Exact Sampling for Regular and Markov Constraints With Belief Propagation

Alexandre Papadopoulos[1,2], François Pachet[1,2], Pierre Roy[1], and Jason Sakellariou[1,2]

[1] Sony CSL, 6 rue Amyot, 75005, Paris, France
[2] Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, France
`alexandre.papadopoulos@lip6.fr`
`pachet@csl.sony.fr`
`roy@csl.sony.fr`
`jason.sakellariou@lip6.fr`

Abstract. Sampling random sequences from a statistical model, subject to hard constraints, is generally a difficult task. In this paper, we show that for Markov models and a set of Regular global constraints and unary constraints, we can perform perfect sampling. This is achieved by defining a factor graph, composed of binary factors that combine a Markov chain and an automaton. We apply a simplified version of belief propagation to sample random sequences satisfying the global constraints, with their correct probability. Since the factor graph is linear, this procedure is efficient and exact. We illustrate this approach to the generation of sequences of text or music, imitating the style of a corpus, and verifying validity constraints, such as syntax or meter.

Keywords: global constraints, unary constraints, Markov constraints, belief propagation, sampling

1 Introduction

Generating novel sequences, such as text or music, that imitate a given style is usually achieved by replicating statistical properties of a corpus. This inherently stochastic process can be typically performed by sampling a probability distribution. In practice, we often need to impose additional properties on sequences, such as syntactic patterns for text, or meter for music, that are conveniently stated using constraint satisfaction approaches. However, typical constraint satisfaction procedures are not concerned with the distribution of their solutions. On the other hand, traditional sampling algorithms are generally not suited to satisfy hard constraints, since they can suffer from high rejection rates or lack coverage of the solution space. Both issues can be avoided, in some cases, by taking advantage of constraint programming techniques.

In this paper, we show how to sample Markov sequences subject to a conjunction of Regular constraints [22], i.e., constraints stated with an automaton, as well as additional unary constraints. Regular grammars can express parts-of-speech patterns on text. In music, METER [26] constrains Markov temporal sequences to be metrically correct. METER can also be expressed as an automaton, as we explain further in this paper. We achieve this result by defining a tree-structured factor graph composed of unary and binary factors. The variables of this graphical model represent the elements of the sequence, and binary factors encode a type of conjunction between the Markov model and the automaton. We apply belief propagation to sample sequences with their right probability.

1.1 Related Work

The combination of statistical and logical methods has been an active research direction in artificial intelligence in the last few years. In constraint programming, stochastic techniques are often used for guiding search, but less for characterising solutions. Some work studies the impact of search heuristics on solution diversity [27], but such endeavours tend to use optimisation techniques [11, 13]. Conversely, introducing constraints to probabilistic graphical models is problematic since hard constraints introduce many zero probabilities, and this causes typical sampling algorithms to suffer from high rejection rates. To overcome such issues, Gogate and Dechter proposed SampleSearch [10], with a guaranteed uniform sampling of the solutions of a CSP, using a complete solver to reduce rejection rates. Likewise, Ermon et. al [8] use a constraint solver in a blackbox scheme, and sample the solution space uniformly, often with better performance. In SAT, Markov logic networks is a well established formalism that unifies probabilistic and deterministic properties [7, 25]. MC-SAT [24] samples from a non-uniform distribution of the satisfying assignments of a SAT formula. Such methods, with applications in verification, model checking, or counting problems, are general but expensive. The solution we propose, which focuses on a specific setting, is not derivable from such general methods, and is both tractable and exact.

2 Sequence Generation with Markov constraints

A Markov chain is a stochastic process, where the probability for state X_i, a random variable, depends only on the last state X_{i-1}. Each random variable X_i takes values amongst an *alphabet*, denoted $\mathcal{X}$. Seen as a generative process, a Markov chain produces sequence $X_1, \ldots, X_n$ with a probability $P(X_1) \cdot P(X_2|X_1) \cdots P(X_n|X_{n-1})$. Order k Markov chains have a longer memory: the Markov property states that $P(X_i|X_1, \ldots, X_{i-1}) = P(X_i|X_{i-k}, \ldots, X_{i-1})$. They are equivalent to order 1 Markov chains on an alphabet composed of k-grams, and therefore we assume only order 1 Markov chains.

