arrangement* courant** (présentés dans les parties 3.1 et 4.1.1) sont construits sur des instances des classes `beat` et `melobeat` décrites en 2.2.1.

Par la suite, on appellera par abus de langage "oracle" une instance issue de la classe `Improvizer`. Celle-ci hérite de la classe `Pythie`, héritant elle-même de la classe `Oracle`.

Classe `Oracle`

La classe `Oracle` correspond à l'implémentation fidèle de la structure théorique de l'oracle définie en 2.1. Elle comprend donc entre autres méthodes la fonction d'apprentissage incrémentale directement traduite de l'algorithme de Crochemore (2.1.2).

Elle contient entre autres les attributs suivants :

– `vectext` : le vecteur des étiquettes de la séquence d'origine dans l'ordre d'apprentissage, l'*écorché* de l'oracle.

– `hashtransition` : la table de transition contenant pour chaque état de la séquence les couples **(clé valeur)** où **valeur** est l'état d'arrivée de la transition partant de l'état et étiquetté par la valeur **clé**.
– `hashsuppl` : la table des liens suffixiels sous la forme de couples **(départ arrivée)**.
– `vectlrs` : donnant les longueurs des plus longs suffixes répétés pour chaque état, le lien suffixiel pointant donc sur l'état réalisant ce plus long suffixe répété.

Classe `Pythie`

La classe `Pythie` apporte une redéfinition des transitions et de leur construction qui est primordiale pour l'utilisation faite de la structure d'oracle dans le processus d'harmonisation par parcours contraint (voir 3.2). Dans un oracle, toutes les transitions arrivant à un état sont nécessairement étiquetées par la même valeur, ainsi dans le cas d'un objet `Pythie` les étiquettes seront sous-entendues puisqu'elles sont déterminées par l'état d'arrivée.

Lors de la construction incrémentale d'un oracle en suivant l'algorithme de Crochemore (2.1.2), on teste en remontant les liens suffixiels l'existence d'une transition portant une étiquette égale à celle de la transition reliant l'état courant et l'état en création (étapes `1)` et `2)`), ce qui revient donc à tester l'égalité entre des états avec le formalisme de `Pythie`. La différence pour la construction et la navigation dans le cas d'un objet `Pythie` repose sur l'introduction d'un **comparateur** pouvant être différent de la relation de stricte égalité entre les deux états. Ainsi, selon le choix de cette fonction de comparaison, la structure de l'automate résultant de la construction sur une même séquence peut être radicalement modifiée comme l'illustre la figure 2.3 comparant l'oracle obtenu par apprentissage de la chaîne "oracle" et l'objet `Pythie` construit sur cette même séquence en ayant choisi une fonction de comparaison qui rend équivalentes deux étiquettes si elles sont toutes les deux consonne ou voyelle.

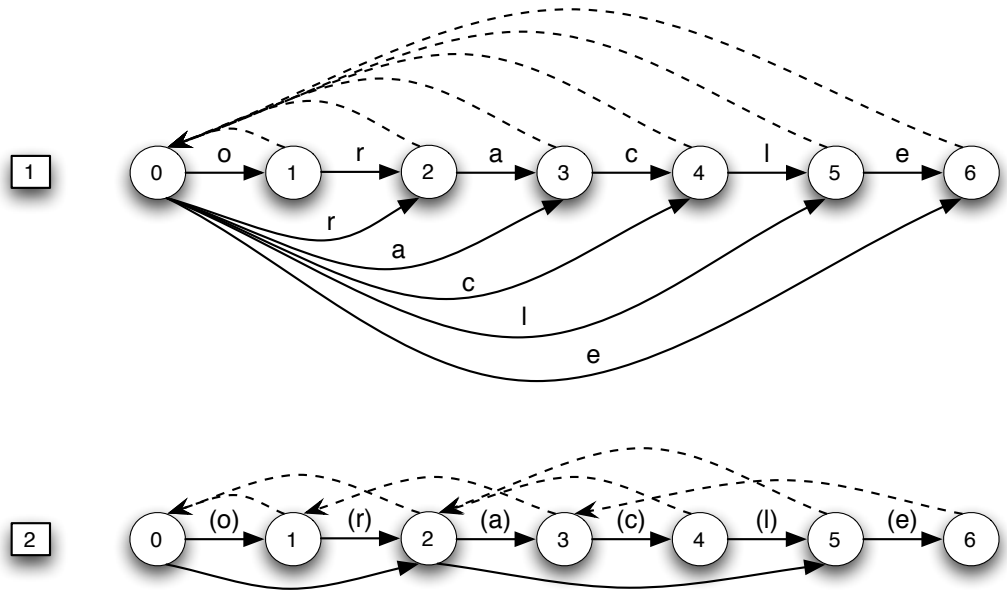


FIGURE 2.3 – Oracle associé au mot "oracle" (1) et Pythie associée avec pour comparateur l'appartenance au groupe des voyelles ou des consonnes (2).

On observe ainsi que des motifs sont repérés entre des états différents mais considérés équivalents alors qu'il n'en était rien dans le cas de la condition d'égalité : cette propriété sera précieuse lors du processus d'harmonisation puisqu'elle permettra d'associer à une entrée une sortie différente mais compatible, apportant ainsi une nouveauté maîtrisée.

Classe `Improvizer`

La classe `Improvizer`, enfin, hérite de la classe `Pythie` en particularisant pour des applications musicales les objets de la séquence apprise lors de la construction : dans le cas du processus d'harmonisation, ceux-ci seront des instances de `Beat` ou de `Melobeat`. Dédiée à la manipulation de séquences musicales, cette classe comporte donc des champs plus spécifiques comme par exemple l'intervalle de **transposition courante** utilisé dans la génération de séquences pour permettre la recherche d'un état à une transposition près.

2.3 Le rôle de la pulsation dans l'acquisition et la restitution musicale

Ce choix d'effectuer l'apprentissage sur les entrées ainsi que la génération puis la restitution de nouvelles phrases (détaillés dans le chapitre 3) en prenant la pulsation comme unité de segmentation est rendu possible par une architecture à plus grande échelle s'inscrivant dans un circuit dont le couple formé par le patch de suivi de partition **Antescofo** conçu et développé à l'IRCAM (Arshia Cont, [13]) et l'objet de suivi de tempo **BeatTracker** développé par Laurent Bonnasse-Gahot ([7]) se trouve à la fois en entrée et en sortie (voir figure 2.4).

Leur association est en effet à l'origine du formatage des entrées MIDI en réalisant en direct la labellisation harmonique de chaque pulsation d'une séquence. D'autre part, chaque phrase calculée par les fonctions de la partie développée en Lisp sous l'environnement Open Music est écrite et sauvegardée selon le formalisme adapté à **Antescofo**, pouvant ainsi être jouée à partir de la position courante de la grille dès leur chargement et ce au tempo variable déterminé par l'objet **BeatTracker**.

2.3.1 Le rôle de l'objet BeatTracker

Le point de départ de ce projet étant de construire un prototype d'instrument pouvant prendre une place à part entière au sein d'une formation musicale, il était primordial que son utilisation n'ampute pas la créativité et l'expressivité en subordonnant les musiciens à un tempo métronomique dicté par ses sorties. L'objet Max/MSP **BeatTracker** a été spécialement développé dans le but d'une intégration avec un logiciel de la famille d'OMax pour atteindre une musicalité plus riche : les utilisations antérieures de l'oracle en mode *beat* dans le cadre de bibliothèques Lisp étaient en effet réalisées avec un tempo fixe imposé par la machine.

