# Uniform random generations and rejection method(I) with binomial majorant

Laurent Alonso

▶ **To cite this version:**

Laurent Alonso. Uniform random generations and rejection method(I) with binomial majorant. 2023. hal-04031703

**HAL Id: hal-04031703**
**https://inria.hal.science/hal-04031703**

Preprint submitted on 16 Mar 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Uniform random generations and rejection method(I) with binomial majorant

Laurent Alonso*

March 16, 2023

**Abstract.** We present three simple algorithms to uniformly generate 'Fibonacci words' (i.e., some words that are enumerated by Fibonacci numbers), Schröder trees of size $n$ and Motzkin left factors of size $n$ and final height $h$. These algorithms have an average complexity of $O(n)$ in the unit-cost RAM model[1] and use only small integers (integers smaller than $n^2$).

**Key words.** Generation, uniform, rejection

## 1 Introduction

One of the first generic methods used to uniformly generate various data structures is the *recursive method*[NW78][FZC94]. It consists in computing the number of structures of different sizes and using these numbers to recursively generate an element. A drawback is that the coefficients are often very large: $\approx C^n$ for some constants $C$ (and so required to link to a library that can compute very large numbers) and the method is very slow. Denise et al.[DZ99] shows that it can be improved by performing lazy floating point operations, this allows to obtain a more efficient algorithm[2] but it remains very difficult to implement.

For some problems: 'Fibonacci words', Schröder trees[PPPR01], Motzkin words[Alo94],[BBJ17], [GBN10] ... rejection methods give efficient algorithms; in general, these methods are based on ingenious one-to-one correspondences and are easy to implement[3].

*INRIA Nancy-Grand Est, 615, rue du Jardin Botanique, 54600 Vandoeuvre-lès-Nancy, France. Email: `Laurent.Alonso@inria.fr`

[1]Since we only present algorithms that use small integers in this article, we will use the unit cost RAM model to give the complexity, you can get the logarithmic cost RAM model by multiplying this complexity by a factor $log\, n$.

[2]This article presents algorithms corresponding to some context-free grammars, the average complexity $O(n^{1+\epsilon})$.

[3]In fact, generating a random Motzkin word gives a simple algorithm but concerning other problems, with low probability, these methods require computing floats with very high accuracy, ...

We try in this paper to extend the method proposed by [Alo94]: this method first chooses the number of horizontal steps $m$ with an adequate probability using a rejection method, then generates a random Motzkin word with $m$ of horizontal steps (which is a simple problem). In [GBN10], Gouyou-Beauchamps et all show that this method can be extended to generate colored unary-binary trees with linear average complexity and partial injection with average complexity in $O(n^{5/4})$. Their method consists in choosing a number $m$ with probability $\frac{F_n(m)}{\sum_i F_n(i)}$ by finding a distribution $\overline{B}_n(m)$ such that :

- it is easy to generate a number $m$ with probability $\frac{\overline{B}_n(m)}{\sum_i \overline{B}_n(i)}$,
- we have for all $m$, $F_n(m) \leq \overline{B}_n(m)$ and we have an efficient algorithm to accept a value of $m$ with probability $\frac{F_n(m)}{\overline{B}_n(m)}$.

This method is quite general, the proposed algorithms are easy to implement and ingenious; however finding a distribution $\overline{B}_n(m)$ that fulfills these properties is not straightforward even if they propose to use for $\overline{B}_n(m)$ a binomial distribution multiplied by a constant.

In this paper, we show that the choice of a binomial distribution (up to a minor modification) close to the final distribution allows to obtain simply an efficient algorithm at least for some basic structures. More precisely, we will show that it is possible to obtain efficient algorithms to randomly and uniformly generate a 'Fibonacci word' of size $n$ (the simplest problem), a Schröder tree of size $n$ (with almost the same algorithm) and finally a Motzkin left factor of size $n$ and final height $h$. As in [GBN10], these algorithms use only small integers (i.e. integers smaller than $n^2$), have an average complexity of $O(n)$ and can be implemented very easily. Another interesting property is that almost all these algorithms use on average only $O(\sqrt{n})$ operations that involve numbers greater than $n$; only the generation of a Motzkin left factor of size $n$ and final height $h$ when $h \geq n - \frac{1}{2} - \frac{\sqrt{8n+9}}{2}$ is different because it uses on average $O(n)$ operations that all involve numbers less than $3n + 7$.

In a future paper[Alo23], we will propose another method which consists in drawing the initial value of $m$ differently: it is enough to generate an integer $m$ with the basic uniform distribution (i.e. with probability in $\frac{1}{n}$). We will show that, at least for simple problems, this leads to very efficient algorithms. Although we will probably choose to use this second method for simple problems, we think that these two papers are interesting because :

- together, they suggest that the initial values of $m$ can be chosen by different types of methods,
- these solutions have different properties: in this paper, we propose to guess the first value of $m$ with a distribution close to the final distribution, which leads to few rejections but to many operations in the main loop. The new paper will propose the opposite, so we will have a very high rejection rate while the time spent in a loop is very low. We suspect that in some cases,

for example if we want to find an algorithm to choose the number of binary nodes and the number of ternary nodes in $3-ary$ trees, having a high rejection rate is prohibitive,

- in practice, it is possible to mix the ideas of these two articles: [Alo23] to generate a first value of $m$ with a distribution close to the final distribution but still following the methods proposed in this article.

In Section 2, we introduced the notations, an abstract algorithm and the different generating functions that will be used to generate the initial $m$ value. Then, in the Sections 3–5, we will see how to instantiate this abstract algorithm to generate 'Fibonacci words', Schröder trees of size $n$ and a Motzkin left factor of size $n$ and given final height $h$ and prove that these algorithms choose a value $m$ with an adequate probability and have an average complexity of $O(n)$.

Finally, after a little discussion, we give in the appendices the proofs of many lemmas. These proofs use only elementary mathematics (and had to be done); as writing detailed proofs would probably be too long and boring, we tried to give just enough details so that they can be checked without too much trouble...

# 2 Notations and Abstract Algorithm

## 2.1 Notations

In the following we will use:

- $[P]$ the Kenneth Iverson's convention[GKP88](page 24): $[P] = 1$ if $P$ is true and $[P] = 0$ if $P$ is false,
- $random(k)$ the function that returns an integer between 0 and $k-1$ with probability $\frac{1}{k}$[4],

and:

- $F(m)$ will count the final number of elements of size $m$,
- $M = \min \operatorname{argmax}_m F(m)$: the first value of $m$ that maximizes $F(m)$,
- $\underline{B}(m)$ is the number of instances of the initial generator that correspond to a choice $m$,
- $B$ a slightly modified function of $\underline{B}$ to impose for example $B(M-1) = B(M)$, ...,
- $\overline{B}(m) = B(m)\frac{F(M)}{B(M)}$ : the normalized function such that $\overline{B}(M) = F(M)$,
- $\underline{G}(m) = \frac{F(m)}{\underline{B}(m)}$, $G(m) = \frac{F(m)}{B(m)}$, $\overline{G}(m) = \frac{F(m)}{\overline{B}(m)}$.

[4]to simplify some notations, when $p$,$q$,$r$,$s$ are four integers, we will sometimes note $random(p/q) \geq r/s$ instead of $random(p\,s) \geq q\,r$, ...

Finally, we will note:

- $\mathcal{R}F(m) = F(m+1)/F(m)$ the ratio between two consecutive terms of $F$ when they are defined,
- if $\mathcal{R}F(m) = \mathrm{n}umerator(m)/\mathrm{d}enominator(m)$ is a ratio between two polynomials in $m$, $f(x) = \mathrm{n}umerator(x) - \mathrm{d}enominator(x)$,

and we will define $\mathcal{R}\underline{B}(m)$, ..., $\overline{g}(x)$ in the same way.

## 2.2 An Abstract Algorithm

With these notations, we will use the following procedure to draw a number $m$ with probability $\frac{F(m)}{\sum_m F(m)}$:

**Algorithm 1** Choose $m$ with probability $\frac{F(m)}{\sum_m F(m)}$:

```
1:  function ACCEPT_M(M,m)        ▷ accept m with probability Ḡ(m) = F(m)/B̄(m)
2:      if m < M then
3:          for all i ∈ [m, M − 1] do        ▷ accept with probability RB̄(i)/RF(i)
4:              if random(RF(i)) ≥ RB̄(i) then return FALSE end if
5:          end for
6:      else
7:          for all i ∈ [M, m − 1] do        ▷ accept with probability RF(i)/RB̄(i)
8:              if random(RB̄(i)) ≥ RF(i) then return FALSE end if
9:          end for
10:     end if
11:     return TRUE
12: end function

13: M ← first value of m which maximalizes F(m)
14: while true do
15:     choose m with probability B(m)/∑_m B(m) = B̄(m)/∑_m B̄(m)
16:     if ACCEPT_M(M, m) then return m end if
17: end while
```

Lines 1 to 12 correspond to the main function that accepts a value $m$ with probability $\frac{F(m)}{\overline{B}(m)}$. It is based on the following relations, when $m < M$:

$$\begin{aligned}\frac{F(m)}{\overline{B}(m)} &= \frac{F(m)\overline{B}(m+1)}{\overline{B}(m)F(m+1)}\frac{F(m+1)\overline{B}(m+2)}{\overline{B}(m+1)F(m+2)}\cdots\frac{F(M-1)\overline{B}(M)}{\overline{B}(M-1)F(M)}\frac{F(M)}{\overline{B}(M)} \\ &= \frac{\mathcal{R}\overline{B}(m)}{\mathcal{R}F(m)}\frac{\mathcal{R}\overline{B}(m+1)}{\mathcal{R}F(m+1)}\cdots\frac{\mathcal{R}\overline{B}(M-1)}{\mathcal{R}F(M-1)}\end{aligned}$$

similarly when $m > M$:

$$\frac{F(m)}{\overline{B}(m)} = \frac{\mathcal{R}F(M)}{\mathcal{R}\overline{B}(M)} \frac{\mathcal{R}F(M+1)}{\mathcal{R}\overline{B}(M+1)} \cdots \frac{\mathcal{R}F(m-1)}{\mathcal{R}\overline{B}(m-1)},$$

and finally to $\frac{F(M)}{\overline{B}(M)} = 1$. It therefore assumes that when $i < M$, $\mathcal{R}F(i) \geq \mathcal{R}\overline{B}(i)$ and when $i \geq M$, $\mathcal{R}F(i) \leq \mathcal{R}\overline{B}(i)$.

