

Chapitre 10

Harmonisation musicale automatique et programmation par contraintes

10.1. De l'harmonisation à la résolution de problèmes

Les informaticiens ont longtemps considéré la musique comme un art particulièrement intéressant. L'histoire de l'informatique musicale remonte au tout début de l'histoire de l'informatique. Des programmes visant à composer de la musique, ou à « faire de la musique » à divers niveaux du processus de composition, ont été conçus depuis les années 1950, certains avec des résultats spectaculaires, comme la Suite Illiac [HILL 93] ou la symphonie de David Cope dans le style de Mozart [COP 95].

On peut remarquer que, au contraire de la musique, les arts visuels, la peinture, la danse ou l'architecture, n'ont pas reçu la même attention de la part des informaticiens. La raison que nous avançons est que, depuis les premières étapes de son évolution, la musique a été l'objet de formalisations. La musique tonale, en particulier, s'est développée dans le cadre d'une structure formelle bien adaptée à la modélisation sur ordinateur et à la résolution de problèmes. Inversement, ces formalisations ont en retour fortement influencé la composition musicale, de l'utilisation rationnelle du tempérament égal, jusqu'aux extrémismes de la musique sérielle. Cet article passe en revue les principales approches et les principaux résultats obtenus dans le domaine de l'*harmonisation automatique*, c'est-à-dire de la

production d'arrangements musicaux (partitions à plusieurs voix) de mélodies données, en examinant notamment les techniques les plus utilisées pour y parvenir, à savoir, les contraintes.

10.1.1. *La musique tonale et les traités d'harmonie*

Dans cet aperçu, nous nous concentrons sur la musique tonale, non par préférence esthétique, mais parce que la musique tonale a été formalisée de façon particulièrement mathématique, par opposition à d'autres courants musicaux.

La musique tonale englobe la majeure partie de la musique occidentale produite et écoutée jusqu'à aujourd'hui : du baroque à la musique classique et romantique, au jazz, au rock et au blues. La musique tonale repose sur la notion de *tonalité*, qui est l'organisation des sons dans des groupes de notes appelés *gammes* (par exemple la gamme de Do majeur comprend les notes Do, Ré, Mi, Fa, Sol, La et Si), eux-mêmes organisés hiérarchiquement dans une œuvre musicale. On considère généralement une certaine tonalité comme la tonalité principale du morceau. On considère alors les notes de la gamme correspondante comme plus importantes que les notes qui n'en font pas partie. L'art de la musique tonale consiste précisément dans l'arrangement de notes de façon à suggérer, préparer, contourner, ou affirmer les centres tonaux. C'est l'organisation des notes dans les tonalités apparentées qui rend cette interaction possible et significative. La musique tonale est généralement opposée – au moins d'un point de vue musicologique – à la « musique atonale », comme *la musique sérielle*, inventée par Schoenberg au début du siècle, dans laquelle les douze notes (par exemple Do, Do#, Ré, Ré#, Mi, Fa, Fa#, Sol, Sol#, La, La#, Si) du système chromatique occidental ont une égale importance. D'autres sortes de musique non tonale existent, comme la musique minimaliste (représentée par des compositeurs comme John Cage ou Steve Reich), la musique ethnique traditionnelle en général (par exemple la musique balinaise ou la musique japonaise), ou la musique dite « modale », bien que l'on puisse la considérer comme une première étape de la musique tonale (par exemple le chant grégorien, ou diverses variétés du rock et du jazz).

De nombreux traités de musique tonale ont été écrits depuis des siècles. Un des traités les plus influents est probablement celui de Johann J. Fux, le *Gradus ad Parnassum*, écrit en 1725 [FUX 65], qui a été le premier à proposer des règles précises de contrepoint, c'est-à-dire l'art de combiner différentes mélodies (on pense que Haydn, Mozart et Beethoven ont étudié la composition au moyen de ce traité). La formalisation concrète de la tonalité, avec en particulier la notion de *basse fondamentale*, a été conçue par Jean-Philippe Rameau et développée dans ses célèbres traités de 1722 [RAM 85]. A l'autre extrême, le traité d'harmonie d'Arnold Schoenberg [SCH 93] est l'un des les plus aboutis de musique tonale (bien qu'il

inclue aussi des descriptions de sa théorie atonale ultérieure). Plus de mille traités d'harmonie ont été écrits, pratiquement tous les compositeurs de musique tonale ayant rédigé leur propre traité. Aujourd'hui, on considère généralement que l'évolution de la musique tonale – au moins d'un point de vue harmonique – est maintenant terminée.

La plupart des traités d'harmonie sont organisés sous la forme d'un recueil de règles indiquant la façon dont les notes peuvent ou ne peuvent être disposées. En particulier, certaines règles spécifient les relations à satisfaire pour des notes simultanées (accords) ou les séquences d'accords. Nous décrivons ces règles au paragraphe suivant.

Plusieurs remarques s'imposent ici. Premièrement, tous les traités d'harmonie ne contiennent pas exactement les mêmes règles. En fait, les règles se sont développées en même temps que la musique. Cependant, on considère qu'il existe un ensemble de règles de base valables au cours de l'évolution de musique tonale et donc toujours présentes, comme la règle des quintes parallèles, illustrée ci-dessous. Deuxièmement, ces règles ne sauraient constituer un algorithme complètement déterminé pour composer de la musique. Ce sont principalement des spécifications de combinaisons interdites de notes, ces combinaisons étant considérées comme dissonantes ou peu agréables à l'oreille. Bien sûr, le compositeur a toujours une grande latitude dans le choix des notes. Quelques compositeurs affirment même – c'est une idée générale dans l'histoire de l'art – que la liberté de la composition vient précisément des contraintes imposées par les règles. En dernier lieu, le respect des règles ne garantit pas que le résultat sera musicalement significatif ou intéressant. Par analogie avec le langage, on peut considérer les règles harmoniques comme purement syntaxiques.

10.1.2. *Les règles d'harmonie*

Nous décrivons ici quelques exemples typiques des règles d'harmonie. Nous prendrons le cas de la musique à quatre voix, c'est-à-dire constituée de quatre mélodies ou *voix* simultanées, traditionnellement interprétées par quatre chanteurs : soprano (la voix la plus aiguë), alto, ténor et basse (la voix la plus grave). Les règles d'harmonie peuvent être classées selon ce à quoi elles s'appliquent dans la partition. Nous n'avons pas l'intention de faire une liste complète de ces règles (ce qui équivaldrait à un traité d'harmonie de plus), mais plutôt de présenter des exemples caractéristiques des règles pour donner une idée de l'ensemble du problème.