Ce patch prend donc en entrée un flux audio ou MIDI, par exemple la section rythmique de la formation, et estime à chaque instant le tempo qui sera ensuite utilisé pour générer et jouer des phrases rythmiquement en place avec l'improvisation en cours. Lors de l'initialisation, l'utilisateur donne une indication de tempo par une battue manuelle servant également à indiquer l'emplacement des pulsations afin d'éviter une synchronisation à contretemps. A la suite de cela, tout au long de la performance, les variations de tempo

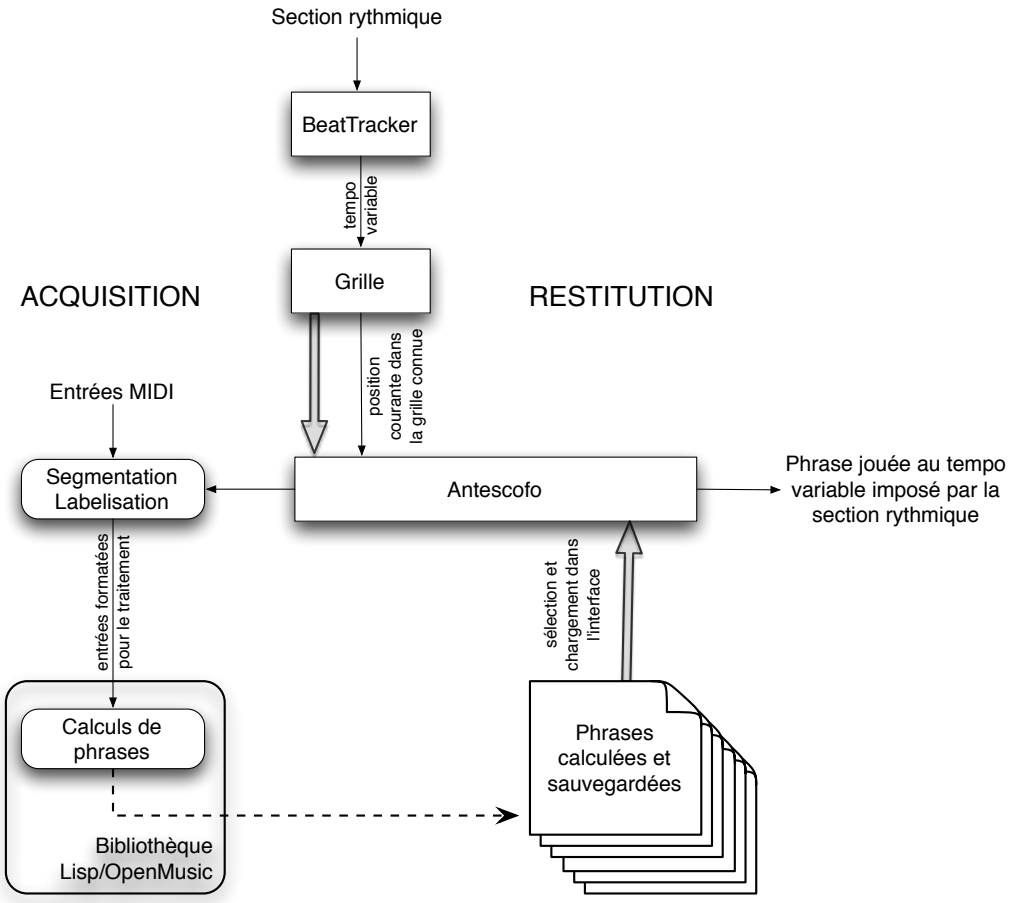


FIGURE 2.4 – L'organisation de l'acquisition et de la restitution autour de la pulsation

sont calculées à chaque instant. (Pour plus de détails sur les algorithmes et l'implémentation, se reporter à [7].) Ces informations servent ensuite d'horloge pour déclencher les différents éléments des séquences chargées dans Antescofo.

2.3.2 L'utilisation d'Antescofo

Antescofo[1] est un système de suivi de partitions polyphoniques et un langage de programmation synchrone pour la composition musicale. Il procède à la reconnaisssance automatique de la position et du tempo dans une partition musicale à partir d'un flux audio provenant d'un ou plusieurs interprètes, permettant ainsi la synchronisation de la performance instrumentale avec des éléments informatiques inscrits dans une partition électronique.

Chaque séquence calculée par les fonctions de la bibliothèque Lisp/OpenMusic et destinée à être jouée (improvisation mélodique harmonisée ou non, arrangement d'une grille,..) est donc écrite et sauvegardée sous le format d'une partition Antescofo. Les phrases ainsi générées viennent enrichir une collection constituée au fil des sessions pour pouvoir être

1. `http://repmus.ircam.fr/antescofo`

chargées par l'utilisateur au cours de l'improvisation.

Dans notre utilisation d'Antescofo, ce sont les pulsations fournies par l'objet BeatTracker qui tiennent lieu de "notes" dans la partition qui viennent déclencher les évènements que sont l'ordre de jouer les tranches MIDI contenues entre deux pulsations. La notation temporelle dans une partition Antescofo pouvant s'effectuer en temps relatif, il est donc possible de faire jouer à un tempo donné une phrase générée à partir d'un matériau de base dont l'acquisition avait été faite à un tempo différent.

2.3.3 La labellisation des entrées avec les accords de la grille

Au début de chaque session d'improvisation, on se place dans le contexte d'une grille harmonique connue. Celle-ci est écrite sous la forme d'une suite de labels d'accords correspondant chacun à une pulsation et est sauvegardée sous le format Antescofo avec un encodage particulier : on utilise le canal MIDI numéro 16 pour représenter la grille avec des conventions permettant d'exprimer la fondamentale de l'accord et sa nature (5 labels d'accords sont actuellement reconnus : **maj7**, **m7**, **7**, **m7b5**, et **dim**). Lors de l'acquisition des entrées, ses informations sont envoyées par Antescofo et figurent donc dans les buffers MIDI dans lesquels viennent lire les fonctions de la bibliothèque OpenMusic. Ces annotations permettent ainsi de segmenter les séquences MIDI par pulsation et d'attribuer à chacune d'entre elles un label harmonique dans l'optique de la construction d'oracles en *mode beat*.

3. L'harmonisation et l'arrangement par parcours contraints dans les oracles appris du corpus

Ce chapitre détaille le module d'harmonisation et d'arrangement prenant en entrée les séquences MIDI annotées obtenues par le procédé d'acquisition exposé dans la partie précédente. Celui-ci se base donc sur un corpus qu'il utilise parallèlement comme terrain d'apprentissage pour en extraire des mécanismes d'harmonisation empiriques, et comme mémoire musicale dans laquelle il va puiser les fragments d'accompagnement une fois l'harmonisation effectuée.

L'intégralité du processus est ici décrite, depuis la création des couples d'oracles issus de l'apprentissage sur des séquences du corpus, jusqu'à leurs parcours en cascade pour calculer l'accompagnement d'une mélodie.

3.1 Tirer un double enseignement du corpus

Si le matériau de base des accompagnements proposés par le système provient bien d'une mémoire musicale, il tient cependant des deux sous-familles présentées en 1 par l'utilisation faite du corpus. En effet, les fragments constituant l'accompagnement généré sont bien des citations provenant de la mémoire musicale parfois à une transposition près (comme dans [20, 25],...), mais ils ne sont pas directement indexés par la mélodie qu'ils supportent et immédiatement recherchés par un calcul de similarité comme dans les travaux cités : il ne sont mis à contribution qu'après une première phase d'harmonisation symbolique effectuée à l'aide d'un modèle créé de l'observation du corpus.

Le processus d'harmonisation et d'arrangement d'une mélodie donnée en entrée est scindé en deux étapes. Le dispositif présenté ici combine ainsi l'utilisation du corpus pour l'extraction de règles empiriques, ainsi que comme répertoire de citations musicales au sein duquel piocher. Ces deux étapes en cascade lors de la création de l'accompagnement d'une mélodie reposent sur des parcours successifs d'un **oracle d'harmonisation** puis d'un **oracle d'arrangement** dont l'apprentissage a été effectué sur un corpus de morceaux présentant une piste mélodique, par exemple un thème puis des chorus, ainsi qu'une piste d'accompagnement pouvant être de n'importe quelle nature : accords plaqués, basse, accompagnement complexe... Par la suite, on entendra par le mot "**corpus**" non pas les morceaux utilisés pour l'apprentissage, mais l'ensemble des couples d'oracles issus de l'apprentissage.

3.1.1 L'oracle d'harmonisation

L'oracle d'harmonisation est l'outil de l'**harmonisation symbolique** de la mélodie à traiter. Chaque état de cet oracle est une instance de la classe `MeloBeat` comportant donc entre autres parmi ses champs une signature mélodique (`MeloSignature`) (réduction d'un fragment de mélodie à une séquence de hauteurs non ordonnées et sans prendre en compte leurs durées) et un label harmonique (`HarmLabel`) correspondant respectivement à la segmentation par pulsation et à l'annotation par des labels d'accords de la piste mélodique du morceau ayant servi à sa construction. L'étape d'harmonisation consiste donc à associer à une entrée mélodique une suite de labels d'accords qui servira lors de la génération de l'accompagnement concret.

3.1.2 L'oracle d'arrangement

Ce second oracle est quant à lui construit sur des objets provenant de la classe `Beat` et est appelé pour effectuer le passage d'une suite de symboles harmoniques que sont des labels d'accords à la réalisation à proprement parler et réalise donc ainsi l'étape d'arrangement. Le but poursuivi en utilisant le mécanisme de l'oracle pour l'étape d'harmonisation symbolique est d'obtenir des progressions harmoniques cohérentes, dans le cas de la phase d'arrangement on cherche à bénéficier des propriétés de continuité pour que la réalisation, c'est-à-dire l'accompagnement concrètement joué, propose une reconstruction homogène à partir des différentes pulsations du corpus assemblées pour créer le nouvel accompagnement. On s'attend donc par exemple à voir les propriétés de continuité se répercuter sur les lignes de basses dans les accompagnements complexes et ainsi éviter des phrasés trop accidentés.

