Finite-Length Markov Processes with Constraints

François Pachet, Pierre Roy, Gabriele Barbieri
Sony CSL Paris
{pachet,roy,barbieri}@csl.sony.fr

Abstract

Many systems use Markov models to generate finite-length sequences that imitate a given style. These systems often need to enforce specific control constraints on the sequences to generate. Unfortunately, control constraints are not compatible with Markov models, as they induce long-range dependencies that violate the Markov hypothesis of limited memory. Attempts to solve this issue using heuristic search do not give any guarantee on the nature and probability of the sequences generated. We propose a novel and efficient approach to controlled Markov generation for a specific class of control constraints that 1) guarantees that generated sequences satisfy control constraints and 2) follow the statistical distribution of the initial Markov model. Revisiting Markov generation in the framework of constraint satisfaction, we show how constraints can be compiled into a non-homogeneous Markov model, using arc-consistency techniques and renormalization. We illustrate the approach on a melody generation problem and sketch some real-time applications in which control constraints are given by gesture controllers.

1 Introduction

Markov processes are a popular modeling tool used in content generation applications, such as text generation, music composition and interaction. Markov processes are based on the "Markov hypothesis" which states that the future state of a sequence depends only on the last state, i.e.,

$$p(s_i|s_1, \ldots, s_{i-1}) = p(s_i|s_{i-1}).$$

The Markovian aspects of musical sequences have long been acknowledged, see e.g. [Brooks *et al.*, 1992]. Many attempts to model musical style have therefore exploited Markov chains in various ways [Nierhaus, 2009], notably sequence generation.

In practice, Markov models are often estimated by counting occurrences and transitions in a corpus of training sequences. Once the model is learn, sequences can be generated simply by *random walk*: the first item is chosen randomly using the prior probabilities; then, a continuation is drawn using the model, and appended to the first item. This is iterated to produce a sequence of length L. This process has the advantage of being simple to implement and efficient.

For instance, the Continuator [Pachet, 2002] uses a Markov model to react interactively to music input. Its success was largely due to its capacity to faithfully imitate arbitrary musical styles, at least for relatively short time frames. Indeed, the Markov hypothesis basically holds for most melodies played by users (from children to professionals) in many styles of tonal music (classical, jazz, pop, etc.). The other reason of its success is the variety of outputs produced for a given input. All continuations produced are stylistically convincing, thereby giving the sense that the system creates infinite, but plausible, possibilities from the user's style.

With the Continuator, a user typically plays a musical phrase using a MIDI keyboard. The phrase is then converted into a sequence of symbols, representing a given dimension of music, such as pitch, duration, or velocity. The sequence is then analyzed by the system to update the Markov model. When the phrase is finished, typically after a certain temporal threshold has passed, the system generates a new phrase using the model built so far. The user can then play another phrase, or interrupt the phrase being played, depending on the chosen interaction mode. It was shown that such incremental learning creates engaging dialogs with users, both with professional musicians and children [Addessi and Pachet, 2005]. Other systems such as Omax [Cont *et al.*, 2007] followed the same principle with similar results.

In such interactive contexts, the control problem manifests itself under different forms:

- The so-called *zero-frequency* problem arises during random walk, when an item with no continuation is chosen (see, e.g. [Chordia *et al.*, 2010]). Many strategies have been devised to circumvent this problem, including restarting the walk from scratch [Dubnov *et al.*, 2003].
- The *end point* or *drift problems* [Davismoon and Eccles, 2010] concern the fact that the generated sequence can violate musical constraints holding, e.g., on the pitch range of the melody.
- User *control constraints*. In a musical context, the user may want the sequence to be globally ascending, pitchwise, or to follow an arbitrary pitch contour. These constraints can be a consequence of a particular gesture, de-

tected, e.g., by input sensors. For instance, it may be interesting to improvise a melody that ends on a particular note. This is a striking effect used in Hard-Rock guitar virtuoso improvisations, which often consist of a frantic sequence of notes ending gracefully on a note of the triad (e.g. the tonic of the underlying chord).

Most approaches proposed to deal with these issues consist in grafting heuristic search on top of random walk: simulated annealing [Davismoon and Eccles, 2010], case-based reasoning [Eigenfield and Pasquier, 2010], generate-and-test [Dubnov *et al.*, 2003]. However, these methods do not offer any guarantee that a solution will be found. Moreover, the solutions found are not consistent with the underlying Markov model, i.e., the probability to draw them is not the Markov probability. Lastly, these methods are not efficient enough for interactive, real-time applications.

Mixed Networks [Dechter and Mateescu, 2004], a special case of graphical models, combine constraints and probabilities in a coherent manner. However the problem addressed is different: infer the *posterior probabilities* of the new model obtained mixing constraints and probabilities. Instead we generate a new model that, when combined with the constraints, is equivalent, probability-wise, to the initial Markov model.

It is important to note that Hidden Markov Models (HMM) cannot be applied, because in our context the Bellman principle does not hold. Control constraints cannot be modeled as *cumulative cost functions*: even anchor constraints (imposing a fixed value at a specific position) may have implications on the whole sequence, as we show below.