Les règles les plus simples énoncent simplement que chaque mélodie a une étendue donnée (tessiture), qui correspond à l'étendue de la voix du chanteur correspondant. Par exemple, le soprano couvre deux octaves du *Do 1* (le Do du

milieu du piano) au *Do 3* (c'est-à-dire vingt-cinq notes). Une autre règle énonce que les voix ne doivent jamais se croiser.

Les règles horizontales s'appliquent aux notes successives d'une voix donnée. Par exemple, on interdit plusieurs intervalles entre les notes, comme l'intervalle de *triton* (figure 10.1) et de façon générale tous les intervalles considérés comme dissonants dans le style de la composition.



Figure 10.1. Les deux notes entourées violent la règle des intervalles horizontaux interdits (ici le triton ascendant entre Mi et Si bémol). Les deux autres intervalles (tierce majeure ascendante et tierce mineure descendante) sont licites

D'autres règles (règles verticales) concernent les notes simultanées ou accords. Par exemple, l'harmonisation telle qu'elle était pratiquée au début de la période baroque n'autorise que trois hauteurs différentes dans chaque accord, et oblige donc à dupliquer une des hauteurs (en général sur une octave différente).

De même, pour certains types d'accord, on interdit la duplication de certaines notes. Par exemple, pour les accords dits de « sixte » (notés « 6 »), la basse ne peut pas être répétée (figure 10.2).

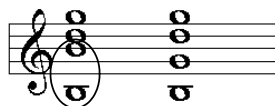


Figure 10.2. Les deux notes entourées du premier accord violent la règle qui interdit la duplication de la basse d'un accord de sixte. L'accord suivant (qui est également un accord de sixte) est licite

Enfin, les règles les plus complexes concernent les séquences d'accords. Une règle typique de cette catégorie est la règle des *quintes parallèles* qui interdit les quintes parallèles entre deux accords successifs. Plus précisément, si deux notes, par exemple la mélodie et la basse, d'un accord forment un intervalle de quinte, la mélodie et la basse de l'accord suivant ne doivent pas former de quinte. Cette règle est illustrée par la figure 10.3. La même règle s'applique aux octaves et aux unissons.

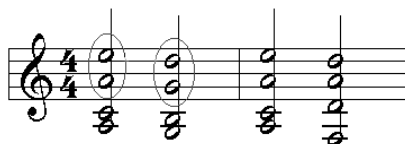


Figure 10.3. *Les deux premiers accords violent la règle des quintes parallèles : les deux intervalles de quinte sont entourés. Les deux derniers accords satisfont la règle des quintes parallèles*

Enfin, un autre élément important de l'harmonisation est ce que l'on appelle le chiffrage des accords. Pour poursuivre l'analogie avec le langage, le chiffrage représente une information sémantique sur l'harmonie. Depuis la musique baroque, diverses sortes de symboles ont été utilisées pour caractériser ou représenter des informations supplémentaires sur les accords dans la partition. Ces symboles (chiffrages) ont été initialement utilisés comme abréviation des accords : au lieu d'écrire les quatre notes réelles composant un accord, on n'indique que la basse et un chiffrage. Le chiffrage permet en principe à l'interprète de reconstruire l'accord d'origine ou un accord similaire. Divers systèmes de chiffrage ont été mis au point au cours de l'évolution de la musique. Il existe aujourd'hui deux grands systèmes. Le premier est le système baroque, dans lequel les chiffrages indiquent les intervalles composant l'accord : par exemple, « 6/3 » signifie que l'accord est composé d'un intervalle de sixte et d'un intervalle de tierce à partir de la note de basse. L'autre système, utilisé pour l'analyse et dans le jazz, est la notation fonctionnelle : les chiffrages indiquent la fonction harmonique de l'accord par rapport à une tonalité donnée, par exemple, l'accord de deuxième degré de la tonalité considérée. Ces systèmes, bien que très différents quant à leur rôle et à leur utilisation, véhiculent tous les deux des informations sémantiques sur l'accord. En pratique, ils donnent des informations supplémentaires permettant de trouver les notes composant l'accord. Dans la plupart des cas, il n'existe qu'un nombre restreint de possibilités pour *comprendre* un accord (c'est-à-dire trouver ses notes réelles) en fonction d'un chiffrage donné.

Comme nous l'avons dit, il n'existe aucun ensemble officiel de règles s'appliquant à tous les styles musicaux. Il y a cependant un consensus implicite quant à certaines règles, que l'on considère comme applicable à une grande variété de styles musicaux (par exemple la règle des quintes parallèles, valable du baroque au classique). D'autres règles sont plus spécifiquement associées à un style (par exemple la limitation des accords à trois hauteurs de notes s'applique typiquement à la musique baroque, alors que la musique des époques suivantes utilise couramment des accords plus complexes). Des ensembles de règles complets sont donnés dans un format naturel dans tous les traités mentionnés ci-dessus. Un jeu de règles plus concis (vingt

règles) est donné dans Tsang et Aitken [TSA 91]. Ballesta [BAL 98] indique une liste de toutes les règles de la musique baroque sous forme de contraintes. Dans tous les cas, il faut souligner que les règles peuvent être exprimées précisément et localement, c'est-à-dire comme des prédicats (généralement des propriétés négatives) portant sur des groupes restreints de notes voisines (verticalement ou horizontalement) et leurs intervalles.

10.1.3. *La composition automatique*

Une des premières applications des règles d'harmonie dans l'informatique musicale a été la composition. Les règles harmoniques y sont compilées dans des automates à états finis. Les transitions correspondent aux suites licites d'un état donné. Si des probabilités sont associées à chaque transition, on obtient un modèle de Markov, dont les premiers utilisateurs ont été Hiller et Isaacson [HILL 93] dans la célèbre *Suite Illiac*, entièrement composée selon ce processus en 1958. Dans cette approche, les règles sont utilisées implicitement comme des contrôles passifs. Le système applique une procédure simpliste de génération et de test, sans effectuer de recherche combinatoire.

10.1.4. *Le problème de la combinatoire*

Le problème de la combinatoire apparaît quand il s'agit de produire des harmonisations avec une voix imposée. Dans ce cas, le problème de l'harmonisation devient naturellement un problème de satisfaction de contraintes (CSP) sur un domaine fini : on peut considérer chaque note inconnue comme une *variable* du CSP dont le domaine est l'ensemble des notes de la gamme de la mélodie. Ce domaine pouvant être restreint aux seules notes appartenant à la tessiture de la voix considérée. On peut alors considérer les règles harmoniques comme des *contraintes* sur ces variables, dans le sens de la satisfaction de contrainte, c'est-à-dire des relations qui doivent être satisfaites dans toutes les solutions (une solution à un CSP est, par définition, une instanciation de toutes les variables du CSP qui satisfait toutes les contraintes du CSP).

