Abstract

Constraint Programming (CP) allows to modelize and solve combinatory problems by specifying some partial information on variables, unknowns of the problem. In this thesis, we have studied musical constraint problems, either stated by contemporary composers, or of musical analysis, or instrumentation, in collaboration with IRCAM (french Institute for Research and Coordination Acoustics / Music). Fourteen such problems have been modelized and solved, which allowed to give a detailed typology. This has been used to conceive and implement OMClouds, a library in the Computer Assisted Composition environment OpenMusic. It is based on a local search algorithm called adaptive search. Its architecture allows in particular to define a constraint problem visually, to solve it, and eventually to edit partial or approached results during the resolution process.

1 Introduction

Computer Assisted Composition (CAC) deals with a symbolic representation of music, mainly at the score level (or before), as opposed to sound synthesis, accoustic, etc. It is now a well-established research area see [3] for a general presentation. CAC provides the composer with computing tools, for him to handle with formal representations of music, and make calculation on these representations. One of the recent CAC softwares is OpenMusic [2], a full visual programming language based on CommonLisp / CLOS, developped at IRCAM. OpenMusic is a functional, object, visual programming language. A set of provided classes and libraries make it a very convenient environment for music composition. OpenMusic is developped jointly by computer scientists and composers, which gives the opportunity to consult musician-users about the use they could do of new computer techniques.

On the other hand, Constraint Programming (CP) is also a well established research area. It is an artificial intelligence technique which has been attracting a growing interest for the last decades, both in the academic and industrial areas. CP allows to represent a problem by specifying some partial information on it. It offers a powerful and intuitive way of describing and solving combinatorial probems. A CP problem is structured as a Constraint Satisfaction Problem, with a set of variables (such as unknows in mathematics), domains of possible values for those variables, and constraints which are predicates stated on the variables. The goal is to find an affectation of values to the variables such that the constraints are satisfied. Many techniques have been developped to solve CSPs. Some of them are called complete, they rely on an exhaustive search of all the possible combinations of values. In order to handle with the combinatorial explosion, high level notions have been introduced to try and reduce the size of the search space, mainly Arc-Consistency, which forbids incompatibilities for the values of the domains. These inconsistent values (values which, once combined to any other one, would lead to a failure) combined can be removed from the domains. Several algorithms have been developped since AC-1-3 [13].

On the other hand, incomplete methods have been used for the last decades in Combinatorial Optimization in order to find optimal or near-optimal solutions. They find their origin in the pioneering work of Lin on the Traveling Salesman Problem [12], see for instance in [1] [10]. The so-called local search techniques encompass a large class of complex methods, the best-known instances being simulated annealing, Tabu search and genetic algorithms, usually referred to as "meta-heuristics". They work by iterative improvement over an initial state. Consider an optimization problem with cost function which makes it possible to evaluate the quality of a given configuration (assignment of variables to current values) and a transition function that defines a set of "neighbors" for each configuration. The basic algorithm consists in starting from a random configuration, explore the neighborhood, select an adequate neighbor and then move to the best candidate. This process will go on until some satisfactory solution is found. To avoid being trapped in local optima, adequate mechanisms should be introduced, such as the adaptive memory of Tabu search, the cooling schedule of simulated annealing or similar stochastic mechanisms. Very good results have been achieved by dedicated and finely tuned local search methods for many problems such as the Traveling Salesman Problem, scheduling, vehicle routing, cutting stock, etc. Recently, these techniques have also been mixed either one with the other, or with complete methods, or with operationnal research, giving some very promising results.

Applying Constraint Programming to classical music is a very natural idea, because of its harmonization rules which are constraint-like stated. The textbooks give rules like "no parallel fifth", "opposite motion between two voices", and so on. They are so to say written in a declarative way. The first applications of CP to music aimed at solving automatically the harmonization exercises, for four voices. The first solver was proposed by Ebcioglu [9], followed by Tsang and Aitken [22]. Philippe Ballesta wrote a solver based on Ilog Solver [5]. Pachet and Roy used the particular structures of the tonal music to build a system with BackTalk [16]. All these systems apply on a very specific problem, automatic harmonization in the choral style, and stand on a specific music theory, tonal music.

