IMPROTEK : INTÉGRER DES CONTRÔLES HARMONIQUES POUR L'IMPROVISATION MUSICALE DANS LA FILIATION D'OMAX

Jérôme Nika
Ircam - Paris, puis Télécom ParisTech
46 rue Barrault - 75013 Paris
jerome.nika@telecom-paristech.fr

Marc Chemillier
Centre d'analyse et de mathématique sociales
EHESS, 190 avenue de France - 75013 Paris
chemilli@ehess.fr

RÉSUMÉ

Le système ImproteK présenté dans cet article propose d'intégrer un cadre rythmique et une structure harmonique sous-jacente dans un contexte d'improvisation. Dans la filiation du logiciel d'improvisation OMax [4, 3, 13], il repose sur la structure d'oracle des facteurs pour tirer profit des propriétés particulièrement riches et pertinentes de cet automate dans un contexte musical [5]. Au cours d'une session d'improvisation, ce système peut s'adapter à une pulsation régulière et proposer des phrases musicales suivant une progression harmonique donnée. ImproteK est conçu comme un instrument interactif dédié à la performance : ses improvisations sont construites à partir de la modélisation stylistique réalisée sur le jeu *live* d'un musicien ou sur un corpus *offline*. Associée à des techniques considérant le corpus comme une mémoire musicale, cette modélisation s'étend aux domaines de l'harmonisation et de l'arrangement dans un module d'interaction harmonique.

1. INTRODUCTION

Le propos d'ImproteK est d'allier modélisation stylistique et interaction pour mettre en place un dialogue original entre des musiciens et un improvisateur virtuel nourrissant sa propre inspiration de leur jeu. Ces deux paradigmes et la structure d'automate au coeur de son implémentation (section 2) font de ce système un cousin du logiciel d'improvisation OMax [4, 3, 13] conçu et développé à l'Ircam.

Dans la lignée des travaux sur la modélisation du style menée par G. Assayag et. al [6], OMax apprend le style d'un improvisateur humain grâce à une représentation basée sur la structure d'oracle introduite par C. Allauzen et. al [1] et étendue à un contexte musical [5]. Le logiciel élabore un modèle du jeu du musicien en temps réel, et est ensuite à même de naviguer à travers cette représentation en suivant des chemins différents de ceux empruntés par son partenaire pour créer ainsi de nouvelles improvisations partageant la même esthétique.

Partageant cette intention, ImproteK se concentre sur un contexte d'improvisation musicale associée à une structure harmonique sous-jacente et présentant une pulsation régulière. Le système propose une interaction enrichie en prenant la pulsation en considération dans le cadre d'une grille d'accords donnée (section 3). Cette conception de l'improvisation est concrètement réalisée par le moyen d'une architecture permettant son intégration au sein d'une formation musicale dont le tempo peut être extrait et suivi (section 4). Enfin, ImproteK offre une interaction harmonique en étendant la modélisation stylistique aux domaines de l'harmonisation et de l'arrangement (section 5).

Cet article rappellera premièrement les principes généraux partagés par OMax et ImproteK pour décrire ensuite les développements spécifiques à ce dernier.

2. MODÉLISATION DU STYLE MUSICAL ET STRUCTURE D'ORACLE

2.1. Modélisation stylistique pour l'improvisation et l'harmonisation

Depuis M&Jam Factory [21], Cypher de R. Rowe [18], ou Voyager de G. Lewis [14], souvent considérés comme les premiers systèmes interactifs en temps réel, de nombreux "partenaires virtuels pour l'improvisation" ont été développés. La plupart d'entre eux ont bénéficié du développement des techniques d'apprentissage automatique avec l'idée grandissante d'arriver toujours plus près du discours musical tenu par l'improvisateur humain en interaction.

Parmi ces dispositifs, le Continuator [16] conçu par F. Pachet modélise l'entrée musicale à l'aide d'un modèle Markovien étendu pour créer de nouvelles phrases à partir de cet apprentissage. Comme dans le mode *free* d'OMax, on ne trouve aucun mécanisme de perception rythmique permettant une synchronisation avec le musicien. A l'inverse, GenJam [7] développé par J. Biles est plus proche des expériences antérieures d'OMax avec un mode *beat* : le logiciel propose un accompagnement comportant un tempo fixé pour guider l'improvisation du musicien humain lors de phases d'écoute. Il répète ensuite certaines des séquences entendues après avoir apporté des modifications à l'aide d'un algorithme génétique. Band Out of a Box de B. Thom [20] implique également un accompagnateur non-interactif avec un tempo imposé dans un schéma d'interaction basé sur une logique de "trading fours" : un improvisateur humain et son partenaire virtuel dialoguent en question-réponse sur des séquences de 4 mesures. Chacune des mesures jouées par l'improvisateur

humain est attribuée à une classe appelée *playing mode*, et la réponse de la machine est composée de 4 mesures appartenant à la même suite de modes.

