Generating $1/f$ Noise Sequences as Constraint Satisfaction: The Voss Constraint

François Pachet[1,2] and **Pierre Roy**[1] and **Alexandre Papadopoulos**[1,2] and **Jason Sakellariou**[1,2]
[1]SONY CSL, 6 rue Amyot, 75005 Paris
[2]Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France
pachetcsl@gmail.com, roy@csl.sony.fr, alexandre.papadopoulos@lip6.fr, jason.sakellariou@lip6.fr

Abstract

Many natural phenomena exhibit power law spectra. In particular, so-called $1/f^{\alpha}$ noise series with α close to 1 (also called pink noise) occur in sound, music and countless human artifacts or natural events, from the fluctuations of the flood levels of the Nile to movements of the stock market. As a consequence, many generative models for $1/f$ noise have been designed to produce series that look or sound "natural" or "human". In this paper, we formulate the generation of $1/f$ series as a hard constraint satisfaction problem, so that $1/f$ noise generation can be used as an add-on to arbitrary sequence generation problems. We take inspiration from a simple yet beautiful stochastic algorithm invented by Voss and introduce the *Voss* constraint. We show that Voss' algorithm can be modeled as a tree of ternary sum constraints, leading to efficient filtering. We illustrate our constraint with a melody generation problem, and show that the addition of the Voss constraint tends indeed to produce sequences whose spectrum have a $1/f$ distribution, regardless of the other constraints of the problem. We discuss the advantages and limitations of this approach and possible extensions.

1 $1/f$ noise in nature and human artefacts

Many natural phenomena have been shown to exhibit so-called $1/f$ fluctuations, such as fluctuations of river flooding or luminosity of stars [Mandelbrot, 1982]. Such fluctuations have also been observed in human artefacts, such as fluctuations in rhythm [Hennig *et al.*, 2011], or music [Voss and Clarke, 1975]. $1/f$ fluctuations have also been shown to be ubiquitous in human cognition (see, e.g., [Farrell *et al.*, 2006a; 2006b]).

$1/f$ noise, also called pink noise, is defined by a power law relation between the frequency f of a signal and its power spectral density $S(f)$:

$$S(f) \propto 1/f^{\alpha}, \text{ with } \alpha \approx 1$$

The ubiquity of $1/f$ fluctuations has therefore been used to enhance artificial artefacts, to make them look or sound more natural or more human. For instance, [Hennig *et al.*, 2011] introduce $1/f$ fluctuations in computer-generated musical rhythms to make them sound less artificial, and show experimentally that these subtle differences are indeed perceived and preferred by humans. Similarly, $1/f$ noise is often used to make images or textures more natural, see e.g., Perlin noise [Perlin, 1985]. Many algorithms have been designed to generate pink noise series (see, e.g., [Kasdin, 1995]), but these algorithms are always designed as self-contained black boxes. As a consequence, it is difficult to generate $1/f$ series that satisfy simultaneously additional properties.

Index	Blue	Green	Red	Tossing
0	0	0	0	All dice are tossed
1	0	0	1	Red is tossed
2	0	1	0	Green and Red are tossed
3	0	1	1	Red is tossed
4	1	0	0	All dice are tossed
5	1	0	1	Red is tossed
6	1	1	0	Green and Red are tossed
7	1	1	1	Red is tossed

Table 1: Voss' dice tossing scheme for producing $1/f$ series. At each row, we produce a random number between 3 and 18 by summing the three 6-sided dice.

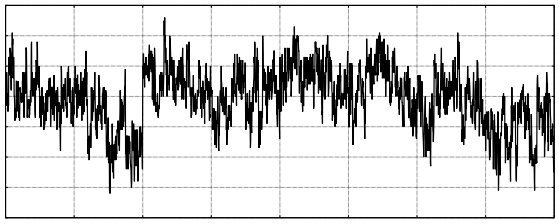

Figure 1: A $1/f$ series generated by Voss' algorithm.

The striking result obtained by [Voss and Clarke, 1978] is that $1/f$ pitch sequences sound more natural than sequences obtained with other distributions (random, Brownian). However, most music examples of $1/f$ sequences exhibited in the literature do not satisfy other basic properties of music such as meter, harmony, or higher-level structure. For instance, the melodies generated in [Voss and Clarke, 1978] do not contain bars, i.e., are not metrically correct: $1/f$ noise is a global property that is not easily combined with other properties. In that case, generating sequences that fulfill metrical properties

is in itself a difficult combinatorial problem [Roy and Pachet, 2013]. In the context of sequence generation, it is therefore natural to look for an $1/f$ *global constraint*, which can be added to other arbitrary constraints, in the framework of constraint satisfaction. The goal of this paper is to propose a simple way to do so in the context of hard constraints, by taking inspiration from an algorithm proposed by physicist Richard Voss and popularized by Martin Gardner in [Gardner, 1978]. Like Voss' algorithm, the scheme we propose does not solve the problem in its full generality, but provides a simple and elegant solution to the issue of $1/f$ sequence generation. Before describing this algorithm and its formulation as a constraint satisfaction problem, we review the state-of-the-art in global constraints dealing with distributions.