3.2 L'apprentissage du corpus d'oracles

Ces deux catégories d'oracles sont obtenues par un apprentissage sur des séquences musicales annotées en suivant l'algorithme de Crochemore (2.1.2) : un oracle d'harmonisation est construit à partir d'une séquence mélodique, et un oracle d'arrangement sur une séquence d'accompagnement. S'ils peuvent êtres créés à partir de telles séquences isolées, la recherche de la richesse musicale conduit plutôt à utiliser des morceaux comportant une mélodie et un accompagnement volontairement associés. La bibliothèque OpenMusic comprend donc un module d'apprentissage à partir de morceaux annotés, et sépare la mélodie de l'accompagnement pour permettre respectivement la construction d'un oracle d'harmonisation et d'un oracle d'arrangement.

Si les constructions de ces deux types d'oracles suivent le même schéma, elles diffèrent cependant sur les labels utilisés pour l'apprentissage et les fonctions de comparaison (introduites en 2.2.2) qui leurs sont associées.

Les labels harmoniques avec critère d'égalité sont utilisés pour la construction d'un **oracle d'arrangement**. Ainsi, lors de l'apprentissage, deux tranches d'accompagnement différentes seront considérées "équivalentes" si elles sont annotées par le même label harmonique, et ce même si les contenus de leurs séquences sont différents.

Les étiquettes utilisées pour l'apprentissage d'un **oracle d'harmonisation** sont les **signatures mélodiques** des fragments de mélodies. Ces champs contenant la liste des notes présentes dans une tranche "`MeloBeat`" sont comparés avec un **critère d'inclusion modulo l'octave** : deux fragments mélodiques différents seront considérés "équivalents"

si toutes les notes présentes dans l'un sont présentes dans l'autre (qui peut en comprendre plus) sans tenir compte de l'octave à laquelle elles se trouvent.

Les oracles ainsi construits permettent donc d'avoir accès à toutes les répétitions de motifs de "classes d'équivalences" des séquences d'apprentissage pour pouvoir ensuite, lors de l'étape de génération d'un accompagnement pour une mélodie donnée en entrée, effectuer la recherche de motifs répétés communs à deux séquences : celle de l'oracle utilisé, et celle de la phrase musicale à accompagner.

3.3 Harmonisation et arrangement "en cascade"

3.3.1 Le parcours contraint d'un oracle

Tout comme *Band out of a box* ([30, 31]), on peut considérer l'utilisation principale de la version actuelle du logiciel OMax comme un dispositif d'interaction avec un ou plusieurs solistes virtuels : des oracles sont construits au fur et à mesure du jeu d'un ou de plusieurs instrumentistes sur des critères de hauteur ou de contenu spectral pour pouvoir ensuite proposer des improvisations innovantes dans le style du matériau ayant suivi à l'apprentissage. Pour ce faire, l'utilisateur d'OMax manipule en temps réel plusieurs paramètres pour contrôler la navigation dans l'oracle (en délimitant des régions, changeant de descripteurs, modifiant le contexte et la continuité souhaités...) et pour transformer le résultat (accélération, ralentissement, transposition,...). Dans le cadre fixé par l'utilisateur du logiciel, la navigation dans l'oracle correspond à un **parcours libre** : le saut d'un état à un autre en suivant les liens de l'oracle s'effectue de manière libre en étant uniquement déterminé par le réglagle de paramètres comme la continuité et le contexte.

Dans le cadre du processus d'harmonisation et d'arrangement d'une mélodie, notre utilisation de la structure d'oracle est différente puisqu'elle repose sur un **parcours contraint**. Un champ de l'entité à laquelle appartiennent les différents états de l'oracle est alors considéré comme un **label** ou une **étiquette** caractérisant un état. Un parcours contraint sur l'oracle revient à chercher dans la séquence d'origine des successions de labels identiques à celles que comporte une nouvelle séquence de labels donnée en entrée, c'est-à-dire procéder à un **filtrage par motif**.

On peut donc par exemple créer une improvisation mélodique à partir de l'apprentissage d'un chorus annoté par des labels harmoniques en effectuant un parcours contraint par une nouvelle grille harmonique donnée en entrée sur l'oracle. Il s'agit donc de suivre un chemin dans l'automate avec des transitions étiquetées par les labels de la grille d'entrée si cela est possible. Quand on ne trouve aucune transition indexée par le label souhaité, on cherche à suivre un lien suffixiel permettant d'aller à un autre état ou l'on pourra éventuellement lire l'étiquette courante (les liens suffixiels garantissent que dans la séquence d'origine, il existe une partie commune entre les deux fragments concaténés), ou bien enfin on cherche cette étiquette dans un état quelconque de l'automate indépendamment du chemin parcouru dans l'oracle jusqu'alors. La navigation recherche donc tout d'abord la **continuité** en essayant de coller aux enchaînements appris, et cherche les labels indépendemment s'ils s'inscrivent dans une succession ne figurant à aucun endroit de l'oracle construit sur la séquence d'origine.

3.3.2 Les parcours successifs des deux oracles

Comme l'illustre la figure 3.1, la création d'un nouvel accompagnement pour une mélodie donnée en entrée fait intervernir des parcours contraints de l'oracle d'harmonisation et de l'oracle d'arrangement successivement.

La phrase à harmoniser est traduite sous la forme d'une suite de `Melobeats` pour obtenir la signature mélodique de chaque pulsation. La séquence ainsi obtenue sert de chemin pour guider le parcours dans l'oracle d'harmonisation. En fonction de la continuité maximale exigée, la recherche s'efforce donc de trouver les plus longs motifs consécutifs possibles dans l'oracle.

On extrait du chemin parcouru dans l'oracle d'harmonisation les labels harmoniques des différents états traversés pour contraindre cette fois le parcours dans l'oracle d'arrangement. L'extraction du contenu MIDI des différents états par lesquels est passée la navigation dans l'oracle d'arrangement fournit enfin l'accompagnement à associer à la mélodie d'entrée.

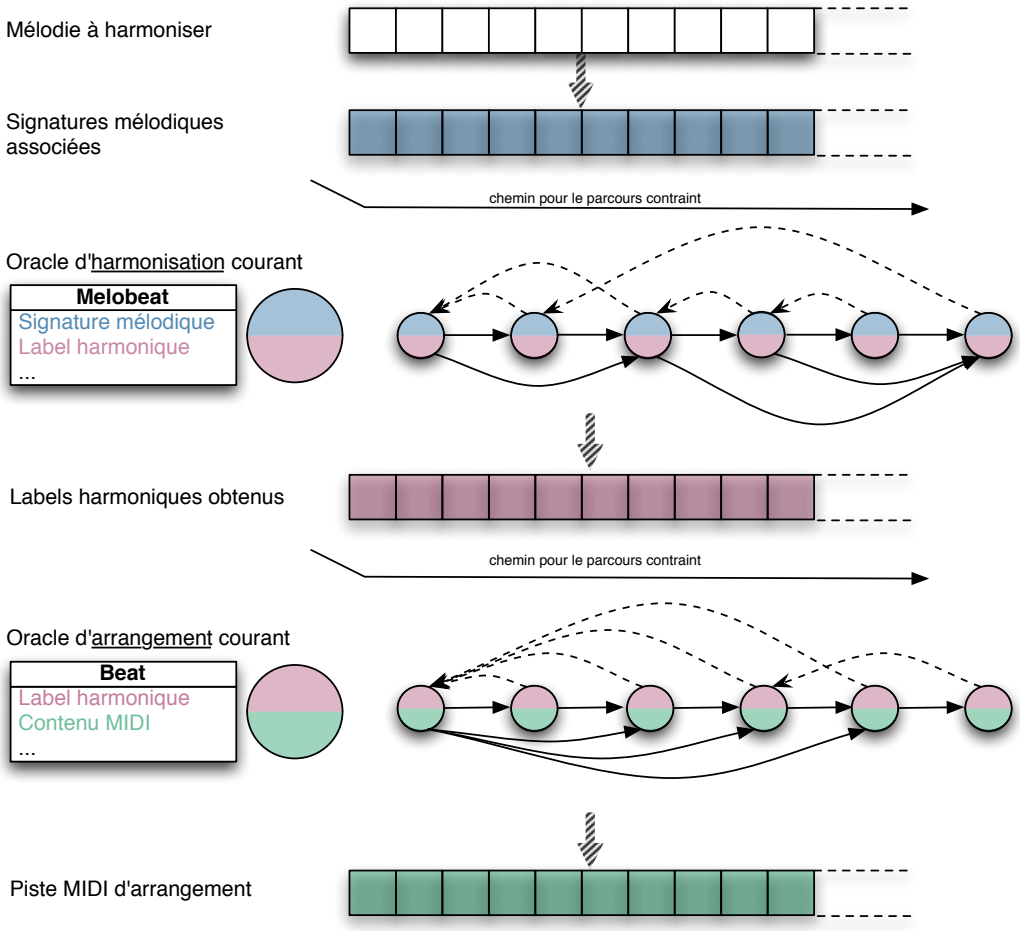


FIGURE 3.1 – Le parcours des 2 oracles courants dans la génération d'improvisations harmonisées

Le résultat obtenu dépend fortement de la **continuité maximale** imposée à la navigation. Le paragraphe suivant expose plus en détail le rôle joué par ce paramètre dans le

parcours d'un oracle.