Ce dernier exemple introduit le sujet récurrent de la meilleure segmentation de l'entrée musicale et de sa représentation. Cet élément est particulièrement crucial dans le cas de systèmes s'attelant à l'harmonisation, l'arrangement ou plus généralement l'accompagnement en faisant appel à un corpus. Qu'ils soient interactifs et dédiés à la performance ou conçus pour être utilisés "offline", ils peuvent être classés schématiquement selon deux critères : l'utilisation exclusive de règles musicales formalisées et l'implication d'un corpus musical (voir [15] pour la situation d'ImproteK au sein d'un système de classification plus exhaustif). Dans le cas de cette dernière catégorie, le corpus peut être considéré comme un environnement d'apprentissage pour créer des modèles génératifs (par exemple C. Chuan et E. Chew [11], le logiciel Mysong-Songsmith de Microsoft [19]) et/ou comme une mémoire musicale dans laquelle des fragments sont recherchés et réagencés pour composer le nouveau matériau (bassiste de jazz virtuel de G. Ramalho, ImPact [17]).

La définition de l'unité de segmentation pour le traitement du corpus est un élément caractéristique de l'approche de ces différents systèmes : le *grain* peut en effet correspondre à quelques notes clés d'une mélodie réduite comme chez C. Chuan et E. Chew, à un seul accord pour les logiciels Songsmith et Band in a Box (PG Music), ou à certaines cadences ou séquences d'accords dans le projet ImPact de G. Ramalho. L'idée est de trouver le meilleur équilibre entre des tranches assez longues pour proposer un accompagnement plausible et cohérent, et des tranches assez fines pour éviter de recopier des fragments trop longs et donc trop identifiables dans le corpus.

Dans le cas d'Improtek, l'unité choisie est la *pulsation*. En effet, le modèle d'automate structurant sa mémoire - l'oracle des facteurs - permet de s'affranchir du choix entre cohérence musicale et originalité par rapport au corpus grâce à la continuité qu'il assure par construction, permettant ainsi de travailler avec un grain aussi fin.

2.2. Oracle et reconnaissance de motifs

L'utilisation première de l'oracle des facteurs a été la reconnaissance de motifs dans des chaînes de caractères, pour être ensuite étendue au calcul de facteurs répétés dans un mot et à la compression de données. Cet automate acyclique représente au moins tous les facteurs d'un mot, et l'algorithme de construction incrémentale est linéaire en sa longueur, en temps comme en espace (pour le détail de l'algorithme, se reporter à [1]).

Les propriétés formelles de la structure d'oracle sont détaillées dans les articles fondateurs, et pleinement appliquées à la problématique de *réinjection stylistique* [4] dans plusieurs documents consacrés à OMax. Aussi, seuls les outils assurant la continuité et la cohérence des séquences générées avec un oracle sont brièvement décrits ici : les transitions et les liens suffixiels.

Les *transitions* (liens en avant, lignes pleines) permettent d'atteindre chaque sous-séquence (ou facteur) de la séquence d'origine en partant de l'état initial. De cette manière, chaque progression entre deux états consécutifs de la suite d'origine peut être générée.

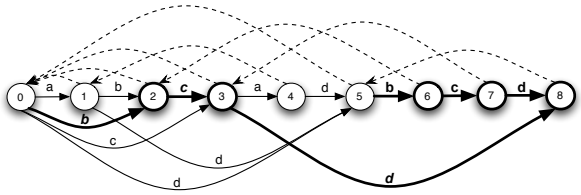

Figure 1. Oracle pour la séquence *abcadbcd* : générer un facteur depuis l'état initial.

La figure 1 présente l'oracle construit sur la séquence *abcadbcd* et illustre ce point en affichant un chemin utilisant des transitions pour produire la sous-séquence *bcd* (0, 2, 3, 8), présente à l'origine entre les états 5 et 8.

L'oracle repère également les motifs répétés dans la séquence d'origine à l'aide des *liens suffixiels* (liens en arrière, lignes pointillées). Ils pointent sur l'état final des occurrences précédemment rencontrées d'un motif.

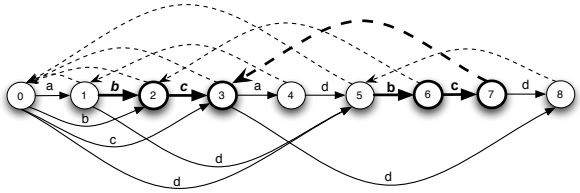

Figure 2. Oracle pour la séquence *abcadbcd* : exemple du motif répété *bc*.

Dans l'exemple de la figure 2, un lien suffixiel connecte les états 7 et 3. 3 est en effet l'état le plus à gauche où le plus long suffixe répété de la séquence terminant en 7 (*abcadbc*) est reconnu : *bc*.

La structure d'oracle peut ainsi être vue comme un outil pour créer de nouvelles séquences grâce à une navigation non-linéaire empruntant ces deux types de liens, sa construction lui permettant de dégager la logique dans les progressions présentes dans les chaînes. Dans le cas d'une application musicale, son aptitude à préserver le discours du matériau original permet de développer une esthétique simultanément nouvelle et proche de celle proposée par les musiciens en interaction.

