Improvisation musicale homme-machine guidée par un scénario temporel

Jérôme Nika, Marc Chemillier

▶ **To cite this version:**

Jérôme Nika, Marc Chemillier. Improvisation musicale homme-machine guidée par un scénario temporel. Revue des Sciences et Technologies de l'Information - Série TSI : Technique et Science Informatiques, 2015, Numéro spécial Informatique Musicale, 7-8 (33), pp.651-684. hal-01107194

HAL Id: hal-01107194
https://hal.science/hal-01107194

Submitted on 7 Sep 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Improvisation musicale homme-machine guidée par un scénario temporel

Jérôme Nika [1], **Marc Chemillier** [2]

1. *Ircam, UMR STMS 9912 CNRS - UPMC*
   1 place Igor Stravinsky
   Paris, France
   jerome.nika@ircam.fr
2. *CAMS, EHESS, UMR 8557 CNRS*
   190 avenue de France
   Paris, France
   chemilli@ehess.fr

RÉSUMÉ. *Cet article propose un modèle pour l'improvisation musicale guidée par une structure formalisée. Il exploite les connaissances a priori du contexte d'improvisation pour introduire de l'anticipation dans le processus de génération. « Improviser » signifie ici articuler une mémoire musicale annotée avec un « scénario » guidant l'improvisation (par exemple une progression harmonique dans le cas du jazz). Le scénario et la séquence étiquetant la mémoire sont représentés comme des mots construits sur un même alphabet. Cet alphabet définit les classes d'équivalences choisies pour décrire les éléments musicaux constituant la mémoire. La navigation dans cette mémoire assure la cohérence et l'homogénéité du résultat musical en exploitant les motifs communs aux deux séquences tout en étant capable de s'éloigner du matériau musical d'origine. Le modèle et l'architecture d'ordonnancement décrits dans cet article sont implémentés dans le système d'improvisation ImproteK, utilisé à plusieurs reprises avec des musiciens experts.*

ABSTRACT. *This paper presents a computer model for musical improvisation guided by a formalized structure. It uses the prior knowledge of the temporal structure of the improvisation to introduce some anticipation in the music generation process. In this framework "improvising" amounts to articulating an annotated memory and a "scenario" guiding and constraining the improvisation (for example a given chord progression in the case of jazz improvisation). The scenario and the sequence describing the musical memory are represented as words defined over the same alphabet. This alphabet describes the equivalence classes chosen to label the musical contents of the online or offline memory. The navigation through this memory searches for continuity in the musical discourse by exploiting similar patterns in the sequences, as well as the ability to go beyond the simple copy of their associated musical contents. The model and the scheduling architecture described in this paper are implemented in the improvisation software ImproteK which has been used at several occasions with expert musicians.*

MOTS-CLÉS : *improvisation musicale guidée, scénario, recherche de motifs, recherche de préfixes, indexation, apprentissage, interaction, ImproteK.*

KEYWORDS: *guided improvisation, music generation, scenario, pattern matching, prefix matching, indexing, learning, interaction, ImproteK*

1. Introduction

1.1. Improvisation structurée et guidée

L'improvisation jazz ou blues s'appuie traditionnellement sur une « grille » définissant une progression harmonique ; la basse continue baroque laisse à l'interprète la réalisation de l'harmonie en improvisant à la main droite à partir d'une basse écrite pour la main gauche... L'existence d'un objet structuré temporellement et antérieur à la performance, non nécessairement décrit avec un vocabulaire harmonique, se retrouve également dans de nombreuses autres traditions improvisées, notamment dans la musique orientale. On peut citer à ce titre la musique classique d'Inde du Nord dans laquelle l'improvisation est guidée à plusieurs échelles : l'enchaînement de parties de longueurs différentes et ayant une identité propre dans l'exemple des raga en fournit un premier niveau. L'évolution au sein de chacune de ces parties est ensuite construite à partir de descriptions détaillées en termes de mélodie, de vitesse ou de registre : par exemple suivre un cheminement mélodique comportant une montée progressive vers la double octave supérieure suivi d'une descente rapide dans le registre médium.

En sortant du champ musical, cette notion de structure formalisée peut se transposer à la *commedia dell'arte* dont le canevas décrit de manière minimale la séquence d'évènements et de situations que suivra l'intrigue de la représentation, ou encore à la tradition orale des contes populaires traditionnels ou des épopées pour lesquels le récit improvisé s'appuie sur une trame constituée d'unités élémentaires, parfois associées à un ensemble de formules consacrées, définissant ainsi une grammaire du récit. L'idée de structure temporelle préexistante et guidant la performance improvisée trouve un écho dans les études cognitives traitant de l'improvisation, qui font émerger une notion de planification plus large que le simple canevas formel (car elle inclut l'ensemble des stratégies narratives de l'improvisateur) et son articulation avec les exigences de réactivité face à des situations imprévues qui caractérise l'improvisation. La notion de « plan » dans (Sloboda, 1982) ou (Shaffer, 1980) est définie comme un objet abstrait représentant la structure fondamentale de la performance, et laissant les dimensions plus fines être générées ou agencées en temps voulu pendant l'exécution. Dépassant la seule séquence de contraintes formelles, (Pressing, 1984 ; 1988) introduit la notion de « référent » en tant que schéma sous-jacent guidant ou aidant la production du matériel musical. Ce schéma sous-jacent peut également s'étendre aux dimensions cognitives,

perceptives ou émotionnelles, et sa relation avec le comportement improvisé peut être, entre autres, métaphorique, imitative, allégorique, ou antagoniste.

L' « improvisation automatique » fera ici référence à un système informatique capable de produire de la musique (typiquement une séquence de notes jouées par un synthétiseur ou une séquence de dates lues dans un fichier audio) en relation directe avec le contexte musical produit par une situation de concert. Le modèle d'improvisation automatique décrit dans cet article, implémenté dans le système d'improvisation ImproteK, repose sur un canevas préexistant à la performance. Dans ce cadre, le canevas guidant le modèle est représenté par une séquence de contraintes formelles pour l'improvisation que le système doit générer. De la même manière que dans les exemples de répertoires improvisés cités précédemment, cet objet ne porte pas la dimension « narrative » de l'improvisation, c'est-à-dire la conduite de l'improvisation dans son évolution fondamentalement esthétique et non explicite. Pour cette raison, le modèle bien que complet et à l'exécution autonome s'efforce d'offrir à un opérateur pilotant le système (lui-même en situation de musicien), un contrôle esthétique et musical, considérant que cette dimension narrative est indissociable de la performance incarnée.

1.2. Systèmes informatiques pour l'improvisation : état de l'art et historique

Parmi les différents projets s'intéressant à la question de l'improvisation musicale automatique, on se limitera ici aux systèmes construisant leurs improvisations en concaténant des fragments musicaux (captés en temps réel ou au préalable) qui sont réinjectés dans la performance après avoir été transformés et réagencés. Ces systèmes peuvent se distinguer selon le paradigme choisi pour guider l'improvisation, et la longueur du futur anticipé qu'il implique. Comme OMax (Assayag, Bloch, Chemillier, Cont, Dubnov, 2006 ; Assayag, Bloch, Chemillier, 2006 ; Lévy *et al*., 2012), dans la lignée duquel le projet décrit dans cet article est construit, les systèmes de « co-improvisation » créent de nouvelles improvisations en naviguant dans une représentation extraite du jeu d'un musicien capté en temps réel. Un paradigme similaire sous-tend également des travaux tels que (Pachet, 2003) ou (François *et al*., 2013). Le parcours de la mémoire musicale suit pas après pas la logique interne du matériau capté et peut laisser à un opérateur le soin de contrôler la génération. D'autres systèmes peuvent être guidés par une écoute réactive et une analyse des entrées musicales : à partir d'une mémoire également captée en temps réel ou d'un corpus prédéfini, des contraintes sur l'instant ou à plus long terme peuvent être imposées pour piloter la navigation dans cette mémoire. SoMax (Bonnasse-Gahot, 2013) traduit à chaque instant le jeu d'un improvisateur sous forme de contraintes, par exemple décrites en termes de contexte harmonique, pour aiguiller la navigation dans un corpus dont la logique interne est suivie quand les contraintes sont relâchées. Axé sur l'interaction et la réactivité, (Moreira *et al*., 2013) fonctionne selon un principe d'observations multimodales du jeu d'un musicien permettant de trouver dans une mémoire un segment musical approprié selon des associations préalablement apprises. Dans cette lignée, (Pachet *et al*., 2013) utilise également des annotations harmoniques a priori

parmi les critères de recherche. Enfin, une structure connue ou définie en amont peut diriger le processus de génération, en exploitant ou non cette connaissance a priori pour introduire de l'anticipation. Dans cette catégorie (Dubnov, Assayag, 2005) puis (Surges, Dubnov, 2013) proposent la création de règles ou de scripts pour contrôler les paramètres de génération d'une session d'improvisation. Avec un horizon plus long, (Donze *et al*., 2013) introduit une structure de contrôle pour guider l'improvisation selon une séquence de référence, jouant ainsi un rôle comparable à celui du scénario dont il sera question dans cet article. On notera également qu'un certain nombre d'autres travaux n'utilisent pas directement le matériau musical capté en direct, mais produisent des improvisations à partir de modèles créés au préalable avec lesquels le musicien interagit durant le concert. Ainsi, (Thom, 2000) construit itérativement des modèles de jeu avec une logique de « trading fours » dans un schéma d'interaction à court terme, et (Sioros, Guedes, 2011a ; 2011b) se concentrent sur l'aspect rythmique en extrayant d'une analyse de l'improvisation en cours les paramètres contrôlant des modèles génératifs prédéfinis.

On se situe ici dans le périmètre de l'improvisation musicale *idiomatique* (Bailey, 1999), en postulant que dans ce cadre et dans un contexte donné l'improvisation se construit en partie sur la mobilisation et la transformation d'éléments que l'on a pu entendre ou jouer soi-même dans un contexte différent mais défini avec le même langage. Dans cette optique, la musicalité même réside dans le caractère inattendu, impromptu ou anticipé de ces mobilisations. Le projet présenté dans cet article s'inscrit dans la continuité des travaux initiés dans (Dubnov *et al*., 1998 ; Assayag *et al*., 1999) sur la navigation dans une cartographie du jeu d'un musicien « analogique » capturant une partie de sa logique musicale. L'application de ces principes à un logiciel d'improvisation en temps réel a donné lieu à la création du logiciel OMax (Assayag, Bloch, Chemillier, Cont, Dubnov, 2006 ; Assayag, Bloch, Chemillier, 2006 ; Lévy *et al*., 2012). La prise en compte de la question de la « pulsation » dans les systèmes d'improvisation et les travaux menés sur l'utilisation des grilles de jazz dans l'improvisation (Chemillier, 2001 ; 2004 ; 2009 ; Chemillier, Assayag, 2008) ont ensuite conduit au développement du système ImproteK. Ce système introduit des contraintes à long terme et des connaissances a priori dans un processus informatique d'improvisation. Celles-ci sont ajoutées au moyen d'un formalisme abstrait pouvant traduire des notions musicales comme celle de pulsation sur le plan rythmique, ou de grille d'accords sur le plan harmonique. L'architecture d'ImproteK décrite dans (Nika, Chemillier, 2012) associe un modèle de génération considérant ces notions musicales et le système de suivi de partition Antescofo (Cont, 2008) permettant la synchronisation du jeu des phrases calculées pendant la performance.

1.3. Un modèle d'improvisation guidée par scénario

En reprenant le schéma décrit précédemment, cet article présente une nouvelle approche de l'improvisation automatique intégrant une notion de *guidage*. Celle-ci exploite les connaissances a priori sur la structure temporelle de l'improvisation, que l'on appellera *scénario*, afin d'intégrer une *anticipation* dans le processus de généra-

tion musicale. La section 2 introduit ainsi le modèle de génération pour l'improvisation idiomatique dans lequel une mémoire musicale structurée et annotée est parcourue en suivant un chemin conforme à un scénario guidant l'improvisation.

