Applications of generating functions to stochastic processes and to the complexity of the knapsack problem

Jorma Jormakka
Department of Communications and Networking, Aalto University, Espoo, Finland
Email Id: jorma.jormakka@ mil.fi

Sourangshu Ghosh
Department of Civil Engineering, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal, India
Email Id: sourangshu@iitkgp.ac.in

Abstract

The paper describes a method of solving some stochastic processes using generating functions. A general theorem of generating functions of a particular type is derived. A generating function of this type is applied to a stochastic process yielding polynomial time algorithms for certain partitions. The method is generalized to a stochastic process describing a rather general linear transform. Finally, the main idea of the method is used in deriving a theoretical polynomial time algorithm to the knapsack problem.

Keywords: stochastic processes, generating functions, polynomial time algorithms, partitions, knapsacks.

1. Introduction

Generating functions are one of the main ways of solving recursion equations. In this article we start by looking at two simple examples of solving stochastic processes with the probability generating function. Then we derive a general theorem of the generating function method for solving stochastic processes of a certain type. Next, we introduce a generating function of an unfamiliar type and show that it has relevance in calculation of partitions in a polynomial time. This result is of interest in computational complexity of some problems e.g. in cryptography. As an example, we give a polynomial time algorithm to the knapsack problem. While there are several books on generating functions (see e.g. [1] and [2]), usefulness of generating functions in deriving polynomial time algorithms seems not to have been noticed before. Probability generating functions are also used to model Queuing System [3], redundant renewable system [4], waiting Time Problems [5]. A good review of application of Probability generating functions to Stochastic Polling systems is done by Vishnevsky [6].

1.2. Simple examples using the probability generating function

Let us apply the probability generating function *(pgf)* to a birth-death process of a M/M/1 queue and of a loss system in Examples 1 and 2. It will be seen that these simple examples can be solved with the generating function method where *pgf* is the generating function.

Example 1: *A process with nonempty boundary.*

Let us consider the following simple birth and death process of an *M/M/1* queue:

$$s_{n,q} = s_{n-1,q} - \lambda s_{n-1,q} - \mu s_{n-1,q} + \lambda s_{n-1,q-1} + \mu s_{n-1,q+1} = a_0 s_{n-1,q} + a_1 s_{n-1,q-1} + a_2 s_{n-1,q+1} \quad (1)$$

In (1) λ is the birth rate (calls/second), μ is the death rate (calls/second), and $s_{n,q}$ is the probability, that the system is at the state q (has q calls in the system) at the discrete time instance n. Let

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 April 2021 doi:10.20944/preprints202104.0706.v1

$G_n(u) = \sum_{q=0}^{\infty} s_{n,q} u^q$ be the probability generating function. Multiplying by u^q and summing in (1) yields (guessing that $s_{n-1,-1} = \mu/\lambda$ and $s_{n-1,0} = 1$)

$$G_n(u) = (a_0 + a_1 u + a_2 u^{-1}) G_{n-1}(u) + \frac{\mu}{\lambda} a_1 - \frac{1}{u} a_2 \ , \tag{2}$$

$$G_n(u) = (a_0 + a_1 u + a_2 u^{-1})^n G_0(u) + \left(\frac{\mu}{\lambda} a_1 - \frac{1}{u} a_2 \right) \frac{((a_0 + a_1 u + a_2 u^{-1})^n - 1)}{((a_0 + a_1 u + a_2 u^{-1}) - 1)} \ .$$

Inserting a_i the familiar steady state solution for state probabilities is thus obtained

$$G(u) = \frac{\frac{\mu}{\lambda} a_1 - \frac{1}{u} a_2}{1 - (a_0 + a_1 u + a_2 u^{-1})} = \frac{1}{1 - \frac{\lambda}{\mu} u} = \sum_{q=0}^{\infty} \left(\frac{\lambda}{\mu} \right)^q u^q \ . \tag{3}$$

The negative side was the need to guess $s_{n-1.-1}$ but we did not need to use balance equations requiring a steady state. In addition to the state probabilities, we may be interested in a situation where each state implies a cost. Let the cost for the state $s_{n,q}$ be $r_{n,q}$. The equation for calculating the costs gives another problem that can be solved by a generating function in the same way:

$$r_{n,q} s_{n,q} = r_{n-1,q} a_0 s_{n-1,q} + r_{n-1,q-1} a_1 s_{n-1,q-1} + r_{n-1,q+1} a_2 s_{n-1,q+1} = b_0 m_{n-1,q} + b_1 m_{n-1,q-1} + b_2 m_{n-1,q+1} \ . \tag{4}$$

In this case it is useful that the generating function method does not assume a steady state since costs are often growing.

