Modélisation de grammaires musicales : approches formelles et statistiques

Guilhem Marion

Introduction

A partir des années 1950, un nouveau champ disciplinaire apparait : la linguistique informatique. C'est un champ interdisciplinaire composé entre autres de la linguistique et de l'informatique. Un des aspects de cette discipline est la recherche d'une formalisation du fonctionnement du langage humain permettant sa reproduction par un système artificiel. On distingue deux approches : statistique et logico-grammaticale. Mais dans quelle mesure est-il possible de modéliser des compétences langagières ?

Pour cela intéressons nous à une expérience de pensée du philosophe américain de l'esprit, John Searl : *la chambre chinoise* [1].

Dans cette experience de pensée, Searle imagine une personne n'ayant aucune connaissance du chinois enfermée dans une chambre. Dans cette chambre, il y un catalogue contenant un ensemble de règles permettant de construire des phrases correctes en chinois. Ces règles sont parfaitement claires et reposent uniquement sur la syntaxe des phrases. Un véritable sinophone situé à l'extérieur de la chambre pose des questions à la personne enfermée qui est ainsi capable d'y répondre en utilisant le catalogue. Du point du vu du sinophone, la personne dans la chambre se comporte comme un individu qui parlerait vraiment le chinois. Cependant, elle ne fait qu'appliquer les règles du catalogue et n'a aucune comprehension des phrases qu'elle entend et produit.

1. J.R Searle, "Minds, Brains and programs", The Behavioral and Brain Sciences, vol. 3, Cambridge University Press, 1980, tr. fr. "Esprits, cerveaux et programmes", in D. Hofstadter, D. Dennett, Vues de l'Esprit, Paris, Interéditions, 1987, pp. 354-373

Ce test apparait dans les années 1980 pour tenter de savoir si un programme informatique peut accéder à le compréhension d'un langage. Il vise à montrer que seulement une intelligence faible peut être atteinte et ne fait que simuler une conscience. Elle montre de plus que le test de Turring est insuffisant pour determiner si un système possède des états mentaux de conscience et d'intentionnalité.

Mais elle nous intéresse dans notre cas car elle nous permet de faire la distinction entre syntaxe et sémantique et nous dit qu'il est rationnellement possible de modéliser la syntaxe d'un langage mais qu'un programme ne peut réellement accéder à sa sémantique, il peut seulement la simuler (cette notion est bien entendu en évolution notamment avec l'émergence des réseaux de neurones). C'est pourquoi dans ce travail nous allons nous intéresser uniquement à la syntaxe de langages musicaux, c'est à dire à la forme qu'ils doivent prendre pour être considérés comme corrects, ou relevant d'un style bien déterminé, le discours musical quant à lui est laissé de coté, relevant en effet d'un travail bien plus complexe.

Une partie des travaux de Noam Chomsky durant la seconde moitié du XX^{me} reposait sur ces questionnements quant au langage naturel, Harris et Chomsky tentèrent d'appliquer la notion de langages formels à la formalisation de la syntaxe des langues : les grammaires génératives décrivent des systèmes de règles de production capables d'engendrer toutes les phrases correctes ; des automates peuvent alors reconnaitre si une phrase appartient au langage ou non : on parle de langage rationnel.

Aujourd'hui comment peut-on appréhender la modélisation du langage musical ? Dans ce travail nous montrerons qu'il existe deux approches possibles : formelle et statistique. Dans un premier temps nous nous intéresserons au contrepoint car il utilise une systématique musicale si bien définie que l'appartenance d'un morceau à ce langage est décidable. Nous expliquerons comment construire un programme le modélisant permettant ainsi de générer des contrepoints valides. Puis, dans un second temps, nous verrons que les méthodes statistiques utilisant les chaines de Markov et les $n-grammes$ peuvent être très efficaces pour modéliser des grammaires musicales dont la systématique n'est pas claire pour l'observateur, on prendra l'exemple de styles de jeux chez les musiciens de Jazz.

