An Automatic Generator for a Large Class of Unimodal Discrete Distributions

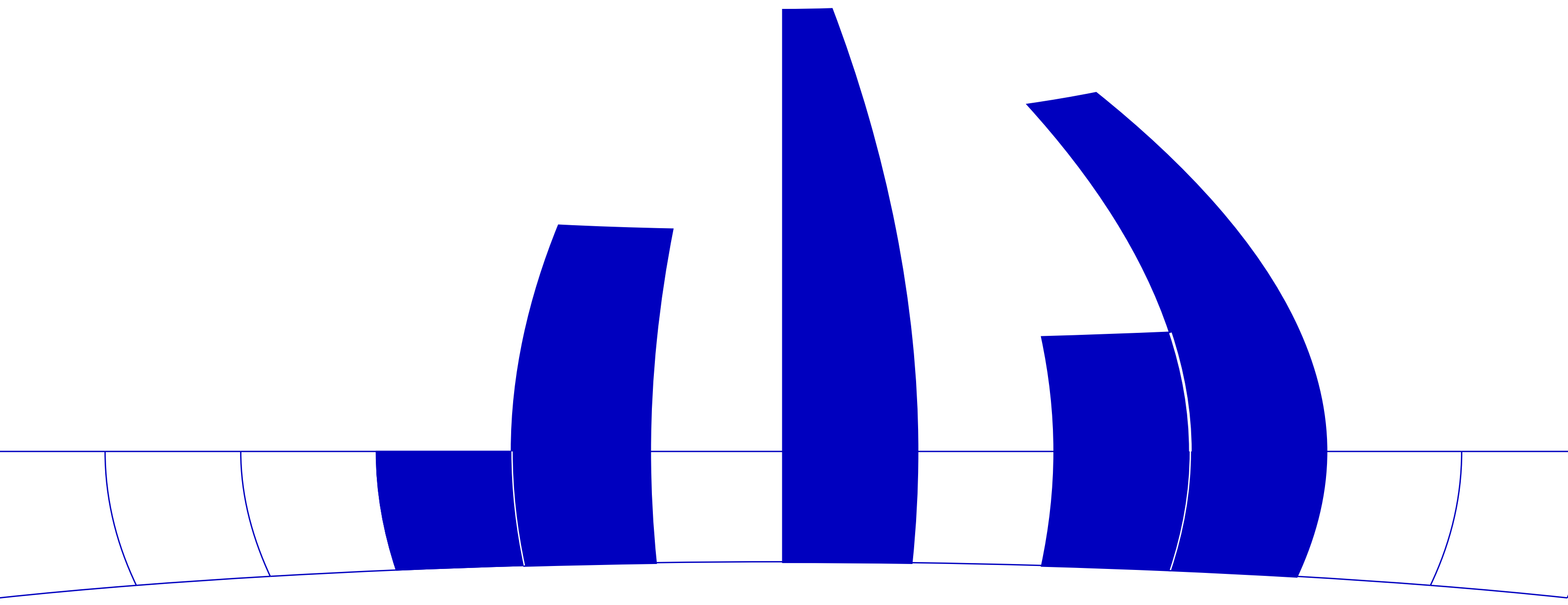
Wolfgang Hörmann and Gerhard Derflinger

Department of Applied Statistics and Data Processing
Wirtschaftsuniversität Wien

Preprint Series

Preprint 19
February 1997

http://statmath.wu-wien.ac.at/



An Automatic Generator for a Large Class of Unimodal Discrete Distributions

Wolfgang Hörmann
University of Economics and
Business Administration Vienna
now: Department of I.E.
Bogazici University, Istanbul
e-mail: whoer@statrix2.wu-wien.ac.at

Gerhard Derflinger
Department of Statistics
University of Economics and
Business Administration Vienna
Aug. 2-6 A-1090, Vienna, Austria

KEY WORDS
Random variate generation, discrete distributions, universal generator, T-concave