Algorithm 1: Voss' algorithm as described by Gardner.

Input: N a sequence length, $nbDice$ a number of dice, $maxDice$ the maximum dice value
Output: A sequence of N values in $[nbDice, nbDice.maxDice]$ with a $1/f$ spectrum

$result \leftarrow$ new integer array of length N
$diceValues \leftarrow$ new integer array of length $nbDice$
for $i = 0, \ldots, N-1$ **do**
  for $d = 1, \ldots, nbDice$ **do**
    if (d^{th} *bit of* i) $\neq$ (d^{th} *bit of* $i-1$) **then**
      `// toss die number d`
      $diceValues(d) \leftarrow$ Random integer in $[1, maxDice]$
  `// sum all dice`
  $result(i) \leftarrow \sum_{k=1}^{nbDice} diceValues(k)$
return $result$

2 Related work

The *spread* constraint [Pesant and Régin, 2005] ensures that the values in a sequence have a given mean and standard deviation. Similarly, the *deviation* constraint [Schaus *et al.*, 2007] ensures that values of a set of variables have a given standard deviation. The *balance* constraint [Beldiceanu *et al.*, 2007] bounds the difference in the number of occurrences of values assigned to variables. No constraint, to our knowledge, deals with more complex distribution of values, in particular taking into account the spectrum of the sequence (seen as a time series). Indeed, the spectrum of a time series is obtained through a complex operation (the Fourier transform), and, to our knowledge, no *Spectrum* constraint has yet been proposed. Recently, the idea of encoding statistical systems as constraint optimization problems has emerged, notably for Markov chains [Morin and Quimper, 2014; Pachet and Roy, 2011] and neural networks [Lombardi and Gualandi, 2013; Bartolini *et al.*, 2011]. This work can be seen as yet another bridge between stochastic algorithms and finite-domain, discrete constraint satisfaction.

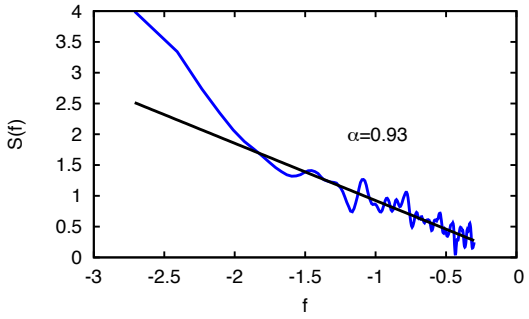

Figure 2: The spectrum of a time series generated with Voss' algorithm is in $1/f$.

3 Voss' algorithm

In our case we do not need to control the spectrum of a sequence precisely but only to enforce a specific, global property of the spectrum. The seminal paper [Gardner, 1978] described a simple algorithm to generate $1/f$ sequences, i.e., sequences of integers having a spectral density in $1/f$. This algorithm, invented by Voss but never published to our knowledge, consists in using a number of dice and summing their values, exploiting the fact that $1/f$ series can be modeled as sums of random variables with specific correlations. At each step, only certain dice are tossed: those who correspond to a bit change when writing a sequence of integers in base 2. As a consequence, it can be seen easily that in the resulting sequence the variance is inversely proportional to the frequency, due to the fact that some dice stay unchanged over longer periods than others. The algorithm is described in Algorithm 1 and the dice tossing scheme is illustrated in Table 1. Three 6-sided dice are considered, represented by 3 colors (Blue, Green, Red) to generate a sequence of 8 values. Each line represents an integer, from 0 to 7 written in base 2. The numbers of the sequence are obtained by summing up the values of the three dice at each line. The trick is that only dice corresponding to a bit change with respect to the previous line are tossed.

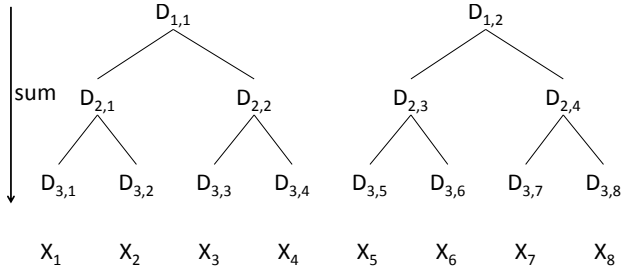

Figure 3: A tree of n-ary sum constraints implements the Voss constraint, here for a sequence length of 8 and 3 dice. Variable $D_{i,j}$ represents the j^{th} toss of die i.

