Guitare électrique, nouvelle génération

François Pachet
SONY CSL-Paris, pachet@csl.sony.fr

Abstract : Nous décrivons un système d'aide à l'improvisation musicale qui permet d'étendre les capacités techniques d'un musicien en produisant des phrases de continuation. Ces continuations sont conformes au style du musicien, style lui-même appris par le système de manière agnostique, en continu. Le système s'oppose aux systèmes de génération musicale automatiques sur plusieurs aspects techniques (suivi de l'harmonie). Il ouvre la voie a plusieurs modes de jeu musicaux collectifs nouveaux, par exemple en permettant aux musiciens d'échanger leurs base stylistiques en temps réel.

1. Introduction

L'improvisation musicale, le Jazz en particulier, est une activité aussi fascinante que frustrante. L'acte d'improvisation met en œuvre, voire à l'épreuve, la relation intime entre pensée musicale et mouvement : l'improvisateur doit tout à la fois écouter, penser, trouver des idées musicales et *in fine* effectuer les gestes musicaux appropriés. La vitesse de ce processus, et le manque de temps pour réaliser cette chaîne de la pensée au geste est dans ce contexte un ingrédient central, c'est probablement ce qui rend l'improvisation excitante. C'est aussi ce qui peut la rendre frustrante, puisque les musiciens, débutants ou professionnels, sont, par définition, limités dans leur habileté technique ainsi que par la morphologie de l'instrument : ça ne va pas toujours aussi vite que l'on voudrait. Cette frustration peut être négative ou positive, par exemple source de raccourcis ou de nouveauté. C'est en tout cas au musicien de décider comment utiliser sa technique ou son manque de technique de manière appropriée.
Nous proposons de concevoir des instruments de musique qui permettent au musicien - d'une certaine manière - d'étendre ses possibilités techniques tout en respectant les règles du jeu de l'improvisation : temps réel, rapidité et surtout primauté du contrôle au musicien, qui doit, qui veut, en dernier ressort, décider et « réaliser » sa pensée.

De nombreux chercheurs en informatique musicale ont abordé le « problème » de l'improvisation. D'un point de vue technique il s'agit la plupart du temps de fabriquer des improvisateurs automatiques en utilisant toute la panoplie des techniques informatiques disponibles (Biles, 1998 ; Ramalho, 1994 pour n'en citer que deux). Certains travaux ont par ailleurs l'ambition de comprendre le processus « cognitif » de l'improvisation, via la conception de modèles et leur expérimentation (Johnson-Laird, 1991). Malgré quelques succès dans des cas très particuliers (le bassiste de Jazz de Ramalho par exemple, cf. Pachet, 2000) les performances de ces systèmes restent très en deçà des performances humaines, et le sentiment d'écouter un automate, au sens primitif du métier à tisser de Vaucanson, est indélébile. En outre, et comme les cartes perforées fournies à ce même métier à tisser, ces systèmes nécessitent, pour fonctionner correctement, l'apport de connaissances humaines relativement complexes et le plus souvent entrées à la main (validation pour les systèmes d'apprentissage supervisés, séquence d'accords sous jacente, tempo, structure du morceau, etc.), ce qui compromet encore leur usage dans des situations de jeu réelles.

Le système que nous présentons ici se distingue de ces approches par plusieurs aspects fondamentaux. D'une part notre système n'est pas autonome, mais au contraire s'intègre de

manière intime au jeu d'un vrai musicien, par opposition aux automates décrits plus haut ou aux systèmes de question/réponse (Biles, 1998 ; Wessel, 1991). Par ailleurs, nous utilisons un module d'apprentissage non supervisé qui ne nécessite pas de connaissances humaines explicites, ni de connaissances sur le morceau (harmonie, tempo, structure, etc.), et s'adapte automatiquement aux changements imprévus - de style, tempo, etc. Le système résultant doit être considéré et le cas échéant évalué, non pas comme un système autonome, mais comme un couple musicien / ordinateur. Une de ses caractéristiques principales est en effet, comme on le verra plus bas de substituer les connaissances symboliques explicites des systèmes traditionnels par du contrôle.

