Analyse musicale et contraintes

Marc Chemillier, Charlotte Truchet
Université de Caen, IRCAM, Université de Paris 6

April 19, 2002

1 Introduction

Si l'utilisation des contraintes pour résoudre des problèmes de génération de musique est une pratique courante, leur utilisation dans le domaine de l'analyse musicale est une approche moins fréquente. On décrira dans cet article l'utilisation de contraintes pour l'analyse musicale sous la forme de CSPs permettant de modéliser l'analyse d'une (ou plusieurs) séquence musicale par un ensemble de contraintes dont la séquence analysée est une solution. Dans cette optique, le CSP proposé comme modèle est d'autant plus "proche" de la séquence analysée qu'il admet relativement peu de solutions, et qu'il capte ainsi une part de la spécificité de cette séquence. Le nombre de solutions d'un CSP modélisant une analyse musicale peut ainsi être vu comme un critère permettant d'évaluer la fidélité d'un modèle à la réalité musicale qu'il s'efforce de décrire : moins il y a de solution, plus le modèle est fidèle. Le cas limite est celui d'un CSP admettant comme solution unique la partition analysée (situation qui correspond aux "modélisations de partition" pratiquées par André Riotte et Marcel Mesnage [Mes91], [RiM88]).

Cette façon de voir l'analyse musicale permet de définir la notion de fidélité d'une analyse par rapport à la pièce musicale qu'elle modélise. Mais on ne dit rien sur la nature de l'analyse elle-même. Il va de soi que la qualité d'une analyse ne peut se réduire à la notion de fidélité que nous proposons ici, mais prend en compte d'autres aspects. Par exemple[1], une analyse d'une pièce musicale qui est codée informatiquement par une série de valeurs x_1, x_2, x_3, ... (codes midi ou autres) peut consister en des variables X_1, X_2, X_3, ... dont l'instanciation sera entièrement déterminée par une contrainte de la forme $X_1 = x_1 \wedge X_2 = x_2 \wedge X_3 = x_3 \wedge \ldots$. Une telle analyse est évidemment très fidèle. Mais elle est en même temps assez peu " analytique ", car essentiellement descriptive.

Bien que cela ne soit pas le propos de cet article, on peut préciser l'opposition entre " analytique " et " descriptif ", en se plaçant au niveau du langage de contraintes utilisé. Ce langage est défini formellement à partir de variables (X_1, X_2, X_3...), de connecteurs logiques ($\vee$, $\wedge$), de quantificateurs, et d'un ensemble de symboles non-logiques comprenant des relations (par exemple $<$), des fonctions

[1] Merci au referee qui nous a suggéré cet exemple.

(par exemple +), et des constantes (par exemple des codes midi). L'ensemble des termes du langage est le plus petit ensemble d'expressions contenant les variables et les constantes, et fermé par l'application des fonctions. Le langage de contraintes est le plus petit ensemble contenant les termes, et fermé par la combinaison de termes par relations ou connecteurs, et par la quantification des termes. En explicitant les éléments qui interviennent dans cette construction, on peut distinguer différentes classes de langages de contraintes (comme on le fait dans la distinction entre logique du premier et du deuxième ordre), et ceci conduit à définir différentes classes d'analyses. Ainsi, on pourra dire, par exemple, qu'une analyse est d'autant plus descriptive, que son langage de contraintes utilise plus de constantes.

Dans cet article, on ne s'intéresse pas à la classe des analyses considérées comme sous-classes particulières du langage général de contraintes défini ci-dessus, mais au nombre de solutions des CSP associés à ces analyses, et à la notion de fidélité d'une analyse musicale telle que nous l'avons introduite précédemment. Trois exemples musicaux sont étudiés dans cet article (textures de Ligeti, canons de harpe d'Afrique centrale, rythmes asymétriques d'Afrique centrale). Du point de vue de la technique de résolution de contraintes, le solveur utilisé repose sur un algorithme de recherche adaptative introduit par Philippe Codognet. Nous nous intéresserons ici au nombre de solutions des CSP utilisés pour modéliser les analyses musicales

2 Recherche adaptative

L'algorithme de recherche adaptative est proposé par Philippe Codognet (LIP6) [Cod00], qui l'a testé sur des problèmes classiques (N-reines, carrés magiques, etc). Le problème est donné sous la forme d'un CSP, avec variables, domaines finis associés, et contraintes sur les variables. Il existe plusieurs solvers de contraintes dans le cas des domaines finis, comme ILOG Solver [Pug94], ou GNU-Prolog [CDi00]. La recherche adaptative fait partie des algorithmes de recherche locale (voir [HGH98]), tel que GSAT [SLM92], qui tirent profit de la représentation du problème en CSP, même s'ils se distinguent des techniques classiques de résolution. De tels algorithmes ont largement prouvé leur efficacité sur des problèmes comme celui du voyageur de commerce, des N-reines, etc.