Voss' algorithm generates $1/f$ series, or at least a very good approximation thereof. We can show it by estimating the power spectral density using $S(f) = 1/T\,|X(f,T)|^2$, where $X(f,T) = \int_0^T x(t)e^{-ift}dt$ is the Fourier transform

of the sequence. We then perform a simple linear regression in the log-log spectrum in order to obtain the slope $-\alpha$. Such a series is illustrated in Figure 1, and its log-log spectrum in Figure 2, which is well approximated by a straight line, at least for values larger than a threshold (-2.5).

4 The Voss constraint

Voss' algorithm is a statistical one, and cannot be, as such, translated easily into a hard constraint satisfaction problem. However, the decomposition of $1/f$ series as sums of random variables can lead to a simple representation as a set of constraints, in combination with a random heuristics for variable value choice. The idea is to consider both sequence elements and dice as constrained variables and relate them through a set of *sum* constraints [Trick, 2003].

The Voss constraint can be posted on sequences of size $N = 2^K$, whose elements have a domain of the form $D = [K, K.R]$, with R a positive integer. The Voss constraint ensures that the values of the variables could have been obtained by tossing dice according to Voss' algorithm presented above, with parameters N, K for $nbDice$ and R for $maxDice$.

We first present a naive implementation of the Voss constraint, and then a more efficient one.

4.1 A naive CSP

The Voss constraint can be implemented naively by introducing a die variable representing each die toss. Die variable $D_{i,j}$ represents the j^{th} toss of die i (first index is die number and second one is toss index). We then relate the sequence variables to sums of these die variables, according to Voss' algorithm.

More precisely, we introduce the Voss tree that represents the structure of the die variables as follows.

Definition 1 ((K, R)-Voss tree). *For a given integer K and R, the (K, R)-Voss tree is a binary tree of variables $D_{i,j}$, with $i = 1, \ldots, K$ and $j = 1, \ldots, 2^i$. Each $D_{i,j}$ has two children $D_{i+1,2j-1}$ and $D_{i+1,2j}$. The domain of each $D_{i,j}$ is $\{1, \ldots, R\}$. K is called the order of the tree, R is its range.*

The total number of variables in a (K, R)-Voss tree is

$$\sum_{i=1}^{K} 2^i = 2^{K+1} - 2 \tag{1}$$

which is less than 2×2^K.

For each sequence variable X_i we introduce a sum constraint equating X_i with the sum of the die variables obtained by walking up the Voss tree from the leaf $D_{k,i}$:

$$X_i = \sum_{l=0}^{K-1} D_{K-l,\lceil i/2^l \rceil} \tag{2}$$

Figure 3 shows the Voss tree for a sequence of length 8, thus $K = 3$. In that case, the number of die variables is 14, and we post the 8 constraints shown in Figure 4.

For a sequence of length $N = 2^K$ the Voss tree contains $2N - 2$ extra die variables (Equation 1).

$$\begin{aligned} X_1 &= D_{1,1} + D_{2,1} + D_{3,1}, \\ X_2 &= D_{1,1} + D_{2,1} + D_{3,2}, \\ &\ldots \\ X_8 &= D_{1,2} + D_{2,4} + D_{3,8}. \end{aligned}$$

Figure 4: The n-ary sum equations defining the Voss constraint.

Definition 2 ((K, R)-Voss constraint). *Let $X_1, \ldots, X_{2^K}$ be a sequence of variables, each with finite domain $D = [K, K.R]$. The Voss constraint is the conjunction of all sum constraints of the (K, R)-Voss tree defined by Equation 2.*

Filtering the Voss constraint can be achieved by filtering each sum constraint individually. However, domain-consistency of sum constraints is pseudo-polynomial [Trick, 2003]. Additionally, domain-consistency for the whole Voss constraint is not guaranteed by domain consistency of individual sum constraints.

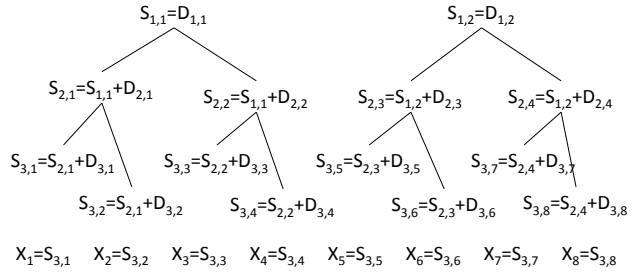

Figure 5: A tree of ternary sum constraints produces an acyclic CSP equivalent to the one in Figure 3. Variable $S_{i,j}$ represents a partial sum of the dice from the top die down to $D_{i,j}$.