Lines 13 to 17 group the main lines of the program: First, we compute the value of $M$ (line 13), then we choose a value of $m$ by repeatedly drawing a value of $m$ with probability $\frac{B(m)}{\sum_m B(m)} = \frac{\overline{B}(m)}{\sum_m \overline{B}(m)}$ and accept it with probability $\frac{F(m)}{\overline{B}(m)}$. We accept in this loop a value $m$ with the probability $\frac{\sum_m F(m)}{\sum_m \overline{B}(m)}$, this value must be big enough to obtain an efficient algorithm. In addition, we have more specifically:

**Lemma 1** *Let us note $C_1$ the complexity necessary to choose a value $m$ with probability $\frac{\overline{B}(m)}{\sum_m \overline{B}(m)} = \frac{B(m)}{\sum_m B(m)}$, $C_2 = \frac{\sum_m |M-m|\overline{B}(m)}{\sum_m \overline{B}(m)} = \frac{\sum_m |M-m|B(m)}{\sum_m B(m)}$ the mean absolute difference of $B$. If we assume that the tests of lines 4 and 8 can be performed with complexity $O(1)$, the average complexity of this algorithm is $O\left((1 + C_1 + C_2)\frac{F(M)}{\sum_m F(m)}\frac{\sum_m B(m)}{B(M)}\right)$.*

*Moreover, if the algorithm leads to a value $m$, the average complexity* [5] *is $O\left(|M-m| + (1 + C_1 + C_2)\frac{F(M)}{\sum_m F(m)}\frac{\sum_m B(m)}{B(M)}\right)$.*

*In the particular case where $C_1 = O(n)$, $C_2 = O(\sqrt{n})$, $\frac{\sum_m F(m)}{F(M)} = \Omega(\sqrt{n})$, $\frac{\sum_m B(m)}{B(M)} = O(\sqrt{n})$, we will obtain an average complexity in $O(n)$ and when the algorithm ends with a value $m$ also $O(n)$.*

*Proof.* We first study the number of operations passed inside a loop. We need by definition, at line 15, $C_1$ operations and as ACCEPT_M is called with a number $m$ (chosen with probability $\frac{B(m)}{\sum_m B(m)}$) at line 16: $O(1) + C_2$ operations. This corresponds to an average complexity of a main loop execution of $O(1)+C_1+C_2$.

We now know that it accepts a value $m$ with the probability $C_3 = \frac{\sum_m F(m)}{\sum_m \overline{B}(m)}$. The average number of loops is therefore

$$1+(1-C_3)+(1-C_3)^2+\ldots = \frac{1}{1-(1-C_3)} = \frac{\sum_m \overline{B}(m)}{\sum_m F(m)} = \frac{F(M)}{\sum_m F(m)}\frac{\sum_m B(m)}{B(M)}.$$

We finish by noting that when the algorithm returns a value $m$, we use in the last execution of the loop $O(C_1 + 1 + |m - M|)$ operations. □

[5] This notion of complexity is important because for some algorithms, its maximum value can be very different from the average complexity ….

## 2.3 Some generating functions

We present three generating functions that we will use in this paper to generate an initial value $m$.

**'Basic Generator'** This is the simplest generator: to generate an integer between $[0, k]$ with probability $\frac{1}{k+1}$, we use $random(k+1)$. So we have $\underline{B}^k(m) = B^k(m) = 1$ when $0 \leq m \leq k$ (and $\mathcal{R}\underline{B}^k(m) = \mathcal{R}B^k(m) = 1$ when $0 \leq m < k$) Its complexity is very low: $O(1)$. We will use it to generate a Motzkin left factor of final height $h$ when $h$ is near $n$.

**'Binomial Generator'** We choose $\underline{B}_M(m) = \begin{pmatrix} 2M \\ m \end{pmatrix}$, $B_M(m) = \underline{B}_M(m)$ when $m \neq M$ and $B_M(M) = B_M(M+1)$. These choices give us an efficient algorithm to draw the first value of $m$: draw $2M$ random value 0 or 1, let $m$ count the number of 0, finally, if we find $m = M$, accept this choice with probability $\frac{M}{M+1}$. We also obtain:

$$\begin{aligned} \mathcal{R}\underline{B}_M(m) &= \frac{2M-m}{m+1} \\ \mathcal{R}B_M(m) &= \mathcal{R}\underline{B}_M(m) \text{ if } m < M-1 \text{ or } m > M \\ \mathcal{R}B_M(M-1) &= \mathcal{R}B_M(M) = 1 \end{aligned}$$

This generator chooses an integer $m$ between 0 and $2M$ with a probability close to the binomial distribution. It has an average complexity of $O(M)$. We will need the following lemma to relate $B_M(M)$ to $\sum_m B_M(m)$.

**Lemma 2** *We have when $M \geq 1$: $\frac{\sum |M-m| B_M(m)}{\sum B_M(m)} = \Theta(\sqrt{M})$ and $\frac{\sum B_M(m)}{B_M(M)} = \Theta(\sqrt{M})$.*

*Proof.* $\underline{B}_M$ is the classical binomial distribution with $p = q = \frac{1}{2}$ multiplied by $2^{2M}$, so we have: $\frac{\sum |M-m| \underline{B}_M(m)}{\sum \underline{B}_M(m)} = \Theta(\sqrt{M})$ and using the Stirling Formula $\frac{\sum \underline{B}_M(m)}{\underline{B}_M(M)} = \Theta(\sqrt{M})$.

We can conclude by noting that when $m \neq M$, $\underline{B}_M(m) = B_M(m)$ and $B_M(M) = \frac{m}{m+1}\underline{B}_M(M)$ (and thus $B_M(M) = \Theta(\underline{B}_M(M))$ and $\sum B_M(m) = \Theta(\sum \underline{B}_M(m))$). □

This algorithm can be coded as:

**function** BIN($M$) ▷ choose $m$ with probability $\frac{B_M(m)}{\sum_m B_M(m)}$
  **while** true **do**
    $m \leftarrow 0$
    **for all** $i \in [1, 2M]$ **do**
      **if** $random(k+1) = 0$ **then** $m \leftarrow m + 1$

**end for**
**if** $m = M$ AND $random(M+1) = 0$ **then** continue **end if**
return $m$
**end while**
**end function**

We will use the 'Binomial Generator' when possible but note that it has the limitation of only being able to draw a number between 0 and $2M$(which sometimes is not enough).

**'Extended Binomial Generator'** The previous generator corresponds to an equiprobable binomial distribution: $p = q = \frac{1}{2}$, we will use here an equivalent of the generic distribution with $p = \frac{1}{k+1}$ and $q = 1 - p$; let's define $\underline{B}_M(m)$ as:

$$\underline{B}_M^{k,\alpha}(m) = k^{(k+1)M+\alpha-m} \begin{pmatrix} (k+1)M+\alpha \\ m \end{pmatrix},$$

we get $\mathcal{R}\underline{B}_M^{k,\alpha}(m) = \frac{(k+1)\,M+\alpha-m}{k(m+1)}$. If $0 \leq \alpha < k$, this function reaches its maximum when $m = M$, it can draw numbers between 0 and $(k+1)M+\alpha$ and it is simple to implement: just draw $(k+1)M+\alpha$ random integers between 0 and $k$, and count the number of 0.

We can now define $B(m)$ as:

- $B_M^{k,\alpha}(m) = \underline{B}_M^{k,\alpha}(m)$ when $m \neq M$,
- $B_M^{k,\alpha}(M) = max(\underline{B}_M^{k,\alpha}(M-1), \underline{B}_M^{k,\alpha}(M+1))$.

This gives us:

- when $m < M-1$ or $m \geq M+1$, $\mathcal{R}B_M^{k,\alpha}(m) = \mathcal{R}\underline{B}_M^{k,\alpha}(m)$,
- when $B_M^{k,\alpha}(M) = B_M^{k,\alpha}(M+1)$

$$\mathcal{R}B_M^{k,\alpha}(M-1) = \frac{(kM+\alpha+1)(kM+\alpha)}{k^2M(M+1)}, \mathcal{R}B_M^{k,\alpha}(M) = 1,$$

- and when $B_M^{k,\alpha}(M) = B_M^{k,\alpha}(M-1)$

$$\mathcal{R}B_M^{k,\alpha}(M-1) = 1, \mathcal{R}B_M^{k,\alpha}(M) = \frac{(kM+\alpha+1)(kM+\alpha)}{k^2M(M+1)}.$$

We will prove the following lemma in the Appendix A.1:

**Lemma 3** *When $0 \leq \alpha \leq k-1$, $k \geq 1$ and $M \geq 1$, we have:*

- $\underline{B}_M^{k,\alpha}(m) \geq \underline{B}_M^{k,\alpha}(m+1)$ *iff* $m \geq M$,
- $\frac{\sum_m B_M^{k,\alpha}(m)}{B_M^{k,\alpha}(M)} = \Theta(\sqrt{M})$,

- $\frac{\sum_m |m-M| B_M^{k,\alpha}(m)}{\sum_m B_M^{k,\alpha}(m)} = O(\sqrt{M})$.