Le principe en recherche locale est de guider la recherche de solution par une mesure de la qualité d'une configuration. On peut résumer grossièrement ce type d'algorithme par : intialisation aléatoire, puis itérativement exploration d'un voisinage, recherche d'une meilleure configuration, remplacement. Cela suppose d'avoir une mesure de la qualité de la configuration courante, ce qui est fait en représentant les contraintes par une fonction de coût, qui sert à la recherche d'une meilleure configuration.

L'algorithme de recherche adaptative fonctionne sur ce principe, mais en affinant la notion de coût. Il s'agit de tirer le maximum d'information à partir des contraintes, au niveau de chaque variable et non plus de la configuration globale. On utilise pour cela une projection des coûts sur chaque variable (la

méthode la plus simple consistant à prendre les coûts des contraintes où la variable figure). Cela permet de sélectionner à chaque pas la variable la plus mauvaise. Nous remplaçons l'étape "exploration du voisinage" par le calcul des coûts de chaque variable, la sélection de la plus chère, et l'exploration du domaine de cette variable pour trouver une meilleure valeur.

3 Textures de Ligeti

Est donnée une séquence d'aggrégats A_i ayant chacun au plus k notes. On cherche une polyphonie sous-jacente à k voix, notée dans un tableau $X_{i,j}$, où i désigne l'accord et j la voix. Les contraintes s'écrivent :

1. Chaque A_i est inclus dans la colonne correspondante X_i.
2. Les voix dont les notes n'apparaissent pas dans les A_i doivent rester sur la même hauteur, $X_{i,j} \notin A_i \rightarrow X_{i,j} = X_{i-1,j}$
3. Les mouvements mélodiques de chaque voix sont limités à zéro, plus ou moins un demi-ton ou moins un ton, $X_{i,j} - X_{i-1,j} \in \{-2, -1, 0, 1\}$ (en valeur midi).
4. par ailleurs, les X_i doivent former un vrai accord, on ajoute donc une contrainte imposant la croissance stricte pour chaque X_i

Dans la partition, les aggrégats sont joués en courts motifs répétés, comme dans la pièce pour clavecin *Continuum*. Nous étudions une texture du début de *Melodien*, pièce pour orchestre. Ici, k vaut 10. La figure 1 montre les notes jouées dans les aggrégats en noir, et une analyse de la polyphonie est représentée par les lignes brisées recouvrant ces notes.

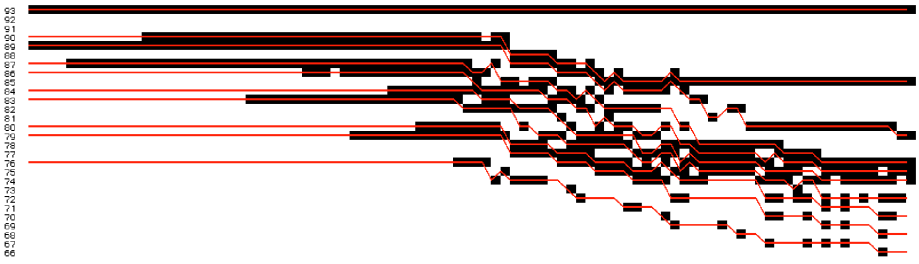

Figure 1: Exemple d'analyse de texture du début de Mélodien

Ce problème rappelle fortement le problème de l'harmonisation automatique, dans le style des chorals de Bach par exemple. La ligne mélodique du choral est remplacée par la séquence d'aggrégats, et la polyphonie à quatre voix par celle à dix voix. Mais les contraintes sont très similaires puisqu'elles formalisent des restrictions du mouvement mélodique. D'un point de vue musical, la principale différence vient du fait que les voix d'un choral sont destinées à être

jouées, tandis qu'ici on cherche des notes cachées : il s'agit d'une structure implicite, sous-jacente à la partition. D'un point de vue contraintes, il faut souligner quelques différences importantes : l'harmonisation automatique est connue pour avoir beaucoup de solutions [Del98], et on peut abstraire certains objets musicaux comme par exemple les degrés harmoniques, ce qui permet de réduire l'espace de recherche [PaR95]. Au contraire, le nombre de solutions est ici très réduit, six dans le cas de *Melodien*.