Plusieurs variantes de ce problème de base existent dans la pratique des musicologues, selon ce qui est imposé. Parmi les exercices traditionnels des classes de composition figurent les suivants :

- *chant donné* : la voix de soprano seule est connue, résoudre l'exercice consiste à composer les trois voix inférieures (alto, ténor et basse) ;
- *basse non chiffrée* : la basse étant imposée, il faut composer les trois voix supérieures (mélodie, alto et ténor). Dans ce cas, il y a seulement une quantité minimale d'informations donnée au départ. Le problème de la basse non chiffrée a

été longtemps débattu dans la théorie de musique, et divers travaux ont montré qu'il est, en pratique, très sous-déterminé (voir la thèse de doctorat de Rothgeb [ROT 68]). La figure 10.4 montre une solution d'un problème de basse non chiffrée à quatre voix ;

– *basse chiffrée* : la voix de basse est imposée, ainsi que des chiffres (figurant au-dessus de chaque note de la basse) indiquant la nature de chaque accord. Cette information réduit considérablement les accords possibles, et de ce fait, elle réduit la combinatoire du CSP correspondant. Il a correspondu à une pratique des clavecinistes qui devaient jouer – en temps réel – des accords satisfaisant la voix de basse, les chiffres et une mélodie chantée ;

– *problèmes à deux voix* : une mélodie étant imposée, il s'agit de trouver la basse seulement. Dans ce cas, on n'applique qu'un sous-ensemble des règles – ou des règles simplifiées.

Il convient de souligner que les deux premiers problèmes introduisent naturellement un point de vue combinatoire. Dans les deux autres, du fait de l'information supplémentaire qui est fournie, l'aspect combinatoire est moins important. L'aspect intentionnel de la composition et l'aspect combinatoire sont plus ou moins exclusifs. Considérons par exemple la figure 10.3 : une solution correcte (les deux derniers accords) produit une progression tonale tout à fait différente (le dernier accord est un accord de Ré mineur) de la première solution, qui est une progression incorrecte (le deuxième accord est un accord de Sol majeur).

Cependant, le problème strictement combinatoire est intéressant en tant que tel du point de vue des contraintes. La section suivante aborde les principales approches permettant de résoudre ce problème à l'aide de contraintes.

Figure 10.4. Une harmonisation à quatre voix de style baroque. La première voix (soprano) est imposée. Les trois autres voix (2 à 4) doivent être composées pour satisfaire les règles d'harmonie baroque (d'après [ROY 98])

10.2. Harmonisation automatique sous contraintes

Cette section passe en revue les principaux travaux essayant de résoudre un des problèmes de l'harmonisation automatiques, notamment les travaux fondés sur la notion de contraintes.

10.2.1. *Les premières tentatives de modélisation du problème par les contraintes*

Les premiers travaux étaient des tentatives innovatrices utilisant une approche déclarative pour représenter les règles musicales. Aucun algorithme de satisfaction de contraintes n'était utilisé, le souci principal étant de contrôler l'explosion combinatoire en fournissant au résolveur davantage de connaissances.

La première tentative de représentation de règles musicales sous forme de contraintes, c'est-à-dire comme des relations déclaratives portant sur des représentations musicales, est probablement celle de Steels [STE 79]. Dans ces travaux, Steels propose d'utiliser des contraintes pour créer des accords de passage, c'est-à-dire les accords qui peuvent être insérés entre deux accords donnés. Ces accords de passage doivent satisfaire diverses contraintes musicales, par exemple des relations d'intervalle entre les fondamentales de l'accord initial, de l'accord de passage et du dernier accord. Steels utilise essentiellement un système de *frames* associé à une recherche en largeur d'abord. L'explosion combinatoire n'est pas explicitement traitée, à l'exception d'un mécanisme d'évaluation « paresseux » intelligent.

Le même auteur a expérimenté plus tard un système musical plus perfectionné visant à résoudre le cas général du problème de l'harmonisation à quatre voix [STE 86]. Ce système possède la propriété intéressante de combiner l'exploration « brutale » (sans procédure de consistance) et la recherche heuristique. Un de ses buts est en fait d'apprendre automatiquement les heuristiques à partir des solutions.

Courtot [COU 90] décrit un système écrit en Prolog-II et utilisé pour définir des structures musicales et les contraintes qui les relie, afin de créer des polyphonies. Ce système, appelé Clara, vise à représenter les modèles du compositeur, et il est fondé sur un système extensible de types. Bien que la notion de contrainte soit explicitement introduite pour définir des types, l'accent est davantage mis sur l'induction de types que sur l'exploration de l'espace de recherche.

Daniel Levitt décrit dans sa thèse [LEV 81, LEV 93] un langage musical à contraintes servant à définir les propriétés stylistiques des accompagnements harmoniques. Ce langage permet de spécifier incrémentalement des contraintes, en commençant par de simples tessitures de voix, pour arriver au mouvement des fondamentales d'accords. La syntaxe de ce langage permet de spécifier des relations

arithmétiques de base entre des hauteurs et des accords, dont on prouve qu'elles suffisent à décrire des œuvres pour piano dans le style ragtime. Le résolveur utilise apparemment une procédure simple de retour arrière (*backtracking*), et n'a donc qu'une capacité restreinte de limiter l'exploration combinatoire. Par conséquent, il ne convient qu'à des problèmes de petite taille ou faciles, tels que les problèmes sous-contraints.

10.2.2. Utilisation de la satisfaction de contraintes

Ce paragraphe décrit les travaux d'harmonisation automatique qui utilisent la vraie satisfaction de contraintes sous diverses formes.

Schottstaedt [SCHO 89] décrit un système complet d'harmonisation à quatre voix basé sur le traité de Fux [FUX 65]. Schottstaedt fait notamment une analyse détaillée de la théorie de Fux, et en classe les règles selon leur importance, représentée par une pénalité : interdiction (pénalité infinie), violation grave (pénalité de 200), violation bénigne (pénalité de 100) et d'autres règles assorties de pénalités plus faibles. La quinte parallèle est un exemple d'interdit (c'est-à-dire qu'aucune exception à cette règle n'est acceptée). La règle de la tessiture (c'est-à-dire les voix doivent rester dans leur tessiture) fait l'objet d'une violation grave. Le système utilise une stratégie de retour arrière de type « meilleur d'abord ».

Pour éviter de rester coincé dans un local extremum et de réduire le temps de calcul, l'algorithme abandonne la branche actuelle de l'arbre quand il trouve une solution. L'algorithme est donc incomplet, mais il est capable de trouver rapidement de bonnes solutions.