4.2 An efficient filtering for the Voss constraint

A much more efficient filtering method for the Voss constraint is obtained by replacing each n-ary sum constraint by a set of ternary constraints representing *partial sums*. More precisely, for each $D_{i,j}$ we introduce a variable $S_{i,j}$ representing the partial sum from $D_{i,j}$ to the top of the Voss tree. In order to avoid computing n-ary sums, $S_{i,j}$ are defined recursively with ternary sum constraints as follows:

$$S_{i,j} = D_{i,j} + S_{i-1,\lceil j/2 \rceil} \tag{3}$$

The sequence variables are given by $X_i = S_{K,i}$.

The number of sum variables ($S_{i,j}$) is equal to the number of die variables so the total number of extra variables ($D_{i,j}$ and $S_{i,j}$) is $2^{K+2} - 4$, also linear in the size of the sequence. The number of ternary sum constraints is $2^{K+1} - 2$. The ternary constraints corresponding to our running example (length 8, 3 dice) are illustrated in Figure 5 and Figure 6.

By construction, the resulting CSP is Berge-acyclic, i.e., the network is a tree and the constraints overlap on, at most, one variable. Consequently, domain consistency for the whole constraint is achieved by propagating each ternary sum

$$
\begin{aligned}
S_{1,1} &= D_{1,1}, S_{1,2} = D_{1,2}, \\
S_{2,1} &= S_{1,1} + D_{2,1}, \\
S_{2,2} &= S_{1,1} + D_{2,2}, \\
&\dots \\
S_{3,1} &= S_{2,1} + D_{3,1}, \\
S_{3,2} &= S_{2,1} + D_{3,2}, \\
&\dots \\
S_{3,8} &= S_{2,4} + D_{3,8}
\end{aligned}
$$

Figure 6: The ternary sum equations defining the Voss constraint.

constraint individually [Beeri *et al.*, 1983; Jégou, 1993]. If sequence variables have a domain size d, each additional variable $S_{i,j}$ has a domain bounded by d. The complexity of filtering one such ternary constraint is in $O(d^2)$. Therefore, for a sequence of length N, the complexity of achieving domain-consistency for the Voss constraint is in $O(N.d^2)$.

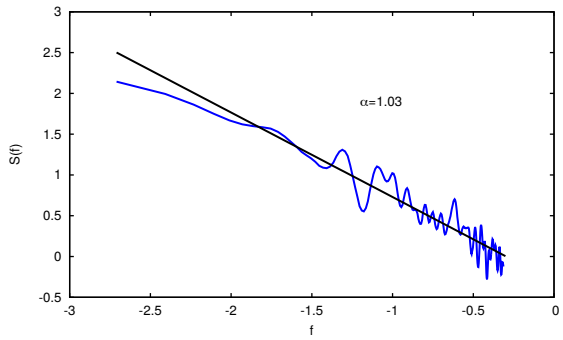

Figure 7: The spectrum of a sequence generated from a CSP with the Voss constraint only exhibits a very similar distribution as the one obtained with the original algorithm in Figure 2.

4.3 Generalization

The constraint can be easily generalized in two ways:

- Arbitrary sequence length. The running example focuses on the case where the length is a power of 2. The tree structure proposed can easily accommodate arbitrary sequence length. The resulting tree is incomplete but the incidence on the spectrum is negligible. For a given sequence length l, the number of dice to consider is $nbDice = \lceil \log_2(l) \rceil$.
- Arbitrary domains sizes. The range of dice $maxDice$ or R is obviously related to the domain size, since values are by construction in $[N, N * R]$. For a contiguous domain $[min, max]$, R is to be chosen as $[\lfloor min/nbDice \rfloor, \lceil max/nbDice \rceil]$.

5 A Melody Generation Example

We revisit the classical melody generation example introduced by Gardner by generating again melodies. Our goal is not to show the effect of $1/f$ sequences on the perception of melodies, which has already been done convincingly in [Voss and Clarke, 1978]. Our goal is to show that 1) the Voss constraint does generate sequences with a $1/f$ spectral density, like the original Voss algorithm, and 2) that this constraint, when added to a CSP with another constraint, here a GCC, still biases the spectrum of the resulting sequence to be in $1/f$.

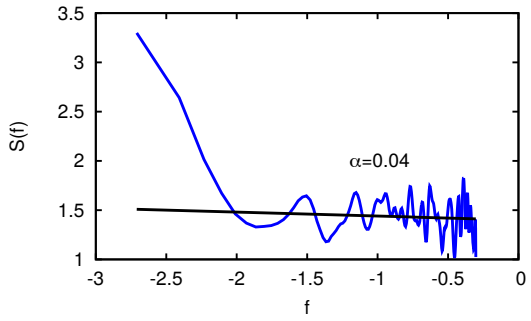

Figure 8: The spectrum of a sequence generated from a CSP with a GCC constraint only. Slope is about 0, far from 1.