Example 2. *A teletraffic loss model.*

The difference between Example 1 and the basic loss model for telegraphic (see e.g. [7][8]) is that the death rate depends on the state:

$$s_{n,q} = s_{n-1,q} - \lambda s_{n-1,q} - q\mu s_{n-1,q} + \lambda s_{n-1,q-1} + (q+1)\mu s_{n-1,q+1} \ , \tag{5}$$

and let as again take the probability generating function $G_n(u) = \sum_{q=0}^{\infty} s_{n,q} u^q$. We obtain a differential equation for the generating function by inserting the equation

$$\sum q s_{n,q} u^q = u \frac{d}{du} \sum s_{n,q} u^q \ . \tag{6}$$

This gives

$$G_{n+1}(u) = G_n(u)(1 - \lambda + \lambda u) + \mu \frac{d}{du} G_n(u) - \mu u \frac{d}{du} G_n(u) \ . \tag{7}$$

Let us look for a solution $G_{n+1}(u) = \alpha(q) G_n(u)$. We obtain a differential equation:

$$\frac{d}{du} G_n(u) = \frac{\lambda(1-u) + \alpha(q) - 1}{\mu(1-u)} G_n(u) \ . \tag{8}$$

The left side in (8) is not dependent on q, thus necessarily $\alpha = 1$ and the familiar steady state solution $G_n(u) = e^{\rho u} = \sum_{q=0}^{\infty} \frac{\rho^q}{q!}$, where $\rho = \lambda / \mu$, is obtained.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 April 2021 doi:10.20944/preprints202104.0706.v1

After these simple examples, let us derive a general result on generating functions.

2. A general result of the generating function method

Let us consider the recursion equation

$$s_{n,q_1,\ldots,q_K,r} = s_{n-1,q_1,\ldots,q_K,r} + \sum_j \frac{T}{N} p_{n-1,q_1,\ldots,q_j+c_j,\ldots q_K,f_j(r,\bar{q},n)} s_{n-1,q_1,\ldots,q_j+c_j,\ldots,q_K,f_j(r,\bar{q},n)} . \quad (9)$$

Here c_i are integers and $f_j(r,\bar{q},n) = f_j(r,q_1,\ldots,q_K,n)$ are integer valued functions. In this section we keep in the equations the length of the time step T/N where T is the total time. Usually in stochastic processes T and N are omitted as unnecessary, but they are useful here in taking the limit $N \to \infty$. The goal of the generating function method is to find functions $G_n(u,v)$ and $B_n(u,v)$ such that

$$G_n(u,v) = G_n(u_1,\ldots,u_K,v) = \sum_{q_1}\cdots\sum_{q_K}\sum_r s_{n,q_1,\ldots,q_K,r} \prod_{j=1}^{K} u_j^{y_j(k,n,q_1,\ldots,q_K,r)} v^{x(k,n,q_1,\ldots,q_K,r)} , \quad (10)$$

and the recursion equation is presented in the form

$$G_n(u,v) = \left(1 + \sum_j \frac{T}{N} g_j(n-1,u,v)\right) G_{n-1}(u,v) + B_{n-1}(u,v) \quad (11)$$

The function $B_{n-1}(u,v)$ describes the contribution of the boundaries. $g_j(n-1,u,v)$ is not always a function, but can be a linear operator. For instance, in the telegraphic loss model in Example 2, the death rate $q_j\mu_j$ is state-dependent in such a way that it causes $g_j(n-1,u,v)$ to be a differential operator when the generating function is the probability generating function.

Depending on how complicated are $p_{n-1,q_1,\ldots,q_K,r}$ and $f_j(r,\bar{q},n)$, it is not always possible, or at least easy, to find a generating function as in (10). However, once there is this kind of form, we can try various analytic methods. For instance, let us assume that $g_j(n-1,u,v) = g_j(u,v)$ is a function (i.e., not an operator). Let N approach to infinity. By Weierstrass inequalities

$$\lim_{N\to\infty}\left(1 + \sum_j \frac{T}{N} g_j(u,v)\right) = \lim_{N\to\infty} \prod_j \left(1 + \frac{T}{N} g_j(u,v)\right) \quad (12)$$

holds. Let us prove a result for certain time-independent Markov processes.

Theorem 1. *Let (9) be a time-independent Markov process and $g_j(n-1,u,v) = g_j(u,v)$ a function, then the steady state probabilities can be obtained from the generating function:*

$$G(u,v) = \frac{B(u,v)}{1-h(u,v)} \quad (13)$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 April 2021 doi:10.20944/preprints202104.0706.v1

if $B_{n-1}(u,v) = B(u,v)$ *is nonzero. There is no steady state if* $B(u,v) = 0$. *The generating function satisfies the equation*

$$G(u,v) = e^{\sum_j Tg_j(u,v)} G_0(u,v) - \frac{B(u,v)}{1-h(u,v)}\left(e^{\sum_j Tg_j(u,v)} - 1\right). \tag{14}$$

Proof: Let us write (11) as

$$G_n(u,v) = h_{n-1}(u,v)G_{n-1}(u,v) + B_{n-1}(u,v) \tag{15}$$

where we may simplify $h_{n-1}(u,v) = h(u,v)$ and $B_{n-1}(u,v) = B(u,v)$ since the process is a time-independent Markov process. Thus

$$G_0(u,v) = \frac{B(u,v)}{1-h(u,v)} + \frac{1}{(h(u,v))^n}\left(G_n(u,v) - \frac{B(u,v)}{1-h(u,v)}\right). \tag{16}$$