3. IMPROVISER DANS UN CONTEXTE MÉTRIQUE ET HARMONIQUE

3.1. Oracles en *Mode beat*

Le principe de la version actuelle du logiciel OMax est de faire de chaque élément musical isolé un état dans une instance de la structure d'oracle. La navigation *libre* [2]

en temps réel à travers une mémoire ainsi constituée en fait donc un instrument interactif dédié à l'improvisation dans un contexte de musique *libre*. L'approche d'ImproteK diffère par l'intégration de deux paradigmes. Premièrement, ses improvisations prennent place dans le cadre d'une structure harmonique représentée par une progression symbolique choisie au début de chaque session (*grille d'accords*), lui permettant ainsi d'étendre la modélisation stylistique aux domaines de l'harmonisation et de d'arrangement. Ensuite, il intègre un cadre métrique en instituant la pulsation comme unité élémentaire de ses acquisitions, restitutions, et générations. Pour ce faire, le module d'improvisation a été développé sur la base d'une version précédente d'OMax (OMax 2.0, 2004) conçue comme une bibliothèque Lisp sous l'environnement OpenMusic [9]. Cette version implémentait la structure d'oracle aussi bien pour un contexte d'improvisation libre (mode *free*, dimension retenue dans la version actuelle d'OMax), que pour le cas d'une pulsation régulière (mode *beat*, point de départ du développement d'ImproteK).

Dans cette seconde optique, chaque état d'un oracle représente une tranche musicale dont la durée est donnée par le tempo du morceau en cours et contient différents types d'événements ayant lieu entre deux pulsations consécutives. Ces événements peuvent être des séquences musicales sous format MIDI - fragments mélodiques ou d'accompagnement - ou des données symboliques comme des labels d'accords. Lors des phases de génération, ces différents éléments constituant les états seront tantôt considérés comme des résultats élémentaires à concaténer, tantôt comme des étiquettes (ou labels) à comparer avec la séquence donnée en entrée et servant de chemin à suivre pour le parcours d'un oracle.

3.2. Ecouter et improviser avec l'*oracle live*

ImproteK improvise en recherchant puis réagençant des unités élémentaires pré-existantes : les nouvelles phrases sont composées en concaténant des tranches de pulsations provenant de sa mémoire musicale. Ces fragments sont collectés de manière continue en suivant le guide que constitue la grille d'accords servant de référence à la session d'improvisation. Ce processus fait apparaître une première instance de la structure d'oracle : l'*oracle live*, qui effectue son apprentissage sur les phrases jouées par les musiciens (figure 3). Ces entrées MIDI sont en effet indexées pulsation après pulsation en temps réel par les labels d'accords de la grille harmonique en cours (voir section 4.2) et sont donc formatées pour la construction de cet objet.

Une "improvisation" du système correspond à la création d'une nouvelle phrase musicale à partir de la mémoire stockée dans cet oracle live. Dans ce cadre, les labels d'accords de l'oracle sont considérés comme des index pour l'heuristique de *navigation contrainte* (G. Assayag). Ce mécanisme revient à chercher les motifs d'une séquence d'entrée (ici la grille harmonique) dans une séquence différente (ici les labels d'accords des états de l'oracle). Quand un label correspond, la tranche de pulsation mélodique associée est retournée pour être ajoutée à la séquence des

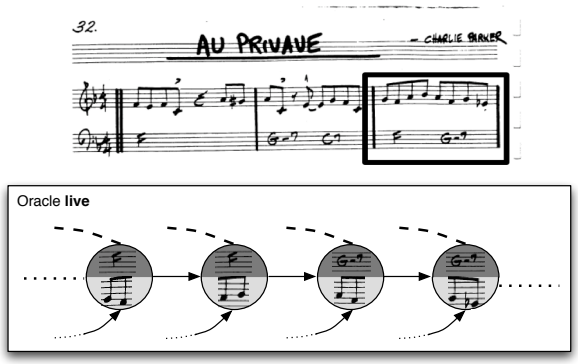

Figure 3. Apprentissage de l'*oracle live*.

résultats des étapes précédentes.

3.3. Navigation contrainte et continuité

Improviser en suivant une grille équivaut ici à suivre un chemin dans l'automate avec des transitions étiquetées par les labels de cette grille. Si aucune transition indexée par le label souhaité n'est trouvée, on cherche à suivre un lien suffixiel permettant d'aller à un autre état où l'on pourra éventuellement lire l'étiquette courante. Les liens suffixiels garantissent qu'il existe, dans la séquence d'origine, une partie commune entre les deux fragments concaténés. La navigation recherche donc en premier lieu la continuité en essayant de coller aux enchaînements appris, et cherche les labels indépendamment s'ils s'inscrivent dans une succession ne figurant à aucun endroit de l'oracle construit sur la séquence d'origine.

Un *paramètre de continuité* est contrôlé à chaque étape du calcul. Il compte le nombre de transitions successives suivies jusqu'alors dans la navigation, et indique ainsi la longueur des segments recopiés de la séquence de l'oracle. En imposant une continuité maximum, on peut donc quantifier l'équilibre souhaité entre la fidélité à la séquence d'origine et l'originalité de la nouvelle improvisation proposée : d'une valeur élevée ressortira une forte similitude, tandis qu'une valeur basse sera à l'origine de plus de surprises.