As outlined by [Pachet and Roy, 2011], control constraints raise a fundamental issue since they establish relationships between items that violate the Markov hypothesis: obviously as soon as the constraint *scope* (the variables on which the constraint holds) is larger than the Markov scope, the information cannot be represented in a Markov model. However, even constraints that remain within the Markov scope (e.g., unary constraints) create implicit dependencies that violate the Markov hypothesis.

[Pachet and Roy, 2011] show that the reformulation of the problem as a constraint satisfaction problem allows, for arbitrary sets of control constraints, to compute optimal, singular solutions, i.e., sequences that satisfy control constraints while being optimally probable. However, what is often needed in practice is a *distribution* of good sequences. We cannot use the approach of [Pachet and Roy, 2011], as it does not produce a distribution of sequences, but only optimal solutions. Furthermore, it involves a complete search-optimization algorithm, which limits real-time use.

In this paper, we show that when control constraints do not exceed the Markov scope (unary and binary adjacent for order-1, as defined below) they can be "compiled" into a new Markov model that is statistically equivalent to the initial one. This yields the advantage of retaining the simplicity of random walk, while ensuring that control constraints are satisfied. This result is obtained by establishing yet another bridge between Markov generation and constraint satisfaction.

Section 2 introduces an example to illustrate our problem, Section 3 states the problem in the framework of constraint satisfaction, Section 4 describes how to build the new model and Section 5 sketches some applications.

Figure 1: The two input melodies used to estimate M.

Melodies	Probabilities	Melodies	Probabilities
CCCD	1/ 32	EECD	1/ 96
CCED	1/ 64	EEED	1/ 192
CECD	1/ 64	EDCD	1/ 96
CEED	1/ 128	EDED	1/ 96
CDCD	1/ 64	DCCD	1/ 96
CDED	1/ 64	DCED	1/ 192
ECCD	1/ 48	DECD	1/ 96
ECED	1/ 96	DEED	1/ 192

Table 1: The 16 4-note melodies satisfying the control constraint and their probabilities in M. The sum of probabilities for these sequences, $\sigma = 77/384$, is not 1 because M generates sequences that do not satisfy the constraint.

2 A Melody Generation Example

Let us consider a Markov model M estimated from the two sequences (melodies) shown in Figure 1.

The prior vector of M is:

$$\begin{array}{ccc} C & D & E \\ (1/2 & 1/6 & 1/3) \end{array}$$

The transition probabilities of M are:

$$\begin{array}{c} \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccc} C & D & E \end{array} \\ \begin{pmatrix} 1/2 & 1/4 & 1/4 \\ 1/2 & 0 & 1/2 \\ 1/2 & 1/4 & 1/4 \end{pmatrix} \end{array}$$

We consider the problem of generating 4-note melodies. There are 60 possible such melodies with non-zero probabilities. For instance, sequence $CDED$ has probability:

$$p_M(CDED) = p_M(C)p_M(D|C)\, p_M(E|D)\, p_M(D|E) = \frac{1}{64}.$$

We then add the control constraint that the last pitch be a D. There are only 16 such sequences (see Table 1).

Our goal is to build a Markov process $\tilde{M}$ that generates exactly these 16 melodies with the same probability distribution. In general, of course, the solutions and their probabilities are not known. We show here how to build $\tilde{M}$ given M and control constraints, for a particular class of constraints.

3 Problem Statement

We are interested in generating fixed-length sequences from a Markov process M that satisfy control constraints. To use a random walk approach, we need a Markov process $\tilde{M}$ that generates exactly the sequences satisfying the control constraints with the probability distribution defined by M. In

general, it is not possible to find such a Markov process because control constraints violate the Markov property, as outlined by [Pachet and Roy, 2011]. However, when control constraints remain within the Markov scope, we show that such a model exists and can be created with a low complexity.

For the sake of simplicity, we consider only order-1 Markov processes, defined by a stochastic transition matrix (i.e., each row sums up to 1). Generalization to higher and variable orders is discussed in Section 4.5.

A Markov process M is defined over a finite state space $A = \{a_1, \ldots, a_n\}$. A sequence s of length L is denoted by $s = s_1, \ldots, s_L$ with $s_i \in A$. S is the set of all sequences of length L generated by M with a non-zero probability:

$$p_M(s) = p_M(s_1) \cdot p_M(s_2|s_1) \cdots p_M(s_L|s_{L-1}).$$

Following [Pachet and Roy, 2011], we represent the sequence to generate as a sequence of finite-domain constrained variables $\{V_1, \ldots, V_L\}$, each with domain A, and Markov properties as *Markov constraints* on these variables, defined below. Control constraints are also represented as finite-domain constraints. The induced CSP is denoted by $\mathcal{P}$ and the set of solutions S_C. Given these notations, $\tilde{M}$ should verify:

(I) $p_{\tilde{M}}(s) = 0$ for $s \notin S_C$,

(II) $p_{\tilde{M}}(s) = p_M(s|s \in S_C)$ otherwise.

These properties state that $\tilde{M}$ generates exactly the sequences $s \in S_C$. Most importantly, sequences in S_C have the same probabilities in M and $\tilde{M}$ up to a constant factor $\alpha = p_M(s \in S_C)$, i.e., $\forall s \in S_C, p_{\tilde{M}}(s) = 1/\alpha \cdot p_M(s)$. In the running example, $\alpha = \sigma$.

