中央研究院
資訊科學研究所
Institute of Information Science, Academia Sinica • Taipei, Taiwan, ROC

TR-IIS-10-002

Heterogeneous Subset Sampling

Meng-Tsung Tsai, Da-Wei Wang, Churn-Jung Liau, Tsan-sheng Hsu

January 28 2010 || Technical Report No. TR-IIS-10-002
http://www.iis.sinica.edu.tw/page/library/LIB/TechReport/tr2010/tr10.html

Heterogeneous Subset Sampling

Meng-Tsung Tsai, Da-Wei Wang, Churn-Jung Liau, Tsan-sheng Hsu

Institute of Information Science
Academia Sinica, Taipei 115, Taiwan
{mttsai,wdw,liaucj,tshsu}@iis.sinica.edu.tw

Abstract. In this paper, we consider the problem of heterogeneous subset sampling. Each element in a domain set has different probabilities of being included in a sample, which is a subset of the domain set. Drawing a sample from a domain set of size n takes $O(n)$ time if a Naive algorithm is employed. We propose a Hybrid algorithm, which requires $O(n)$ preprocessing time and $O(n)$ extra space. It draws a sample in $O(n\sqrt{p^*})$ time on average where p^* is $\min(p_\mu, 1-p_\mu)$ and p_μ denotes the mean of inclusion probabilities. In addition to the theoretical analysis, we evaluate the performance of the Hybrid algorithm via experiments and give an application for particle-based simulations on the spread of a disease.

1 Introduction

We begin by formulating the problem of heterogeneous subset sampling.

Problem HSS. Heterogeneous subset sampling involves drawing several samples from a domain set of size n. Without loss of generality, let the domain set be $D = \{1, \ldots, n\}$ in which each element i is associated with an inclusion probability p_i. For each drawn sample, which is a subset of the domain set, the probability of element i being included is equal to the given inclusion probability p_i. As it is necessary to draw several samples from the exemplar application in this paper, and potential new applications, we need to devise an efficient method to achieve the task. We assume the source used to generate variates from the standard uniform distribution $U(0, 1)$ is given.

The homogeneous case, in which all p_i are identical, is almost equivalent to generating variates from a binomial distribution. However, only a few works, such as [1], have considered the heterogeneous case and they focus on the size of the drawn sample rather than the elements included in it. Although this approach calculates the size of the drawn sample, it does not provide an efficient reduction method for generating the included elements. An intuitive way to achieve reduction is to sample with replacement, but the HSS problem is a case of sampling

without replacement. Trivially applying the method of sampling with replacement to the HSS problem may need infinite computation time in the worst case. With regard to the elements included in a sample, to the best of our knowledge, only the Naive algorithm introduced in Section 2.1 is known [2] [3]. The computational cost of solving the HSS problem with the Naive algorithm is $O(n)$ time for each sample. There is no preprocessing time.

In this paper, we consider the distribution of inclusion probabilities, which may be biased in many applications. For instance, in a particle-based simulation of the spread of a disease, the mean of the distribution is biased toward being small. We propose the Hybrid algorithm, which imposes a tighter bound on the computation time, as a trade-off for preprocessing time and extra space. Our algorithm requires $O(n)$ preprocessing time, $O(n)$ extra space, and $O(n\sqrt{p^*})$ time on average to draw each sample, where p^* is $\min(p_\mu, 1 - p_\mu)$ and p_μ denotes the mean of inclusion probabilities. The Hybrid algorithm is more efficient than the Naive algorithm when p_μ deviates from $1/2$ significantly. Particle-based simulation on the spread of a disease is a particular case that can cope with the requirement for speedup. We consider this case in detail in Section 3.

The remainder of this paper is organized as follows. In Section 2, we introduce the proposed Hybrid algorithm, provide the proof of its correctness, and discuss. In Section 3, we evaluate the performance of the Hybrid algorithm via experiments and introduce an application for particle-based simulation of the spread of a disease. Then, in Section 4, we summarize our conclusions.

2 Algorithms

In this section, three algorithms for solving the HSS problem are introduced. The first two algorithms are the building blocks for the third one.