5.1 Implementation

We implemented the Voss constraint in BackJava, an in-house Java finite-domain constraint solver similar in nature to Choco [choco Team, 2010], as well as the MusES musical object library [Pachet, 1994]. All the experiments ran on a machine with a Core i7, 2.3 GHz CPU, with 16GB RAM, and running an Oracle Java 7 VM under Windows 8.

5.2 Generating Long Melodies

We consider a melody generation problem defined by three sets of constraints. We generate sequences of integers (representing the pitch of musical notes) of length 512, so that the spectrum can be computed without side effects. The range of pitches (i.e., the domain D of sequence variables) is $[0, 16]$, i.e., covers roughly 2 octaves in a diatonic setting. The number of dice is 9 ($= \log_2(512)$), and their range is $[0, 1, 2]$. We consider three cases:

1. A Voss constraint only holding on the whole sequence,
2. A global cardinality constraint (GCC) [Régin, 1996] only. The GCC ensures that there is a balanced distribution of values in the resulting sequence. We post the GCC as follows: We consider N_{opt} the optimum distribution, that would ensure that all values are equally represented: $N_{opt} = N/|D|$. We enforce a GCC forcing each value of D to occur exactly this optimum value,
3. A Voss constraint and the GCC constraint.

For each case we compute the log-log spectrum and estimate the slope of curve as previously. The spectrum of case #1 (Voss constraint only, see Figure 11) is shown in Figure 7. It can be seen clearly that the spectrum is in $1/f$. For case #2 (GCC only, Figure 12) the spectrum (Figure 8) is clearly not in $1/f$. Case #3 (same GCC plus the Voss constraint, see Figure 13) shows that the spectrum (Figure 9) is again in $1/f$ thanks to the Voss constraint. The distribution of the slopes

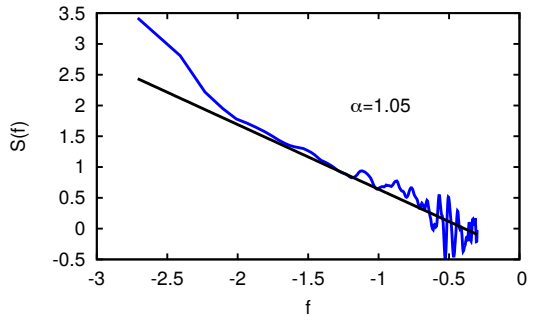

Figure 9: The spectrum of a sequence generated from a CSP with a GCC constraint and a Voss constraint. The α coefficient is clearly much closer to 1 than with the GCC only.

of the various spectra for 100 solutions is shown in Figure 10 and confirms that the presence of the Voss constraint does bias the spectrum toward an $1/f$ distribution. Running times and number of backtracks, averaged for one solution, are reported for each setup on Table 2. It can be seen that adding a Voss constraint increases the cost of finding solutions in a reasonable way.

setup	running times (sec)	backtrack count
Case #1	.07	0
Case #2	1.25	0
Case #3	2	12

Table 2: Average running times and number of backtracks, for one solution, for the three experimental setups.

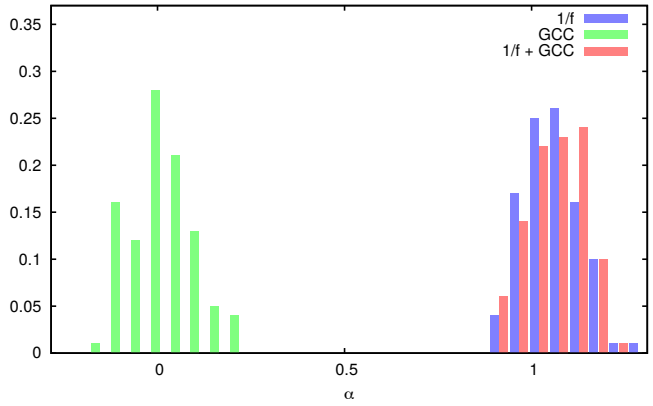

Figure 10: The histogram of slope values for the three cases (Voss constraint only, GCC only and both). It can be seen that the Voss constraint biases the spectrum as expected, even in the presence of a GCC constraint.

5.3 Shorter Melodies

The impact of the Voss constraint on shorter melodies can be assessed through examples rather than from the analysis of the spectrum. Here, melodies are generated from a Markov model estimated from two well-known songs of Michel Legrand ("You must believe in Spring" and "How do you keep the music playing?").