Two solvers have been developed for contemporary music, PWConstraints [11] and Situation [19]. Both are based on the forward checking algorithm. They are useful for certain types of problems, but still lack the efficiency, the genericity (for PWConstraints) and the ease of use that are crucial to real production situations.

Some applications of Constraint Programming to other fields of Computer Music are to be found in [15], where constraints are used as high level tools to control spatialization, [20] for a model of timed concurrent constraints as "concurrent" musicians in an orchestra, [17] for musical mosacing, a way of recreating pieces of music from a sample database, and [6] where constraints specify hierarchical temporal relations between some boxes, which can contain sound files for instance.

The applications of CP to music thus cover a very wide range of musical structures and ideas. This is to be linked with the great variety of musical

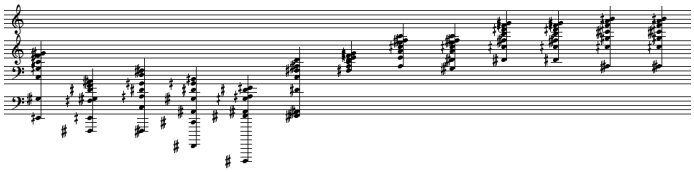

Figure 1: A solution of problem 1, with exactly two common notes on the whole sequence.

applications of computer science to music. In any way, it can hardly be asserted that constraints are used as a specific tool in music : most of the time, the applications are mainly guided by musical requirements and CP seems to have been chosen as a tool among others.

2 Musical CSPs

This section details some musical constraint satisfaction problems. The first 9 problems are pure CAC problems, which means they have been proposed by composers or musical assistants, except for problems 3 and 9 which are classical musical problems. Problems 10, 11 and 12 come from musical analysis, from works by Marc Chemillier. Problem 13 is a CAC problem, but cannot be easily compared to the other ones. Problem 14 is an instrumental problem, which deals with physiological constraints. We have chosen to classify the CSPs according to their musical nature.

2.1 Harmony

There are three problems dealing with harmonical structures. Problem 1 has been stated by Fabien Levy, composer. It concerns n aspectral chords, which are modelized by three parameters : a virtual fundamental, the interval between two frequencies, and a number of notes. The goal is to find some of these chords such that two successive chords have a fixed number of common notes (optionnally, with the a number of common notes comprised between fixed minimum and maximum / with the same common notes on the whole sequence). In addition, there are optional constraints ruling the behaviour of the sequence (increasing or decreasing intervals, increasing or decreasing fundamentals). Figure 1 gives an example with exactly two common notes, figure **??** an example with an increasing fundamentals and decreasing intervals, and figure **??** an example with the same common note for the whole sequence.

Problem 2 is rather complex. It has been stated by the composer Georges Bloch. Given a particular rhythmical structure of 6 or 12 voices, the goal is to find a harmony which minimizes a musical distance, the so-called Estrada distance, between two successive chords. Estrada distance measures the differences in the intervals between two chords. A second constraint consist in minimizing a

distance (corresponding to the cycle of fifths) between the virtual fundamental of two successive chords. A third constraint is better expressed as a preference, and states that the melodies formed on each voice have to be near to a fixed melodic profile. Finally, a constraint is added in order to avoid trivial solutions (same note everywhere). This problem is overconstrained.

Problem 3 consists in sorting a sequence of chords, such that two successive chords have the maximal number of common notes. As a CSP, this is exactly a Travelling Salesman Problem. Take the cities as the chords, and the distance between two cities as the number of common notes.

It is worth noticing that these three problems have quite similar contraints : minimize a distance-like function between two successive chords. From this point of view, these problems are very similar. But we see here the main challenge of applying constraint techniques to musical representation : although these three problems are stated on similar structures (sequence of chords), their representations are very different : sets of frequencies, a very complicated structure inherited from some rhythmical properties, and a set of integers.

