# MODELLING DISCRETE INPUT PROBABILITY DISTRIBUTIONS

Johann Christoph Strelen
Rheinische Friedrich–Wilhelms–Universität Bonn
Römerstr. 164,
53117 Bonn, Germany
strelen@cs.uni-bonn.de

**KEYWORDS**

Stochastic Simulation, Fitting Probability Distributions, Discrete Probability Distributions, Genetic Algorithm, Inverse Transformation Method.

## ABSTRACT

Sometimes input probability distributions for stochastic models are not so simple that standard distributions suit. In this case, we model with weighted sums of standard distributions. These composed distributions may have many parameters which must be estimated. This is not easy with common estimation methods like maximum-likelihood. We use the genetic algorithm for that. The design of the composed distributions can be done such that the inverse tranformation method for the generation of random variates is feasible.

## 1 INTRODUCTION

Stochastic models and discrete simulation are indispensable means for the quantitative analysis of systems. It is well known that missing to carefully model the influences from outside, especially the load, may lead to wrong results and ultimately to wrong decisions based on the simulation results.

Influence from outside of the model like load or failure of system components can be incorporated into the model using observed traces or input models, namely independent random variables, random vectors, or stochastic processes. Data from traces can be used directly.

If input is modeled, input data are realisations of the model. The use of random variates which are drawn from common standard distributions, is well understood and common since long time, see any book on discrete simulation like (Law and Kelton 2000). The use of generated random vectors and stochastic processes is much more difficult, not so popular, a topic of current research, see for example (Strelen and F.Nassaj 2007) and the citations there.

In this paper, we consider independent random variates with more general distributions which are composed of some standard distributions, with emphasis on discrete distributions. We show that such composed distributions are suitable for the inverse transformation method for generating random variates.

The common way to select a standard distribution is as follows. A sample of data is gathered in some real system, and a standard distribution is searched from which the data could be realizations. Usually this is done in three steps: In the first step, a specific standard distribution is hypothesized, for example Weibull with two parameters. In the second step, the distribution is fitted to the data, namely distribution parameters are estimated. In the third step, one must decide if the found distribution is representative for the data, for example with goodness of fit tests.

The ExtertFit software (Law and Kelton 2000) determines which of 39 considered probability distributions represent best some given data. To this end, the distributions are fitted to the data using maximum-likelihood estimators, and the quality of each fit is judged with goodness of fit tests.

But it may happen that there is no suitable standard distribution for the observed data. The data may indicate more than one maximum of the density or the probability function, but all standard distributions are unimodal, hence none of them fits. Or, a discrete sample is very irregular, some values occur often, the others in between seldom.

Under these circumstances, a composed distribution, namely a weighted sum of standard distributions, may be appropriate. Such a weighted sum is convenient for generating random variates.

Moreover, a distribution which is composed of standard distributions with invertible distribution functions may be invertible itself. In this case, the inverse-transform method can be applied to generating random variates. This allows for some variance reduction methods.

But such a composed distribution has many parameters which must be estimated, in general. This is a difficult task, since the common maximum-likelihood estimators lead to a system of nonlinear equations which must be solved; the number of equations equals the number of unknown parameters.

In (Strelen 2003), we proposed to apply a genetic algorithm instead which has advantages compared to the tradi-

tional way. We develop this idea further in this paper. Here we only need to distinguish three slightly different techniques, one for closed-form distribution functions, the second with numerical integration of densities, and the third for discrete probabilities - on the other hand, maximum-likelihood estimators require many individual solutions for different distributions. And, even more important, many parameters can be considered easily with the new technique.

In the first place, the genetic algorithm serves the purpose to estimate parameters in our technique, but as a byproduct, standard distributions can be judged if they are suitable, and one obtaines a measure how good the fit is.

The genetic algorithm (Back, Fogel, and Michalewicz 1997, Davis 1991, Goldberg 1989) is a stochastic global search method that mimics the metaphor of natural biological evolution. We use it to minimize the distance between an empirical distribution of the observed sample and the (composed) distribution which is to be fitted.

For numerical calculation we use MATLAB (Matlab) with the Genetic Algorithm Toolbox (Chipperfield, Fleming, Pohlheim, and Fonseca).

In section 2, we report on the genetic algorithm as far as important for our method. Section 3 describes the fitting technique, gives some examples, and points out the applicability of the inverse transformation method. Section 4 is up to an intricate example: We consider a measured sample of packet sizes in a computer network from Klemm, Lindemann, and Lohmann (2002).

