Générateurs aléatoires sous contraintes : application à l'improvisation musicale

Thibaut Roperch

18 juin 2017

1 Remerciements

Je tiens à remercier toutes les personnes qui m'ont accompagné durant ces 10 semaines de TER.

Tout d'abord, j'adresse mes remerciements à M. Adrien Goëffon et M. Frédéric Saubion, enseignants chercheurs encadrant mon projet et à l'initiative de ce sujet d'étude pour leur disponibilité et leur accompagnement tout au long de ces 10 semaines, ainsi qu'à Dylan Bunel avec qui j'ai collaboré pour mener à bien ce projet.

Je tiens à remercier également mes camarades de promotion, dont Théo Voillemin pour ses conseils musicaux avisés, Vincent Hénaux et Nathan Desages pour leur présence et leur bonne humeur au quotidien ainsi que Maxime Leblanc pour l'aide qu'il m'a apportée lors de la rédaction de mon rapport.

Sommaire

2 Introduction

L'improvisation musicale peut parfois se rapporter à un processus de génération aléatoire de notes choisies dans une gamme donnée. En réalité, l'improvisation est orchestrée par l'esprit créatif du musicien, son style et les contraintes du morceau musical sur lequel il joue.

Improviser requiert ainsi de connaître les propriétés du morceau et de savoir les exploiter afin de permettre au musicien de produire une mélodie relativement fidèle à l'originale.

Le projet consiste à développer un générateur de suites de nombres aléatoires en respectant certaines contraintes.
Le générateur sera utilisé dans le but de reproduire le processus humain d'improvisation, en partant des contraintes d'un ensemble de mélodies provenant d'une ou plusieurs musiques d'origine. Les entiers générés seront associés à une hauteur de note. Ainsi, les improvisations algorithmiques pourront être converties dans un format musical classique puis être écoutées et appréciées.

Afin de mener à bien ce travail de recherche et de développement, celui-ci est organisé et réalisé en deux parties.

Le premier objectif de la première partie est de définir un format de représentation des mélodies ; il sera utilisé par le générateur aléatoire et par les programmes chargés d'extraire et d'épurer une piste d'un morceau musical afin de n'en garder que la mélodie. Le second objectif de cette partie est de générer un modèle de Markov à partir d'un ensemble de mélodies et d'en calculer les contraintes mélodiques associées.
Il est à noter que, pour cette étude, la rythmique d'un morceau n'est pas prise en compte. Seules les notes importent et constituent à elles seules une mélodie. Aussi, les accords de notes sont évités (les notes ne sont pas jouées simultanément).

La seconde partie est destinée à implémenter le générateur aléatoire ; il doit être capable d'utiliser les contraintes d'un ensemble de mélodies dans le but d'essayer de produire la meilleure suite de notes possible. La valuation d'une suite d'entiers par rapport à d'autres individus est un aspect important du générateur aléatoire sous contraintes. Une suite de notes est évaluée par rapport aux propriétés mélodiques données.

Durant les dix semaines consacrées à l'élaboration de ce projet, je me suis principalement concentré à la réalisation des objectifs de la première partie. Ce projet a été mené à bien en étroite collaboration avec Dylan Bunel, qui a travaillé sur la deuxième partie de ce TER.

Je présente dans ce rapport les recherches que j'ai entreprises dans le but d'éclaircir certaines parties du sujet. Je me suis principalement concentré à étudier le fonctionnement du processus markovien issu de la chaîne de Markov, ainsi qu'à investiguer sur les différents formats de fichiers musicaux.
Je décris également le travail que j'ai effectué pour répondre aux exigences du sujet, ainsi que les choix d'implémentation que j'ai faits.

3 Recherches en amont

3.1 Chaîne de Markov

3.1.1 Présentation

Une chaîne de Markov est un modèle de représentation de données basé sur le processus de Markov, un processus stochastique intégrant la propriété de Markov. Cette propriété est vérifiée si la probabilité qu'un nouvel élément soit tiré au sort ne dépende que de l'élément actuel.
Une mélodie peut être représentée par une chaîne de Markov. Les éléments se suivent, il est donc possible d'en extraire une matrice de statistiques. Les statistiques calculées sont les fréquences auxquelles un élément est suivit par un autre élément.

Cette matrice de statistiques peut servir à générer aléatoirement des nouveaux éléments de cette façon : un élément est pioché dans la liste des éléments `n+1` de `n`. Les éléments avec la plus haute fréquence ont plus de chance d'être tirés au sort que les éléments avec une fréquence plus faible.
Une telle matrice peut donc être utilisée à des fins d'improvisation.

Le processus de décision Markovien, qui consiste à choisir un élément `n+1` en partant d'un élément `n`, inclut une notion de récompense. Un agent est chargé de choisir l'élément qui viendra compléter la chaîne de Markov. Lorsqu'un élément est choisi, celui-ci donne à l'agent la récompense qui est associée à sa décision.
Dans une mélodie, chaque note se vaut. Il n'y a pas de note plus intéressante qu'une autre, une mélodie est évaluée dans son ensemble. Les récompenses sont donc égales.

Une chaîne de Markov peut être représentée par un automate de Markov. Les états sont les éléments de la chaîne, et une transition entre deux états A et B est la fréquence à laquelle l'élément B suit l'élément A dans la chaîne.