Let $N \to \infty$. Then the generating function approaches a steady state function $G_N(u,v) \to G(u,v)$. (14) is obtained by the limit procedure

$$\lim_{N\to\infty}(h(u,v))^N = \lim_{N\to\infty}\prod_j\left(1 + \frac{T}{N}g_j(u,v)\right)^N = e^{\sum_j Tg_j(u,v)},$$

$$G_0(u,v) = \frac{B(u,v)}{1-h(u,v)}\left(1 - e^{-\sum_j Tg_j(u,v)}\right) + e^{-\sum_j Tg_j(u,v)} G(u,v). \tag{17}$$

Equation (17) holds also if we start from $G_N(u,v)$ instead of $G_0(u,v)$

$$G(u,v) = G_N(u,v) = \frac{B(u,v)}{1-h(u,v)} + e^{-\sum_j Tg_j(u,v)}\left(G(u,v) - \frac{B(u,v)}{1-h(u,v)}\right). \tag{18}$$

Solving (18) yields

$$G(u,v)\left(1 - e^{-\sum_j Tg_j(u,v)}\right) = \frac{B(u,v)}{1-h(u,v)}\left(1 - e^{-\sum_j Tg_j(u,v)}\right), \tag{19}$$

$$G(u,v) = \frac{B(u,v)}{1-h(u,v)} \quad .$$

If $B(u,v)$ is zero in (13), there is no stationary state because in this case

$$G(u,v) = h(u,v)G(u,v). \tag{20}$$

and the process must be constant. □

If $g_j(u,v)$ is derived from equations for $rs_{n,q_1,\ldots,q_K,r}$ instead for $s_{n,q_1,\ldots,q_K,r}$ we get similar formulas for the cost distribution. Calculating the stationary state solution, or the cost values, usage of the generating function method involves expanding a polynomial into series. If there is a steady

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 April 2021 doi:10.20944/preprints202104.0706.v1

state, the generating function method can be unnecessarily heavy. If $g_j(n-1,u,v)$ is an operator, the method is often difficult. In the situations described by Remark 1, the method is both relatively easy to apply and one of the few solution methods that work.

Remark 1. The generating function method is most suitable if:

1) $B_{n-1}(u,v)$ vanishes e.g. because the state space is infinite or boundaries are nice,
2) $g_j(n-1,u,v)$ is a function,
3) $h_n(u,v)$ or $B_{n-1}(u,v)$ are time-dependent, also in a non-Markov way.

Remark 2. In Example 2 $B_{n-1}(u,v)=0$ in and $g_j(n-1,u,v)=-\lambda+\lambda u+\mu\frac{d}{du}-\mu u\frac{d}{du}$ is a linear operator.

The generating function method is of course well known: find a generating function, solve the equation with it, expand the solution into a series and pick up the terms whose powers correspond to those of $s_{n,q_1,\ldots,q_K,r}$. The nice thing with the method is that it works with non-stationary processes which are not even Markovian. Cost distributions are one example we already mentioned. The interesting thing with generating functions is that there probably are undiscovered types of useful generating functions for many problems. Let us look at some non-stationary processes for calculating partitions. They lead to unfamiliar looking generating functions.

3. *A generating function for calculating of partitions in polynomial time*

Let us consider the problem of calculating partitions. Algorithmic solutions are always possible (see e.g. [8][9][10][11] but the computational complexity becomes a limiting factor very fast. Limit theorems are one possibility. Limit theorems can only be found for very common partitions in the literature. Some general methods to derive new limit theorems for partitions are known. One is summing over some variable(s) in a partition or relaxing a restriction. If this converts the partition to a known partition, lower or upper bounds may be obtained to the original partition. The other method is exclusion and inclusion[12][13]. It is possible to divide the set of partitions into two subsets: the one where each element contains a unit k and the set where no element contains this unit. The first set has a one-to-one and onto mapping into partitions of $m-k$ while in the second set all units larger than k can be decreased by one. The latter set can be mapped one-to-one and onto to a number of lower dimensional partitions. Estimating the corresponding partitions from above and below may give a recursion equation to the original partition, and it may result to a limit theorem. There is also an interesting connection between partitions and theta functions (see [14][15][16]), and it also may lead to limit theorems. However, if the goal is not a limit theorem but the exact result, then often there is only an exponential time algorithm that counts the partitions. In this article we show that there can be a better way: there are polynomial time algorithms for calculating the exact value of some partitions, even though the partitions grow exponentially.