Pour *Melodien*, on a 49 colonnes et 10 notes par colonne, ces notes restant dans un ambitus de 66 à 93. Nous avons choisi les fonctions de coût suivantes, en reprenant la notation et la numérotation des contraintes ci-dessus, avec i l'indice dans les colonnes et j dans les lignes :

1. $f_1(i, X_{i,j}) = k - Card\{l \in [1,k], X_{i,l} \notin \Lambda_i\}$
2. $f_2(i, X_{i,j}) = \sum_{l<j} max(0, 1 + X_{i,l} - X_{i,j}) + \sum_{l>j} max(0, 1 + X_{i,j} - X_{i,l})$
3. $f_3(i, X_{i,j}) =$ si $X_i \in \Lambda_i$ alors 0 sinon $|X_{i,j} - \Lambda_{(i-1),j}|$
4. la dernière contrainte peut être passée en réduction de domaine

Il y a plusieurs manières de représenter ce problème. La plus simple serait de prendre $49 * 10$ variables et de résoudre un CSP à 500 variables, sur un domaine de taille 27. Pour réduire le nombre de variables, on peut aussi ne prendre comme variables que les notes cachées, 83 dans le cas de *Melodien*, et écrire les contraintes sur ces seules notes. En effet, les autres variables sont presque déjà instanciées dans la partition. Mais ce n'est pas satisfaisant car cela ne permet pas de vérifier si le processus de modélisation est valable pour toute la partition. Nous avons choisi une solution intermédiaire, en résolvant le CSP colonne par colonne : chaque résolution porte donc sur 10 variables, avec des domaines petits (de taille 4 puisqu'on passe à la ligne $i+1$ la contrainte de mouvement mélodique à partir de l'instantiation trouvée à la ligne i, ce qui reviendrait à un filtrage dans une méthode classique). D'un point de vue CSP, on perd de l'information de cette manière, puisque cela revient à interdire les backtracks à une ligne antérieure. En pratique, cela n'a pas d'importance car on trouve très facilement les solutions, sinon nous pourrions fixer un nombre maximum d'itérations pour l'algorithme, décider qu'on a un échec quand il est atteint et backtracker à la ligne précédente. De cette manière, nous pouvons vérifier la validité du modèle musical sur toute la partition. En pratique, le temps de calcul est assez stable, environ 7 secondes sur un Mac G4.

Un algorithme spécifique décrit dans [Che99] montre que l'analyse de cette texture n'a que 6 solutions. Plus précisément, il y a trois transitions possibles de la ligne 21 à la ligne 22 et deux de la ligne 36 à la ligne 37. La figure 2 montre ces deux possibilités, qui donnent en valeur midi, et en notant entre parenthèses les notes cachées :

$$X_{36} = ((67)(70)\ 72\ 73\ 75\ 76\ 77\ 80\ 85\ 93)$$
$$X_{37} = (67\ 70\ 72\ 74(75)\ 76\ 77\ 80\ 85\ 93)$$

$$X_{36} = ((67)(70)\ 72\ 73\ 75\ 76\ 77\ 80\ 85\ 93)$$
$$X_{37} = (67\ 70\ 72(73)74\ 76\ 77\ 80\ 85\ 93)$$

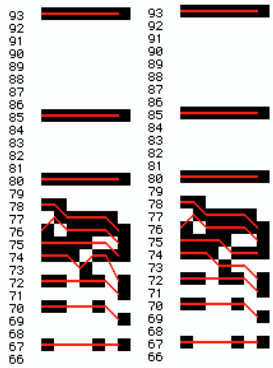

Figure 2: Deux solutions entre les lignes 36 et 37 de Melodien

Les fréquences d'apparition de ces deux solutions, figure 3, dans la résolution par recherche adaptative ne sont pas équiprobables.