Markov chains have been classically used for generating sequences that imitate a given style [5, 14, 23]. A Markov chain is trained by learning the transition probabilities on a corpus. For example, a musical piece can be represented as a sequence of complex objects, constituted of pitch, duration, metrical position, and more [17]. A Markov chain trained on this corpus will produce musical sequences in the style of the composer. With text, we can use a Markov chain whose alphabet is the set of words of the corpus, to generate new sentences in the style of its author.

Markov generation can be controlled using Markov constraints. This allows us to specify additional properties that a sequence should verify. For example, Meter [26] imposes that sequences of notes are metrically correct. Often, such constraints can be conveniently stated using a Regular constraint [22], defined with an automaton $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$, where Q is a set of states, Σ an alphabet defining labels on transitions, δ the transition function linking a state $q \in Q$ and a label $a \in \Sigma$ to the successor state $q' = \delta(q, a)$, $q_0 \in Q$ the initial state, and $F \subseteq Q$ the set of accepting states. In this case, we have $\Sigma = \mathcal{X}$, i.e. transitions are labelled using states of the Markov chain, so that the automaton recognises admissible Markov sequences. Combining Markov constraints with other constraints, we can restrict the solution space in any desirable way [15], but without any guarantee that the generated sequences will reflect the original distribution in any way. In the specific case of unary constraints, we can have this guarantee [2]. The result presented here can be seen as a further generalisation of this result to a set of Regular constraints. A specific implementation of this idea was used to generate non-plagiaristic sequences [18].

3 Background on Belief Propagation

Let $X_1, \dots, X_n$ be n discrete random variables, and let $p(X_1, \dots, X_n)$ be a distribution of the random sequence $X_1, \dots, X_n$. A graphical model [21] is a compact representation of p as the product of m factors holding on a subset of the variables, i.e. $p(X_1, \dots, X_n) = \prod_{j=1}^{m} f_j(S_j)$, where the factor f_j is a function holding on a subset $S_j \subseteq \{X_1, \dots, X_n\}$ of the variables. CSPs can be seen as graphical models, where solutions are uniformly distributed.

Belief propagation, specifically the sum-product algorithm [20] is an algorithm for performing statistical inference, based on a *factor graph* representation. A factor graph is a bipartite undirected graph $G = (X, F, E)$, representing the factorisation of a probability function. Nodes represent either variables or factors, and edges connect factors to the variables to which that factor applies: $X = \{X_1, \dots, X_n\}$, $F = \{f_1, \dots, f_m\}$, and an edge (X_i, f_j) is in E iff $X_i \in S_j$.

Example 1. Consider a probability function holding on three variables X_1, X_2, X_3, defined as the product of four factors $p(X_1, X_2, X_3) = f_1(X_1, X_2) \cdot f_2(X_2, X_3) \cdot f_3(X_1, X_3) \cdot f_4(X_3)$. The corresponding factor graph is shown on Figure 1.

The main use of factor graph in statistical inference is to compute marginals. Marginals are defined for each variable: $p_i(X_i) = \sum_{\{X_j | j \neq i\}} p(X_1, \dots, X_n)$. Once marginals have been computed, sampling can be performed easily. When the factor graph is a *tree*, computing marginals is polynomial. Tree factor graphs correspond to Berge-acyclic constraint networks, and such results generalise the well-known results in constraints [3, 6, 9].

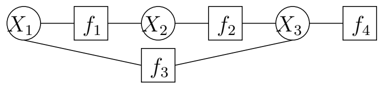


Fig. 1. The factor graph for the function $p(X_1, X_2, X_3) = f_1(X_1, X_2) \cdot f_2(X_2, X_3) \cdot f_3(X_1, X_3) \cdot f_4(X_3)$.