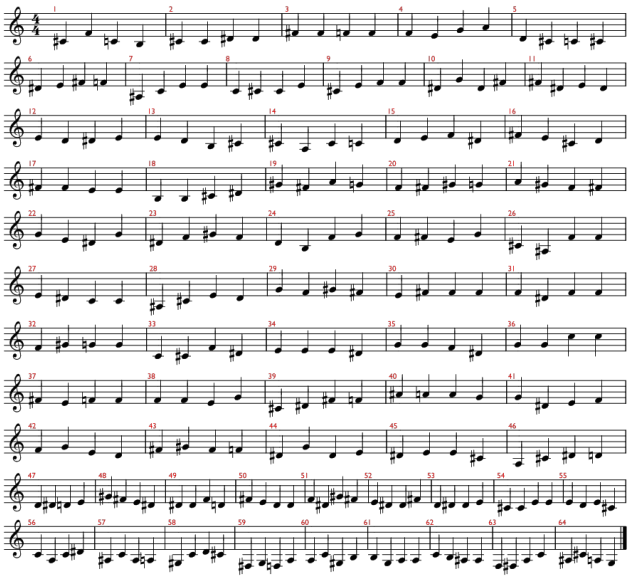

Figure 11: A melody generated with the Voss constraint only.

In the first case (Figure 14) we use the *meter* constraint [Roy and Pachet, 2013] to enforce metrical properties of the generated sequence (notes do not cross bars), and a total duration of 12 bars (in $4/4$). Without the Voss constraint, meter has a tendency to generate solutions which include long cycles (see, e.g., the series of 5 consecutive C at the end of the melody in Figure 14). Also, *meter* (which uses a Markov model of order 1) may generate series of large intervals in a row, which sounds awkward. These two cases seem to be less frequent when using the Voss constraint, because of the imposition of the specific correlation structure (Figure 15).

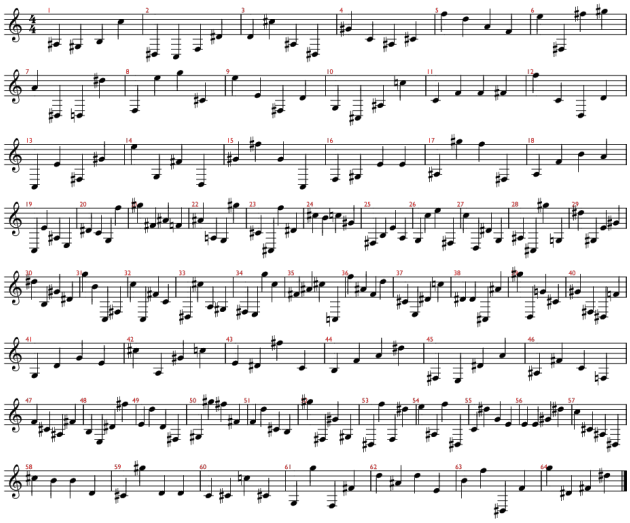

Figure 12: A melody generated with a GCC constraint only, i.e., a balanced distribution of pitches.

6 Discussion

We have introduced the Voss constraint, which ensures that the elements of a sequence are generated from the dice tossing scheme invented by Voss as described by Gardner. We showed that the constraint can be implemented as a tree of ternary sum constraints, yielding an efficient domain-consistency procedure, so its overhead is arguably small. The Voss constraint can be added to any sequence generation problem to introduce a pressure for the sequence to exhibit a $1/f$ spectral density in average, and therefore look or sound more *natural*.

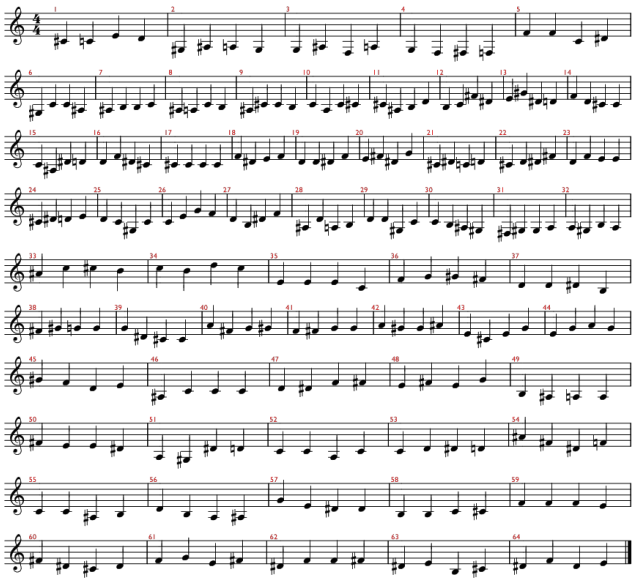

Figure 13: A melody generated with a GCC and Voss constraint.