ABSTRACT:
The automatic Algorithm ARI developed in this paper can generate variates from a large class of unimodal discrete distributions. It is only necessary to know the mode of the distribution and to have a subprogram available that can evaluate the probabilities. In a set up step the algorithm constructs a table mountain shaped hat function. Then rejection inversion, a new variant of the rejection method for discrete distributions that needs only one uniform random number per iteration, is used to sample from the desired distribution. It is shown that the expeceted number of iterations is uniformly bounded for all T-concave discrete distributions. Utilizing a simple squeeze or an auxiliary table of moderate size, which is initialized during generation and not in the set up, Algorithm ARI is fast, at least as fast as the fastest known methods designed for the Poisson, binomial and hypergeometric distributions. The set up time of the algorithm is not affected by the size of the domain of the distribution and is about ten times longer than the generation of one variate. Compared with the very fast and well known alias and indexed search methods the set up of Algorithm ARI is much faster but the generation time is about two times slower. More important than the speed is the fact that Algorithm ARI is the first automatic algorithm that can generate samples from discrete distributions with heavy tails.

INTRODUCTION

Automatic (also called universal or black-box) algorithms that can generate random variates from a variety of different distributions have important advantages for the user as a single algorithm coded and tested only once can do the same job as a library of algorithms tailored for standard distributions. Especially for discrete distributions two universal methods are well known and frequently used (cf. (Devroye 1986) or (Dagpunar 1988), and the references given there): Inversion by sequential search, which can be accelerated by an index table, and the alias method. Both algorithms need a time consuming set up and large tables which grow linearly with the size of the domain of the distribution but the generation of the random variates itself is then very fast. Less well known are automatic rejection algorithms for log-concave discrete distributions ((Devroye 1987) and (Hörmann 1994)) and for unimodal discrete distributions with finite variance (see Devroye 1986 p. 495) which need no tables and only a short set up. The speed of these generators is slow compared with the indexed search and alias methods but not affected by the size of the domain of the distribution. Interesting but – as far as we know – not discussed in literature is the fact that the published automatic algorithms for discrete distributions can not handle distributions with heavy tails as these distributions are not log-concave and have no finite variance. In addition such distributions have a very large number of probabilities significantly larger than zero, which makes the alias method practically impossible. For the indexed search method set up and generation become extremely slow, and with or without index table the expected number of operations to generate one random variate diverges for distributions that have no expectation.

We are convinced that, although most of the classical discrete distributions are log-concave, it can be of importance for simulation practioners to have available an automatic algorithm that can handle discrete distributions with heavy tails.

So we present an automatic algorithm based on rejection-inversion, a new variant of rejection for discrete distributions which was recently developed in (Hörmann and Derflinger 1996) and is explained in Section 2. The choice of the continuous hat, discussed in Section 3, is based on the extension of the idea of transformed density rejection (Hörmann 1995) to discrete distributions. Section 4 gives the details and a formal description of the algorithm, whereas Section 5 compares the properties and performance of our new algorithm with algorithms suggested in the literature.

REJECTION-INVERSION

When the classical rejection algorithm from a continuous hat $h(x)$ is used to generate variates from a discrete distribution with probabilities p_k it is necessary that $p_k \le h(x)$ for all x between $k-1/2$ and $k+1/2$. Then we have:

Algorithm: Rejection

```
1. Generate a random variate $X$ with density proportional to $h$ and a uniform random number $V$. Let $K \leftarrow \lfloor X + 1/2 \rfloor$.
2 If $V h(X) \le p_K$ return $K$ else go to 1.
```

