HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

ANALYZING THE STOCHASTIC COMPLEXITY VIA TREE POLYNOMIALS

Petri Kontkanen, Petri Myllymäki

June 3, 2005

HIIT TECHNICAL REPORT 2005–4

ANALYZING THE STOCHASTIC COMPLEXITY VIA TREE POLYNOMIALS

Petri Kontkanen, Petri Myllymäki

Helsinki Institute for Information Technology HIIT

Tammasaarenkatu 3, Helsinki, Finland

PO BOX 9800

FI-02015 TKK, Finland

http://www.hiit.fi

HIIT Technical Reports 2005–4

ISSN 1458-9478

URL: http://cosco.hiit.fi/Articles/hiit-2005-4.pdf

NB. The HIIT Technical Reports series is intended for rapid dissemination of results produced by the HIIT researchers. Therefore, some of the results may also be later published as scientific articles elsewhere.

Analyzing the Stochastic Complexity via Tree Polynomials

Petri Kontkanen, Petri Myllymäki
Complex Systems Computation Group (CoSCo)
Helsinki Institute for Information Technology (HIIT)
University of Helsinki & Helsinki University of Technology
P.O. Box 9800, FIN-02015 HUT, Finland.
{Firstname}.{Lastname}@hiit.fi

Abstract

Stochastic complexity of a data set is defined as the shortest possible code length for the data obtainable by using some fixed set of models. This measure is of great theoretical and practical importance as a tool for tasks such as model selection and data clustering. Unfortunately straightforward computation of this measure requires exponential time with respect to the size of the data. The purpose of this paper is to analyze the connection between the stochastic complexity and so-called tree polynomials, and to show that this connection provides a novel, useful framework for understanding the theoretical and practical properties of the stochastic complexity. As a concrete example of the usefulness of this framework, we show how the tree polynomials can be used to derive recursion formulas for efficient computation of the stochastic complexity in the case of n observations of a single multinomial random variable with K values. The time complexity of the new formula is $\mathcal{O}(n+K)$ as opposed to $\mathcal{O}\left(n^2 \log K\right)$ obtained with previous results. Moreover, we present an approximation algorithm which works in time $\mathcal{O}(K)$, while still providing extremely accurate results with large sample sizes.

1 Introduction

Minimum encoding length principle performs induction by seeking a theory that allows the most compact encoding of both the theory and available data. Intuitively speaking, this approach can be argued to produce the best possible model of the problem domain, since in order to be able to produce the most efficient coding, one must capture all the regularities present in the domain. Consequently, the minimum encoding approach can be used for constructing a solid theoretical framework for statistical modeling.

The most well-founded formalization of the minimum encoding approach is the Minimum Description Length (MDL) principle developed by Rissanen [1, 2, 3]. The main idea of this principle is to represent a set of models (model class) by a single model imitating the behaviour of any model in the class. Such representative models are called *universal*. The universal model itself does not have to belong to the model class as often is the case. Unlike some other approaches, like for example Bayesianism, the MDL principle does not assume that the chosen model class is correct. It even says that there is no such thing as a true model or model class, as acknowledged by many practitioners. The model class is only used as a technical device for constructing an efficient code. For more discussion on the theoretical motivations behind the MDL see, e.g., [3, 4, 5, 6, 7, 8].

The most important notion of the MDL is the stochastic complexity, which is the shortest description length of a given data relative to a given model class (set of models). The exact definition of the stochastic complexity is based on the Normalized Maximum Likelihood (NML) distribution described in [9, 3]. For multinomial (discrete) data, this definition involves an exponential sum over all the possible data matrices of fixed size. This sum is called the *regret*, and it can be interpreted as a penalty term for the complexity of the model class.

Our previous work [10, 11, 12] focused on finding efficient algorithms for computing the regret. Although we were able to remove the exponentiality of the regret sum, even fastest of the algorithms was still superlinear with respect to the size of the data, which makes the algorithms infeasible for large or even moderate size data sets. The topic of this paper is to present a novel theoretical framework for analyzing the properties of the regret. This framework is based on the *tree polynomials* introduced in [13]. We will apply the properties of these polynomials to derive a linear time regret computation algorithm for multinomial data. Furthermore, we will construct an alternative proof for the recursive regret formula discussed earlier in [10, 11].