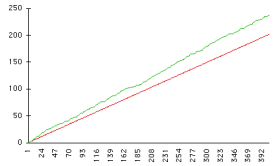

Figure 3: Deux solutions entre les lignes 36 et 37 de Melodien

4 Canons Nzakara

Ce problème est posé dans [Che95]. On s'intéresse aux formules jouées sur une harpe à cinq cordes par le peuple Nzakara en Afrique Centrale. Les formules sont jouées en *ostinato*, des poèmes Nzakara étant chantés sur chacune. Pour la modélisation, on représentera les notes Nzakara par une approximation midi : 60, 62, 64, 67, 70. Les formules sont construites à partir d'accords de deux notes, dont les valeurs sont $\{(60,64),(60,67),(62,67),(62,70),(64,70)\}$. Pour calculer des formules de n couples consécutifs, on prend n variables $V_1 \ldots V_n$ dont les domaines sont les couples Nzakara. On notera $\underline{V_i}$ la note inférieure de l'accord i et $\overline{V_i}$ la note supérieure. La structure des formules est représentée par deux contraintes.

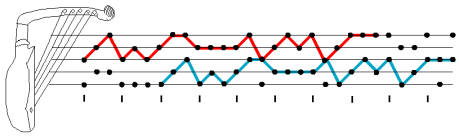

Figure 4: Un canon de la catégorie limanza, pour $n = 30$ et $p = 6$

La première impose une forme en canon à deux voix. La transposition est donnée par une fonction simple : $t(60) = 64$, $t(62) = 67$, $t(64) = 70$. Les formules Nzakara reprennent la voix inférieure la mélodie de la voix supérieure, décalée de p accords (p entier fixé), et transposée par t.

$\underline{V_{(i+p \mod n)}} = t(\overline{V_i})$

La deuxième contrainte sert à éviter les solutions triviales et les répétitions, que les Nzakara ne font jamais.

$V_{(i+1 \mod n)} \neq V_i$

$V_{(i+p \mod n)} \neq V_i$

Les valeurs Nzakara pour n et p sont $n = 10$, $p = 4$ ou $n = 20$, $p = 4$ (catégorie ngbakia) et $n = 30$, $p = 6$ (catégorie limanza, voir figure 4). Ainsi posé, ce CSP n'a pas de solutions. Le nombre minimal d'erreurs est égal à $pgcd(n, p)$ [Che95], et on constate chez les Nzakara que les canons utilisés par les harpistes ont toujours ce nombre minimal d'erreurs.

La première solution pour résoudre ce CSP serait d'utiliser un Prolog, ce que nous avons essayé en Sicstus Prolog. Nous utilisons un prédicat *transpo* pour la fonction de transposition. Comme nous savons qu'il n'y a pas de solutions, il faut transformer un peu la modélisation du CSP : on ajoute ainsi un prédicat $transpo(60, 70)$, qui modélise "l'erreur" faite par les Nzakara. La solution Nzakara est alors trouvée en une demi-heure. Mais l'ajout d'un transpo n'est pas très satisfaisante, car rien ne garantit que le nombre de $transpo(60, 70)$ sera minimal.

Nous avons également résolu ce CSP en recherche adaptative avec de bons résultats. Les coûts sont simplement définis par : pour la première contrainte, 1 si $\underline{V_{(i+p \mod n)}} = t(\overline{V_i})$ et 0 sinon, et pour la deuxième contrainte, 1 de pénalité chaque fois que $V_{(i+1 \mod n)} = V_i$ ou $V_{(i+p \mod n)} = V_i$. D'après [Che95], il n'y a que deux solutions, dans le cas $n = 10$ et $p = 4$, voir figure 5. L'une est dégénérée (elle est formée de deux fois la même sous-séquence). La figure 6 montre les fréquences d'apparition des solutions : sur 500 tests, la solution Nzakara (non-dégénérée) apparaît avec une probabilité de 0, 32, ce qui est cohérent et montre que les fonctions de coûts choisies ne sont pas biaisées.