Lors de la navigation contrainte, les labels successifs de la grille sont lus pour trouver des pulsations indexées par les mêmes labels dans l'oracle live. Si la valeur courante du paramètre de continuité n'excède pas le maximum imposé, la recherche est effectuée en suivant des modes hiérarchisés :

- *Mode "continuity"* : Si son label correspond, suivre une transition, mettre à jour la position de la pulsation courante dans l'oracle, et retourner le fragment mélodique associé. Sinon, passer en *mode suffix* si des liens suffixiels efficaces sont trouvés, ou en *mode nothing* dans le cas contraire.
- *Mode "suffix"* : Suivre le lien suffixiel pointant sur le plus long suffixe répété pour atteindre ensuite un label correspondant (voire figure 4), mettre à jour la position de la pulsation courante dans l'oracle, et re-

tourner le fragment de mélodie associé. Sinon, passer en *mode nothing*.

– *Mode "nothing"* : La recherche de motif est effectuée indépendamment du contexte et le label est cherché dans tout l'oracle. Une transposition peut être opérée si nécessaire.

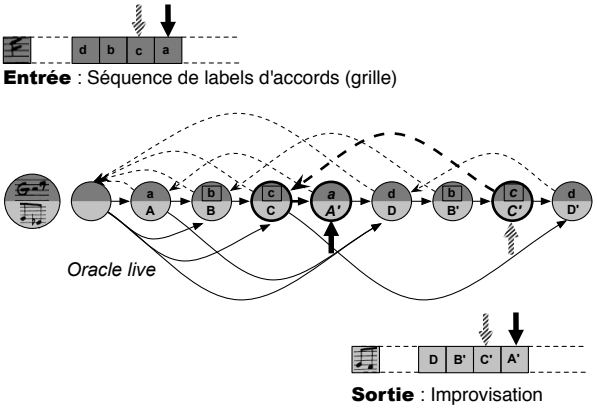

Figure 4. Navigation contrainte à travers l'oracle live.

L'exemple donné par la figure 4 illustre une étape de la navigation. A ce stade, les labels d'accords *d*, *b*, et *c* ont été cherchés et trouvés dans l'oracle live, et la concaténation des fragments musicaux associés *D*, *B´* et *C´* forme la phrase en construction. La position courante dans l'oracle est l'avant-dernier état (*c / C´*) et le label d'accord suivant lu dans la grille donnée en entrée est *a*. Aucune transition pointant sur un label correspondant ne pouvant être trouvé, la recherche continue en *mode suffix*. Le lien suffixiel partant de l'état courant pointe, par définition, sur l'état final du suffixe de *dbc* répété le plus à gauche (*bc*), soit l'état (*c / C*). Une transition correspondant au label *a* recherché est trouvée dans cet état. Le lien suffixiel est donc finalement suivi pour sauter de (*c / C´*) à (*c / C*), ou l'on arrive dans le contexte commun (*bc*) pour pouvoir finalement atteindre la nouvelle pulsation (*a / A´*).

Cette notion de contexte commun est essentielle puisqu'elle assure la cohérence et la musicalité des transitions dans les séquences produites même si la navigation implique des sauts entre différentes zones de l'oracle. En effet, dans le cas de cet exemple, on remarque que ni la séquence d'accords donnée en entrée ni la phrase musicale en sortie ne se trouvaient inscrits en l'état dans la mémoire de l'oracle live.

4. VERS UN INSTRUMENT INTERACTIF

4.1. Architecture

Le jeu des musiciens est reçu, segmenté, formaté et sauvegardé en temps réel via une nouvelle interface développée sous Max/MSP. A travers cette même interface, l'utilisateur d'ImproteK sélectionne une grille d'accords symbolique qui constituera la colonne vertébrale de l'improvisation. Il a également accès aux paramètres de contrôle guidant le calcul des différentes formes de phrases musicales pouvant être générées : de nouvelles improvisations accompagnées ou non (voir section 5) ou des arrangements "live" de la grille (voir paragraphe 4.5).

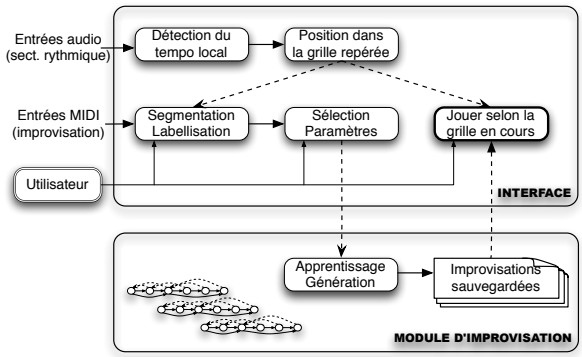

Figure 5. Architecure

Ces instructions de calcul sont envoyées via le protocole OSC au module d'improvisation développé sous OpenMusic présenté dans la section précédente (figure 5), suivant ainsi l'architecture générale d'OMax 2.0 qui fonctionnait en interaction avec Max/MSP grâce à une interface basique ne prenant pas en charge la gestion du tempo développée dans le paragraphe suivant.

Sous cette version, une pulsation métronomique était imposée par la machine, cette condition impliquant nécessairement la perte d'une partie de la richesse musicale et contraignant l'expression des musiciens. Suite à ce constat, un module de suivi de pulsation a été implémenté afin de pouvoir extraire directement le tempo de la session d'improvisation en cours.