3.1.2 Exemple

Voici la matrice de la chaîne de Markov associée à la mélodie de l'annexe A, ainsi que l'automate généré par cette matrice :

	6(3)	11(3)	0(4)
6(3)	0.0	0.5	0.5
11(3)	1.0	0.0	0.0
0(4)	0.0	0.0	0.0

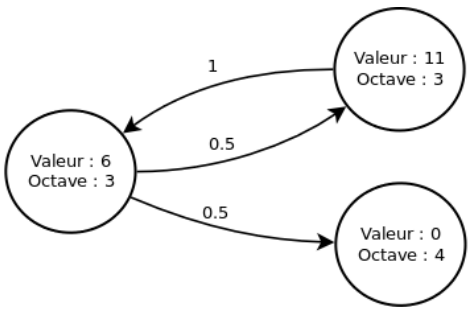

FIGURE 1 – Matrice de statistiques et automate associé

La matrice de la figure 1 se lit en lignes puis en colonnes. On lira par exemple :
"La note 6 de l'octave 3 est suivie à 50 % par la note 11 de l'octave 3 et à 50 % par la note 0 de l'octave 4".

3.2 Formats de notation musicale

3.2.1 Formats existants

Il existe plusieurs formats de fichier pour la notation musicale, tels que le format MIDI (Musical Instrument Digital Interface) et le format MusicXML, qui sont les plus populaires et les plus utilisés.

Le format MIDI utilise des messages destinés à jouer et arrêter une note codée sur sept bits, ce qui permet de représenter des notes réparties sur dix octaves. Un message MIDI contient également la vélocité de la note, la piste sur laquelle elle est jouée, sa position dans la piste et sa durée.
Des logiciels peuvent lire des fichiers formatés en MIDI, comme *Finale* ou *MuseScore*. Ils affichent le fichier musical sous forme de partition de musique.

Le format MusicXML est, comme son nom l'indique, un format de fichier basé sur le métalangage XML. L'avantage de ce format par rapport au MIDI est qu'il ne nécessite pas de librairie spécifique pour pouvoir être lu, le rendant facilement exploitable par un programme. Ainsi, bon nombre de logiciels acceptent ce format, comme *Finale*, *MuseScore* et *GuitarPro*.
Ce type de fichier est défini par deux DTD distinctes décrivant chacune une structure que le fichier MusicXML peut adopter :

score-partwise Les parties musicales (une partie par instrument) sont primaires, et les mesures sont contenues dans chaque partie. On retrouve ainsi le même ensemble de mesures dans chaque partie.

score-timewise Les mesures sont primaires, et les parties musicales sont contenues dans chaque mesure. On retrouve donc le même ensemble de parties dans chaque mesure.

Les notes y sont écrites en notation anglaise : la valeur d'une note est entre A (La) et G (Sol). Les demi-tons sont gérés par la balise *alter*, qui peut valoir *-1* dans le cas d'un bémol (un demi-ton en dessous de la note), ou *1* dans le cas d'un dièse (un demi-ton au-dessus de la note).
L'octave d'une note est représentée par un entier supérieur ou égal à 0.

Les accords sont gérés avec la balise *chord*. La première note d'un accord est suivie par ses autres notes, chacune précédée par la balise *chord*.

Les deux formats de fichiers musicaux seront acceptés par les programmes chargés d'extraire une mélodie à partir d'une partition.

3.2.2 Mon format de mélodie

Après avoir étudié les formats musicaux existants, je pense que le métalangage XML est un format adapté pour représenter une mélodie. Les informations à représenter sont les suivantes :

Une suite de notes Les notes composant la mélodie sont listées dans la balise *notes*.

Une note Une note est décrite par une valeur et une octave dans la balise *note*. De plus, une note possède un identifiant unique ; cet attribut est noté *id*.

Une valeur de note La valeur d'une note est sa hauteur dans son octave. Elle est relative à la note la plus basse d'une octave et tient compte des demi-tons ; elle vaut donc entre 0 et 11. Je considère qu'une octave possède 12 hauteurs de note différentes. Certains modèles divisent l'octave en 19 notes au lieu de 12, pour traiter différemment les dièses et les bémols.

Une octave de note Comme en MusicXML, l'octave d'une note est représentée par un entier supérieur ou égal à 0.

Un exemple de mélodie au format choisi est disponible en annexe A. Ce format de représentation de mélodie est décrit par la DTD du fichier `DTD/melodie.dtd`.

3.3 Format des contraintes et des autres propriétés

Une mélodie possède plusieurs propriétés mélodiques. On pourrait calculer son amplitude, la répartition des notes qui la composent, ou encore rechercher les patterns qui la caractérisent. Certaines de ces propriétés seront exploitées par le générateur aléatoire sous contraintes, les autres serviront de critères de comparaison entre deux mélodies.
Les trois contraintes initiales ont été convenues avec nos enseignants encadrants, en nous laissant la liberté d'en implémenter d'autres par la suite.

Dans un souci de cohérence avec le format de représentation d'une mélodie, j'ai opté pour une représentation des contraintes et des propriétés mélodiques en XML.
Les contraintes à représenter sont les suivantes :

Note min et note max La note la plus basse et la note la plus haute d'une mélodie.

Allure de la mélodie L'allure moyenne d'une mélodie est représentée par un rectangle. Ce rectangle est défini par une hauteur, une largeur et une précision de couverture des notes. Ce dernier paramètre autorise une relaxation de la contrainte, pour éviter qu'elle ne soit trop forte, ou trop contraignante. Il est ainsi appelée *objectif* pour le générateur aléatoire sous contraintes.

Couples de notes Cette contrainte assure qu'une note *n1* est à une distance d d'une note *n2*, et ce x fois sur 100. Ici aussi, le dernier paramètre de cette contrainte permet de laisser au générateur aléatoire une liberté d'improvisation.

Les autres propriétés mélodiques à représenter sont les suivantes :

Motifs de notes (patterns) Les plus grands patterns détectés dans une mélodie. Un pattern est décrit par sa taille en nombre de notes, le nombre de fois qu'il est présent dans la mélodie, son amplitude (la différence entre sa note la plus haute et sa note la plus basse), ainsi que ses positions dans la mélodie à laquelle il appartient.

