Computing 46, 165–173 (1991)

Computing

Expected Time Analysis of a Simple Recursive Poisson Random Variate Generator

L. Devroye, Montreal

Received October 10, 1989

Abstract — Zusammenfassung

Expected Time Analysis of a Simple Recursive Poisson Random Variate Generator. We consider a well-known recursive method for generating Poisson random variables with parameter λ, and show how it can be manipulated to produce random variates at an expected time cost of $O(\log\log\lambda)$. Despite the fact that the expected time is not uniformly bounded in λ, the algorithm should prove useful for extremely large values of λ because virtually all numerical problems associated with the evaluation of the factorial function are eliminated. The probabilistic analysis presented here is applicable in other situations as well, in which there are a random number of levels of recursion.

1980 AMS Subject Classification: 65C10, 68J05

Key words and Phrases: Random variate generation. Poisson distribution. Recursive algorithm. Expected time analysis. Probabilistic methods.

Untersuchung des mittleren Zeitverbrauchs für die rekursive Erzeugung von Poisson-verteilten Zufallszahlen. Wir betrachten einen bekannten Generator für Poisson-verteilte Zufallszahlen mit Parameter λ und ändern ihn so ab, daß der mittlere Zeitbedarf $O(\log\log\lambda)$ wird. Die Zeit ist zwar in λ nicht gleichmäßig beschränkt, der Generator ist aber gerade für sehr große λ nützlich, weil die Berechnung von Faktoriellen vermieden wird. Die hier beschriebene Methode läßt sich auch in anderen Situationen anwenden, wenn die Anzahl der Rekursionen selbst Zufallsveränderliche ist.

1. Introduction

In this paper, we consider random variate generation for the Poisson distribution defined by the probabilities

$$p_i = \frac{\lambda^i e^{-\lambda}}{i!}, \qquad i \geq 0.$$

The Poisson distribution plays a key role in probability, statistics and simulation. Hence the need to have a battery of good random variate generators. Desirable properties include

1. Time efficiency for fixed λ.
2. Universally bounded expected time (over all λ).
3. Space efficiency for fixed λ.
4. Space efficiency for the entire range of λ's.

5. Number of uniform random variates needed.
6. Numerical accuracy of the solution.
7. Length of the program.
8. Understandability of the algorithm.
9. Lack of reliance on external non-uniform random variate generators.
10. Independence from external function calls.

These features are sometimes contradictory. Table methods are notoriously fast, but require considerable space, and do not allow frequent changes in λ between calls (Devroye, 1986, Ch. 7). Moderately fast methods with good space efficiency and uniformly bounded expected times were developed in Devroye (1980, 1987), Ahrens and Dieter (1980, 1982, 1987), Schmeiser and Kachitvichyanukul (1981), Kachitvichyanukul (1982), Ahrens, Kohrt and Dieter (1983), and Stadlober (1988), using standard principles such as rejection, acceptance-complement and ratio-of-uniforms. All of these papers, without exception, have had to deal with the serious problem of computing logarithms of large factorials when λ is large. For surveys on Poisson random variate generation, one can consult Devroye (1986) or Stadlober (1988), where most algorithms are compared in extensive tests.

In this paper, we reconsider an old recursive method mentioned e.g. in Ahrens and Dieter (1974), Pokhodzei (1984) and Stadlober (1988). It is simple in conception, and effectively avoids the computation of large factorials. In addition, its expected time complexity grows extremely slowly ($O(\log\log\lambda)$) as $\lambda \to \infty$. We also believe that the probabilistic analysis of the given recursive method may prove fruitful in other situations as well. To understand the algorithm, we require the following basic properties of the Poisson distribution.

A. The sum of two independent Poisson random variables with parameters μ and λ is Poisson $(\mu + \lambda)$.

B. Let X be a gamma (n) random variable. Given X, let Y be a binomial $(n-1, (X-\lambda)/X)$ random variable when $X \geq \lambda$, and let Z be a Poisson $(\lambda - X)$ random variable when $X < \lambda$. Then
$$(n + Z)I_{[X<\lambda]} + (n - 1 - Y)I_{[X \geq \lambda]}$$
is Poisson (λ), where I is the indicator function.