When $0 \leq \alpha \leq k-1$, this algorithm can be coded as:

**function** EXTENDED_BIN($M, k, \alpha, which$)
  **Require:** $M > 0$, $k$, $0 \leq \alpha \leq k-1$
  **while** true **do**
    $m \leftarrow 0$
    **for all** $i \in [1, (k+1)M + \alpha]$ **do**
      **if** $random(k+1) = 0$ **then** $m \leftarrow m+1$ **end if**
    **end for**
    **if** $m \neq M$ **then** return $m$ **end if**
    **if** $which$ = 'M+1' AND $random(k(M+1)) \geq k\,M + \alpha$ **then**
      continue ▷ reject $m$
    **else if** $which$ = 'M-1' AND $random(k\,M + \alpha + 1) \geq k\,M$ **then**
      continue ▷ reject $m$
    **end if**
    return $m$
  **end while**
**end function**

It generates an integer $m$ with probability $\frac{B_M^{k,\alpha}(m)}{\sum_{i=0}^{(k+1)M+\alpha} B_M^{k,\alpha}(i)}$ and its average complexity is $O(k(M+1)+\alpha)$.

# 3 Generation of a 'Fibonacci word'

Fibonacci numbers have a very long history: they correspond to the sequence A000045 in The On-Line Encyclopedia of Integer Sequences[OEI22] and are defined by the equation $F_n = F_{n-1} + F_{n-2}$ with $F_0 = F_1 = 1$, they are related to the golden number $\varphi = \frac{1+\sqrt{5}}{2}$ by the well known relation $F_n = \frac{\varphi^{n+1} - (1-\varphi)^{-(n+1)}}{\sqrt{5}} = \left\lceil \frac{\varphi^{n+1}}{\sqrt{5}} \right\rfloor$ (using the nearest rounding function).

Let us note first that we can obtain a simple algorithm by recursively choosing a letter $a$ with probability $\frac{F_{n-1}}{F_n} \approx \varphi$ and a letter $b$ with probability $\frac{F_{n-2}}{F_n} \approx \varphi^2$, but this algorithm is not very efficient. So we prefer to use a rejection algorithm: we simply build a sequence of letters by choosing $a$ with probability $\alpha$ and $b$ with probability $\alpha^2$ with $\alpha + \alpha^2 = 1$ ($\alpha = \frac{-1+\sqrt{5}}{2}$) and we stop when we have obtained a word of $\mathcal{F}_n$ or a word of $\mathcal{F}_{n+1}$ (a failure). In this last case, we start a new generation from the beginning. We can check that in one try, we generate each word of $\mathcal{F}_n$ with a probability $\alpha^n$ and each word of $\mathcal{F}_{n-1}$ followed by a letter $b$ with a probability $\alpha^{n+1}$ ; the probability of failure is therefore $\frac{\alpha^{n-1} F_{n-1} \alpha^2}{\alpha^n F_n + \alpha^{n+1} F_{n-1}} \approx \frac{\alpha}{\varphi+\alpha} = 1 - \frac{\sqrt{5}+1}{2\sqrt{5}}$. This proves that the average number of trials is $O\left(\frac{2\sqrt{5}}{\sqrt{5}+1}\right) = O(1)$ and that the average complexity is $O(n)$.

However, the main step of this algorithm is to generate a real $x$ in the interval $[0,1]$ and to compare it to $\alpha$. Thus, even if we start generating the

most significant digits of $x$, we need to check if we have enough digits to be sure that $x < \alpha$ or $x \geq \alpha$. If this is not the case, we need to draw the following digits...

We present here an algorithm that draws a random word containing the letters $a$ and $b$ such that $n = |a|+2|b|$ but using only small integers (less than $n^2$). We have $F_n(m) = \begin{pmatrix} n-m \\ m \end{pmatrix}$ where $m$ counts the number of letters $b$, which gives us: $\mathcal{R}F(m) = \frac{(n-2m)(n-1-2m)}{(m+1)(n-m)}$, $f(x) = (n-2x)(n-1-2x)-(x+1)(n-x)$, and the following lemma which groups together some necessary properties of $F_n(m)$ and is proved in the Appendix A.2:

**Lemma 4** *When $n > 0$,*

- *there exists a unique value $\tilde{m}$ in $[0 \dots \lfloor n/2 \rfloor]$ such that $f(x) > 0$ with $x \in [0, \lfloor n/2 \rfloor]$ if and only if $x < \tilde{m}$,*
- $\tilde{m} = \frac{5n-3-\sqrt{5n^2+10n+9}}{10}$, $M = \lceil \tilde{m} \rceil \approx \frac{5-\sqrt{5}}{10} n$,
- $\frac{F_n}{F_n(M)} = \Omega(\sqrt{n})$.

The existence of $\tilde{m}$ gives us a linear method for computing $M$(Algorithm 1, line 13): we can either make $m$ go from 0 to $\lfloor n/2 \rfloor$ and stop as soon as $\mathcal{R}F_n(m) \leq 1$ (i.e. $f(m) \leq 0$) or we can compute $\lceil \tilde{m} \rceil$ directly.

We choose the 'Binomial Generator' to obtain the first value of $m$(Algorithm 1, line 15). We thus obtain:

$$\begin{aligned}
\mathcal{R}\overline{B}_M(m) &= \mathcal{R}B_M(m) \\
\mathcal{R}\underline{G}_M(m) &= \frac{(n-2m)(n-1-2n)}{(2M-m)(n-m)} \\
\mathcal{R}\overline{G}_M(m) &= \frac{(n-2m)(n-1-2n)}{(2M-m+[m=M]-[m=M-1])(n-m)}.
\end{aligned}$$

We now need the following lemma (which is proved in the Appendix A.2):

**Lemma 5** *We have:*

- $\mathcal{R}\overline{B}_M(m) < \mathcal{R}F_n(m)$ *if and only if* $m < M$,

to show that we accept a value of $m$ correctly with a probability of $\mathcal{R}\overline{G}_M(m) = \frac{\mathcal{R}F_n(m)}{\mathcal{R}\overline{B}_M(m)}$ and complete lines 4 and 8 of the Algorithm 1.

Now as a consequence of Lemma 1, $C_1 = O(n)$, $C2 = O(\sqrt{n})$, ..., we obtain:

**Proposition 1** *This algorithm chooses $m$ with probability $\frac{F_n(m)}{F_n}$ and has an average complexity in $O(n)$ using only numbers with $2log_2(n)$ bits.*

**Remark:** We use numbers with $2log_2(n)$ bits:

- $n$ times to compute the value $M$ that maximizes $F_n^m$. But we can also compute a rough approximation $m' = \left\lceil \frac{5-\sqrt{5}}{10} n \right\rceil$ and then search around $m'$ to find the value $M$ using $O(1)$ times such large numbers,
- $|m - M|$ times to check the acceptance of the value $m$ with a probability of $F_n^m / \overline{B}_M(m)$. So, we can expect $|m - M|$ to be $O(\sqrt{M}) = O(\sqrt{n})$.

This suggests that using a library optimized to compute with numbers of $2\log_2(n)$ bits or not will not change the average complexity of this algorithm much.

The final algorithm code can be written as in Algorithm 2.

**Algorithm 2** Draw a random Fibonacci word with $|a| + 2|b| = n$:

```
function ACCEPT_M(n,M,m)    ▷ accept m with probability Ḡ(m) = F_n(m) / B̄_M(m)
    if m < M then
        for all i ∈ [m, M − 1] do
            v ← random((n − 2i)(n − 2i − 1))
            if v ≥ (n − i)(2M − i − [i = M − 1]) then return FALSE end if
        end for
    else
        for all i ∈ [M, m − 1] do
            v ← random((n − i)(2M − i + [i = M]))
            if v ≥ (n − 2i)(n − 2i − 1) then return FALSE end if
        end for
    end if
    return TRUE
end function

Require: n > 0
M ← first value of m which maximalizes F_n(m)
while true do                                                    ▷ Choose m
    m ← BIN(M)
    if ACCEPT_M(n, M, m) then break end if
end while
                ▷ draw a random word with n − 2m letters a and m letters b.
b ← m, a ← n − 2m
while a + b > 0 do
    if random(a + b) < a then
        add a letter a, a ← a − 1,
    else
        add a letter b, b ← b − 1.
    end if
end while
```

# 4 Schröder numbers

The Schröder numbers are defined by the recurrence $S_0 = 1$, $S_1 = 2$, $S_n = 3S_{n-1} + \sum_{k=1}^{n-2} S_k S_{n-k-1}$. $S_n \approx \frac{\sqrt{2}+1}{2^{3/4}\sqrt{\pi}} \frac{(3+2\sqrt{2})^n}{n\sqrt{n}}$(see [Au19]) is also the number of paths $x(1,1)$, $\bar{x}(1,-1)$, $z(0,2)$ that go from $(0,0)$ to $(2n,0)$ and do not go below the x axis.

Penaud et al.[PPPR01] propose to use a rejection method. This method is more complicated than for the Fibonacci number because it is based on correspondences between the Schröder path of size $2n$ and a subset of prefixes of Schröder paths of size $2n+1$. The prefixes of Schröder paths of size $2n+1$ are generated letter by letter: $x$ is chosen first, then $x$ is chosen with probability $\sqrt{2}-1$, $\bar{x}$ with probability $\sqrt{2}-1$, $z$ with probability $(\sqrt{2}-1)^2 = 3-2\sqrt{2}$, ... and the path is rejected if it passes under the y-axis; then the matches are used to reject this path or convert it to a Schröder path. In summary, this method is simple but requires some time to compute the floats with high accuracy.