Ebcioğlu [EBC 87] est probablement le premier à avoir réalisé un système capable de produire la musique à quatre voix de haute qualité, de façon entièrement automatique. Son système, qui est le fruit d'un travail considérable, contient divers modules interconnectés traitant différents aspects du processus de composition, dont l'harmonisation, la génération de mélodies, l'analyse de mélodies et la génération de notes de passage. Sur le problème de l'harmonisation lui-même, le système d'Ebcioğlu contient environ 350 règles, représentant le style de Johann Sebastian Bach tel qu'analysé par Ebcioğlu. Ces règles sont représentées dans un langage à contraintes spécifiquement conçu pour cette tâche, appelée BSL. BSL est essentiellement un langage de programmation logique par contraintes mettant en œuvre le saut arrière (*backjumping*), technique que l'auteur considère nécessaire dans la recherche combinatoire spécifique à l'harmonisation musicale.

Le système d'Ebcioğlu est intéressant parce qu'il est le premier système capable de résoudre un problème musical difficile et parce qu'il produit des résultats de haute qualité. Cependant, il est difficile à comprendre et à évaluer, car il est réalisé dans un langage spécial et contient de très nombreux modules interdépendants. Dellacherie [DEL 98] a essayé de clarifier ce système du point de vue du traitement de contraintes en extrayant son système de règles et en le réécrivant en Eclipse ; les résultats ont été médiocres, en raison du manque d'expressivité d'Eclipse (c'est un langage à contraintes qui ne traite pas complètement la cohérence d'arcs, inventé par Mackworth [MAC 77], dans des domaines finis). Comme l'a montré Ovans, le traitement de la cohérence d'arcs est nécessaire dans l'harmonisation musicale ; la cohérence partielle (*bound consistency*) n'est pas suffisante pour obtenir des performances acceptables.

Ovans et Davidson [OVA 92b] (voir aussi [OVA 92a]) ont été les premiers à souligner l'aspect combinatoire profond de l'harmonisation entièrement automatique et à préconiser l'utilisation de la cohérence d'arcs. Ils ont réalisé un système destiné à résoudre l'harmonisation à deux voix selon les règles de Fux, et ont comparé sur le même problème, plusieurs techniques de résolution, comme le retour arrière classique, la vérification avant (*forward checking*) et la cohérence d'arcs. Bien qu'aucune méthode ne semble être toujours meilleure que les autres, Owens souligne que la consistance d'arcs devrait toujours être utilisée pour résoudre les problèmes d'harmonisation, et critique sur ce point le système d'Ebcioğlu, qui aurait été, selon lui, nettement plus efficace s'il avait eu recours à la consistance d'arcs plutôt qu'au saut arrière. Il est intéressant de noter que l'on considère aujourd'hui ces conclusions comme valides dans la programmation par contraintes en général. Cependant, le système d'Ovans ne produisant que des harmonies à deux voix, ses résultats ne sont pas entièrement convaincants.

Le premier système d'harmonisation automatique à quatre voix, à l'aide de techniques de programmation logique avec contraintes, est décrit dans Tsang et Aitken [TSA 91]. Ce système effectue une harmonisation à quatre voix à l'aide d'un jeu de vingt règles ; il est mis en œuvre dans CLP(R) [JAF 87]. Ces auteurs utilisent une représentation directe des objets musicaux (hauteurs, intervalles et types d'accords). Le résultat est un système très coûteux en temps et en volume (70 mégaoctets de mémoire pour harmoniser une mélodie à onze notes). Cependant, du fait de la simplicité du jeu de règles, il est souvent utilisé comme terme de comparaison.

Une étude très détaillée de l'harmonisation à quatre voix utilisant des techniques de satisfaction de contrainte conventionnelles est la thèse de doctorat de Philippe Ballesta [BAL 98]. Ballesta a transcrit un jeu complet de règles d'harmonisation à l'aide du système *PECOS* d'Ilog, ancêtre d'Ilog Solver. Le problème ainsi résolu est celui de la *basse chiffrée*, et ce travail inclut des descriptions détaillées de la

représentation des objets musicaux et des contraintes. Les solutions produites par ce système sont généralement considérées comme relativement bonnes du point de vue musical. La performance du système, cependant, n'est pas aussi satisfaisante, en particulier parce qu'il utilise des contraintes pour représenter toutes les relations musicales, au-delà des règles harmoniques. Par exemple, la relation entre un accord et sa note de basse est représentée comme une contrainte, au même titre que la relation entre deux notes et l'intervalle entre elles, ou bien encore les relations entre la hauteur d'une note et son nom dans une gamme donnée. Ce choix radical produit un système élégant et expressif, mais limité dans sa capacité de manipulation de mélodies très complexes ou très longues.

10.2.3. Structuration du problème

Les diverses tentatives de résolution du problème de l'harmonisation avec les techniques de contraintes *stricto sensu* décrites ci-dessus ont montré que les contraintes sont un paradigme utile, mais qu'elles sont toujours limitées et, dans l'ensemble, incapables de traiter des mélodies réalistes en un temps raisonnable. D'autres approches ont été utilisées pour essayer d'améliorer ces travaux, notamment en cherchant les meilleures représentations possibles du problème. Nous décrivons ici deux de ces approches.

La première consiste dans l'exploitation plus active des structures d'accords. Dans les approches précédentes, les accords étaient toujours traités comme des groupes simples de notes (ce qu'ils sont, d'ailleurs, en un certain sens). Cependant, du point de vue des contraintes, c'est une erreur. En effet, certaines règles harmoniques importantes (comme la règle des quintes parallèles décrite au paragraphe 10.1.2) manipulent explicitement des accords et non des notes. Bien sûr, il est possible de représenter ces règles comme des contraintes entre notes. C'est notamment l'approche suivie par Ovans, Ballesta ou Tsang et Aitken.

Nous avons montré dans [PAC 95] qu'en ajoutant à la définition du CSP des variables représentant les accords (en plus des variables représentant les notes de base), on pourrait réduire considérablement la complexité théorique du problème. Plus précisément, la complexité peut être réduite d'un facteur équivalent à la densité d'accords dans l'espace des quadruplets de notes. *A priori*, l'introduction de variables (au sens des CSP à domaines finis) représentant les accords pose un problème combinatoire élémentaire : le domaine de telles variables doit contenir tous les accords de quatre notes possibles dans une tonalité donnée, et ces accords sont en bien trop grand nombre pour que le problème soit possible à résoudre de façon efficace (la taille des domaines des variables est un paramètre essentiel de la combinatoire d'un CSP). On contourne cette difficulté en décomposant le processus de satisfaction de contraintes en plusieurs phases.