Avec le souci de fonder le traitement informatique de la question musicale sur un processus constituant une métaphore de la réalité de l'improvisation, la section 3 décrit la première *étape* d'une *phase* du processus de génération, qui utilise des outils classiques d'algorithmique du texte pour reconnaître à chaque instant les préfixes du scénario courant dans la mémoire musicale, à la manière d'un musicien improvisateur mobilisant un « cliché » qu'il aurait entendu ou joué lui-même dans un contexte différent.

La section 4 décrit la seconde étape d'une phase de génération : la navigation dans la mémoire en cherchant à allier la continuité du discours musical et la possibilité de s'éloigner du matériau original.

La section 5 évoque ensuite certains des paramètres algorithmiques permettant à un opérateur-musicien pilotant le système de contrôler la génération au cours de l'improvisation, ainsi que de définir des heuristiques appropriées au cadre idiomatique.

La section 6 expose enfin la mobilisation dynamique du modèle en temps réel dans un contexte de performance et décrit l'architecture au sein de laquelle ce processus de génération est intégré. L'ordonnancement des appels au modèle permet d'allier l'anticipation permise par la connaissance a priori d'un scénario et la réactivité nécessaire à l'improvisation par la mobilisation dynamique de processus statiques.

2. Modèle d'improvisation automatique guidée

2.1. Scénario et mémoire structurée

Le processus d'improvisation guidée est modélisé comme l'articulation entre une séquence de référence et une mémoire structurée et annotée dans laquelle on viendra rechercher des séquences musicales qui seront transformées et réagencées pour créer les improvisations :

– le *scénario* correspond à une séquence symbolique de référence guidant l'improvisation et définie sur un alphabet adapté au contexte musical,

– la *mémoire* à une séquence de contenus étiquetée par une séquence symbolique définie sur le même alphabet que le scénario.

Dans ce contexte, « improviser » signifiera parcourir la mémoire pour collecter des blocs parmi ses contenus musicaux et les concaténer pour créer une phrase musicale en suivant la séquence d'étiquettes imposée par le scénario. À la manière d'un improvisateur employant un motif qu'il a entendu ou joué lui-même dans un contexte différent mais présentant localement une progression commune avec le nouveau contexte dans lequel il joue, il s'agit donc de trouver des segments de la mémoire correspondant aux portions successives du scénario à suivre et de les enchaîner. Les séquences du scénario et de la mémoire étant différentes dans le cas général, il faut donc exploiter

au mieux leurs motifs communs pour extraire et assembler les segments de la mémoire les plus longs possible ou répondant à d'autres critères imposés traduisant des choix musicaux, tout en évitant l'écueil de la simple recopie dans le cas où les deux séquences présentent une grande similarité.

2.2. Notations

Le scénario et la séquence de *labels* étiquetant la mémoire sont représentés comme des mots sur un même alphabet. Le choix d'un alphabet fixe les classes d'équivalences des éléments musicaux constituant la mémoire. Cette mémoire peut être instantanée, apprise à la volée au cours d'une performance, ou faire appel à un corpus appris antérieurement. On suppose donné un scénario S de taille s dont on notera $S[T]$ la lettre d'indice T. Ayant choisi au préalable une unité temporelle pour le découpage des segments, $S[T]$ correspond à la classe d'équivalence souhaitée pour l'improvisation à produire au temps T. Étant donnée une mémoire M de taille m, la lettre $M[P]$ correspond à la classe d'équivalence étiquetant le fragment musical correspondant au temps P. Par la suite, on confondra le mot M constitué par la séquence de labels qui étiquette la mémoire avec son contenu. Dans les rares cas où la distinction sera nécessaire, les labels constituant le mot M seront notés en lettres minuscules, et les contenus associés en lettres majuscules : on notera que selon l'alphabet choisi pour décrire la mémoire et le scénario, différents contenus élémentaires A, A', A'',... seront regroupés dans une même classe d'équivalence puisqu'ils seront étiquetés dans M par un même label a. Enfin, on notera $\{i_T\}_{0 \leq T < s}$, la suite des indices des états de M à concaténer pour produire l'improvisation.

2.3. Principe général

L'existence d'un scénario donne accès à une connaissance de la structure temporelle de l'improvisation à produire qui sera mise à profit pour prendre en compte les futures classes demandées aux dates $T+1$, $T+2$,... pour la génération du temps T. Le *scénario courant* à la date T, noté S_T, correspond au suffixe du scénario original débutant à la lettre d'indice T, soit $S[T]...S[s-1]$. À chaque instant, on poursuit l'improvisation depuis le dernier état lu dans la mémoire à la date $T-1$ en respectant la contrainte donnée par le scénario courant. Le modèle d'improvisation proposé repose sur des *phases successives de navigation dans la mémoire* [1]. Une phase de navigation dans la mémoire est constituée de deux *étapes* traduisant les deux préoccupations de continuité et de renouvellement introduites précédemment, et exploitant respectivement les caractéristiques de la structure du scénario et de la mémoire :

1. Rechercher un *point de départ* dans la mémoire (étapes « 1 », Figure 1),
2. Suivre des *enchaînements conformes au scénario* (étapes « 2 », Figure 1).

1. Ces phases de navigation dans la mémoire segmentent l'improvisation dans son processus algorithmique de production, et ne correspondent pas en général à des « phrases » musicales distinctes.

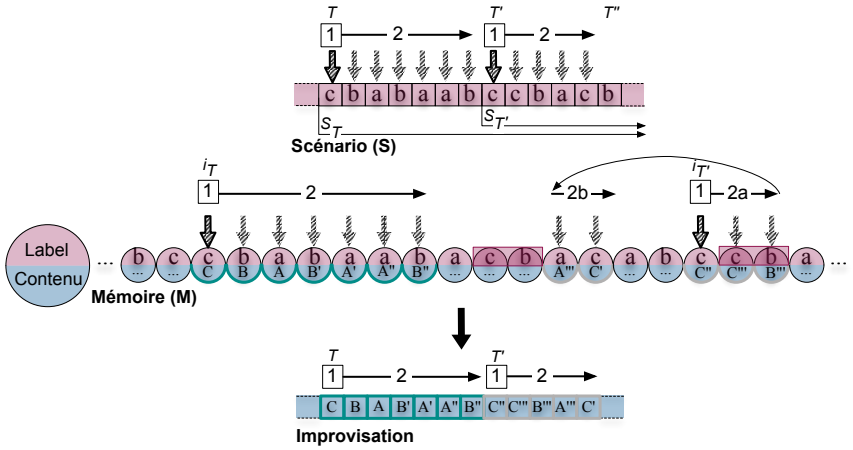

Figure 1. Improviser en articulant une mémoire annotée et un scénario : exemple de deux phases de navigation débutant respectivement en T et T'

La Figure 1 donne un exemple schématisant deux phases successives, de T à $T'-1$ puis de T' à $T''-1$. La première phase (débutant en T) cherche à générer l'improvisation à partir de la date T sur le scénario courant S_T, suffixe de S. À l'issue de cette première phase, on a pu traiter le préfixe $S[T]...S[T'-1]$ du suffixe S_T de S, et une nouvelle phase de recherche sur le suffixe $S_{T'}$ de S doit donc être lancée à partir de la date T' pour compléter l'improvisation jusqu'à une date $T''-1$.

2.4. Phase de génération sur le scénario courant S_T

2.4.1. Étape 1 : recherche d'un point de départ dans la mémoire

Chaque phase est amorcée par la recherche d'un motif initial conforme à une section du scénario courant dans la mémoire, soit un préfixe de S_T dans M. Le premier élément de ce motif fournit un *point de départ* pour la phase de navigation : un état de la mémoire à partir duquel on est assuré de pouvoir trouver au moins une séquence correspondant à une partie du scénario courant, dont en premier lieu le motif trouvé lui-même. Cette première étape est détaillée dans la section 3.

L'ensemble des indices i des états candidats pour constituer un point de départ pour la génération de l'improvisation à partir de la date T est défini par :

$$D_T = \{i < m \mid \exists\, c_f \geq 0,\, M[i]...M[i+c_f] \in \text{ Préfixes}(S_T)\}$$

où c_f représente la *continuité par rapport au* ***futur*** *du scénario*. c_f mesure la durée de la séquence de la mémoire qui pourra être littéralement recopiée tout en respectant le

scénario à partir de l'état d'indice i choisi. L'exemple ci-dessous reprend la première étape de la phase de navigation débutant en T en Figure 1. En numérotant les lettres à partir de 0, on a par exemple pour la première phase :

$$M = b\underline{C}\,\underline{C}babaab\underline{a}\underline{Cba}\,\underline{C}ab\underline{C}\,\underline{Cba}$$
$$S_0 = cbabaabccbacb$$
$$D_0 = \{1, 2, 10, 13, 16, 17\}$$

où D_0 contient les indices des positions de M en lesquelles débute un préfixe de S_0. Les lettres correspondantes figurent en majuscule dans M, et les préfixes qu'ils débutent sont soulignés. Leurs longueurs donnent les valeurs de c_f associées.

2.4.2. Étape 2 : suivre une suite d'enchaînements satisfaisant le scénario

À partir du point de départ choisi, on cherche à parcourir la mémoire en suivant une séquence d'*enchaînements conformes au scénario courant*. Le chemin suivi dans la mémoire passe par des états dont les labels correspondent au scénario et dont les contenus musicaux sont concaténés pour former l'improvisation. Lorsque cela est possible, une solution triviale est de suivre linéairement le motif de la mémoire choisi à l'étape précédente. Cependant, les perspectives peuvent être élargies en tirant profit d'une analyse de la structure de la mémoire. Dans le cas général, si S et M sont différents et si aucun n'est facteur de l'autre, il s'agira d'exploiter au mieux non seulement leurs motifs communs mais également leurs similarités internes pour tenter de créer un discours musical continu en prolongeant le parcours au-delà des motifs communs. Cette volonté de pouvoir s'éloigner du matériau initial nous poussera à élargir l'ensemble des enchaînements candidats après la lecture d'un état dans la mémoire.

On envisagera ainsi les *enchaînements conformes au scénario* possibles à partir d'un état donné d'indice k_0 comme l'*ensemble des états de la mémoire présentant un passé commun avec cet état et satisfaisant la prochaine contrainte imposée par le scénario*. On définit donc E_{T,k_0}, l'ensemble des indices k des états de la mémoire M conformes avec le temps T du scénario S et pouvant constituer un enchaînement avec l'état précédent $M[k_0]$, par :

$$\begin{aligned} E_{T,k_0} = \{k < m \mid \exists\, c_p \in [1, k],& \\ & M[k - c_p]...M[k-1] \in \text{Suffixes}(M[0]...M[k_0]), \text{ et} \\ & M[k] = S[T]\} \end{aligned}$$

Quand $M[k_0 + 1] = S[T]$, donc $M[k_0 + 1] \in E_{T,k_0}$, le motif commun à M et S peut être suivi linéairement en choisissant l'enchaînement fourni par l'état suivant de la mémoire. Pour éviter une simple recopie de séquences tout en préservant une certaine continuité musicale, on autorise donc un parcours non-linéaire si les sauts effectués se font entre deux segments présentant un passé commun, c'est-à-dire si les facteurs de la mémoire se terminant aux points de départ et d'arrivée de ces sauts présentent un suffixe commun. La longueur c_p de ce suffixe commun quantifie ainsi la *continuité par*

*rapport au **passé** de la mémoire*. Cette seconde étape, qui sera détaillée dans la section 4, s'arrête lorsqu'aucun enchaînement n'est possible, c'est-à-dire quand $E_{T,i_{T-1}} = \emptyset$.

L'exemple ci-dessous reprend un pas de la deuxième étape de la phase de navigation débutant en T' sur la Figure 1 (2a-2b). En numérotant les lettres à partir de 0, on a pour l'ensemble $E_{10,18}$:

$$M = \underline{bccb}A\,\underline{b}Aa\underline{b}A\,\underline{cb}Acabcc\boldsymbol{b}A \qquad M[0]...M[18] = b...bcc\boldsymbol{b}$$
$$S = cbabaabccb\boldsymbol{a}cb \qquad S[10] = \boldsymbol{a}$$
$$E_{10,18} = \{4, 6, 9, 12, 19\}$$

$E_{10,18}$ contient les indices des lettres de M égales à $S[10] = a$, et précédées d'un suffixe de $M[0]...M[18]$. Les lettres correspondantes figurent en majuscule dans M, et les suffixes de $M[0]...M[18]$ qui les précèdent sont soulignés (sauf $M[0]...M[18]$ lui-même pour faciliter la lecture). Leurs longueurs donnent les valeurs de c_p associées.