Our main result is that for a certain class of induced CSPs, hereafter referred to as *Binary-Sequential CSPs*(BSC), there exists a *non-homogeneous* Markov process $\tilde{M}$ that satisfies (I) and (II). We define the scope of a constraint as the interval between its leftmost and rightmost variables. A Binary-Sequential CSP is a CSP that contains only constraints whose scope remains within the scope of the Markov order. With a Markov order of 1, these constraints consist in 1) unary constraints and 2) binary constraints among adjacent variables (see Section 4.1).

A non-homogeneous Markov process (NHM) is a Markov process whose transition matrices change over time [Kolarov and Sinai, 2007]. A NHM of length L is defined as a series of transition matrices $\tilde{M}^{(i)}, i = 0, \ldots, L-1$.

We now describe how to build $\tilde{M}$ from M and its induced BSC, and show that $\tilde{M}$ achieves the desired properties.

4 Construction of $\tilde{M}$

$\tilde{M}$ is obtained by applying two successive transformations to the initial model M. The first transformation exploits the induced CSP to filter out state transitions that are explicitly or implicitly forbidden by the constraints. This is achieved by replacing the corresponding transition probabilities by zeros in the initial transition matrices. A side-effect is that the transition matrices are not stochastic anymore (rows do not sum up to 1 any longer). The second transformation consists in renormalizing those matrices to obtain a proper (non-homogeneous) Markov model, a step which turns out to be non trivial. We prove that this model satisfies (I) and (II).

4.1 Induced CSPs

We consider a BSC with unary control constraints $U_1, \ldots, U_L$ and binary constraints $B_1, \ldots, B_{L-1}$. U_i defines the states that can be used at position i in the sequence. B_i defines the allowed state transitions between positions i and $i+1$. Markov constraints, denoted by $K_1, \ldots, K_{L-1}$, are posted on all pairs of adjacent variables. Markov constraints represent the following relation:

$$\forall i, \forall a, b \in A, K_i = true \Leftrightarrow p_M(b|a) > 0.$$

The CSP induced by our running example is the following:

$$\underset{\{C,D,E\}}{V_1} \xrightarrow{K_1} \underset{\{C,D,E\}}{V_2} \xrightarrow{K_2} \underset{\{C,D,E\}}{V_3} \xrightarrow{K_3} \underset{\{C,D,E\}}{V_4} \leftarrow \underset{V_4=\{D\}}{U_4}$$

The first step of our process is to make $\mathcal{P}$ arc-consistent. Arc-consistency consists in propagating the constraints in the whole CSP, through a fixed-point algorithm that considers constraints individually [Bessière *et al.*, 1995]. This ensures that each constraint c holding on variables V_i and V_j satisfies:

$$\forall x \in \mathcal{D}(V_i), \exists y \in \mathcal{D}(V_j) \text{ such that } c(x, y) = true.$$

General algorithms for achieving arc-consistency were proposed [Mackworth, 1977] but specific constraints can be filtered more efficiently [Bessière *et al.*, 1999]. Specific filtering methods for Markov constraints are described in the next section.

It is important to note here that enforcing arc-consistency on a BSC $\mathcal{P}$ is sufficient to allow the computation of the transition matrices once for all, prior to the generation, with no additional propagation. This can be shown as follows:

Proposition: If $\mathcal{P}$ is arc-consistent, then for all consistent partial sequences $s_1 \ldots s_i$ (i.e., sequences that satisfy all the constraints between variables $V_1, \ldots, V_i$), the following properties hold:

(P1) $\exists s_{i+1} \in \mathcal{D}(V_{i+1})$ such that $s1 \ldots s_i s_{i+1}$ is consistent.

(P2) $s1 \ldots s_i s_{i+1}$ is consistent $\Leftrightarrow$ $s_i s_{i+1}$ is consistent.

Proof. $\mathcal{P}$ is of width 2 as its constraint network is a tree; [Freuder, 1982] proved that arc-consistency enables a backtrack-free resolution of a CSP of width 2, therefore every partial consistent sequence can be extended to a solution, which is equivalent to P1. The condition in P2 is obviously sufficient; it is also necessary as no constraint links V_{i+1} back to any other variable than V_i. □

P1 and P2 ensure that the domains of variables after arc-consistency contain exactly the valid prefixes and suffixes of the corresponding transitions. The next step is to extract the matrices from the domains, as described in Section 4.3.

4.2 Enforcing arc-consistency of $\mathcal{P}$

Arc-consistency of the Markov constraints K_i can be achieved efficiently with the following propagators:

On *instantiation*: If V_i is instantiated with $a \in A$, remove every $b \in A$ such that $p_M(b|a) = 0$ from the domain of V_{i+1}. Conversely, if V_{i+1} is instantiated with $b \in A$, remove all the $a \in A$ such that $p_M(b|a) = 0$ from the domain of V_i.

On *removal*: If $a \in A$ is removed from the domain of V_i, remove all the $b \in A$ such that $p_M(b|c) = 0, \forall c \neq a$ from the domain of V_{i+1}. The same strategy is applied when a value is removed from the domain of V_{i+1}. This can be implemented efficiently by associating a support counter with each value in the domain of V_{i+1}.

Arc-consistency of binary control constraints can be implemented with a general binary arc-consistency algorithm or a specific one, depending on the nature of the constraint.