Application au langage musical

La langage musical est-il assimilable à un langage au sens de la théorie des langages formels ? Difficile à dire : l'art, pour Tolstoï, est

> « l'activité humaine par laquelle une personne peut volontairement et au moyen de signes extérieurs communiquer à d'autres les sensations et les sentiments qu'il éprouve [2] »

On parle alors de langage artistique, il est donc évident que le formalisme de tels langages n'est jamais figé et toujours en perpétuelle évolution. De plus on peut penser que la syn-

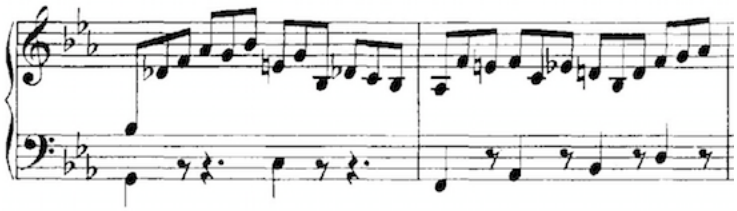
2. Léon Tolstoï, *Qu'est-ce que l'art ?*, Traduction de Teodor de Wyzewa, puf, 1918

taxe nourrit le discours et n'existe pas indépendamment. Cependant, il existe des formes musicales à la syntaxe si rigoureuse qu'elle parait suffisante à la composition d'une œuvre : le contrepoint en est l'exemple le plus marquant. On retrouve en effet en conservatoire des classes de composition de contrepoint où la sensibilité n'est absolument pas une demande et la seule exigence est le respect rigoureux des règles. Ces règles se trouvent compilées dans un ouvrage de Noel Gallon et Marcel Bitsch : *Traité de contrepoint*. J'ai donc voulu réaliser moi même une petite expérience afin de voir ce que la modélisation formelle de ces règles permettait d'obtenir.

La contrepoint : une courte définition

Le contrepoint est une technique d'écriture musicale issue de l'époque baroque qui s'intéresse la superposition de lignes musicales.

Quand on compose ou analyse une partition, on distingue deux grands aspects : l'harmonie et la mélodie. Une mélodie est un ensemble de notes qui se suivent et qui peut se chanter aisément, l'harmonie quant à elle est définie par un ensemble de notes jouées en même temps, on ne peut le chanter seul. On parle alors de vision horizontale et verticale du fait de la forme des partitions musicales.



Dans une approche harmonique on s'intéresse aux enchainements d'accords [3], le contrepoint quant à lui s'intéresse à la qualité des lignes mélodiques ; les accords créés par la rencontre de plusieurs notes sont considérés comme secondaires. Par exemple, là où l'approche harmonique voit un simple mouvement par acheminement d'accords, l'approche contrapuntique voit une multiplicité de lignes mélodiques se combinant. Les lignes mélodiques ainsi formées sont régies par des règles de mouvements strictes.

Cependant l'approche harmonique n'est pas exclue pour autant, lors de moments rythmiques précis (temps forts), les principes de contrepoint interdisent les dissonances. Par contre, en dehors des ces moments rythmiques, les lignes mélodiques peuvent prendre des libertés et ainsi se rencontrer de temps à autre en formant des harmonies tout à fait inattendues, c'est aussi l'intérêt du contrepoint.

On considère généralement la musique de Bach comme la référence de l'équilibre de ces deux aspects de l'écriture musicale.

3. Notes jouées simultanément

La forme d'exercice que l'on retrouve le plus souvent en conservatoire est l'écriture d'un contrepoint à partir d'un *cantus firmus* qui est la voix de départ de la composition ; on utilise la plupart du temps une mélodie efficace écrite en ronde [4]. Le contrepoint doit alors s'associer rythmiquement et harmoniquement au cantus firmus mais aussi respecter les règles d'écriture de mélodie contrapuntique.

La contrepoint comme langage rationnel

Je vais dans cette partie tenter de montrer que le langage contrapuntique se prête bien à l'utilisation de la modélisation par contraintes de part sa nature de langage rationnel (qui prouve sa calculabilité d'après la théorie des langages formels). Pour cela, montrons que l'on peut construire un automate fini déterministe le reconnaissant.