We have:

$$F_n = S_n = \sum_{m=0}^{n} \frac{1}{n+m+1} \left( \begin{array}{c} n+m+1 \\ m, m+1, n-m \end{array} \right) = \sum_{m=0}^{n} F_n(m)$$

this formula corresponds for some $m$ to draw a random word with $m+1$ letters $x$, $m$ letters $\bar{x}$, $n-m$ letters $z$, then use the "Cycle Lemma"[DZ90] which gives a $n+m+1$-1 correspondence between such words and the Schröder path of size $2n$ which contains $m$ letters $x$.

We can now use the 'Binomial generator' as for the Fibonacci algorithm to draw a value $m$ with probability $\frac{F_n(m)}{F_n}$. When $m$ is chosen with an adequate probability, we finish by generating a random word with $m+1$ letters $x$, $m$ letters $\bar{x}$, $n-m$ letters $z$ and use the "Cycle Lemma"[DZ90] to get the final Schröder path.

So we have:

$$\begin{aligned}
\mathcal{R}F_n(m) &= \frac{(n+m+1)(n-m)}{(m+1)(m+2)} \\
\mathcal{R}\underline{G}_M(m) &= \frac{(n+m+1)(n-m)}{(m+2)(2M-m)} \\
\mathcal{R}\overline{G}_M(m) &= \frac{(n+m+1)(n-m)}{(m+2)(2M-m+[m=M]-[m=M-1])}
\end{aligned}$$

We will finally need a lemma which is proved in the Appendix A.3:

**Lemma 6** *When $n > 1$,*

- *there exists a unique value $\tilde{m}$ in $[0 \dots n]$ such that $f(x) > 0$ with $x \in [0,n]$ if and only if $x < \tilde{m}$,*
- $\tilde{m} = -1 + \frac{\sqrt{2n^2+2n}}{2}$, $M = \lceil \tilde{m} \rceil \approx \frac{n}{\sqrt{2}}$,

- $\frac{F_n}{F_n(M)} = \Omega(\sqrt{n})$,
- $\mathcal{R}\overline{B}_M(m) < \mathcal{R}F_n(m)$ *if and only if* $m < M$,

which implies that the modified algorithm generates each word with uniform probability and has an average complexity of $O(n)$ using only numbers with $2log_2(n)$ bits[6].

# 5 Motzkin left factors with size $n$ and final height $h'$

Many efficient algorithms are known to efficiently generate the Motzkin word with a fixed size, the simplest ones are based on a certain rejection method: Alonso[Alo94], Bacher et al.[BBJ17], Brlek et al. [BPR06]. Similarly, Barcucci et al.[BPS94] proposes an efficient algorithm based on a simple rejection method to generate Motzkin left factors of size $n$.

All these algorithms draw some sequence of $n$ small random numbers (often an integer between 1 and 3) and then accept/reject these sequences by doing $O(n)$ tests using numbers with $\log n$ bits. It seems difficult to extend these methods to generate a random Motzkin left factor of given size $n$ and final height $h'$ except:

- Brlek et al.[BPR06] can be modified to transform a Motzkin left factor with overall height greater than or equal to $h$ into a Motzkin left factor with final height $h'$. This gives us a valid algorithm but its average complexity will remain in $O(n)$ only if $h'$ is small enough: $h' = O(\sqrt{n})$,
- Alonso[Alo94] because this paper is an extension of this algorithm.

To simplify the notation, let us note $h = h' + 1$. When $h$ is small enough: $h \leq \frac{3n-25}{7}$, we can use the 'Binomial Generator' and get a valid algorithm with average complexity in $O(n)$, but it fails when $h$ is larger. We will therefore choose the 'Extended Binomial Generator' which allows us to obtain an algorithm[7] which works when $h < n - \frac{1}{2} - \frac{\sqrt{8n+9}}{2}$. Finally, to be exhaustive, we will prove that we can simply use the 'Basic Generator' when $h \geq n - \frac{1}{2} - \frac{\sqrt{8n+9}}{2}$.

Indeed, using the "Cycle Lemma"[DZ90], we obtain the number of Motzkin left factors of final height $h' = h - 1$ with $0 < h \leq n + 1$ :

$$F_n^h = \frac{h}{n+1} \sum_{m=0}^{\lfloor \frac{n+1-h}{2} \rfloor} \binom{n+1}{m, m+h} = \sum_{m=0}^{\lfloor \frac{n+1-h}{2} \rfloor} F_n^h(m)$$

[6] With Maple[Map], we can also compute the limit of the rate of acceptance in the main loop, we obtain $\lim_{n=\infty} C_n = 2^{-3/4} = 0.5946\ldots$

[7] Of course, if we want to generate only a Motzkin word ($h = 1$), this algorithm is less efficient than the others because it sometimes requires to use $2\log n$ integer bits (instead of $\log n$ bits)

where $F_n^h(m) = \frac{h}{n+1}\begin{pmatrix} n+1 \\ m, m+h \end{pmatrix}$ is the number of Motzkin left factor of final height $h-1$ that contains $n$ letters and $m$ letter $\bar{x}$ (ie. step $(1,-1)$). Therefore, after choosing $m$ with adequate probability, we can finish by mixing $m+h$ letters $x$ (ie. step $(1,1)$), $m$ letters $\bar{x}$ and $n+1-h-2m$ letters $z$ (ie. step $(1,0)$) and use the "Lemma Cycle" to get the final Motzkin left factor.

This give us:

$$\mathcal{R}F_n^h(m) = \frac{(n+1-h-2m)(n-h-2m)}{(m+1)(m+1+h)},$$

Now, we first need to establish some results about $F_n^h$ (this lemma is proved in the Appendix A.4):

**Lemma 7** *When $n > 1$,*

- *there exists a unique value $\tilde{m}$ in $[0 \dots \lfloor \frac{n+1-h}{2} \rfloor]$ such that $f(x) > 0$ with $x \in [0, \lfloor \frac{n+1-h}{2} \rfloor]$ if and only if $x < \tilde{m}$,*
- *$\tilde{m} = \frac{2(n+1)}{3} - \frac{h}{2} - \frac{\sqrt{4n^2+20n+28-3h^2}}{6} \leq \frac{n-h}{3}$, $M = \lceil \tilde{m} \rceil \approx \tilde{m}$,*
- *when $h < n - \frac{1}{2} - \frac{\sqrt{8n+9}}{2}$, $\tilde{m} > 1$, $M \geq 2$,*

*When $M \geq 2$, $\frac{F_n^h}{F_n^h(M)} = \Omega(\sqrt{M})$.*

## 5.1 Generation of a Motzkin left factor when $h < n - \frac{1}{2} - \frac{\sqrt{8n+9}}{2}$

We can now fix the values of $k$ and $\alpha$ in order to instantiate the Abstract Algorithm (and prove that these values give a valid algorithm with linear complexity). We choose:

- $k$ as the smallest integer such that $(k+1)M \geq (k+1)\tilde{m} \geq \frac{n+1-h}{2}$ : $k = \lceil \frac{n+1-h}{2\tilde{m}} \rceil - 1$,
- $\alpha$ as the smallest integer between 0 and $k-1$ which makes the root of $\underline{g}(x) = 0$ close of $M$ : $\alpha = max(k-1, \lfloor k - k(M - \tilde{m}) \rfloor)$.

We have:

$$\begin{aligned} \mathcal{R}\underline{G}_M^{k,\alpha}(m) &= \frac{k(n+1-h-2m)(n-h-2m)}{(m+1+h)((k+1)M+\alpha-m)} \\ \mathcal{R}\overline{G}_M^{k,\alpha}(m) &= \mathcal{R}\underline{G}_M^{k,\alpha}(m) \text{ when } m < M-1 \text{ or } m \geq M+1 \end{aligned}$$

and when $B_M^{k,\alpha}(M) = B_M^{k,\alpha}(M+1)$

$$\begin{aligned} \mathcal{R}\overline{G}_M^{k,\alpha}(M-1) &= \frac{k^2(n+3-h-2M)(n+2-h-2M)(M+1)}{(M+h)(k\,M+\alpha+1)(k\,M+\alpha)} \\ \mathcal{R}\overline{G}_M^{k,\alpha}(M) &= \mathcal{R}F_n^h(M) \end{aligned}$$

and when $B_M^{k,\alpha}(M) = B_M^{k,\alpha}(M-1)$

$$\begin{aligned} \mathcal{R}\overline{G}_M^{k,\alpha}(M-1) &= \mathcal{R}F_n^h(M-1) \\ \mathcal{R}\overline{G}_M^{k,\alpha}(M) &= \frac{k^2(n+1-h-2M)(n-h-2M)M}{(M+1+h)(k\,M+\alpha+1)(k\,M+\alpha)}. \end{aligned}$$

Using the Proposition 1, we can prove that the complexity of the algorithm is on average linear in $n$. It only remains to prove that the algorithm is valid: $\mathcal{R}G_M^{k,\alpha}(m) > 1$ iff $m < M$; this is proved with the following lemma which is proved in the Appendix A.4:

**Lemma 8** *We have when* $m \leq M-1$, $\mathcal{R}G_M^{k,\alpha}(m) > 1$ *and when* $m \geq M$, $\mathcal{R}G_M^{k,\alpha}(m) \leq 1$.