Arc-consistency of $\mathcal{P}$ necessitates enforcing arc-consistency of all constraints until a fixed-point is reached, i.e., no more values are removed.

In our example, arc-consistency removes D from the domain of V_3 yielding the following domains:

$$\underset{\{C,D,E\}}{V_1} \xrightarrow{K_1} \underset{\{C,D,E\}}{V_2} \xrightarrow{K_2} \underset{\{C,E\}}{V_3} \xrightarrow{K_3} \underset{\{D\}}{V_4}$$

Note that arc-consistency of Markov constraints as such solves the zero-frequency problem, regardless of control constraints: no choice made during the random walk can lead to a zero-frequency prefix.

In the case where arc-consistency detects the unsatisfiability of $\mathcal{P}$, i.e., the solution space S_C is empty, we can apply constraint relaxation techniques.

The next step is to extract the matrices from the domains.

4.3 Extraction of the Matrices

Recall that our goal is to generate a non-homogeneous Markov model, represented by a series of transition matrices. An intermediary series of L matrices $Z^{(0)}, \ldots, Z^{(L-1)}$ are obtained by zeroing, in the initial matrix, the elements that correspond to values or transitions that were removed during arc consistency. More precisely, the algorithm is:

- Inizialization:
  $Z^{(0)} \leftarrow M_0$ (the prior probabilities of M),
  $Z^{(i)} \leftarrow M, \forall i = 1, \ldots, L-1$ (the transitions).
- For each $a_k \in A$ removed from the domain of V_i:
  $Z^{(i)}_{j,k} \leftarrow 0, \forall j = 1, \ldots, n$ (set the k-th column to zero).
- All forbidden transitions in the binary constraints should also be removed from the matrices:
  $Z^{(i)}_{j,k} \leftarrow 0, \forall i, j, k$ such that $B_i(a_j, a_k) = false$.

The matrices obtained for our example are the following:

$$Z^{(0)} = \begin{pmatrix} 1/2 & 1/6 & 1/3 \end{pmatrix} Z^{(1)} = \begin{pmatrix} 1/2 & 1/4 & 1/4 \\ 1/2 & 0 & 1/2 \\ 1/2 & 1/4 & 1/4 \end{pmatrix}$$

$$Z^{(2)} = \begin{pmatrix} 1/2 & 0 & 1/4 \\ 1/2 & 0 & 1/2 \\ 1/2 & 0 & 1/4 \end{pmatrix} Z^{(3)} = \begin{pmatrix} 0 & 1/4 & 0 \\ 0 & 0 & 0 \\ 0 & 1/4 & 0 \end{pmatrix}$$

Modifying the initial matrix makes it non stochastic anymore. The next section describes how to renormalize the matrices to satisfy property (II).

4.4 Renormalization

We build the final transition matrices $\tilde{M}^{(i)}$ of $\tilde{M}$ from the intermediary matrices $Z^{(i)}$. Transition matrices could be renormalized *individually*, i.e., by dividing each row by its sum. This would indeed produce an non-homogeneous Markov model, but this model does not satisfy property (II) above, as it generates sequences with a different probability distribution than M.

For our example, the matrices after individual normalization are:

$$\tilde{Z}^{(0)} = \begin{pmatrix} 1/2 & 1/6 & 1/3 \end{pmatrix} \tilde{Z}^{(1)} = \begin{pmatrix} 1/2 & 1/4 & 1/4 \\ 1/2 & 0 & 1/2 \\ 1/2 & 1/4 & 1/4 \end{pmatrix}$$

$$\tilde{Z}^{(2)} = \begin{pmatrix} 2/3 & 0 & 1/3 \\ 1/2 & 0 & 1/2 \\ 2/3 & 0 & 1/3 \end{pmatrix} \tilde{Z}^{(3)} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

In this model, $p_{\tilde{Z}}(CCCD) = 1/6$ and $p_{\tilde{Z}}(CDCD) = 1/16$. The ratio between these 2 probabilities (16/6) is different from the original ratio (64/32=2).

The normalization process should indeed maintain the initial probability distribution. It turns out that a simple right-to-left process can precisely achieve that. The idea is to back propagate the perturbations in the matrices induced by individual normalization, starting from the right-most one.

To do this, we first normalize *individually* the last matrix $Z^{(L-1)}$. We then propagate the normalization from right to left, up to the prior vector $Z^{(0)}$. The elements of the matrices $\tilde{M}^{(i)}$ and the prior vector $\tilde{M}^{(0)}$ are given by the following recurrence relations:

$$\tilde{m}^{(L-1)}_{j,k} = \frac{z^{(L-1)}_{j,k}}{\alpha^{(L-1)}_j}, \quad \alpha^{(L-1)}_j = \sum_{k=1}^{n} z^{(L-1)}_{j,k}$$

$$\tilde{m}^{(i)}_{j,k} = \frac{\alpha^{(i+1)}_k z^{(i)}_{j,k}}{\alpha^{(i)}_j}, \quad \alpha^{(i)}_j = \sum_{k=1}^{n} \alpha^{(i+1)}_k z^{(i)}_{j,k} \quad 0 < i < L-1$$

$$\tilde{m}^{(0)}_{k} = \frac{\alpha^{(1)}_k z^{(0)}_{k}}{\alpha^{(0)}}, \quad \alpha^{(0)} = \sum_{k=1}^{n} \alpha^{(1)}_k z^{(0)}_{k}$$

By construction, when $\alpha^{(i)}_j = 0$, the j-th columns of the preceding $Z^{(i)}$ contain only 0 as well. By convention, the division yields 0 since there is no normalization to back propagate. These coefficients can be computed in $O(L \times n^2)$. We now show that this model satisfies the 2 desired properties.