Property B is shown in Pokhodzei (1984) among other places. In fact, it requires no proof when we consider a homogeneous Poisson point process on the line with unit intensity, and keep in mind that in such a process the intervals are i.i.d. exponentially distributed random variables. Thus, X is the position of the n-th point in the point process. The returned random variate is the cardinality of the interval $(0,\lambda)$. If $X < \lambda$, we need to count how many points fall in the interval (X,λ), and this is obviously Poisson with parameter $\lambda - X$. If however $X \geq \lambda$, then $n-1$ points fall uniformly and at random in the interval $(0, X)$. To obtain the number falling in $(0,\lambda)$, just subtract from $n-1$ a binomial $(n-1, (X-\lambda)/X)$ number of points.

Property B allows us to generate a Poisson random variate recursively by choosing n in a careful fashion. Consider first the following naive nonrecursive version, with $\lambda = n$ integer.

```
  generate a gamma $(n)$ random variate $X$
  if $X < n$
     then return $n + Z$
        ($Z$ is a Poisson $(n - X)$ random variate
        generated by a simple linear time method
        (Devroye, 1986, pp. 504–505))
     else return $n - 1 - Y$
        ($Y$ is a binomial $\left(n - 1, \frac{X - n}{X}\right)$ random variate generated
        by a waiting time method such as shown in Devroye (1986), p. 525))
```

Assume that a gamma random variable is generated in uniformly bounded expected time. The expected time in case $X < n$ is bounded by $O(1 + \mathbf{E}|n - X|) = O(1 + \sqrt{n})$. Similarly, in case $X \geq n$, the expected time is bounded by $O(1 + \mathbf{E}|X - n|) = O(1 + \sqrt{n})$. Even in this form, the algorithm is quite attractive. Indeed, the numerical problems alluded to earlier are no longer present, since they are effectively avoided in the simple Poisson and binomial generators mentioned above.

Next, suppose we recurse based upon the choice $n = \lfloor \lambda \rfloor$. Then we don't really gain much since we inherit the $O(\sqrt{n})$ complexity from the binomial portion of the algorithm. An improvement is possible however if we carefully choose $n - \lambda$ so that the binomial portion is reached rather infrequently, while keeping the recursion parameter $\lambda - X$ reasonably small so that the recursion step is efficient. The crucial choice is $n - \lambda \simeq -\sqrt{c\lambda \log \lambda}$ for some constant $c > 2$. However, to make our argument more transparent, and to provide a simpler analysis, we will take for $\lambda > 1$,

$$n \overset{\text{def}}{=} \lceil \lambda - \lambda^p \rceil,$$

where $p \in (\frac{1}{2}, 1)$ is a constant to be picked later. We also stop recursing when the Poisson parameter drops below a threshold level $t > 2^{1/(1-p)}$. The choice of n makes the algorithm shown below different from Ahrens and Dieter (1974) or Pokhodzei (1984).

```
Recursive Poisson generator
if $\lambda \leq t$
   then return a Poisson $(\lambda)$ random variate
      generated by a waiting time method (page 524 of Devroye, 1986)
   else compute $n \leftarrow \lceil \lambda - \lambda^p \rceil$
         generate a gamma $(n)$ random variate $X$ by a uniformly
            fast algorithm
         if $X \geq \lambda$ then return $n - 1 - Y$
                  ($Y$ is a binomial $\left(n - 1, \frac{X - \lambda}{X}\right)$ random variate
                  generated by a waiting time method (page 524
                  of Devroye, 1986))
               else return $n + W$
                  ($W$ is a Poisson $(\lambda - X)$ random variable generated
                  by recursing)
```

The main result of this paper is

Theorem 1. *Let $t > 2^{1/(1-p)}$ and $p \in (\frac{1}{2}, 1)$ be fixed constants. Then, for $\lambda \geq t$, the recursive algorithm takes expected time bounded by $C + D \log\log \lambda$, where C and D are constants depending upon t and p only.*

2. Analysis of a Related Algorithm

The recursive algorithm shown above has a random number of recursive levels which is possibly unbounded, but remains in fact below $O(\log\log \lambda)$ with high probability. To break the analysis up into understandable independent pieces, we first consider an algorithm in which the number of recursive levels is deterministically bounded by a constant plus $O(\log\log \lambda)$.