In Section 2 we introduce the notation and instantiate the NML distribution for the multinomial model class. We will also shortly discuss our previous work on the regret computation methods. The topic of Section 3 is to derive the generating function for the sequence of regret terms. This generating function turns out to be the key element for finding the connection between the tree polynomials and the regret, which is discussed in Section 4. The definition and basic properties of the tree polynomials are also reviewed there. Furthermore, at the end of the section we will derive the new regret computation algorithms. Finally, Section 5 gives the concluding remarks and presents some ideas for future work.

2 Multinomial NML

Consider a discrete data set (or matrix) $\mathbf{x}^n = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ of n outcomes, and let $\hat{\theta}(\mathbf{x}^n)$ denote the maximum likelihood estimate of data $\mathbf{x}^n$. The minimax optimal *Normalized Maximum Likelihood (NML)* distribution [9] is now defined as

$$P_{NML}(\mathbf{x}^n \mid \mathcal{M}) = \frac{P(\mathbf{x}^n \mid \hat{\theta}(\mathbf{x}^n), \mathcal{M})}{\mathcal{R}^n_{\mathcal{M}}}, \tag{1}$$

where the *regret* $\mathcal{R}^n_{\mathcal{M}}$ relative to a model class $\mathcal{M}$ is given by

$$\mathcal{R}^n_{\mathcal{M}} = \sum_{\mathbf{x}^n} P(\mathbf{x}^n \mid \hat{\theta}(\mathbf{x}^n), \mathcal{M}), \tag{2}$$

and the sum goes over all the possible data matrices of size n. Definition (1) is intuitively very appealing: every data matrix is modeled using its own maximum likelihood (i.e., best fit) model, and then a penalty for the complexity of the model class $\mathcal{M}$ is added to normalize the distribution.

Our previous work on regret computation methods [10, 11, 12] focused on two model classes. In the first case, the problem domain consisted of a single multinomial variable. The second one was a multi-dimensional generalization, which was shown to be suitable for, e.g. data clustering. In the following, we will concentrate on the single-dimensional case. We believe, however, that the methods presented in the current work will also generalize to the multi-dimensional case.

In the single-dimensional case we assume that our problem domain consists of a discrete random variable X with K values, and that our data $\mathbf{x}^n$ is multinomially distributed. The corresponding model class, which we denote by $\mathcal{M}_K$, is defined via the following simplex-shaped parameter space:

$$\{(\theta_1, \ldots, \theta_K) : \theta_v \geq 0, \theta_1 + \cdots + \theta_K = 1\}, \text{ where } \theta_v = P(X = v),\ v = 1, \ldots, K. \quad (3)$$

The NML distribution for the $\mathcal{M}_K$ model class is given by (see, e.g., [10, 11])

$$P_{NML}(\mathbf{x}^n \mid \mathcal{M}_K) = \frac{\prod_{v=1}^{K} \left(\frac{h_v}{n}\right)^{h_v}}{\mathcal{R}^n_{\mathcal{M}_K}}, \quad (4)$$

where h_v is the frequency of value v in $\mathbf{x}^n$, and

$$\mathcal{R}^n_{\mathcal{M}_K} = \sum_{\mathbf{x}^n} P(\mathbf{x}^n \mid \hat{\theta}(\mathbf{x}^n), \mathcal{M}_K) \quad (5)$$

$$= \sum_{h_1 + \cdots + h_K = n} \frac{n!}{h_1! \cdots h_K!} \left(\frac{h_1}{n}\right)^{h_1} \cdots \left(\frac{h_K}{n}\right)^{h_K}. \quad (6)$$

Looking at (5) it is clear that the regret $\mathcal{R}^n_{\mathcal{M}_K}$ involves a sum over K^n terms. In [10, 11] we presented a recursion formula for removing the exponentiality of (5):

$$\mathcal{R}^n_{\mathcal{M}_{K_1+K_2}} = \sum_{k=0}^{n} \frac{n!}{k!(n-k)!} \left(\frac{k}{n}\right)^k \left(\frac{n-k}{n}\right)^{n-k} \mathcal{R}^k_{\mathcal{M}_{K_1}} \mathcal{R}^{n-k}_{\mathcal{M}_{K_2}}, \quad (7)$$

which holds for all $K_1, K_2 \geq 1$. A straightforward algorithm based on this formula from [10, 11] was used to compute the regret $\mathcal{R}^n_{\mathcal{M}_K}$ in time $\mathcal{O}\left(n^2 \log K\right)$. In [14, 12, 15] this was improved to $\mathcal{O}\left(n \log n \log K\right)$ by writing the regret as a convolution sum and then using the Fast Fourier algorithm. However, the practical relevance of the latter result is unclear due to numerical instability problems it easily produces.