4 Belief Propagation for Markov and Regular

We apply those techniques to the problem of sampling constrained Markov sequences, and describe belief propagation in the case where we impose sequences to be recognised by an automaton $\mathcal{A}$, i.e. to belong to the language $\mathcal{L}(\mathcal{A})$ of words recognised by $\mathcal{A}$. This is equivalent to sampling the target distribution p_{target} defined as:

$$p_{target}(X_1, \ldots, X_n) \propto \begin{cases} P(X_2|X_1) \cdots P(X_n|X_{n-1}) \cdot & \text{if } X_1 \cdots X_n \in \mathcal{L}(\mathcal{A}) \\ \quad P_1(X_1) \cdots P_n(X_n) & \\ 0 & \text{otherwise} \end{cases}$$

We use the symbol $\propto$ to indicate that the equality holds after normalisation, so that p_{target} defines a probability function. $P(X_2|X_1) \cdots P(X_n|X_{n-1})$ gives the typical order 1 Markov probability of the sequences $X_1, \ldots, X_n$, provided it is accepted by the automaton. Additionally, we add unary constraints P_i, i.e. factors biasing each variable X_i individually. Implicitly, there is a big factor holding on the full sequence $X_1, \ldots, X_n$ taking value 1 when $X_1 \cdots X_n \in \mathcal{L}(\mathcal{A})$, and value 0 otherwise, corresponding to a hard global constraint. Consequently, the factor graph of p_{target} is not a tree.

We propose a reformulation of $p_{target}(X_1, \ldots, X_n)$ into a new function p_{reg} of $Y_1, \ldots, Y_n$, where the new Y_i variables take values $(a, q) \in \mathcal{X} \times Q$, where $a \in \mathcal{X}$ is a state of the Markov chain, and $q \in Q$ is a state of the automaton. Recall that transitions of the automaton are also labelled with elements of $\mathcal{X}$. This function p_{reg} is composed of simple binary factors, and its factor graph, which is tree structured, is shown on Figure 2.

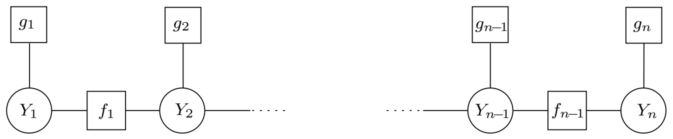


Fig. 2. The factor graph of the distribution on Markov sequences accepted by an automaton $\mathcal{A}$, defined by $p_{reg}(Y_1, \ldots, Y_n)$

We define a binary factor combining the Markov transition probabilities with the valid transitions from the automaton, as follows:

$$f((a,q),(a',q')) \propto \begin{cases} P(a'|a), & \text{if } q' = \delta(q,a'), \\ 0 & \text{otherwise} \end{cases}$$

This factor gives the probability for choosing, from state q, the transition labelled with a', which reaches q' (denoted by $q' = \delta(q,a')$). This probability depends on the label a of the transition that was used to reach q, and is given by the Markov transition probability from a to a'. This factor is applied along the sequence, i.e. $f_i = f, \forall 1 \leq i < n,$. The binary factors imply that non-zero probability sequences correspond to a walk in the automaton. Unary factors g_i additionally impose that such walks start from the initial state (enforced by g_1) and end at an accepting state (enforced by g_n), while taking into account the unary constraints of p_{target} (enforced by all g_i):

$$g_1((a,q)) \propto \begin{cases} P_1(a), & \text{if } q = \delta(q_0,a) \\ 0, & \text{otherwise.} \end{cases} \qquad g_n((a,q)) \propto \begin{cases} P_n(a), & \text{if } q \in F \\ 0, & \text{otherwise.} \end{cases}$$

Other unary factors are simply defined as $g_i((a,q)) \propto P_i(a)$.

Theorem 1. *Sampling p_{target} is equivalent to sampling p_{reg}, and projecting each resulting sequence $(a_1,q_1),\dots,(a_n,q_n)$ to $a_1,\dots,a_n$.*

Proof. We prove there is a one-to-one correspondence between non-zero probability sequences of p_{reg} and p_{target}, and that corresponding sequences have the same probability.