Proposition: The $\tilde{M}^{(i)}$ are stochastic matrices and the non-homogeneous Markov process $\tilde{M}$ defined by the $\tilde{M}^{(i)}$ matrices and the prior vector $\tilde{M}^{(i)}$ satisfies (I) and (II).

Proof. The $\tilde{M}^{(i)}$ matrices are stochastic by construction, i.e., each row sums up to 1. The probability of a sequence $s = s_1 \ldots s_L$ to be generated by $\tilde{M}$ is:

$$\begin{aligned} p_{\tilde{M}}(s) &= p_{\tilde{M}}(s_1) \cdot p_{\tilde{M}}(s_2|s_1) \cdot \ldots \cdot p_{\tilde{M}}(s_L|s_{L-1}) \\ &= \tilde{m}^{(0)}_{k_1} \cdot \tilde{m}^{(1)}_{k_1,k_2} \cdot \ldots \cdot \tilde{m}^{(L-1)}_{k_{L-1},k_L} \\ &= \frac{1}{\alpha^{(0)}} \cdot z^{(0)}_{k_1} \cdot z^{(1)}_{k_1,k_2} \cdot \ldots \cdot z^{(L-1)}_{k_{L-1},k_L}, \end{aligned}$$

where k_i is the index of s_i in A. Hence, by construction of $Z^{(i)}$:

Melodies	Probabilities	Melodies	Probabilities
CCCD	12/77 (1/ 32)	EECD	4/77 (1/ 96)
CCED	6/77 (1/ 64)	EEED	2/77 (1/ 192)
CECD	6/77 (1/ 64)	EDCD	4/77 (1/ 96)
CEED	3/77 (1/ 128)	EDED	4/77 (1/ 96)
CDCD	6/77 (1/ 64)	DCCD	4/77 (1/ 96)
CDED	6/77 (1/ 64)	DCED	2/77 (1/ 192)
ECCD	8/77 (1/ 48)	DECD	4/77 (1/ 96)
ECED	4/77 (1/ 96)	DEED	2/77 (1/ 192)

Table 2: The probabilities of S_C sequences in $\tilde{M}$ and M (between parenthesis). The ratio of probabilities is constant: $\alpha^{(0)} = 77/384$.

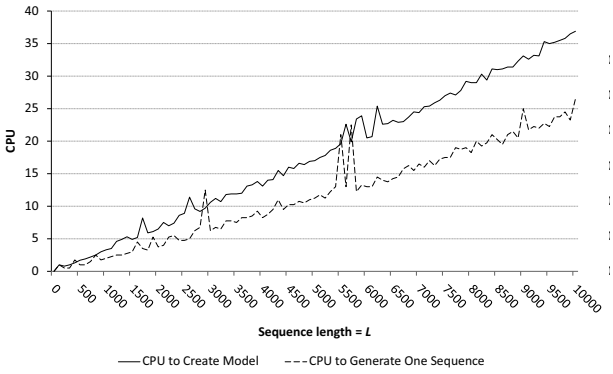

Figure 2: Performance of the algorithm as a function of the sequence length L; the computation time needed to create $\tilde{M}$ (solid line) and the computation time needed to generate one sequence (dashed line) both grow linearly with L.

(I) $p_{\tilde{M}} = 0$ for $s \notin S_C$,

(II) $p_{\tilde{M}} = 1/\alpha^{(0)} \cdot p_M(s)$ otherwise.

□

$\alpha^{(0)}$ is precisely the probability for sequences in M of satisfying the control constraints, i.e., $\alpha^{(0)} = p_M(S_C)$.

The final matrices for the running example are the following:

$$\tilde{M}^{(0)} = \begin{pmatrix} \frac{39}{77} & \frac{12}{77} & \frac{26}{77} \end{pmatrix} \tilde{M}^{(1)} = \begin{pmatrix} 6/13 & 4/13 & 3/13 \\ 1/2 & 0 & 1/2 \\ 6/13 & 4/13 & 3/13 \end{pmatrix}$$

$$\tilde{M}^{(2)} = \begin{pmatrix} 2/3 & 0 & 1/3 \\ 1/2 & 0 & 1/2 \\ 2/3 & 0 & 1/3 \end{pmatrix} \tilde{M}^{(3)} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

It is interesting to observe that even the addition of a simple unary constraint (here, last item = D) has an impact that propagates back to priors. In our example, $p_{\tilde{M}}(C)$ is slightly increased (from .5 to .506), $p_{\tilde{M}}(D)$ is decreased (from .1666 to .1558) and $p_{\tilde{M}}(E)$ increased (from .333 to .337). Table 2 shows the $\tilde{M}$ probabilities of all sequences in S_C. These probabilities are indeed equal to the initial probabilities, to the constant multiplicative factor $\alpha^{(0)}$.