Let us consider the recursion equations for the 3-dimentional Heisenberg group in [21] p.26:

$$s_{n,q,r}=s_{n-1,q,r}+s_{n-1,q-1,r+q-n}\quad 0\le q\le n,\ 0\le r\le q(n-q),\ s_{0,0,0}=1 \tag{21}$$

The recursion equation is solved with the generating function

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 April 2021 doi:10.20944/preprints202104.0706.v1

$$G_n(u,v)=\sum_{q=0}^{n}\sum_{r=0}^{q(n-q)} s_{n,q,r}u^q v^{r+q^2/2}=\prod_{k=1}^{n}(1+uv^{k-1/2}). \tag{22}$$

We can expand the product and get the equations where $\delta_k \in \{0,1\}$

$$\sum_{k=1}^{n}\delta_k = q\,,\ \sum_{k=1}^{n}\delta_k(k-\frac{1}{2})=r+\frac{1}{2}q^2,\ \text{ thus} \tag{23}$$

$$s_{n,q,r}=P'_{q,n}(r+q/2+q^2/2)=P_{q,n-q}(r).$$

Here $P'_{q,j}(m)$ is the number of ways to represent m as a sum of q nonempty numbers, which are in strictly increasing order and the largest is at most j. $P_{q,j}(m)$ is the number of ways to represent m as a sum of q numbers, which are in increasing order and the largest is at most j.

The solution is thus the partition $s_{n,q,r}=P_{q,n-q}(r)$. The recursion equation (21) branches into two paths on each step, thus $\sum_{q,r} s_{n,q,r}=2^n$, so it is a graph of a free group on two generators and corresponds to the hyperbolic covering space. But there are relations and in the quotient space, the covering space of the Heisenberg manifold of dimension 3, the graph of (21) grows only as a third degree polynomial (see [21] for a detailed discussion). Thus, some of the $s_{n,q,r}$ must grow exponentially. Quite interestingly, the $s_{n,q,r}$ turn out to be partitions that have a simple description.

In the paper we try to see to what extent the method can be generalized. If we can find a more general recursion equation which can be solved with a generating function that generalizes (22), then we have a way to calculate directly some other partitions. As earlier work on this issue we can mention the Jacobi's identities [15][16]. They are sum=product equations connecting generating functions and partitions.

Let us notice that in the 3-dimensional Heisenberg group there are two generators $g_1=\begin{bmatrix}1&1&0\\0&1&0\\0&0&1\end{bmatrix}$ and $g_2=\begin{bmatrix}1&0&0\\0&1&1\\0&0&1\end{bmatrix}$. A word of length n letters in positive generators has the form $X=\begin{bmatrix}1&n-q&r\\0&1&q\\0&0&0\end{bmatrix}$ where q is the number of letters g_2. The generators operate by multiplying from the left and e.g. $g_1: X\to\begin{bmatrix}1&n+1-q&r+q\\0&1&q\\0&0&1\end{bmatrix}$. We can write this as $g_1:\begin{bmatrix}n\\q\\r\end{bmatrix}\to\begin{bmatrix}n+1\\q\\r+q\end{bmatrix}$. This gives (21), see p.26 in [7]. Let us now consider a space with *i* generators which operate by a linear transform, not by simple multiplication from left, and we try to put many parameters that can be chosen.

Theorem 2. *Let the pair* M_i, K_i *operate by a linear transform* $y=M_i x+K_i$, $x,y\in X\subset \mathbf{N}^3$, *where*

$$M_i=\begin{bmatrix}1&0&0\\0&1&0\\\sigma^{-1}\beta_i&-ac_i\sigma^{-1-n}&\sigma^{-1}\end{bmatrix},\ K_i=\begin{bmatrix}1\\c_i\\\sigma^{-1}(\beta_i-\gamma_i)\end{bmatrix},\ x=\begin{bmatrix}n\\q\\r\end{bmatrix},\ \sigma,\ a,\ c_i,\beta_i \text{ and } \gamma_i \text{ are integers} \tag{24}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 April 2021 doi:10.20944/preprints202104.0706.v1

($n \ge 0, q \ge 0,\ r \ge 0$ are also integers). This linear transform gives a recursion equation

$$s_{n,q,r} = \sum_i s_{n-1,q-c_i,\sigma r-\beta_i n+qac_i\sigma^{1-n}+\gamma_i},\ s_{0,q,r} = \delta_{q=0\wedge r=0},\ s_{n,q,r} = 0 \text{ if } q<0 \text{ or } r<0. \tag{25}$$

The numbers $s_{n,q,r}$ satisfy the sum=product equation

$$\sum_{q=0}^{\infty}\sum_{r=0}^{\infty} s_{n,q,r} u^q v^{\sigma^n r+\frac{1}{2}aq^2+bq} = \prod_{k=1}^{n}\left(\sum_i u^{c_i} v^{\sigma^{k-1}\beta_i k-\sigma^{k-1}\gamma_i-\frac{1}{2}ac_i^2+bc_i}\right) \tag{26}$$

where b can be freely selected to get a more convenient result.

Proof: The linear transform is

$$y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \sigma^{-1}\beta_i & -ac_i\sigma^{-1-n} & \sigma^{-1} \end{bmatrix}\begin{bmatrix} n \\ q \\ r \end{bmatrix} + \begin{bmatrix} 1 \\ c_i \\ \sigma^{-1}(\beta_i-\gamma_i) \end{bmatrix} = \begin{bmatrix} n+1 \\ q+c_i \\ \sigma^{-1}(r+\beta_i n-ac_i\sigma^{-n}q-\gamma_i) \end{bmatrix}, \tag{27}$$

i.e., the ith generator maps $\begin{bmatrix} n-1 \\ q-c_i \\ \sigma r-\beta_i n+ac_i\sigma^{1-n}q+\gamma_i \end{bmatrix} \to \begin{bmatrix} n \\ q \\ r \end{bmatrix}$.