Although our previous regret computation algorithms are efficient enough to allow the use of NML for moderately sized datasets, they are still superlinear with respect to n. In the next two sections we will introduce a novel framework for analyzing the properties of the multinomial regret, which will eventually lead to a linear time algorithm.

3 The Regret Generating Function

The relation between the regret terms and tree polynomials is based on the similarity of their generating functions. In this section we will start the discussion of this relation by finding the generating function for the sequence of regret terms. Our derivation mostly follows [16], where a similar problem was studied.

One of the most powerful ways to analyze a sequence of numbers is to form a power series with the elements of the sequence as coefficients. The resulting function is called the *generating function* of the sequence. Generating functions can be seen as a bridge between discrete mathematics and continuous analysis. They can be used for finding recurrence formulas and asymptotic expansions, proving combinatorial identities and finding statistical properties of a sequence. Good sources for further reading on generating functions are [17, 18, 19, 20].

The (ordinary) generating function of a sequence $\langle a_n \rangle = (a_0, a_1, a_2, \ldots)$ is defined as a series

$$A(z) = \sum_{n \geq 0} a_n z^n, \quad (8)$$

where z is a dummy symbol (or a complex variable). The importance of generating functions is that the function $A(z)$ is a representation of the whole sequence $\langle a_n \rangle$. By studying this function we can get important information about the sequence, such as the asymptotic form of the coefficients.

To start the derivation of the generating function for the regret terms let us consider the sequence $\langle n^n/n! \rangle$. As in [16], we denote the function generating this sequence by $B(z)$. Although there is no closed-form formula for $B(z)$, we will soon see that this function is nevertheless a suitable starting point. In order to find the connection between $B(z)$ and the regret terms, we square $B(z)$, which yields

$$\begin{aligned}
B^2(z) &= \left(\sum_{h_1 \geq 0} \frac{h_1^{h_1}}{h_1!} z^{h_1} \right) \cdot \left(\sum_{h_2 \geq 0} \frac{h_2^{h_2}}{h_2!} z^{h_2} \right) && (9) \\
&= \sum_{h_1, h_2 \geq 0} \frac{h_1^{h_1} h_2^{h_2}}{h_1! h_2!} z^{h_1 + h_2} && (10) \\
&= \sum_{n \geq 0} \left(\sum_{h_1 + h_2 = n} \frac{n^n}{n!} \frac{n!}{h_1! h_2!} \frac{h_1^{h_1} h_2^{h_2}}{n^{h_1 + h_2}} \right) z^n && (11) \\
&= \sum_{n \geq 0} \frac{n^n}{n!} \left(\sum_{h_1 + h_2 = n} \frac{n!}{h_1! h_2!} \left(\frac{h_1}{n} \right)^{h_1} \left(\frac{h_2}{n} \right)^{h_2} \right) z^n && (12) \\
&= \sum_{n \geq 0} \frac{n^n}{n!} \mathcal{R}^n_{\mathcal{M}_2} z^n, && (13)
\end{aligned}$$

where the last equality follows from (5). Thus, we have proven that the function $B^2(z)$ generates the sequence $\langle \frac{n^n}{n!} \mathcal{R}^n_{\mathcal{M}_2} \rangle$. It is now straightforward to generalize this to

$$\begin{aligned}
B^K(z) &= \sum_{n \geq 0} \frac{n^n}{n!} \left[\sum_{h_1 + \cdots + h_K = n} \frac{n!}{h_1! \cdots h_K!} \left(\frac{h_1}{n} \right)^{h_1} \cdots \left(\frac{h_K}{n} \right)^{h_K} \right] z^n && (14) \\
&= \sum_{n \geq 0} \frac{n^n}{n!} \mathcal{R}^n_{\mathcal{M}_K} z^n, && (15)
\end{aligned}$$

which generates $\langle \frac{n^n}{n!} \mathcal{R}^n_{\mathcal{M}_K} \rangle$. Consequently, $B^K(z)$ is the sought-for function and we call it the *regret generating function*.