Modified recursive Poisson generator

```
if $\lambda \leq t$ then return a Poisson ($\lambda$) random variate
          generated by a waiting time method (Devroye, 1986, p. 524)
        else compute $n \leftarrow \lceil \lambda - \lambda^p \rceil$
          generate a gamma ($n$) random variate $X$ by a uniformly
            fast algorithm
          case
            A: $X \geq \lambda$: return $n - 1 - Y$
              ($Y$ is a binomial $\left(n - 1, \frac{X - \lambda}{X}\right)$ random variate
              generated by a waiting time method
              (Devroye, 1986, p. 524))
            B: $X \leq n - \lambda^p$: return $n + W + Z$
              ($W$ is Poisson ($\lambda + \lambda^p - n$) generated by recursing)
              ($Z$ is Poisson ($n - \lambda^p - X$) generated by
              a simple method (Devroye, 1986, pp. 504–505))
            C: $n - \lambda^p < X < \lambda$: return $n + W$
              ($W$ is a Poisson ($\lambda - X$) random variable generated
              by recursing)
```

The correctness of the algorithm flows from the observation that in view of property A, case B returns X plus a Poisson ($\lambda - X$) random variable W, just as in case C. We will show that it is extremely unlikely to enter cases A or B when λ is "large". Thus, we keep going to case C, thereby reducing the parameter quickly. In at most a double-logarithmic number of recursive steps done this way, the parameter becomes $O(1)$, at which time we may occasionally enter case B, or we may even quit because of case A or because $\lambda < t$. At this stage of the algorithm, λ is so small that none of these excursions contributes anything substantial to the overall expected complexity.

Theorem 2. *Let $t > 2^{1/(1-p)}$ and $p \in (\frac{1}{2}, 1)$ be fixed constants. Then, for $\geq t$, the modified recursive algorithm takes expected time bounded by $C + D \log\log \lambda$, where C and D are constants depending upon t and p only.*

3. Proof of Theorem 2

The key in the proof is the observation that we cannot recurse too often. Indeed, if we begin with a parameter λ, then after one recursion, the input parameter is not larger than

$$\lambda + \lambda^p - n = \lambda + \lambda^p - \lceil \lambda - \lambda^p \rceil \leq 2\lambda^p .$$

After k levels of recursion, the input parameter does not exceed

$$2^{1+p+\cdots+p^{k-1}}\lambda^{p^k} \leq 2^{1/(1-p)}\lambda^{p^k}.$$

When this drops below t, we cannot possibly recurse any further. Hence the number of recursive levels does not exceed

$$\frac{\log\log\lambda - \log\log(t2^{-1/(1-p)})}{\log(1/p)}.$$

Thus, the expected time cost due to the fixed portions of the algorithm (if and case statements, gamma generator) is bounded by $A + B\log\log\lambda$ for some constants A, B depending upon p, t only. Note also that without the middle case statement, we would not have had a deterministic upper bound on the number of recursions, hence its inclusion. The additional expected time complexity is entirely due to the simple binomial and nonrecursive Poisson calls. Employing time bounds for these algorithms mentioned e.g. in Devroye (1986), we note that if X is gamma (n), the expected time bound due to the binomial portion does not exceed

$$\sup_{\lambda \geq t} \mathbf{E}\left\{\left(1 + (n-1)\frac{X-\lambda}{X}\right) I_{[X \geq \lambda]}\right\}, \tag{1}$$

while the nonrecursive Poisson calls in part **B** of the case statement yield expected time not exceeding

$$(A + B\log\log\lambda) \sup_{\lambda \geq t} \mathbf{E}(n - \lambda^p - X) I_{[X < n - \lambda^p]} . \tag{2}$$

The expected value in (1) does not exceed

$$\begin{aligned} \mathbf{E}(1 + X - \lambda) I_{[X \geq \lambda]} &\leq \mathbf{E}(1 + X - n) I_{[X \geq \lambda]} \\ &\leq ((\mathrm{Var}\{X\} + 1)\mathbf{P}\{X - n \geq \lambda^p - 1\})^{1/2} \end{aligned}$$

where we used the Cauchy-Schwarz inequality. Obviously, $\mathrm{Var}\{X\} = n$. Also, Chernoff's exponential bounding method (Chernoff, 1952; Bennett, 1962; Chow and Teicher, 1978) implies that for any $\varepsilon > 0$, we have

$$\mathbf{P}\{X \geq n + \varepsilon\} \leq \left(1 + \frac{\varepsilon}{n}\right)^n e^{-\varepsilon} \leq e^{-\varepsilon^2/(2n+\varepsilon)}.$$