# 2 THE GENETIC ALGORITHM

The genetic algorithm (GA) is a stochastic global search method that mimics the metaphor of natural biological evolution. It differs from traditional search and optimization methods, significant differences are:

- Genetic algorithms search generations of approximations in parallel, not a single sequence
- Genetic algorithms require only the objective function, no derivatives
- Genetic algorithms use probabilistic transition rules, not deterministic ones
- Genetic algorithms work on an encoding of the parameter set rather than the parameter set itself

Genetic algorithms (Back, Fogel, and Michalewicz 1997, Davis 1991, Goldberg 1989) operate on *populations* of individuals applying the principle of survival of the fittest. *Individuals* are tuples of decision variable values which are encoded as strings over the binary alphabet or other alphabets. The individuals are approximations of the desired solution, and their *fitness* measures the acuracy which is fixed by an objective function. At each *generation*, a new population is created by the process of selecting individuals according to their level of fitness and breeding them together using operators borrowed from natural genetics. The *recombination operator* is used to exchange parts of the strings between pairs, or larger groups, of individuals, according to some probabilistic rule. *Mutation* will cause a single bit to change its state with some probability, in the binary string representation. *Selection* serves the purpose to select individuals for the next generation.

This process leads hopefully to the evolution of populations of individuals that are better approximations to the solution than their parents.

We use the Genetic Algorithm Toolbox (Chipperfield, Fleming, Pohlheim, and Fonseca) which, broadly speaking, works as follows. In each problem which is to be solved here, the genetic algorithm is applied to searching for a minimum of the objective function. The result is a tuple of values of the real valued decision variables $d_i$, $i = 1, \ldots, N^{(\mathrm{var})}$. For each decision variable, a minimum and a maximum is given in advance, and the values are encoded with *PRECI* binary digits. Grey coding is used, hence adjacent values differ in just one digit. An $N^{(\mathrm{var})}$-tuple of decision values is an individual, and $N^{(\mathrm{ind})}$ individuals are a population. The encoded values of an individual are concatenated, hence form a binary string of $N^{(\mathrm{var})} * PRECI$ digits which is termed the chromosome of the individual. The chromosomes of all individuals of a population are the matrix *CHROM* with $N^{(\mathrm{ind})}$ rows, each a chromosom.

The genetic algorithm works as follows. The matrix *CHROM* is initialized with uniformly distributed random numbers. For each individual, its chromosome is decoded into a decision variable tuple $\mathbf{d}$, and the objective function $Z(\mathbf{d})$ is calculated.

In a loop, *MAXGEN* populations are calculated, one after the other: For each individual, the fitness with linear ranking is calculated according to the value of the objective function $Z(\mathbf{d})$ of its decision variables, as follows. The individuals are sorted according to descending values of their objective function values, i.e. the best individual gets the position $pos = N^{(\mathrm{ind})}$. Each individual on position *pos* gets the rank $2(pos-1)/(N^{(\mathrm{ind})}-1)$, rank 2 is the best, rank 0 the worst.

*GGAP*, $0 < GGAP \leq 1$, is the generation gap: The $(1 - GGAP) * N^{(\mathrm{ind})}$ individuals with the best fitness values remain unchanged, and the other $GGAP * N^{(\mathrm{ind})}$ individuals are selected for breeding offspring with the method of stochastic universal sampling: Each individual *i* gets a probability $p_i$ which is proportional to its fitness value. The selection is according to these probabilities where individual *i* is selected at least $\lfloor p_i N^{(\mathrm{ind})} GGAP \rfloor$ times and at most $\lceil p_i N^{(\mathrm{ind})} GGAP \rceil$ times ($\lfloor . \rfloor$ means the floor, and $\lceil . \rceil$ the ceil).

The selected chromosomes are pairwise recombined with the single-point crossover operator, each pair with probability 0.7, and with probability 0.3, the two individuals of a pair remain unchanged with respect to recombination. If the number of selected individuals is odd, one more remains unchanged.

For this crossover, a position within the two chromosomes of a pair is selected at random, and the left part of one chro-

mosome is concatenated with the right part of the other, and vice versa. The result are two new chromosomes.

Now on all chromosomes, the mutation operator is applied. Each binary digit flips with probability $0.7/L^{(\mathrm{ind})}$ where $L^{(\mathrm{ind})} = N^{(\mathrm{var})} * PRECI$ is the number of binary digits in each chromosome.

The objective function values are now calculated for offspring.

The reinsertion step performs insertion of offspring into the current population, replacing least fit parents with offspring.

Now the next population is ready and the process continues if the number of populations is less than $MAXGEN$.

The individual with the best objective function value is the result, and this value measures the accuracy of the fitted probability distribution.

# 3 FITTING INPUT PROBABILITY DISTRIBUTIONS

In principle, we proceed as usual but with two main differences: Our fitted distributions are more complex and, in general, have many parameters which must be estimated, and we search with the genetic algorithm.

As basis, a sample $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ of data is given which was measured for a specific aspect of a real system. Then a composed distribution is selected and fitted to the data,

$$\begin{aligned} F(x) &= q_1 F_1(x) + q_2 F_2(x) + \ldots + q_K F_K(x), \\ & \quad q_k > 0,\ q_1 + q_2 + \ldots + q_K = 1, \end{aligned} \tag{1}$$

where $F_1,\ F_2, \ldots$ are standard distribution functions (CDF).