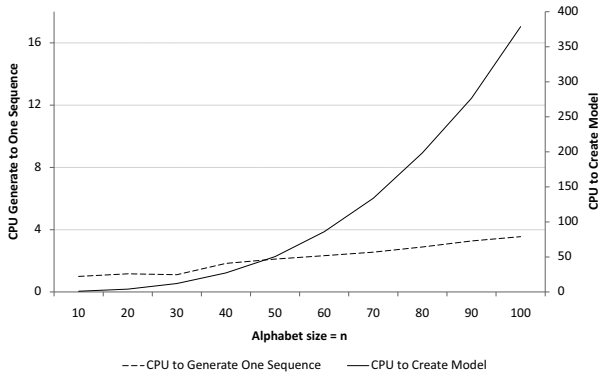

Figure 3: Performance of the algorithm as a function of the size of the alphabet n; the computation time needed to compute $\tilde{M}$ (solid line) is a quadratic function of n; the computation time needed to generate one sequence (dashed line) grows linearly with n.

4.5 Discussion

Our algorithm generates a model that satisfies the desired properties. Its complexity is low, as it involves only performing arc-consistency once on the induced CSP, and a renormalization in $O(L \times n^2)$. This theoretical worst-case complexity is confirmed experimentally, see Figure 2 and Figure 3.

As shown in our example, renormalization may have a small impact on the transition matrices, although a difficult one to quantify precisely. In practice, it may be interesting to sacrifice accuracy and skip the renormalization step (i.e., use individual normalization): in that case, a larger class of constraints can be handled. A necessary and sufficient condition is that the induced CSP should be made *backtrack-free*, to ensure that any choice performed during the random walk leads to a sequence in S_C. Property 2 of BSCs (see Section 4.1) is no longer true, so propagation is needed after each instantiation during the random walk to filter out values made incompatible with the current choice. Additionally, the algorithm is considerably simplified: transition matrices need not be constructed explicitly since their only use is to propagate the normalization. Instead, variable domains can be used directly to perform a random draw at each step of the random walk, with a prior individual normalization of the initial matrix M, i.e., arc-consistency does most of the job, which is to propagate the local effects of control constraints throughout the CSP.

As an illustration, let us consider our example in which the control constraint is changed to: "any three consecutive item should be different". This can be implemented by posting an *AllDiff* constraint [Régin, 1994] on all consecutive triplets of variables. Such a constraint exceeds the Markov scope (order 1 here) so the induced CSP is not a BSC. However, we can make the induced CSP backtrack-free by applying path-consistency (strong-3 consistency, see [Freuder, 1982]), with a complexity of $O(n^3)$. In our example, it turns out that the CSP is already path-consistent so no value is removed. S_C now contains 6 sequences, whose probabilities are shown in Table 3. The probabilities are not equivalent to those in M but

Melodies	Probabilities	Melodies	Probabilities
CDEC	1/4 (1/ 48) = 12:1	DECD	1/12 (1/ 48) = 4:1
CEDC	1/4 (1/ 96) = 24:1	ECDE	2/9 (1/ 48) = 10.6:1
DCED	1/12 (1/ 96) = 8:1	EDCE	1/9 (1/ 96) = 10.6:1

Table 3: Sequences satisfying a moving ternary AllDiff constraint with their probability using individual renormalization and their initial probabilities. The proportions of probabilities are not constant, i.e., sequences are not generated with the right probabilities.

this approximation is the price to pay to get a random walk that satisfy those ternary constraints.

We have described our solution for fixed order-1 Markov models. Generalization to order d consists in 1) introducing order-d Markov constraints (this step is straightforward), and 2) applying the rest of the method to $n^d \times n$ transition matrices (rows are d-grams and columns are the state) instead of the $n \times n$ transition matrices described here. In practice, most d-grams have no continuation, so sparse representations (graphs, oracles) are more appropriate than matrices.

In the case where proper normalization can be sacrificed, performing arc-consistency with an order-1 model allows to use variable-order random walk, provided the CSP is a BSC. This is achieved by adding to the basic order-1 procedure a phase consisting in removing higher order prefixes that contain a removed prefix of order 1. This is because all prefixes of length > 1 are also prefixes of length 1. This ensures that all solutions with longer prefixes will be found and only them. Contrarily to the approach of [Pachet and Roy, 2011] our approach scales-up well so composition applications can be envisaged. Once $\tilde{M}$ is built, the generation of a sequence does not require any computation other than a standard random walk algorithm whose complexity is a linear function of both the size of the alphabet n and the length of the sequence to generate L, i.e., $O(L \times n)$, as can be seen in Figure 2 and Figure 3 .

5 Applications

This method applies in general to style-imitation sequence generation requiring control constraints on individual items at arbitrary positions. A text generation application is currently developped, to generate text *in the style of X*, with additional constraints on rhymes, phrase structure and semantics. We describe below two musical applications.

5.1 Representing Musical Intentions

Unary constraints can be used to represent various musical intentions, when producing an melody from a Markov model M, and an input melody provided in real-time. For instance, we can define the following types of melodic output:

Continuation: input is continued to produce a sequence of the same size. A constraint is posted on the last note to ensure that it is "terminal", i.e., occurred at the end of an input melody, to produce a coherent ending.

Variation: is generated by adding two unary constraints that the first and last notes should be the same, respectively, as the first and last notes of the input.