Répartition des notes La proportion d'une note dans un ensemble de notes. Ici, la différence d'octave entre deux notes n'est pas faite, afin d'en déduire plus tard la tonalité de la mélodie. La somme de la proportion de chacune des douze notes vaut 1.

Un exemple de modélisation des contraintes est disponible en annexe B, C et D. Un exemple de modélisation des autres propriétés mélodiques est disponible en annexe E et F. Le format de modélisation d'une mélodie est décrit par la DTD du fichier `DTD/modelisation.dtd`.

4 Travail réalisé

4.1 Résumé des objectifs

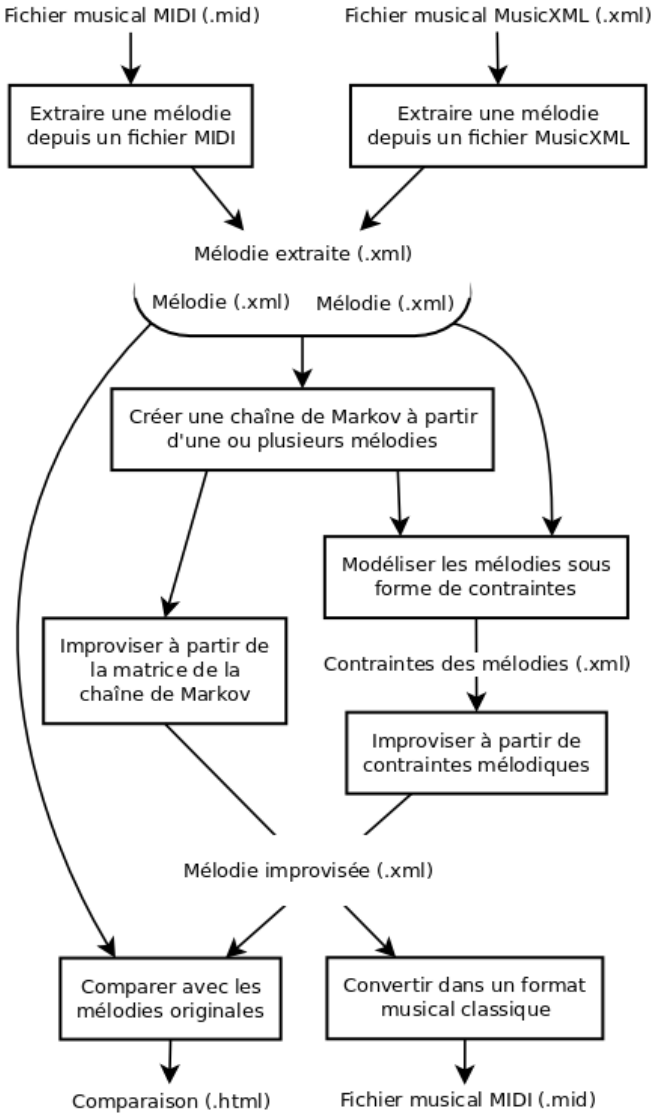

FIGURE 2 – Résumé des objectifs

4.2 Extraction d'une mélodie depuis un fichier musical

Un fichier musical peut être de deux formats différents, il y a ainsi deux programmes d'extraction de mélodie. Les deux programmes chargés d'extraire la mélodie d'une partition au format MIDI et au format MusicXML sont respectivement `ModeleMarkov/src/epurer_midi.cpp` et `ModeleMarkov/src/epurer_musicxml.cpp`.

Une partition peut contenir plusieurs parties musicales. Lorsqu'une mélodie est extraite d'un fichier, elle est épurée pour conserver seulement la hauteur des notes et oublier leur durée. Les accords sont traités de sorte à n'en garder que la première note.

4.2.1 Parties musicales

Lorsque le fichier musical comporte plus d'une partie, l'utilisateur est invité à donner l'indice de la partie qu'il veut extraire.
Dans un fichier au format MusicXML, les parties sont décrites. Il est donc possible de récupérer leur nom, souvent associé à l'instrument de musique jouant cette partie. En MIDI, la librairie que j'utilise ne permet pas de récupérer le nom des parties. Il faut alors ouvrir le fichier avec un programme de visualisation de partitions et consulter l'ordre des parties musicales. Parfois, la piste 0 d'une partition MIDI est une piste d'expression, elle ne contient qu'un message MIDI qui donne la dernière note de chaque piste.

4.2.2 Accords de notes

Les accords sont traités de la façon suivante : seule la première note de chaque accord est extraite et ajoutée à la mélodie.
En MusicXML, les notes précédées par la balise *chord* sont ignorées. En MIDI, la librairie implémente une méthode permettant de connaître la position d'une note dans une piste. Si la position d'une note est identique à celle de la note précédente, il suffit de l'ignorer et de passer à la note suivante.