2.2 Rhythms

Four problems concern rhythmical structures. Problem 4 has been stated by the composer Mauro Lanza. The goal is to find n rhythmical patterns of fixed lengths, each pattern played on one voice, such that two different voices never play an onset simultaneously for a fixed duration. We can use the Chinese Remainder Theorem to reduce this problem to an array of integer equations, which gives a caracterization of the number of solutions : there is no solution unless the greatest common divisor between the lengths of any two voices is greater than the duration.

Problem 5 has been stated by the composer Geoffroy Drouin. It relies on the famous Fibonnacci series defined as $u_0 = 0$, $u_1 = 1$, $u_n = u_{n-1} + u_{n-2}$. The goal is to find durations which are values of this series, two successive durations being next to each other in the series, and the sum of all the durations being fixed. There are two modelizations for this problem, either taking the variables as the durations, or the variables ranging over $\{-1, 1\}$, representing the back or forward move in the indexes of the series.

Problem 6 has been stated by the composer Peter Klanac. He wanted to write a smooth accelerando, but still playable by the interpret, who usually reads the new tempo on the score as a ratio with the current tempo (such as : quarternote = eigthnote). Variables are tempi $t_1...t_n$, ranging over the integer set from 40 to 250. Constraints rule the ratios between t_i and t t_{i+1}, t_{i+2}, t_{i+3}, and t_{i+4}. The ratios have to be as near as possible of fixed values ($0,9$, $0,8$, $0,75$ and $0,666$). This problem would be seen as a geometrical series by a mathematician, but it is not, because we have to stay in the integer set. This problem has then no solution, and the goal is to find approximate solutions. Concerning the optimization options, Peter Klanac decided to optimize, in the first place, the number of exact ratios on t_i and t_{i+3} (count the number of times when $t_i/t_{i+1} = 0.75$, and then to minimize the gap between minimum

and maximum of t_{i+1}/t_i and 0.9. It is important to notice that the constraints are written exactly in the same way, but the notion of approximate solutions are not the same for the composer.

Problem 7 has been proposed by Gilbert Nouno, for a piece with Steve Coleman. It has not been used because of real time requirements. The data are a set of integer durations, which come from a rhythm played on congas by a percussionnist. This durations correspond to a symbolic rhythm, but they may differ from the exact durations. The goal is to find the symbolic rhytm which fits the played duration best.

The observation of these four problems does not allow much deduction : all of them deal with rhythms, but they have very little in common. The musical structures are definitely not the same : patterns, tempi, played durations. We can notice that the question of representing rhythms in CAC is usually a difficult problem. Rhythm is a notion shared by all musicians, but this notion is very contextual, and thus their representations depends on the musical context.

2.3 Melodies

Two problems concern melodies, in the sense of horizontal sequences of notes. Problem 8 has been submitted by Gilbert Nouno, musical assistant of Michael Jarrel. It is about musical gestures, a gesture being either an interval or a short series of intervals (the domains are closed under the opposite sign). A first note is fixed, to which the gestures are added in order to form a melody. The goal is to find a series of these gestures, such that the resulting notes are in a fixed set. There are two possible modelisations, with the variables as gestures and constraints on notes or with variables as notes and constraints on gestures. The first one is better because of the musical purpose, which is to find gestures, and because we will search for approximate solutions. There are three other constraints. Firstly, in order to avoid trivial solution, we impose that two successive gestures are neither the same nor opposite. Secondly, the percentages of some particular values for gestures may be fixed. Finally, the whole sequence has to correspond to melodic profile, given as a curve, but this can be passed as a domain reduction.

Problem 9 is a well known problem, the all-intervals series. The goal is to find a musical series (permutation of the 12 notes of the chromatic scale, but it can be generalized to a permutation of the n first integers), such that the intervals heard when the series is played are all different. There is a trivial solution 1 12 2 11 3, and of course we want non-trivial solutions.

These two problems are very interesting, representing the two extremes of musical CSPs : one is academic, well defined, difficult, and the existence of solutions is known. The other one is and looks "real", the modelisation is not obvious, it has many optional constraints, and the number of solutions varies from many to zero. Still, they share the same property to have two possible modelizations, one with notes and one with intervals, modelisations equivalent if we search for exact solutions, but not for approximate solutions.