Dans un premier temps, on ne considère que les variables représentant les notes et les contraintes (ou règles musicales) entre ces « variables notes », de façon à réduire l'ensemble des valeurs de notes possibles pour chaque « variable note » du problème. A l'issue de cette première phase, les domaines des variables notes sont sensiblement réduits, ce qui permet de définir des variables représentant les accords dont les domaines sont de taille « raisonnable » (techniquement, le domaine d'une « variable accord » est équivalent au produit cartésien des domaines des quatre « variables notes » le composant. De ce fait, plus ces variables notes ont des petits domaines, plus le nombre d'accords que l'on peut construire est petit) ; enfin, une dernière phase consiste à créer et résoudre le CSP contenant les « variables accords » et les contraintes d'accords (c'est-à-dire, les contraintes correspondant à des règles musicales entre accords, comme la règle des quintes parallèles déjà présentée à plusieurs reprises dans cet article). Les solutions de ce problème sont les harmonisations recherchées.

En pratique, cette approche permet d'obtenir quasiment en temps réel les solutions d'exercices à quatre voix (avec ou sans chiffrage). Les détails de la conception et de la mise en œuvre sont fournis dans [ROY 98]. Soulignons que ces résultats sont obtenus sans ajout de connaissances au système.

Une autre approche a été proposée pour réduire la complexité du problème d'harmonisation [HEN 96, ZIM 01]. Elle consiste à construire, avant la résolution, un plan d'harmonisation contenant les informations sur l'harmonie à produire. Ce plan représente l'intention harmonique du morceau, qui avait disparu lors de l'introduction du problème combinatoire (voir paragraphe 10.1.4). Le plan contient des informations de chiffrage comme le degré (par exemple I ou II ; voir paragraphe 10.1.2). Cette information réduit considérablement le nombre de notes qu'il est possible d'utiliser, puisqu'il détermine les classes de hauteur de chaque accord (par exemple Do, Mi, Sol) et ne laisse indéterminées que les octaves réelles de ces classes de hauteur (par exemple. Do₀, Mi₂, Sol₂, Do₂). Une fois que le plan harmonique est produit, un système de contraintes directes (écrit en Oz) calcule les notes réelles (ce que l'on appelle la « réalisation » des accords). Ce second problème est beaucoup plus simple que le problème de l'harmonisation d'un chant donné (donc sans chiffrage) à quatre voix.

10.2.4. *Éléments algorithmiques*

Il serait long et fastidieux de décrire en détail l'algorithme complet d'un système de résolution de contraintes pour l'harmonie. Néanmoins, on peut facilement esquisser à grands traits son fonctionnement.

L'algorithme de résolution de la plupart des solveurs de contraintes comme le système BackTalk est fondé sur le principe de la *cohérence d'arcs*, définie par Mackworth [MAC 77]. Une contrainte C , portant sur deux variables X et Y , est dite arc cohérente si pour toute valeur x du domaine de X , il existe au moins une valeur y dans le domaine de Y telle que $C(x,y)$ (c'est-à-dire, telle que le couple de valeur (x,y) satisfasse la contrainte C). Cette définition se généralise de façon évidente aux contraintes portant sur plus de deux variables.

Une remarque importante est que l'on peut toujours rendre arc cohérente une contrainte en supprimant des valeurs dans les domaines des variables impliquées. Pour s'en persuader, il suffit de penser que si l'on supprime toutes les valeurs des domaines des variables, la contrainte est alors trivialement arc cohérente.

Une propriété fondamentale des CSP est que l'on peut toujours rendre une contrainte arc cohérente en réduisant les domaines des variables *sans modifier l'espace des solutions* du CSP. On peut ainsi utiliser la cohérence d'arcs pour réduire la taille du CSP, et donc sa combinatoire, tout en ne supprimant aucune solution.

Voici un algorithme simple pour le faire : soient deux variables X et Y de domaines $\text{dom}(X)$ et $\text{dom}(Y)$, et C une contrainte portant sur X et Y :

Algorithme A :

```

pour tout x dans dom(X)
  si pour tout y dans dom(Y) on a non(C(x,y))
    alors supprimer la valeur x de dom(X)
pour tout y dans dom(Y)
  si pour tout x dans dom(X) on a non(C(x,y))
    alors supprimer la valeur y de dom(Y)

```

Il est aisé de vérifier que l'application de cet algorithme, à une contrainte quelconque du CSP, ne modifie aucunement l'espace des solutions, puisque les valeurs que l'on supprime des domaines ne peuvent faire parties d'aucune solution.

L'algorithme sur lequel s'appuient les solveurs de contraintes est le suivant : soit P un CSP, nous notons $\text{dom}(V)$ le domaine d'une variable V :

Algorithme B :

```

| tant qu'il existe C non arc cohérente P
| rendre C arc cohérente en utilisant l'algorithme A

```

Dans BackTalk, l'algorithme B est inclus dans une boucle d'exploration arborescente dont voici une description simplifiée.

Mécanisme général de résolution des CSP dans BackTalk :

```

1. tant qu'il existe dans P une variable non instanciée
2.     soit v une telle variable
3.         choisir x dans dom(v)
4.             empiler l'état courant de P
5.             instancier v avec la valeur x
6.             rendre P arc cohérente avec
Algorithm B
7.         si
8.             il existe v tel que dom(v)={}
9.         alors
10.            dépiler l'état précédent de P
11.            supprimer x de dom(v)
12. si toutes les variables sont instanciées
13.     retourner la solution
14. sinon
15.     il n'existe pas de solution

```

La ligne 5 consiste à explorer une nouvelle branche de l'arbre de recherche. Les lignes 10 et 11 consistent à abandonner l'exploration de la branche choisie en ligne 5, et à continuer dans le reste de l'arbre de recherche. A chaque nouvelle branche, on applique l'algorithme B afin de réduire la taille du problème sans changer l'espace des solutions.

10.2.5. Les techniques de recherche locale

Une autre approche de la satisfaction de contraintes consiste à exploiter des algorithmes incomplets, mais qui sont plus adaptés aux problèmes pour lesquels une solution approchée suffit, et qui ont relativement une grande densité de solutions. En composition notamment, des solutions approchées sont souvent aussi satisfaisantes que d'hypothétiques solutions exactes. Le système OMClouds de Charlotte Truchet [TRU 03], est une bibliothèque de contraintes intégrée au langage visuel OpenMusic (voir le chapitre sur la programmation visuelle pour la composition, même volume). OMClouds s'appuie sur un algorithme incomplet de recherche locale qui fournit très rapidement des solutions approchées, c'est-à-dire violant aussi peu de contraintes que possible.