Démonstration.

i) On définit l'alphabet Σ comme le produit cartésien des différents mouvements permis (On les notera par des chiffres romains : V est la quinte ascendante, -III est la tierce descendante, etc ...) avec l'ensemble des différentes durées de notes possibles. Tout contrepoint étant transposable, on peut se restreindre à la gamme de Do majeur et n'autoriser que les mouvements entiers, le nombre de mouvements permis est défini par l'ambitus maximal [5].

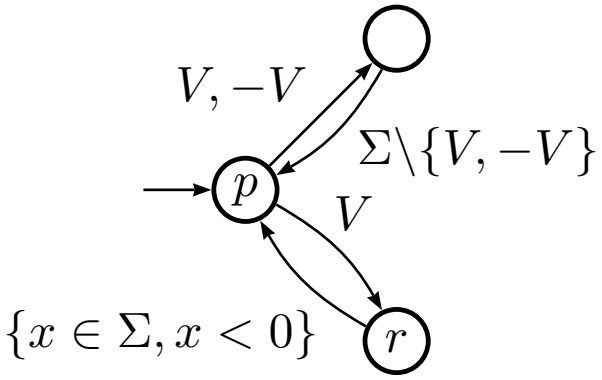
ii) Toutes les règles sont présentées sous formes d'interdictions ou d'obligations de certaines configurations -à l'exception de deux règles qui sont des maximisations que l'on traitera plus tard-, par exemple : « On ne peut écrire plusieurs quintes consécutives » ou « une quinte ascendante s'accompagne d'un mouvement descendant [6] ». Il est clair que ces configurations peuvent se representer par un nombre fini d'états d'un automate étant donné que le nombre de règles est fini. Chaque état possède un état de sortie qui correspond soit au mouvement obligatoire, soit à $\Sigma \backslash \{\text{mouvements interdits }\}$

Si l'on ne veut pas forcement un automate minimal, il est aisé de construire un automate tel que chaque état est une configuration possible (mouvement permis), on y ajoute quelques états pour définir les règles qui prennent en compte les n-derniers mouvements.

iii) Tous les états sont acceptants, car un contrepoint peut faire la taille que l'on désire, il peut donc terminer dans n'importe quel état.

L'automate entier ne peut être représenté ici, cependant voici un exemple avec les règles précédentes, en omettant la durée des notes.

4. Note de quatre temps
5. L'ambitus en musique est l'écart entre le note la plus haute et la plus basse.
6. Cette règle n'est pas exprimée de la sorte, je l'utilise pour l'exemple.



□

L'appartenance d'un morceau au langage contrapuntique est donc décidable. On a ainsi montré qu'il est théoriquement possible de modéliser formellement les règles du contrepoint, qu'en est-il en pratique ?

Mise en place d'un générateur par contraintes

Dans cette partie je vais expliquer ma démarche quant au développement d'un générateur de contrepoints par programmation par contraintes. Ce programme a été réalisé en Java en utilisant la bibliothèque *javax.sound.midi* pour la gestion des fichiers midi.

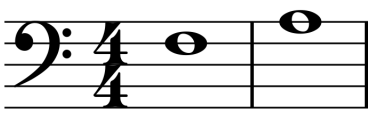
Pour implémenter ces règles au sein d'un programme informatique, il faut tout d'abord choisir les structures de données les plus pertinentes. J'ai choisi d'utiliser trois classes : la Partition, la Voix et la Note afin de pouvoir gérer différents niveaux d'abstraction et de simplifier le travail. La méthode de modélisation est simple : une méthode de Voix crée une liste de toutes les notes possibles dans une tonalité donnée et ce pour tous les rythmes envisageables. Cette liste est envoyée à une autre méthode qui parcourt et enlève les notes qui ne respectent pas au moins une règle. On choisit à l'issue du filtrage une note au hasard parmi les notes valides. Chaque règle est donc modélisée par un bloc d'instruction "Si", la condition de ce bloc d'instruction est la négation de la modélisation logique de la règle.