3.3.3 La fonction de navigation

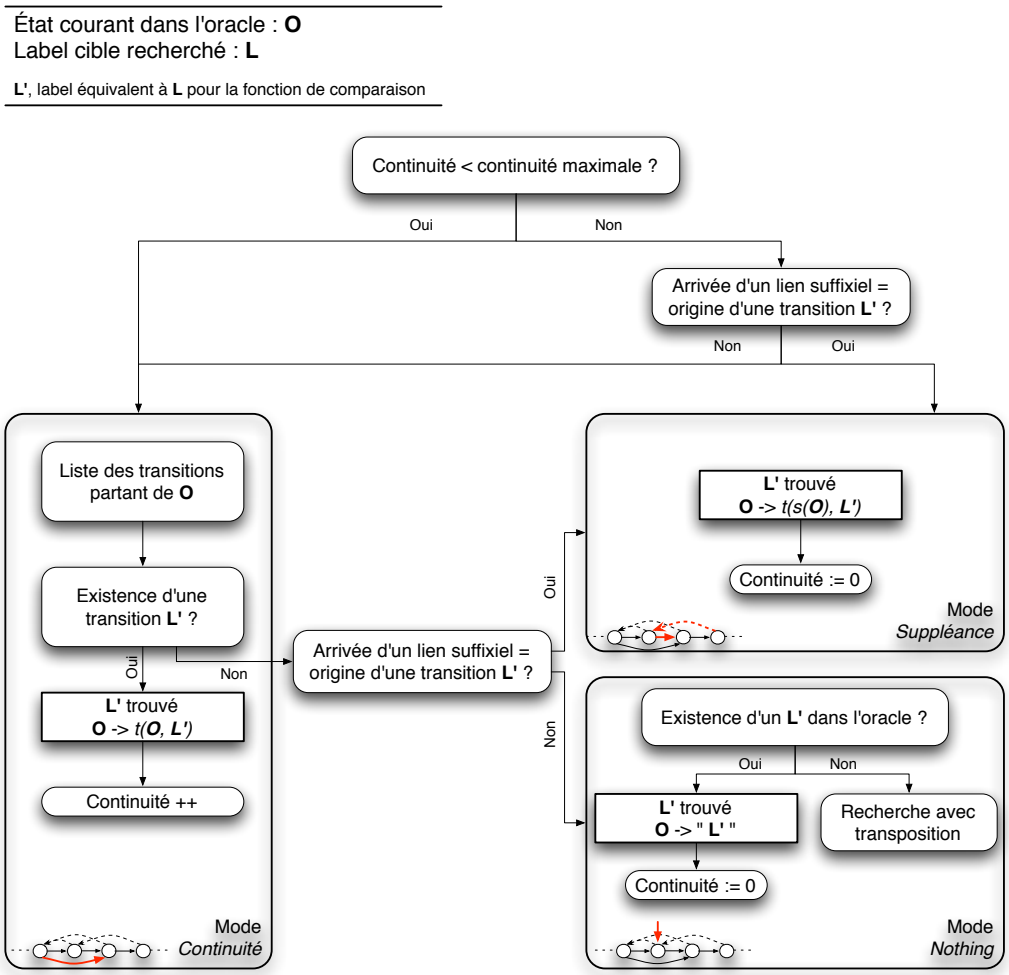


FIGURE 3.2 – La navigation dans un oracle en suivant un parcours contraint

La navigation

À chaque étape, on teste la **continuité** du calcul dans l'oracle, c'est-à-dire le nombre de transitions successives autorisées (voir figure 3.2). Ce contrôle limite la longueur des segments recopiés dans la séquence d'origine. Si la continuité est inférieure à une valeur maximale (contenue dans un champ de la classe `Improvizer`) ou s'il n'y a pas de lien suffixiel à partir de l'état obtenu, on se met en mode *continuité* en augmentant la continuité de 1, sinon en mode *suppléance* en réinitialisant la continuité à 0.

En mode *continuité* : on choisit une transition avec la bonne étiquette s'il en existe et on avance en sélectionnant l'état correspondant, sinon on passe en mode *suppléance* s'il y a des liens suffixiels ou en mode *nothing* s'il n'y en a pas.

En mode *suppléance* : on suit un lien suffixiel s'il y en a (en choisissant le plus long suffixe répété) et on avance en sélectionnant le l'état trouvé si c'est possible, sinon on passe en mode *nothing*.

En mode *nothing* : on cherche le label courant indépendamment de ce qui précède et on avance si on le trouve en sélectionnant l'état correspondant, si dans ce dernier cas l'état n'est pas trouvé, celui-ci sera vide dans la séquence générée.

La structure d'oracle (plus précisemment d'`Improvizer`) et la fonction de navigation étant génériques, cet algorithme s'applique quelle que soit la nature des objets sur lesquels l'oracle a été construit. Dans le cas de l'oracle d'harmonisation, un chemin de signatures mélodiques est donné pour être comparé avec le champ `MeloSignature` des états de l'oracle, avec un critère d'inclusion modulo l'octave comme dans la phase de construction (ce qui n'était a priori pas nécessairement le cas). Dans le cas de l'oracle d'arrangement, une séquence de labels d'accords constitue le chemin à parcourir dans l'oracle.

On observe bien ici les avantages de l'utilisation de la structure d'oracle pour cette application en terme d'homogénéité et d'innovation : la possibilité de suivre des liens suffixiels permet de multiplier les possibiltés de retourner une séquence cohérente au regard de celle sur laquelle est construit l'oracle.

Transposition

Au cours de la navigation dans l'oracle, il arrive qu'on cherche le label courant indépendamment de la continuité de ce qu'on a lu précédemment (mode *nothing*, voir figure 3.2). Dans cette situation, on peut se permettre de faire une transposition qui augmente le nombre de possibilités de l'oracle. La recherche du label se fait non plus seulement à l'unisson, mais aussi à une distance maximale d'une tierce mineure au-dessus ou au-dessous de la grille donnée.

L'oracle d'harmonisation et l'oracle d'arrangement peuvent tous deux avoir recours à la transposition lors d'un parcours. Dans le premier cas, si une signature mélodique n'est pas trouvée dans l'oracle, la transposition revient simplement à faire la recherche sur une signature mélodique dont les hauteurs MIDI ont toutes été augmentées ou diminuées d'une même valeur. L'intégralité des champs sensibles à la transposition seront modifiés dans l'état trouvé afin d'obtenir le bon label harmonique pour l'étape suivante.

De la même manière, si un label d'accord n'est pas trouvé dans l'oracle lors de la phase d'arrangement, le label est transposé en modifiant la fondamentale et les autres champs dont le contenu MIDI seront eux aussi modifiés en conséquence. (Le processus de transposition est détaillé pour la classe `Beat` dans [9].)

3.4 Jouer avec le corpus

3.4.1 Le corpus disponible et son enrichissement

Le corpus d'oracles peut être enrichi au cours de la performance en appelant les fonctions d'apprentissage décrites en 3.2 sur des séquences venant d'être enregistrées. Ce procédé est d'ailleurs le seul moyen de création de corpus : il n'y a pas lieu ici de distinguer un apprentissage *online* et *offline* comme dans certains systèmes comme *Band Out of a Box*, le corpus d'oracles disponible au lancement de l'application est en fait constitué des éléments de corpus créés lors des sessions précédentes qui ont été conservés. Il n'y a donc

aucune différence conceptuelle ou d'utilisation entre ces derniers et ceux qui sont ajoutés pendant la session.

A l'heure actuelle, le corpus disponible est constitué d'oracles construits sur

- 4 morceaux enregistrés par Bernard Lubat : 2 de ses "Chansons enjazzées" *D'ici d'en bas* et *J'aime pour la vie*, ainsi que les standards *Goodbye Pork Pie Hat* et *All the things you are*,
- les standards du Realbook *Alice in Wonderland*, *Au Privave*, *Blues for Alice*, *Stolen moments*, *Song for my father*, *It don't mean a thing*,
- et de manière plus isolée : un extrait du *Calendário do Som* d'Hermeto Pascoal et *Le mauvais sujet repenti* de Georges Brassens.