As mentioned above, there is no closed-form formula for $B(z)$ and little is known about the function. Therefore, as in [16], we will write the function $B^K(z)$ in a different, more useful form using the so-called *Cayley's tree function* $T(z)$ [21, 13, 22, 23], which generates the sequence $\langle n^{n-1}/n! \rangle$, i.e.,

$$T(z) = \sum_{n \geq 1} \frac{n^{n-1}}{n!} z^n. \tag{16}$$

This sequence counts the *rooted labeled trees*, hence the name of the function. A basic property of the tree function is the functional equation

$$T(z) = z e^{T(z)}, \tag{17}$$

which we will make use of below.

To show the connection between $T(z)$ and $B(z)$, following [16] we first differentiate and multiply (16) by z, which gives

$$zT'(z) = z \cdot \sum_{n \geq 1} \frac{n \cdot n^{n-1}}{n!} z^{n-1} \tag{18}$$

$$= \sum_{n \geq 0} \frac{n^n}{n!} z^n - 1, \tag{19}$$

from which it easily follows that

$$B(z) = zT'(z) + 1. \tag{20}$$

On the other hand, differentiating the functional equation (17) yields

$$zT'(z) = \frac{T(z)}{1 - T(z)}. \tag{21}$$

Finally, combining (20) and (21) leads to

$$B(z) = \frac{T(z)}{1 - T(z)} + 1 = \frac{1}{1 - T(z)}, \tag{22}$$

and thus the regret generating function $B^K(z)$ can be written as

$$B^K(z) = \frac{1}{(1 - T(z))^K}. \tag{23}$$

4 Tree Polynomials

The topic of this section is to present the elegant connection between the multinomial regret terms and so-called *tree polynomials* introduced in [13]. This connection has both theoretical and practical merits. On the theoretical side, it provides another framework or view for analyzing the properties of multinomial NML. An immediate practical consequence of the connection is the linear time algorithm for computing the regret derived at the end of this section.

We start the discussion by defining the tree polynomials t_n [13] by the means of their generating function

$$\frac{1}{(1 - T(z))^y} = \sum_{n \geq 0} t_n(y) \frac{z^n}{n!}, \tag{24}$$

where y is a real number and $T(z)$ is the tree function discussed in the previous section. The coefficient of y^k in $t_n(y)$ has a combinatorial interpretation: it is the number of mappings from an n-element set into itself having exactly k cycles. In practice, the tree polynomials can be found via recurrence

$$t_n(y - 2) = \frac{y - 2}{n} \sum_{k=1}^{n} \binom{n}{k} k^{k-1} t_{n-k}(y), \tag{25}$$

from which t_n can be obtained when $t_{n-1}, \ldots, t_0$ are known [13]. The first few tree polynomials are given by

$$t_0(y) = 1, \tag{26}$$

$$t_1(y) = y, \tag{27}$$

$$t_2(y) = y^2 + 3y, \tag{28}$$

$$t_3(y) = y^3 + 9y^2 + 17y, \tag{29}$$

$$t_4(y) = y^4 + 18y^3 + 95y^2 + 142y. \tag{30}$$

The connection to the multinomial regret terms comes from the fact that for positive integer values of y, the generating functions (24) and (23) are the same. Thus, by comparing the coefficients of z^n in both cases we get

$$\frac{n^n}{n!}\mathcal{R}^n_{\mathcal{M}_K} = \frac{t_n(K)}{n!} \tag{31}$$

$$\mathcal{R}^n_{\mathcal{M}_K} = \frac{t_n(K)}{n^n}, \tag{32}$$

for positive integers K. Consequently, the nth tree polynomial t_n is an elegant, compact representation of the infinite regret sequence $(\mathcal{R}^n_{\mathcal{M}_1}, \mathcal{R}^n_{\mathcal{M}_2}, \ldots)$, i.e. this polynomial can be used to evaluate the regret for all the values of K when n is fixed.