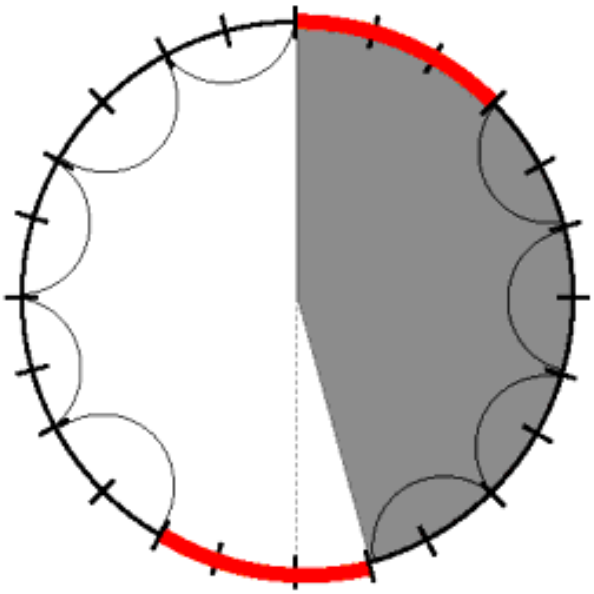

Figure 2: The rhythmical imparuty property.

2.4 Musical analysis

There are three problems, from works with Marc Chemillier, dealing with musical analysis issues. Problem 10 concerns a musical skeleton found by Chemillier in a piece of Ligeti, Melodien (and also in Continuum). A long part of Melodien shows some agregates which are moving slightly. The analysis found an harmonical structure with some properties, from which the agregate sequence can be extracted. Here, the variables are the notes of the underlying harmony, and constraints are used to restrict the melodic motions of these notes. Some of these notes are really played in the score, but not all of them.

Problem 11 concerns a very particular property found by Simha Arom in Aka pygmies music in Central Africa, and studied by Chemillier. This is called "rhytmical imparity", and caracterized rhythmical series of groups of 2 and 3 time units, played repetitively. Such a sequence is rhythmically impair iff it is impossible to cut it into two equal pieces, see figure 2. This is easy to model as a CSP, though enumerating the solutions is not obvious if not by running a CSP program.

Problem 12 is a model given by Chemillier of some musical structures in Nzakara music, in Central Africa. Nzakara people play some vocal pieces accompagnied by a traditional harp, who repeats a canon-like sequence in an *obstinato*. This canon respects some rules : they have two voices, never play the same chords twice in a row, last a fixed duration, and the lower voice is transposed from the upper one, apart from a few notes. Chemillier showed

that the number of errors was determined by the duration and the gap of the canon. Here the variables are again the chords, and constraints are the same as mentionned.

The idea of modelling a musical analysis, provided it is expressed in a mathematical-like language, as constraints, can seem straightforward. A musical analysis consists in giving *a posteriori* some rules that the piece respects. Here, there is an interesting parallel between constraint programming and such musical modelling. We could argue that the number of solutions to the CSPs is in some way representative of the accuracy of the analysis : if the CSPs has no solution, the analysis is false, if it has many, then the analysis can apply to many other pieces than the target one, and is not very specific to a particular score, then not very accurate. If the CSPs has one or a few solutions, then we can argue the analysis is quite specific to the score (see the analysis of Riotte and Mesnage [14], [18]). To go a little further, we can add that the number of constraints gives an indication of the analysis' interest. The limit case would be to modellize a score by simply enumerating its content (constraints such as "first note is this", etc). The less constraints there are, the more high level is the analysis, in some way. it was not our goal to work on musical analysis, by lack of competences, and we do not pretend to have found a universal way of doing analysis, which is a work far too subtle to be represented in such a way. Those remarks are about a particular work, where the parallelism between both representations (CSPs and analysis) applies well.

2.5 Others

Problem 13 deals with enumerating musical scales in quarter tones. The whole set of these is far too big to listen to, and some constraints are added here in order to restrict the enumeration of these scales to the ones of musical interest. The most interesting constraint we have tried is probably the one which forces the scale to have a least number of chords "sounding" in quarter-tones, that is, in which the inner intervals do not reduce as semitones intervals.