*Moreever:*

- *when* $B_M^{k,\alpha}(M) = B_M^{k,\alpha}(M+1)$, $\frac{k(n+3-h-2M)(n+2-h-2M)}{(M+h)(k\,M+\alpha)} > 1$,
- *when* $B_M^{k,\alpha}(M) = B_M^{k,\alpha}(M-1)$, $\frac{k(n+1-h-2M)(n-h-2M)}{(M+1+h)(k\,M+1+\alpha)} \leq 1$

So we can update the main loop of the previous algorithm to draw $m$ with probability $\frac{F_n^h(m)}{F_n^h}$ to get an algorithm that works when $h \leq n - \frac{1}{2} - \frac{\sqrt{8n+9}}{2}$ in average time in $O(n)$ and we only use numbers with less than $2\log_2(n)$ bits: see Algorithms 3 and 4.

**Algorithm 3** Accept $m$ with probability $\frac{F_n^h(m)}{\overline{B}_M^{k,\alpha}(m)}$ when $1 \le \delta \le k-1$:

**function** ACCEPT_M($M, k, \alpha, which, m$)
  **if** $m > (n+1-h)/2$ **then** reject $m$ **end if**
  **if** $m < M$ **then**
    ▷ accept $m$ with probability $\mathcal{R}\overline{B}_M^{h,\alpha}(M-1)/\mathcal{R}F_n^h(M-1)$
    **if** $which =$ 'M+1' **then**
      **if** $random(k(M+1)) \ge k\,M + \alpha + 1$ **then** reject $m$ **end if**
      $v \leftarrow random(k(n+3-h-2M)(n+2-h-2M))$
      **if** $v \ge (M+h)(k\,M+\alpha)$ **then** reject $m$ **end if**
    **else**
      $v \leftarrow random((n+3-h-2M)(n+2-h-2M))$
      **if** $v \ge M(M+h)$ **then** reject $m$ **end if**
    **end if**
    **for all** $i \in [m, M-2]$ **do**
      ▷ accept $m$ with probability $\mathcal{R}\overline{B}_M^{h,\alpha}(i)/\mathcal{R}F_n^h(i)$
      $v \leftarrow random(k(n+1-h-2i)(n-h-2i))$
      **if** $v \ge (i+1+h)((k+1)M+\alpha-i)$ **then** reject $m$ **end if**
    **end for**
  **else**
    ▷ accept $m$ with probability $\mathcal{R}\overline{B}_M^{h,\alpha}(M)/\mathcal{R}F_n^h(M)$
    **if** $which =$ 'M-1' **then**
      **if** $random(k\,M+\alpha) \ge k\,M$ **then** reject $m$ **end if**
      $v \leftarrow random((M+h+1)(k\,M+\alpha+1))$
      **if** $v \ge k(n+1-h-2M)(n-h-2M)$ **then** reject $m$ **end if**
    **else**
      $v \leftarrow random((M+1)(M+1+h))$
      **if** $v \ge (n+1-h-2M)(n-h-2M)$ **then** reject $m$ **end if**
    **end if**
    **for all** $i \in [M+1, m-1]$ **do**
      ▷ accept $m$ with probability $\mathcal{R}F_n^h(i)/\mathcal{R}\overline{B}_M^{h,\alpha}(i)$
      $v \leftarrow random((i+1+h)((k+1)M+\alpha-i))$
      **if** $v \ge k(n+1-h-2i)(n-h-2i)$ **then** reject $m$ **end if**
    **end for**
  **end if**
  accept $m$
**end function**

**Algorithm 4** Draw a Motzkin Left Factor with size $m$ and final height $h-1$

1: **if** $h \geq n + \frac{1}{2} - \frac{\sqrt{8n+9}}{2}$ **then** FAILS **end if** ▷ $h$ is too big
2: $M \leftarrow$ first value of $m$ which maximalizes $F_n^h(m)$
3: $\tilde{m} \leftarrow \frac{4n+4-3h-\sqrt{4n^2+20n+28-3h^2}}{6}$
4: $k \leftarrow \lceil \frac{n+1-h}{2\tilde{m}} \rceil - 1$
5: $\alpha \leftarrow max(k-1, k - \lceil k(M-\tilde{m}) \rceil)$
6: **if** $(k\,M+\alpha+1)(k\,M+\alpha) \geq k^2 M(M+1)$ **then**
7: $\quad which$ = 'M+1'
8: **else**
9: $\quad which$ = 'M-1'
10: **end if**
11: **while** true **do**
12: ▷ draw $m$ with probability $\frac{B_M^{k,\alpha}(m)}{\sum_{i=0}^{(k+1)M} B_M^{k,\alpha}(i)}$
13: $\quad m \leftarrow$ EXTENDED_BIN$(M, k, \alpha)$
14: ▷ accept $m$ with probability $F_n^h(m)/\overline{B}_M^{k,\alpha}(m)$
15: $\quad$ **if** ACCEPT_M$(M, k, \alpha, which, m)$ **then** break **end if**
16: **end while**
17: ▷ draw a Motzkin left factor of size $n$ with $m+h-1$ letters $x$ and $m$ letters $\bar{x}$

We can verify that, except when we check if $h$ is small enough (line 1) or when we compute $k$ and $\alpha$ (lines 3-5), we use only $2\log_2(n)$ bits numbers. However, it seems more difficult to completely avoid using some floating-point computations in the pre-computation phase of the algorithm (lines 1-5)[8].

## 5.2 Generation of a Motzkin left factor when $h \geq n - \frac{1}{2} - \frac{\sqrt{8n+9}}{2}$

In this case, we know that $M$ is equal to 0 or 1 ; we need to find a value $m$ in $[0, \lfloor \frac{n+1-h}{2} \rfloor]$ with an adequate probability: $\frac{F_n^h(m)}{F_n^h}$.

Let us use the 'Basic Generator' with $k = \lfloor \frac{n+1-h}{2} \rfloor \leq \frac{1}{4} + \frac{\sqrt{8n+9}}{4} = O(\sqrt{n})$. We have $\mathcal{R}G_M^k(m) = \mathcal{R}F_n^h(m)$ and therefore $\mathcal{R}G_M^k(m) > 1$ if and only if $m < M$, so the algorithm is valid.

We can apply Proposition 1 with $C_1 = O(1)$, $C_2 = O(k)$, $\frac{F(M)}{\sum_m F(m)} = O(1)$, $\frac{\sum_m B(m)}{B(M)} = k+1$ to prove that the total complexity of the main loop is, on average, $O(1+k^2) = O(n)$ as desired.

[8]The most problematic problem is the computation of $k$. In fact, it seems possible (as $M \geq 2$) to replace this computation by $k \leftarrow \lceil \frac{n+1-h}{2(M-1)} \rceil - 1$ but this would mean completely rewriting the proofs of Lemma 8. Concerning $\alpha$, it is enough to find a value of $\alpha\ in [0, k-1]$ such that $\max(\mathcal{R}G(M), \mathcal{R}G(M+1)) \leq 1 \leq \min(\mathcal{R}G(M-2), \mathcal{R}G(M-1))$, which can be done by testing all potential values in $O(k)$.

# 6 Discussion

Many recent generation algorithms use rejection methods to efficiently generate simple structures. Most of them are based on clever one-to-one correspondance. Here we try to look for methods that try to break the problem into small parts: find some $m$ with adequate probability and get a simple problem to solve. As far as we know, this method has been used to generate Motzkin words, partial injection, colored unary-binary trees [GBN10].

After adapting differently this method to Fibonacci numbers, we obtain an efficient and very easy to implement algorithm and were surprised to find that we can use it with minor modifications to generate Schroëder trees[9]. To study the limits of this kind of algorithms, we try to generate Motzkin left factors with final height $h$: a problem that seems more complicated because the number of words varies a lot: from $M_n^0 \approx \frac{3^n}{n\sqrt{n}}$ to $M_n^n = 1$ [10] ; we needed to extend the 'Binomial Generator' but we obtained a final algorithm which is efficient and can still be implemented relatively easily.

Note also that, as in [GBN10], we looked for generators that attempt to reproduce the desired distribution: same value of $M$ that maximizes the initial and final distributions, and similar "dispersion" (within a constant factor) and we found that at least for these three problems, there are some that are really simple. But is this a good idea or a misleading one? That will be the subject of a future article[Alo23]. Let's just note here that this article will allow to implement the 'Binomail generator' and the 'Extended binomial generator' more efficiently (and thus to decrease the average complexity of the algorithms presented in the article).

What more can we say? Perhaps, that the problems presented in this article correspond to numbers whose asymptotics are in $l(n)C^n$ where $C$ is a solution of a polynomial of degree 2; this probably explains why we can solve them using numbers with only $2\,log_2(n)$ bits. When $C$ is a solution of a polynomial of degree 3, ..., this method (when valid) will probably require using numbers of $3\,log_2(n)$ bits, ....

Can we do better ? For example, can we find an algorithm that uses only integers smaller than $n$ to generate a Fibonacci word of 'size' $n$ ?

**Comparaison with Gouyou-Beauchamps et all [GBN10]** In [GBN10], they present a generic algorithm for drawing a number $m$ per rejection, so that the distributions we obtained 'automatically' can be used to define their initial distribution $\overline{B}_n(m)$, a distribution that allows us to find efficient algorithms for accepting $m$ with probability $\frac{F_n(m)}{\overline{B}_n(m)}$. In fact, we can think of our algorithm as an instantiation of [GBN10] that uses a fixed method to accept a value of $m$ with

[9] In fact, it seems to give a $O(n)$ algorithm for many similar problems: Motzkin words, binary tree forest, ternary tree forest, ... which are not explained in this article and also a $O(n^{5/4})$ algorithm for partial injection

[10] also note that we have $\ln(M_n^{n-\sqrt{2n}}) \approx \frac{\sqrt{2}}{2}\sqrt{n}\ln n$ as the 'Extended Binomial Generator' only works when $h \leq n - \frac{1}{2} - \frac{\sqrt{8n+9}}{2}$

probability $\frac{F_n(m)}{\overline{B}_n(m)}$. Fixing this method allows us to quickly guess whether we have a chance of finding an efficient algorithm for certain structures by choosing a generator and checking (for certain values of $n$) whether $\mathcal{R}\overline{B}_n(m) < \mathcal{R}F_n(m)$ iff $m < M$. Of course, this also restricts the potential choices of $\overline{B}_n(m)$ but, at least for the structures we examined, we were able to quickly obtain an efficient algorithm.