2. Description du système

Notre système est constitué de deux modules, fonctionnant tous deux en temps réel: un module d'analyse et un générateur de phrases. Ces modules prennent en entrée des flux d'informations Midi. En principe, toute information Midi peut être traitée par ces modules, mais nous n'avons utilisé pour l'instant que les hauteurs (pitch) et vélocité. Les expérimentations réalisées ont utilisé une guitare Midi - d'où le titre - dans un contexte Jazz, mais sont aisément transposables à d'autres instruments Midi (clavier, saxophone, violon) et d'autres styles.

Continuation et non pas réponse

Plusieurs modes de jeu peuvent être envisagés avec le système : le nombre de musiciens peut être arbitraire, et pour chaque musicien, il existe plusieurs modes de contrôle individuel (voir 4.2). Dans le mode par défaut, à un seul musicien, le jeu en temps réel du musicien est donné en entrée du système, et la sortie du système est envoyée à un synthétiseur dont la sortie est mixée avec l'entrée. Dans nos expérimentations, nous avons utilisé le même son pour le jeu « humain » et la sortie de système, rendant ainsi la contribution du système pratiquement indistinguable de celle du musicien (voir plus loin à ce sujet).

Dans ce mode standard, le système joue le rôle d'un continuateur de séquence, à la manière des mécanismes de *sequence completion* sur certains systèmes d'exploitation. Le jeu du musicien humain est continuellement segmenté en phrases musicales, en utilisant un seuil temporel (de l'ordre de 200 millisecondes). Dès qu'une phrase est détectée (i.e. un silence plus long que le seuil), celle-ci est envoyée en même temps à l'analyseur et au générateur. L'analyseur construit ainsi incrémentalement un modèle des phrases récurrentes du musicien. Le module de génération tente, lui, de compléter la séquence d'entrée en utilisant le modèle construit par l'analyseur. Ce dernier point est important : la séquence générée par le système n'est pas une « réponse », comme dans le système de Biles (1998) par exemple, mais bien une complétion. Musicalement, la différence est en effet de taille puisque dans le cas d'une réponse, on peut considérer que les deux phrases (la question et la réponse) sont bien distinctes (et d'ailleurs souvent jouées, en Jazz, par des instruments différents). Dans le cas de la complétion, il s'agit de la *même phrase*, qui est simplement initiée par le musicien et continuée par le système.

Détection de patterns et l'algorithme LZ

Le module d'analyse (ou d'apprentissage) tente systématiquement de repérer dans les phrases qui lui sont données en entrées des patterns récurrents. De nombreuses techniques existent pour repérer de telles régularités (voir par exemple P. Y Rolland (1999) pour les techniques basées sur la programmation dynamique). Nous avons utilisé pour nos expérimentations une

méthode particulièrement simple, issue du domaine de la compression de données : l'algorithme Lempel-Ziv (Ziv & Lempel, 1978), dont il a été montré qu'il se prêtait bien à la modélisation de patterns mélodiques (Assayag et al., 1999), tout en ayant de bonnes performances. Cette technique consiste à construire un arbre de suffixes (nommé arbre LZ) par une simple lecture de gauche à droite de la séquence d'entrée. Chaque fois qu'un nouveau suffixe est détecté, celui-ci est ajouté à l'arbre, comme fils du sous-pattern dont il est l'extension directe. Le procédé est dans son principe très simple, et est décrit en détail par exemple dans (Assayag et al. 99). Nous l'avons modifié pour l'adapter à notre contexte en deux points importants.