5 Imparité rythmique

Les pygmées Aka utilisent une formule rythmique qui peut se monnayer par des battements réguliers, groupés par 2 et par 3 en fonction d'accents irrégulièrement

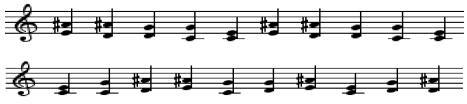

Figure 5: Deux solutions, pour $n = 10$ et $p = 4$

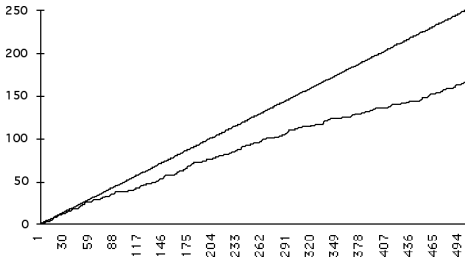

Figure 6: Fréquence d'apparition des deux solutions pour $n = 10$ et $p = 4$.

espacés. Les groupes sont répartis comme suit : 3 2 2 2 2 3 2 2 2 2 2.

Cette formule a une propriété intéressante que Simha Arom a appelée " l'imparité rythmique ". La figure 7 les groupes disposés sur un cercle (la formule est jouée en boucle). Lorsqu'on essaie de couper le cercle en deux, comme une orange, on ne peut le faire en deux parties égales, car quelque soit le point de partage, il manque une unité d'un côté. Il existe une dissymétrie intrinsèque dans la formule, qui se divise toujours en parties inégales, que Simha Arom appelle " moitié moins un " et " moitié plus un ".

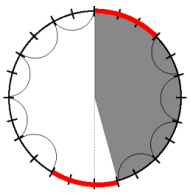

Figure 7: Propriété de l'imparité rythmique.

Dans l'énumération complète des formules rythmiques vérifiant la propriété d'imparité rythmique, il faut noter que si le nombre d'unités sur le cercle est impair, la propriété est triviale. Ce cas correspond à des formules rythmiques ayant un nombre impair de 3. À l'opposé, si les nombres de 2 et de 3 sont tous les deux pairs, on a un résultat surprenant, qui est une sorte de " théorème des valeurs intermédiaires " : il existe nécessairement un point du cercle où celui-ci se coupe en deux partie égales.

En ce qui concerne les formules rythmiques effectivement utilisées en Afrique

centrale, il faut préciser que les rythmes de ce type sont toujours associés à une pulsation régulière dont la périodicité est une puissance de deux. Plus formellement, cela signifie que le nombre total d'unités sur le cercle doit être de la forme $2n$ (rythme binaire) ou $2n.3$ (rythme ternaire).

Avec deux groupes de 3, on vérifie facilement que les seules solutions sont du type 3 $2k$ 3 $2(k+1)$. Les valeurs de k donnant des rythmes binaires ou ternaires sont $k = 0$, $k = 1$, $k = 2$ et $k = 4$ (en se limitant à 24 unités au maximum), soient 8, 12, 16 ou 24 unités. La valeur $k = 3$ est exclue car elle donne 20 unités. La formule des pygmées Aka correspond à $k = 4$.

Avec quatre groupes de 3, la proposition ci-dessus montre qu'il n'y a pas de solutions. Avec six groupes de 3, il existe seulement deux solutions (avec un nombre d'unités inférieur à 24), dont l'une est constituée d'un couple de solutions rétrogrades l'une de l'autre :

solution 1 = 3 3 3 2 3 3 3 2 2

solution 2 = 3 3 3 2 3 3 2 3 2

rétrogradée = 2 3 2 3 3 2 3 3 3

Les travaux de Simha Arom ont montré que les formules vérifiant l'imparité rythmique sont fréquentes dans cette région d'Afrique centrale. Le fait intéressant qui ressort de cette énumération est que l'on trouve presque toutes les formules possibles. Le tableau suivant en dresse l'inventaire (limité à moins de 24 unités sur le cercle), en indiquant les ethnies dans lesquelles elles sont utilisées. En bonne logique, il faudrait expliquer pourquoi la solution 1 avec six groupes de 3 n'est pas utilisée. Peut-être est-elle trop semblable à la solution 2 (elles ne diffèrent que par la permutation de deux valeurs) ?

nombre de 3		
2	3 3 2 ($k = 0$)	Zande
2	3 2 3 2 2 ($k = 1$)	Aka, Gbaya, Nzakara
2	3 2 2 3 2 2 2 ($k = 2$)	Gbaya, Ngbaka
2	3 2 2 2 2 3 2 2 2 2 2 ($k = 4$)	Aka
4	pas de solution	
6	3 3 3 2 3 3 3 2 2	non utilisée
6	3 3 3 2 3 3 2 3 2	Aka

On peut trouver une propriété nécessaire et suffisante à l'imparité rythmique : les groupes de trois doivent être face à face, séparés soit de $n/2 - 1$, soit de $n/2 - 2$. Cela se voit immédiatement sur une dessin, figure 8.