But it may be sensible to modify the standard distributions. A location parameter $\beta$ or a scale paremeter $\gamma$ may be introduced if not present. Instead of the random variable $X$ with the CDF $F_k(x)$ then we consider $Y = \gamma X + \beta$ with the CDF $F_k((x-\beta)/\gamma)$ and the density $f_k((x-\beta)/\gamma)/\gamma$ where it exists. Moreover, the distribution may be cut, we consider the random variable $Z$ with the CDF

$$F_Z(x) = \begin{cases} F_k(x)/F_k(\eta), & x \le \eta, \\ 1, & \eta < x. \end{cases}$$

$\beta$, $\gamma$, and $\eta$ are additional parameters.

For the fitting, the parameters $q_k$ and parameters for each standard distribution $F_k$ are estimated. We denote the tuple of these decision variables with $\mathbf{d}$ and indicate the dependency of the distribution function sometimes writing $F_{\mathbf{d}}(x)$.

For discrete random variables with integer values, the composed distribution is

$$\begin{aligned} p(x) &= q_1 p_1(x) + q_2 p_2(x) + \ldots + q_K p_K(x), \\ & \quad q_1 + q_2 + \ldots + q_K = 1, \\ & \quad x = \ldots, -1, 0, 1, \ldots, \end{aligned} \tag{2}$$

where $p_1(x),\ p_2(x), \ldots$ are (modified) standard distributions.

Once the decision variables are estimated one decides if the fitted distribution (1) is correct.

A random number with such a composed distribution can be generated as follows: First draw a random number $k$ with the discrete distribution $(q_1, \ldots, q_K)$. Secondly, draw a random number with the distribution $F_k(x)$. We will say something more about generation of random variates in the subsection.

For fitting with the GA, we need an empirical distribution. We use a specific kind of empirical distribution functions, namely step functions. They are defined with the order statistics as follows:

$$\hat{F}(x) = \begin{cases} 0, & x < x_{(1)}, \\ i/n, & x_{(i)} \le x < x_{(i+1)},\ i = 1, \ldots, n-1, \\ 1, & x_{(n)} \le x. \end{cases} \tag{3}$$

Here, $x_{(i)},\ i = 1, \ldots, n$, denotes the ordered sequence with the elements $x_i$.

Remark 1. If in the ordered sequence $x_{(i)} = x_{(i+1)}$, then there is no $x$ for which $\hat{F}(x) = i/n$, $i/n$ is not in the range of $\hat{F}$. This occurs due to finite accuracy of real numbers in the computer and with discrete samples.

Remark 2. If all sample values $x_{(l)},\ l = 1, \ldots, n$, are different, $\hat{F}(x_{(l)}) = l/n$ holds. If not, this is true only for values $x_{(l)}$ which occur only once, and for just one of some equal values. Namely for multiple values $x_{(l_1)} = x_{(l_2)} = \ldots$, $\hat{F}(x_{(l_i)}) = \max\{l_j/n\}$ holds for $i = 1, 2, \ldots$.

For discrete random variables with integer values, we use frequency distributions,

$$\hat{p}(x) = \hat{F}(x) - \hat{F}(x-1),\ x = \ldots, -1, 0, 1, \ldots. \tag{4}$$

Again we use the notation $p_{\mathbf{d}}(x)$ to indicate the dependency of the parameters.

For continuous distributions, our objective function for the genetic algorithm is

$$Z(\mathbf{d}) = \sum_{i=1}^{n} [\hat{F}(x_i) - F_{\mathbf{d}}(x_i)]^2 \tag{5}$$

where $F_{\mathbf{d}}(x)$ is the selected (composed) distribution function with the parameter tuple $\mathbf{d}$.

$\sqrt{Z(\mathbf{d})/n}$ is a L2-norm of the difference between the empirical distribution function and the fitted distribution function and measures how accurate the distribution function $F_{\mathbf{d}}(x)$ fits the given data.

Sometimes there is no closed form of a standard distribution function available, for example if it is Gamma or Lognormal. Here we use the density $f_k(x)$ and calculate the distribution function $F_k(x)$ at the values $x_i,\ i = 1, \ldots, n$, approximately with very simple numerical integration. In particular, we take

$$\begin{aligned} \tilde{F}(x_1) &= 1/n, \\ \tilde{F}(x_i) &= 1/n + \sum_{j=2}^{i} (x_j - x_{j-1}) f_k(x_{j-1}),\ i = 2, \ldots, n, \\ F_k(x_i) &\approx \tilde{F}(x_i)/\tilde{F}(x_n),\ i = 1, \ldots, n. \end{aligned}$$

For discrete samples, the objective function is

$$Z(\mathbf{d}) = \sum_{i=1}^{n} [\hat{p}(x_i) - p_{\mathbf{d}}(x_i)]^2. \tag{6}$$

In the GA algorithm, for each decision variable, the range of possible values must be given in advance. Sometimes this is not obvious. The weights $q_k$ in (1) are restricted to $q_1 + q_2 + \ldots + q_K = 1$ - what is the range of $q_k$? We consider the parameters $u_k$ instead which have the unit range, $0 \le u_k \le 1$:

$$\begin{aligned} q_1 &= u_1, \\ q_2 &= (1-u_1)u_2, \\ q_3 &= (1-u_1)(1-u_2)u_3, \ldots, \\ q_K &= (1-u_1)(1-u_2)\ldots(1-u_{K-1}). \end{aligned}$$

If there are parameters $a_1$ and $a_2$ with $0 \le a_1 \le a_2$, we replace $a_1$ with $b_1 a_2$, $0 \le b_1 \le 1$. Moreover, we replace an integer valued parameter $j$ with $j = \lfloor y \rfloor$, $y$ real valued.