Let $(a_1,q_1),\dots,(a_n,q_n)$ be a sequence such that $p_{reg}((a_1,q_1),\dots,(a_n,q_n)) \geq 0$. This means that q_1 is the successor of the initial state q_0 for a_1 (from the definition of g_1), q_i is the successor of state q_{i-1} for a_i, for each $i > 1$ (from the definition of f), and q_n is an accepting state (from the definition of g_n). In other words, $a_1,\dots,a_n$ is accepted by the automaton, and, according to the definitions of the factors, with probability exactly equal to p_{target}.

Conversely, suppose that $a_1,\dots,a_n$ is a sequence with a non-zero p_{target} probability. Since $\mathcal{A}$ is deterministic, there exists a unique sequence of states $q_0,q_1,\dots,q_n$, with $q_n \in F$, that recognises $a_1,\dots,a_n$, and therefore a unique sequence $(a_1,q_1),\dots,(a_n,q_n)$ with a p_{reg} probability equal to $p_{target}(a_1,\dots,a_n)$.

In order to sample sequences from this factor graph, we adapt the general sum-product algorithm [21], and simplify it for the following reasons: the factor graph has no cycle (removing any issue for converging to a fixed point), the factor graph is almost a linear graph (induced by the sequence), factors are only unary and binary, and the procedure is used only for sampling individual sequences. This algorithm is shown on Algorithm 1 for self-containedness. It computes the backward messages $m_{i\leftarrow}$, the forward messages $m_{i\rightarrow}$, and the actual sequence $y_1,\dots,y_n$, all highlighted in blue in the algorithm. The exact justification of the algorithm is a well-established result [12, 20], and we only give an intuitive explanation. During the backward phase, $m_{i\leftarrow}$ contains the marginal of Y_i of the product of all factors of p_{reg} holding on $Y_i,\dots,Y_n$. This represents the impact on Y_i of the sub-factor graph "to the right" of Y_i, in the same way that arc-consistency guarantees that a value can be extended to a full instantiation. Eventually, $m_{1\leftarrow}$ is the marginal of Y_1 of all p_{reg}, and a value is drawn randomly according to this distribution. The full sequence is generated during the forward phase. At each iteration, $p_i(Y_i)$ is the marginal over Y_i of p_{reg} given the partial instantiation. In order to sample several sequences, the backward phase needs to be performed only once, and the forward phase will sample a new random sequence every time, with its correct probability. From a constraint programming point of view, computing the marginals at each step is a generalisation to random variables of computing arc-consistent domains. The time for sampling one sequence is bounded by $O(n \cdot (|\mathcal{X}||Q|)^2)$.

Algorithm 1: Sum-product algorithm for sampling Markov with Regular

```
Data: Function $p_{reg}(Y_1, \ldots, Y_n)$ and its factor graph
Result: A sequence $y_1, \ldots, y_n$, with probability $p_{reg}(y_1, \ldots, y_n)$
// Backward phase
$m_{n\leftarrow} \leftarrow g_n$
for $i \leftarrow n-1$ to 1 do
    foreach $y \in \mathcal{X} \times Q$ do
        $m_{i\leftarrow}(y) \leftarrow \sum_{y' \in \mathcal{X} \times Q} g_i(y) \cdot f_i(y, y') \cdot m_{i+1\leftarrow}(y')$
    Normalise $m_{i\leftarrow}$
// Forward phase
$p_1 \leftarrow m_{1\leftarrow}$
$y_1 \leftarrow$ Draw with probability $p_1(y_1)$
for $i \leftarrow 2$ to $n$ do
    foreach $y \in Q$ do
        $m_{i\rightarrow}(y) \leftarrow f_{i-1}(y_{i-1}, y)$
    Normalise $m_{i\rightarrow}$
    foreach $y \in \mathcal{X} \times Q$ do $p_i(y) \leftarrow m_{i\rightarrow}(y) \cdot m_{i\leftarrow}(y)$
    $y_i \leftarrow$ Draw with probability $p_i(y_i)$
return $(y_1, \ldots, y_n)$
```

5 Examples

If no automaton is imposed, our model, which imposes only unary constraints, is equivalent to the model in [16]. We compared the new model with our old model, and observed it behaves equivalently, with the benefit of an improved efficiency. We generated sequences of 16 notes with a Markov order 1 in the style of Bill Evans, with two unary constraints constraining the first and last note. Our old model could sample an average of 450 sequences per second, while our new model produces an average of 1200 sequences per second, almost three times more.