The main idea of rejection-inversion is now to save the second uniform random number V. For this purpose, after the evaluation of K, X is retained and used for rejection. By the point a_k the interval $(k-1/2, k+1/2)$ is divided into the interval $(k-1/2, a_k)$ of rejection and the interval $(a_k, k+1/2)$ of acceptance. If $X \ge a_k$ then k will be accepted. To compute a_k we need $H(x) = \int h(x)\,dx$ (without loss of generality we assume $\lim_{x\to\infty} H(x) = 0$) and the inverse function H^{-1} (which is also necessary to generate X by inversion). From the equation

$$\int_{a_k}^{k+1/2} h(x)\,dx = H(k+1/2) - H(a_k) = p_k$$

we get $a_k = H^{-1}(H(k+1/2) - p_k)$. This gives the acceptance condition $X \ge H^{-1}(H(k+1/2) - p_k)$. We denote the left border of the domain with m and the right border with b. For the case $k = m$ we can make rejection impossible by just defining $h(x)$ for $x \ge a_m$ only. X itself is then generated by inversion as $X = H^{-1}(U)$ where U is uniformly distributed in the interval $(H(a_m), H(b+1/2))$. Thus the acceptance condition simplifies to $U \ge H(k+1/2) - p_k$ and we have the following:

Algorithm: Rejection-Inversion

```
1 Generate a uniform random number $U$. Let
  $U \leftarrow H(a_m) + U(H(b+1/2) - H(a_m))$,
  $X \leftarrow H^{-1}(U)$ and $K \leftarrow \lfloor X + 1/2 \rfloor$.
2 If $U \ge H(K+1/2) - p_K$ return $K$ else go to 1.
```

h can be used as a hat function for rejection inversion if $\int_{k-1/2}^{k+1/2} h(x)\,dx \ge p_k$. We will restrict our attention to convex hat functions where we have the easy to check sufficient condition $p_k \le h(k)$. Of course a convex hat function can not be used to obtain a good fit for a large class of unimodal distributions but we will explain in the next section how to construct table-mountain shaped hat functions consisting of a convex left and right tail region and a uniform center part. Using exponential tails this hat function was applied for several continuous and discrete standard distributions and for universal algorithms (see (Devroye 1986), (Devroye 1987), (Hörmann 1994), (Hörmann 1995), and references given there). The well known idea of decomposition can be used to decide which of the three parts should be generated.

Transformed Probability Rejection

The automatic construction of table-mountain shaped hat functions for continuous distributions is discussed in detail in (Hörmann 1995). The idea called transformed density rejection is simple: Use a transformation $T(x)$ that transforms the density f into a concave function $g(x) = T(f(x))$. Then construct a picewise linear function $l(x)$ which is the minimum of the three lines touching $f(x)$ in the mode m, in $x_l < m$ and $x_r > m$, respectively. As g is concave we have

$$g(x) \le l(x) = \min\left(g(x_l) + g'(x_l)(x - x_l),\ g(m),\right.$$
$$\left. g(x_r) + g'(x_r)(x - x_r)\right)$$

and define $h(x) = T^{-1}(l(x))$. In (Hörmann 1995) general conditions for transformations suitable for random number generation are discussed. In this paper we restrict our attention to the only simple group that fullfills all conditions: i.e. $T_c(x) = -x^c$ for $-1 < c < 0$ and $T_0(x) = \log(x)$. It is easy to see that the corresponding table mountains have exponential tails for T_0 and tails of the form $(a + bx)^{1/c}$ for T_c. Such a hat function can be constructed if we can find a c such that $g(x) = T_c(f(x))$ is concave on its domain. In this case we call the density T_c-concave generalizing the well known property log-concave which is identical with our T_0-concave. For two times differentiable densities the condition for

T_c-concave is $f''(x)+(c-1)f'(x)^2/f(x) \leq 0 \quad \forall x$ in the domain of f. Thus the class of T_c-concave distributions is larger for c small (i.e close to -1) than for c close to 0. The last problem is the choice of the points of contact. The below theorem is a special case of theorems proven in (Hörmann 1995).