Having now established the connection, we can use all the properties of the tree polynomials for finding new facts about the regret terms. As a first example, we will consider an interesting relation between the binary regret and Ramanujan's Q-function [13, 23], which is used in the analysis of algorithms and several discrete probability problems, such as hashing, union-find algorithms, caching and integer factorization. We have

$$\mathcal{R}^n_{\mathcal{M}_2} = 1 + Q(n), \tag{33}$$

where

$$Q(n) = 1 + \frac{n-1}{n} + \frac{(n-1)(n-2)}{n^2} + \cdots \tag{34}$$

is the Ramanujan's Q-function. The proof of (33) follows immediately from (32) and from the formula (see [13])

$$t_n(2) = n^n(Q(n)+1). \tag{35}$$

Now we get to the computational aspects of the tree polynomials. Although the coefficients of the tree polynomials can be obtained from (25), that process would not be efficient with respect to the regret computation. Fortunately, it turns out that there is no need to find the actual tree polynomials. To illustrate this, we will derive two recursion formulas for the regret terms using the tree polynomials. The first one is (7), which is originally from [11]. We start with a basic convolution property of the tree polynomials from [13],

$$t_n(y_1+y_2) = \sum_{k=0}^{n} \binom{n}{k} t_k(y_1)t_{n-k}(y_2). \tag{36}$$

By using (32), we can now write

$$n^n \cdot \mathcal{R}^n_{\mathcal{M}_{K_1+K_2}} = \sum_{k=0}^{n} \binom{n}{k} k^k (n-k)^{n-k} \mathcal{R}^k_{\mathcal{M}_{K_1}} \mathcal{R}^{n-k}_{\mathcal{M}_{K_2}}, \tag{37}$$

from which (7) easily follows.

Another tree polynomial formula from [13], derived below, leads to a linear time regret computation algorithm. As in [13], we start by considering

$$z \cdot \frac{d}{dz} \sum_{n \geq 0} t_n(y-1)\frac{z^n}{n!} = z \cdot \sum_{n \geq 1} n t_n(y-1)\frac{z^{n-1}}{n!} \tag{38}$$

$$= \sum_{n \geq 0} n t_n(y-1)\frac{z^n}{n!}. \tag{39}$$

On the other hand, by differentiating the generating function (24) and applying (21), we have

$$z \cdot \frac{d}{dz} \frac{1}{(1-T(z))^{y-1}} = \frac{z(y-1)}{(1-T(z))^{y}} \cdot T'(z) \tag{40}$$

$$= \frac{y-1}{(1-T(z))^{y}} \frac{T(z)}{(1-T(z))} \tag{41}$$

$$= (y-1)\left(\frac{1}{(1-T(z))^{y+1}} - \frac{1}{(1-T(z))^{y}}\right) \tag{42}$$

$$= (y-1)\left(\sum_{n\geq 0} t_n(y+1)\frac{z^n}{n!} - \sum_{n\geq 0} t_n(y)\frac{z^n}{n!}\right). \tag{43}$$

Comparing the coefficients of z^n in (39) and (43), we get

$$n t_n(y-1) = (y-1)(t_n(y+1) - t_n(y)), \tag{44}$$

$$t_n(y+1) = t_n(y) + \frac{n t_n(y-1)}{y-1}, \tag{45}$$

from which we get the final regret recursion

$$\mathcal{R}^n_{\mathcal{M}_K} = \mathcal{R}^n_{\mathcal{M}_{K-1}} + \frac{n}{K-2}\mathcal{R}^n_{\mathcal{M}_{K-2}}, \tag{46}$$

which holds for all $K > 2$. It is now straightforward to write a linear-time algorithm based on (46). The computation starts with the trivial case $\mathcal{R}^n_{\mathcal{M}_1} \equiv 1$. Next, the old recursion formula (37) is used to compute $\mathcal{R}^n_{\mathcal{M}_2}$ in time $\mathcal{O}(n)$. Finally, recursion (46) is applied $K-2$ times to end up with $\mathcal{R}^n_{\mathcal{M}_K}$. The time complexity of the whole computation is $\mathcal{O}(n+K)$, which is a major improvement over the previous methods.

Finally, it should be noted that since the time complexity of the above formula is dominated by the binary case, an additional speedup is possible by using an approximation for computing the $\mathcal{R}^n_{\mathcal{M}_2}$ term. In [10] we presented an extremely accurate, constant time regret approximation based on the generating function (23). The error of the approximation was shown to go down with the rate $\mathcal{O}\left(1/n^{3/2}\right)$, which means that the approximation is accurate enough for all the practical purposes. By using this approximation, the time complexity of computing $\mathcal{R}^n_{\mathcal{M}_K}$ drops to $\mathcal{O}(K)$.