Maximisation

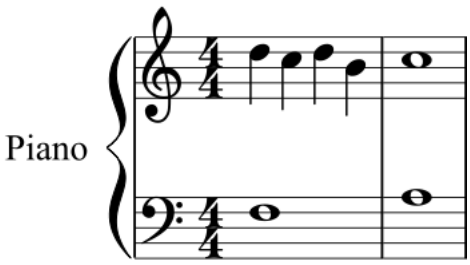
Maintenant que nous avons modélisé le plupart des règles intéressons nous à la maximisation. Les deux règles de maximisation sont : « Il faut le plus possible utiliser le mouvement conjoint [7] » et « Il faut faire le plus de mouvements contraires [8] possible ». Une solution simple aurait été d'utiliser le paradigme glouton et préférer à chaque étape les mouvements conjoints et contraires. Or nous allons montrer qu'ici ce paradigme n'est pas optimal, et ce en exhibant un simple exemple.

7. Intervalle allant d'une note à une note voisine contiguë.
8. Lorsque deux parties se déplacent en sens inverse

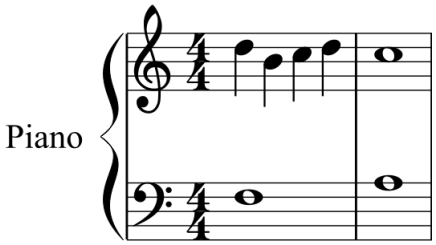
Démonstration. On souhaite écrire un contrepoint pour le cantus firmus suivant :



La paradigme glouton ne peut donner meilleur résultat que :



Or un optimum est :



Cela est du à la présence de nombreuses règles : par exemple il est obligatoire de faire un mouvement conjoint à la barre de mesure, or si l'on veut respecter la règles harmoniques (seul l'unisson, la tierce, la quinte, la quinte augmentée et la sixte sont autorisés sur les temps forts) le seul moyen pour le glouton qui a commencé par faire des mouvements conjoints est de rater un mouvement conjoint et rater le mouvement contraire. Alors que la méthode optimale est de rater le premier mouvement conjoint afin de maximiser plus tard les mouvements conjoints et contraires.

Ce paradigme n'est donc pas optimal dans notre cas.

□

De plus, ce paradigme est en pratique loin d'être optimal pour notre problème et crée des situations pathologiques récurrentes en diminuant considérablement l'ensemble des partitions possibles (il écrit un mouvement conjoint ou contraire dès qu'il le peut).

La solution idéale serait donc d'énumérer toutes les possibilités et de choisir l'optimum global. Or, le nombre de solutions est borné par la fonction :

$$f : \left| \begin{array}{ccc} \mathbf{N} & \longrightarrow & \mathbf{N} \\ n & \longmapsto & 5^n \end{array} \right.$$

avec n la taille de partition en nombre de temps.

Il est donc impossible d'énumérer les contrepoints valides d'un cantus firmus de 20, 30 ou 40 temps. Il faut donc trouver une heuristique. L'idée est de trouver un maximum local en utilisant l'aléatoire : on fait tourner le programme sur un serveur de calcul pendant un temps indéfini proportionnel à la précision attendue, on paramètre le programme pour qu'il enregistre uniquement les partitions dont les paramètres à maximiser sont supérieurs à un seuil satisfaisant, on augmente ainsi considérablement la rapidité en diminuant les écritures fichier. Le programme classe toutes les partitions valides aux valeurs intéressantes dans une arborescence de fichiers, il ne nous reste plus qu'à choisir la meilleure partition au bout de l'intervalle de temps qui nous convient. Pour éviter les doublons la technique consiste à utiliser la fonction de hachage MD5 comme nom de fichier, ainsi on ne peut avoir deux fichiers contenant la même partition.