Techniquement, ces algorithmes incomplets sont basés sur un mécanisme de coûts : chaque contrainte est associée à un coût en cas de violation, ce qui permet de pondérer les différentes contraintes du problème selon leur importance. En outre, OMClouds reprend la philosophie du système *Situation* concernant les variables : elles ne représentent pas directement des objets musicaux, mais des nombres entiers, l'association entre ces nombres et les objets musicaux est de la responsabilité des utilisateurs. L'intégration de OMClouds dans OpenMusic en fait un outil destiné aux compositeurs. OMClouds a été utilisé, par exemple, pour classer une suite

d'accords de façon à maximiser le nombre de notes communes entre deux accords successifs ; ou encore pour résoudre le problème d'harmonisation automatique de canons rythmiques dans *Empreinte sonore de la fondation Bayeler* de Georges Bloch.

Le OMClouds repose sur l'algorithme suivant, dû à Codognet et Diaz [COD 01] : on parcourt l'espace de recherche en se guidant par une mesure de la qualité de l'instanciation courante des variables, et en évitant de revenir trop souvent aux mêmes instanciations. Cela suppose une mesure de la qualité des instanciations, qui est donnée par une fonction de coût sur les contraintes.

L'algorithme utilise ensuite les projections des coûts sur chaque variable du problème afin de trouver la variable la « plus coûteuse » et d'essayer d'améliorer l'instanciation courante en changeant la valeur de cette variable. Voici l'algorithme :

E est le seuil d'arrêt de l'algorithme, normalement fixé à 0, Ltabu est la longueur de la liste tabou, et V+ la plus mauvaise variable pour une itération donnée :

```

Initialisation aléatoire des variables
Répéter
  1.Calcul des coûts de toutes les variables sauf les
  variables marquées tabou, sélection de la plus chère V+
  2.Test du coût global en remplaçant V+ par toutes les
  valeurs de son domaine, sélection de la valeur la plus
  avantageuse v'
  3.Si à la fin de l'exploration, aucune valeur
  n'améliore le coût global, alors V+ est marquée tabou
  pendant Ltabu itérations, sinon, V+ est instanciée à v'
  4.Si toutes les variables sont tabou, alors
  réinitialisation aléatoire
Jusqu'à avoir une erreur globale inférieure à E

```

Un des points remarquables de cet algorithme est qu'il est très rapide, en comparaison des algorithmes complets. Par ailleurs, au contraire des algorithmes de recherche complets, il peut être arrêté à tout instant, pour récupérer la meilleure solution trouvée jusqu'alors.

Notons enfin qu'il existe d'autres applications de cet algorithme à la musique, par exemple pour le séquençage temporel de samples audio, avec la technique du *musicing* introduite par Zils et Pachet [ZIL 01], ou bien encore la génération de playlists pour les systèmes de gestion de contenu musicaux : voir [PAC 00] pour des techniques complètes, et [AUC 02] pour des techniques à base de recherche locales, similaires à celle écrite ici.

10.2.6. *Autres approches*

Une série de solveurs de contraintes conçus pour des problèmes musicaux a été réalisée à l'IRCAM et utilisée par les compositeurs de musique contemporaine. L'un des plus récents, *Situation*, permet de spécifier des polyphonies à l'aide d'un jeu de contraintes spécifiques intégrées [RUE 98]. Il n'est pas destiné à des exercices de résolution de musique tonale, mais est plutôt conçu comme outil concret de composition. Une caractéristique intéressante du moteur est sa capacité à énoncer les contraintes de façon abstraite : elles peuvent ainsi porter sur des « points » (par exemple des notes, des accords ou des durées) et sur des « distances » (par exemple des intervalles de hauteur, des intervalles de temps, des intervalles d'accords, etc.). Cette couche abstraite permet d'utiliser le solveur dans un grand nombre de situations, et pas uniquement dans le domaine des hauteurs, car l'utilisateur peut choisir la façon d'associer ces « points » et ces « distances » aux entités musicales.

D'un point de vue technique, le moteur de *Situation* est basé sur un mécanisme de vérification *a priori* (*forward checking*) assorti d'une évaluation paresseuse [RUE 97] et utilise une représentation unidimensionnelle du problème (c'est-à-dire que des variables représentent des notes ou des attributs de notes). Des expériences menées pour définir avec *Situation* des contraintes correspondant aux règles de l'harmonie tonale [GUE 98] ont fait apparaître les mêmes limitations que dans les travaux de Ballesta. Cependant, les contraintes intégrées permettent de trouver des solutions rapidement et sont particulièrement bien adaptées aux besoins de compositeurs de musique contemporains.

Le travail de Ramirez et Peralta [RAMI 98], bien que non directement lié à l'harmonisation à quatre voix, mérite d'être mentionné ici : ils abordent la question de la production d'une séquence d'accords (en réalité une séquence de noms d'accords) qui harmonise de façon satisfaisante une mélodie donnée. Ils se limitent à des accords de trois notes. L'aspect intéressant de ces travaux est qu'ils essaient de maximiser le nombre de suites d'accords familières, qui sont stockées dans un dictionnaire. Un algorithme simple, qui n'est pas fondé sur les techniques de cohérence d'arcs, permet de résoudre le problème. Dans ce cas, la véritable satisfaction de contraintes n'est pas nécessaire, puisque la recherche combinatoire est très réduite par rapport au problème de l'harmonisation à quatre voix. Cependant, le système résout un véritable problème d'harmonisation musicale.

Enfin, Phon-Amnuaisuk et Wiggins [PHO 98] proposent une comparaison entre une approche à base de règle (représentation de la connaissance explicitement symbolique) et des approches fondées sur des algorithmes génétiques ; ils concluent à la supériorité de la première approche du point de vue de la qualité des solutions obtenues. Bien sûr, il reste difficile de fournir explicitement au système les règles d'harmonie. Des travaux récents ont abordé cette question en essayant d'induire

automatiquement des jeux de règle ou des grammaires à partir de divers recueils de partitions à l'aide de techniques statistiques [PON 99].

10.2.7. *Les systèmes existants*

Il n'existe pas, à notre connaissance, de logiciel commercial d'harmonisation fondé sur des techniques de contraintes, sans doute parce que bien que les problèmes d'harmonisation (notamment dans la musique tonale) soient des exercices intéressants, ils n'ont que des applications limitées. En pratique, de nombreux synthétiseurs proposent des fonctions d'harmonisation de base (comme des accords obtenus à l'aide d'une seule touche sur certains claviers électroniques). Ces harmonisations sont généralement exécutées de façon réactive sur la base d'une note associée à un accord, c'est-à-dire sans tenir compte du contexte (les notes précédentes de la mélodie). Dans ces systèmes, les exigences du temps réel interdisent de toute façon l'utilisation du retour arrière !