Problem 14 is about fingerings for guitar. There are two CSPs, the first one to find fingerings for a single chord for the guitar, the second one to find fingerings for a chord sequence, such that for instance the number of position's changes is minimal. This problem is difficult to formalize because of the great number of parameters, due to the interprets' features (size of the hand, barr or not , etc).

2.6 Special features of these musical CSPs

As a preliminary remark, let us stress the fact that most of these CSPs have been given by composers, or musicians. This confirms the idea that constraint representations has a role to play in music. Main features of the CSPs are summarized table 1. As mentionned, these CSPs show a great diversity both in their constraints and in their musical structures. Every kind of classical constraints primitives, from equality to capacity, appears. Musical structures

Table 1: MAin features of the musical CSPs

CSP	Var	Dom	Cont	Nature	Solutions	Type
1	60	1000	2 / 4	Acc freq	Oui	Résolution
2	144	24	4	Mélodies	Non	Résolution
3	50	Perm	2	Accords	Non	Optimisation
4	25	40	2	Rythmes	+/-	Résolution
5	?	∞	1	Dures	+/-	Rsolution
6	25	144	3	Tempi	Oui	Optimisation
7	20	?	1	Durées	Oui	Optimisation
8	40	6	4	Mélodies	Non	Résolution
9	20	Perm	1	Notes	Oui	Résolution
10	10/500	4	4	Accords	Oui	Génération
11	30	Perm	1	Rythmes	Oui	Génération
12	30	5	2	Accords	Non	Résolution
13	24	3	?	Intervalles	Bcp	Génération
14	?	20	?	Doigts	Bcp	Rs / Opt

ranges over the whole set of object which can be found in a score, from notes to tempi, and even out of the score for problems 7 and 14. This is of course due to the fact that every composer thinks with its own musical structures. We can also notice that the variables are often not exactly taken on the score, but above it, as some integers which represent a feature of a musical object (typically, a MIDI value for a note).

During the modellisation process, we often had to add some contraints to ensure that the solutions would not be degenerated (chords with only one note for instance). This is why there is an alldifferent constraint in nearly every CSP.

More important, the goal is not always to find a solution, which may seem unusual to a constraint researcher. Nor is it to answer anything. It is to provide interesting instanciations to the composer. Firstly because most of the CSPs are overconstrained, and have no solutions at all. Secondly, it happens that an approximate solution is musically more interesting than an exact one. We shall

never forget the exact place of constraint solving in the compositional process : it is a mere tool, for the composer. It is used at the rough draft stage of the composition, not only because it is CAC, but also because beeing a solution doesn't guarantee beeing musically good. In particular, the solutions are nearly always re-written by hand. So the usual CSP paradigm "problem → solution" is no longer valid, we'd rather consider it as "specify some partial information on musical objects → make suggestions to the composer".

3 Adaptive search

3.1 Introduction

Heuristic (i.e. non-complete) methods have been used in Combinatorial Optimization for finding optimal or near-optimal solutions for a few decades. They work by iterative improvement over an initial state. Consider an optimization problem with cost function which makes it possible to evaluate the quality of a given configuration (assignment of variables to current values) and a transition function that defines for each configuration a set of "neighbors". The basic algorithm consists in starting from a random configuration, explore the neighborhood, select an adequate neighbor and then move to the best candidate. This process will continue until some satisfactory solution is found.

We use a new heuristic method proposed by Philippe Codognet and Daniel Diaz, called Adaptive Search (AS) for solving CSP. [7] shows some benches where AS is compared for instance to Localizer, and shown to be very fast. Our method can be seen as belonging to the GSAT [21], Walksat [4] and Wsat(OIP) [23] family of local search methods.

3.2 Algorithm

The input of the method is a problem in CSP format. Although we will completely depart in adaptive search from the classical constraint solving techniques (i.e. Arc-Consistency and its extensions), we will take advantage of this formulation of a problem as a CSP, in order to analyze the current configuration more carefully than a global cost function to be optimized. Accurate information can be collected by inspecting constraints and combining this information on variables.