2.5. Articulation des phases

L'algorithme 1 résume le schéma général du processus d'improvisation. Les phases successives de navigation dans la mémoire correspondent à la boucle principale de l'algorithme. L'étape de recherche d'un point de départ correspond donc aux lignes 2-3, où l'on construit D_T au sein duquel il sera choisi. L'étape de navigation correspond à la boucle interne (lignes 4-6), où la mémoire est parcourue en choisissant des positions i_T dans les $E_{T,i_{T-1}}$ successifs jusqu'à ce qu'aucun enchaînement ne soit plus possible.

Si le schéma de l'algorithme 1 peut s'exécuter linéairement en suivant littéralement la boucle principale jusqu'à ce que la fin du scénario soit atteinte pour produire d'un bloc une phrase musicale répondant à la spécification donnée par le scénario, il peut également être segmenté en différents processus concurrents correspondant à des phases de navigation couvrant des sections du scénario qui se chevauchent. Ces aspects d'ordonnancement visant à allier l'anticipation permise par la connaissance du scénario et la possibilité d'avoir des contrôles réactifs lors de la performance sont traités dans la section 6.

2.6. Contrôles, filtrages, et heuristiques

Différents paramètres permettent de piloter la génération des séquences produites par le modèle, notamment lors des définitions des ensembles D_T et E_{T,k_0} ou des choix des positions à jouer parmi les positions candidates (« $i_T \leftarrow i \in D_T$ » et « $i_T \leftarrow i \in E_{T,i_{T-1}}$ », algorithme 1). Ces paramètres permettent de restreindre ces ensembles en filtrant les solutions, ou au contraire de les étendre : la généricité du modèle permet en effet de définir des critères d'appariement et de comparaisons en terme d'équivalences propres au contexte idiomatique, et donc à l'alphabet utilisé (en

Algorithme 1. Algorithme principal d'improvisation

Entrées :
S, scénario de taille s
M, mémoire de taille m
Résultat :
$\{i_T\}_{0 \leq T < s}$, indices des états de M à concaténer pour produire l'improvisation

Initialisation :
$T = 0$
Tant que $T < s$ **faire**
  $i_T \leftarrow i \in D_T$
  $T \leftarrow T + 1$
  Tant que $(T < s) \& (E_{T,i_{T-1}} \neq \emptyset)$ **faire**
    $i_T \leftarrow i \in E_{T,i_{T-1}}$
    $T \leftarrow T + 1$

cas d'échec d'une recherche, la marche à suivre peut également être définie par des heuristiques pertinentes spécifiques au contexte).

Parmi les contraintes utilisées pour filtrer les solutions qui seront abordées dans cet article, on peut citer la localisation du motif dans la mémoire, la préférence pour un segment assurant la meilleure continuité avec le motif produit lors de la phase précédente, ou encore des intervalles imposés pour les paramètres c_f et c_p. A titre d'exemple, exiger la valeur maximale pour c_f entraînera la recherche d'un préfixe de taille maximale et assurera d'avoir un résultat homogène et continu sur sa durée ; à l'inverse une petite valeur impliquera des extractions de courts segments dans des zones potentiellement différentes de la mémoire : selon les situations musicales, on pourra successivement préférer un état assurant une grande cohérence du discours musical à long terme et très proche de la séquence d'apprentissage, ou au contraire un résultat plus morcelé. Ces contrôles peuvent être confiés à l'opérateur-musicien et/ou paramétrisés automatiquement par l'écoute des entrées musicales au cours de l'improvisation. Si l'ensemble des solutions n'est pas réduit à un seul élément une fois les filtrages opérés, la position i_T de la mémoire utilisée pour devenir le temps T de l'improvisation est tirée aléatoirement parmi les positions restantes. Des exemples de tels contrôles et heuristiques sont développés en section 5.

3. Recherche de préfixes du scénario courant dans la mémoire

En supposant l'improvisation déjà complète jusqu'à la date précédente $T - 1$ à laquelle s'est terminée une phase de navigation, on détaille dans cette section la re-

cherche d'un point de départ dans la mémoire pour créer un fragment d'improvisation commençant au temps T. La recherche est donc guidée par le suffixe [2] S_T du scénario original S=S_0 qui contient la séquence des classes d'équivalences $S[T]$, $S[T+1]$,... imposées pour tous les temps à partir de T. Les préfixes de S_T dans M sont indexés afin de choisir celui qui fournira l'état de départ en fonction du système de contraintes imposé pour le filtrage des solutions (longueur du préfixe, c_f, localisation dans la mémoire...). Le parcours à partir de cette date T, décrit en section 4, se développera ensuite jusqu'à une date $T' - 1$ à laquelle se présentera à nouveau la nécessité d'une recherche de point de départ, donc de préfixes pour le prochain suffixe du scénario considéré $S_{T'}$.

3.1. Définitions

On se ramène au problème général de l'indexation de tous les préfixes d'un mot X=$X[0]...X[x-1]$ dans un texte Y=$Y[0]...Y[y-1]$. Cette recherche vise à construire une table dont les entrées sont les longueurs des préfixes de X trouvés dans Y, associées aux positions de Y auxquels ces préfixes débutent. Elle s'appuie sur l'algorithme introduit dans (Morris, Pratt, 1970) qui décrit le parcours de l'automate déterministe reconnaissant le langage A^*X (où A est l'alphabet sur lequel sont définis X et Y) en suivant le chemin étiqueté par les lettres de Y. Cet algorithme permet ainsi de trouver toutes les occurrences du mot X dans Y, et utilise pour cela une fonction de suppléance calculée au préalable sur le mot X indexant ses similarités internes qui seront exploitées pour éviter les comparaisons inutiles. L'algorithme d'indexation des préfixes de X dans Y décrit dans cette section se décompose en une analyse préalable du mot X et la localisation en elle-même : lors de la localisation, les plus longs préfixes du mot X terminant à chaque position du texte Y sont indexés (voir 3.2), et les éventuels préfixes de longueurs inférieures qu'ils contiendraient sont déduits grâce aux calculs préalables effectués pendant la phase d'analyse de X qui construit la fonction de suppléance (voir 3.3).

– En reprenant le vocabulaire usuel, on dira qu'un mot est un facteur *propre* de X s'il est facteur de X et différent de X.

– Un *bord* d'un mot non vide X est un facteur propre de X qui est à la fois un préfixe et un suffixe de X.

– On notera f la fonction de suppléance issue de l'algorithme de Morris et Pratt. Elle est définie par :

$$\begin{cases} f(0) = -1 \\ f(i) = \text{longueur du plus long bord de } X[0]...X[i-1] \text{ pour } i \in [1, x] \end{cases}$$

2. Dans le cas de scénarios cycliques, par exemple pour une improvisation jazz basée sur une grille harmonique dont on ne connaît pas au préalable le nombre de répétitions, la même méthode de recherche sera appliquée à des permutations circulaires successives du scénario plutôt qu'à des suffixes.

f est construite avant la phase de localisation lors d'une phase d'analyse de X. Pour la construction de la table des bords et de f, se reporter à (Crochemore *et al*., 2001).

$$X = \underline{abab}cab\underline{abab}\mathbf{b}c$$
$$f(11) = 4$$

Dans le cas de l'exemple ci-dessus, si lors d'une étape de la localisation des occurrences de X dans un texte Y on compare $X[11]$ = b à une lettre du texte $Y[j]$ et que celle-ci est différente, on comparera ensuite $Y[j]$ à $X[f(11)] = X[4]$ = c. On sait en effet que $X[11]$ = b est précédée par une séquence de taille $f(11) = 4$ (abab) qui est également préfixe du mot et qui vient donc d'être trouvée dans Y si $X[11]$ a été atteint.

– On note $P(i)$ = {longueur de u, u est un bord de $X[0]...X[i]$}.

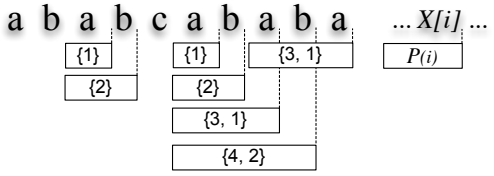

Figure 2. Exemples de $P(i)$, ensemble des longueurs des bords de $X[0]...X[i]$

Dans l'exemple de la Figure 2, le dernier b en position $j = 8$ termine la séquence abab de longueur 4 qui est préfixe du mot, et contenant elle-même le préfixe ab de longueur 2. On obtient donc pour cette lettre $P(8) = \{4, 2\}$.

$P(i)$ est par définition l'ensemble des longueurs des bords de $X[0]...X[i]$ et donne donc immédiatement l'ensemble des longueurs des préfixes de X se terminant en position i dans X. Pour tout $i \in [1, x-1]$, pour tout $L \in P(i) \neq \emptyset$, i est donc la position de fin d'un préfixe de X de longueur L. Finalement pour tout $i \in [1, x-1]$, pour tout $L \in P(i) \neq \emptyset$, un préfixe de X de longueur L débute dans X en position $(i - L + 1)$. La connaissance des $P(i)$ donne ainsi la localisation des préfixes de X dans lui-même qui sera utilisée dans l'algorithme de reconnaissance des préfixes de X dans Y détaillé en 3.2 et 3.3.

3.2. Reconnaissance du plus long préfixe de X se terminant en une position donnée de Y

L'algorithme d'indexation des préfixes de X dans Y présenté en algorithme 2 et illustré en Figure 3 reprend le déroulé de l'algorithme de Morris et Pratt : les caractères du mot et du texte aux positions courantes respectives i et j sont comparés un à un

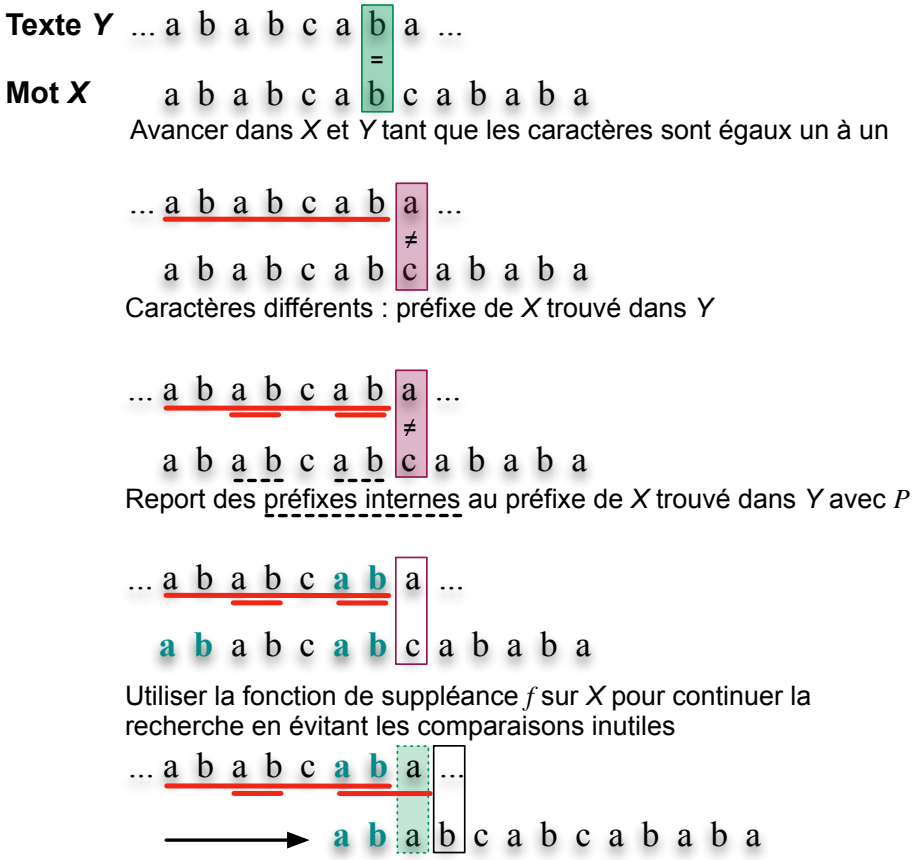

Figure 3. Indexation des préfixes d'un mot X dans un texte Y avec f et P données

(ligne 2, algorithme 2) et les compteurs sont incrémentés tant qu'ils sont égaux (étape 1, Figure 3 / lignes 8-9, algorithme 2), l'indice i de la position dans le mot fournissant la longueur du motif reconnu à ce stade. Si la lettre $X[i]$ du mot est différente de la lettre courante du texte $Y[j]$, cette dernière est ensuite comparée à $X[f(i)]$, la lettre suivant le plus long bord de $X[0]...X[i-1]$ de longueur $f(i) < i$ ayant donc été trouvé dans le texte. Remonter ainsi la fonction de suppléance dans le mot jusqu'à un état auquel on pourra lire le caractère courant $Y[j]$ du texte permettra de reprendre les comparaisons après le plus long préfixe de X que l'on sait avoir déjà trouvé dans Y et se terminant en $Y[j-1]$ pour éviter les comparaisons inutiles (étape 4, Figure 3 / lignes 6-7, algorithme 2). Enfin, quand i atteint la taille du mot, une occurrence complète de X est trouvée à la position gauche $(j-i)$ dans Y (ligne 11, algorithme 2).