Theorem 1: Let f be a T_c-concave density with mode m. Then the area below the table-mountain $h(x) = T^{-1}(l(x))$ is minimized when x_r and x_l fulfill the condition

$$f(x) = f(m)\left(\frac{1}{c+1}\right)^{1/c} \text{ for } c<0 \quad \text{and}$$

$$f(x) = f(m)/e \text{ for } c = 0.$$

The area below the hat function is equal to $f(m)(x_r - x_l)$ and bounded for all T_c-concave distributions by

$$t_o = \frac{1}{1-(1/(1+c))^{1+1/c}} \text{ for } c<0 \quad \text{and}$$

$$t_o = \frac{e}{e-1} = 1.582\ldots \text{ for } c = 0.$$

With the choice $x_l = m - t_o/f(m)$, $x_r = m + t_o/f(m)$, the area below the hat function is lower or equal $2t_o$ for arbitrary T_c-concave distributions.

To obtain a similar theorem for discrete distributions we need a continuous continuation of the p_k's. So we define:

$$\tilde{g}(x) = T_c(p_{\lfloor x\rfloor})(1-(x-\lfloor x\rfloor)) + T_c(p_{\lfloor x\rfloor+1})(x-\lfloor x\rfloor)$$

$$\tilde{f}(x) = T_c^{-1}(\tilde{g}(x)).$$

A discrete distribution is called T_c-concave if there exists a T_c-concave continuation of the p_k's. It is obvious that for such distributions $\tilde{f}$ is the minimal T_c-concave continuation and that $\int \tilde{f}(x)\,dx \leq 1$. This is enough to see that the below theorem is a direct consequence of Theorem 1.

Theorem 2: Let $\tilde{f}$ be the the minimal T_c-concave continuation of a T_c-concave discrete distribution with mode m. Then the area below the table-mountain $h(x) = T^{-1}(l(x))$ of $\tilde{f}$ is minimized when $\tilde{x}_r$ and $\tilde{x}_l$ fulfill the condition

$$\tilde{f}(\tilde{x}) = p_m\left(\frac{1}{c+1}\right)^{1/c} \text{ for } c<0 \quad \text{and}$$

$$\tilde{f}(\tilde{x}) = p_m/e \text{ for } c = 0.$$

The area below the hat function is equal to $p_m(\tilde{x}_r - \tilde{x}_l)$ and bounded for all T_c-concave discrete distributions by

$$t_o = \frac{1}{1-(1/(1+c))^{1+1/c}} \text{ for } c<0 \quad \text{and}$$

$$t_o = \frac{e}{e-1} = 1.582\ldots \text{ for } c = 0.$$

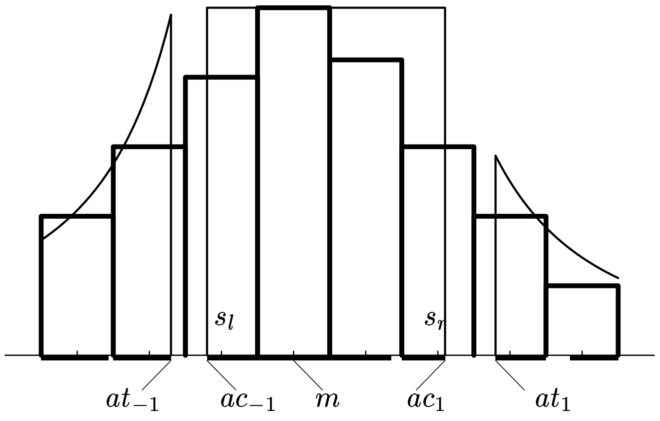
With the choice $\tilde{x}_l = m - t_o/p_m$, $\tilde{x}_r = m + t_o/p_m$, the area below the hat function is lower or equal $2t_o$ for arbitrary T_c-concave discrete distributions.