Les couples d'oracles issus des morceaux cités peuvent être utilisés de manière autonome ou concaténés afin d'effectuer des parcours retournant des séquences puisant des éléments dans plusieurs d'entre eux.

3.4.2 Les oracles *courants*

Par l'intermédiaire de l'interface, l'utilisateur peut sélectionner et modifier à chaque instant les oracles *courants*, c'est-à-dire les éléments de corpus qui seront utilisés lors des prochaines générations dont il lancera l'exécution.

La séparation entre les étapes d'harmonisation symbolique et d'arrangement ainsi que l'introduction d'une grammaire de labels entre l'oracle d'harmonisation et l'oracle d'arrangement permet d'augmenter considérablement les possibiltés par rapport aux architectures dans lesquelles chaque fragment d'accompagnement est directement indexé par le fragment mélodique associé. En effet, on peut choisir comme oracles courants des objets construits sur des morceaux différents, et ainsi, par exemple, faire jouer à Georges Brassens une improvisation sur un thème de Bill Evans ayant été harmonisée par Bernard Lubat...

4. Vers un instrument interactif

Le module d'harmonisation présenté dans le chapitre précédent peut fonctionner de manière autonome et être utilisé à partir de l'interface : une fois l'oracle d'harmonisation et l'oracle d'arrangement choisis, il peut calculer et sauvegarder sous la forme d'un fichier MIDI une version harmonisée d'une séquence captée par le système. Mais il a surtout été conçu pour s'intègrer dans un cadre interactif plus étendu dédié à l'improvisation.

En effet, acceptant en entrée des phrases musicales annotées par pulsation, il peut aussi bien être appliqué à des mélodies provenant du jeu d'un musicien qu'à des phrases calculées respectant le même format. Ainsi, des improvisations mélodiques créées par le sysème peuvent elles aussi être harmonisées et arrangées pour faire du système une force de proposition musicale dans le cadre de sessions d'improvisations au sein d'une formation.

Dans cette optique, de nombreux paramètres de la génération et de la navigation dans les oracles sont laissés à la discretion de l'utilisateur qui peut les modifier simplement au cours de la performance grâce à une interface graphique pour façonner ses propres improvisations. Cette faculté associée à des possibilités d'arrangement de la grille en temps réel peut donc tour à tour faire de l'interprète informatique un accompagnateur ou un soliste, produisant des phrases musicales plus ou moins complexes, structurées, et extravagantes selon sa sensibilité musicale et ses choix en réaction à la situation.

4.1 Les principales fonctions accessibles à l'utilisateur

4.1.1 Harmonisation et création d'improvisations harmonisées

Les entrées MIDI du système sont enregistrées dans des buffers et annotées harmoniquement tout au long de la performance (partie 2.3.3). L'**oracle *live*** associé au morceau en cours est destiné à effectuer son apprentissage sur le contenu de ses buffers. Le contrôle de cet apprentissage est laissé à l'utilisateur qui décide en direct des phrases musicales qu'il veut envoyer à l'oracle pour l'apprentissage. Celui-ci pouvant être remis à zéro à tout moment et acceptant également d'être rempli par des mélodies extérieures à la session en cours en faisant glisser des fichiers MIDI dans l'interface, il peut contenir à chaque instant une très maigre quantité de connaissances provenant uniquement des phrases jouées dans les secondes qui ont précédé tout comme il peut être constitué d'un important matériau hybride associant par exemple les phrases jouées par un musicien au fil de la performance avec des thèmes et des chorus provenant de sessions antérieures ou de standards.

Comme le schématise la figure 4.1, un parcours de l'oracle *live* contraint par les labels d'accords du morceau en cours (dont l'exécution peut être aisément commandée depuis l'interface) produira une **nouvelle "improvisation" mélodique** compatible avec la grille par construction, et qui pourra être harmonisée et accompagnée selon le processus vu en

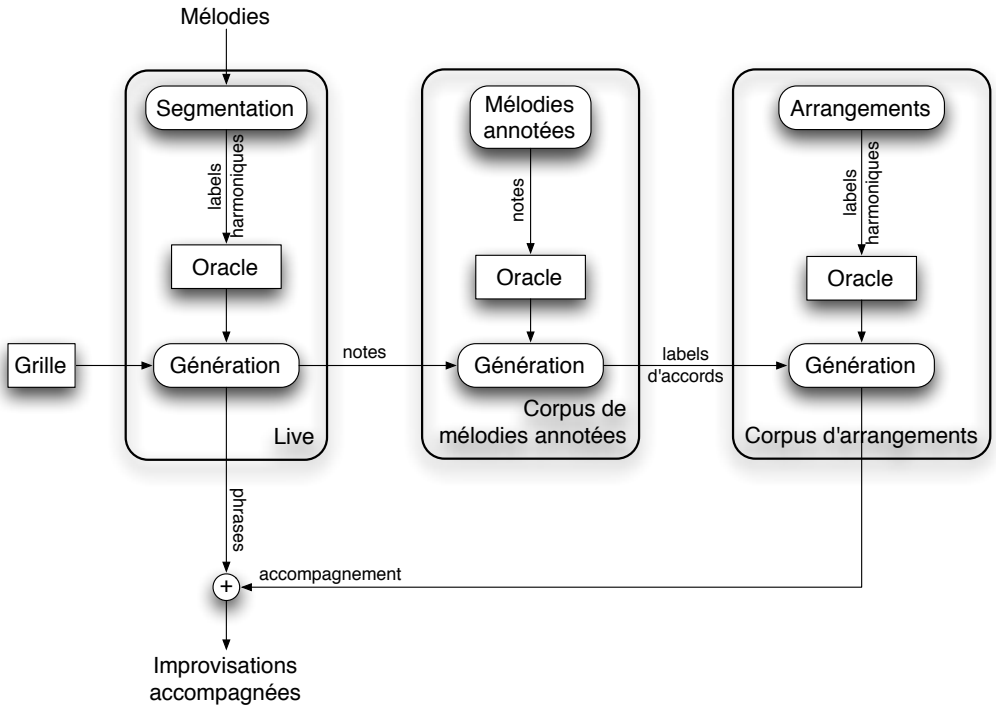


FIGURE 4.1 – Génération d'improvisations accompagnées à partir de l' *oracle live*

partie 3.3.2, permettant ainsi à l'interprète informatique de "prendre un solo" tout en l'accompagnant.

4.1.2 Arrangement en temps réel

Une fois la grille du morceau chargée, l'utilisateur peut choisir parmi deux méthodes d'arrangement de la grille en temps réel entre lesquelles il peut naviguer au fil même de la performance tout en en modifiant les paramètres à chaque instant. Il s'agit bien ici d'arrangement de la grille indépendamment de l'entrée provenant de l'instrument soliste : contrairement aux possibilités de génération d'improvisations harmonisées décrites dans le paragraphe suivant, la sortie consiste en un unique accompagnement.

Arrangements produits par un parcours de l'Oracle *live* Cette méthode représentée dans la partie droite de la figure 4.2 tire simplement profit de l'**Oracle *live***. En effet, celui-ci se construit à la demande de l'utilisateur au fur et à mesure de la performance pour pourvoir calculer des improvisations grâce à des parcours contraints par la grille. De la même manière qu'il permet de créer des nouvelles mélodies lorsque sa construction s'effectue sur une entrée mélodique, la redirection de son entrée vers un instrument jouant un rôle d'accompagnateur (entrée "Accompagnement" sur la figure 4.2) fournit le matériau pour la génération de nouveaux accompagnements. On peut en effet au cours de la performance vider la mémoire de l'**Oracle *live*** qui faisait jusqu'alors son apprentissage sur un instrument soliste pour le rediriger vers un instrument accompagnateur. Ainsi après un temps suffisant d'apprentissage, un parcours de l'oracle avec pour chemin d'entrée les labels d'accords de la grille du morceau créera et jouera un nouvel accompagnement adapté