For the numerical calculation we used MATLAB (matlab) with the Genetic Algorithm Toolbox (Chipperfield, Fleming, Pohlheim, and Fonseca) which provides very comfortable functions for genetic algorithms. It remains mainly to specify the number of decision variables (parameters), their type (integer or real), their degrees of accuracy (number of binary digits), and an objective function. Further specifications concern the features of the genetic algorithm like the number of generations, the number of individuals in a population, and others.

**Example 1, Fitting Data Drawn from a Two-Mode Weibull Distribution**. The sample consists in 800 realizations according to the mixed distribution function $F(x) = 1 - 0.5\exp[-(x/3)^2] - 0.5\exp[-(x/17)^5]$, the fitted distribution function (figure 1) is $F(x) = 1 - 0.496\exp[-(x/3.03)^{1.90}] - 0.504\exp[-(x/17.1)^{5.46}]$. The accuracy is 0.004, see also figute 1.

In figures 1 and 3, the smooth curve is the fitted theoretical distribution function, and the scribbling curve which is shifted by 1 to the right in order to separate the graphs is the empirical distribution function.

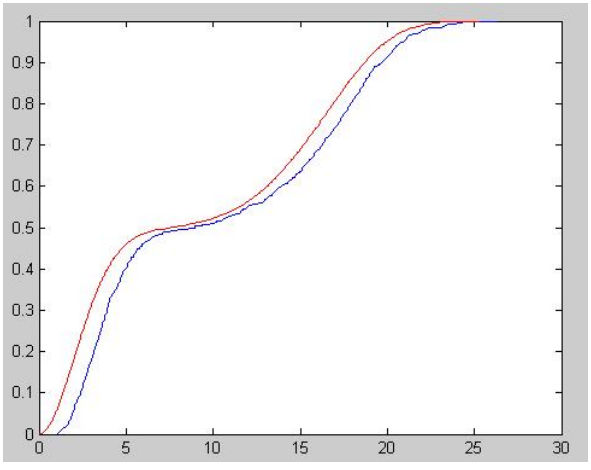


Figure 1: A Fitted Two-Mode Weibull Distribution Function

**Example 2, Fitting Data drawn from a Three-Mode Poisson distribution** The sample consists in 64000 realizations from the distribution

$$p(x) = 0.333p_8(x) + 0.333p_8(x-16) + 0.334p_8(x-32),$$

where

$$p_\lambda(x) = \begin{cases} 0, & x < 0, \\ e^{-\lambda}\lambda^x/x!, & 0 \le x, \end{cases}$$

the Poisson distribution. 16 and 32 are location parameters. The fitted distribution is

$$p(x) = 0.342p_{8.02}(x) + 0.330p_{7.06}(x-17) + 0.328p_{8.04}(x-32),$$

the accuracy is 0.0014. In the following figure, a "*" marks some sample probability and a "o" some fitted probabilities.

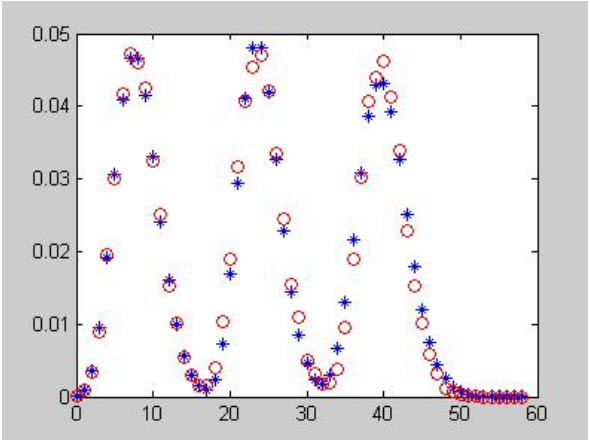


Figure 2: A Fitted Tree-Mode Poisson Distribution

**Example 3, Decision between Gamma and Weibull Distribution**. The sample consists in 800 realizations of a Weibull random variable with the parameter values $\alpha = 3$ and $\beta = 3$. It is tried to fit these data with a distribution function, mixed Weibull and Gamma,

$$F(x) = pF_1(x) + (1-p)F_2(x)$$

where $F_1$ is Weibull and $F_2$ is Gamma. After 400 generations, the best parameters are $p = 0.997$, $\alpha = 3.00$, $\beta = 3.02$ where the accuracy is 0.007, see also figure 3. That means, the genetic algorithm found out that the data should be fitted with the Weibull distribution, not with the Gamma distribution, and calculated the parameters.