Algorithme 2. Reconnaissance des préfixes d'un mot X dans un texte Y

Entrées :
X, mot de longueur x
f, fonction de suppléance pour X
Y, texte de longueur y
Résultat :
Localisation et longueurs des préfixes de X trouvés dans Y

Initialisation :
$i = 0\,;\, j = 0$
1 **Pour** $j \in [0, y-1]$ **faire**
2 **si** $(i > -1)\&(Y[j] \neq X[i])$ **alors**
3 **si** $i > 0$ **alors**
4 Préfixe de X de longueur i trouvé : $Y[j-i]...Y[j-1]$
5 *Ajouter les préfixes de longueur* $< i$ *entre* $(j-i)$ *et* $(j-1)$
6 **Tant que** $(i > -1)\&(Y[j] \neq X[i])$ **faire**
7 $i \leftarrow f(i)$
8 $i \leftarrow i + 1$
9 $j \leftarrow j + 1$
10 **si** $i > x - 1$ **alors**
11 Occurrence de X trouvée : $Y[j-i]...Y[j-1]$
12 *Ajouter les préfixes de longueur* $< x$ *entre* $(j-i)$ *et* $(j-1)$
13 $i \leftarrow f(i)$
14 **si** $i > 0$ **alors**
15 Préfixe de X de longueur i trouvé : $Y[j-i]...Y[j-1]$
16 *Ajouter les préfixes de longueur* $< i$ *entre* $(j-i)$ *et* $(j-1)$

Pour passer de la reconnaissance des occurrences du mot X dans le texte Y à celle des préfixes de X dans Y, il suffit d'effectuer quelques enregistrements supplémentaires. Au cours de ce processus, un préfixe de X est balayé dans Y :

– quand les deux caractères courants sont différents alors que les précédents étaient égaux : si $Y[j] \neq X[i]$ et $i > 0$, on a trouvé dans Y un préfixe de X de taille i se terminant en $j-1$ et débutant donc en $j-i$ (étape 2, Figure 3 / ligne 4, algorithme 2),

– quand la recherche atteint la fin du texte avec $i > 0$: un préfixe de taille i était en train d'être reconnu (ligne 14, algorithme 2),

– quand une occurrence complète de X est trouvée dans Y.

Par définition de la fonction de suppléance f, le retour en arrière effectué dans le mot en cas d'échec à une étape (i, j) est tel qu'un préfixe reconnu se terminant à l'indice $(j-1)$ dans un des trois cas listés ci-dessus est le plus long préfixe terminant en cet indice, et que les indices du texte $(j-i)...(j-1)$ sur lesquels il s'étend ne seront plus parcourus lors de la recherche. Afin que le retour en arrière soit valide, il reste donc à rechercher les éventuels préfixes qui seraient contenus par ce plus long préfixe avant de remonter un chemin de suppléance. Ainsi, si à l'étape (i, j) comparant $X[i]$ et $Y[j]$ un préfixe de X de taille i a été reconnu en position $(j-i)$ dans Y, la sous-routine présentée en algorithme 3 reporte les préfixes internes à $X[0]...X[i-1]$ dans $Y[j-i]...Y[j-1]$ (étape 3, Figure 3 / lignes 5, 12, et 16, algorithme 2).

Algorithme 3. Sous-routine : préfixes de longueur $< i$ entre $(j-i)$ et $(j-1)$

```
1 Pour i' ∈ [1, i − 1] faire
2     Pour L ∈ P(i') faire
3         Préfixe de X de longueur L trouvé dans Y
4         débutant en (j − i) + (i' − L + 1)
```

En effet, si pour $i' \in [1, i-1]$, $L \in P(i') \neq \emptyset$, un préfixe de X de longueur L débute dans X en position $(i'-L+1)$ (voir 3.1), il suffit pour obtenir sa position dans Y de décaler cet indice de $(j-i)$, position à laquelle débute le plus long préfixe de X dans Y de longueur i terminant en position $(j-1)$ trouvé à l'étape (i, j).

3.3. Indexer les préfixes contenus dans le plus long préfixe de X terminant en une position donnée dans Y

En plus de son rôle de fonction de suppléance, la fonction f construite à partir de la table des bords du mot X contient les informations permettant d'obtenir les $P(i)$, longueurs de tous les bords des $X[0]...X[i]$ pour $i \in [1, \text{x-1}]$. Ceci justifie donc la définition de P introduite en 3.1, et son utilisation pour localiser les préfixes de X dans lui-même par leurs positions droites et leurs longueurs pour les reporter ensuite dans Y.

En remarquant que le bord d'un bord de X est lui-même un bord de X, et en procédant par récurrence (Crochemore *et al.*, 2001), on obtient que pour un mot X non vide et n le plus grand des entiers k pour lequel $f^k(X)$ est défini (soit $f^n(X) = 0$), $f(X)$,$f^2(X)$,...,$f^n(X)$ est la suite des longueurs des bords de X classées par ordre décroissant. En remontant le chemin de suppléance à partir d'une position i dans le mot X tant que $f^n(i)$ est défini, on obtient donc :

$$P(i-1) = \{f(i),\, f^2(i),\, ...,\, f^n(i) > 0\} \text{ pour } 1 < i < x$$

Un exemple de l'indexation des préfixes d'un mot X terminant en un indice donné grâce à f est donné en Figure 4.

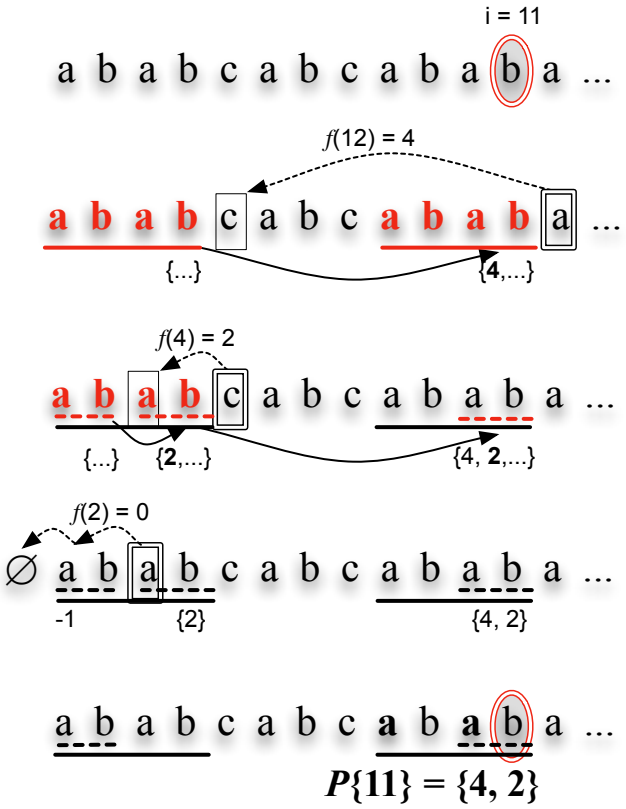

Figure 4. Exemple : obtenir $P(i)$, l'ensemble des longueurs des bords de $X[0]...X[i]$ à partir de $f(i+1)$, $f^2(i+1)$,...$f^n(i+1)$

Algorithme 4. Sous-routine : préfixes de longueur $< i$ entre $(j-i)$ et $(j-1)$

1 **Pour** $i' \in [1, i-1]$ **faire**
2 **Pour** $f^n(i'+1), n \in \mathbb{N}^*$ défini **faire**
3 Préfixe de X de longueur $f^n(i'+1)$ trouvé dans Y
4 débutant en $(j-i)+(i'-f^n(i'+1)+1)$

La sous-routine présentée en algorithme 3 et appelée aux lignes 5, 12, et 16 de l'algorithme 2 revient donc finalement à l'algorithme 4. Pour éviter les informations redondantes, seuls les plus longs préfixes débutant à chaque indice sont enregistrés (si

$X[i]$ est la première lettre d'un préfixe de longueur L, il l'est bien sûr également de préfixes de longueurs 1 à L). Finalement, la phase d'analyse consiste en la construction de f sur X (Crochemore *et al*., 2001) comme pour l'algorithme de Morris et Pratt.

La phase de localisation procède ensuite simplement à des enregistrements additionnels de certaines positions non retenues par ce dernier sans avoir à effectuer de comparaisons ou de retours supplémentaires par rapport à l'algorithme de départ dont la phase de localisation s'exécute en temps $\Theta(y)$ sans excéder plus de $2y - 1$ comparaisons entre des lettres de X et de Y, et dont la phase d'analyse s'effectue en temps $\Theta(x)$. Si sa fonction de suppléance n'est pas optimisée, par exemple relativement à celle de (Knuth *et al*., 1977) ou de (Boyer, Moore, 1977), le choix de l'algorithme de Morris et Pratt et de sa fonction de suppléance comme point de départ de l'algorithme de reconnaissance des préfixes se justifie par le fait qu'il en résulte une implémentation simple due au lien direct entre bords et préfixes, et une rapidité d'exécution tout à fait suffisante pour le cas d'application (ce résultat est empirique : le logiciel ImproteK au sein duquel le modèle est implémenté à été utilisé à de nombreuses reprises lors de sessions d'improvisation ou de concerts depuis 2012) [3]. Enfin, ce choix est motivé par le souci de traiter simplement la question musicale par un processus informatique cherchant en premier lieu une cohérence métaphorique avec la réalité de l'improvisation, en cherchant à ce que les opérations informatiques puissent correspondre à des opérations réellement pratiquées par des musiciens.

4. Parcours linéaire ou non-linéaire de la mémoire

L'index D_T des préfixes du scénario courant S_T dans la mémoire M est obtenu au début de chaque nouvelle phase en appliquant l'algorithme décrit dans la section précédente pour $X = S_T$ et $Y = M$. Il offre une cartographie des facteurs équivalents dans ces deux séquences, et ainsi une représentation de leurs similarités structurelles. Le choix du segment à retourner en fonction du système de contraintes imposé relève ensuite du choix musical et sort du champ algorithmique. La section suivante supposera donc le segment de la mémoire choisi parmi les candidats de D_T (ce choix sera abordé en 5.2 à travers les liens entre les paramètres algorithmiques et les paramètres de jeu musical dans le cas d'une improvisation jazz).