5 Conclusion And Future Work

In this paper we have presented a novel framework for analyzing the properties of the multinomial NML distribution by connecting the regret and tree polynomials. Since polynomials are obviously much easier to understand and analyze than the complicated sum of combinatorial terms present in the definition of the NML, this connection will provide the researchers of the NML an alternative, more intuitive framework. As a practical demonstration of the usefulness of this framework, we derived the first linear time algorithm for computing the exponential regret sum.

In the future, our plan is to study theoretical properties of the regret more extensively via tree polynomials. Another natural topic is to extend the current work to more complex cases such as the clustering model class studied in our previous work. Even if it turns out that the regret generating function is not available in these cases, we believe that the current framework might still be useful in, e.g., deriving accurate approximations.

References

[1] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:445–471, 1978.

[2] J. Rissanen. Stochastic complexity. *Journal of the Royal Statistical Society*, 49(3):223–239 and 252–265, 1987.

[3] J. Rissanen. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42(1):40–47, January 1996.

[4] A. Barron, J. Rissanen, and B. Yu. The minimum description principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760, October 1998.

[5] P. Grünwald. *The Minimum Description Length Principle and Reasoning under Uncertainty*. PhD thesis, CWI, ILLC Dissertation Series 1998-03, 1998.

[6] J. Rissanen. Hypothesis selection and testing by the MDL principle. *Computer Journal*, 42(4):260–269, 1999.

[7] Q. Xie and A.R. Barron. Asymptotic minimax regret for data compression, gambling, and prediction. *IEEE Transactions on Information Theory*, 46(2):431–445, March 2000.

[8] J. Rissanen. Strong optimality of the normalized ML models as universal codes and information in data. *IEEE Transactions on Information Theory*, 47(5):1712–1717, July 2001.

[9] Yu M. Shtarkov. Universal sequential coding of single messages. *Problems of Information Transmission*, 23:3–17, 1987.

[10] P. Kontkanen, W. Buntine, P. Myllymäki, J. Rissanen, and H. Tirri. Efficient computation of stochastic complexity. In C. Bishop and B. Frey, editors, *Proceedings of the Ninth International Conference on Artificial Intelligence and Statistics*, pages 233–238. Society for Artificial Intelligence and Statistics, 2003.

[11] P. Kontkanen, P. Myllymäki, W. Buntine, J. Rissanen, and H. Tirri. An MDL framework for data clustering. In P. Grünwald, I.J. Myung, and M. Pitt, editors, *Advances in Minimum Description Length: Theory and Applications*. The MIT Press, 2005.

[12] P. Kontkanen and P. Myllymäki. A fast normalized maximum likelihood algorithm for multinomial data. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, 2005.

[13] D.E. Knuth and B. Pittel. A recurrence related to trees. *Proceedings of the American Mathematical Society*, 105(2):335–349, 1989.

[14] M. Koivisto. *Sum-Product Algorithms for the Analysis of Genetic Risks*. PhD thesis, Report A-2004-1, Department of Computer Science, University of Helsinki, 2004.

[15] P. Kontkanen and P. Myllymäki. Computing the regret table for multinomial data. Technical Report 2005–1, Helsinki Institute for Information Technology (HIIT), 2005. http://cosco.hiit.fi/Articles/hiit-2005-1.pdf.

[16] W. Szpankowski. *Average case analysis of algorithms on sequences*. John Wiley & Sons, 2001.

[17] H.S. Wilf. *generatingfunctionology (second edition)*. Academic Press, 1994.

[18] V.K. Balakrishnan. *Schaum's Outline of Theory and Problems of Combinatorics*. McGraw-Hill, 1995.

[19] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics (second edition)*. Addison-Wesley, 1994.

[20] D.E. Knuth. *The Art of Computer Programming, vols. 1–3 (third edition)*. Addison-Wesley, 1997–1998.

[21] R.M. Corless, G.H. Gonnet, D.E.G. Hare, D.J. Jeffrey, and D.E. Knuth. On the Lambert W function. *Advances in Computational Mathematics*, 5:329–359, 1996.

[22] C. Chauve, S. Dulucq, and O. Guibert. Enumeration of some labelled trees. In D. Krob, A.A. Mikhalev, and A.V. Mikhalev, editors, *Formal Power Series and Algebraic Combinatorics, FPSAC'00*, pages 146–157. Springer-Verlag, 2000.

[23] P. Flajolet, P.J. Grabner, P. Kirschenhofer, and H. Prodinger. On Ramanujan's Q-function. *Journal of Computational and Applied Mathematics*, 58:103–116, 1995.