We may also note that we pay a price: even though we get similar complexity, the proofs are more complicated and the algorithm used to accept a value of $m$ when generating a Motzkin left factor with $h = 0$ is less elegant than the algorithm presented in [Alo94] ; this will also be true if we try to instantiate our method with the 'Binomial Generator' to generate partial injections and compare it with [GBN10][11].

Finally, notice that there is a slight difference in the distribution used to guess the initial value of $m$,

- we choose standard binomial distributions, but we modify them slightly by decreasing the value of $\overline{B}_n(M)$ (to force $\overline{B}_n(M) = \overline{B}_n(M \pm 1)$),
- in [GBN10], they use the standard binomial distribution which explains that they need to choose a value of $\overline{B}_n(M) =$ which is sometimes larger than $F_n(M)$.

# References


[Alo94] Laurent Alonso. Uniform generation of a motzkin word. *Theoretical Computer Science*, 134(2):529–536, 1994.

[Alo23] Laurent Alonso. Uniform random generations and rejection method(ii) with trivial majorant. 2023.

[Au19] Yu Hin Au. Some properties and combinatorial implications of weighted small schröder numbers. *arXiv: Combinatorics*, 2019.

[BBJ17] Axel Bacher, Olivier Bodini, and Alice Jacquot. Efficient random sampling of binary and unary-binary trees via holonomic equations. *Theoretical Computer Science*, 695:42–53, 2017.

[BPR06] Srecko Brlek, Elisa Pergola, and Olivier Roques. Non uniform random generation of generalized motzkin paths. *Acta Informatica*, 42:603–616, 2006.

[BPS94] Elena Barcucci, Renzo Pinzani, and Renzo Sprugnoli. The random generation of directed animals. *Theor. Comput. Sci.*, 127:333–350, 1994.

[DZ90] Nachum Dershowitz and Shmuel Zaks. The cycle lemma and some applications. *European Journal of Combinatorics*, 11(1):35–40, 1990.


[11] For this problem, it is better to use the 'Extended Binomial Generator' with $k = \lceil \sqrt{n} \rceil + 1$ to choose $n - m$, which gives a valid $O(n)$ complexity algorithm that is valid when $n \geq 2$.

[DZ99] Alain Denise and Paul Zimmermann. Uniform random generation of decomposable structures using floating-point arithmetic. *Theoretical Computer Science*, 218(2):233–248, 1999.

[FZC94] Philippe Flajolet, Paul Zimmermann, and Bernard Van Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theor. Comput. Sci.*, 132:1–35, 1994.

[GBN10] Dominique Gouyou-Beauchamps and Cyril Nicaud. Random generation using binomial approximations. *Discrete Mathematics & Theoretical Computer Science*, pages 359–372, 2010.

[GKP88] Ronald L. Graham, Donald Ervin Knuth, and Oren Patashnik. Concrete mathematics. 1988.

[Map] Maplesoft, a division of Waterloo Maple Inc.. Maple.

[NW78] Albert Nijenhuis and Herbert S. Wilf. Combinatorial algorithms (second edition). 1978.

[OEI22] OEIS Foundation Inc. The On-Line Encyclopedia of Integer Sequences, 2022. Published electronically at `http://oeis.org`.

[PPPR01] Jean-Guy Penaud, Elisa Pergola, Renzo Pinzani, and Olivier Roques. Chemins de schröder et hiérarchies aléatoires. *Theoretical Computer Science*, 255(1):345–361, 2001.

# A Appendices: proofs

## A.1 Appendix: 'Extended Binomial Generator's proof

*Proof of lemma 3.* We have:

$$\frac{\underline{B}_M^{k,\alpha}(m+1)}{\underline{B}_M^{k,\alpha}(m)} = \frac{(k+1)M+\alpha-m}{k(m+1)} \leq 1$$

iff $(k+1)M+\alpha-m \leq k(m+1)$ iff $M+\frac{\alpha-k}{k+1} \leq m$ iff $M \leq m$.

Let us first prove that the relations are valid for $\underline{B}_M$. Let us note:

$$l(\alpha) = \frac{\underline{B}_M^{k,\alpha}(M)}{\sum_m \underline{B}_M^{k,\alpha}(m)} = \frac{((k+1)M+\alpha)!k^{k\,M+\alpha}}{M!(k\,M+\alpha)!(k+1)^{(k+1)M+\alpha}}.$$

Using the Stirling Formula, we get:

$$l(0) = \frac{((k+1)M)!k^{k\,M}}{M!(k\,M)!(k+1)^{(k+1)M}} \approx \frac{\sqrt{2(k+1)}}{2\sqrt{\pi\,k\,M}}$$

and more precisely by using $\sqrt{2\pi\,n}\left(\frac{n}{e}\right)^n \leq n! \leq \sqrt{2\pi\,n}\left(\frac{n}{e}\right)^n \exp\left(\frac{1}{12n}\right)$:

$$\begin{aligned} 0.84 &\leq exp\left(-\frac{1}{6}\right) \leq exp\left(-\frac{1}{12\,M}-\frac{1}{12\,k\,M}\right) \leq \frac{2\sqrt{\pi\,k\,M}l(0)}{\sqrt{2(k+1)}} \\ &\leq exp\left(\frac{1}{12(k+1)M}\right) \leq exp\left(\frac{1}{24}\right) \leq 1.05. \end{aligned}$$

We also have:

$$\frac{l(\alpha)}{l(\alpha-1)} = \frac{k((k+1)M+\alpha)}{(k+1)(k\,M+\alpha)} = 1-\frac{\alpha}{(k+1)(k\,M+\alpha)}$$

therefore we obtain: $l(\alpha) \leq l(0)$ and

$$\frac{l(\alpha)}{l(0)} \geq 1-\sum_{i=1}^{\alpha}\frac{i}{(k+1)(k\,M+i)} \geq 1-\frac{\alpha(\alpha+1)}{2(k+1)(k\,M+1)} \geq 1-\frac{k}{2(k\,M+1)} > \frac{1}{2}.$$

Finally, note that $n=(k+1)M+\alpha$, with a constant factor $(k+1)^n$, $\underline{B}_M$ corresponds to the binomial distribution with $p=\frac{1}{k+1}$, $q=1-p$. The variance of $\underline{B}_M$ is therefore $n,p,q = O\left(\frac{(M+1)(k+1)k}{(k+1)^2}\right) = O(M+1)$. Using Jensen's inequality, we obtain: $\frac{\sum_m |m-M|\underline{B}_M^{k,\alpha}(m)}{\sum_m \underline{B}_M^{k,\alpha}(m)} = O(\sqrt{M})$.

We end by noting that these relations remain valid for $B_M$ as $B_M(M) = \frac{kM}{kM+\alpha+1}\underline{B}_M(M)$ or $B_M(M) = \frac{kM+\alpha}{k(M+1)}\underline{B}_M(M)$ and thus $B_M(M) = \underline{B}_M(M)(1-O(1/M))$. □

## A.2 Appendix: Fibonacci's proofs

*Proof of Lemma 4.* Basic computations give $f'(x) = -5n + 10x + 3$, $f(0) = n^2 - 2n \geq 0$ if $n > 1$, $f(\lfloor n/2 \rfloor) = -\lceil n/2 \rceil (\lfloor n/2 \rfloor + 1) < 0$ if $n \geq 1$. This implies that there exists a unique value of $x \in [0, \lfloor n/2 \rfloor]$: $\tilde{m}$ such that $f(x) > 0$ if and only $x < \tilde{m}$.

Using $\mathcal{R}F_n(x) > 1$ if and only if $f(x) > 0$ with $\mathcal{R}F_n(m) = F_n(m+1)/F_n(m)$, we get $M = \lceil \tilde{m} \rceil$. Solving $f(\tilde{m}) = 0$ in $[0, \lfloor n/2 \rfloor]$ gives:

$$\tilde{m} = \frac{5n - 3 - \sqrt{5n^2 + 10n + 9}}{10} = \frac{5 - \sqrt{5}}{10} n - \frac{3 + \sqrt{5}}{10} + O(1/n) \approx \frac{5 - \sqrt{5}}{10} n.$$

Suppose that $\epsilon \geq 0$ and $n > 2$, we have:

$$\begin{aligned} \mathcal{R}F_n(\tilde{m} + \epsilon) &= \frac{(n - 2\tilde{m})(n - 1 - 2\tilde{m})}{(\tilde{m} + 1)(n - \tilde{m})} \frac{(1 - 2\frac{\epsilon}{n - 2\tilde{m}})(1 - 2\frac{\epsilon}{n - 1 - 2\tilde{m}})}{(1 + \frac{\epsilon}{\tilde{m}+1})(1 - \frac{\epsilon}{n - \tilde{m}})} \\ &\geq 1(1 - 2\frac{\epsilon}{n - 2\tilde{m}} - 2\frac{\epsilon}{n - 1 - 2\tilde{m}} - \frac{\epsilon}{\tilde{m} + 1} + 0) \geq 1 - \frac{5\epsilon}{\tilde{m}} \end{aligned}$$

by noting that $\tilde{m} \leq \frac{n-1}{3}$ (and so $n - 1 - 2\tilde{m} \geq \tilde{m}$). Therefore, when $i \geq 0$, we get $\mathcal{R}F_n(M + i) \geq 1 - \frac{5i}{\tilde{m}}$ and when $i < \frac{\tilde{m}}{5}$, $\frac{F_n(M+i)}{F_n(M)} \geq \prod_{j=0}^{i-1}(1 - \frac{5\,(j+1)}{\tilde{m}}) \geq 1 - \frac{5i(i+1)}{2\tilde{m}}$. This implies that when $0 \leq i \leq \frac{\sqrt{\tilde{m}-1}}{\sqrt{5}}$, $F_n(M + i) \geq \frac{1}{2}F(M)$ and thus $F_n = \Omega(\sqrt{M})F_n(M) = \Omega(\sqrt{n})F_n(M)$. □

*Proof of Lemma 5.* We have $\mathcal{R}\overline{B}_M(M-1) = \mathcal{R}\overline{B}_M(M) = 1$ and using the previous lemma, $\mathcal{R}F_n(M-1) > 1$, $\mathcal{R}F_n(M) \leq 1$. This validates the inequality for $m = M - 1$ and $m = M$.