4.1. Utiliser les similarités internes de la mémoire

La progression à travers la mémoire peut simplement consister à suivre linéairement la séquence sélectionnée correspondant à un préfixe du scénario courant, soit à choisir à chaque pas $i_T = i_{T-1} + 1$ tant que cela est possible (si $M[i_{T-1} + 1] = S[T]$, $M[i_{T-1} + 1] \in E_{T,i_{T-1}}$). Cependant, en utilisant les caractéristiques de la mémoire,

3. Le premier terrain d'expérimentation musicale avec le modèle ayant été les « grilles » de standards de jazz, les séquences traitées comportaient de nombreuses régularités. En effet, du fait du rythme harmonique généralement à la blanche ou à la ronde celles-ci sont souvent de la forme $...x^4y^2z^2x^8...$ l'algorithme naïf aurait donc été fortement sous-optimal.

on peut chercher à prolonger le parcours le plus continûment possible, ou à l'inverse à altérer la fidélité au matériau d'origine pour proposer une séquence nouvelle. Si la première option sera généralement préférée dans les cas où les facteurs communs au scénario et à la mémoire sont courts, la deuxième sera à l'inverse plus souhaitable si le scénario à suivre S et la séquence étiquetant la mémoire M sont similaires. Après avoir cherché les séquences littérales présentes dans la mémoire, on recherche donc maintenant des enchaînements correspondant à la suite du scénario qui ne se trouveraient pas en l'état dans la mémoire mais qui préserveraient la continuité du discours musical. Pour ce faire, la mémoire est apprise dans une structure d'automate permettant de repérer ses similarités internes au fur et à mesure de sa construction : l'oracle des facteurs (Allauzen *et al*., 1999 ; Lefebvre *et al*., 2002).

L'oracle des facteurs a été introduit pour pallier la difficulté de construire simplement un automate fini déterministe reconnaissant exactement le langage constitué par l'ensemble des facteurs d'un mot. L'oracle des facteurs construit sur un mot X est un automate fini déterministe reconnaissant tous les facteurs de X, ainsi que quelques mots supplémentaires. Dans le cadre d'une application musicale, cette structure d'automate présente l'avantage de conserver l'aspect séquentiel de la mémoire musicale structurée temporellement, et de ne pas agglomérer en un seul état toutes les instances regroupées au sein d'une même classe d'équivalence. De plus son algorithme de construction étant incrémental et linéaire en la longueur de la séquence en temps comme en espace (se reporter à (Allauzen *et al*., 1999) pour l'algorithme de construction), il est tout à fait adapté à des applications temps réel (voir (Assayag, Bloch, Chemillier, Cont, Dubnov, 2006) pour l'introduction de cette structure dans la problématique de la *réinjection stylistique* pour l'improvisation musicale).

Comme la plupart des systèmes utilisant l'oracle des facteurs pour des applications d'improvisation musicale (Assayag, Bloch, Chemillier, Cont, Dubnov, 2006 ; Bonnasse-Gahot, 2013 ; Surges, Dubnov, 2013 ; François *et al*., 2013 ; Donze *et al*., 2013) on en fait ici une utilisation détournée en exploitant des liens de constructions pour introduire de la « surprise » par rapport à la séquence d'origine, et non pas pour procéder à une recherche de motifs à proprement parler. On utilise en effet sa fonction de construction, la *fonction de lien suffixiel*, pour indexer des similarités internes à la séquence étiquetant la mémoire. La fonction de lien suffixiel appliquée en position i d'un mot X pointe sur l'état d'arrivée du plus long suffixe propre de $X[0]...X[i]$ répété à gauche. Les liens suffixiels repèrent ainsi des motifs répétés au sein de la séquence et assurent l'existence d'un suffixe commun entre les éléments qu'ils relient. Ce contexte commun aux deux zones de la mémoire représente donc ici un passé musical commun. Le postulat au cœur des modèles musicaux utilisant l'oracle des facteurs est qu'un parcours non linéaire d'une mémoire ainsi cartographiée permet de créer des phrases musicales se renouvelant par rapport au matériau d'origine tout en préservant la continuité du discours musical.

4.2. Parcours guidé de la mémoire : suivre des enchaînements satisfaisant le scénario

La navigation contrainte présentée dans cette section adapte l'heuristique de parcours d'un oracle des facteurs présentée dans (Assayag, Bloch, 2007) à l'improvisation guidée. On entend par navigation contrainte le fait de suivre dans l'automate un chemin dont les transitions sont imposées par les lettres du scénario à partir de l'état de départ préalablement sélectionné. Selon l'évolution des différents paramètres de continuité et des contraintes qui leurs sont imposées, elle alterne entre des phases de progression linéaire n'employant que des transitions reliant deux états consécutifs dans la séquence originale, et des sauts en suivant des liens suffixiels. Pour essayer de poursuivre le parcours une fois le facteur initial recopié, si aucune transition indexée par le label souhaité n'est trouvée à partir de l'état courant, on cherche à suivre un lien suffixiel permettant d'atteindre un autre état où l'on pourra éventuellement lire l'étiquette courante. Lorsque l'on effectue de tels sauts, les liens suffixiels garantissent qu'il existe dans la séquence d'origine une partie commune entre les deux fragments concaténés.

Afin de connaître la longueur du *contexte* commun entre deux évènements musicaux dans une séquence (Assayag, Dubnov, 2004) fait appel à la méthode présentée dans (Lefebvre, Lecroq, 2000) pour le calcul du vecteur *lrs*, linéaire en temps et en mémoire, en l'intégrant dans l'implémentation de la construction de l'oracle. La longueur de ce contexte correspond au paramètre c_p introduit en 2.4.2. Chaque état de l'oracle mémoire en position k_0 stocke donc une table de positions de la mémoire avec lesquels il partage un suffixe/passé musical commun, et un sous-ensemble de E_{T,k_0} contenant les états candidats pour constituer un enchaînement à partir de cet état satisfaisant le scénario au temps T est obtenu en extrayant de cette table les éléments dont le label correspond à $S[T]$. Par ailleurs, les résultats engendrés lors des expériences d'utilisation du logiciel menées avec des musiciens montrent en pratique que le sous-ensemble de E_{T,k_0} atteint en suivant un chemin suffixiel vers l'avant ou l'arrière à partir de l'état k_0 suffit pour s'éloigner de la séquence d'origine.

L'exemple donné par la Figure 5 illustre un pas de ce parcours avec le choix de $i_{T+3} \in E_{T+3,i_{T+2}}$. On suppose que l'étape de recherche d'un préfixe du scénario courant a retourné l'état (d, D) (début du préfixe de longueur 3 dbc) comme point d'entrée dans la mémoire et qu'à ce stade, les labels d, b, et c ont été cherchés et trouvés dans l'oracle des facteurs contenant la mémoire musicale. La concaténation des fragments musicaux associés D, B' et C' forme la phrase musicale en construction et la position courante dans l'oracle des facteurs est l'avant-dernier état (c, C'). Le scénario exige ensuite que le label de l'état suivant soit équivalent à a. Aucune transition pointant sur un label correspondant ne pouvant être trouvée, on cherche donc à atteindre une autre zone de la mémoire présentant un passé commun et à partir de laquelle on pourra trouver le label cherché grâce aux propriétés des liens suffixiels. Le lien suffixiel partant de l'état courant (c, C') pointe, par définition, sur l'état final du suffixe répété le plus à gauche du préfixe du mot terminant en (c, C'). (ici bc) soit l'état (c, C). Une transition correspondant au label recherché a est trouvée dans

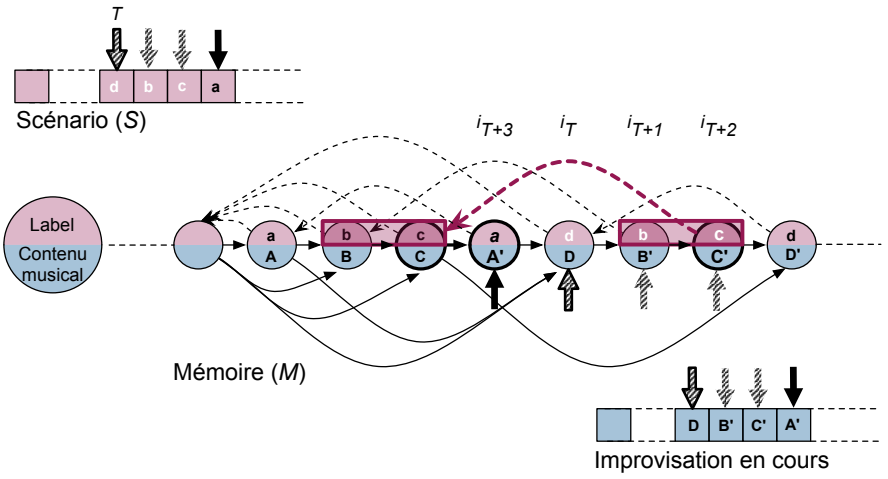

Figure 5. Parcours contraint de la mémoire après choix du point de départ

cet état. Le lien suffixiel est donc finalement suivi pour sauter de (c, C') à (c, C), où 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 le parcours implique des sauts entre différentes zones de l'automate. Ici, ni la suite de labels donnée en entrée ni la phrase en sortie ne se trouvaient inscrites en l'état dans la mémoire.

5. Contrôles, filtrages, et heuristiques

5.1. Paramètres algorithmiques / paramètres de jeu musical : exemples de filtrage

5.1.1. Jouer sur c_f, continuité par rapport au futur du scénario

Le choix du segment de la mémoire parmi les préfixes du scénario courant S_T indexés lors de la phase de recherche est premièrement contraint par l'intervalle de longueur imposé à c_f. On peut en effet vouloir favoriser les facteurs les plus longs offrant l'assurance d'un segment musical d'une cohérence maximale, ou à l'inverse préférer introduire une certaine discontinuité dans l'exploitation de la mémoire musicale. De plus, de la même manière que le contrôle de la « qualité » des sauts décrit plus loin, imposer des contraintes sur leur fréquence et les instants auxquels on les autorise constitue un paramètre de jeu musical permettant de filtrer les solutions.

Le paramètre de continuité par rapport au futur du scénario c_f défini en 2.4.1 prend à l'issue de chaque étape de recherche (section 3) la valeur de la longueur du préfixe du scénario courant choisi dans la mémoire : on sait en effet que la continuité peut être assurée pour les c_f prochaines lettres du scénario. Limiter à chaque instant à une valeur maximale C le nombre de transitions successives suivies dans l'automate mémoire depuis le début de la dernière phase de navigation sans avoir suivi un lien suffixiel, donc la longueur du préfixe du motif recopiée littéralement jusqu'à ce stade, traduit des choix musicaux tendant à rendre prépondérante tantôt la logique musicale du scénario, tantôt celle de la mémoire. Ce paramètre peut-être fixé de manière indépendante ou relativement aux valeurs prises par c_f. Dans ce cas :

– Imposer $C < c_f$ force à suivre un lien suffixiel avant même d'avoir exploité l'intégralité du motif commun au scénario courant et à la mémoire et d'aller donc chercher à poursuivre l'improvisation en puisant dans une autre zone de la mémoire alors que la continuité était encore assurée.

– Quand $C = c_f$, des fragments de la mémoire seront littéralement recopiés et enchaînés, et on pourra par exemple filtrer le choix des motifs après l'étape d'indexation de telle manière qu'ils soient liés par des liens suffixiels, et ainsi rechercher la continuité entre les différents segments issus de chaque phase de navigation.

– Enfin comme dans l'exemple en Figure 5, imposer $C > c_f$ revient à utiliser l'analyse de la structure de la mémoire pour tenter de continuer à suivre le scénario en prolongeant le parcours au-delà des motifs communs.

5.1.2. Jouer sur c_p, continuité par rapport au passé de la mémoire

La structure d'oracle des facteurs de la mémoire peut donc être mise à profit comme dans le cas présenté dans la section précédente pour effectuer des sauts permettant de prolonger le parcours dans la mémoire au-delà de la duplication d'un facteur équivalent à un préfixe du scénario présent littéralement. La « qualité » d'un saut suivant un lien - ou un chemin - suffixiel est évaluée par la longueur du contexte commun entre les états de départ d et d'arrivée a, soit la longueur du suffixe commun entre $M[0]...M[d]$ et $M[0]...M[a]$. On retrouve ainsi la mesure de la continuité par rapport au passé de la mémoire, c_p, introduite en 2.4.2. En imposant des contraintes en temps réel sur ce paramètre, on peut donc à chaque instant quantifier le degré de cohérence souhaité dans l'évolution de la progression : plus on impose à la continuité c_p d'être longue, plus on espère que le fait de s'inscrire dans un passé commun rendra un saut entre deux états imperceptible. À l'inverse, plus on l'autorise à être courte, plus on espère se différencier du matériau d'origine.