To construct the hat $h(x)$ according to Theorem 2 it is enough to determine the integer $x_l = \lfloor \tilde{x}_l \rfloor$ and to compute $\tilde{g}'(x_l)$ as $\tilde{g}(x_l+1) - \tilde{g}(x_l)$ and for x_r analogously. This rule is also correct for the case that $\tilde{x}_l$ itself is an integer and $\tilde{g}'(\tilde{x}_l)$ does not exist, but a proof of this fact requires some details of the proof of Theorem 1.

The Algorithm

We can not use rejection-inversion for the whole table-mountain constructed according to the above theorems because, as it is not convex in the cutting points $\tilde{s}_l$ and $\tilde{s}_r$ of tail and center part, the condition $\int_{k-1/2}^{k+1/2} h(x)\,dx \geq p_k$ might not be fulfilled. So we use the uniform head for $s_l \leq k \leq s_r$ where s_l and s_r are integers defined as $s_l = \lfloor \tilde{s}_l + 0.5 \rfloor$ and $s_r = \lfloor \tilde{s}_r + 0.5 \rfloor$. The area below the hat is therefore a bit larger but we can use the "trick" explained before the Algorithm Rejection-Inversion to make rejection impossible for the border points s_l, s_r, $s_l - 1$ and $s_r + 1$. The last part of Theorem 2 remains correct and the area below the hat function is always smaller than $2t_o$ for that version.
One consideration that therefore must be added to the description of rejection inversion is that for $k \geq s_r + 1$ and for $s_l \leq k < m$ the area of acceptance is in the interval $(a_k, k+1/2)$. For other values of k it is better to define a_k such that the interval $(k-1/2, a_k)$ is the region of acceptance. Figure 1 shows the three parts of the hat (thin lines) and the histogramm (thick lines) of the desired distribution. The thick parts of the x-axis denote the regions of acceptance, the variable names *ac* (= *a* center) and *at* (= *a* tail), -1 (= left) and 1 (= right) are also used in the description of Algorithm ARI.

Figure 1



Addressing the choice of the transformation it is clear that among the class T_c with $-1 < c \leq 0$ the cases $c = 0$ and $c = -1/2$ have the simplest form and are fastest to compute. The largest class of distributions can be generated for c close to -1. On the other hand Theorem 2 states that the area below the table-mountain hat is bounded by t_o or $2t_o$ which is monotonly decreasing in c and diverges for $c \to -1$. Table 1 gives all information about T_c necessary to code the functions required for the below Algorithm.

Table 1

c	0	c	$-1/2$
$T_c(x)$	$\log(x)$	$-x^c$	$-1/\sqrt{x}$
$T_c^{-1}(x)$	e^x	$(-x)^{1/c}$	x^{-2}
$F(x)$	e^x	$\frac{-(-x)^{1+1/c}}{1+1/c}$	$-1/x$
$F^{-1}(x)$	$\log(x)$	$(x(1+1/c))^{\frac{c}{1+c}}$	$-1/x$
t_o	$\frac{e}{e-1}$	$\frac{1}{1-\left(\frac{1}{1+c}\right)^{1+\frac{1}{c}}}$	2

The choice of the points of contact depends on the information available for the desired distribution. If the location of the mode of the distribution is not known it is necessary to use numerical search to find it. It is also possible to design a rejection inversion algorithm similar to that below but without the center part. To compute the optimal values of x_l and x_r if the mode is known a search algorithm that includes many evaluations of the p_k is necessary. One advantage of this method is that it is enough to know the p_k's up to proportionality but the set-up becomes really slow and depends on the size of the domain of the distribution. A second possibility is to use the last part of Theorem 2 (the minimax approach) for the choice of x_l and x_r. It guarantees that the area below the hat is uniformly bounded for all T_c-concave distributions but is far away from optimal for many standard distributions. Therefore we think it is better to take a value close to the optimal value for the normal distribution and $c = -1/2$ (which is $x_r = m + 0.664/p_m$) as many of the classical discrete distributions have the normal distribution as limiting case. If the area below the hat becomes larger than $2t_o$ for this choice of x_r and x_l take the minimax approach as last resort that guarantees that the area below the hat is uniformly bounded.