Let us write the transformation (24) as a recursion equation:

$$s_{n,q,r} = \sum_i s_{n-1,q_i,r_i} = \sum_i s_{n-1,q-c_i,\sigma r-\beta_i n+ac_i\sigma^{1-n}q+\gamma_i}. \tag{28}$$

Let

$$q' = q-c_i,\ r' = \sigma r-\beta_i n+ac_i\sigma^{1-n}q+\gamma_i. \tag{29}$$

Then

$$\begin{aligned} \sigma^n r+\frac{1}{2}aq^2+bq &= \sigma^n(\sigma^{-1}r'+\sigma^{-1}\beta_i n-\sigma^{-n}ac_i q'-\sigma^{-1}ac_i{}^2-\sigma^{-1}\gamma_i) \\ &\quad +\frac{1}{2}aq'^2+ac_i q'+\frac{1}{2}ac_i^2+bq'+bc_i \\ &= \sigma^{n-1}r'+\frac{1}{2}aq'^2+bq'+\sigma^{n-1}\beta_i n+\frac{1}{2}ac_i^2+bc_i-ac_i^2-\sigma^{n-1}\gamma_i \\ &= \sigma^{n-1}r'+\frac{1}{2}aq'^2+bq'+\sigma^{n-1}\beta_i n+h_i+\sigma^{n-1}\gamma_i, \end{aligned} \tag{30}$$

where $h_i = -\frac{1}{2}ac_i^2+bc_i$. Thus

$$s_{n-1,q-c_i,\sigma r-\beta_i n+qac_i\sigma^{1-n}+\gamma_i} u^q v^{\sigma^n r+\frac{1}{2}aq^2+bq} = s_{n-1,q_i',r'} u^{q'} v^{\sigma^{n-1}r'+\frac{1}{2}aq'^2+bq'} u^{c_i} v^{\sigma^{n-1}(n\beta_i-\gamma_i)+h_i}. \tag{31}$$

Let the generating function be

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 April 2021 doi:10.20944/preprints202104.0706.v1

$$G_n(u,v) = \sum_{q=-\infty}^{\infty} \sum_{r=-\infty}^{\infty} s_{n,q,r} u^q v^{\sigma^n r + \frac{1}{2}aq^2 + bq} . \quad (32)$$

Multiplying both sides of (28) by $u^q v^{\sigma^n r + \frac{1}{2}aq^2 + bq}$, summing over q and r, and using the initial values (let us assume the initial values make the boundary disappear) gives

$$G_n(u,v) = G_{n-1}(u,v) \sum_i u^{c_i} v^{\sigma^{n-1}\beta_i n + h_i - \sigma^{n-1}\gamma_i} . \quad (33)$$

The claim of the theorem follows. □

Let us look at the special case, where $\sigma = 1$ and there are only two alternative outcomes for an event: q can either increase by c_1 or decrease by $-c_2$. Then the generating function can be expressed as (here $h_i' = -\frac{1}{2}ac_i^2 + bc_i - \gamma_i$)

$$G_n(u,v) = \prod_{k=1}^{n} (u^{c_1} v^{\beta_1 k + h_1'} + u^{c_2} v^{\beta_2 k + h_2'}) \quad (34)$$
$$= u^{c_1} v^{\frac{1}{2}n(n+1)\beta_1 + h_1'} \prod_{k=1}^{n} \left(1 + u^{c_2 - c_i} v^{(\beta_2 - \beta_1)k + h_2' - h_1'}\right) .$$

This function can be easily expanded into a power series and the result gives partitions satisfying

$$\sum_{k=1}^{n} \delta_k = \frac{q}{c_2 - c_1} , \; j = \sum_{k=1}^{n} \delta_k k = \frac{1}{\beta_2 - \beta_1}\left(\frac{h_1' - h_2'}{c_2 - c_1} q + bq + \frac{1}{2}aq^2\right), \; \delta_k \in \{0,1\}.$$

That is, $s_{n,q,r}$ gives the number of $q/(c_2 - c_1)$ nonzero numbers in strictly increasing order, not larger than n and giving the sum j. The recursion equation (28) grows with a polynomial speed as the range of the indices is bounded by a polynomial. While we sum to infinity in (28), the indices q and r grow as polynomials of n. The case (23) is obtained by selecting $c_1 = \gamma_i = b = \beta_1 = 0$ and $\sigma = a = \beta_2 = c_2 = 1$.

Let us note that in a general linear transform we should have $r_i = \sigma_i r - \beta_i n + c_i q + \gamma_i$ where all parameters are real numbers in (28), but instead we have $r_i = \sigma r - \beta_i n + a\sigma^{n-1} c_i q + \gamma_i$ and all parameters are integers. These limitations were necessary in order to generalize the method (21)-(22).