5.2. Définir des heuristiques selon l'alphabet : exemple de l'improvisation jazz

Dans le cadre d'application où l'alphabet choisi correspond à des labels harmoniques, on peut par exemple chercher à improviser sur la grille d'un standard de jazz en utilisant un corpus constitué d'un ensemble de morceaux différents. Comme évoqué précédemment, la nature de l'alphabet permet de définir en complément de l'al-

gorithme générique des heuristiques qui lui sont adaptées. Pour cet alphabet, on peut donc par exemple ajouter l'équivalence de séquences à la transposition près ou par transformations définies par des grammaires de substitution d'accords (Chemillier, 2004). En procédant à la recherche décrite en section 3 modulo l'opération de transposition, les motifs communs aux différentes grilles sont repérés au-delà des différences de modes ou de tonalité. On poursuit donc ainsi l'analogie avec un musicien improvisateur cherchant face à une nouvelle grille à réutiliser des « clichés » qu'il aurait pu entendre ou jouer lui-même dans le cadre d'une autre grille présentant localement des enchaînements communs. En favorisant les préfixes les plus longs, on mise ainsi sur le fait que les récurrences de cadences particulières dans les progressions harmoniques de jazz permettront de produire une narration crédible en concaténant des fragments inscrits dans des évolutions différentes.

Cette approche présente également un intérêt dans l'optique de la performance temps réel, particulièrement si le scénario étiquetant la mémoire correspond à la même séquence que le scénario à suivre. En effet, dans un contexte d'improvisation à partir d'une mémoire vide au départ et se nourrissant à la volée des improvisations fournies par les musiciens analogiques qui entourent le système, elle permet dans plusieurs cas de jouer sur des portions de scénario non encore découvertes et pour lesquelles aucune tranche musicale n'a encore été mémorisée. Par exemple, dans le cas d'un morceau présentant des cadences équivalentes dans des modes différents, plus localement des marches harmoniques, ou plus simplement encore des mêmes successions d'accords répétées, il sera possible de jouer sur la suite du scénario en puisant dans la mémoire incomplète apprise jusqu'à ce stade. La définition d'heuristiques pertinentes dans le cas d'un alphabet donné va de paire avec l'apparition de nouveaux contrôles offerts à l'opérateur-musicien lors de la performance : dans le cas de cet exemple on peut tantôt préférer choisir les chemins les plus longs quels que soient les sauts de transposition nécessaires, permettant de recopier les plus longs segments possibles mais présentant le risque d'introduire une discontinuité du discours musical perceptible entre deux portions issues de deux recherches successives; ou bien à l'extrême inverse choisir les chemins minimisant les transpositions quitte à passer à côté d'enchaînements ou de cadences complètes qui seraient présentes dans une autre tonalité locale.

Hors contexte d'improvisation, la construction de D_T pour tous les instants T du scénario offre de plus une représentation de tous les chemins possibles pour parcourir une mémoire en n'autorisant que les facteurs d'un mot-contrainte qu'est le scénario. Le résultat peut ainsi également fournir un outil visuel d'analyse permettant d'observer les similarités et inclusions entre deux séquences. La Figure 6 représente les évolutions possibles dans la grille du standard *Blue in green* (représentée sur trois lignes) en utilisant uniquement les plus longs facteurs issus de la grille d'*Autumn leaves* (dont la grille est rappelée dans la deuxième moitié de la figure à la transposition la plus proche, une case pour une mesure de 4 temps). Chaque flèche pointe donc sur le temps le plus éloigné de la grille de *Blue in green* - le scénario S - pouvant être atteint en recopiant un facteur extrait d'une captation labellisée d'*Autumn leaves* - la mémoire M - dans son état original ou transposée.

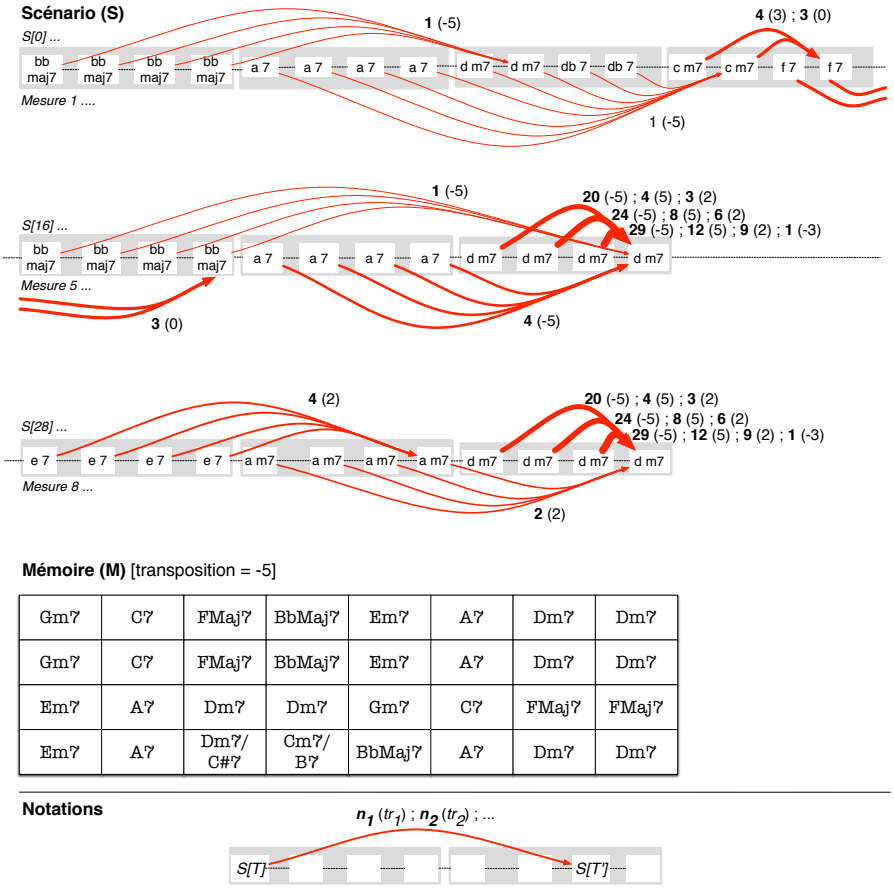

Figure 6. Couvrir la grille de Blue in green (S) avec des facteurs issus d'Autumn leaves (M). S : une case pour un temps, M : une case pour une mesure de 4 temps

L'alphabet commun à ces deux mots est l'ensemble des accords de quatre notes issus de l'harmonisation de la gamme majeure, représentant ainsi le langage le plus utilisé dans les scénarios jazz que sont les grilles de standards. Les nombres en gras étiquetant une flèche (ou un faisceau de flèches) correspondent aux nombres d'occurrences dans M du facteur de S que recouvre la flèche si l'on transpose M du nombre de demi-tons indiqué entre parenthèses (plus le nombre total d'occurrences est élevé, plus la flèche est épaisse). En reprenant les notations de 2.4.1, la figure est obtenue en

reportant dans le scénario pour chaque instant T le nombre d'occurrences à la transposition indiquée des facteurs de M associés à la valeur maximale de c_f pour D_T (soit donc le nombre d'états du scénario recouvert par la flèche partant de la position T).

Même si ne figure en Figure 6 qu'un sous-ensemble très réduit de chacun des D_T et qu'aucune information ne traduit (directement) les $E_{T,k0}$, son observation fournit certains indices quant à la manière dont un opérateur-musicien peut contrôler le système pour improviser sur la grille de *Blue in green* en utilisant, parmi d'autres, une captation d'*Autumn Leaves*. La présence de longues flèches couvrant jusqu'à 10 temps, soit plus de 2 mesures, laisse espérer que sur ces portions de la grille, un résultat musical cohérent pourra être obtenu même si l'on ne se contente pas que de recopier les portions de la mémoire littéralement. De plus, le fait que plusieurs faisceaux de flèches se tuilent en recouvrant des portions communes indique que les enchaînements entre les différents fragments extraits de la mémoire pourront se faire de manière continue musicalement puisqu'il existe des suffixes communs entre les différentes positions de la mémoire concernées. En revanche, aucune flèche ne partant du dernier temps de la mesure 7, la transition avec un des 4 facteurs choisi pour débuter la mesure 8 risque d'être abrupte. De plus, si l'on se contente de concaténer les plus longs segments, la même phrase musicale sera nécessairement jouée sur les mesures 1 à 3 et 5 à 7, puisqu'il n'existe qu'un seul facteur dans M réalisant *BbMaj*7/*A*7/*Dm*7 ... / . Ainsi, si on ne souhaite pas répéter les mêmes motifs musicaux, on pourra donc à ces instants de la grille interdire l'emploi du plus long préfixe, jouer sur les contraintes de filtrages imposées sur c_p pour forcer la recombinaison, contraindre les zones de la mémoire à utiliser pour aller chercher dans un autre chorus enregistré sur *Autumn Leaves*, ou bien encore aller piocher dans une interprétation d'un autre morceau de la mémoire.

Parmi les contrôles principaux pour guider l'improvisation au cours de la performance, on trouve donc en premier lieu le choix de la zone de la mémoire dans laquelle aller chercher les fragments satisfaisant le scénario afin de pouvoir alterner entre la mémoire à court-terme constituée de la musique captée sur l'instant ainsi qu'un corpus constitué au préalable, potentiellement hétérogène, pour pouvoir créer des improvisations « hybrides » [4].

6. Ordonnancer et séquencer pour la performance

6.1. Architecture et mobilisation dynamique du modèle

Le modèle d'improvisation guidée par une séquence de référence constitue une entité autonome utilisable dans une performance pour générer des séquences autonomes correspondant à une spécification donnée, ou dans le cadre d'un processus hors-ligne,

4. Voir vimeo.com/104791211 pour un exemple vidéo extrait d'une captation de concert présentant une improvisation sur une grille harmonique donnée à partir d'une mémoire constituée du matériau capté pendant la performance ainsi que d'un corpus très hétérogène de plus d'une dizaine d'enregistrements de ballades ou de standards de jazz différents interprétés par des musiciens différents.

par exemple dans une démarche de composition. Cette section propose une mobilisation dynamique pour la performance. Les trois aspects de son intégration à un environnement temps réel sont l'apprentissage du matériel musical fourni par les musiciens qui l'entourent, la génération, et le jeu des nouvelles phrases improvisées en synchronisation avec la session d'improvisation en cours. Aux deux éléments centraux du modèle que sont le scénario et la mémoire, s'ajoute un troisième élément : un buffer contenant les *« intentions » de jeu anticipées à court terme*, évoluant en s'affinant au cours du temps.

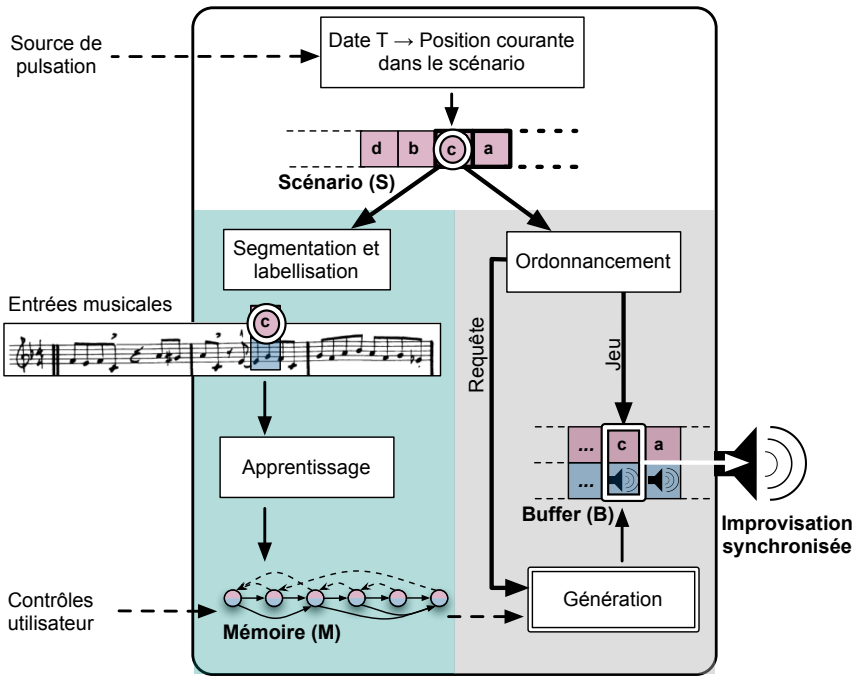

Figure 7. Intégration du modèle dans un environnement de performance

On représente le modèle intégré au sein d'un environnement d'improvisation par le schéma général en Figure 7. L'écoute d'une source de pulsation fournit la date courante T, et donc la classe d'équivalence associée $S[T] = c$ dans le scénario. À chaque mise à jour de la date T:

– Le label associé à la nouvelle position courante dans le scénario $S[T]$ annote la portion du flux musical entendu depuis la date $T - 1$. L'entité formée par ce fragment musical associé à la classe d'équivalence $S[T]$ est apprise à la volée dans l'oracle des facteurs constituant la mémoire de l'improvisation en cours.