D'une part, nous avons besoin d'une représentation efficace de la technique d'analyse, pour pouvoir être utilisée en temps réel, à la fois pour la mise à jour de l'arbre LZ (analyse) et pour la génération. Assayag et al. ont proposé une représentation duale de l'arbre LZ, qui en théorie permet d'accélérer la procédure de génération, au prix d'un encombrement mémoire très important. Après expérimentations, nous ne sommes pas persuadés que cette technique soit efficace pour les corpus considérés (de l'ordre de 40.000 nœuds). Nous avons opté pour une technique consistant à exploiter l'arbre LZ « standard », augmenté par une table de Hash, permettant d'obtenir directement la liste des nœuds LZ pour chaque donnée musicale (dans notre cas un pitch Midi). Cette table est mise à jour à chaque création de nœud LZ. En phase de génération, pour une séquence commençant par un item i, la table donne ainsi directement la liste des nœuds possibles.

La deuxième modification concerne la nature intrinsèquement incomplète de l'algorithme standard de génération à partir d'un arbre LZ. Pour chaque génération, le système calcule d'abord toutes les continuations possibles, puis en tire une au hasard, de manière pondérée, en fonction des probabilités d'occurrence de chaque continuation. Par construction cependant, l'arbre LZ n'est pas complet (par opposition aux modèles basés sur les chaînes de Markov). Ainsi, un pattern musical donné peut se trouver à plusieurs endroits de l'arbre. Par exemple, le pattern ABC se trouve à deux endroits de l'arbre de la figure 1.

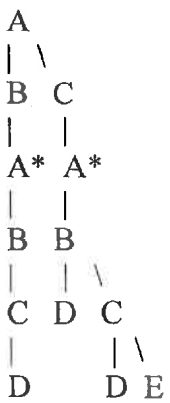

Figure 1 . Un arbre LZ contenant deux fois la sous séquence ABC (chaque début de pattern est indiqué par une étoile).

Pour détecter et traiter ces multiples occurrences, nous avons modifié légèrement l'algorithme de génération standard LZ, en calculant, pour chaque sous séquence d'entrée à « continuer », toutes les continuations possibles, à n'importe quelle position dans l'arbre (au lieu de ne considérer, à chaque fois, que la racine).

Dans notre exemple, pour une séquence d'entrée ABC, la table de Hash nous donne directement les trois nœuds contenant la donnée « A ». Trois traversées de l'arbre sont donc effectuées à partir de chacun de ces nœuds, pour vérifier qu'ils correspondent bien à la séquence ABC. Deux seulement de ces nœuds satisfont cette propriété, et les continuations possibles sont donc {D, D, E} avec comme poids respectifs {1, 1, 1}. Un tirage aléatoire est alors effectué pour choisir la continuation.
Nous obtenons ainsi un système capable de traiter environ 30.000 nœuds en temps réel, c'est à dire avec un temps de réponse pour la génération de moins de 200 millisecondes, avec une implémentation Java utilisant MidiShare (Orlarey et al., 1998) et tournant sur un PC Pentium III 600 Mhz. Le capteur Midi utilisé est un système Roland GK-II / GR30.

3. Remplacer la connaissance par du contrôle

Notre système est limité intentionnellement dans ses connaissances musicales. Les limitations sont de trois types : rythmique, harmonique et polyphonique. Nous pensons que ces limites sont en fait la source même de l'efficacité du système et rendent possible son usage en situation réelle. Nous discutons ici ces trois points.
Premièrement, le système n'a pas de connaissance rythmique, ni pour l'analyse ni pour la génération. Les phrases générées sont toutes des séquences linéaires de double croches (du moins on peut les interpréter ainsi). Quelques expérimentations avec des structures non linéaires ne nous ont pas convaincu que l'on pouvait faire beaucoup mieux dans notre contexte. En effet, la génération par le système de séquences non linéaire est difficile à contrôler, et engendre immanquablement des phrases maladroites, dans le style des automates musicaux. Ce défaut provient probablement du fait que nous ne savons pas bien modéliser le rythme musical autrement que par sa réduction à une partition. En revanche, le système mémorise et génère des informations de vélocité, et les phrases engendrées restituent une impression de naturel indéniable.