Le système *Tonica* (distribué par Software Partners) est un excellent système d'harmonisation à base de techniques connexionnistes. Il harmonise des mélodies qu'on lui donne, et possède une interface très flexible. Le système procède en deux phases. Il produit d'abord une analyse de la mélodie en termes d'accords (l'équivalent des chiffrages), puis une harmonisation réelle de la mélodie et la description précise des accords. Comme de nombreuses analyses harmoniques sont possibles pour une mélodie donnée, le système n'effectue pas de recherche profonde. L'utilisateur a la responsabilité de choisir une analyse qui convient à son intention. En pratique, cette approche est beaucoup plus intuitive pour des musiciens, et le système a donc une utilité certaine. Toutefois, il ne propose pas de nouvelle solution au problème de l'harmonisation du point de vue de la résolution de problèmes. Par ailleurs, plusieurs approches ont été proposées pour la résolution de problème d'harmonisation à l'aide de techniques d'apprentissage comme les réseaux connexionnistes, par exemple dans [HIL 92], avec d'excellents résultats.

10.3. Conclusion : le problème est-il résolu ?

On peut maintenant considérer le problème technique de l'harmonisation à quatre voix comme résolu grâce aux techniques de satisfaction de contraintes utilisant la cohérence d'arcs associée à une structuration adéquate du sous-problème de la manipulation correcte des variables d'accord. On en est arrivé là après plusieurs années d'essais et d'erreurs, en commençant par des approches « brutales » (par exemple, celle de Schottsdaedt) pour passer à des langages à contraintes spécifiques (Ebcioğlu) et à techniques de consistance d'arcs (Ovans), assorties d'une bonne structuration du problème (Pachet et Roy).

Cependant, une partie non résolue est le sous-problème de la production de mélodies musicalement agréables ou intéressantes. Divers aspects de ce qui rend les mélodies intéressantes demeurent non résolus.

Tout d'abord, on ne connaît pas de règles rendant compte de ce qui fait qu'une mélodie est intéressante, pour autant que de telles règles existent. Comme nous l'avons vu, la formalisation de la musique a surtout porté sur l'établissement de « règles syntaxiques ». Plusieurs tentatives ont été faites pour formuler les propriétés explicites des mélodies qui semblent « agréables » ou « équilibrées » (voir par exemple les approches statistiques et les travaux sur la propriété « $1/f$ » en musique [VOS 78]), mais leurs formulations n'ont pas atteint un niveau permettant de les convertir en spécifications opérationnelles.

Deuxièmement, l'intérêt musical résulte probablement de l'interaction entre des critères incompatibles. Par exemple, il est bien connu que des mouvements conjoints (c'est-à-dire des voix allant dans la même direction, vers le haut ou vers le bas) semblent agréables, mais cela va à l'encontre des diverses règles opposées aux mouvements parallèles (quintes ou octaves parallèles). Comment ces critères coexistent-ils exactement ? Les travaux de Ballesta décrits ci-dessus comportent des études des critères d'optimisation choisis parmi les diverses solutions. Ballesta propose plusieurs critères pour décrire des solutions mélodiques et les met en œuvre à l'aide de l'algorithme de séparation-évaluation (*branch and bound*) du solveur. Par exemple, une solution agréable peut être décrite par les propriétés suivantes :

- la voix de soprano ne doit pas changer trop souvent de direction ;
- le soprano et la basse doivent être autant que possible en mouvement contraire ;
- la voix de soprano doit se déplacer autant que possible par intervalles conjoints (notes voisines) ;
- les voix intermédiaires doivent être maintenues aussi proches que possible .

Ce problème d'optimisation multicritère est très difficile et aucune expérience systématique n'a encore été faite sur des mélodies réalistes, mais la question mérite certainement davantage d'attention.

Enfin, les mélodies agréables sont des mélodies qui contiennent des modèles familiers et des suites harmoniques ou des « solutions » harmoniques typiques. Mais dans le contexte de systèmes d'harmonisation automatique, il est difficile de se conformer à cette exigence. En effet, nous sommes loin d'avoir élaboré un catalogue de formules harmoniques familières, *a fortiori* de pouvoir favoriser l'utilisation de telles formules dans les harmonisations produites. Les travaux de Ramirez et Peralta constituent un premier effort dans cette direction.

L'harmonisation automatique a depuis longtemps soulevé des questions difficiles pour le domaine de la programmation par contraintes et a incité les chercheurs à trouver des solutions à ces questions, élargissant ainsi notre connaissance des structures musicales tout comme celle des problèmes de satisfaction de contraintes en général. Comme nous avons essayé de le montrer, ce problème – ou plutôt cette catégorie de problèmes – continuera à apporter des questions stimulantes à la recherche sur les contraintes.

10.4. Bibliographie

- [AUC 02] AUCOUTURIER J.J., PACHET F., « Scaling up Music Playlist Generation », *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, Lausanne, Suisse, août 2002.
- [BAL 98] BALLESTA P., « Contraintes et objets : clefs de voûte d'un outil d'aide à la composition », dans M. Chemillier et F. Pachet (dir.), *Informatique Musicale, Recherche et Applications*, Hermès, Paris, 1998.
- [BRO 94] BROWN F., MOUTON R., « L'automate musical en temps réel Kantor », *Premières Journées d'Informatique Musicale*, Labri, Bordeaux, 1994.
- [COD 01] CODOGNET P., DIAZ D., « Yet another local search method for constraint solving, LNCS 2246 », *First Symposium on Stochastic Algorithms: Foundations and Applications, SAGA 2001*, Berlin, Allemagne, p. 73-90, 13-14 décembre 2001.
- [COP 87] COPE D., « An Expert System for Computer-Assisted Music Composition, » *Computer Music Journal*, 11 (4), p. 30-46, 1987.
- [COU 90] COURTOT F., « A Constraint Based Logic Program for Generating Polyphonies », *International Computer Music Conference*, Glasgow, 1990. Voir également dans Balaban M. et al. (dir.), *Understanding Music with AI*, MIT Press/AAAI Press, Cambridge, p. 156-181, 1992.
- [DEL 98] DELLACHERIE P., Modélisation et optimisation d'un harmoniseur automatique de chorals en PLC (FD), Rapport de DEA, Université de Caen, 1998.
- [EBC 87] EBCIOGLU K., Report on the Choral Project: An Expert System for Harmonizing Four-Part Chorales, IBM Rapport technique RC 12628, 1987.
- [EBC 93] EBCIOGLU K., « An Expert System for Harmonizing Four-Part Chorales », dans S.M. Schwanauer et D.A. Levitt (dir.), *Machine Models of Music*, MIT Press, Cambridge, p. 385-401, 1993.
- [FUX 65] FUX J.J., *The Study of Counterpoint from Johann Joseph Fux's Gradus ad Parnassum*, W.W. Norton & Company, New York, 1965.
- [GUE 98] GUÉNÉGO J.-L., Une application de résolution de contraintes en harmonie tonale, IRCAM/Université Paris VI, <http://www.ircam.fr/equipes/repmus/Rapports/JLGuenego98>, 1998.