For other values of $m$, we have $\mathcal{R}\overline{B}_M(m) < \mathcal{R}F_n(m)$ if and only if $\underline{g}(m) = (n-2m)(n-2m-1) - (2M-m)(n-m) > 0$. We have $\underline{g}(0) = n(n-2M-1) > 0$, $\underline{g}(n/2-1) = -(2M+1-n/2)(n/2-1) < 0$ when $n > 2$, $\underline{g}'(x) = 6x - 3n + 2M + 2$. This implies that when $x \in [0, n/2-1]$, there exists a value $\tilde{x}$ such that $\underline{g}(x) \geq 0$ iff $x \leq \tilde{x}$.

It is therefore sufficient to prove that $\underline{g}(M-2) > 0$ and $\underline{g}(M+1) \leq 0$.

Using $M = \lceil \tilde{m} \rceil$, we obtain (as $\frac{\delta \underline{g}(M-2)}{\delta M} < 0$):

$$\begin{aligned} \underline{g}(M-2) &= 5M^2 - (5n+14)M + n^2 + 5n + 8 \\ &\geq 5(\tilde{m}+1)^2 - (5n+14)(\tilde{m}+1) + n^2 + 5n + 8 \\ &= \frac{11}{10} - \frac{3n}{2} + \frac{7\sqrt{5n^2 + 10n + 9}}{10} > 0. \end{aligned}$$

Similarly, we get (as $\frac{\delta \underline{g}(M+1)}{\delta M} < 0$ when $n > 6$):

$$\begin{aligned} \underline{g}(M+1) &= 5M^2 - (5n-10)M + n^2 - 4n + 5 \\ &\leq 5\tilde{m}^2 - (5n-10)\tilde{m} + n^2 - 4n + 5 \\ &= \frac{29}{10} + \frac{3n}{2} - \frac{7\sqrt{5n^2 + 10n + 9}}{10}, \end{aligned}$$

which is less than or equal to 0 when $n \geq 20$.

Finally, we can calculate the value of $\underline{g}(M+1)$ when $n < 20$, we find:

| $n$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| $M$ | 0 | 0 | 0 | 1 | 1 | 1 | 2 | 2 | 2 | 2 |
| $\underline{g}(M+1)$ | 5 | 2 | 1 | 2 | 0 | 0 | -3 | -4 | -3 | 0 |

| $n$ | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|
| $M$ | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 5 | 5 | 5 |
| $\underline{g}(M+1)$ | -10 | -8 | -4 | -18 | -15 | -10 | -3 | -24 | -18 | -10 |

which ends the proof by noting that when $n \leq 3$, there exist no word with $M+2$ letters $|b|$. □

## A.3 Appendix: Schröder's proof

*Proof of Lemma 6.* Indeed, $\mathcal{R}F_n(x) > 1$ if and only if $f(x) = (n-x)(n+1+x) - (x+1)(x+2) > 0$. Basic computations give $f'(x) = -4x - 4$, $f(0) = n(n+1) - 2 \geq 0$ if $n \geq 1$, $f(n) = -(n+1)(n+2) < 0$. This implies that exists a unique value of $x \in [0, n]$: $\tilde{m}$ such that $f(x) > 0$ if and only $x < \tilde{m}$.

The relation $\mathcal{R}F_n(x) > 1$ if and only if $f(x) > 0$ with $\mathcal{R}F_n(m) = F_n(m+1)/F_n(m)$ gives $M = \lceil \tilde{m} \rceil$. By solving $f(\tilde{m}) = 0$ in $[0, n]$, we obtain:

$$\tilde{m} = -1 + \frac{\sqrt{2n^2+2n}}{2} = \frac{n}{\sqrt{2}} - \frac{4-\sqrt{2}}{4} + O(1/n) \approx \frac{n}{\sqrt{2}}.$$

We can also note that $\tilde{m} < \frac{n}{\sqrt{2}}$ (and so $n - \tilde{m} \geq (\sqrt{2}-1)\tilde{m}$).

To prove that $F_n = \Omega(\sqrt{n})F_n(M)$, we can imitate the proof of Lemma 4. Indeed, assuming that $\epsilon \geq 0$, we have :

$$\mathcal{R}F_n(\tilde{m}+\epsilon) = \frac{(1-\frac{\epsilon}{n-\tilde{m}})(1-\frac{\epsilon}{n+1-\tilde{m}})}{(1+\frac{\epsilon}{\tilde{m}+1})(1+\frac{\epsilon}{\tilde{m}+2})} \geq 1 - \frac{7\epsilon}{\tilde{m}}.$$

We obtain therefore when $0 \leq i < \frac{\tilde{m}}{7}$, $\frac{F_n(M+i)}{F_n(M)} \geq 1 - \frac{7\,i(i+1)}{2\tilde{m}}$, which gives us: $F_n = \Omega(\sqrt{n})F_n(M)$.

We have $\mathcal{R}\overline{B}_M(M-1) = \mathcal{R}\overline{B}_M(M) = 1$ and $\mathcal{R}F_n(M-1) > 1$, $\mathcal{R}F_n(M) \leq 1$. This validates the last inequality for $m = M-1$ and $m = M$.

For other values of $m$, we have $\mathcal{R}\overline{B}_M(m) < \mathcal{R}F_n(m)$ if and only if $\underline{g}(m) = (n-m)(n+1+m) - (2M-m)(m+2) > 0$. We have $\underline{g}(0) = n^2+n-4M > 0$, $\underline{g}(n) = -(2M-n)(n+2) < 0$ (when $n > 3$), $\underline{g}'(x) = 1 - 2M$. This implies that when $x \in [0, n]$, there exists a value $\tilde{x}$ such that $\underline{g}(x) \geq 0$ iff $x \leq \tilde{x}$. It is therefore sufficient to prove that $\underline{g}(M-2) > 0$ and $\underline{g}(M+1) \leq 0$.

Using $M = \lceil \tilde{m} \rceil$, we obtain

$$\begin{aligned} \underline{g}(M-2) &= -2M^2 + n^2 + M + n - 2 > -2(\tilde{m}+1)^2 + n^2 + (\tilde{m}+1) + n - 2 \\ &= \frac{\sqrt{2n^2+2n} - 4}{2} \end{aligned}$$

which is greater than 0 when $n \geq 3$.

Similarly,

$$\underline{g}(M+1) = -2M^2+n^2-5M+n+1 \leq -2\tilde{m}^2+n^2-5\tilde{m}+n+1 = \frac{8-\sqrt{2n^2+2n}}{2}$$

which is less than or equal to 0 when $n \geq 6$.

Finally, we can compute the values of $\underline{g}(M-2)$ and $\underline{g}(M+1)$ when $n < 6$, we find:

| $n$ | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| $M$ | -1 | 0 | 1 | 2 | 3 | 3 |
| $\underline{g}(M-2)$ | -5 | 0 | 3 | 4 | 3 | 13 |
| $\underline{g}(M+1)$ | 4 | 3 | 0 | -5 | -12 | -2 |

□

## A.4 Appendix: Motzkin left factor's proofs

*Proof of Lemma 7.* Indeed, $\mathcal{R}F_n^h(x) > 1$ if and only if $f(x) = (n-h-2x)(n+1-h-2x) - (x+1)(x+1+h) > 0$. Basic computations give $f'(x) = -4n+6x+3h-4$, $f(0) = (n-h)^2+n-2h-1 > 0$ if $n \geq 1$, $f(\lfloor \frac{n+1-h}{2} \rfloor) = -(\lfloor \frac{n+1-h}{2} \rfloor + 1)(\lfloor \frac{n+1-h}{2} \rfloor + 1 + h) < 0$. This implies that exists a unique value of $x \in [0, \lfloor \frac{n+1-h}{2} \rfloor]$: $\tilde{m}$ such that $f(x) > 0$ if and only $x < \tilde{m}$.

The relation $\mathcal{R}F_n^h(x) > 1$ if and only if $f(x) > 0$ with $\mathcal{R}F_n^h(m) = F_n^h(m+1)/F_n^h(m)$ gives $M = \lceil \tilde{m} \rceil$. Solving $f(\tilde{m}) = 0$ in $[0, n]$ gives:

$$\tilde{m} = \frac{2(n+1)}{3} - \frac{h}{2} - \frac{\sqrt{4n^2+20n+28-3h^2}}{6} \leq \frac{n-h}{3} \leq \frac{n+1-h}{3},$$

so $\tilde{m} = 1$ corresponds to $h = n - \frac{1}{2} + \frac{\sqrt{8n+9}}{2}$ which implies $M \geq 2$ when $n - \frac{1}{2} + \frac{\sqrt{8n+9}}{2}$ because $\frac{\delta \tilde{m}}{\delta h} < 0$.