Deuxièmement, le système est par essence monophonique, et ne génère donc pas d'accords. Même si techniquement l'arbre LZ peut aisément accommoder des structures harmoniques, quelques expérimentations nous ont convaincu que la polyphonie ne pouvait se traiter comme la mélodie du point de vue du contrôle. La gestion analyse / génération d'accords et de séquences d'accords est traitée par ailleurs (Chemillier, 2001 ou Pachet, 1999) par d'autres techniques, avec un point de vue générationnel tout à fait différent : il s'agit plus de produire des variations sur une même structure algébrique, que de continuer, à proprement parler, des séquences.

Enfin, le système n'a pas de connaissance harmonique. L'harmonie est un point central dans l'improvisation de Jazz car la séquence d'accords sous-jacente à une mélodie permet en quelque sorte de décider quelles sont les notes utilisable à tout instant (avec bien sûr la flexibilité que l'on sait). Notons ici que la détection de l'harmonie sous jacente est relativement aisée à effectuer pour un musicien : celui-ci sait en général assez bien ou il se trouve dans la séquence d'accords ou la mélodie. En revanche, cette information est assez difficile à donner à l'ordinateur. D'une part il faut en effet donner à l'avance la séquence d'accords (ce qui empêche les changements et variations imprévus). D'autre part, il faut indiquer à l'ordinateur où en sont les musiciens, et donc résoudre les nombreux et délicats problèmes de synchronisation en temps réel, suivi de partition et autres.
Même si ces problèmes sont solubles en théorie, nous avons choisi de ne pas représenter d'information harmonique de manière explicite pour trois raisons principales. D'une part il est relativement facile de « corriger » ou plutôt relancer le système lorsque celui-ci quitte trop

brutalement l'harmonie sous- jacente : il suffit de jouer quelques notes (tierce ou quinte) pour relancer le système dans la « bonne » direction. Deuxièmement, le système apprend continuellement toutes les phrases jouées par le musicien. Le plus souvent, les improvisations de Jazz consistent à exploiter une structure harmonique de manière répétitive. Ainsi, les enchaînements harmoniques spécifiques de la grille vont être progressivement « appris » par le système. Un exemple typique de ce phénomène est le morceau « So What » de Mils Davis, qui ne comprend que deux accords (D min et D# min). A force d'analyser des phrases jouées sur cet enchaînement, le système finit par connaître des patterns de transition D min / D# min et donc les rejoue lui-même.
Enfin, nous avons développé un mode de contrôle spécifique qui permet d'influencer la génération par des informations harmoniques extérieures sans analyse harmonique complexe (*lightweight*) et sans complexifier le système. L'idée est d'introduire un biais pour le calcul des continuations, sous la forme de contraintes. Des informations provenant de l'extérieur (par exemple un deuxième musicien jouant du piano Midi) peuvent être envoyées en temps réel au module de génération : typiquement, les 8 dernières notes (pitch) jouées par ce musicien. Ces informations peuvent alors être utilisées pour influencer le choix des continuations, à partir de l'arbre LZ. Au moment du tirage aléatoire, la procédure standard (décrite plus haut) consiste à associer des poids en fonction des probabilités d'occurrence de chacune des continuations possibles. Nous pouvons changer ce schéma, et remplacer ces probabilités d'occurrence par une fonction quelconque des continuations, par exemple en privilégiant les notes qui sont égales ou proches harmoniquement des notes reçues par l'entrée extérieure. L'effet immédiat de ce changement de fonction aléatoire est d'influencer la génération de l'arbre vers la direction harmonique correspondant aux notes extérieures. Ainsi, on peut se rapprocher plus ou moins d'un musicien, tout en gardant le même schéma de contrôle sur le système. Ceci permet alors, in fine, de donner une sensibilité harmonique au système, sans avoir entré d'information symbolique (la séquence d'accords), et avec toute la flexibilité souhaitée (changements de tempo et d'harmonie par exemple).