4.2.3 Transcription des notes

La hauteur d'une note n'est pas représentée de la même manière dans un fichier au format MusicXML, utilisant la notation anglaise pour la valeur de la note, que dans un fichier au format MIDI, associant un entier à chaque note. Ainsi, j'ai établi et j'utilise la correspondance de notes suivante (pour la première octave, notée octave 0) :

Notation française	MusicXML	MIDI	Ma notation
Do	C	12	0
Do# ou Réb	C+1 ou D-1	13	1
Ré	D	14	2
Ré# ou Mib	D+1 ou E-1	15	3
Mi	E	16	4
Fa	F	17	5
Fa# ou Solb	F+1 ou G-1	18	6
Sol	G	19	7
Sol# ou Lab	G+1 ou A-1	20	8
La	A	21	9
La# ou Sib	A+1 ou B-1	22	10
Si	B	23	11

En MIDI, l'octave d'une note n'est pas représentée : sa hauteur dépend de son octave et de sa valeur. La note la plus basse de l'octave 0 vaut 12. Avec ma notation, et comme en MusicXML, la valeur d'une note est toujours contenue dans un intervalle fixe - entre 0 et 11 inclus - et son octave est dissociée de sa valeur.
Ainsi, lorsqu'une note MIDI est lue, sa valeur et son octave sont recalculées de cette façon :

$$octave(x) = (x - 12) \ / \ 12$$

$$valeur(x) = (x - 12) \ \% \ 12$$

4.3 Modélisation d'un ensemble de mélodies

La modélisation d'une mélodie a pour objectif de récupérer certaines de ses propriétés et de s'en servir comme des contraintes d'improvisation, exploitées par le générateur aléatoire. Quant à elles, les propriétés mélodiques sont destinées à être utilisées par le programme qui est chargé de comparer un ensemble de mélodies originales avec une mélodie improvisée.

L'intérêt de modéliser un ensemble de plusieurs mélodies est de pouvoir improviser sur un style de mélodies, et non plus une seule mélodie.

Les aspects de la modélisation d'un ensemble de mélodies sont les suivants :
- Construction d'une chaîne de Markov par mélodie, calcul d'une matrice commune à toutes les chaînes.
- Calcul des propriétés mélodiques suivantes (les trois premières serviront de contraintes d'improvisation) :
  - Note la plus haute et note la plus basse
  - Allure de la mélodie représentée par un rectangle
  - Couple de notes
  - Motifs de notes (patterns)
  - Répartition des notes

Les contraintes et les autre propriétés de mélodies sont calculées et données par le programme `ModeleMarkov/src/modeliser.cpp`.

4.3.1 Chaîne de Markov

Une chaîne de Markov est construite à partir d'une ou plusieurs mélodies extraites de fichiers musicaux. Chaque note de chaque mélodie est ajoutée à un objet de type `ChaineMarkov` via une méthode d'ajout d'élément.

À l'ajout, le nouvel élément est mis en fin de chaîne. Si celle-ci contient maintenant deux éléments ou plus, la liste des éléments suivant l'élément précédemment rajouté est actualisée : le dernier élément ajouté à la chaîne de Markov s'ajoute à la liste des suivants de l'avant dernier élément.

La chaîne est vidée de ses éléments lorsqu'une autre mélodie est lue, car les mélodies ne se suivent pas forcément. En d'autres termes, la première note d'une mélodie n'est pas précédée par la dernière note de la mélodie précédemment lue.

Lorsque toutes les notes de toutes les mélodies ont été ajoutés à la chaîne, la construction de la matrice de statistiques peut être engagée. Les valeurs sont calculées à partir des occurrences des éléments suivant un élément, divisées par le nombre total d'éléments suivants. Ainsi, la somme de chaque ligne de la matrice vaut 1. Un exemple de matrice est disponible en page 5 (figure 1).

La chaîne de Markov est une classe générique (ou patron de classe), implémentée dans le fichier `ModeleMarkov/src/chaine_markov.tpp`. Elle peut donc être réutilisée pour n'importe quel type d'élément, à condition que celui-ci possède les méthodes requises par la chaîne de Markov dont voici la signature, pour un élément de type T :

- `ajouterAuxSuivants : T* -> void`
- `piocherParmiSuivants const : void -> T*`
- `occurencesDesSuivants const : void -> map<T*, int>`
- `nombreDeSuivants const : void -> int`

4.3.2 Note la plus haute et note la plus basse

Lorsqu'une note est lue, elle est comparée à la note la plus basse enregistrée à ce stade de la mélodie. Si elle est strictement inférieure, la note lue est la nouvelle note la plus basse parmi toutes les notes lues jusqu'ici, toutes mélodies confondues.

Même principe pour trouver la note la plus haute d'un ensemble de mélodies : si une note est strictement supérieure à la note enregistrée comme étant la plus haute, elle la remplace.

Une note est supérieure à une autre si sa hauteur est strictement supérieure. La hauteur d'une note est calculée comme suit :

$$hauteur = valeur + octave * 12$$

4.3.3 Allure de la mélodie représentée par un rectangle

L'objectif de cette contrainte est de définir, pour toutes les notes de la mélodie, un rectangle d'une hauteur et d'une largeur fixes dans lequel est contenu un certain pourcentage de notes.

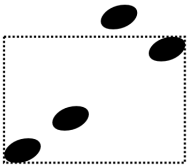

FIGURE 3 – Un exemple de rectangle appliqué à une note

Dans l'exemple ci-dessus, le rectangle a une hauteur et une largeur de 4 notes. Il couvre dans cette situation trois notes sur quatre, sa précision est donc de 75 %.

Le rectangle d'une mélodie doit se dimensionner en fonction de son allure. En effet, plus les notes de la mélodie sont proches en hauteur, plus la largeur du rectangle sera importante. À l'inverse, si l'écart de hauteur entre les notes est important, la hauteur du rectangle aura tendance a être supérieure à sa largeur.

Ainsi, la façon dont les dimensions du rectangle sont calculées dépend principalement de la différence de hauteur qu'il y a entre deux notes consécutives. Au fur et à mesure que la mélodie évolue, le rectangle se façonne en fonction de l'écart de hauteur séparant deux notes consécutives.
Je me sers également d'un autre paramètre : la monotonie. Elle représente la tendance qu'a un groupe de notes à se répartir sur une hauteur donnée. Plus la monotonie est forte, plus la largeur du rectangle tend à être importante.