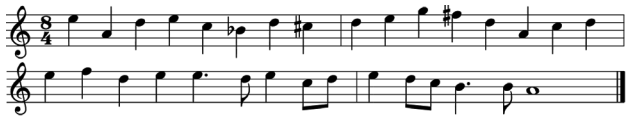

Figure 4: A melody taken as a training sequence.

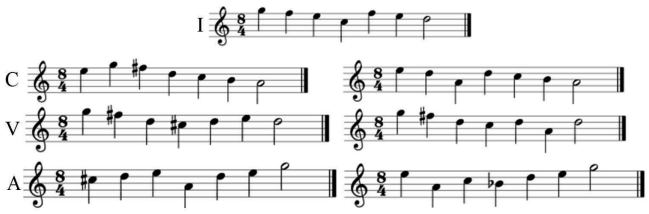

Figure 5: Examples of continuations (C), variations (V) and answers (A) generated from the input melody (I).

Answer: is like a *Continuation*, but the last note should be the same as the first input note. This creates a phrase that resolves to the beginning, producing a sense of closure.

For all types, unary constraints are posted on each intermediary note that they should *not* be initial nor final, to avoid *false starts/ends* within the melody.

Figure 4 shows a melody, taken from [Davismoon and Eccles, 2010] from which we build a Markov model M. Figure 5 shows examples of continuations, variations and answers, built from M and the constraints corresponding to each melody type, from an input (I). It is clear that these melodies belong to the corresponding musical categories. Continuations end naturally, with the same 3 notes as the input (a consequence of the constraint on the last note); variations sound similar to the input; and answers sound as responses to the input. This shows how unary constraints can have a global effect on the structure of the generated melodies.

5.2 Generating Virtuoso Melodies

Our approach trivially extends to non-homogeneous Markov models (M). In practice, this enables a simple representation of non-constant harmonic contexts. In the case of melodies spanning different tonalities, each transition between two notes belongs to the model of the corresponding tonality. As an example, let us consider the melody extracted from an improvisation performed by jazz guitarist John McLaughlin during a famous concert [GuitarTrio, 1977]. This melody consists of 43 notes played on a sequence of three chords (see Figure 6). The 8 first notes belong to the harmonic context of $Gmin7\flat5$, the 20 next to $C7\flat9$, and the remaining 15 to $Fmin$.

Negotiating such chord transitions fast is precisely one of the most sought after skills of virtuoso jazz improvisers. Our technique can easily produce such melodies in real-time. To do so, we consider a training set made of typical jazz scales, as used in jazz training. As a simple example we consider the scales described in [Leavitt, 2005]: so-called "staircase" melodies, in all tonalities (for each key and each chord type). Figure 8 shows the staircase melody for C minor.

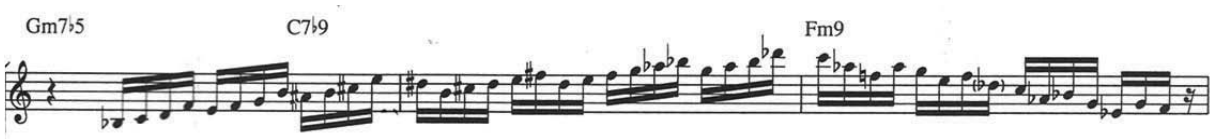

Figure 6: The original melody improvized by jazz guitarist John McLaughlin. The chords are written above the music: the first 8 notes are in $Gmin7\flat5$, the 20 next to $C7\flat9$, and the remaining 15 to $Fmin$.

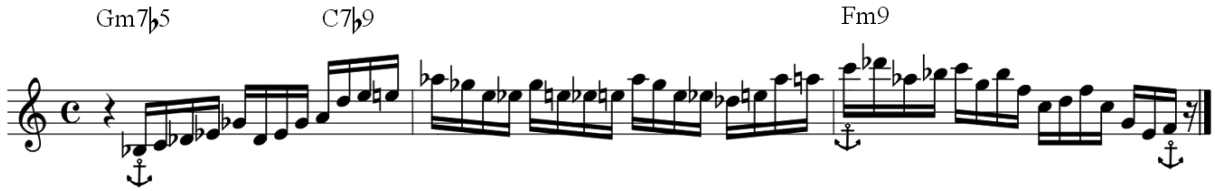

Figure 7: The melody generated by the system: the three notes marked with anchors are constrained to be the same as that of the original melody.

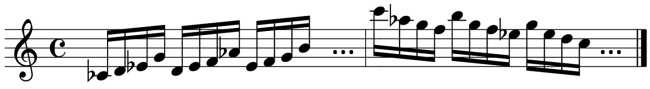

Figure 8: The training melodies consists in "staircase" scales, transposed in all tonalities. The figure shows parts of the $Cmin$ scale.

The sequence to generate can be seen as produced by a non-homogeneous Markov model, whose transition matrices change according the changes in tonality. To further illustrate our approach, we add three constraints stating that the first, middle, and last notes should be the ones played by John McLaughlin. Figure 7 shows one melody generated by the system. It can be observed that the melody is indeed 1) a concatenation of "staircase" chunks, 2) complies with the harmonic context, and 3) satisfies the three anchor constraints. Such melodies are generated in less than a millisecond.