Thus, we have

$$
\begin{aligned}
(1) &\le \sup_{\lambda \ge t} \sqrt{n+1}\, e^{-(\lambda^p-1)^2/(4n+2(\lambda^p-1))} \le \sup_{\lambda\ge t} \sqrt{\lambda+1}\, e^{-(\lambda^p-1)^2/4\lambda} \\
&= \sup_{\lambda\ge t} \sqrt{\lambda+1}\, e^{-\lambda^{2p-1}/4} e^{\lambda^{p-1}/2} e^{-1/(4\lambda)} \\
&\le e^{t^{p-1}/2} \sup_{\lambda \ge t} \sqrt{\lambda+1}\, e^{-\lambda^{2p-1}/4},
\end{aligned}
$$

which is a finite constant. Note that we used the fact that $t \ge 1$. A similar exponential left tail bound is valid for the gamma distribution:

$$\mathbf{P}\{X \le n - \varepsilon\} \le \left(1 - \frac{\varepsilon}{n}\right)^n e^{\varepsilon} \le e^{-\varepsilon^2/2n}.$$

Using this and the Cauchy-Schwarz inequality, we see that for $\lambda \ge t \ge 1$, the expected value in (2) does not exceed

$$
\begin{aligned}
(\mathbf{E}(n-\lambda^p-X)^2\mathbf{P}\{X<n-\lambda^p\})^{1/2} &\le ((\mathrm{Var}\{X\}+\lambda^{2p})e^{-\lambda^{2p}/2n})^{1/2} \\
&\le \sqrt{n+\lambda^{2p}}\, e^{-\lambda^{2p}/4n} \le \sqrt{\lambda+\lambda^{2p}}\, e^{-\lambda^{2p-1}/4},
\end{aligned}
$$

which is uniformly bounded over $\lambda \ge t$. Hence, (2) does not exceed a constant plus a constant times log log λ when $\lambda \ge t \ge 1$. This concludes the proof of the Theorem.

4. Fine-tuning Theorem 2

Under the conditions of Theorem 2, we see that the expected complexity due to the nonrecursive binomial and Poisson calls is $O(1)$. Thus, virtually the entire expected complexity is hidden in the if statement, the computation of n, the gamma generator, the case decision and the recursive call. As $\lambda \to \infty$, the expected time is asymptotic to a constant times log log λ/log$(1/p)$. From this, we are tempted to conclude that the best value is $p = 1/2$. However, for $p = 1/2$, the proof breaks down! To prove that Theorem would still work when $p = 1/2$, we let $\lambda_0, \lambda_1, \ldots$ be the (random) sequence of input parameters as we move down the recursive levels, where $\lambda_0 = \lambda$. Let n_i be the value of n corresponding to λ_i, and let X_i be gamma (n_i). Even though the λ_i's and n_i's are random, we have $\lambda_{i+1} \le 2\lambda_i^p$. Replace (2) by the following upper bound:

$$
\begin{aligned}
\sum_{i=0}^{\infty} \mathbf{E}\{(n_i-\lambda_i^p-X_i)I_{[X_i<n_i-\lambda_i^p]}I_{[\lambda_i>t]}\} &\le \sum_{i=0}^{\infty} \mathbf{E}\{2\lambda_i^p e^{-1/4\lambda_i^{2p-1}} I_{[\lambda_i>t]}\} \\
&= \mathbf{E}\left\{\sum_{i=0}^{\infty} 2\lambda_i^p e^{-1/4\lambda_i^{2p-1}} I_{[\lambda_i>t]}\right\}.
\end{aligned}
$$

However, when $t > 2^{1/(1-p)}$, the sequence λ_i decreases at a better than exponential rate to zero, so that the random variable behind the last expectation is in fact deterministically bounded. Hence the claim that for large λ, the expected time is about equal to the expected time needed to generate about log log λ gamma distributed random variables.