As an example of the ideas in Theorem 2, we will derive a polynomial time algorithm for the knapsack problem. It has a very simple generating function. Let us notice that the recursion equation for $n > 0$

$$s_{n,j} = s_{n-1,j} + s_{n-1,j-d_n} , \qquad s_{n,j} = 0 \text{ if } j < 0 . \quad (35)$$

Gives the equation

$$G_n(x) = G_{n-1}(x)(1 + x^{d_n}) \quad (36)$$

for the generating function $G_n(x) = \sum_{j=0}^{\infty} s_{n,j} x^j$. That is

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 April 2021 doi:10.20944/preprints202104.0706.v1

$$G_n(x) = \prod_{k=1}^{n} (1 + x^{d_k}) . \tag{37}$$

Thus, $s_{n,j}$ is the number of ways to express j as a sum

$$j = \sum_{k=1}^{n} \delta_k d_k \tag{38}$$

where $\delta_k \in \{0,1\}$. The knapsack problem is, given j and a knapsack $\{d_k\}_{k=1}^{n}$, is there a way to express j as $j = \sum_{k=1}^{n} \delta_k d_k$? If we generate all numbers $s_{n,j}$ from the recursion (35), then we can answer the knapsack problem by yes/no. Let us find a bound to the number of numbers $s_{n,j}$. Clearly $j \le \sum_{k=1}^{n} d_k$ on each step n. Let us assume that $\exists \alpha > 0$, $C > 0$ constants, such that for every $n > 0$ holds $d_n < Cn^{\alpha}$. The number of numbers $s_{k,j}$, $k \le n$, where $j \le \sum_{i=1}^{k} d_i$ in $s_{k,j}$ has a (huge) polynomial upper bound

$$\#\{s_{k,j}\} \le \sum_{k=1}^{n} \sum_{i=1}^{k} d_i < C \sum_{k=1}^{n} \sum_{i=1}^{k} n^{\alpha} \approx n^{2+\alpha} . \tag{39}$$

Consequently, we have a polynomial time algorithm for this knapsack problem with a polynomial limit on the knapsack. Clearly, the algorithm is not very useful as it is usually much worse than going through all 2^n combinations of the knapsack's numbers except for very huge numbers n. We are only interested in the existence of the algorithm.

Let us consider, what should be acceptable sequences of knapsacks for evaluating the running time of an algorithm to solve them. The argument which is used as the measure of complexity should grow in a linear manner and the running time of the algorithm is a function of this argument. The number n of elements in the knapsack is one measure of complexity. It is not a sufficient measure alone: if we can select knapsacks freely, then we can easily find a set of knapsacks growing so fast that there cannot be a polynomial time algorithm solving the knapsacks. We must include some measure of the size of the knapsack to the argument in addition to n. It is natural to use n and one of the following:

$$j, \ \sum_{k=1}^{n} d_k \quad \text{or the set of } d_k . \tag{40}$$

We should require that the argument grows linearly. One natural interpretation of linearity is that n grows linearly and there is a polynomial bound on the other complexity measure. A polynomial bound on any of the mentioned measures gives a polynomial upper bound to the computation of the numbers $s_{k,j}$ as in (39).

We can also obtain a polynomial algorithm for knapsacks where the numbers grow exponentially. This corresponds to taking a binary representation of one of the entities (40) and using the length in bits as a measure of complexity. This kind of measure gives another natural interpretation of linearity of the argument and it is similar to the one used in evaluating the speed of an algorithm for basic arithmetic operations, like multiplication.

Let us consider knapsacks where the numbers in the knapsacks grow faster than any polynomial, but their lengths in bits grow at most with a polynomial speed. We derive a polynomial

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 April 2021 doi:10.20944/preprints202104.0706.v1

time algorithm. Let $\alpha > \beta \geq 1$, C and M be selected. Let $A_n = \{d_{k,n}\}_{k=1}^{n}$ be a sequence of knapsacks. If at any index n there is $d_{k,n}$ such that $d_{k,n} > Cn^{\alpha}$, let us select $\lfloor Mn^{\beta} \rfloor$ numbers $r_{n,i} \leq Cn^{\alpha}$ but growing with the same speed ($r_{n,i} \approx Cn^{\alpha}$) and let us start a branch of knapsacks $\{d_{k,n} \bmod r_{n,i}\}_{k=1}^{n}$ and $j_{n,i} = j \bmod r_{n,i}$. As the length of the numbers in bits grows at most as a polynomial, we can take modulo $r_{n,i}$ in polynomial time. Each of these modulo knapsacks has a polynomial time algorithm since the numbers in the knapsacks have a polynomial bound. At most we are starting $n^{1+\beta}$ branches. Thus, we have a polynomial bound $n^{2+\beta+\alpha}$ to the number of numbers $s_{n,j}$. For each knapsack we get $\lfloor Mn^{\beta} \rfloor$ answers to the question: is there a solution modulo $r_{n,i}$. If there is an answer to the original knapsack, there must be a solution to each knapsack modulo any number. This yields a good probability that we can say if there is no answer to the knapsack but in this way we cannot get a full guarantee that there is an answer.