For the performance of random variate generation algorithms for discrete distributions the expected number of evaluations of the probabilities is of great importance as they are expensive to evaluate for nearly all discrete distributions. In (Hörmann and Derflinger 1996) Theorem 2 it is proven that for T_c-concave monotone distributions $m - a_m$ is a lower bound for $k - a_k$ for k between m and the point of contact x_r. This bound can be used as a "squeeze" as k can be accepted without the evaluation of a probability if $m - a_m > k - x$

A second possibility is to add a table of arbitrary size that stores the right hand side of the acceptance condition for the values of k in an interval containing the mode. The important difference to the table-helped alias and inversion algorithms is that the table is not calculated in a set-up but only during generation if a value is needed it is also stored for later use. Of course the speed up of this variant depends on the size of the help-table used and on the size of the main part of the distribution but for example a table with 1000 floating point numbers and a table of 1000 logical variables that stores, whether a value was already computed, taking only 9 K of memory on our machine brings a realy remarkable speed-up for most distributions in practical use. In the below description of the algorithm the lines implementing the squeeze step are marked with "*", the lines utilizing the auxiliary table are marked with "+". For most distributions it is enough to take one of the two possibilities either the squeeze or the auxiliary table. The algorithm becomes slow but remains correct when omitting all lines marked with "+" or "*".

Algorithm ARI (Automatic Rejection Inversion):

```
0 (Set-up) Prepare as macros or functions $T(x)$,
  $F(x)$, $F^{-1}(x)$ (taking a fixed value for $c$ and the
  information given in Table 1) and $P(k) = p_k$.
  Let $b_{-1}$ be the smallest integer of the domain
  and $b_1$ the largest ($b_i$ may be $\pm\infty$ or the small-
  est/largest integer representable on the used
  computer as well). Set $m$ to the mode of the
  distribution and $d = \max(2, \lfloor 0.664/P(m) \rfloor)$
```

```
0.1 Do for $i = \pm 1$:
        Set $x_i = m + i * d$,
        if$(i * x_i + 1 > i * b_i)$ set $v_i = 0$ and $s_i = b_i$.
        else set $y_i = T(P(x_i))$,
            $ys_i = i * (T(P(x_i + i)) - y_i)$,
            $s_i = \lfloor 0.5 + x_i + (T(P(m)) - y_i)/ys_i \rfloor$.
            Set $Hat_i = F(y_i + ys_i * (s_i + i * 1.5 - x_i))/ys_i - i * P(s_i + i)$,
            $at_i = x_i + (F^{-1}(ys_i * Hat_i) - y_i)/ys_i$,
            $xsq_i = i * (at_i - (s_i + i))$,
            $v_i = i*(F(y_i+ys_i*(b_i+i*0.5-x_i))/ys_i - F(y_i + ys_i * (at_i - x_i))/ys_i)$.
        Set $ac_i = s_i + i * (P(s_i)/P(m) - 0.5)$.

0.2 Set $v_c = P(m)*(ac_1 - ac_{-1})$, $v_t = v_c + v_{-1} + v_1$,
    $v_{cr} = v_c + v_1$. Set $t_o = 1/(1-(1/(1+c))^{(1+1/c)})$.
    If $v_t > t_o$ set $d = \lfloor t_o/P(m) \rfloor$ and go to step 0.1.

0.3 Let $N$ be the size of the auxiliary table.
    Set $n_{-1} = \max(b_{-1}, m - \lfloor N/2 \rfloor)$
    and $n_1 = n_{-1} + N - 1$.
    If $(n_1 > b_1)$ set $n_1 = b_1$ and $n_{-1} = n_1 - N + 1$.
    Prepare for the range $(n_{-1}, n_1)$ a table $hb$ of
    boolean variables initialized to "true" and a table
    $hp$ of floats, which need not be initialized.

1.0 Generate a uniform random number $U$ and set
    $U = U * v_t$.

1.1 If $(U \le v_c)$
        set $X = U * (ac_1 - ac_{-1})/v_c + ac_{-1}$,
        $k = \lfloor X + 0.5 \rfloor$
        If$(k < m)$ set $i = -1$ else set $i = 1$.
  *     If $(i * (ac_i - s_i) > i * (X - k))$ return $k$.
  +     if $(i * k \le i * n_i)$
  +         if$(hb_k)$ set $hp_k = 0.5 - P(k)/P(m)$
  +             and $hb_k =$"false".
  +         Set $h = hp_k$.
  +     else
            set $h = 0.5 - P(k)/P(m)$.
        If $(h \le i * (k - X))$ return $k$
        else go to step 1.0.

1.2 else if $(U \le v_{cr})$ set $i = 1$, $U = U - v_c$
    otherwise set $i = -1$, $U = U - v_{cr}$.
    Set $U = Hat_i + i * U$,
    $X = x_i + (F^{-1}(U * ys_i) - y_i)/ys_i$
    and $k = \lfloor X + 0.5 \rfloor$.
  * If $(i * k \le i * x_i + 1$ and $xsq_i \le i * (X - k))$
  *     return $k$.
  + if $(i * k \le i * n_i)$
  +     if$(hb_k)$ set $hp_k = i * F(y_i + ys_i * (k + i * 0.5 - x_i))/ys_i - P(k)$ and $hb_k =$"false".
  +     Set $h = hp_k$.
  + else
        set $h = i * F(y_i + ys_i * (k + i * 0.5 - x_i))/ys_i - P(k)$.
    If $(i * U \ge h)$ return $k$ else go to step 1.0.
```