4. De nouveaux modes de jeu

Le système que nous proposons ici introduit plusieurs modes de jeu nouveau avec un ordinateur.

1. Contrôleurs de base

D'une part, pour faciliter l'usage du système, nous avons développé une série de contrôleurs déclenchables en temps réel pendant le jeu de l'improvisation. Ces contrôleurs permettent de modifier certains paramètres du système, pour les deux modules d'analyse et de génération.
Ces paramètres sont en particulier :

- déclenchement systématique pour chaque phrase détectée (on/off)
- apprentissage systématique des phrases détectées (on/off)
- superposition (on/off). Ce paramètre permet de décider si l'on autorise ou non les superpositions de phrases musicales. Par défaut, le système s'interrompt dès que le musicien commence une nouvelle phrase. Avec un peu d'entraînement, on peut ainsi arriver à jouer de longues phrases « sans couture » mélangeant les contributions du musicien et de l'ordinateur.

Tous ces contrôles sont déclenchables à partir d'un pédalier Midi. Des expérimentations avec des gants Midi sont en cours.

Par ailleurs, un certain nombre de paramètres hors temps réel permettent d'adapter le système à un style donné et sont accessibles via une interface graphique simple (Cf. Figure 2) : le nombre de notes généré par le système (multiple du nombre de notes de la phrase d'entrée), le tempo (idem), le nombre de fois que chaque séquence d'entrée est apprise (ici, 3), et l'apprentissage éventuel de transpositions de la séquence.

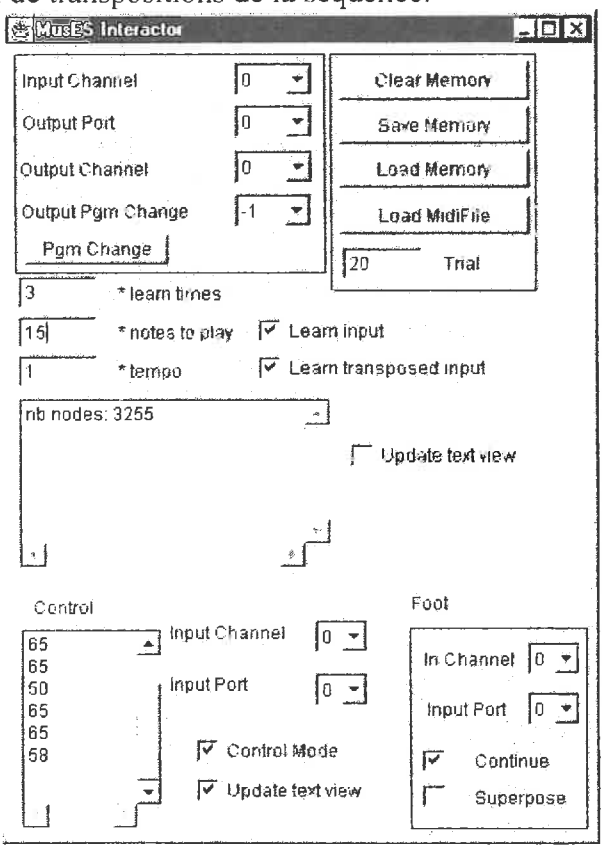

Figure 2. L'interface graphique du système et les paramètres non temps réel.