We can decide deterministically if the knapsack has a solution by looking at the solutions of the modular knapsacks in the following way. First, we select the numbers $r_{n,i}$ co-prime in the algorithm above. We assume that the number j is bounded by $\exp(n^{\gamma})$ for some fixed $\gamma > 0$. The product h of the numbers $r_{n,i}$ grows as $n^{\alpha n^{\beta}}$. We can select α and β such that $j < h$ holds always. Then, if $x = \sum_{k=1}^{n} \delta_k d_k$ is a solution to all modular knapsacks, $x = j$ by the Chinese reminder theorem. Let us assume that all modular knapsacks have a solution. We must check if they have a common solution in order to use the Chinese reminder theorem.

The recursion corresponding to (35) for each modular knapsack can be calculated backwards from $s_{n,j_{n,i},i}$ to values $s_{k,m,i}$, $k < n$, up to $s_{0,0,i}$. Considered as a graph with nodes and links, the number of nodes $s_{k,m,i}$ depends on n in a polynomial manner as the modular knapsack has a polynomial bound. There are two reasons why the number of nodes $s_{k,m,i}$ does not grow exponentially: there are missing links because a path $s_{n,j_{n,i},i} \to s_{k,m,i} \to s_{0,0,i}$ through some $s_{k,m,i}$ does not exist, and there are loops, meaning that several paths lead to the same node. A missing link and a loop can be associated with nodes. Thus, the numbers of loops and missing links have polynomial bounds. With loops there are the starting node which starts two paths and the ending node which creates a loop when following paths $s_{n,j_{n,i},i} \to s_{k,m,i} \to s_{0,0,i}$. With missing links there is a node $s_{k,m,i}$ from which there is missing the link $s_{k,m,i} \to s_{k-1,m,i}$ or the link $s_{k,m,i} \to s_{k-1,m-d_k \bmod r_{n,i},i}$. Both links cannot be missing as the graph is connected. Missing links in the graph of a modular knapsacks can only be caused by the situation that the path $s_{n,j_{n,i},i} \to s_{k,m,i} \to s_{0,l,i}$ would lead to $l > 0$. (If the knapsack is not modular, we also get a missing link if $s_{k,m} \to s_{k-1,m-d_k}$ and $m < d_k$ at some level k.) If there are no missing links on a node, it starts the two paths of a new loop.

Let us start at the graph $i = 1$ and loop over all graphs. In the step of this loop we first look at the graph i and follow the polynomial number of paths from $s_{n,j_{n,i},i}$ to $s_{0,0,i}$. There are only a polynomial number of paths to follow because on each level k there are only a polynomial number of nodes. Following a path means continuing to maximum two links from a node, but the number of nodes is not growing faster than a polynomial. We also follow a path in all other graphs while

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 April 2021 doi:10.20944/preprints202104.0706.v1

following a path in the graph i. At each graph i' we start at $s_{n,j_{n,i'},i'}$ and follow the same link as in the graph i. The links are easily identified: $s_{k,m,i'} \to s_{k-1,m,i'}$ and $s_{k,m,i'} \to s_{k-1,m-d_k \bmod r_{n,i'},i'}$, thus we can follow the same link in every graph. There are still only a polynomial number of paths to follow even as we follow all graphs. In case none of the graphs have a missing link at the level k we follow both links to the level $k-1$. If on some path at a node $s_{k,m,i}$ any graph i has a missing link, we follow only the other (existing) link in every graph. If at a level k on some path there are graphs i and i' where $s_{k,m,i}$ and $s_{k,m',i'}$ are missing different links, we stop the path. If we have found a path that can be followed in all graphs i' to $s_{0,0,i'}$ according to these rules when this algorithm has been completed, there is a common solution to all modular knapsacks. Then the original knapsack has a solution. If all paths stop, there is no solution. This algorithm requires polynomial time and we obtained a deterministic polynomial time algorithm to the knapsack problem.

The knapsack problem is NP complete but it is usually not clearly stated how knapsacks can be selected, one usually only talks of n as the measure of complexity [17][18][19][20]. If knapsacks can be completely freely selected, then taking a sequence of knapsacks where the length of the numbers grows faster than a polynomial, gives a sequence that cannot be calculated in polynomial time. Indeed, all such computations on the numbers which require all bits of the number will take a time that grows faster than any polynomial. This includes basic arithmetic operations like addition, subtraction, multiplication, division and taking modulus. In the other cases we find a polynomial time algorithm. Whether this polynomial time algorithm to the knapsack problem should be considered as a solution to the problem if P equals NP, we will leave to the reader to judge. In this paper we will not verify what kind of knapsack sequences are needed to NP completeness, nor if an acceptable algorithm should be some kind of an abstract algorithm that does not mention the sizes of the numbers in knapsacks at all. Most probably, for NP completeness we need the knapsacks that can grow exponentially but whose logarithms grow as polynomials. If this is the correct interpretation of the complexity of the knapsack then the answer probably is: P=NP, but the known polynomial time algorithm is totally inefficient.