5. Optimizing p

From the previous section, we retain that p should ideally be close to 1/2. In fact, the conclusion of Theorem 2 remains valid if p satisfies one of the following inequalities,

$$p \geq \frac{1}{2}\left(1 + \frac{\log(2\log\lambda + c\log\log\lambda)}{\log\lambda}\right), \qquad \text{some } c \geq 2,$$

$$p \geq \frac{1}{2}\left(1 + \frac{\log(c\log\lambda)}{\log\lambda}\right), \qquad \text{some } c > 2,$$

and p is truncated to $[1/2, 3/4]$. Also, the constant t should be larger than 16. The verification of all this is left to the reader. Note that with the former choice of p with equality instead of inequality, $n - \lambda \sim -\lambda^p \sim -\sqrt{2\lambda\log\lambda}$.

6. Theorem 1 and its Proof

The difficulty with the original recursive generator is that the number of recursive calls is not deterministically bounded any longer. Nevertheless, the expected time spent on the binomial generator remains $O(1)$, as we established in the proof of Theorem 2. Thus, the overall expected complexity is bounded by $O(1)$ plus a constant time the expected number of recursions. But this is easily shown to be $O(\log\log\lambda)$. We proceed as follows. We take out the stopping rules in the algorithm and recurse forever, thus starting with parameter $\lambda_0 = \lambda$, and noting that λ_{i+1} is equal to $(\lambda_i - X_i)_+$, where X_i is gamma (n_i) and $n_i = \lceil \lambda_i - \lambda_i^p \rceil$. Note here that whenever $X_i \geq \lambda_i$, we have $\lambda_j \equiv 0$ for all $j > i$, which is fine since this indicates an exit via the binomial step in the algorithm. Note in particular that the number of recursive levels, N, is given by

$$N = \sum_{i=0}^{\infty} I_{[\lambda_i > t]}.$$

Let $\mathscr{F}_i$ be the σ-algebra generated by $\lambda_0, \lambda_1, \ldots, \lambda_i$, and let Z_i be the indicator of the event $X_i \leq n_i - \lambda_i^p$ (case B of the modified algorithm). The following two facts are known:

A. For $t2^{1/(1-p)} = d > 1$, we have

$$\sum_{i=0}^{\infty} I_{[Z_i=0]} I_{[\lambda_i > t]} \leq \frac{\log\log\lambda - \log\log d}{\log(1/p)}$$

by an argument as in the proof of Theorem 2. Here we used the fact that when $Z_i = 0$, then necessarily, $\lambda_{i+1} \leq 2\lambda_i^p$.

B. For all i,

$$\mathbf{E}\{I_{[Z_i=1]} I_{[\lambda_i > t]} | \mathscr{F}_i\} \leq \beta I_{[\lambda_i > t]},$$

where $\beta \in (0, 1)$ is a constant. It suffices to recall from the proof of Theorem 2 that (omitting subscripts for now) when $\lambda \ge t$

$$\mathbf{P}\{X < n - \lambda^p\} \le e^{-\lambda^{2p}/2n} \le e^{-\lambda^{2p-1}/2} \le e^{-t^{2p-1}/2} \overset{\text{def}}{=} \beta .$$

Note the following:

$$N = \sum_{i=0}^{\infty} I_{[\lambda_i > t]} = \frac{1}{1-\beta} \sum_{i=0}^{\infty} (1-\beta) I_{[\lambda_i > t]}$$

$$\le \frac{1}{1-\beta} \sum_{i=0}^{\infty} \mathbf{E}\{I_{[Z_i=0]} I_{[\lambda_i > t]} \mid \mathscr{F}_i\} .$$

Take expected values, and observe that

$$\mathbf{E}N \le \frac{1}{1-\beta} \sum_{i=0}^{\infty} \mathbf{E}\{I_{[Z_i=0]} I_{[\lambda_i > t]}\}$$

$$= \frac{1}{1-\beta} \mathbf{E}\left\{\sum_{i=0}^{\infty} I_{[Z_i=0]} I_{[\lambda_i > t]}\right\} \le \frac{1}{1-\beta} \times \frac{\log\log\lambda - \log\log d}{\log(1/p)} .$$

This concludes the proof of Theorem 1.

We finally note that the choices for p mentioned in the previous section do not affect the log-log behavior of the expected time of the recursive algorithm.

7. The Binomial Distribution

A related recursive algorithm for the binomial distribution was suggested by Relles (1972) and studied by Ahrens and Dieter (1974), Nekrutkin and Pokhodzei (1980) and Pokhodzei (1984). See e.g. section 10.4.5 of Devroye (1986) and exercise 7 on page 545, where the reader is asked to use arguments not unlike those used in this paper to prove that the expected time complexity of the recursive binomial method can be $O(\log\log n)$ provided that the parameters in the algorithm are fine-tuned.