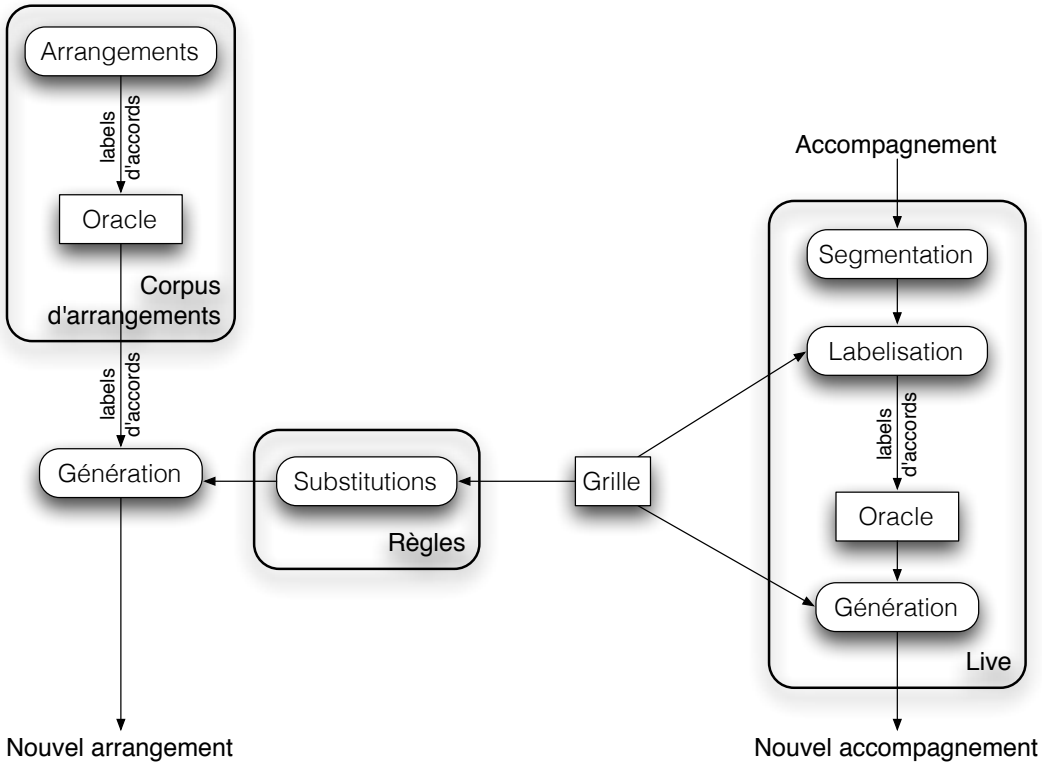


FIGURE 4.2 – Architecture des fonctions d'arrangement

à la grille par construction. De plus, chaque phrase calculée étant sauvegardée dans le format de lecture adéquat pour *Antescofo* pour pouvoir être lancée après son calcul (y compris lors d'une toute autre session), l'accompagnement ainsi créé pourra selon le choix de l'utilisateur provenir d'un calcul sur des évènements venant juste de se produire comme sur des phrases jouées antérieurement.

Arrangements produits par l'application de règles de substitutions et réalisés par l'Oracle d'arrangement Ce deuxième dispositif (partie gauche de la figure 4.2) est le seul de tout l'environnement à faire intervenir des règles harmoniques formalisées. Dans l'état actuel, celle-ci correspondent à la grammaire de Steedman introduite par Marc Chemillier dans l'environnement *OMax 2.0* [8].

Le calcul de grilles obtenues par différentes applications successives de cette grammaire à la grille du morceau courant peut être lancée depuis l'interface Max/MSP afin d'obtenir de nouvelles grilles symboliques consistant en une suite de labels d'accords théoriquement compatibles avec la grille originale. Ces différentes séquences de labels d'accords servent ensuite - tout comme celles obtenues par le passage dans le premier oracle dans le processus d'harmonisation (3.3.2) - de chemin à parcourir dans l'**Oracle d'arrangement** courant pour obtenir une réalisation de ces grilles et générer ainsi un nouvel accompagnement.

4.2 L'interface

La communication via le protocole OSC entre le coeur de calcul développé dans l'environnement OpenMusic et le patch Max/MSP où se situent les entrées et sorties MIDI du système permet de piloter tous les paramètres de la génération et de l'apprentissage depuis une interface faisant partie du patch Max/MSP. C'est à travers celle-ci que l'utilisateur :

- choisit une des grilles disponibles selon le morceau qui sera joué,
- initialise l'objet **BeatTracker** avec une battue manuelle,
- synchronise **Antescofo** sur la grille du morceau en cliquant sur une case de la représentation de la grille apparaissant dans l'interface,
- gère en temps réel la construction de l'oracle *live* et la génération de phrases d'accompagnement et d'improvisations harmonisées ou non (voir figure 4.3),
- lance le jeu de phrases calculées dernièrement ou au cours de sessions précédentes.

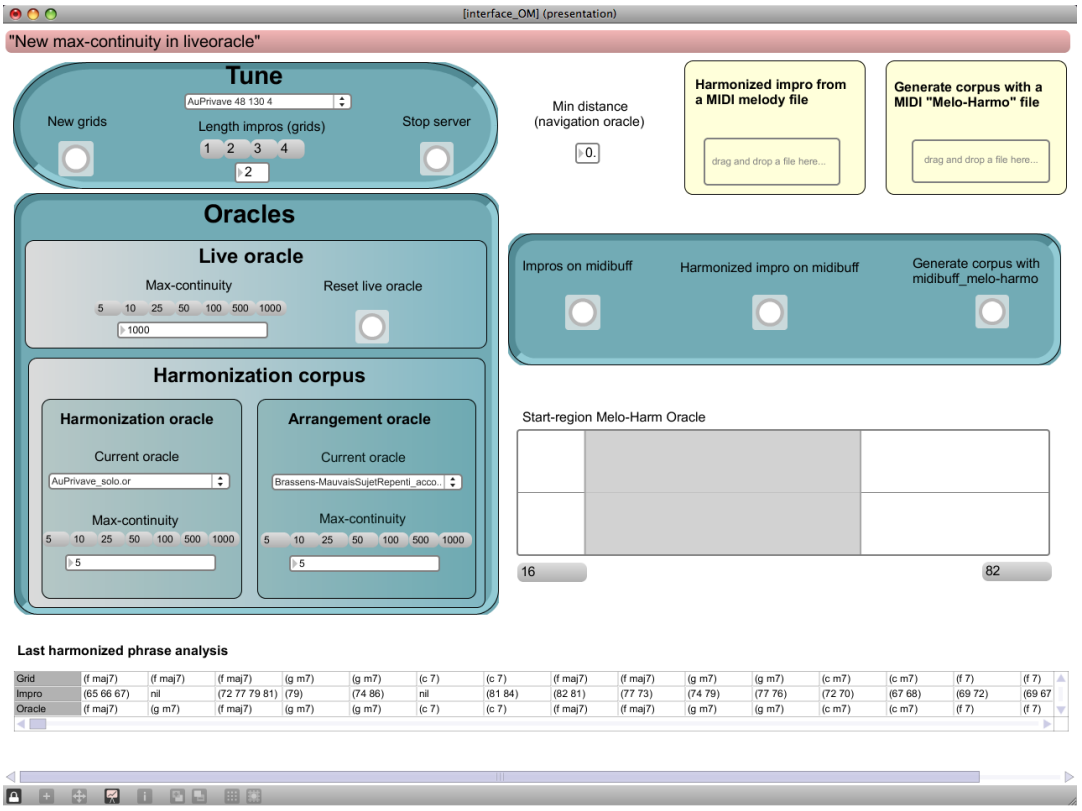


FIGURE 4.3 – Version actuelle de la fenêtre de l'interface pilotant la génération de phrases musicales

L'utilisateur a aussi bien accès aux phrases calculées à partir d'évènements venant juste de se produire qu'à celles issues de sessions antérieures. Il lance simplement ces phrases à l'aide de touches spécifiques du clavier correspondant aux différentes catégories d'éléments : arrangements et accompagnements générés par calcul à partir de la grille en cours, improvisations mélodiques ou d'accompagnement, improvisations mélodiques harmonisées, ou enfin la grille elle-même.

Grâce au format d'écriture des partitions Antescofo et à la connaissance de la position courante dans la grille, chaque phrase lancée se met immédiatement à jouer à partir de cette position. Il est donc ainsi possible de changer très fréquemment la phrase en cours de jeu pendant un seul et même passage de la grille et par exemple de naviguer entre plusieurs arrangements différents d'une même improvisation pendant que celle-ci est jouée.

4.3 Jouer avec les paramètres de l'oracle

La figure 4.3 présente la version actuelle de la partie de l'interface permettant de gérer l'apprentissage et la génération de phrases à partir des oracles courants. Ceux-ci peuvent donc être changés à chaque instant pour créer de nouvelles improvisations harmonisées. De plus, la liste des oracles disponibles peut-être enrichie au fil de la performance par de nouveaux oracles construits par l'apprentissage de l'association entre mélodie et accompagnement dans les phrases venant d'être jouées par les musiciens et retenues par l'utilisateur. Ces nouveaux éléments de corpus viendront immédiatement s'ajouter à ceux déjà existants pour pouvoir être directement utilisés. Il est donc ainsi possible de créer des improvisations harmonisées dans la dynamique de ce que vient de jouer le groupe.