– Si ce temps de l'improvisation a déjà été calculé par le modèle et stocké dans le buffer, il est joué en synchronisation avec les autres musiciens.

– Une requête de génération sur un suffixe S_V du scénario ($V \geq T$) est envoyée au modèle si nécessaire.

6.2. Ordonnancement des phases de navigation : anticipations et réécritures

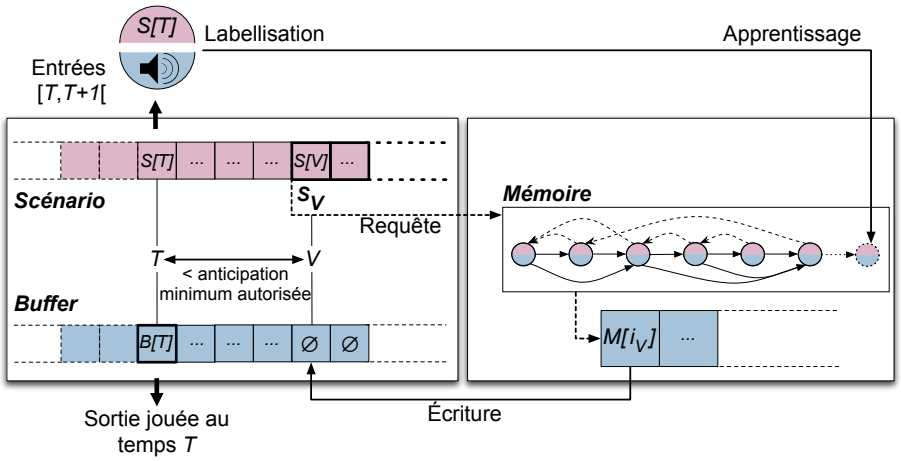

Figure 8. Ordonnancement des requêtes envoyées au modèle

Avec une anticipation sur le temps de la performance, le modèle génère les segments musicaux par phases de navigation comme présenté en Figure 8. Lorsqu'à un temps T une requête de génération associée à un suffixe S_V lui est envoyée, il génère l'improvisation de la date V à la date T' (incluse) à laquelle une nouvelle phase est simplement amorcée. La génération se termine donc par une nouvelle étape d'indexation de préfixes de $S_{T'}$ dans la mémoire M en empiétant sur la phase suivante pour choisir le point de départ de la prochaine étape de parcours de la mémoire qui sera effectuée en temps nécessaire à la réception de la prochaine requête. Le calcul de l'improvisation jusqu'au point de départ de la prochaine phase inclus permet de garder une avance raisonnable afin d'éviter que le temps de la génération ne se laisse rattraper par le temps de la performance. Le temps de calcul nécessaire à l'indexation des préfixes du scénario courant dans la mémoire est en effet supérieur au simple choix d'un enchaînement satisfaisant le prochain temps du scénario.

Algorithme 5. Partition dynamique : ordonnancer les réactions et les appels au modèle

Entrées :
T, date / position courante dans le scénario
S, scénario original à suivre ; S_T suffixe de S commençant à l'indice T
ÉlémentReçu = *(Idx, Contenu)*, nouvel élément reçu, généré par le modèle
Initialisation :
Buffer (contenant les éléments musicaux à jouer) = $\emptyset$
V (indice de la première position vide du buffer) = 0
TempsCourantJoué = faux

1 **Dès que** T mise à jour **faire**
2 Apprendre entrées musicales $[T-1, T[$ étiquetées par $S[T-1]$ dans M
3 TempsCourantJoué ← faux
4 **si** *Buffer*$[T]$ **alors**
5 Jouer(*Buffer*$[T]$)
6 TempsCourantJoué ← vrai

7 **Dès que** $V - T <$ anticipation minimum imposée **faire**
8 $T' \leftarrow \max(T, V)$
9 Requête génération (T', $S_{T'}$)

10 **Dès que** modification de paramètres ou de S affectant la la date $T' \geq T$ **faire**
11 Requête génération (T', $S_{T'}$)

12 **Dès que** *ÉlémentReçu* = *(Idx, Contenu)* **faire**
13 **si** $(Idx = T)\&(\neg$ TempsCourantJoué$)$ **alors**
14 Retard ← Date(mise à jour T) - Date(*ÉlémentReçu*)
15 Jouer(*Contenu*, Retard)
16 TempsCourantJoué ← vrai
17 *Buffer*[*Idx*] ← *Contenu*
18 V ← max(*Idx*+1, V)

La mise en œuvre de cette architecture implique trois processus parallèles écoutant et réagissant respectivement à l'environnement extérieur, aux éléments produits par le modèle, et aux instructions données par l'opérateur. Ils se retrouvent dans les trois blocs formant la « partition dynamique » générique donnée en algorithme 5, résumant l'ordonnancement des appels au modèle et le jeu des séquences dont le calcul a été anticipé ou non :

– L'écoute de l'environnement extérieur (musiciens co-improvisateurs et source de pulsation) orchestre l'apprentissage du matériau musical, l'ordonnancement des requêtes au modèle, et le jeu des éléments anticipés contenus dans le buffer (lignes 1-13 algorithme 5).

– À la réception d'un nouvel élément généré par le modèle (lignes 14-20, algorithme 5), celui-ci est enregistré dans le buffer ou immédiatement joué s'il s'agit de l'élément de l'improvisation correspondant à la date courante.

– Enfin, si une nouvelle recherche est forcée ou quand une contrainte est modifiée par l'utilisateur (lignes 21-22, algorithme 5), une requête intégrant ce nouveau paramètre est transmise au modèle pour la génération des temps suivants, y compris si ceux-ci sont déjà remplis dans le buffer.

Dès que la date courante est mise à jour, une requête sur un suffixe du scénario peut être envoyée au modèle pour générer une partie de l'improvisation à partir de la date correspondante. Celle-ci peut correspondre à la première date V pour laquelle aucun élément d'improvisation n'a encore été calculé si le délai d'anticipation minimum a été atteint (Figure 8), ou à une date $T' \geq T$ si une réaction est imposée par une modification du scénario ou de paramètres extérieurs affectant la position T', quitte à réécrire une portion de l'improvisation déjà anticipée (Figure 9).

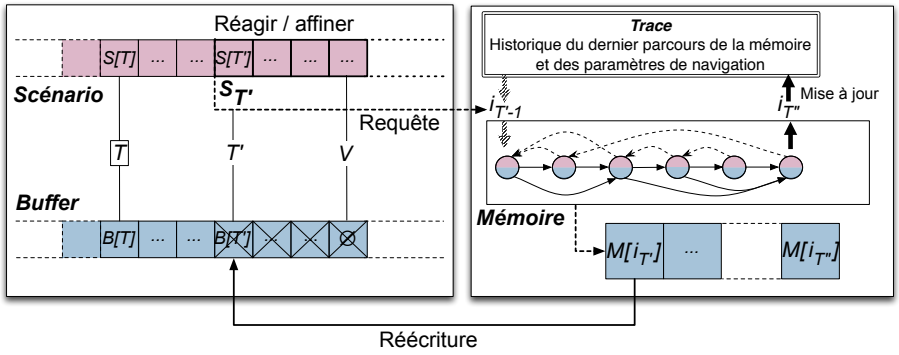

Figure 9. Affiner l'improvisation ou réagir à une modification affectant une date $T' \geq T$ en réécrivant une partie de l'improvisation déjà calculée

On cherche ainsi à pouvoir allier l'anticipation à des contrôles réactifs. En effet, en imposant régulièrement de nouvelles recherches au fur et à mesure que la mémoire musicale se remplit et à chaque nouvelle instruction de l'opérateur (contraintes de génération, alternance entre corpus antérieur et mémoire immédiate,...), les intentions de jeu à court terme contenues dans le buffer peuvent évoluer et se modifier en permanence (Figure 9). En enregistrant dans une *trace* un historique des états de la mémoire parcourus et des contraintes appliquées lors des phases successives, on assure la cohérence entre ces phases de navigation portant sur des suffixes du scénario qui se chevauchent impliquant ainsi que l'anticipation des prochains temps de l'improvisation s'affine au cours du temps en se réécrivant. De cette manière, on pourra par exemple réutiliser un motif venant d'être joué par un musicien ou imposer des contraintes de génération pertinentes sur l'instant. Une « réaction » n'est donc finalement pas envi-

sagée ici comme une réponse épidermique instantanée, mais comme une révision des intentions à court-terme à la lumière des nouveaux évènements de l'environnement, tout en continuant à satisfaire le scénario.

6.3. Implémentation

Le modèle génératif présenté dans les sections 2, 3, et 4 a été implémenté sous la forme d'une bibliothèque Common Lisp dans l'environnement *OpenMusic* (Bresson *et al*., 2011). La mémoire et les méthodes de navigation sont mobilisées dans ce contexte d'interaction à travers un système producteur-consommateur impliquant des processus parallèles se partageant l'accès à la mémoire musicale pour l'apprentissage et la génération, et aux ordres et informations reçus de la partition dynamique.

Le système Antescofo (Cont, 2008) utilisé pour ordonnancer et séquencer dans un cadre de performance improvisée permet de jouer les improvisations produites par le modèle en synchronisation avec l'environnement musical, présentant potentiellement un tempo fluctuant. La partition dynamique (algorithme 5) contenant l'ordonnancement des appels au modèle, le buffer, et le déclenchement du jeu des séquences calculées est implémentée dans son langage associé (Echeveste, Giavitto, Cont, 2013 ; Echeveste, Cont *et al*., 2013) pour obtenir un cadre générique indépendant des situations d'improvisations et des types de scénario. Notamment grâce à une structure `whenever`, la syntaxe du langage permet de définir simplement des réactions aux occurrences d'évènements complexes et non-ordonnés.

Enfin, les stratégies de synchronisation internes pour la gestion des retards (lignes 14-16 algorithme 5) couplées avec l'anticipation sont utilisées pour maintenir la cohérence musicale suite aux modifications au cours de la performance des paramètres de génération. Au-delà des questions d'ordonnancement traitées dans cette section, l'intégration du modèle dans un environnement haut niveau permettant également la définition de « plans d'improvisation » est décrite dans (Nika *et al*., 2014).

7. Conclusions et perspectives

En s'appuyant sur l'existence d'une structure formalisée antérieure à la performance dans de nombreux répertoires improvisés, cet article propose un modèle d'improvisation guidée par un scénario afin d'exploiter certaines connaissances a priori sur la structure de l'improvisation pour introduire une anticipation dans la génération musicale. Ce modèle est implanté dans une nouvelle architecture intégrée dans le système ImproteK afin d'allier cette anticipation avec des contrôles réactifs par l'ordonnancement dynamique de processus statiques, en envisageant une réaction comme une révision des anticipations. Le schéma général du modèle est motivé par la spécificité de l'approche basée sur une métaphore de la réalité de l'improvisation afin de tenter de capturer certains aspects de l'improvisation musicale idiomatique par la modélisation informatique : les choix technologiques sont validés et affinés en interaction avec des musiciens professionnels au cours de séances d'expérimentation effectuées dans

le cadre d'une enquête ethnomusicologique mêlant sessions de travail et situations de concert [5] (la méthode et les résultats sont décrits dans (Chemillier, Nika, à paraître 2015)).