Automata can be handy for expressing patterns on text or music. For example, in music, a semiotic structure is a symbolic description of higher level patterns, from manual annotation or using pattern discovery techniques [4]. Semiotic structures can be easily stated using automata. In text, automata can be used to state syntactic rules over sequences of words.

METER [26], in its most basic form, imposes that a sequence of variables have a fixed total duration D, assuming each value has a specific duration, and assuming the existence of a special padding value with a null duration, which is used only at the end of the sequence. METER can also be encoded using REGULAR. We build an automaton $\langle Q, \mathcal{X}, \delta, q_0, F\rangle$ where each state represents a partial duration between 0 and D, i.e. $Q = \{q_0, \dots, q_D\}$. For every element $e \in \mathcal{X}$ of the Markov chain, we add a transition from q_{o_1} to q_{o_2} labelled by e iff $o_2 = o_1 + d(e)$, where $d(e)$ is the duration of e. Finally, we set $F = \{q_D\}$. This ensures that any accepting sequence will have a total duration of D exactly. By imposing this automaton to the Markov model, we can sample metrically correct Markov sequences with their correct probabilities. We tested this with a toy problem: produce sequences of a variable number of words, but with fixed number of syllables equal to 36, i.e. the duration of a word is its number of syllables. We are able to sample around in average 1100 sequences per second, against 230 sequences per second produced by a CP model with a single METER constraint, almost five times more.

In previous work, we introduced MAXORDER, which limits the maximum order of generated sequences, i.e. the length of exact copies made from the input corpus [19]. This constraint was filtered by computing a particular automaton and propagating it using REGULAR. We can use this automaton with the model of this paper, in order to sample Markov sequences with a maximum order guarantee. Furthermore, by computing the intersection between the METER and the MAXORDER automaton, we can also impose meter on such sequences.

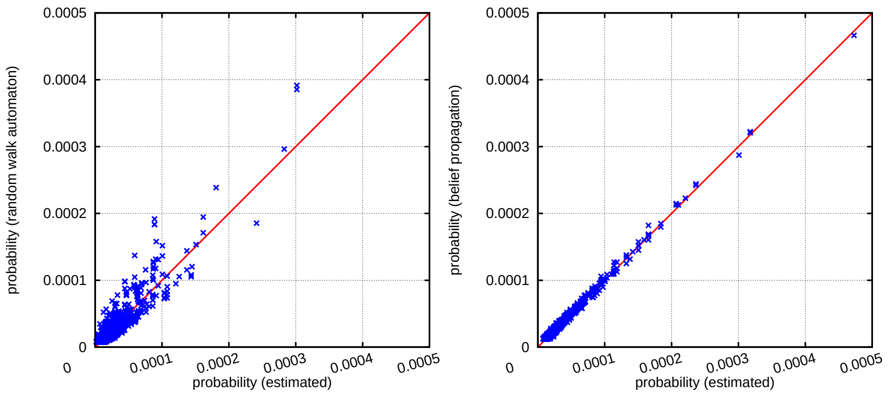
6 Evaluation

We compare our fixed-length belief propagation model with a random walk in the automaton. The purpose of this experiment is to show that a random walk in the automaton does not sample sequences correctly, and confirm empirically that our belief propagation-based model is correct, with a limited time penalty.