Les contrôles disponibles ne se limitent pas au seul choix des oracles d'harmonisation et d'arrangement : en effet pour chacun des trois oracles en jeu on peut définir la continuité maximum à imposer lors de la navigation par parcours contraint (paragraphe 3.3.3). Cette caractéristique a une influence capitale sur l'improvisation produite puisqu'elle quantifie le degré de recopie du matériau ayant servi à l'apprentissage et avec cela le degré d'innovation apporté, ainsi que l'homogénéité et la cohérence des progressions harmonique et du phrasé avec lesquelles elles sont rendues. Enfin, en plus des évènements enregistrés sur les entrées, il est possible de faire glisser des fichiers MIDI étrangers à la session en cours dans l'interface, et ainsi d'injecter par exemple le chorus d'un morceau différent dans l'oracle *live* pour enrichir les prochaines improvisations.

4.4 Evaluer "l'audace" des improvisations crées

Les paramètres de la génération d'improvisations harmonisées tout comme les éléments à faire intervenir dans la création des différents oracles étant contrôlables, la possibilité est laissée de tenter de créer des phrases plus ou moins innovantes voire extravagantes (dans l'absolu ou relativement au déroulement de l'improvisation en cours).

Par exemple, l'introduction dans l'oracle *live* d'une mélodie basée sur une grille harmonique très éloignée de la grille en cours donnera naissance à une improvisation radicalement différente de celles obtenues à partir de melodies construites sur la grille du morceau en cours. De la même manière, le jeu de combinaisons possibles des continuités maximales des 3 oracles impliqués dans la création de nouvelles phrases laisse le champ libre à une multitude de possibilités.

S'il est impossible dans l'état actuel d'évaluer avant l'écoute la nature du rendu de l'étape d'arrangement, il est possible de se faire une idée du résultat de l'étape d'harmonisation grâce à l'affichage dans l'interface de la comparaison entre les labels harmoniques de la grille en cours et ceux de la dernière improvisation accompagnée produite.

La figure 4.4 montre un exemple de résultats obtenus dans deux cas très différents : la même improvisation mélodique générée à partir d'un chorus d'*Au Privave* a été successivement harmonisée par l'oracle d'harmonisation issu de l'apprentissage de ce même morceau

avec une grande continuité, puis avec l'oracle construit sur *J'aime pour la vie* de Bernard Lubat, qui a la particularité de ne comporter que des accords **7**, et avec une continuité très faible.

Grid	(f maj7)	(f maj7)	(f maj7)	(g m7)	(g m7)	(c 7)	(c 7)	(f maj7)	(f maj7)	(g m7)	(g m7)	(c m7)	(c m7)	(f 7)	(f 7)
Impro	(65 66 67)	nil	(72 77 79 81)	(79)	(74 86)	nil	(81 84)	(82 81)	(77 73)	(74 79)	(77 76)	(72 70)	(67 68)	(69 72)	(69 67)
Oracle	(f maj7)	(g m7)	(f maj7)	(g m7)	(g m7)	(c 7)	(c 7)	(f maj7)	(f maj7)	(g m7)	(g m7)	(c m7)	(c m7)	(f 7)	(f 7)
(bb 7)	(bb 7)	(bb 7)	(bb 7)	(bb m7)	(bb m7)	(eb 7)	(eb 7)	(f maj7)	(f maj7)	(g m7)	(g m7)	(a m7)	(a m7)	(d 7)	(d 7)
(74)	(72 68 65)	(74 72 68)	(65 74 72)	(68 65 74)	(73 68 65)	(72 70 67)	(65 63 62)	(60 62 63)	(65 66 67)	(69 70 72 73)	(75 72 73 75	(79 81 83)	(84 81)	(78 74)	nil
(bb 7)	(bb 7)	(bb 7)	(bb 7)	(bb m7)	(bb m7)	(eb 7)	(eb 7)	(f maj7)	(f maj7)	(g m7)	(g m7)	(a m7)	(a m7)	(d 7)	(d 7)
(g m7)	(g m7)	(g m7)	(g m7)	(g m7)	(g m7)	(c 7)	(c 7)	(f maj7)	(f maj7)	(d 7)	(d 7)	(g m7)	(g m7)	(c 7)	(c 7)
(77)	(75)	(74 72)	(70 68)	(69 72)	(70 67)	(72 70)	(67 72)	(69 65 72)	(69 65 72)	(69 66 72)	(69 66 69)	(74)	(70 71 73 67)	(69 70)	(72 73 75 76)
(f maj7)	(eb 7)	(f maj7)	(g m7)	(g m7)	(g m7)	(c 7)	(c 7)	(f maj7)	(f maj7)	(d 7)	(d 7)	(g m7)	(g m7)	(c 7)	(c 7)

Grid	(f maj7)	(f maj7)	(f maj7)	(g m7)	(g m7)	(c 7)	(c 7)	(f maj7)	(f maj7)	(g m7)	(g m7)	(c m7)	(c m7)	(f 7)	(f 7)
Impro	(65 66 67)	nil	(72 77 79 81)	(79)	(74 86)	nil	(81 84)	(82 81)	(77 73)	(74 79)	(77 76)	(72 70)	(67 68)	(69 72)	(69 67)
Oracle	(d 7)	(d 7)	(d 7)	(f 7)	(f 7)	(d 7)	(g 7)	(d 7)	(g 7)	(g 7)	(g# 7)	(g# 7)	(d 7)	(d 7)	(f 7)
(bb 7)	(bb 7)	(bb 7)	(bb 7)	(bb m7)	(bb m7)	(eb 7)	(eb 7)	(f maj7)	(f maj7)	(g m7)	(g m7)	(a m7)	(a m7)	(d 7)	(d 7)
(74)	(72 68 65)	(74 72 68)	(65 74 72)	(68 65 74)	(73 68 65)	(72 70 67)	(65 63 62)	(60 62 63)	(65 66 67)	(69 70 72 73)	(75 72 73 75	(79 81 83)	(84 81)	(78 74)	nil
(f 7)	(d 7)	(f 7)	(d 7)	(f 7)	(f 7)	(g 7)	(g# 7)	(f# 7)	(c 7)	(e 7)	(g 7)	(a 7)	(g 7)	(bb 7)	(a 7)
(g m7)	(g m7)	(g m7)	(g m7)	(g m7)	(g m7)	(c 7)	(c 7)	(f maj7)	(f maj7)	(d 7)	(d 7)	(g m7)	(g m7)	(c 7)	(c 7)
(77)	(75)	(74 72)	(70 68)	(69 72)	(70 67)	(72 70)	(67 72)	(69 65 72)	(69 65 72)	(69 66 72)	(69 66 69)	(74)	(70 71 73 67)	(69 70)	(72 73 75 76)
(g 7)	(g 7)	(e 7)	(e 7)	(a 7)	(bb 7)	(g 7)	(g 7)	(d 7)	(f 7)	(f# 7)	(d 7)	(d 7)	(g 7)	(f 7)	(g 7)

FIGURE 4.4 – Harmonisation d'une même improvisation sur *Au Privave* avec l'oracle d'harmonisation appris sur *Au Privave* (en haut) et l'oracle d'harmonisation appris sur *J'aime pour la vie* de Bernard Lubat (en bas). (Extrait de l'interface Max/MSP)

Conclusion et perspectives

A l'image de l'organisation de ce document, la nature de ce projet a permis d'aborder un éventail de problématiques allant de l'application de concepts algorithmiques à des questions musicales, en passant par des questions techniques d'implémentation informatique, jusqu'à la recherche d'une certaine ergonomie dans le but de créer un outil interactif. La qualité recherchée étant donc celle de l'interaction, le but poursuivi tout au long de la conception de cet instrument était de privililégier la multiplication des contrôles afin de laisser les choix esthétiques et techniques à la discrétion de l'utilisateur potentiel de ce prototype expérimental.

Ce choix de s'éloigner de l'automatisation pour multiplier le champ des possibles transparaissant dans la conception de l'outil en tant que tel se retrouve également dans les choix musicaux qui ont été implicitement pris : en effet malgré la présence de labels harmoniques annotant toutes les phrases, aucune règle musicale n'intervient en dehors du terrain expérimental qui y est consacré avec le module d'arrangement de la grille par substitution. Le processus d'harmonisation et d'arrangement utilise les labels comme un alphabet de passage pour multiplier les possibilités mais n'a aucune idée du sens porté par ces étiquettes.