4.2. "Get rhythm"

Pendant la performance, une source de pulsation extérieure est nécessaire pour segmenter les entrées provenant des musiciens dans le processus d'écoute, ainsi que pour jouer des phrases rythmiquement en place avec l'improvisation en cours. Pour ce faire, ImproteK fait intervenir l'association d'un module de suivi de pulsation et d'un système de suivi de partition jouant le rôle d'un séquenceur en émettant en temps réel la position courante dans la grille harmonique. Cette donnée sert dans un premier temps à marquer chaque pulsation de l'improvisation live écoutée avec le label d'accord courant. D'autre part, les phrases produites par le module d'improvisation étant des séquences labélisées, ce signal déclenche le jeu des tranches de pulsations correspondantes.

ImproteK se destinant à trouver sa place au sein d'une formation musicale, il était fondamental que son utilisation n'ampute pas la créativité et l'expressivité en subordonnant les musiciens au tempo métronomique imposé par ses sorties. L'objet Max/MSP *beat tracker* [8] a été spécialement développé dans l'optique d'être intégré à un environnement parent d'OMax pour atteindre une musi-

calité plus riche, les expériences antérieures de l'oracle en *mode beat* impliquant un tempo fixe dicté par le système.

Ce module traite des flux MIDI ou audio (par exemple la section rythmique de la formation) et estime en continu le tempo local qui sera utilisé pour générer et jouer des improvisations ajustées à la pulsation humaine nécessairement variable. Pendant une phase d'initialisation, l'utilisateur fournit une indication de tempo par une battue manuelle dont le but est également de donner la phase initiale en indiquant les dates absolues des pulsations pour éviter ainsi une synchronisation à contre-temps. Après cette étape, les variations de tempo sont calculées tout au long de la performance (la description de cette unité, n'étant pas le sujet du présent article, voir [8] pour plus de détails). Ce signal est ensuite utilisé comme une horloge venant déclencher les différents éléments constituant les séquences chargées dans Antescofo.

4.3. Utiliser un *score follower* comme séquenceur

Antescofo [12] est un système de suivi de partitions polyphoniques et un langage de programmation synchrone pour la composition musicale conçu par A. Cont. Cet objet, développé en tant que module externe pour Max/MSP, procède à la reconnaissance automatique de la position et du tempo dans une partition musicale à partir d'un flux audio provenant d'un ou plusieurs interprètes. Il permet ainsi la synchronisation de la performance instrumentale avec des éléments informatiques inscrits dans une partition électronique.

Chaque séquence calculée par le module d'improvisation et destinée à être restituée est donc écrite et sauvegardée sous le format d'une partition Antescofo, pouvant ainsi être jouée à partir de la position courante de la grille dès son chargement. Les phrases ainsi générées viennent enrichir une collection constituée au fil des sessions pour pouvoir être chargées par l'utilisateur au cours de l'improvisation. Dans notre utilisation d'Antescofo, ce sont les pulsations fournies par l'objet *beat tracker* qui tiennent lieu de "notes" dans la partition. Elles viennent déclencher les évènements que sont les ordres 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.

4.4. Segmenter et indexer les entrées

Ce couple d'objets en charge de la gestion de la pulsation se trouve en aval pour jouer les séquences générées, mais également en amont dans le processus d'acquisition en temps réel. 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.

Lors de l'acquisition des entrées, ces 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 de segmenter les séquences MIDI par pulsations et d'attribuer à chacune d'elles un label harmonique dans l'optique de la construction d'oracles en *mode beat* ou de séquences musicales.

4.5. Un instrument force de propositions

ImproteK est un instrument joué par un interprète interagissant avec le reste de la formation au même titre qu'un instrumentiste traditionnel. Ce dernier pilote le système à l'aide de l'interface graphique pour contrôler en temps réel les paramètres des étapes d'apprentissage comme de génération qui sont initiées sur demande. Pendant la performance, il sélectionne les phrases musicales à donner en entrée pour la modélisation stylistique, choisit les oracles courants (l'oracle live tout comme les autres instances impliquées dans le module d'harmonisation et d'arrangement décrit dans la section 5) et il détermine les paramètres techniques comme la continuité maximum pour la navigation dans chaque oracle. Entre autres caractéristiques, le terrain d'apprentissage, la longueur ou même "l'audace" des improvisations sont donc laissées à sa discrétion.

Cet interprète n'est pas uniquement en charge de l'écoute et de l'apprentissage : il dirige pleinement le jeu au moyen de contrôles au clavier. Il peut accéder aux phrases créées à partir des derniers évènements musicaux entendus comme à celles issues des sessions précédentes. Différentes formes d'improvisations lui sont disponibles : elles peuvent être par exemple des phrases mélodiques accompagnées ou non, ou des arrangements live de la grille. 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 la phrase en cours de jeu pendant un seul et même passage de la grille et par exemple de naviguer entre plusieurs improvisations pour créer une improvisation hybride.

Enfin, le rôle joué par ImproteK peut évoluer pendant une même session : il peut tout aussi bien être soliste et/ou accompagnateur selon la nature des séquences choisies par l'interprète pour remplir l'oracle live. En effet, faire effectuer à cet oracle son apprentissage sur le jeu d'un accompagnateur permettra de réaliser ensuite un arrangement de la grille en temps réel. L'autre approche disponible de la question de l'accompagnement est celle fournie par le module d'harmonisation et d'arrangement présenté dans la section suivante.