Utilisations du programme

Ce programme nous permet de générer de "bonnes" partitions de contrepoint selon les critères de Marcel Bitsch et Noel Gallon. Il nous permet d'observer comment se comporte une utilisation indépendante de syntaxe de contrepoint, une étude comparative permet de remarquer les différences entre ce que fait le programme et la réalité des contrepoints écrits par des compositeurs et ainsi mettre en évidence des règles implicites. Il peut aussi être utile pour des applications d'aide à la composition ou de génération de musique en temps réel -jeux vidéo par exemple.

L'approche statistique de la modélisation des langages musicaux

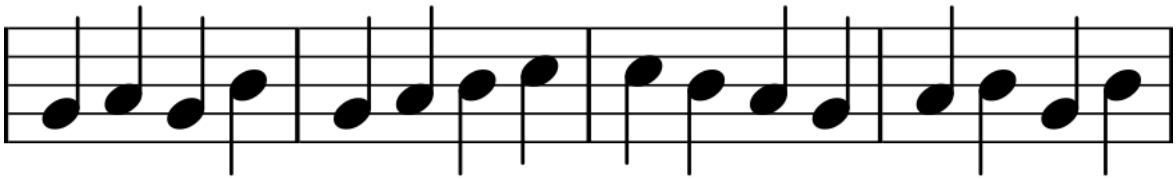
Il est très rare que des langages comme le contrepoint soient aussi bien définis, dans le cas contraire il faut trouver une autre façon d'établir ces règles. Une méthode très courante est l'apprentissage statistique. Elle consiste à étudier un large corpus d'un langage, un programme en définit l'alphabet et calcule la probabilité d'apparition de chacune des lettres en fonction des n précédentes : on parle de modèle de $n - gramme$ qui correspond à un modèle de Markov d'ordre n.

Ce genre de démarche en traitement automatique des langues a conduit à la modélisation plus ou moins précise de syntaxes -notamment en langue anglaise, avec de grands

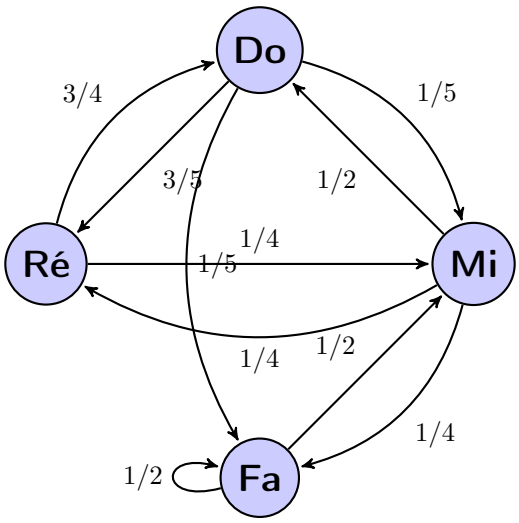
corpus comme l'œuvre de Shakespeare [9]. Si l'on applique ce modèle au langage musical, on modélise la syntaxe commune à tout le corpus, c'est à dire le style musical ou le style de jeu d'un musicien. Ce genre d'analyse statistique peut être utilisé pour établir des clusters de compositeurs et ainsi permettre à un programme après une écoute d'un morceau inconnu de déterminer son auteur.

Exemple de modèle de Markov sur une mélodie simple

Nous allons ici construire une chaine de Markov correspondant à la mélodie suivante, cela permettra de comprendre comment procéder pour de grands corpus.



On a donc quatre états : Do, Ré, Mi et Fa, on calcule maintenant le poids des transitions :



9. Le site internet *What would Shakespeare have said ?* permet de prédire statistiquement la fin de phrases selon la "grammaire de Shakespeare". L'article *Natural language processing : What would Shakespeare say ?* du site Gigadom se propose de comparer ces résultats avec un programme similaire modélisant la grammaire anglaise actuelle.