This idea was used to implement an augmented instrument, called Virtuoso. With Virtuoso, the user plays bebop melodies by targeting specific notes ahead of time. These targets are selected using a gesture controller and transformed into unary constraints, e.g., on the last note. The underlying harmony is provided in real time by an mp3 file previously analyzed, from which time-lined harmonic metadata is extracted. Videos are available online (`www.youtube.com/watch?v=9f3caQNgxmI`, `/watch?v=GLUPcWT3tjc`, `/watch?v=LuvfWI7iMH4`).

6 Conclusion

We propose a general solution to the issue of generating finite-length Markov chains with control constraints. The solution exploits the fruitful connection between Markov processes and constraint satisfaction, initiated in [Pachet and Roy, 2011] but in a random walk setting, i.e., without search. We have shown that for constraints that remain within the Markov scope, arc-consistency enables us to solve the zero-frequency problem, as well as to compile control constraints in the form of a non-homogeneous Markov model. This model can in turn be used straightforwardly with random walk to generate sequences that satisfy the constraints with their original probabilities.

More complex constraints can be handled provided they can be filtered to obtain backtrack-free CSPs, at the cost of inaccurate probabilities. The approach generates all the sequences satisfying the constraints with no search, so is suitable for real-time, interactive applications.

Acknowledgments

Research reported in this paper was partly funded by the EU project MIROR and carried out at the Sony Computer Science Laboratory in Paris. We thank Simon Colton and Michael Spranger for their thoughtful remarks and suggestions.

References

[Addessi and Pachet, 2005] A.-R. Addessi and F. Pachet. Experiments with a musical machine: Musical style replication in 3/5 year old children. *British J. of Music Education*, 22(1):21–46, 2005.

[Bessière *et al.*, 1995] C. Bessière, E. C. Freuder, and J.-C. Régin. Using inference to reduce arc consistency computation. In *Proc. of the IJCAI'95*, pages 592–598. Morgan Kaufmann, 1995.

[Bessière *et al.*, 1999] C. Bessière, E. C. Freuder, and J.-C. Régin. Using constraint metaknowledge to reduce arc consistency computation. *Artificial Intelligence*, 107(1):125–148, 1999.

[Brooks *et al.*, 1992] F. P. Brooks, Jr., A. L. Hopkins, Jr., P. G. Neumann, and W. V. Wright. *An experiment in musical composition*, pages 23–40. MIT Press, Cambridge, MA, USA, 1992.

[Chordia *et al.*, 2010] P. Chordia, A. Sastry, T. Mallikarjuna, and A. Albin. Multiple viewpoints modeling of tabla se-

quences. In *Proc. of Int. Symp. on Music Information Retrieval*, pages 381–386, Utrecht, 2010.

[Cont *et al.*, 2007] A. Cont, S. Dubnov, and G. Assayag. Anticipatory model of musical style imitation using collaborative and competitive reinforcement learning. *Springer Verlag LNCS 4520*, pages 285–306, 2007.

[Davismoon and Eccles, 2010] S. Davismoon and J. Eccles. Combining musical constraints with Markov transition probabilities to improve the generation of creative musical structures. *EvoApplications*, 2:361–370, 2010.

[Dechter and Mateescu, 2004] Rina Dechter and Robert Mateescu. Mixtures of deterministic-probabilistic networks and their and/or search space. In *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI-04)*, pages 120–129, Arlington, Virginia, 2004. AUAI Press.

[Dubnov *et al.*, 2003] S. Dubnov, G. Assayag, O. Lartillot, and G. Bejerano. Using machine-learning methods for musical style modeling. *IEEE Computer*, 10(38), 2003.

[Eigenfield and Pasquier, 2010] A. Eigenfield and P. Pasquier. Realtime generation of harmonic progressions using controlled Markov selection. In *Proc. of 1st Int. Conf. on Computational Creativity*, pages 16–25, Lisbon, Portugal, 2010. ACM Press.

[Freuder, 1982] E. C. Freuder. A sufficient condition for backtrack-free search. *Journal of the Association for Computing Machinery*, 29(1):24–32, 1982.

[GuitarTrio, 1977] GuitarTrio. *Friday Night in San Francisco, choruses by Al Di Meola, John McLaughlin and Paco De Lucia*. Artist Transcriptions series. Hal Leonard, Milwaukee, USA, 1977.

[Kolarov and Sinai, 2007] L. Kolarov and Y.G. Sinai. *Theory of Probability and Random Processes*. Springer, 2007.

[Leavitt, 2005] W. Leavitt. *A Modern Method for Guitar*. Berklee Press, Boston, USA, 2005.

[Mackworth, 1977] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[Nierhaus, 2009] G. Nierhaus. *Algorithmic Composition, Paradigms of Automated Music Generation*. Springer-Verlag, 2009.

[Pachet and Roy, 2011] F. Pachet and P. Roy. Markov constraints: steerable generation of Markov sequences. *Constraints*, 16(2):148–172, 2011.

[Pachet, 2002] F. Pachet. The continuator: Musical interaction with style. In *Proceedings of ICMC*, pages 211–218, Goteborg, Sweden, 2002.

[Régin, 1994] J.-C. Régin. A filtering algorithm for constraints of difference in csps. In *Proc. of the 12th National Conf. on Artificial Intelligence*, pages 362–367, 1994.