- [HEN 96] Henz M., *et al.*, « CompoZe- Intention-Based Music Composition Through Constraint Programming », *8th IEEE International Conference on Tools with AI*, Toulouse, France, 1996.
- [HIL 92] HILD H., FEULNER J., MENZEL W., « HARMONET: A Neural Net for Harmonizing Chorals in the Style of J.S. Bach », dans R.P. Lippmann, J.E. Moody, D.S. Touretzky (dir.), *Advances in Neural Information Processing 4 (NIPS 4)*, p. 267-274, Morgan Kaufmann, 1992.
- [HILL 93] HILLER L., ISAACSON L., « Musical Composition with a High-Speed Digital Computer », dans M. Schwanauer et D. Levitt (dir.), *Machines Models of Music*, MIT Press, Cambridge, p. 9-21, 1993.
- [JAF 87] JAFFAR J., LASSEZ J.L., « Constraint Logic Programming », *IEEE 4th International Conference on Logic Programming*, Melbourne, 25-29 mai 1987.
- [LEV 93] LEVITT D.A., « A Representation of Musical Dialects », dans S.M. Schwanauer et D.A. Levitt (dir.), *Machine Models of Music*, MIT Press, Cambridge, p. 456-469, 1993.
- [LEV 81] LEVITT D., « A Melody Description System for Jazz Improvisation », Masters Thesis, MIT Artificial Intelligence Lab, Cambridge, 1981.
- [MAC 77] MACKWORTH A.K., « Consistency in networks of relations », *Artificial Intelligence*, vol. 8 (1), p. 99-118, 1977.
- [OVA 92a] OVANS R., Efficient Music Composition via Consistency techniques, Rapport technique CSS-IS TR 92-02, Université Simon Fraser, Canada, 1992.
- [OVA 92b] OVANS R., DAVISON R., An Interactive Constraint-Based Expert Assistant for Music Composition. Ninth Canadian Conference on Artificial Intelligence, Université British Columbia, Vancouver, p. 76-81, 1992.
- [PAC 95a] PACHET F., ROY P., « Integrating constraint satisfaction techniques with complex object structures », *15th Annual Conference of the British Computer Society Specialist Group on Expert Systems, ES'95*, Cambridge, p. 11-22, 1995.
- [PAC 95b] PACHET F., ROY P., « Mixing constraints and objects: a case study in automatic harmonization », *TOOLS Europe '95*, Prentice-Hall, p. 119-126, 1995.
- [PAC 00] PACHET F., ROY P., CAZALY D., « A Combinatorial Approach to Content-Based Music Selection », *IEEE Multimedia*, 7(1):44-51, mars 2000.
- [PHO 98] PHON-AMNUAISUK S., WIGGINS G., « The Four-Part harmonization problem: A comparison between genetic algorithms and a Rule-Based system », *Proceedings of the AISB '99 Symposium on Musical Creativity, AISB Symposium*, Edinbourg, 1998.
- [PON 99] PONSFORD D., WIGGINS G., MELLISH C., « Statistical Learning of Harmonic Movement », *Journal of New Music Research*, 28:2, p. 150-177, 1999.
- [RAM 85] RAMEAU J.-P., *Treatise on Harmony*, Dover Publications, New York, 1985.
- [RAMI 98] RAMIREZ R., PERALTA J., « A constraint-based melody harmonizer », *ECAI'98 Workshop on Constraints and Artistic Applications*, Brighton, 1998.

- [ROT 68] ROTHGEB J., *Harmonizing the Unfigured Bass: a Computational Study*, Thèse de doctorat, Université d'Indiana, 1968.
- [ROY 98] ROY P., *Satisfaction de contraintes et programmation par objets*, Thèse de doctorat, Université Paris 6, 1998.
- [RUE 97] RUEDA C., VALENCIA F., « Improving forward-checking with delayed evaluation », *Proceedings of CLEI'97*, Santiago, Chili, 1997.
- [RUE 98] RUEDA C., LINDBERG M., LAURSON M., BLOCH G., ASSAYAG G., « Integrating Constraint Programming in Visual Musical Composition Languages », *ECAI Workshop on Constraints for Artistic Applications*, Brighton, 1998.
- [SCH 93] SCHOENBERG A., *Theory of Harmony*, Université de California Press, 1993.
- [SCHO 89] SCHOTTSTAEDT B., « Automatic species counterpoint. Tech. Rep. STAN-M-19, Stanford University CCRMA. A short report with source code appeared », dans M.V. Mathews et J.R. Pierce (dir.), *Current Directions in Computer Music Research*, MIT Press, Cambridge, 1989.
- [STE 79] STEELS L., *Reasoning modeled as a Society of Communicating Experts*, AI-TR-542. Artificial Intelligence Lab, MIT, Cambridge, 1979.
- [STE 86] STEELS L., *Learning the craft of musical composition*, ICMC, The Hague, 1986
- [TRU 03] TRUCHET C., ASSAYAG G., CODOGNET P., « OMClouds, a Heuristic Solver for Musical Constraints », *MIC03, Metaheuristics International Conference*, Kyoto, Japon, août 2003.
- [TSA 91] TSANG C.P., AITKEN M., *Harmonizing music as a discipline of constraint logic programming*, ICMC, Montréal, 1991.
- [VOS 78] VOSS R., CLARKE J., « 1/f noise in Music », *Journal of the Acoustical Society of America*, 63, p. 258-263, 1978.
- [WIG 98] WIGGINS G., « The Use of Constraint Systems for Musical Composition », *ECAI'98 Workshop on Constraints and Artistic Applications*, Brighton, 1998.
- [ZIL 01] ZILS A., PACHET F., « Musical Mosaicing », *Proceedings of DAFX 01*, Université de Limerick, 2001.
- [ZIM 01] ZIMERMANN D., « Modelling Musical Structures », *Constraints Journal*, Kluwer Publisher, 6(1):7-19, 2001.