A partir de cette chaine de Markov on déduit une grammaire probabiliste simple qui nous dit par exemple que l'on ne peut pas jouer de Ré après un Fa. Si l'on procéde sur de grands corpus, les gestes musicaux anecdotiques sont effacés et les gestes idiomatiques quant à eux sont accentués. On modélise donc la syntaxe du corpus. On peut ensuite utiliser ce modèle pour faire de la génération, il suffit de partir d'un état donné et de tirer une variable aléatoire X dans un ensemble E et de définir autant de sous ensemble de E que de transitions, si $X \in E_i$ alors on choisit la transition i. On génère alors des compositions "à la manière de".

C'est cette méthode que Jon Gillick, Kevin Tang et Robert M. Kellerdes des universités de Standford, Cambridge et Harvey Mudd College ont utilisée. [10].

Leur but fut de générer des improvisations de solo de Jazz en utilisant des grammaires probabilistes (premiers travaux par Keller en 2007), l'article présente l'apprentissage automatique de telles grammaires. Ils partent de corpus de plusieurs musiciens de Jazz reconnus (Keith Jarett, Miles Davis, etc ...) et établissent des modèles probabilistes pour chacun d'entre eux afin de capturer des gestes idiomatiques de ces musiciens. En utilisant chacun de ces modèles pour la génération, le programme permet de générer des solos à la manière de célèbres musiciens et ce sur n'importe quelle grille d'accords. Un programme utilisant ce principe fut mis en place par Robert Keller de l'Université de Harvey Mudd College, il s'appelle Impro-Visor et on le trouve aisément sur internet.

Ces travaux peuvent être extrêmement intéressants d'un point de vue musicologique, il permettent d'apprendre rapidement beaucoup de choses sur un style musical ou un jeu de musicien. Les musiciens de Jazz seront par exemple ravis de pouvoir se servir de tels programmes pour récupérer des phrasés intéressants sans avoir à écouter toute la discographie d'un musicien.

Conclusion

Dans ce travail nous avons vu comment il était possible de modéliser des grammaires musicales grâce à des principes d'informatique théorique. Cela promet beaucoup d'avenir à la collaboration des disciplines relevant de la musique et de l'informatique. Le premier programme de génération de contrepoint m'a par exemple permis de mettre en évidence des principes de composition non présents dans l'ouvrage de Marcel Bitsch et Noel Gallon, on parle alors de règles implicites. C'est en effet l'analyse de ces partitions qui peut permettre de mettre en évidence les aspects "non-modelisables" de l'écriture musicale, on peut ainsi mieux différencier ce qui dépend de la création et de la sensibilité de ce qui dépend de la syntaxe. Les analyses de grands corpus quant à elles peuvent nous permettre d'observer l'évolution des langages musicaux sous un autre regard, on peut étudier l'évolution des gestes idiomatiques des grands styles musicaux, observer l'apparition, la disparition de certains

10. Gillick, Jon, Kevin Tang, and Robert M. Keller. « Machine learning of jazz grammars. » Computer Music Journal 34.3 (2010) : 56-66.

motifs mélodiques ou harmoniques.

Il serait intéressant de continuer ces travaux d'une part en améliorant le plus possible la qualité des modélisations en utilisant des concepts issus de l'informatique, mais aussi de travailler en partenariat avec des musicologues ou musiciens afin de faire avancer ensemble l'état de l'art de cette discipline. Un prochain travail possible serait de mettre en place des outils d'analyse statistique vus dans la seconde partie sur des partitions issues du générateur de contrepoint et de véritables compositeurs de contrepoint afin de mettre en évidence les differences grammaticales entre les compositeurs et par rapport à la "grammaire idéale".

Références

[1] Olivier Carton. *Langages formels, calculabilité et complexité*. Vuibert, 28 octobre 2008.

[2] Thomas Cormen, Charles Leiserson, and Ronald Rivest. *Algorithmique*. DUNOD, 2010.

[3] Noël Gallon and Bitsch Marcel. *Traité de contrepoint*. Durand edition, jan 2009.

[4] Jon Gillick, Tang Kevin, and M. Keller Robert. Machine learning of jazz grammars. *Computer Music Journal p. 56-66.*, 34.3, 2010.

[5] Searle J.R. Minds, brains and programs. *The Behavioral and Brain Sciences*, 3, 1980.