Le rectangle d'une mélodie est donc construit lors de la lecture de chacune des notes qui la composent. Les dimensions du rectangle sont altérées de la façon suivante :

hauteur À la lecture d'une nouvelle note, la moyenne des écarts de hauteur entre deux notes consécutives est recalculée. La moyenne calculée à l'itération précédente est soustraite à cette moyenne, la différence est ajoutée à la hauteur du rectangle. Cette valeur peut être négative.

lageur La largeur du rectangle augmente en fonction de la monotonie de la mélodie. Si les `x + 1` notes précédentes étaient monotones au moment de leur lecture, avec `x` étant la largeur du rectangle, alors la largeur du rectangle augmente de `1`. Pour qu'une note soit monotone, il faut que sa différence de hauteur avec la note précédente soit inférieure ou égale à la différence de hauteur moyenne entre deux notes consécutives. Ainsi, une note qui est trop haute ou trop base par rapport à sa voisine cassera la monodie de la mélodie.

Lorsque plusieurs mélodies sont modélisées ensemble, le programme calcule plusieurs rectangles, à raison d'un par mélodie. Celui-ci effectue une moyenne sur leurs dimensions pour obtenir un rectangle moyen, ainsi fidèle à toutes les mélodies.
Cette fidélité est appelée *précision*, ou *objectif* par le générateur aléatoire sous contraintes.

Pour la calculer, toutes les mélodies doivent être relues, en appliquant à chaque note le rectangle moyen obtenu précédemment.

La difficulté principale du calcul de la précision réside dans le fait que le rectangle peut avoir, pour une note donnée, plusieurs positions possibles. Voici un exemple explicitant cette situation avec un rectangle de largeur 4 et de hauteur 2 :

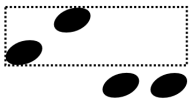

FIGURE 4 – Position possible n ° 1

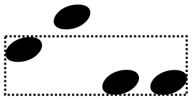

FIGURE 5 – Position possible n ° 2

On peut voir ici que le rectangle en seconde position couvre plus de notes que celui en première position. La position n ° 2 est donc retenue. Ainsi, pour cette note et ce rectangle, la précision est de `0.75`.
Le rectangle est appliqué à chaque note de la mélodie. On obtient ainsi une précision moyenne du rectangle. L'opération est répétée pour chacune des mélodies, jusqu'à obtenir une précision globale du rectangle.

4.3.4 Couples de notes

Cette contrainte permet d'assurer que des couples de notes soient présents dans la mélodie générée. Un couple est défini par deux notes, une distance les séparant et la probabilité à laquelle le couple peut exister.

L'utilité de la chaîne de Markov est justement d'étudier les notes deux à deux et d'en tirer des conclusions statistiques. Ainsi, pour calculer cette contrainte, je me sers de la matrice obtenue à partir de la chaîne de Markov.
Par exemple, si la matrice indique que 50 % des notes suivant 6(3) sont des 11(3), alors le couple 6(3) - 11(3) sera ajouté à la contrainte en tant que couple de notes, avec les paramètres `distance = 1` et `probabilité = 0.5`.

Étant donné que la matrice est construite avec un ensemble de mélodies, la contrainte comporte tous les couples de notes de toutes les mélodies.

4.3.5 Motifs de notes (patterns)

Un pattern est un ensemble de notes répété au moins deux fois dans une même mélodie à deux endroits différents. J'ai considéré qu'un pattern était intéressant et caractéristique d'une mélodie lorsqu'il est composé d'au moins trois notes.

La détection de patterns au sein d'une mélodie se déroule en deux temps :

1. Recherche de tous les motifs
2. Tri sur les patterns trouvés

La recherche de tous les motifs consiste à, pour chaque note, essayer de la retrouver plus loin dans la mélodie. Si c'est le cas, les notes suivantes sont comparées afin de déterminer la taille du potentiel pattern. Si la suite de notes a une taille équivalente ou supérieure à trois, le motif est enregistré dans un tableau associatif de la forme suivante :

clef Le pattern (suite de notes)

valeur Les positions auxquelles le pattern est présent dans la mélodie

Dans tous les cas, la recherche continue avec la note suivante. La quête de motifs pourrait être optimisée en ignorant les x prochaines notes de la mélodie, avec x la taille du pattern précédemment détecté, mais dans certaines situations, cette solution risque d'omettre des patterns qui auraient pu être intéressants. Voici un exemple représentatif de cette situation :

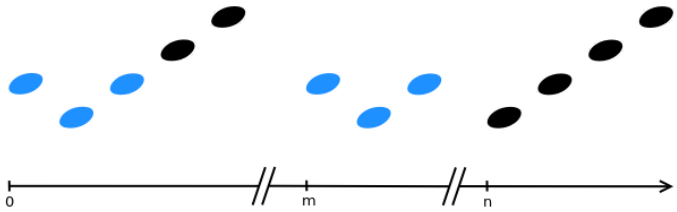

FIGURE 6 – Pattern bleu détecté

La première note du pattern, en position `0`, est recherchée dans la suite de la mélodie. À la fin du parcours, nous constatons qu'un seul pattern est détecté : le pattern bleu. Il est trouvé aux positions `0` et `m`.
Nous pourrions envisager de reprendre la recherche en partant de la note en position `3`, pour passer le pattern déjà trouvé. Or, si nous faisions cela, le pattern de quatre notes qui est aux positions `1` et `n` ne serait jamais détecté alors qu'il est tout aussi intéressant que le pattern bleu, voire plus étant donné sa longueur.