Dans cette optique, si la notion de "primitives" présente dans les systèmes présentés dans l'état de l'art ([26, 25, 16, 17, 20]) est intéressante en tant que calcul statistique décrivant pertinemment des caractéristiques musicales qualitatives (la densité, le caractère rythmique, le contour mélodique...), l'interprétation quasi-systématique de ces indicateurs en terme "d'intentions de jeu" supposées auxquelles sont associées automatiquement des catégories de réactions référencées ne trouvait pas sa place dans ce dispositif. Cependant, l'intégration de telles grandeurs en tant que descripteurs sera la prochaîne étape du développement de notre outil. Les fonctions calculant tous les indicateurs recensés dans les travaux cités (aussi bien en direct sur les entrées du système que sur les phrases déjà calculées et le contenu des oracles) sont déjà implémentées, et il reste maintenant à décider des modalités de leur visualisation graphique. A travers cette représentation, elles aiguilleront les choix du membre de la formation utilisant le logiciel en lui permettant de repérer rapidement les caractéristiques musicales qualitatives des différents fragments qui lui sont possibles de jouer pour lui permettre une maîtrise plus grande et un contrôle plus fin de ses sorties. L'idée serait d'aboutir à un système comparable à celui du logiciel OMax qui permet de délimiter des régions pour la navigation grâce à la visualisation des liens créés ([18]). On pourrait de la même manière limiter la navigation à des zones très rythmées, syncopées, présentant des chromatismes,...

Un autre enrichissement, cette fois directement lié au parcours d'un oracle est à l'étude : celle de la première étape de l'harmonisation qui consiste actuellement en un parcours

contraint de l'oracle courant d'harmonisation sur un certain descripteur de la mélodie qu'est la signature mélodique de la tranche (hauteur des notes non ordonnées et sans prendre en compte les durées). Même si les résultats obtenus avec ce descripteur très simple sont satisfaisants, d'autres descripteurs peuvent être envisagés : un nouveau champ a été ajouté en s'inspirant de [29, 28] pour intégrer la durée relative de chaque note au sein d'un `Melobeat`. Dans ce cas, le champ observé consiste en un vecteur de 12 coordonnées (une par demi-ton) représentant la durée relative de chaque note au sein de la tranche. L'implémentation de cette nouvelle description et de toutes les fonctions associées est réalisée, mais son utilisation nécessite une étude préalable : en effet, le comparateur associé à de tels labels fait intervenir un calcul de distance, et la définition des "classes d'équivalences" nécessite donc la définition d'une distance seuil à partir de laquelle deux états appartiendraient à la même classe. L'étude d'une distance limite pertinente dans le cadre de la construction d'un oracle et dans le cadre de son parcours doit donc être menée si l'on veut pouvoir choisir d'utiliser également cette deuxième famille de descripteurs.

Enfin, dans la lignée du travail ébauché avec l'affichage des labels harmoniques trouvés lors de la phase d'harmonisation, une étude systématique de l'influence de la nature du corpus et du format des oracles qui le constituent est à mener. Tout d'abord en étudiant les conséquences de la proximité ou non (avec une mesure qui reste à définir) de la progression harmonique de la mélodie à harmoniser et de celles sur lesquelles ont été construit l'oracle d'harmonisation. D'autre part l'influence de la taille des oracles sur la qualité du rendu se pose : l'utilisation d'oracles d'harmonisation et d'arrangement construits sur un seul morceau produisant déjà des résultats très pertinents, il pourrait être interessant d'observer la manière dont les résultats évoluent sur des grands oracles dans lesquels de plus en plus d'oracles issus de l'apprentissage sur un unique morceau ont été concaténés. A cela peut d'ailleurs s'ajouter la question de l'évaluation des résultats obtenus en travaillant sur des grands oracles construits à partir d'oracles issus de morceaux et de styles très hétérogènes. Si des considérations de théorie musicale permettent de donner un cadre à l'étude des résultats de l'étape d'harmonisation symbolique, une méthodologie rigoureuse pour l'évaluation des résultats de l'étape d'arrangement reste à déterminer.

.

Bibliographie

[1] C. Allauzen, M. Crochemore, and M. Raffinot. Factor oracle : A new structure for pattern matching. In *SOFSEM 99 : Theory and Practice of Informatics*, pages 758–758. Springer, 1999.

[2] C. Allauzen, M. Crochemore, and M. Raffinot. Factor oracle, suffix oracle. 1999.

[3] G. Assayag and G. Bloch. Navigating the oracle : A heuristic approach. In *International Computer Music Conference*, volume 7, pages 405–412, 2007.

[4] G. Assayag, G. Bloch, and M. Chemillier. «omax-ofon». *Sound and Music Computing (SMC)*, 2006.

[5] G. Assayag, G. Bloch, M. Chemillier, A. Cont, and S. Dubnov. Omax brothers : a dynamic yopology of agents for improvization learning. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 125–132. ACM, 2006.

[6] G. Assayag and S. Dubnov. Using factor oracles for machine improvisation. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 8(9) :604–610, 2004.

[7] L. Bonnasse-Gahot. Donner à omax le sens du rythme : vers une improvisation plus riche avec la machine. *Rapport interne, École des Hautes Études en sciences sociales*, 2010. `http://www.cs.cmu.edu/~schneide/tut5/node42.html`.

[8] M. Chemillier. Toward a formal study of jazz chord sequences generated by steedman´ s grammar. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 8(9) :617–622, 2004.

[9] M. Chemillier. Implémentation lisp des modèles pour l'improvisation, cours modèles mathématiques en informatique musicale, master atiam. `http://ehess.modelisationsavoirs.fr/atiam/atiam.html`, 2009.

[10] M. Chemillier. Oracle des facteurs, cours modèles mathématiques en informatique musicale, master atiam. `http://ehess.modelisationsavoirs.fr/atiam/atiam.html`, 2009.

[11] M. Chemillier. Tutorial pour les classes lisp oracle et improvizer. `http://ehess.modelisationsavoirs.fr/atiam/improtek/index.html`, 2010.

[12] C.H. Chuan and E. Chew. A hybrid system for automatic generation of style-specific accompaniment. In *4th Intl Joint Workshop on Computational Creativity*, 2007.

[13] A. Cont. Antescofo : Anticipatory synchronization and control of interactive parameters in computer music. In *Proceedings of the International Computer Music Conference*, 2008.

[14] K. Ebcioglu. An expert system for harmonizing chorales in the style of JS Bach* 1. *The Journal of Logic Programming*, 8(1-2) :145–185, 1990.

[15] Norio Emura, Masanobu Miura, and Masuzo Yanagida. A modular system generating jazz-style arrangement for a given set of a melody and its chord name sequence. *Acoustical Science and Technology*, 29(1) :51–57, 2008.

[16] M. Goto, I. Hidaka, H. Matsumoto, Y. Kuroda, and Y. Muraoka. A Jazz Session System for Interplay among All Players-VirJa Session (Virtual Jazz Session System)-. In *In Proc. of ICMC 1996*. Citeseer, 1996.

[17] I. Hidaka, M. Goto, and Y. Muraoka. An automatic jazz accompaniment system reacting to solo. In *In Proc. of ICMC*. Citeseer, 1995.

[18] B. Lévy. Visualising omax. Master's thesis, Master ATIAM, Université Pierre et Marie Curie, Paris VI - IRCAM, 2009.

[19] A. Mancheron, C. Moan, and J. Holub. Combinatorial characterization of the language recognized by factor and suffix oracles. *International Journal of Foundations of Computer Science*, 16(6) :1179–1192, 2005.

[20] D. Martín. Automatic accompaniment for improvised music. Master's thesis, Département de technologies de l'information et de la communication, Universitat Pompeu Fabra, Barcelone, 2009.

[21] PG Music. Band in a Box. *PG Music Software, URL : http ://www. pgmusic. com/index. html*.

[22] F. Pachet. The muses system : an environment for experimenting with knowledge representation techniques in tonal harmony. In *First Brazilian Symposium on Computer Music, SBC&M*, volume 94. Citeseer, 1994.

[23] F. Pachet and P. Roy. Mixing constraints and objects : a case study in automatic harmonization. In *Proceedings of TOOLS Europe*, volume 95, pages 119–126. Citeseer, 1994.

[24] F. Pachet and P. Roy. Musical harmonization with constraints : A survey. *Constraints Journal*, 6, 2001. 6(1) :7-19 2001.

[25] G. L. Ramalho. *Construction d'un agent rationnel jouant du jazz*. PhD thesis, Université Paris VI, 1997.

[26] G.L Ramalho, P.Y Rolland, and J.G. Ganascia. An artificially intelligent jazz performer. *Journal of New Music Research*, 28(2) :105–129, 1999.

[27] R. Ramirez and J. Peralta. A constraint-based melody harmonizer. In *ECAI'98 Workshop on Constraints and Artistic Applications*, 1998.

[28] Microsoft Research. Songsmith.

[29] I. Simon, D. Morris, and S. Basu. MySong : automatic accompaniment generation for vocal melodies. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 725–734. ACM, 2008.

[30] B. Thom. BoB : an interactive improvisational music companion. In *Proceedings of the fourth international conference on Autonomous agents*, pages 309–316. Citeseer, 2000.

[31] B. Thom. Interactive improvisational music companionship : A user-modeling approach. *User Modeling and User-Adapted Interaction*, 13(1) :133–177, 2003.

[32] N. Yogev and A. Lerch. A System for Automatic Audio Harmonization. 2008.