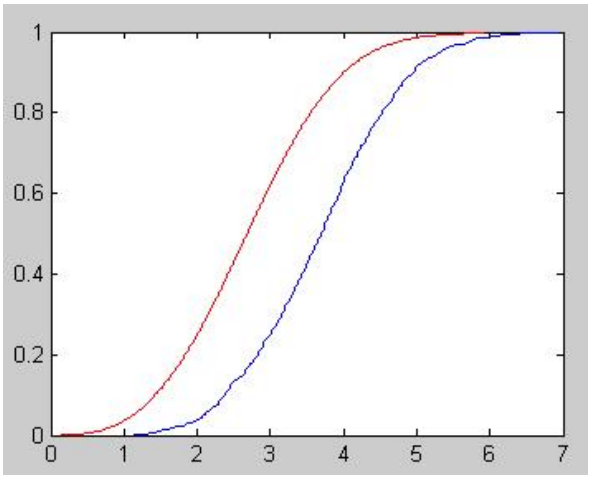


Figure 3: Decision in Favour of a Weibull Distribution

## Inverse Transformation Method

The inverse transformation method is a technique for generating random variates. With a uniformly between 0 and 1 distributed ($\mathscr{U}[0,1]$-distributed) random number u, the variate $x$ is $F^{-1}(u)$ where $F^{-1}$ is the quasi-inverse of a CDF $F$,

$$F^{-1}(u) = \begin{cases} \inf\{z|F(z) \geq u\}, & u > 0, \\ \sup\{z|F(z) = 0\}, & u = 0. \end{cases}$$

This technique has some advantages. For the generation of a random variate $x$, right one random number $u$ is needed, and $x$ is monotonically increasing with $u$, hence positively correlated. Both features are important for some variance reduction methods.

For continuous distributions, the technique needs an inverted CDF.

For discrete distributions, the inverse transformation method is easily implemented, but not fast, in general:

**Algorithm 1** Let $\{x_1,...,x_I\}$ denote the range of the distribution. The random variate is found by searching the smallest value $x_j$ where $F(x_j) \geq u$ holds.

With linear search, this needs $O(I)$ steps, with binary search $O(\log I)$ steps.

For some discrete distributions, a fast inverse transformation method is possible. We assume a random variable $X$ with an integer range $(i,...,j)$, probabilities $P\{X = x\} = p_x$, $x = i,...,j$, and the CDF $F(x)$ which is in fact a step function.

**Theorem 1** *If there is a CDF $\tilde{F}(x)$ which has the inverse $\tilde{F}^{-1}(u)$, and for which $F(x) = \tilde{F}(x)$, $x = i,...,j$, holds, then the random variable*

$$Y = \begin{cases} \lfloor \tilde{F}^{-1}(U) \rfloor + 1, & U < 1, \\ j, & U = 1, \end{cases}$$

*has the distribution $P\{Y = x\} = p_x$, $x = i,...,j$, if $U$ is $\mathscr{U}[0,1]$-distributed.*

**Proof** For $0 \leq U < \tilde{F}(1) = F(1) = p_1$, $Y = 1$ holds. For $\tilde{F}(x-1) < U \leq \tilde{F}(x)$, $Y = x$, $x = 2,...,j$, and $P\{\tilde{F}(x-1) < U \leq \tilde{F}(x)\} = P\{F(x-1) < U \leq F(x)\} = p_x$ hold. □

**Example 4** Consider the geometric distribution with parameter $p$, $0 < p < 1$. The probabilities are $p_x = p(1-p)^x$, $x = 0,1,...$, and the CDF is $F(x) = 1-(1-p)^{\lfloor x \rfloor + 1}$ for $x \geq 0$. The CDF $\tilde{F}(x) = 1-(1-p)^{x+1}$ equals the CDF $F(x)$ at $x = 0,1,...$, and has the inverse $\tilde{F}^{-1}(u) = \ln(1-u)/\ln(1-p) - 1$ for $u < 1$.

Now we define a subclass of composed distributions, namely composed distributions with non-overlapping components which allow for the inverse transformation method provided that the the inverse functions of the components are available.

**Definition 1** *A* composed distribution with non-overlapping components *is a composed distribution (1) where the following holds.*

*1. Let $\mathscr{X}_k$ denote the the range of the random variable $X_k$ of the component CDF $F_k(x)$, $k = 1,...,K$. For these ranges, $\{\mathscr{X}_1,...,\mathscr{X}_K\}$ is a partition of the range of the composed distribution. This partition is sorted as follows, namely for each $x \in \mathscr{X}_j$ and each $y \in \mathscr{X}_k$, $x < y \Rightarrow j < k$ holds.*

*2. The quasi-inverses of the components are available.*

It is well known how to generate a random variate from a composed distribution (1):

1. Generate a $\mathscr{U}[0,1]$-distributed random number $u_1$. Apply algorithm 1 with the discrete distribution $(q_1,...,q_K)$ for generating the random variate $k$.