There are several limitations to our proposal. First, Voss' algorithm imposes a specific structure of correlation, corresponding to the ordering imposed by an increasing integer sequence written in base 2. However, any other ordering could be used as well. This can be addressed by a generalization of the algorithm in which the number of dice to toss is chosen at random at each step, which would also result in a $1/f$ distribution. The probability to toss d dice is proportional to $1/2^d$ to match *on average* the frequency of dice re-tossing performed by Voss's original algorithm. To do so, however, requires the use of dynamic constraints (sums constraints would be posted depending on the value of this variable) or reified constraints, which would increase the filtering cost. Another approach is to draw a correlation structure at random before the resolution, but this does not ensure that the chosen structure is compatible with the other constraints.

Another extension of the algorithm concerns the value of α. Voss' algorithm produces exactly $1/f$ series ($\alpha = 1$). It could be interesting to control the value of α, typically within [0,2]. This could be done, again, by introducing a variable representing the number of dice to toss at each step, and biasing its probability distribution through a specific value ordering heuristics. Another limitation concerns the coding. The

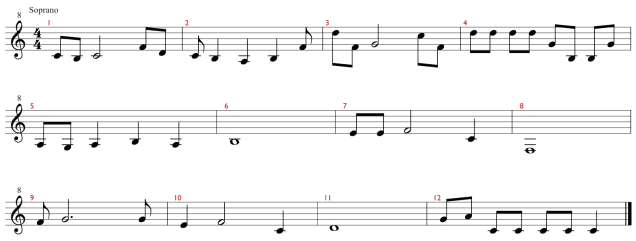

Figure 14: A shorter melody generated from a Markov model of 2 songs of Michel Legrand, enforcing only metrical properties (here, a total duration of 12 bars) with the meter constraint.

Voss constraint requires a coding from the domain (e.g., musical notes) to integers (in our example the pitch of a note). This coding is arbitrary and the user may want to experiment with various codings (e.g., integers representing the degree of the note in a scale) and there is no *best* coding *a priori*. Finally, one could ensure that solutions look more random by enforcing additional constraints on dice. For instance one could enforce a uniformly distributed set of values, to simulate the effect of the law of big numbers. This could be done by posting cardinality or *nValue* [Bessière *et al.*, 2006] constraints on each die (i.e., for a die i, all the $D_{i,j}$).

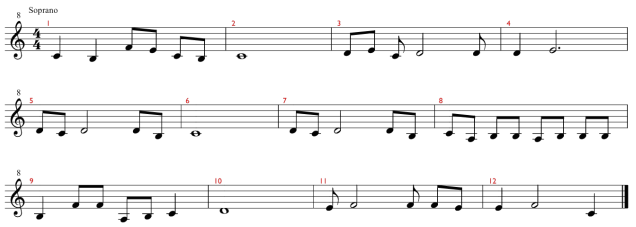

Figure 15: Same example as in Figure 14 but with a *Voss* constraint added.

There are obviously limits to representing stochastic properties with hard constraints. Another approach, in progress, is to model $1/f$ as a statistical constraint in the line of Walsh [2002] or Rossi *et al.* [2014] using the same die structure. However, we think that the work presented here contributes to constraint-based sequence generation by providing insights on this elusive but ubiquitous $1/f$ phenomenon, seen from a generative perspective.

Acknowledgements

This research is conducted within the Flow Machines project which received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 291156. We wish to thank Mirko Degli Esposti for insightful discussions about $1/f$ distributions.

References

[Bartolini *et al.*, 2011] Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. Neuron constraints to model

complex real-world problems. In Jimmy Ho-Man Lee, editor, *Principles and Practice of Constraint Programming - CP 2011, Perugia, Italy*, volume 6876 of *Lecture Notes in Computer Science*, pages 115–129. Springer, 2011.

[Beeri *et al.*, 1983] Catriel Beeri, Ronald Fagin, David Maier, and Mihalis Yannakakis. On the Desirability of Acyclic Database Schemes. *J. ACM*, 30(3):479–513, 1983.

[Beldiceanu *et al.*, 2007] Nicolas Beldiceanu, Mats Carlsson, Sophie Demassey, and Thierry Petit. Global constraint catalogue: Past, present and future. *Constraints*, 12(1):21–62, 2007.

[Bessière *et al.*, 2006] Christian Bessière, Emmanuel Hébrard, Brahim Hnich, Zeynep Kiziltan, and Toby Walsh. Filtering algorithms for the nvalue constraint. *Constraints*, 11(4):271–293, 2006.

[choco Team, 2010] choco Team. choco: an Open Source Java Constraint Programming Library. Research report 10-02-INFO, École des Mines de Nantes, 2010.

[Farrell *et al.*, 2006a] Simon Farrell, Eric-Jan Wagenmakers, and Roger Ratcliff. $1/f$ noise in human cognition: Is it ubiquitous, and what does it mean? *Psychonomic Bulletin and Review*, 13(4):737–741, 2006.

[Farrell *et al.*, 2006b] Simon Farrell, Eric-Jan Wagenmakers, and Roger Ratcliff. Estimation and interpretation of $1/f$ noise in human cognition. *Psychonomic Bulletin and Review*, 11:579–615, 2006.