5. IMPROVISATIONS HARMONISÉES ET ARRANGÉES

Ce module d'harmonisation et d'arrangement peut être utilisé de manière autonome comme une entité indépendante créant un accompagnement pour une mélodie sans interaction temps réel. Cependant, il a été pensé pour être intégré dans l'environnement plus vaste d'ImproteK afin de composer de nouvelles improvisations accompagnées.

Il ne connaît ni applique aucune règle musicale et est exclusivement fondé sur l'utilisation d'un corpus qui est simultanément considéré comme un terrain d'apprentissage pour extraire des mécanismes empiriques, et comme une mémoire musicale dans laquelle piocher pour concrétiser les accompagnements après qu'ils ont été calculés.

5.1. L'apprentissage

5.1.1. Le corpus

L'apprentissage du corpus est effectué sur trois éléments : la *piste mélodique* (thème, improvisation, solo, etc.), la *piste d'accompagnement*, et la *grille harmonique associée*. Il n'est pas conduit sur des données purement symboliques comme des partitions mais sur les performances en direct, et est actuellement constitué de standards de jazz et de morceaux de Bernard Lubat.

On emploiera le terme "corpus" pour désigner l'ensemble des modèles, qu'ils soient extraits d'un apprentissage différé ou construits lors de la performance en cours. Ici encore, le système écoute les musiciens, segmente ses entrées par pulsation, et apprend ces séquences ainsi que les associations entre ces trois éléments (mélodie, grille, accompagnement) en construisant des instances d'oracle. La phase d'apprentissage est en réalité double : un couple d'oracles est construit pour chaque élément du corpus : un oracle d'harmonisation et un oracle d'arrangement (figure 6).

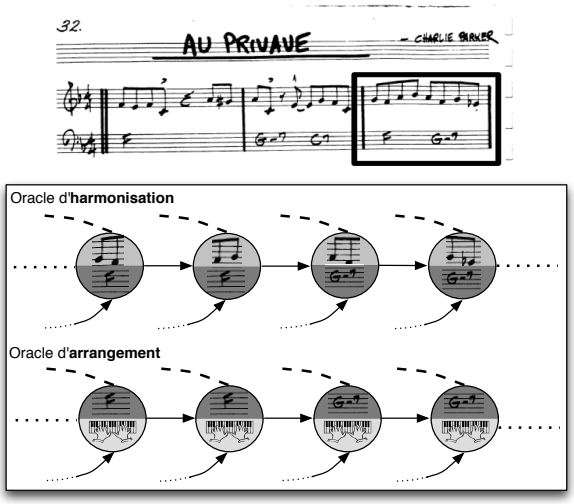

Figure 6. Apprentissage des oracles d'arrangement et d'harmonisation

Un *oracle d'harmonisation* mémorise les séquences d'associations entre les fragments mélodiques et les labels d'accords pour les pulsations qu'il couvre. Dans le cas d'un *oracle d'arrangement*, les associations apprises sont celles entre les labels d'accords et les fragments d'accompagnement. Comme leurs noms l'indiquent, ces deux objets seront respectivement utilisés pour associer une progression harmonique symbolique à une mélodie, et un accompagnement à une progression harmonique symbolique.

5.1.2. Classes d'équivalence

Cette construction incrémentale tout comme les filtrages par motif successifs décrits dans le prochain paragraphe (5.2.1) est en pratique exécutée sur des classes d'équivalence. Pour un oracle d'harmonisation, deux états sont considérés comme équivalents s'ils sont indexés par les mêmes notes sans prendre en compte leurs durées, leurs répétitions ou leur ordre dans la tranche de pulsation. De la même manière, deux états d'un oracle d'arrangement sont équivalents s'ils sont indexés par les mêmes labels d'accords, quels que soient les fragments d'accompagnement associés.

Ces équivalences ont été introduites pour éviter la rigidité improductive qu'aurait pu amener un critère d'égalité stricte. Les conséquences structurelles pour les oracles concernent le nombre de liens suffixiels efficaces comme l'illustre l'exemple simplifié de la figure 7 appliqué à une chaîne de caractère : un oracle construit sur le mot *oracle*.

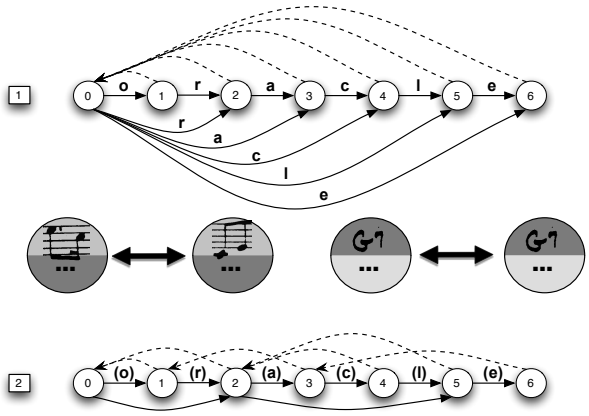

Figure 7. Classes d'équivalence et liens suffixiels.