2. Generate the desired random variate $x$ with the CDF $F_k(x)$, using one or more different, independent $\mathscr{U}[0,1]$-distributed random numbers $u_2, u_3, ...$.

For composed distributions with non-overlapping components, the inverse transformation method can be implemented as follows.

**Algorithm 2**

1. Generate a $\mathscr{U}[0,1]$-distributed random number $u$. Apply algorithm 1 with the discrete distribution $(q_1,...,q_K)$ for generating the random variate $k$.

2. With the $\mathscr{U}[0,1]$-distributed random number $u' = (u - (q_1 + ... + q_{k-1}))/q_k$, generate the desired random variate $x$ from the CDF $F_k(x)$ with the inverse transformation method (we define $(q_1 + ... + q_0) = 0$).

We want to show that this algorithm is correct and is an implementation of the inverse transformation method. To this end we note that only a single random number $u$ is used and we show that

1. $u'$ is a realization form $\mathscr{U}[0,1]$

2. This realization is independent from the variate $k$

3. The generated random variates $x$ are monotone in the random numbers $u$.

First, let $U$ be a $\mathscr{U}[0,1]$-distributed random number. Define the random variables $L$ and $U'$ such that $q_1 + ... + q_{L-1} < U \leq q_1 + ... + q_L$, $U' = (U - (q_1 + ... + q_{L-1}))/q_L$. $0 < U' \leq 1$ follows.

Moreover, for $0 < u_0 \leq 1$

$$\begin{aligned} P\{0 < U' \leq u_0 | L = k\} &= \\ P\{0 < U' \leq u_0, L = k\}/P\{L = k\} &= \\ P\{0 < (U - (q_1 + ... + q_{k-1}))/q_k \leq u_0, L = k\}/q_k &= \\ P\{0 < (U - (q_1 + ... + q_{k-1})) \leq u_0 q_k, L = k\}/q_k &= \\ P\{q_1 + ... + q_{k-1} < U \leq q_1 + ... + q_{k-1} + u_0 q_k\}/q_k &= \\ u_0 q_k/q_k &= \\ u_0. \qquad (7) \end{aligned}$$

Hence, $U'$ is $\mathscr{U}[0,1]$-distributed.

Secondly, from (7), $P\{U' \leq u_0, L = k\} = u_0 P\{L = k\}$ follows, hence $U'$ and $L$ are independent.

Thirdly, consider two random variates $x$ and $y$, $x < y$, which are generated by algorithm 2 with the $\mathscr{U}[0,1]$-distributed random numbers $u$ and $v$. Then $u < v$ holds which can be seen as follows. Either, $x$ and $y$ are elements of the same subrange $\mathscr{X}_k$. Then $u < v$ holds because $x$ and $y$ are generated with the inverse transformation method using the CDF $F_k(x)$. Or they are from different subranges, $x \in \mathscr{X}_k$ and $y \in \mathscr{X}_l$. Then, $k < l$ holds according to the definition of composed distribution with non-overlapping components, and $q_1 + ... + q_{k-1} < u \leq q_1 + ... + q_k$ and $q_1 + ... + q_{k-1} + q_k + ... + q_{l-1} < v \leq q_1 + ... + q_l$. Hence, $u < v$ holds.

Thus we have

**Theorem 2** *Algorithm 2 implements the inverse transformation method.*

Before closing this subsection about the inverse transformation method, we consider two standard distributions which are suitable as components of composed distributions with or without non-overlapping components, the single point distribution and the discrete trapezoid distribution which are not very common in simulation.

**Definition 2** *A random variable $X$ is* single point distributed *if the probability mass is concentrated at a single value $x^{(single)}$, $P\{X = x^{(single)}\} = 1$.*

**Definition 3** *A random variable $X$ has a* discrete trapezoid distribution *if the probabilities are*

$$P\{X = x\} = p_0 + \Delta(x - x_0),\ x = x_0, ..., x_0 + l - 1. \qquad (8)$$

As parameters, we use $x_0$, $l \geq 1$, integer, and $a$, $0 \leq a \leq 1$, real. For $l > 1$, we set $p_0 = 2a/l$ and $\Delta = 2(1-2a)/l/(l-1)$. For $l = 1$, $a$ must be $1/2$, and we set $\Delta = 0$. Then all probabilities sum to 1 and are non-negative.

**Remark** The probabilities are linear in $x$, the parameter $a$ determines the slope which is positive for $a < 1/2$ and negative for $a > 1/2$. For $a = 1/2$, this is the discrete uniform distribution.

The discrete trapezoid distribution is invertible, see theorem 1. For $\Delta \neq 0$,

$$F^{-1}(u) = x_0 + \begin{cases} \left\lfloor -\frac{p_0}{\Delta} + \frac{1}{2} + \text{sign}(\Delta)\sqrt{(\frac{p_0}{\Delta} - \frac{1}{2})^2 + \frac{2u}{\Delta}} \right\rfloor, & u < 1, \\ l - 1, & u = 1, \end{cases}$$

and for $\Delta = 0$,

$$F^{-1}(u) = x_0 + \begin{cases} \lfloor lu - 1 \rfloor + 1, & u < 1, \\ l - 1, & u = 1. \end{cases}$$

# 4 EXAMPLE: MODELLING A MEASURED DISCRETE SAMPLE

In this section, we consider a sample of measured packet sizes in an IP-network, see Klemm, Lindemann, and Lohmann (2002). We model the empirical data with a composed distribution, since the distribution has a quite irregular shape. The range is 32,...,1500 with 52 holes, some values occur much more often than the values surrounding them.