Now suppose that $\epsilon \geq 0$, we have:

$$\mathcal{R}F_n^h(\tilde{m}+\epsilon) = \frac{\left(1-\frac{2\epsilon}{n-h-2\tilde{m}}\right)\left(1-\frac{2\epsilon}{n+1-h-2\tilde{m}}\right)}{\left(1+\frac{\epsilon}{\tilde{m}+1}\right)\left(1+\frac{\epsilon}{\tilde{m}+h+1}\right)} \geq 1 - \frac{6\epsilon}{\tilde{m}}$$

so when $0 \leq i < \frac{\tilde{m}}{6}$, we get $\frac{F_n^h(M+i)}{F_n^h(M)} \geq 1 - \frac{3i(i+1)}{\tilde{m}}$ which gives us $F_n^h = F_n^h(M)\Omega(\sqrt{\tilde{m}}) = F_n^h(M)\Omega(\sqrt{M})$ when $M \geq 2$. □

*Proof of Lemma 8 when $m \leq M-2$ or $m \geq M+1$.* We have $\underline{g}(-h-1) = k(n+3+h)(n+2+h) > 0$ and

$$\underline{g}\left(\frac{n-h}{2}\right) = -\frac{n+h+2}{2}\left((k+1)M + \alpha - \frac{n-h}{2}\right) < 0$$

as $(k+1)M \geq (k+1)\tilde{m} \geq \frac{n+1-h}{2}$.

We have:

$$\underline{g}'(x) = (8k+2)x - (4n+2-4h+M)k - M - \alpha + h + 1,$$

so $\underline{g}$ has a unique root $\tilde{x}$ in $\left[-h-1, \frac{n-h}{2}\right[$, and we have $\underline{g}(x) > 0$ if $x < \tilde{x}$ and $\underline{g}(x) < 0$ if $x > \tilde{x}$.

We have $\alpha = max(k-1, \lfloor k - k(M-\tilde{m})\rfloor)$, so:

$$(k+1)\tilde{m}+k-1 \leq M+k(\tilde{m}+1)-1 \leq (k+1)M+\alpha \leq M+k(\tilde{m}+1) < (k+1)\tilde{m}+k+1.$$

But:

$$\begin{aligned} \underline{g}(\tilde{m}) &= k\, f(\tilde{m}) - (\tilde{m}+1+h)((k+1)M+\alpha-\tilde{m}) + k(\tilde{m}+1+h)(\tilde{m}+1) \\ &= (\tilde{m}+1+h)((k+1)\tilde{m}+k-((k+1)M+\alpha)) \end{aligned}$$

and

$$g_1(x) = \underline{g}(x+1) - \underline{g}(x) = -((k+1)M+\alpha) - 2(2n-2h-4x-1) + h + 2x + 2.$$

Therefore when $h < n-1$ (which is always true) as $4h-4n+7\tilde{m}+1 < 0$:

$$\begin{aligned} g(\tilde{m}+1) &= g(\tilde{m}) + g_1(\tilde{m}) \leq (4h-4n+7\tilde{m}+1)k + 2h + 2\tilde{m} + 4 \\ &\leq \frac{-10\tilde{m}^2 + (15n-11h+13)\tilde{m} - (4n-4h-1)(n-h+1) + 4\, f(\tilde{m})}{2\tilde{m}} \\ &= \frac{2\tilde{m}^2 + (h-n-3)\tilde{m} - 5h + n - 3}{2\tilde{m}} \end{aligned}$$

which is negative when $n \geq 9$, because $\tilde{m} \leq \frac{n-h}{3} \leq \frac{n-h+3+\sqrt{(n-h)^2+34h-2n+33}}{4}$. Finally, when $n \leq 8$, we check that $2\tilde{m}^2 + (h-n-3)\tilde{m} - 5h + n - 3 < 0$ for all possible values of $h$.

Similarly, as $4n-4h-7\tilde{m}+7 > 0$, we have:

$$\begin{aligned} g(\tilde{m}-1) &= g(\tilde{m}) - g_1(\tilde{m}-1) \geq (4n-4h-7\tilde{m}+7)k - 2h - 2\tilde{m} \\ &\geq \frac{-4\tilde{m}^2 - (7n-3h+7)\tilde{m} + (4n-4h+7)(n+1-h) - 4\, f(\tilde{m})}{2\tilde{m}} \\ &= \frac{-16\tilde{m}^2 + 9(n-h+1)\tilde{m} - 3h + 7n + 11}{2\tilde{m}} \end{aligned}$$

which is positive because $\tilde{m} \leq \frac{n-h}{3} \leq \frac{9n-9h+9+\sqrt{81(n-h)^2-354h+610n+785}}{32}$.

So we get $\underline{g}(M-2) \geq \underline{g}(\tilde{m}-1) > 0$ and $\underline{g}(M+1) \leq \underline{g}(\tilde{m}+1) < 0$. This is equivalent to $\mathcal{R}G_M^{k,\alpha}(m) > 1$ if $m \leq M-2$ and $\mathcal{R}G_M^{k,\alpha}(m) < 1$ if $m \geq M+1$. □

*Proof of Lemma 8 when $m = M$ or $m = M+1$.* Note first that when $B_M^{k,\alpha}(M) = B_M^{k,\alpha}(M-1)$, $\mathcal{R}G_M^{k,\alpha}(M-1) = \mathcal{R}F_n^h(M-1) > 1$ while when $B_M^{k,\alpha}(M) = B_M^{k,\alpha}(M+1)$, $\mathcal{R}G_M^{k,\alpha}(M) = \mathcal{R}F_n^h(M) \leq 1$.

We also have as $0 \leq \alpha \leq k-1$:

$$\frac{k^2(n+3-h-2M)(n+2-h-2M)(M+1)}{(M+h)(k\,M+\alpha+1)(k\,M+\alpha)} \geq \frac{k(n+3-h-2M)(n+2-h-2M)}{(M+h)(k\,M+\alpha)},$$

and:

$$\frac{k^2(n+1-h-2M)(n-h-2M)M}{(M+h+1)(k\,M+\alpha+1)(k\,M+\alpha)} \leq \frac{k(n+1-h-2M)(n-h-2M)}{(M+h+1)(k\,M+1+\alpha)},$$

which turn into equalities when $k=1$ and $\alpha=0$. Moreover, when $k=1$, we have $\alpha=0$ and $B_M^{k,\alpha}(M) = B_M^{k,\alpha}(M-1) = B_M^{k,\alpha}(M+1)$ ; we can thus conclude directly.

We can now note that $\alpha = max(k-1, \lfloor k-k(M-\tilde{m})\rfloor)$ implies that $\tilde{m} + \frac{k-1-\alpha}{k} \leq M \leq \tilde{m} + \frac{k-\alpha}{k}$.

When $B_M^{k,\alpha}(M) = B_M^{k,\alpha}(M+1)$ (which is equivalent to $(kM+\alpha+1)(kM+\alpha) \geq k^2M(M+1)$ and implies $\alpha > 0$), we have:

$$\mathcal{R}\overline{G}_M^{k,\alpha}(M-1) \geq \frac{k(n+3-h-2M)(n+2-h-2M)}{(M+h)(k\,M+\alpha)}$$

which is greater than 1 if $g_1(M) = k(n+3-h-2M)(n+2-h-2M) - (M+h)(k\,M+\alpha) > 0$. We have $g_1'(x) = (3h-4n+6x-10)k - \alpha < 0$ if $x < \frac{n+1-h}{2} + \frac{k(n+7)+\alpha}{6k}$. Therefore:

$$\begin{aligned} g_1(M) &\geq g_1\left(m + \frac{k-\alpha}{k}\right) \\ &= \alpha\frac{k(7\sqrt{-3h^2+4n^2+20n+28} - 3h - 4n - 10) + 24\alpha}{6k} \\ &\geq \alpha\frac{7\sqrt{-3h^2+4n^2+20n+28} - 3h - 4n - 10}{6} > 0 \end{aligned}$$

when $h \leq n+1 < \frac{-2n-5+7\sqrt{16n^2+80n+113}}{26}$.

When $B_M^{k,\alpha}(M) = B_M^{k,\alpha}(M-1)$ (which implies $\alpha < k-1$), we have:

$$\mathcal{R}\overline{G}_M^{k,\alpha}(M) \leq \frac{k(n+1-h-2M)(n-h-2M)}{(M+h+1)(k\,M+1+\alpha)}$$

which is less than or equal to 1 if $g_1(M) = k(n+1-h-2M)(n-h-2M) - (M+1+h)(k\,M+\alpha+1) \leq 0$. We have $g_1'(x) = (3h-4n+6x-3)k - \alpha - 1 < 0$ if $x < \frac{n+1-h}{2} + \frac{k\,n+\alpha+1}{6k}$. Therefore:

$$\begin{aligned} g_1(M) &\leq g_1\left(m + \frac{k-\alpha-1}{k}\right) \\ &= -(k-\alpha-1)\frac{k(7\sqrt{-3h^2+4n^2+20n+28} - 3h - 4n - 34) + 24 + 24\alpha}{6k} \\ &\leq -(k-\alpha-1)\frac{7\sqrt{-3h^2+4n^2+20n+28} - 3h - 4n - 34}{6} < 0 \end{aligned}$$

when $h \leq n+1 < \frac{-2n-17+7\sqrt{16n^2+64n+25}}{26}$ and $n > 0$. □