Dans le premier exemple, chaque lien suffixiel pointe par convention sur l'état initial car aucun motif répété n'est reperé (chaque lettre n'apparaissant qu'une fois). Dans le deuxième cas introduisant les classes d'équivalence *voyelles* et *consonnes*, des motifs répétés sont observés et la structure est plus complexe.

5.2. Harmonisation et arrangement

5.2.1. Cascade d'oracles

Harmonisation et arrangement sont réalisés en cascade : une navigation contrainte est effectuée dans un premier

temps à travers un oracle d'harmonisation, puis à travers un oracle d'arrangement (figure 8).

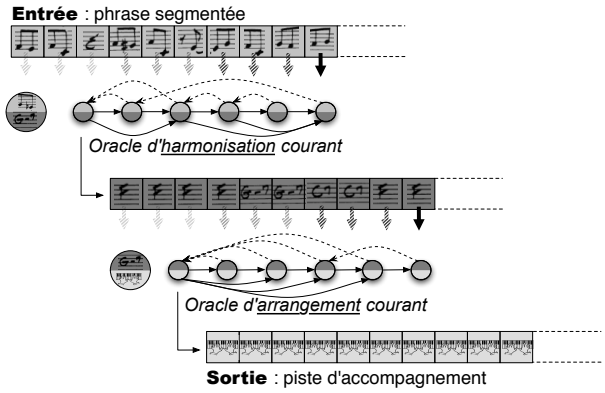

Figure 8. Harmonisation et arrangement en cascade.

L'étape d'harmonisation retourne une progression de labels d'accords, et cette séquence symbolique devient elle-même le chemin à suivre pour la navigation dans l'oracle d'arrangement. La recherche de motifs est cette fois exécutée sur les étiquettes que constituent ces labels d'accords pour obtenir finalement une séquence formée par la concaténation des tranches d'accompagnement trouvées à chaque étape de la recherche.

La continuité intrinsèque à l'utilisation de la structure d'oracle apporte des progressions harmoniques cohérentes pour la phase d'harmonisation. Elle propose également une reconstruction homogène à partir des différentes pulsations du corpus assemblées pour créer le nouvel accompagnement. Ces propriétés de continuité se répercutent par exemple sur les lignes de basses dans les accompagnements complexes évitant ainsi des phrasés trop accidentés.

5.2.2. Intermédiaire formel

Il est important de souligner que, même si une étape intermédiaire implique des données symboliques comme des labels d'accords, aucune règle harmonique n'est appelée pour procéder à l'harmonisation et à l'arrangement. La dénomination choisie pour ce langage formel intermédiaire est seulement destinée à l'utilisateur pour rendre la performance plus intuitive : le système lui-même ignore la signification musicale de ce marquage et ne considère deux labels d'accords différents que comme deux index à comparer.

L'insertion de ce langage formel à un niveau intermédiaire séparant le procédé en deux étapes disjointes est motivée par trois raisons. Tout d'abord, il découle naturellement de la notation usuelle des standards de jazz comme on peut par exemple les trouver dans les Realbooks où un thème est mis en regard d'une suite d'accords correspondante. De plus, cette factorisation du problème permet de multiplier les possibilités : une phrase peut en effet être harmonisée à l'aide d'un oracle donné et ensuite arrangée avec un oracle d'arrangement appris sur un tout autre corpus. Enfin, cet intermédiaire permettra d'implémenter dans un développement futur un niveau optionnel de substitutions basé sur une grammaire [10].

Les notations et le vocabulaire employés ne doivent pas masquer la portée générale de ce module. En effet, les termes "harmonisation" et "arrangement" viennent des situations dans lesquelles ImproteK a été utilisé jusqu'à maintenant : des sessions d'improvisation dans un contexte de jazz tonal. Dans d'autres environnements musicaux, sa généricité lui permet de comprendre d'autres formes d'associations verticales pouvant être indexées de manière agnostique en employant une autre grammaire.

6. EXPÉRIMENTATIONS ET RÉSULTATS

ImproteK a servi de partenaire virtuel à des musiciens professionnels, en particulier Bernard Lubat lors de séances d'improvisation menées à Uzeste depuis 2011. Des exemples audio et vidéo illustrant les différents points abordés dans l'article sont disponibles sur la page :
http ://ehess.modelisationsavoirs.fr/improtech/improtek.

Parmi eux, le contrôle du logiciel en temps réel à l'aide de l'interface Max/MSP est présenté par le biais d'une improvisation basée sur des transcriptions du style mambo d'Erroll Garner. L'interaction directe entre la machine et un instrumentiste est illustrée par des extraits de sessions où le musicien, Bernard Lubat, accompagne la machine jouant une improvisation inspirée de son propre chorus, suivis d'une discussion sur le résultat.

Une autre série d'exemples expose l'éventail de résultats possibles pour le module d'harmonisation et d'arrangement en proposant successivement plusieurs accompagnements pour une même improvisation jouée en boucle. En effet, selon les choix de l'utilisateur concernant les différentes parties du corpus à utiliser et les paramètres comme la continuité imposée, le rendu peut aller de la simple imitation lorsque les oracles d'harmonisation, d'arrangement, et live sont choisis dans une même partie du corpus, à l'originalité voire l'extravagance quand des oracles complètement indépendants ont été activés.