2. *Modes de jeu*

Une conséquence directe de la conception du système, séparant module d'analyse et de génération, et qu'il permet de considérer de nouveaux modes de jeu pour l'improvisation. Nous listons ici quelques-uns de ces modes :

- *Autarcie simple*. C'est le mode par défaut, décrit ici : un seul musicien avec un seul système. L'apprentissage est effectué off-line, à partir du jeu du musicien (sauvegardable par fichier) ou d'un ensemble de MidiFiles. Nous avons en particulier expérimenté avec des MidiFiles de chorus de Pat Martino (Hauser, 1994).
- *Autarcie multiple*. Dans ce mode, chaque musicien a son propre système. C'est la simple démultiplication du précédent, mais chaque musicien reste autarcique.
- *Maître / Esclave*. Dans ce mode, un musicien utilise le système par défaut, un autre produit une influence extérieure, par exemple harmonique, comme décrit dans la section 3. Ce schéma de base est lui-même cumulable avec d'autres, voire peut être mis en boucle.
- *Cumulatif*. Ici plusieurs musiciens utilisent le système en partageant simplement la base de patterns. Ceci permet alors de constituer une base commune musicale, et de jouer collectivement avec.

- *Partage*. Chaque musicien utilise la base de patterns crée par un autre. Ceci permet de jouer avec des patterns inhabituels : les patterns typiques de saxophone, piano ou guitare sont, en effet, assez différents, pour des raisons de morphologie d'instrument.

5. Résultats

Il est bien sûr difficile de décrire la musique et en particulier une improvisation. Le système présenté ici peut produire de longues phrases souvent impressionnantes, linéaires mais expressives, dans des styles guitaristiques divers (Pat Martino, Pat Metheny, Alan Holdsworth, John McLaughlin). Toute évaluation du système doit cependant être conduite non pas sur le système lui-même, mais sur le couple musicien/système. A cet égard, les écoutes réalisées devant différents auditeurs montrent que la plupart de ceux ci ne sont pas capables de distinguer les contributions humaines et de la machine. Par ailleurs, la qualité de l'improvisation (ici encore, difficile à quantifier) dépasse de loin, à notre avis, ce que produisent les systèmes autonomes. Laissons les auditeurs juger d'eux-mêmes : quelques enregistrements sont disponibles à www.csl.sony.fr/~pachet/music. Enfin, nous pensons que nos expérimentations permettent de relancer la question de l'improvisation dans un contexte à la fois plus réaliste (on passe du système autonome à l'instrument intelligent) et, finalement, plus humain.

References

Assayag, G. Dubnov, S. Delerue, O. (1999) Guessing the Composer's Mind: Applying Universal Prediction to Musical Style, Proc. ICMC 99, Beijing, China, I.C.M.A., San-Francisco.

Biles, John A. (1998) Interactive GenJam: Integrating Real-Time Performance with a Genetic Algorithm, Proc. ICMC 98, Ann Arbor, Michigan.

Chemillier, Marc (2001) Improvising Jazz Chord Sequences by Means of Formal Grammars, submitted to ICMC 2001, Cuba.

Heuser, Jorg (1994) Pat Martino – His contributions and influence to the history of modern Jazz guitar. Ph.D thesis, University of Mainz (Ge).

Johnson-Laird, P. N. (1991). "Jazz Improvisation: A Theory at the Computational level." Representing Musical Structures, P. Howell, R. West, and I. Cross, eds., Academic Press, 291-325.

Pachet, F. Surprising harmonies. *JIM 1999*, Issy-Les-Moulineaux.

Pachet, F. Qu'est ce qu'une mélodie intéressante. *La Recherche*, numéro spécial sur les origines de l'art, Novembre 2000.

Martino, Pat. (1993) Creative Force, Vol. 1 and 2, CPP/Beldwin, Miami.

Ramalho G., Ganascia J.-G. (1994) Simulating Creativity in Jazz Performance. Proceedings of the National Conference in Artificial Intelligence, pp. 108-113, AAAI-94, Seattle, AAAI Press.

Rolland, P.Y. 1999. Discovering Patterns in Musical Sequences. *Journal of New Music Research* 28:4, December 1999. Pages 334-350.

Wessel, David, Improvisation with Highly Interactive Real-Time Performance Systems, in Proc. Int. Computer Music Conf. (ICMC'91), pp. 344-347.

Ziv, J. Lempel, A. "Compression of individual sequences via variable rate coding", *IEEE Trans. Information Theory*, 24, (5), 1978.