Ainsi, lorsque l'algorithme de recherche de patterns est terminé, il est possible que le tableau associatif contienne des patterns utilisant les mêmes notes. La superposition de patterns n'est pas forcément gênante d'un point de vue musical, mais elle l'est si l'objectif est de trouver les patterns qui caractérisent la mélodie ; parmi deux patterns qui exploitent les mêmes notes, il y en a sûrement un qui est plus intéressant que l'autre. De plus, il y en a possiblement un qui est un sous-pattern de l'autre. C'est pourquoi un tri sur les patterns est nécessaire.

Le tri de pattern constitue la seconde partie de cette problématique de détection de patterns.
La liste des patterns est parcourue, ils sont vérifiés deux à deux ; s'ils se chevauchent (si un motif commence ou se termine dans l'autre), ils sont comparés afin de n'en garder qu'un. Chacune de leurs positions doit être contrôlée. La comparaison peut s'effectuer sur plusieurs étapes :

Dans un premier temps, leur présence au sein de la mélodie est appréciée ; celui qui est le plus présent est conservé. La proportion d'un motif est calculée de la façon suivante :

$$proportion = taille * nb\_positions$$

Si les deux patterns de la mélodie ont la même proportion, leur taille est alors comparée pour tenter d'éliminer le pattern le plus petit. Mais encore une fois, il n'est pas impossible qu'il en résulte une égalité.

Le dernier critère de sélection mesure la dispersion d'un pattern. Le calcul est le suivant, avec `n = nb_positions` :

$$dispersion = \sum_{i=0,\ i \neq n}^{n} position_{i+1} - position_i$$

Le motif qui est réparti à des endroits plus éloignés que l'autre est gardé. S'il y a ici aussi une égalité des valeurs, les deux patterns sont conservés, jugés tous deux aussi importants pour la mélodie.

Lorsque plusieurs mélodies sont modélisées ensemble, les patterns trouvés dans chacune d'elles sont réunis.

4.3.6 Répartition des notes

Calculer la répartition des notes d'une mélodie permet de calculer plus tard sa tonalité. L'octave n'importe pas pour cette partie, ainsi les deux notes 6(3) et 6(4) compteront comme une même note, à savoir 6.

La répartition d'une note est exprimée en pourcentage de notes sur le nombre total de notes composant la mélodie. Dans le cas de plusieurs mélodies, le calcul est effectué sur toutes les notes de toutes les mélodies.

4.4 Génération d'une mélodie (improvisation)

Le sujet propose d'improviser sur une ou plusieurs mélodie(s) avec un générateur aléatoire sous contraintes. Il pourrait être intéressant de comparer une mélodie improvisée par un tel générateur avec une mélodie générée totalement aléatoirement, normalement très peu fidèle à l'originale, et une mélodie issue de la chaîne de Markov, supposément très fidèle au style d'origine.

4.4.1 Improvisation markovienne

L'improvisation markovienne utilise les données de la matrice obtenue à partir de la (des) chaîne(s) de Markov de la (des) mélodie(s) d'origine. Ces statistiques sont utilisées comme des probabilités ; le dernier élément généré produit à son tour un élément en fonction de la fréquence des éléments qui le suivent dans la matrice, représentative des éléments qui le suivent dans la (les) mélodies originale(s).
La première note de l'improvisation est choisie parmi toutes les notes rencontrées dans les mélodies d'origine, chacune ayant la même probabilité d'être tirée au sort.

L'improvisation issue d'une chaîne de Markov est assez fidèle à la mélodie originale, car elles contient exactement les mêmes couples de notes, donc possiblement la même allure, les mêmes patterns, la même tonalité ainsi que les mêmes statistiques, surtout si la mélodie générée comporte beaucoup de notes.

4.4.2 Improvisation avec contraintes

L'improvisation avec contrainte utilise le fichier de contraintes généré par le programme de modélisation de mélodies.

Cet aspect du sujet a été traité par Dylan Bunel. Il explique en détails dans son rapport la problématique, les choix effectués et le fonctionnement de la génération aléatoire sous contraintes.

4.5 Comparaison de mélodies

Il est difficile de juger de la qualité d'une mélodie, sinon en l'écoutant et en donnant un avis relativement objectif. Ainsi, avec le programme `ModeleMarkov/src/comparer.cpp`, je tente de juger la mélodie improvisée en comparant sa structure avec celle des mélodies d'origine.

La comparaison de mélodies reprend le fichier contenant la mélodie improvisée, ainsi que les fichiers de contraintes générés par le programme de modélisation (un fichier pour les mélodies originales, un autre pour la mélodie générée).
Les valeurs des contraintes sont comparées pour évaluer la qualité de l'improvisation par rapport aux mélodies sources, et ce sur plusieurs critères :

L'amplitude L'intervalle de notes de l'improvisation est comparé avec l'original. Si les deux intervalles n'ont aucune note en commun, l'amplitude de la mélodie improvisée sera évaluée à `0`. Sinon, le nombre de notes en commun entre les deux intervalles est divisé par le nombre total de notes couvertes par les mélodies (l'intersection des intervalles divisée par l'union de ceux-ci).

L'allure Les dimensions des rectangles sont comparées ; si elles sont différentes, l'aire du rectangle de l'improvisation est évaluée par rapport à celle du rectangle original. Sinon, plus la précision du rectangle cible est proche de celle du rectangle source, plus la valuation de l'improvisation est élevée.

La chaîne de Markov La matrice de la chaîne de Markov de la mélodie générée est soustraite à la matrice originale. La moyenne des différences de statistiques des notes est calculée. Plus cette moyenne est faible, plus les matrices sont proches, donc plus la valuation de la mélodie générée augmente. De plus, la chaîne de Markov associée à l'improvisation est envoyée dans la matrice originale afin de calculer sa récompense. Pour rappel, une transition d'une note vers une autre note rapporte `1` point si elle est présente dans la matrice.