7. CONCLUSION

Le système d'improvisation musicale présenté dans cet article est capable de dégager les logiques des associations horizontales et verticales sous-tendant une performance d'improvisation pour devenir lui-même force de proposition en développant son esthétique propre inspirée de celle de ses partenaires. Sa matière première est en effet leur jeu : il est simultanément employé comme terrain d'apprentissage pour la modélisation du style et comme mémoire musicale pour concrétiser ses propres improvisations.

Les propriétés de l'oracle structurant sa mémoire permettent de s'affranchir du dilemme du choix entre innovation et cohérence en assurant la continuité par construction, et en permettant ainsi de travailler avec le grain fin qu'est la pulsation. Cette unité est choisie comme élément de base pour les calculs comme pour la restitution finale

et est servie par un module de suivi de pulsation pour une interaction enrichie.

ImproteK est en effet conçu comme un instrument à part entière et nécessite un interprète pour diriger l'apprentissage, la génération, et le jeu en temps réel. Il peut tour à tour être soliste ou accompagnateur, et peut également proposer des improvisations accompagnées grâce au module d'harmonisation et d'arrangement.

A plus court terme que certains objectifs techniques comme la prise en charge de l'audio, les directions prises actuellement par le projet ImproteK visent à évaluer finement la compatibilité entre la progression harmonique de la grille de la session d'improvisation et celles résultant du module d'harmonisation et d'arrangement. Au stade actuel, l'interprète observe la comparaison entre les deux grilles par l'intermédiaire de l'interface affichant les labels d'accords respectifs après chaque création d'improvisation accompagnée. Cette étude permettra une meilleure intégration de l'interaction harmonique dans l'instrument, et rendra son utilisation plus intuitive.

8. REMERCIEMENTS

Avec le soutien de l'ANR.
Projet IMPROTECH ANR-09-SSOC-068.

Nous souhaitons remercier la famille OMax, Gérard Assayag, Georges Bloch, et Benjamin Lévy pour les échanges fructueux d'expériences et d'idées concernant la conception et l'implémentation d'ImproteK. Nous remercions également Laurent Bonnasse-Gahot qui lui a donné le sens du rythme avec le *beat tracker*, ainsi que Carlos Agon et Jean Bresson pour leurs conseils concernant OpenMusic, et Arshia Cont pour ses modifications d'Antescofo sur mesure. Nous adressons enfin un remerciement tout particulier à *La Compagnie Lubat* pour ses accueils répétés et les collaborations toujours plus enrichissantes.

9. REFERENCES

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

[2] G. Assayag and G. Bloch, "Navigating the oracle : A heuristic approach," in *International Computer Music Conference*, vol. 7, 2007, pp. 405–412.

[3] G. Assayag, G. Bloch, and M. Chemillier, "Omaxofon," *Sound and Music Computing (SMC)*, 2006.

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

[5] G. Assayag and S. Dubnov, "Using factor oracles for machine improvisation," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 8, no. 9, pp. 604–610, 2004.

[6] G. Assayag, S. Dubnov, and O. Delerue, "Guessing the composerÕs mind : Applying universal prediction to musical style," in *Proceedings of the International Computer Music Conference*, 1999, pp. 496–499.

[7] J. Biles, "Genjam : Evolutionary computation gets a gig," in *Proceedings of the 2002 Conference for Information Technology Curriculum, Rochester, New York, Society for Information Technology Education*, 2002.

[8] L. Bonnasse-Gahot, "Donner à omax le sens du rythme : vers une improvisation plus riche avec la machine," École des Hautes Études en sciences sociales, Tech. Rep., 2010, http ://ehess.modelisationsavoirs.fr/improtech/docs.

[9] J. Bresson, C. Agon, and G. Assayag, "Openmusic 5 : A cross-platform release of the computer-assisted composition environment," in *10th Brazilian Symposium on Computer Music, Belo Horizonte, MG, Brésil*, 2005.

[10] M. Chemillier, "Toward a formal study of jazz chord sequences generated by steedman´ s grammar," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 8, no. 9, pp. 617–622, 2004.

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

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

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

[14] G. Lewis, "Too many notes : Computers, complexity and culture in voyager," *Leonardo Music Journal*, pp. 33–39, 2000.

[15] J. Nika, "Intégrer l'harmonie dans un processus informatique d'improvisation musicale," Master's thesis, Master ATIAM, Université Pierre et Marie Curie, Paris VI - IRCAM, 2011, http ://articles.ircam.fr/textes/Nika11a/index.pdf.

[16] F. Pachet, "The continuator : Musical interaction with style," *Journal of New Music Research*, vol. 32, no. 3, pp. 333–341, 2003.

[17] G. Ramalho, P. Rolland, and J. Ganascia, "An artificially intelligent jazz performer," *Journal of New Music Research*, vol. 28, no. 2, pp. 105–129, 1999.

[18] R. Rowe, *Interactive music systems : machine listening and composing*. MIT press, 1992.

[19] 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*. ACM, 2008, pp. 725–734.

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

[21] D. Zicarelli, "M and jam factory," *Computer Music Journal*, vol. 11, no. 4, pp. 13–29, 1987.