APPLICATION

To judge the value of an automatic algorithm it is important to discuss its possible range of applications. To use Algorithm ARI the location of the mode and a function to evaluate the p_k's are necessary. Algorithm ARI also works if the p_k's are only known up to proportionality but a rough estimate ($\pm 30\%$ are no problem) of the modal probability is required to choose the points of contact.

Algorithm ARI is guaranteed to work for T_c-concave distributions. For families of distributions which are T_c-concave for a fixed c Theorem 2 implies that the expected number of iterations is uniformly bounded. This is of course true for the important class of log-concave distributions (including among others the Poisson, binomial, negative binomial and hypergeometric distributions). In this case it is possible and – according to our experience – fastest to take $c = 0$ but $c = -1/2$ is a good choice too. For unimodal discrete distributions with heavier tails it is best to try $c = -1/2$ first. Then it is possible to use Mathematica or some other mathematical software package to check whether the distribution is concave for that c. (This can be done for example by plotting the linear interpolation of $T(p_k)$.) If $c = -1/2$ is not enough it is best to try a c a bit smaller than $1/a$ if the tails of the distribution are proportional to x^a. For a family of distributions this is more difficult as it is often necessary to find a c as function of the parameters of the distribution. We found such choices of c for the Zipf (or Zeta), the digamma and the trigamma distributions (Johnson et al. 1992). If c tends to -1 for certain parameters Theorem 2 does no longer guarantee that the area below the hat is uniformly bounded for that family but it is still possible.

For the case that it is not possible (or too time consuming) to find a c that guarantees T_c concavity of the distribution it is important to note that T_c concavity is a sufficient but not a necessary condition. For the validity of Algorithm ARI without squeeze it is enough that the used hat is above the p_k's and this is possible for distributions as long as the region about the point of contact is T_c-concave and the rest of the distribution is not too T_c-convex. It is possible to check the validity of the hat in a setup step, but this requires the evaluations of all p_k's and then the use of the alias or the indexed search method to generate the random variates is certainly more appropriate. A second possibility is to check, that the hat is larger than the probabilities, during generation for those k's for which the evaluation of p_k is necessary. Of course there remains the danger that the generated variates have not exactly the required distribution but generating large samples without detection of errors implies that differences between generated and required distribution are very small.