We used each method to sample sequences of words based on Pushkin's *Eugene Onegin*, of length 8, of Markov order 1 and with a max order less than 4, imposed using a max order automaton [19]. We implemented our experiments in Oracle Java 7, and ran them on an iMac with a 3.4GHz Intel Core i7 CPU, and 16GB RAM. The automaton was computed in about 200ms. For the random walk method, we imposed the length by rejecting shorter sequences. In total, we sampled over 20 million sequences. Of those, 5 million were unique sequences. The baseline random walk procedure generated an average of 5500 sequences per second (counting only non-rejected sequences), while the belief propagation-based method generated an average of 3500 sequences per second. For comparison, our

Regular-based CP model produced only about 50 sequences per second. We filtered those that were generated over 50 times, of which there were about 47000 with random walk, and about 35000 with belief propagation. We estimated the probability of a sequence by computing the sum of the probability of all unique sequences found by either method, and use this for normalising.

We plot our results on Figure 3. Each point on either graph corresponds to a sequence. Its value on the x-axis is its probability, estimated as described previously, while the values on the y-axis is the empirical probability, i.e. the frequency at which the specific sequence has been sampled compared to the total number of sequences. Figure 3(a) shows that the baseline sampling approach performs poorly: many sequences, even of similar probability, are over or under-represented. On the other hand, Figure 3(b) provides a striking empirical confirmation of the correctness of the belief propagation model.



(a) Random walk in the automaton (b) Belief propagation

Fig. 3. Sampling with random walk in the automaton compared to belief propagation.

7 Conclusion

We defined a belief propagation model for sampling Markov sequences that are accepted by a given automaton. To this aim, we introduced a tree-structured factor graph, on which belief propagation is polynomial and exact. This factor graph uses binary factors, which encode a type of conjunction between the underlying Markov model and the given automaton. We showed that this procedure allows us to sample sequences faster than equivalent CP models, and demonstrated that such sequences are sampled with their exact probabilities.

This result can be used for sequence generation problems in which users want a set of solutions that are both probable in a given statistical model, and satisfy hard regular constraints. More generally, we believe that this approach offers an interesting bridge between statistical inference and constraint satisfaction.

Acknowledgments. This research is conducted within the Flow Machines project funded by the ERC under the European Unions 7th Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 291156. We thank Ricardo de Aldama, funded by the Lrn2Cre8 project (FET grant agreement no. 610859) for discussions about sampling and CSP.

References

1. Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA. AAAI Press (2006)
2. Barbieri, G., Pachet, F., Roy, P., Esposti, M.D.: Markov constraints for generating lyrics with style. In: Raedt, L.D., Bessière, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P.J.F. (eds.) ECAI. Frontiers in Artificial Intelligence and Applications, vol. 242, pp. 115–120. IOS Press (2012)
3. Beeri, C., Fagin, R., Maier, D., Yannakakis, M.: On the Desirability of Acyclic Database Schemes. J. ACM 30(3), 479–513 (1983)
4. Bimbot, F., Deruty, E., Sargent, G., Vincent, E.: Semiotic structure labeling of music pieces: Concepts, methods and annotation conventions. In: Gouyon, F., Herrera, P., Martins, L.G., Müller, M. (eds.) Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012. pp. 235–240. FEUP Edições (2012), http://ismir2012.ismir.net/event/papers/235-ismir-2012.pdf
5. Brooks, F.P., Hopkins, A., Neumann, P.G., Wright, W.: An experiment in musical composition. Electronic Computers, IRE Transactions on 6(3), 175–182 (1957)
6. Dechter, R., Pearl, J.: Tree Clustering for Constraint Networks. Artif. Intell. 38(3), 353–366 (1989)
7. Domingos, P.M., Kok, S., Poon, H., Richardson, M., Singla, P.: Unifying logical and statistical AI. In: Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA [1], pp. 2–9, http://www.aaai.org/Library/AAAI/2006/aaai06-001.php
8. Ermon, S., Gomes, C.P., Selman, B.: Uniform solution sampling using a constraint solver as an oracle. In: de Freitas, N., Murphy, K.P. (eds.) Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, August 14-18, 2012. pp. 255–264. AUAI Press (2012), https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=2288&proceeding_id=28
9. Freuder, E.C.: A Sufficient Condition for Backtrack-Free Search. J. ACM 29(1), 24–32 (1982)
10. Gogate, V., Dechter, R.: Studies in solution sampling. In: Fox, D., Gomes, C.P. (eds.) Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008. pp. 271–276. AAAI Press (2008), http://www.aaai.org/Library/AAAI/2008/aaai08-043.php
11. Hebrard, E., Hnich, B., O'Sullivan, B., Walsh, T.: Finding diverse and similar solutions in constraint programming. In: Veloso, M.M., Kambhampati, S. (eds.) Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA. pp. 372–377. AAAI Press / The MIT Press (2005), http://www.aaai.org/Library/AAAI/2005/aaai05-059.php