Our method is not limited to any specific type of constraint, though we need, for each constraint, an *error function* that will give an indication on how much the constraint is violated. Adaptive search relies on iterative repair, seeking to reduce the error on the worse variable so far. The basic idea is to compute the error function of each constraint, then combine for each variable the errors of all constraints in which it appears, therefore projecting constraint errors on involved variables. Finally, the variable with the maximal error will be chosen as a "culprit" and thus its value will be modified. In this second step we select the value in this variable's domain that has the best value, that is, the value for

which the total error in the next configuration is minimal.

In order to prevent the algorithm from being trapped in local minima, the adaptive search method also includes an adaptive memory as in Tabu Search: each variable leading to a local minimum is marked and cannot be chosen for the few next iterations.

Repeat, until a solution is found or a maximal number of iterations is reached

- Compute errors of all constraints and combine errors on each variable by considering for a given variable only the constraints on which it appears.
- select the variable X (not marked as Tabu) with highest error and evaluate costs of possible moves from X
- if no improving move exists
  then mark X tabu for a given number of iterations else select the best move (min-conflict) and change the value of X accordingly

3.3 Using AS for musical problems

This method answers well to our specifications for musical constraint system. First, the variables are always instanciated, and the value of successive instanciations is being continuously improved. So we always have partial results (either local minima, or the best local minimum found so far, or the current instanciation). This allows to give partial solutions on demand, keeping the CSP formalism. This idea is used in OMClouds to edit partial results during the resolution, or to deal with approximate solutions for overconstrained problems.

Furthermore, the representation of constraints with cost-functions gives an additional flexibility to the program. We can easily give more or less importance to the constraints, simply by weighing their cost-functions, and eventually by playing with the end treshhold.

Finally, it deals well with the requirement of CAC to give answers on request. Computing the first ten solutions of a CSP, a complete solver is likely to give very similar solutions, with for instance the first variables having the same values. On the opposite, a incomplete solver will be randomly reset ten times, and give probably ten very different solutions, thus providing the user with a welcome diversity.

For all of these reasons, Adaptive Search was particularly well adapted to the composers needs.

4 OMClouds

This last section presents the OMClouds library, which implements the adaptive search algorithm in OpenMusic. OMClouds is distributed with OpenMusic since april, 2003, version 4.6.5 and up.

4.1 Technical choices

About the CSP modelisation, the variables are supposed to be homogeneous, that is, we suppose that the constraints state in the same way on all the variables. Thus, variables don't need to be named, and are represented by their indexes, as if they were placed in a list. Constraints hold on some variables located by the gap in their indexes. So the constraints are in the form of "minimize a distance between variable i and variable $i + 1$". There are three kind of CSPs : lists (variables are placed in a list), cycles (variables are in a list that is supposed to be repeated cyclically), and permutations (domains are the permutations of the variables). The interface is visual as in OpenMusic. OMClous has generally been made in order to respect the main features of OM as far as possible.

To take profit of OM being object oriented, we have chosen to define a CSP object (three of them actually, one for each kind of CSP), which contains all the information needed to represent, solve, and edit the CSP : current configuration, domains, constraints, and also tabu list, errors, global error, and so on. The main functions about the resolution are thus generic functions, which can be redefined for new CSPs structures, or in order to modify some heuristic choices like the neighborhood exploration for instance.

4.2 CSP definition

A problem is defined in an OM box, called for instance the function cree-varliste in case of a list of variables (the architecture is comparable for the three kinds of CSPs). This box is an instance of asboxlist, a subclass of the OM class box-with-patch, which has an attached patch. When opened, the patch of any asbox shows some predefined icons representing the current variable, the whole variable list, and an output, plus two buttons to add either variables, or outputs, see figure 3. This allows to define the constraint visually, in OpenMusic manner, by using the constraint primitives (see figure 4), and connecting them to the "state" outputs. In this case, the constraint are translated into cost functions following tabular 3. A min-conflict like translation has also been implemented. Finally, it is also possible for the user to write his own cost-functions, just by connecting them to the state outputs.