It is impossible to give a "fair" comparison of different automatic algorithms for discrete distributions or between universal algorithms and algorithms tailored for a (parameter) family of distributions. Especially comparative timings are depending heavily on the distributions, on the speed of the uniform generator used and on the form the evaluation of the p_k's is implemented. Is it fair in the comparisons to include an auxiliary table of small or moderate size as it is possible for Algorithm ARI? Therefore we present no table with execution times but we try to describe the characteristics. This assessment of generators is partly based on comparative timings we did on a PC and a DEC-station using our C-implementation of the different methods for the Poisson, binomial and hypergeometric distributions.

As stated in the introduction one main advantage of Algorithm ARI is that it is the first automatic algorithm that works for a large class of discrete distributions with heavy tails. This is not the case for the universal methods suggested in the literature. The two table methods alias (ALI) and indexed search (IS) are very fast but require a really time consuming setup. Among the automatic algorithms based on rejection the two variants of DEV (cf. (Devroye 1987) and (Devroye 1986 p. 495)) are slow but simple and almost without setup. DLC (Hörmann 1994) is faster but more complicated and with longer setup. Concerning the performance characteristics for distributions that can be generated by all of the above universal methods ARI lies in the middle in several aspects. Of course it is more closely related to the rejection algorithms but it requires only one uniform random number per iteration and the expected number of uniforms required to generate one discrete variate is therefore below 1.5 for most distributions which is considerably less than for DEV and DLC. The marginal execution time to generate one variate is between the very fast table methods and the rejection algorithms. Using an auxiliary table of moderate size ARI is not more than two times slower than ALI and IS but at least twice as fast as DLC and six to ten times faster than DEV. Without an auxiliary table ARI is still a bit faster than DLC and considerably faster than DEV. For the setup time ARI is between the two groups again. The time for the setup is uniformly bounded and not affected by the size of the domain of the distribution but it requires the evaluation of 9 probabilities and is therefore almost twice as slow as the setup of DLC. Compared with algorithms tailored for standard distributions ARI using a moderate auxiliary table is at least as fast as the fastest methods for the Poisson, binomial and hypergeometric (Stadlober and Niederl 1993) and Zipf (Hörmann and Derflinger 1996) distributions as long as the parameters of the distributions are fixed. For the varying parameter situation ARI is of course much slower than algorithms tailored for a certain distribution.

Summarizing we are convinced that the proposed Algorithm ARI is useful to generate discrete random variates. It is the first automatic generator that can cope with discrete distributions with heavy tails, it is uniformly fast for a large class of distributions, the expected number of uniforms required is small and the setup is not too time consuming. Concerning the generation time Algorithm ARI is really fast, especially when the memory for an auxiliary table is available.

REFERENCES

J. Dagpunar. 1988. *Principles of Random Variate Generation.* Clarendon Press, Oxford.

L. Devroye. 1986. *Non-Uniform Random Variate Generation.* Springer-Verlag, New-York.

L. Devroye. 1987. "A simple generator for discrete log-concave distributions." *Computing,* 39, 87–91.

W. Hörmann. 1994. "A universal generator for discrete log-concave distributions." *Computing,* 52, 89–96.

W. Hörmann. 1995. "A rejection technique for sampling from T-concave distributions." *ACM Transactions on Mathematical Software,* 21, 182–193.

Hörmann, W. and G. Derflinger. 1996. "Rejection-inversion to generate variates from monotone discrete distributions." *ACM Transactions on Modelling and Computer Simulation,* 6, 169–184.

Johnson, N.L., Kotz, S. and A.W. Kemp. 1992. *Univariate Discrete Distributions.* 2 edn., John Wiley, New-York.

Stadlober, E. and F. Niederl 1994. *Generation of non-uniform random variates with C-Rand.* Tech. rept. 15. Inst. of Statistics, Technical University Graz, Austria.