For a first analysis, we plotted the empirical probabilities in different scales. Thus we found the values which are not existent or which are prevalent. We model the prevalent value probabilities with 26 single point distributions. The following figure shows the empirical probabilities with logarithmic scale (but "0" means value zero). The circled points are the identified prevalent values.

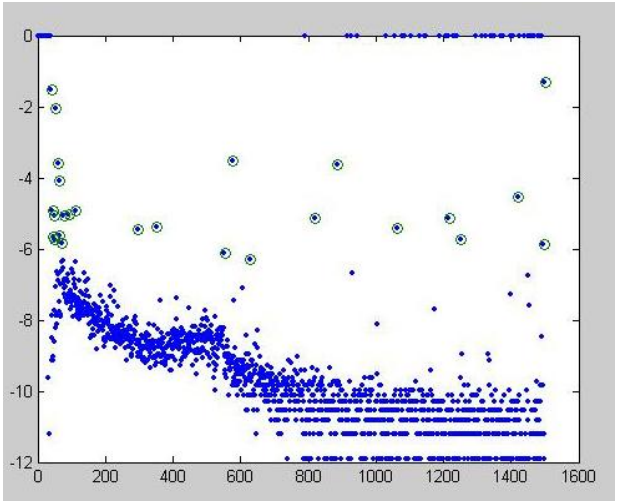


Figure 4: Empirical Probabilities in Logarithmic Scale

For all other range values, we provided six discrete trapezoid distributions, over six subranges which constitute a partition of the range. Thus we get a composed distribution with 32 components. Since the 26 prevalent values are elements of the subranges, the composed distribution is not with non-overlapping components.

Prevalent probabilities can be identified looking into the plots. Actually, we selected them algorithmically. Broadly speaking, we compared all probabilities with a moving average, and if they are considerably bigger (twofold or more), we marked them as prevalent, if they are also bigger than 0.0013.

For the discrete trapezoid distributions, we looked into the plots and defined points where we had the impression that the slope should change: They became the boundaries of the

subranges with different trapezoid distributions: 32, 75, 200, 350, 550, 800, 1500.

The fitted composed distribution, based on 40000 sample points, has a very good accuracy of 0.0002. The following figures also exhibit the accuracy.

The next figure presents the empirical CDF and the fitted CDF which is augmented by 0.1 in order to separate the graphs.

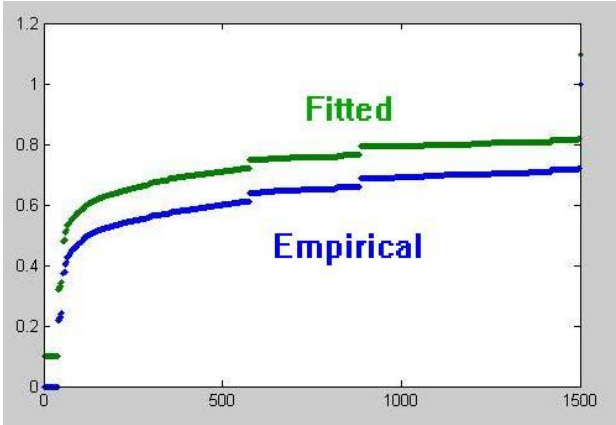


Figure 5: Empirical CDF and Fitted CDF plus 0.1

Figure 6 is a P-P plot with the fitted CDF values versus the empirical CDF values. For perfectly equal values, all points would lie on a straight line with an intercept of 0 and a slope of 1.

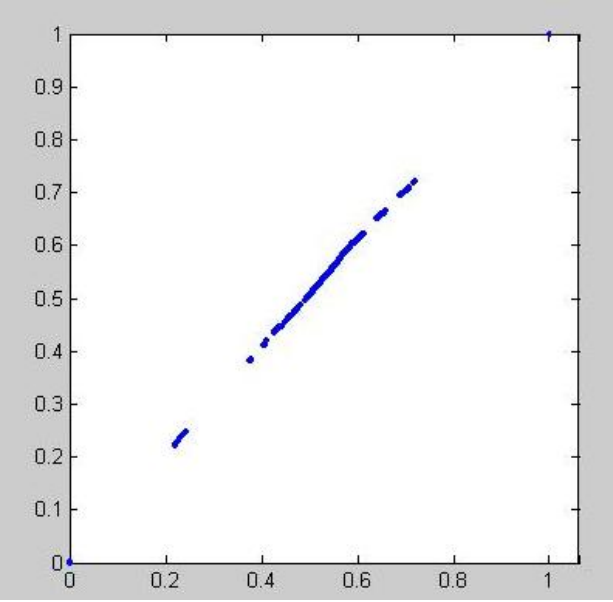


Figure 6: Probability-Probability Plot

Now we present two figures with differences, first beween the fitted CDF values and the empirical CDF values, and then beween the fitted probabilities and the empirical probabilities.