4.3 Resolution

The asboxes have two outputs. The first one gives the instance of the corresponding CSP class, and is intended to be directly connected to the resolution box. In this configuration, the resolution box simply takes the CSP object and applies adaptive search until the global error is below a certain ϵ, zero by default. When a solution is found, the values are given in the output of the resolution box.

The second way of using OMClouds allows to deal with approximate solution, during the search process. Again, the resolution box is an instance of the box-resol, subclass of the OpenMusic OMBoxCall class. The evaluation method has

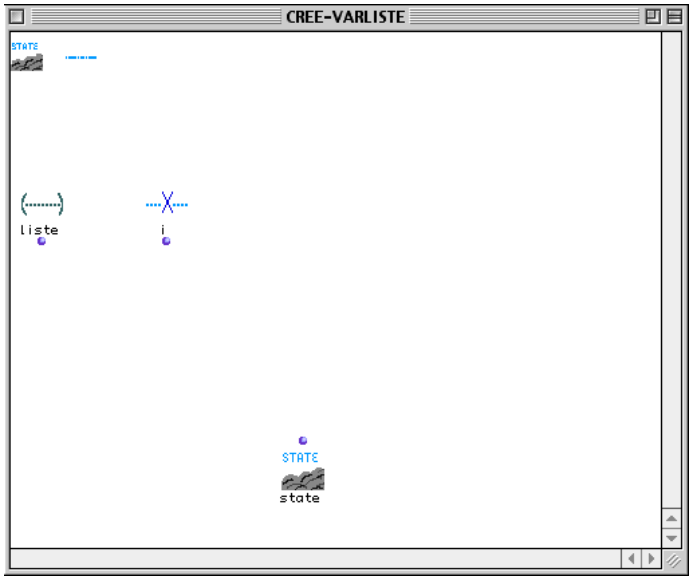

Figure 3: Visual definition of the constraints.

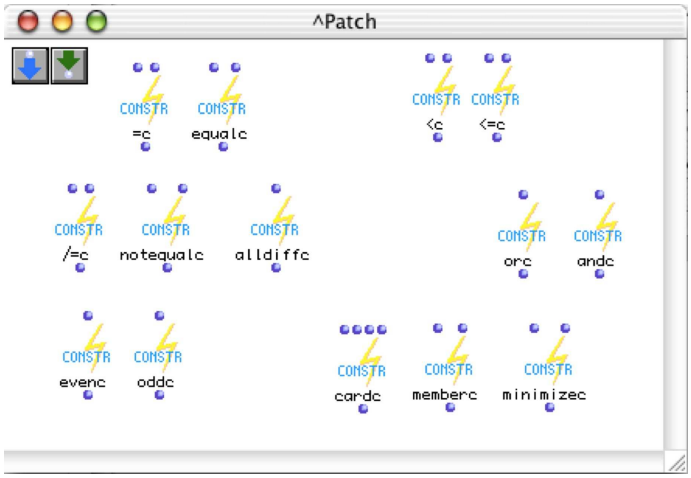

Figure 4: The constraint primitives provided with the library.

Table 2: The constraints are automatically translated into cost-functions, according to the following rules.

Contrainte C	Cot f_C
x_1 =c x_2	— x_1 - x_2 —
x_1 equalc x_2	0 si x_1 equal x_2
	1 sinon
x_1 ¡c x_2	max (1, x_1 - x_2)
x_1 ¡=c x_2	max (0, x_1 - x_2)
x_1 /=c x_2	1 si $x_1 = x_2$)
	0 sinon
x_1 notequalc x_2	1 si x_1 equal x_2
	0 sinon
C_1 andc C_2	max (f_{C_1}, f_{C_2})
C_1 orc C_2	min (f_{C_1}, f_{C_2})
evenc x	0 si x est pair
	1 sinon
oddc x	0 si x est impair
	1 sinon
minimizec x	x
alldiffc x	nombre de redondances dans x
cardc i l el n	— (nombre d'lments de la l valant el) - n —

been redefined for this box-resol boxes, in order to deal with the second output of a asbox class. This second output is set by the resolution box to the values of the last local minimum encountered during the search process. This allows to connect these values to any kind of OM calculus or musical editors. Then, during the search process, the resolution box re-evaluates its second input each time a new local minimum is found. This activates the evaluation of everything that is connected to it, from the second output of the asbox. What is important here is the fact that the local minimum can be shown during the search process, and processed or edited by any kind of OM calculation. According to the notes on the musical CSPs, OMClouds does not simply give the intermediate values, but also leaves the possibility to process them so that they can be edited in a score like editor.