12. Mezard, M., Montanari, A.: Information, physics, and computation. Oxford University Press (2009)
13. Nadel, A.: Generating diverse solutions in SAT. In: Sakallah, K.A., Simon, L. (eds.) Theory and Applications of Satisfiability Testing - SAT 2011 - 14th International Conference, SAT 2011, Ann Arbor, MI, USA, June 19-22, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6695, pp. 287–301. Springer (2011), `http://dx.doi.org/10.1007/978-3-642-21581-0_23`
14. Nierhaus, G.: Algorithmic composition: paradigms of automated music generation. Springer (2009)
15. Pachet, F., Roy, P.: Markov constraints: steerable generation of markov sequences. Constraints 16(2), 148–172 (2011)
16. Pachet, F., Roy, P., Barbieri, G.: Finite-length markov processes with constraints. In: Walsh, T. (ed.) IJCAI. pp. 635–642. IJCAI/AAAI (2011)
17. Pachet, F., Roy, P.: Imitative leadsheet generation with user constraints. In: Schaub, T., Friedrich, G., O'Sullivan, B. (eds.) ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014). Frontiers in Artificial Intelligence and Applications, vol. 263, pp. 1077–1078. IOS Press (2014), `http://dx.doi.org/10.3233/978-1-61499-419-0-1077`
18. Papadopoulos, A., Pachet, F., Roy, P.: Generating Non-plagiaristic Markov Sequences With Max Order Sampling. In: Degli Esposti, M., Altmann, E., Pachet, F. (eds.) Universality and Creativity in Language (forthcoming). Lecture Notes in Morphogenesis, Springer (2015)
19. Papadopoulos, A., Roy, P., Pachet, F.: Avoiding plagiarism in markov sequence generation. In: Brodley, C.E., Stone, P. (eds.) Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada. pp. 2731–2737. AAAI Press (2014), `http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8574`
20. Pearl, J.: Reverend bayes on inference engines: A distributed hierarchical approach. In: Waltz, D.L. (ed.) Proceedings of the National Conference on Artificial Intelligence. Pittsburgh, PA, August 18-20, 1982. pp. 133–136. AAAI Press (1982), `http://www.aaai.org/Library/AAAI/1982/aaai82-032.php`
21. Pearl, J.: Probabilistic reasoning in intelligent systems – networks of plausible inference. Morgan Kaufmann series in representation and reasoning, Morgan Kaufmann (1989)
22. Pesant, G.: A regular language membership constraint for finite sequences of variables. In: Wallace, M. (ed.) Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3258, pp. 482–495. Springer (2004), `http://dx.doi.org/10.1007/978-3-540-30201-8_36`
23. Pinkerton, R.C.: Information theory and melody. Scientific American (1956)
24. Poon, H., Domingos, P.M.: Sound and efficient inference with probabilistic and deterministic dependencies. In: Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA [1], pp. 458–463, `http://www.aaai.org/Library/AAAI/2006/aaai06-073.php`
25. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62(1-2), 107–136 (2006), `http://dx.doi.org/10.1007/s10994-006-5833-1`
26. Roy, P., Pachet, F.: Enforcing meter in finite-length markov sequences. In: desJardins, M., Littman, M.L. (eds.) Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA.

AAAI Press (2013), http://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/view/6422

27. Schreiber, Y.: Value-ordering heuristics: Search performance vs. solution diversity. In: Cohen, D. (ed.) Principles and Practice of Constraint Programming - CP 2010 - 16th International Conference, CP 2010, St. Andrews, Scotland, UK, September 6-10, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6308, pp. 429–444. Springer (2010), http://dx.doi.org/10.1007/978-3-642-15396-9_35