4. Conclusions

The article derives a general theorem for the generating function method of solving stochastic processes. As an application of the method, a new type of a generating function for calculating a class of stochastic processes was presented. The generating function is interesting because it gives a link by which some exponentially growing partitions can be calculated exactly in polynomial time. The last example of a polynomial time algorithm to the knapsack problem may also be of some interest in the P=NP problem context. It is a purely theoretical algorithm without practical applications, but shows the idea in Theorem 2 and motivates why developing a method to derive polynomial time algorithms for partitions can have some relevance.

Acknowledgements:

The authors thanks Mikko Kiviharju, Matti Lehtinen, Edvard Fagerholm and Kaisa Nyberg for helpful comments.

5. References

[1] H. S. Wilf, *generatingfunctionology,* Academic Press, 1994.

[2] P. Flajolet, R. Sedgewick, *Analytic combinatorics,* online, 2006.

[3] Ayyappan G, Karpagam S. An $M^{[X]}/G(a,b)/1$ Queueing System with Breakdown and Repair, Stand-By Server, Multiple Vacation and Control Policy on Request for Re-Service. Mathematics. 2018; 6(6):101. https://doi.org/10.3390/math6060101

[4] Dimitrov B, Rykov V, Milovanova T. Renewal Redundant Systems Under the Marshall–Olkin Failure Model. A Probability Analysis. Mathematics. 2020; 8(3):459. https://doi.org/10.3390/math8030459

[5] Kim B, Kim J, Kim J. Waiting Time Problems for Patterns in a Sequence of Multi-State Trials. Mathematics. 2020; 8(11):1893. https://doi.org/10.3390/math8111893

[6] Vishnevsky V, Semenova O. Polling Systems and Their Application to Telecommunication Networks. Mathematics. 2021; 9(2):117. https://doi.org/10.3390/math9020117

[7] Basharin, G.P., Gaidamaka, Y.V. & Samouylov, K.E. Mathematical theory of teletraffic and its application to the analysis of multiservice communication of next generation networks. Aut. Control Comp. Sci. 47, 62–69 (2013). https://doi.org/10.3103/S0146411613020028

[8] Kiseleva, E.M., Koriashkina, L.S. Theory of Continuous Optimal Set Partitioning Problems as a Universal Mathematical Formalism for Constructing Voronoi Diagrams and Their Generalizations. I. Theoretical Foundations. Cybern Syst Anal 51, 325–335 (2015). https://doi.org/10.1007/s10559-015-9725-x

[9] Andrews, G.E. Partitions: At the Interface of q-Series and Modular Forms. The Ramanujan Journal 7, 385–400 (2003). https://doi.org/10.1023/A:1026224002193

[10] Farkas H.M. (2008) Theta Functions in Complex Analysis and Number Theory. In: Surveys in Number Theory. Developments in Mathematics (Diophantine Approximation: Festschrift for Wolfgang Schmidt), vol 17. Springer, New York, NY. https://doi.org/10.1007/978-0-387-78510-3_4

[11] Meglicki B., Generating functions partitioning algorithm for computing power indices in weighted voting games. 2010. arXiv:1011.6543 **[cs.GT]**

[12] F. P. Kelly, *Reversibility and Stochastic Networks*, New York: Wiley, 1979.

[13] D. E. Knuth, *The Art of Computer Programming, Vol. 4, Fascicle 3, Generating All Combinations and Partitions,* Addison Wesley, Upper Saddle River, NJ, 2005.

[14] D. Kresher, and D. Stinson, *Combinatorial algorithms, generation, enumeration, and search,* London: CRC Press, 1999.

[15] K. Jacobi, *Fundamenta Nova Theoriae Functionum Ellipticarum* (Königsberg,1829), see e.g. E. T. Whittaker, and G. N. Watson, *A course in modern analysis*, chap. XXI Theta functions, Cambridge, 1952.

[16] Srivastava, H.M., Chaudhary, M.P. & Chaudhary, S. A Family of Theta-Function Identities Related to Jacobi's Triple-Product Identity. Russ. J. Math. Phys. 27, 139–144 (2020). https://doi.org/10.1134/S1061920820010148

[17] (2008) The Knapsack Problem. In: Combinatorial Optimization. Algorithms and Combinatorics, vol 21. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-71844-4_17

[18] Bartlett M., Frisch A.M., Hamadi Y., Miguel I., Tarim S.A., Unsworth C. (2005) The Temporal Knapsack Problem and Its Solution. In: Barták R., Milano M. (eds) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. CPAIOR 2005. Lecture Notes in Computer Science, vol 3524. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11493853_5

[19] Chu, P., Beasley, J. A Genetic Algorithm for the Multidimensional Knapsack Problem. Journal of Heuristics 4, 63–86 (1998). https://doi.org/10.1023/A:1009642405419

[20] Feng Y, Yu X, Wang G-G. A Novel Monarch Butterfly Optimization with Global Position Updating Operator for Large-Scale 0-1 Knapsack Problems. Mathematics. 2019; 7(11):1056. https://doi.org/10.3390/math7111056

[21] J. Jormakka, "The existence of quasiregular mappings from R^3 to closed orientable 3-manifolds," *Ann. Acad. Sci. Fennica*, Series A, 1988, pp.1-44.