La généricité de l'association « scénario / mémoire » incite à explorer d'autres directions que l'improvisation jazz pour lequel il a été initialement conçu. Dans un contexte de performance, des scénarios décrits avec d'autres vocabulaires ou en termes de descripteurs musicaux adaptés permettront d'aborder de nouvelles dimensions de l'improvisation musicale, en définissant par exemple des profils de densité ou d'énergie en guise de scénario. Dans une perspective d'analyse, on peut citer à titre d'exemple un travail de modélisation des rythmes asymétriques (aksak) joués sur le luth dans un répertoire traditionnel de Turquie (Chemillier, 2014). Ces rythmes sont très difficiles à battre à l'écoute de la partie de luth pour un auditeur non acculturé, ne sachant pas quelles sont les règles qui gouvernent le choix des notes jouées par le luth en fonction des pulsations sur lesquelles elles se trouvent. Avec une mémoire formée d'un ensemble de formules jouées sur cet instrument, et des scénarios reprenant la succession des pulsations sur lesquelles reposent ces rythmes, la validation ou non par des experts des nouvelles formules engendrées par le modèle a permis de mettre en évidence certaines de ces règles.

Sur le plan algorithmique, les améliorations envisagées à court terme visent à optimiser les recherches et à les intégrer dans un processus unifié pour traiter la problématique générale soulevée par la question musicale : le parcours optimal d'un texte (la mémoire) guidé par un mot-contrainte (le scénario) en exploitant les similarités internes du texte pour effectuer des tuilages permettant de dépasser la simple concaténation de motifs communs indépendants les uns des autres. Le but de ce travail était de mettre en place un modèle pour l'improvisation guidée segmentant le processus de génération en étapes pouvant faire sens musicalement avec le souci d'une relative simplicité technologique. Il sera donc suivi de nombreux raffinements et optimisations. Dans un premier temps, l'articulation et les recouvrements entre les informations manipulées par deux phases successives ou deux étapes au sein d'une même phase méritent d'être étudiés : à l'heure actuelle, on ne tient pas compte du fait que deux appels successifs à l'algorithme de reconnaissance de préfixes se font sur des suffixes d'un même mot, ni du fait que certaines informations soient connues sur la structure de la mémoire contenue dans un oracle des facteurs pour procéder à la reconnaissance des préfixes du scénario courant. De plus, dans l'optique d'aboutir à un instrument logiciel riche et d'augmenter les contrôles donnés à l'opérateur sur l'improvisation comme le propose (Maniatakos *et al*., 2010), on cherchera à intégrer dans le modèle d'improvisation guidée les généralisations de l'oracle des facteurs décrites dans (Maniatakos, 2012), permettant d'introduire des contrôles déclaratifs et de dépasser les simples spécifications de zones de la mémoire ou le filtrage des solutions. Enfin, des modifications seront apportées par la suite pour tenir compte des derniers progrès réalisés dans le domaine de la combinatoire des mots sur le problème de la

5. Voir les vidéos sur youtube.com/channel/UCAKZIW0mMWCrX80yS96ZxAw et les ressources sur improtekjazz.org

détection de motifs dans un texte (Crochemore, Ilie *et al.*, 2013) ou de l'indexation des similarités internes du texte (Alstrup *et al.*, 2004 ; Crochemore, Iliopoulos *et al.*, 2013) qui pourront être associées à des analyses complémentaires comme les *k-covers* (Smyth, 2013) fournissant une représentation compacte de l'ensemble des enchaînements présents dans la mémoire.

Remerciements

Les auteurs souhaitent remercier Jean Bresson, Jean-Louis Giavitto, et les différents lecteurs pour les échanges et retours particulièrement constructifs. Ce travail a été en partie réalisé dans le cadre du projet IMPROTECH ANR-09-SSOC-068.

Bibliographie

Allauzen C., Crochemore M., Raffinot M. (1999). Factor oracle: A new structure for pattern matching. In *Sofsem 99: Theory and practice of informatics*, p. 758–758.

Alstrup S., Gavoille C., Kaplan H., Rauhe T. (2004). Nearest common ancestors: A survey and a new algorithm for a distributed environment. *Theory of Computing Systems*, vol. 37, n° 3, p. 441–456.

Assayag G., Bloch G. (2007). Navigating the oracle: A heuristic approach. In *International Computer Music Conference*, p. 405–412. Copenhagen.

Assayag G., Bloch G., Chemillier M. (2006). OMax-ofon. *Sound and Music Computing (SMC)*.

Assayag G., Bloch G., Chemillier M., Cont A., Dubnov S. (2006). OMax brothers: a dynamic topology of agents for improvization learning. In *1st ACM workshop on audio and music computing multimedia*, p. 125–132. ACM, Santa Barbara, California.

Assayag G., Dubnov S. (2004). Using factor oracles for machine improvisation. *Soft Computing*, vol. 8, n° 9, p. 604–610.

Assayag G., Dubnov S., Delerue O. (1999). Guessing the composer's mind: Applying universal prediction to musical style. In *International Computer Music Conference*, p. 496–499. Beijing.

Bailey D. (1999). *L'improvisation: sa nature et sa pratique dans la musique*. Outre mesure. (Version originale : Improvisation, its nature and practice in Music, Ashbourne, Mootland Publishing, 1980)

Bonnasse-Gahot L. (2013). Online arrangement through augmented musical rendering. *Ircam - STMS, Rapport interne projet ANR Sample Orchestrator 2, ANR-10-CORD-0018*.

Boyer R. S., Moore J. S. (1977). A fast string searching algorithm. *Communications of the ACM*, vol. 20, n° 10, p. 762–772.

Bresson J., Agon C., Assayag G. (2011). OpenMusic: visual programming environment for music composition, analysis and research. In *19th ACM international conference on multimedia*, p. 743–746. Scotssdale.

Chemillier M. (2001). Improviser des séquences d'accords de jazz avec des grammaires formelles. In *Journées d'informatique musicale*, p. 121–126. Bourges. Consulté sur http://ehess.modelisationsavoirs.fr/marc/publi/jim2001/jim2001.pdf

Chemillier M. (2004). Toward a formal study of jazz chord sequences generated by Steedman's grammar. *Soft Computing*, vol. 8, n° 9, p. 617–622.

Chemillier M. (2009). L'improvisation musicale et l'ordinateur. *Terrain*, n° 53, "Voir la musique", p. 67–83. Consulté sur http://terrain.revues.org/13776

Chemillier M. (2014). La machine aksak et les fascinantes formules asymétriques du petit luth de Turquie (à propos du livre de Jérôme Cler : Yayla, musique et musiciens de villages en Turquie méridionale). *L'Homme*, n° 211, p. 129-40. Consulté sur http://ehess.modelisationsavoirs.fr/seminaire/seminaire13-14/10-12mars2014-ImproteK-2/CR-jerome.pdf

Chemillier M., Assayag G. (2008). OMax: présentation multimédia des recherches sur l'improvisation et l'ordinateur de l'Ircam et de la Cie Lubat. *Musimédiane*, n° 3, "Musiques non écrites". Consulté sur http://www.musimediane.com/numero3/chemillier/chemillier01.html

Chemillier M., Nika J. (à paraître 2015). « Etrangement musical » : les jugements de goût de Bernard Lubat à propos du logiciel d'improvisation ImproteK. *Cahiers d'ethnomusicologie*, n° 28.

Cont A. (2008). Antescofo: Anticipatory synchronization and control of interactive parameters in computer music. In *International Computer Music Conference.*

Crochemore M., Hancart C., Lecroq T. *et al.* (2001). *Algorithmique du texte* (vol. 3). Vuibert Paris.

Crochemore M., Ilie L., Iliopoulos C. S., Kubica M., Rytter W., Waleń T. (2013). Computing the longest previous factor. *European Journal of Combinatorics*, vol. 34, n° 1, p. 15–26.

Crochemore M., Iliopoulos C. S., Kociumaka T., Kubica M., Langiu A., Pissis S. P. *et al.* (2013). Order-preserving suffix trees and their algorithmic applications. *arXiv preprint arXiv:1303.6872*.

Donze A., Libkind S., Seshia S. A., Wessel D. (2013). *Control improvisation with application to music*. Rapport technique n° UCB/EECS-2013-183. EECS Department, University of California, Berkeley.

Dubnov S., Assayag G. (2005). Improvisation planning and jam session design using concepts of sequence variation and flow experience. In *International conference on Sound and Music Computing*, p. 24–26.

Dubnov S., Assayag G., El-Yaniv R. (1998). Universal classification applied to musical sequences. In *International Computer Music Conference*, p. 332–340. Ann Arbor, Michigan.

Echeveste J., Cont A., Giavitto J.-L., Jacquemard F. (2013). Operational semantics of a domain specific language for real time musician–computer interaction. *Discrete Event Dynamic Systems*, p. 1–41.

Echeveste J., Giavitto J.-L., Cont A. (2013). *A Dynamic Timed-Language for Computer-Human Musical Interaction*. Rapport de recherche n° RR-8422. INRIA. Consulté sur http://hal.inria.fr/hal-00917469

François A. R., Schankler I., Chew E. (2013). Mimi4x: an interactive audio–visual installation for high–level structural improvisation. *International Journal of Arts and Technology*, vol. 6, n° 2, p. 138–151.

Knuth D., Morris J. H., Pratt V. R. (1977). Fast pattern matching in strings. *SIAM journal on computing*, vol. 6, n° 2, p. 323–350.

Lefebvre A., Lecroq T. (2000). Computing repeated factors with a factor oracle. In *Proceedings of the 11th australasian workshop on combinatorial algorithms*, p. 145–158.

Lefebvre A., Lecroq T., Alexandre J. (2002). Drastic improvements over repeats found with a factor oracle.

Lévy B., Bloch G., Assayag G. (2012). OMaxist dialectics. In *International conference on new interfaces for musical expression*, p. 137–140.

Maniatakos F. (2012). *Graphs and automata for the control of interaction in computer music improvisation.* Thèse de doctorat, Université Pierre et Marie Curie.

Maniatakos F., Assayag G., Bevilacqua F., Agón C. (2010). On the architecture and formalisms for computer-assisted improvisation. *Proc. of Sound and Music Computing (SMC)*, p. 28.

Moreira J., Roy P., Pachet F. (2013). Virtualband: Interacting with stylistically consistent agents. In *Proc. of International Society for Music Information Retrieval Conference*, p. 341–346. Curitiba.

Morris J. H., Pratt V. R. (1970). *A linear pattern-matching algorithm*.

Nika J., Chemillier M. (2012). Improtek: intégrer des contrôles harmoniques pour l'improvisation musicale dans la filiation d'OMax. In *Journées d'Informatique Musicale (JIM 2012), Mons, Belgique*, p. 147–155. Consulté sur http://hal.archives-ouvertes.fr/hal-01059324

Nika J., Echeveste J., Chemillier M., Giavitto J.-L. (2014). Planning Human-Computer Improvisation. In *International Computer Music Conference*, p. 1290–1297. Athens, Greece. Consulté sur http://hal.inria.fr/hal-01053834

Pachet F. (2003). The continuator: Musical interaction with style. *Journal of New Music Research*, vol. 32, n° 3, p. 333–341.

Pachet F., Roy P., Moreira J., d'Inverno M. (2013). Reflexive loopers for solo musical improvisation. In *Sigchi conference on human factors in computing systems*, p. 2205–2208. ACM, Paris.

Pressing J. (1984). Cognitive processes in improvisation. *Advances in Psychology*, vol. 19, p. 345–363.

Pressing J. (1988). Improvisation: methods and models. *John A. Sloboda (Hg.): Generative processes in music, Oxford*, p. 129–178.

Shaffer L. H. (1980). 26 analysing piano performance: A study of concert pianists. *Advances in Psychology*, vol. 1, p. 443–455.

Sioros G., Guedes C. (2011a). Complexity driven recombination of midi loops. In *International society for music information retrieval (ISMIR 2011)*, p. 381–386.

Sioros G., Guedes C. (2011b). A formal approach for high-level automatic rhythm generation.

Sloboda J. A. (1982). Music performance. *The psychology of music*, p. 479–496.

Smyth W. (2013). Computing regularities in strings: A survey. *European Journal of Combinatorics*, vol. 34, n° 1, p. 3–14.

Surges G., Dubnov S. (2013). Feature selection and composition using PyOracle. In *Ninth artificial intelligence and interactive digital entertainment conference.* Boston, Massachusetts.

Thom B. (2000). Bob: an interactive improvisational music companion. In *Proceedings of the fourth international conference on autonomous agents*, p. 309–316.