[Gardner, 1978] Martin Gardner. Mathematical games-white and brown music, fractal curves and one-over-f fluctuations. *Scientific American*, 238(4):16–32, 1978.

[Hennig *et al.*, 2011] Holger Hennig, Ragnar Fleischmann, Anneke Fredebohm, York Hagmayer, Jan Nagler, Annette Witt, Fabian J. Theis, and Theo Geisel. The nature and perception of fluctuations in human musical rhythms. *PLoS ONE*, 6(10):e26457, 10 2011.

[Jégou, 1993] Philippe Jégou. On the consistency of general constraint-satisfaction problems. In Richard Fikes and Wendy G. Lehnert, editors, *Proceedings of the 11th National Conference on Artificial Intelligence. Washington, DC, USA, July 11-15, 1993.*, pages 114–119. AAAI Press / The MIT Press, 1993.

[Kasdin, 1995] N. Jeremy Kasdin. Discrete simulation of colored noise and stochastic processes and $1/f^{\alpha}$ power law noise generation. *Proceedings of the IEEE*, 83(5):802–827, May 1995.

[Lombardi and Gualandi, 2013] Michele Lombardi and Stefano Gualandi. A new propagator for two-layer neural networks in empirical model learning. In Christian Schulte, editor, *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, volume 8124 of *Lecture Notes in Computer Science*, pages 448–463. Springer, 2013.

[Mandelbrot, 1982] Benoit B. Mandelbrot. *The fractal geometry of nature*. W.H. Freeman, 1st edition, August 1982.

[Morin and Quimper, 2014] Michael Morin and Claude-Guy Quimper. The markov transition constraint. In Helmut Simonis, editor, *Integration of AI and OR Techniques in Constraint Programming - 11th International Conference, CPAIOR 2014, Cork, Ireland, May 19-23, 2014. Proceedings*, volume 8451 of *Lecture Notes in Computer Science*, pages 405–421. Springer, 2014.

[Pachet and Roy, 2011] François Pachet and Pierre Roy. Markov constraints: steerable generation of Markov sequences. *Constraints*, 16(2), 2011.

[Pachet, 1994] François Pachet. An Object-Oriented Representation of Pitch-Classes, Intervals, Scales and Chords. In *Proceedings of Journées d'Informatique Musicale (JIM)*, 1994.

[Perlin, 1985] Ken Perlin. An Image Synthesizer. *Proceedings of ACM SIGGRAPH*, 24(3), 1985.

[Pesant and Régin, 2005] Gilles Pesant and Jean-Charles Régin. SPREAD: A balancing constraint based on statistics. In Peter van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005, Sitges, Spain*, volume 3709 of *Lecture Notes in Computer Science*, pages 460–474. Springer, 2005.

[Régin, 1996] Jean-Charles Régin. Generalized arc consistency for global cardinality constraint. In William J. Clancey and Daniel S. Weld, editors, *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, August 4-8, 1996, Volume 1.*, pages 209–215. AAAI Press / The MIT Press, 1996.

[Rossi *et al.*, 2014] Roberto Rossi, Steven David Prestwich, and S. Armagan Tarim. Statistical constraints. In Torsten Schaub, Gerhard Friedrich, and Barry O'Sullivan, editors, *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 777–782. IOS Press, 2014.

[Roy and Pachet, 2013] Pierre Roy and François Pachet. Enforcing Meter in Finite-Length Markov Sequences. In Marie desJardins and Michael L. Littman, editors, *AAAI*. AAAI Press, 2013.

[Schaus *et al.*, 2007] Pierre Schaus, Yves Deville, Pierre Dupont, and Jean-Charles Régin. The deviation constraint. In Pascal Van Hentenryck and Laurence A. Wolsey, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, CPAIOR, Brussels, Belgium*, volume 4510 of *Lecture Notes in Computer Science*, pages 260–274. Springer, 2007.

[Trick, 2003] Michael A. Trick. A dynamic programming approach for consistency and propagation for knapsack constraints. *Annals OR*, 118(1-4):73–84, 2003.

[Voss and Clarke, 1975] Richard Voss and John Clarke. $1/f$ noise in music and speech. *Nature*, 258:317–318, 1975.

[Voss and Clarke, 1978] Richard Voss and John Clarke. $1/f$ noise in music: Music from $1/f$ noise. *Journal of the Acoustical Society of America*, 63:258–263, 1978.

[Walsh, 2002] Toby Walsh. Stochastic constraint programming. In Frank van Harmelen, editor, *Proceedings of the 15th Eureopean Conference on Artificial Intelligence, ECAI'2002, Lyon, France, July 2002*, pages 111–115. IOS Press, 2002.