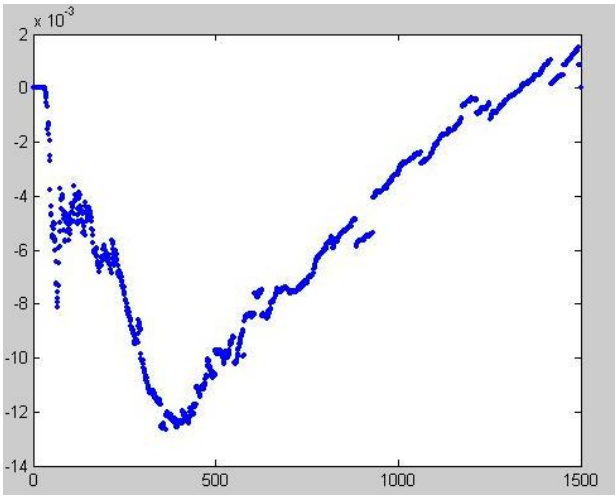


Figure 7: Differences between Fitted and Empirical CDF Values

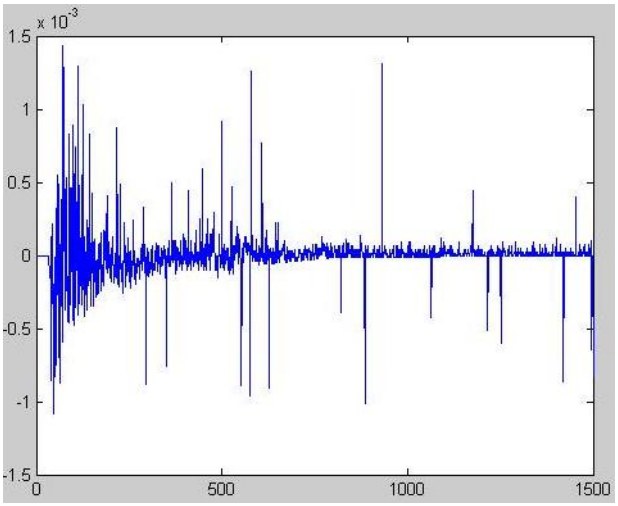


Figure 8: Differences between Fitted and Empirical Probabilities

# CONCLUSION

We presented a new technique for modelling and fitting complex input distributions for stochastic models which has some advantages compared to common techniques. These ideas are intended to be brought in a fitting tool.

# REFERENCES


Bäck, T., D. Fogel, and Z. Michalewicz. (Eds.) 1997. *Handbook of evolutionary computation*. Bristol, New York: Oxford University Press.

Chipperfield, A., P. Fleming, H. Pohlheim, and C. Fonseca. *Genetic algorithm toolbox for use with MATLAB*. http://www.shef.ac.uk/∼gaipp/ga-toolbox/.

Davis, L. (Ed.) 1991. *Handbook of genetic algorithms*. New York: Van Norstrand Reinhold.

Goldberg, D. 1989. *Genetic algorithms in search, optimization, and machine learning*. Reading,MA: Addison-Wesley.

Klemm, A., C. Lindemann, and M. Lohmann. 2002. Traffic modeling of IP networks using the batch Markovian arrival process. In *Proceedings of the 12th Int. Conf. on Modelling Tools and Techniques for Computer and Communication System Performance Evaluation*, 92–110. London: Springer. LNCS 2324.

Law, A. M., and W. D. Kelton. 2000. *Simulation modeling and analysis*. third ed. New York: McGraw-Hill.

Matlab. <www.mathworks.com>.

Strelen, J. C. 2003. The genetic algorithm is useful to fitting input probability distributions for simulation models. In *Proceedings of the Business and Industry Symposium - ASTC 2003*, ed. M. Ades and L. Deschaine, 8–13. San Diego: Society for Computer Simulation.

Strelen, J. C., and F.Nassaj. 2007. Analysis and generation of random vectors with copulas. In *To appear in the Proceedings of the 2007 Winter Simulation Conference, Washington*, ed. S.G.Henderson, B.Biller, M.-H.Hsieh, J.Shortle, J.D.Tew, and R.R.Barton.

# AUTHOR BIOGRAPHY

**JOHANN CHRISTOPH STRELEN** received the Dipl.-Math. and Dr. rer. nat. degrees in mathematics and the Habilitation degree in Computer Science from the Technische Hochschule Darmstadt, Germany. He is Professor of Computer Science at the university of Bonn, Germany. His research interests include performance evaluation, distributed systems, and simulation. His e-mail address is <strelen@cs.uni-bonn.de> and his web address is <web.cs.uni-bonn.de/IV/strelen/strelen_en.html>.