En modélisant le problème sous la forme d'un CSP, on peut tenter d'estimer le nombre de solutions, pour des nombres quelconques de groupes de deux et de trois. On note n_2 le nombre de groupes de deux et n_3 le nombre de groupes de trois, fixés. Les variables sont notées $V_1 \dots V_n$, et l'on se place sur le domaine des permutations de l'ensemble constitué de n_2 deux et n_3 trois (les V_i prennent donc des velurs 2 ou 3). La contrainte est alors très simple à exprimer. Par commodité, on oublie les modulos dans la notation, en supposant que les formules rythmiques sont répétés indéfiniment. $\forall k, k' \leq n, \sum_{i=k}^{i=k+k'-1} V_i \neq \sum_{i=k+k'}^{i=k-1} V_i$

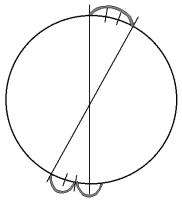

Figure 8: Impossible de n'avoir que des groupes de deux face à un groupe de trois.

La fonction de coût pour V_1 compte simplement 1 si on a une séparation en deux au niveau de V_1, 0 sinon. Pour ce problème, nous avons utilisé la recherche adaptative pour compter le nombre de solutions, ou au moins pour en avoir une approximation selon n_2 et n_3. En notant k le nombre de solutions, et en supposant qu'elles sont toutes équiprobables, cela nous ramène au "Collector's Problem" en probabilité (voir par exemple [Rca98]. Le nombre moyen de tirages pour trouver toutes les solutions est $k * ln(k)$. Plus précisément, la probabilité de trouver toutes les solutions en t tirages est $\frac{(t-1)!}{t^{k-1}*(t-k)!}$. Ainsi pour 30 solutions et 10000 tirages la probabilité d'avoir toutes les solutions est de $0,96$. De sorte qu'en adaptant le nombre de tirages on peut estimer sinon avoir trouvé toutes les solutions, au moins en avoir un bon minorant.

Evidemment, k est inconnu, et la seule chose que l'on peut mesurer est une approximation de k, utilisée pour se valider elle-même. Cela constitue à l'évidence un biais, lié au fait que la distribution des solutions n'est pas nécessairement équiprobable. Il pourrait exister des solutions "rares" qui échapperaient au solver. C'est en particulier le cas des solutions répétitives (analogues à la solution dégénérée du problème Nzakara). Soit n_p le plus petit entier telle qu'une permutation de n_p sur les variables conserve la solution, *ie* $V_{1+n_p} \dots V_{n+n_p} = V_1 \dots V_n$, les solutions répétitives sont celles pour lesquelles $n_p < n$ (on a évidemment toujours n comme majorant de n_p, et même $n_p | n$), par exemple 2 3 3 2 3 3 2 3 3 où $n_p = 3$ et $n = 9$. Ces solutions devraient apparaître n/n_p fois plus souvent. En réalité, le rapport n/n_p reste petit et ces cas rares, nous les considérons négligeables.

Cela repousse le biais sur le solver lui-même. L'experience sur le problème Nzakara ayant montré que le solver n'était pas biaisé, nous conserverons cette hypothèse ici. En réalité, cette expérience a été validée par un programme [2] en OPL (ILOG) qui a permis de confirmer tous les chiffres trouvés avec l'AS.

Expérimentalement, nous avons obtenu les valeurs suivantes (tableau 9, avec n_2 en abscisse et n_3 en ordonnée.