Les patterns La valeur moyenne, maximale et minimale des patterns improvisés sont comparées aux valeurs des patterns originaux. Plus précisément, la valeur d'un pattern est obtenue de la façon suivante :

$$valeur = taille * amplitude * nb\_positions$$

Cette contrainte dépend fortement du nombre de notes contenues dans une mélodies : plus il y a de notes, plus le taux de présence de patterns est fort.

La répartition des notes La proportion de chaque note de l'improvisation est comparée avec la proportion de cette même note dans les mélodies d'origine. Une fois de plus, la répartition des notes est très sensible au nombre de notes contenues dans la mélodie. La tonalité est obtenue en prenant les huit notes les plus présentes dans la mélodie (celles dont la proportion est la plus élevée). La tonalité des deux mélodies est appréciée en comparant les notes les plus présentes ; si les huit notes sont les mêmes, alors la tonalité est la même. Ce critère est binaire.

Un indice de ressemblance de l'improvisation avec les mélodies originales en est déduit, exprimé en pourcentage.
Pour que cet indice soit représentatif, j'ai effectué des tests sur plusieurs mélodies générées. L'indice de ressemblance dépend des mélodies originales, de leur nombre de notes, du type d'improvisation, du nombre de notes de la mélodie générée, ...

Ainsi, en moyenne, les mélodies générées avec Markov sont fidèles à l'originale avec un indice allant de 70 % à 90 %, pour un nombre de notes allant de 20 à 250. Les mélodies générées avec le générateur aléatoire sous contraintes sont fidèles de 60 % à 80 %, pour un nombre de notes allant de 20 à 250. Les mélodies générées aléatoirement dépassent rarement les 40 % de fidélité pour 250 notes générées.
On peut donc déduire que les mélodies générées avec Markov sont *trop* fidèles à la mélodie originale, et que le générateur sous contraintes est un bon compromis entre génération aléatoire et fidélité avec le morceau d'origine.

4.6 Conversion d'une mélodie vers un fichier musical

Une mélodie générée peut être convertie au format MIDI. Le procédé de création et d'extraction de mélodie de fichier MIDI utilisent la même librairie.

Le programme `ModeleMarkov/src/convertir_midi.cpp` lit une suite de notes contenue dans un fichier au format XML et l'écrit dans un fichier MIDI, dans la piste d'indice 1.

4.7 Interface graphique

Mon interface graphique est capable de lire et jouer une suite de notes écrite dans le format XML présenté dans la section *3.2.2.*
Il est également possible d'y afficher certaines propriétés calculées lors de la modélisation de la mélodie, comme les rectangles lissant la mélodie et ses patterns caractéristiques.

Deux visualisations différentes sont proposées :

1. Les notes sont affichées à la suite, sur la même ligne. Une note est de la forme X(Y), avec X la valeur de la note et Y son octave. Cette vue permet d'afficher les caractéristiques numériques des notes.
2. Les notes sont disposées à la manière d'une partition musicale. Une note est affichée sur un axe vertical, plus ou moins haute en fonction de sa valeur et de son octave.

Cette visualisation permet de se rendre compte rapidement de l'amplitude des notes et peut facilement se donner une idée de la mélodie.

L'objectif initial de cette interface est de pouvoir écouter les mélodies extraites de fichiers musicaux MIDI ou MusicXML, ainsi que les mélodies improvisées sur ces morceaux et d'afficher leurs propriétés calculées. Lorsque la mélodie est jouée, les notes lues sont mises en valeur ; il est ainsi facile de suivre l'évolution de la mélodie.

L'interface permet également de simplifier les étapes à franchir et les commandes à exécuter pour improviser depuis un ensemble de fichiers musicaux. En effet, il suffit d'y glisser déposer les partitions sur lesquelles on veut improviser, choisir un ou plusieurs type(s) de génération parmi la génération aléatoire sous contrainte, la génération totalement aléatoire et le processus de décision markovien.
Une fois la sélection des partitions faite, l'interface donne la commande à exécuter pour générer les mélodies. Cette commande exécute un script qui appelle les programmes chargés d'extraire une mélodie d'un fichier musical, de modéliser une mélodie, d'improviser et de comparer l'improvisation avec l'ensemble des mélodies originales.
Lorsque les fichiers contenant une mélodie extraite ou improvisée sont créés, l'interface s'actualise et affiche les mélodies générées, avec la possibilité de les écouter et de les convertir au format MIDI.

L'interface est réalisée en HTML5/CSS3 et JavaScript, le script est codé en Bash (`./launch.sh --help` pour consulter l'aide). L'interface lit les *logs* générés par le script, via AJAX, ce qui lui permet de savoir quand les fichiers sont générés et ainsi afficher les mélodies correspondantes.

4.8 Exécution des programmes

Chacun de mes programmes est affecté à une tâche particulière. Ils prennent deux fichiers en argument :
- Le fichier d'entrée qui sera exploité
- Le fichier de sortie dans lequel sera écrit le résultat de l'exécution du programme

5 Idées d'amélioration

La rythmique pourrait être ajoutée aux mélodies générées. Selon moi, le rythme apporte une autre dimension à la musique. Il faudrait établir de nouvelles contraintes, mettre à jour le programme de modélisation des propriétés mélodiques et les utiliser dans le générateur aléatoire.

On pourrait aussi envisager d'autoriser les notes simultanées, et ainsi générer des accords, voire improviser à partir d'un accord donné.

6 Conclusion

Tout au long de ce projet, j'ai eu occasion d'exploiter mes compétences acquises ces dernières années en programmation, notamment en C++.
J'ai également su mettre en application mes connaissances en Bash et en HTML/CSS/JS.