8. Implementation

The computation of n, oddly enough, can dramatically slow down the performance, as it involves the computation of λ^p. The situation gets worse when p is picked as a function of $\log\lambda$. If we try to speed up matters by selecting $n = c\lambda$ with $c \in (0, 1)$, the number of recursions becomes $O(\log\lambda)$. To obtain $O(\log\log\lambda)$ expected complexity, we seem to be forced to introduce a costly nonlinearity in the definition of n. Our recommendation is to take p a fixed constant in the interval (0.55, 0.75). In view of the bounds given on the number of recursive levels, the threshold t should be picked close to $2^{1/(1-p)}$ when p is fixed beforehand. However, the ultimate choice of t and p will also depend heavily on the relative speeds of the different components of the program.

Acknowledgements

e author's research was sponsored by NSERC Grant A-3456 and FCAR Grant EQ-1678. Part of this earch carried out while writing the Division of Statistics, University of California at Davis.

References

1] Ahrens, J. H., Dieter, U.: "Computer methods for sampling from gamma, beta, Poisson and binomial distributions," *Computing*, **12**, 223–246, (1974).
2] Ahrens, J. H., Dieter, U.: "Sampling from binomial and Poisson distributions: a method with bounded computation times," *Computing*, **25**, 193–208, (1980).
3] Ahrens, J. H., Dieter, U.: "Computer generation of Poisson deviates from modified normal distributions," *ACM Transactions on Mathematical Software*, **8**, 163–179, (1982).
4] Ahrens, J. H., Dieter, U.: "A convenient sampling method with bounded computation times for Poisson distributions," in: *First International Conference on Statistical Computation, Izmir, Turkey*, pp. 4–17, 1987.
5] Ahrens, J. H., Kohrt, K. D., Dieter, U.: "Algorithm 599. Sampling from gamma and Poisson distributions," *ACM Transactions on Mathematical Software*, **9**, 255–257, (1983).
6] Bennett, G.: "Probability inequalities for the sum of independent random variables," *Journal of the American Statistical Association*, **57**, 33–45, (1962).
7] Chernoff, H.: "A measure of asymptotic efficiency of tests of a hypothesis based on the sum of observations," *Annals of Mathematical Statistics*, **23**, 493–507, (1952).
8] Chow, Y. S., Teicher, H.: *Probability theory*, New York: Springer, 1978.
9] Devroye, L.: "The computer generation of Poisson random variables," *Computing*, **26**, 197–207, (1981).
10] Devroye, L.: *Non-uniform random variate generation*, New York: Springer, 1986.
11] Devroye, L.: "A simple generator for discrete log-concave distributions," *Computing*, **39**, 87–91, (1987).
12] Fishman, G. S.: "Sampling from the Poisson distribution on a computer," *Computing*, **17**, 147–156, (1976).
13] Kachitvichyanukul, V.: "Computer generation of poisson, binomial, and hypergeometric random variates," Ph.D. Dissertation, School of Industrial Engineering, Purdue University, 1982.
14] Nekrutkin, V. V., Pokhodzei, B. B.: "A method of modelling a binomial distribution," *USSR Computational Mathematics and Mathematical Physics* **20(4)**, 248–253, (1980).
15] Pokhodzei, B. B.: "Beta- and gamma-methods of modelling binomial and Poisson distributions," *USSR Computational Mathematics and Mathematical Physics* **24(1)**, 114–118, (1984).
16] Relles, D. A.: "A simple method for generating binomial random variables when *n* is large," *Journal of the American Statistical Association* **67**, 612–613, (1972).
17] Schmeiser, B. W., Kachitvichyanukul, V.: "Poisson random variate generation," Research Memorandum 81-4, School of Industrial Engineering, Purdue University, West Lafayette, Indiana, 1981.
18] Stadlober, E.: "Sampling from Poisson, binomial and hypergeometric distributions: ratio of uniforms as a simple fast alternative," Habilitationsschrift, Institute of Statistics, Technical University of Graz, Austria, 1988.

Luc Devroye
School of Computer Science
McGill University
805 Sherbrooke Street West
Montreal, Canada H3A 2K6