Chaque ligne ou chaque colonne du tableau est une suite numérique (à n_2 ou n_3 constant) dont on s'efforce de trouver la construction. On peut utiliser pour cela le site www.research.att.com/ njas/sequences qui permet de saisir les

[2] Merci à Louis-Martin Rousseau, de l'Université de Montréal

	1	3	5	7	9	11	13	15
2	1	1	1	1	1	1	1	1
4	1	2	3	4	5	6	7	8
6	1	4	7	12	19	26	35	46
8	1	5	14	30	55	91	140	
10	1	7	26	66	143	273		

Figure 9: Nombre de solutions trouvées par la recherche adaptative, en fonction de n_2 (abscisse) et n_3 (ordonnées).

premières valeurs d'une suite, et propose une définition formelle de cette suite tirée de sa base, et se rapprochant au plus près des valeurs saisies. Pour $n_3 = 8$, la ligne correspondante du tableau est surprenante, car elle contient exactement les premières sommes de carrées, ce qui reste pour l'heure complètement inexpliqué.

6 Conclusion

Dans les deux premiers exemples musicaux traités (textures de Ligeti, canons de harpe d'Afrique centrale), on connaissait à l'avance les nombres de solutions, qui pouvaient être calculés par des méthodes spécifiques, et on savait que ce nombre était relativement faible. L'algorithme de recherche adaptative a donné rapidement toutes les solutions. Dans le troisième exemple (rythmes asymétriques d'Afrique centrale), le nombre de solutions n'est pas connu précisément, mais l'algorithme de recherche adaptative a permis d'en obtenir une estimation en faisant tourner le programme suffisamment longtemps pour que la probabilité de laisser des solutions de côté soit très faible.

Dans ces différents cas, on a énuméré certaines combinaisons caractérisées par une propriété formelle, ce qui a montré que les combinaisons vérifiant la propriété étudiée étaient plutôt rares. Sur le plan cognitif, ce constat a une portée particulière en ce qui concerne les exemples musicaux de tradition orale (harpe nzakara, rythmes asymétriques). La plupart des combinaisons possibles étant attestées, on se trouve en présence de ce qu'on pourrait appeler un espace "raréfié" que la tradition orale semble avoir exploré de manière rationnelle, c'est-à-dire en faisant l'inventaire quasi exhaustif de ses éléments. Si ces éléments avaient été choisis au hasard, la probabilité pour qu'ils tombent à l'intérieur de l'espace raréfié aurait été quasi nulle. Il faut donc supposer qu'il existe une sorte d'attirance vers ces configurations rares et improbables. De même que l'algorithme de recherche adaptative semble "attiré" vers les solutions d'un CSP (par le fait que les valeurs sont toujours remplacées par des valeurs de moindre coût), la pensée musicale des artistes de tradition orale semble "attirée" vers les configurations vérifiant certaines propriétés remarquables.

References

[Che95] Marc Chemillier. *La musique de la harpe, Une esthétique perdue.* Editions E. de Dampierre, Presses de l'Ecole Normale Supérieure, 1995.

[Che99] Marc Chemillier. Générateurs musicaux et singularités. *JIM'99*, pages 75 – 85, 2001.

[Cod00] Philippe Codognet. Adaptive search, preliminary results. *BOOK of the 4th ERCIM/CompulogNet Workshop*, 2000.

[CDi00] Philippe Codognet and Daniel Diaz. The implementation of gnu prolog. *SAC'OO 15th*, 2000.

[Del98] P. Dellacherie. Modélisation et optimisation d'un harmoniseur automatique de chorals en plc(fd). *Mémoire de DEA, Université de Caen*, 1998.

[HGH98] Jin-Kao Hao, Philippe Galinier, and Michel Habib. Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Journal of Heuristics*, 1998.

[Mes91] Marcel Mesnage. Sur la modélisation des partitions musicales. *Analyse Musicale*, 1991.

[PaR95] François Pachet and Pierre Roy. Integrating constraint satisfaction techniques with complex object structures. *15th Annual Conference of the British Computer Society Specialist Group on Expert Systems*, pages 11–22, Décembre 1995.

[Pug94] J.-F. Puget. A c++ implementation of clp. *SICIS'94*, 1994.

[Rea98] K. L. Q. Read. A lognormal approximation for the collector's problem. *The American Statistician*, 52(2), 1998.

[RiM88] André Riotte and Marcel Mesnage. Analyse musicale et systèmes formels : un modèle informatique de la 1ère pièce pour quatuor à cordes de stravinsky. *Analyse Musicale*, 1988.

[SLM92] Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. *AAAI'92*, pages 440–446, 1992.