Même si nous ne travaillions pas sur la même partie du sujet, la collaboration avec Dylan Bunel a fait appel à mon esprit d'équipe et à ma faculté d'adaptation, car certains points de notre travail étaient à discuter et décider ensemble.
Bien que ce travail puisse être amélioré, je suis satisfait de l'aboutissement de ce projet. Les outils que nous proposons sont fonctionnels et exploitables, et répondent aux exigences du sujet.

Enfin, les recherches que j'ai effectuées pour cette étude m'ont permis de voir la musique d'un autre œil. En effet, J'ai été amené à raisonner sur la façon dont on improvise.
Moi-même étant musicien, je ne m'étais pas spécialement interrogé sur ce processus naturel, laissant faire mon imagination. Il a fallu que je m'y intéresse de près afin de le comprendre avant d'envisager de reproduire ce phénomène algorithmiquement.

7 Outils utilisés

7.1 Programmes

LilyPond Système de gravure musicale en mode texte
`http://lilypond.org/index.fr.html`

MuseScore Logiciel d'édition de partitions (MIDI, MusicXML entre autres)
`https://musescore.org/fr`

Finale Logiciel d'édition de partitions (MIDI, MusicXML entre autres)
`http://www.finalemusic.com/`

7.2 Librairies C++

RapidXml (1.13) Analyseur XML choisi pour lire les fichiers XML
`http://rapidxml.sourceforge.net/`

Midifile Lire et écrire des fichiers MIDI standards
`http://midifile.sapp.org/`

7.3 Librairies JavaScript

audiosynth Synthétiseur audio basé sur le procédé de forme d'onde dynamique
`https://github.com/keithwhor/audiosynth`

8 Bibliographie

Article au sujet de l'improvisation algorithmique :
"Le test de Turing-Parker : un ordinateur peut-il improviser comme Charlie Parker ?" (David Louapre)

Informations sur les processus et modèles associés à Markov :
- Modèle de Markov caché (Wikipédia)
- Chaine de Markov (Wikipédia)
- Processus de décision markovien (Wikipédia)
- Modèle de Markov appliqué à la médecine (C. Chouaid)
- Modèle de Markov appliqué au traitement d'images (Wojciech Pieczynski)
- Chaine de Markov appliquée à la médecine (Sory Traore, Gilles Hunault entre autres)
- Ouverture à l'apprentissage par renforcement via Markov (Rémi Munos)

Informations sur le format de fichier MusicXML pour la notation musicale :
- Fichiers musicaux au format MusicXML
- Représentation d'une note en MusicXML
- Types de structures MusicXML
- Accords en MusicXML

Relation entre les notes MIDI et le nom des notes (notation anglaise) :
MIDI Note to Pitch Table

A Exemple de mélodie au format choisi

```
<notes>
  <note id="0">
    <valeur>6</valeur>
    <octave>3</octave>
  </note>
  <note id="1">
    <valeur>11</valeur>
    <octave>3</octave>
  </note>
  <note id="2">
    <valeur>6</valeur>
    <octave>3</octave>
  </note>
  <note id="3">
    <valeur>0</valeur>
    <octave>4</octave>
  </note>
</notes>
```

B Exemple de fichier de contraintes : note min-max

```
<elements-min-max>
  <element-min>
    <note>
      <valeur>6</valeur>
      <octave>3</octave>
    </note>
  </element-min>
  <element-max>
    <note>
      <valeur>0</valeur>
      <octave>4</octave>
    </note>
  </element-max>
</elements-min-max>
```

C Exemple de fichier de contraintes : rectangles

```
<rectangles>
  <rectangle>
    <objectif>0.833333</objectif>
    <hauteur>5</hauteur>
    <largeur>3</largeur>
  </rectangle>
</rectangles>
```

D Exemple de fichier de contraintes : couples de notes

```
<couples-notes>
  <couple>
    <note>
      <valeur>6</valeur>
      <octave>3</octave>
    </note>
    <note>
      <valeur>11</valeur>
      <octave>3</octave>
    </note>
    <distance>1</distance>
    <probabilite>0.500000</probabilite>
  </couple>
  <couple>
    <note>
      <valeur>6</valeur>
      <octave>3</octave>
    </note>
    <note>
      <valeur>0</valeur>
      <octave>4</octave>
    </note>
    <distance>1</distance>
    <probabilite>0.500000</probabilite>
  </couple>
  <couple>
    <note>
      <valeur>11</valeur>
      <octave>3</octave>
    </note>
    <note>
      <valeur>6</valeur>
      <octave>3</octave>
    </note>
    <distance>1</distance>
    <probabilite>1.000000</probabilite>
  </couple>
</couples-notes>
```

E Exemple de fichier de contraintes : patterns

```
<patterns>
  <pattern id="0">
    <taille>13</taille>
    <nombre>3</nombre>
    <amplitude>8</amplitude>
    <positions>
      <indice>15</indice>
      <indice>50</indice>
      <indice>85</indice>
    </positions>
  </pattern>
</patterns>
```

F Exemple de fichier de contraintes : répartition des notes

```
<repartition-notes>
  <note-unique valeur="0">0.250000</note-unique>
  <note-unique valeur="1">0.000000</note-unique>
  <note-unique valeur="2">0.000000</note-unique>
  <note-unique valeur="3">0.000000</note-unique>
  <note-unique valeur="4">0.000000</note-unique>
  <note-unique valeur="5">0.000000</note-unique>
  <note-unique valeur="6">0.500000</note-unique>
  <note-unique valeur="7">0.000000</note-unique>
  <note-unique valeur="8">0.000000</note-unique>
  <note-unique valeur="9">0.000000</note-unique>
  <note-unique valeur="10">0.000000</note-unique>
  <note-unique valeur="11">0.250000</note-unique>
</repartition-notes>
```