The main classes used in OMClouds are shown on figure **??**.

We have measured some performances in OMClouds on the all-intervals series, n-queens, and asynchronous rhythms problems. The results are far slower than the one presented in [8], with an implementation in C of the same algorithm. Several factors explain this, among others, we have chosen to generate the cost-functions and pass them as parameters, which forbids optimization on these, although they are often calculated in a single step of the program. Anyway, the results are sufficient for our application.

5 Conclusion (english)

We have presented a catalogue of fourteen musical Constraint Satisfaction Problems, and a set of requirement for using so a formalism in Computer Assisted Composition. A local search method has mainly been used to solve these problems, and implemented as a library of the software OpenMusic. Music offers to Constraint Programming original problems, and an application where the use of a solver must be seen in a very flexible way.

We have contributed (from far) to eight creations. Generally, constraint programming has confirmed to apply well to contemporary music. The instrumental application to guitar fingerings seems promising and could be extended to other instruments (but in order to to it, a musical expert is needed). Furthermore, local search, where a 'best instanciation computed so far" can be stored, could be used in anytime applications.

References

[1] E. H. L. Aarts and J. K. Lenstra. *Local search in combinatorial optimization.* John Wiley and Sons, 1997.

[2] Carlos Agon. *An environment for computer assisted composition.* Thèse de doctorat, IRCAM-Universit de Paris VI, 1998.

[3] Gérard Assayag. Applications on contemporary music creation, esthetic and technical aspects. *1st Symposium on Music and Computers*, 1998.

[4] H. Kautz B. Selman and B. Cohen. Noise strategies for improving local search. *Proc. AAAI'94*, 1994.

[5] Philippe Ballesta. *Contraintes et objets, clefs de voûte d'un outil d'aide à la composition*. Editions Hermès, 1998.

[6] Anthony Beurivé and Myriam Desainte-Catherine. Representing musical hierarchies with constraints. *Proceedings of MusiCons workshop at CP'01*, 2001.

[7] Philippe Codognet, Daniel Diaz, and Charlotte Truchet. The adaptive search method for constraint solving and its application to musical csps. *Proceedings of the First International Workshop on Heuristics*, 2002.

[8] Daniel Diaz and Philippe Codognet. Design and implementation of the gnu prolog system. *Journal of Functional and Logic Programming*, 6, 2001.

[9] Kemal Ebcioglu. *An expert system for harmonizing chorals in the style of J.-C. Bach*. AAAI Press, 1987.

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

[11] Mikaël Laurson. Patchwork : a visual programming language and some musical applications. *Sibelius Academy*, 1996.

[12] S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44:2245–2269, 1965.

[13] A.K. Mackworth and E.C. Freuder. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems [ac1-3]. *Artificial Intelligence*, 25:65–74, 1985.

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

[15] François Pachet, Olivier Delerue, and Peter Hanappe. Dynamic audio mixing. *Proceedings of ICMC*, 2000.

[16] 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.

[17] François Pachet and Aymeric Zils. Musical mosaicing. *Proceedings of DAFX'01*, 2001.

[18] 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.

[19] Camilo Rueda, Mikael Laurson, Georges Bloch, and Gérard Assayag. Integrating constraint programming in visual musical composition languages. *Proceedings of ECAI'98*, 1998.

[20] Camilo Rueda and Franck Valencia. Formalizing timed musical processes with a temporal concurrent constraint programming calculus. *Proceedings of MusiCons Workshop at CP'01*, 2001.

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

[22] C. P. Tsang and M. Aitken. Harmonizing music as a discipline of constraint logic programming. *ICMC*, pages 61–64, 1991.

[23] Joachim P. Walser. *Integer Optimization by Local Search : a Domain-independant Approach*. Springer Verlag, 1999.