2.1 Naive Algorithm

Algorithm 1 details the steps of a Naive algorithm for the HSS problem. To determine whether an element i should be included in a drawn sample, the algorithm compares its inclusion probability p_i with a variate generated from $U(0, 1)$. Because the size of the domain set D is n, the total cost is $O(n)$, which would be the lower bound of the HSS problem if it were necessary to make the decisions one by one.

2.2 Sieve Algorithm

The rationale for the proposed Sieve algorithm, Algorithm 2, is that instead of making a decision about each element in the domain set, we consider a smaller

Algorithm 1: Naive algorithm

input : a domain set $D = \{1, \ldots, n\}$ in which each element i is associated with an inclusion probability p_i

output: S, a drawn sample

$S \leftarrow \phi$

foreach $i \in D$ **do**
  $t \leftarrow U(0, 1)$
  if $t < p_i$ **then**
    $S \leftarrow S \cup \{i\}$
  end
end
return S

subset selected by a designed mechanism. Then, we only make decisions about the elements in the subset.

The designed mechanism is based on a simple idea. Specifically, for each element i in the domain set, we decouple the decision about it being included in a drawn sample into two decisions, A_i and B_i. For each pair of decisions A_i and B_i, we assign inclusion probabilities to it so that $Pr(A_i) \times Pr(B_i)$ is equal to p_i and $Pr(A_i)$ of all elements are identical. Hence, the original decision is included if and only if both decoupled decisions are included. Since it is meaningful to make decision B_i if and only if the outcome of A_i is included, the decision A_i acts like a sieve by removing unnecessary correspondent decision B_i. To guarantee the success of the decoupling, we let $Pr(A_i)$ be equal to p_{max}, the maximum of all p_i's.

One advantage of the decoupling procedure is that the outcome of a decision B_i is meaningful if and only if the correspondent decision A_i is included. Therefore, the outcomes of A_i's identify a subset of all B_i's to be determined. Another advantage is that all A_i's form a homogeneous case of the HSS problem which can be solved efficiently with two building blocks, namely, Procedure `SWOR(`k, D`)` and a binomial sampling from $B(|D|, p_{max})$. Procedure `SWOR(`k, X`)` randomly selects a subset of X from all X's subsets of size k. This is a classic result reported in [4]. $B(n, p)$ denotes the binomial distribution of n trials with success rate p.

We verify the equivalence of the Sieve and Naive algorithms by proving the following assumptions, which are obviously true for the Naive algorithm. First, the probability of an element being included in a drawn sample is equal to the given inclusion probability. Second, the events of the elements being included in a drawn sample are mutually independent. The proofs of the above assumptions for the Sieve algorithm are given in Lemmas 1 and 2 respectively.

Algorithm 2: Sieve algorithm

input : a domain set $D = \{1, \ldots, n\}$ in which each element i is associated with an inclusion probability p_i
output : a drawn sample S

$p_{max} \leftarrow \max_{i \in D} p_i$
$k \leftarrow B(|D|, p_{max})$
$R \leftarrow \texttt{SWOR}(k, D)$
$S \leftarrow \phi$
foreach $i \in R$ **do**
    $t \leftarrow U(0, 1)$
    if $t < p_i / p_{max}$ **then**
        $S \leftarrow S \cup \{i\}$
    end
end
return S

Procedure $\texttt{SWOR}(k, X)$

built-in: an array E of size n, where $E[i] = i$ for all $1 \leq i \leq n$
input : an integer k, where $k \leq |X| \leq n$ and a set $X = \{X_1, \ldots, X_{|X|}\}$
output : R, a randomly selected subset of X, whose size is k

$R \leftarrow \phi$
for $i = 1$ **to** k **do**
    $t \leftarrow U(0, 1)$
    $t \leftarrow \min(\lfloor t(n - i + 1) \rfloor, n - i)$
    exchange $E[i + t]$ with $E[i]$
    $R \leftarrow R \cup \{X_{E[i]}\}$
end
re-swap array E into the built-in state
return R

Lemma 1. *The probability p'_i of element i being included in a drawn sample is equal to p_i.*

Proof.

$$\begin{aligned}
p'_i &= \sum_{x=0}^{n} Pr((k \leftarrow B(n, p_{max})) = x) Pr(i \in \texttt{SWOR}(k, D)) Pr(B_i) \\
&= \sum_{x=0}^{n} \binom{n}{x} p_{max}^x (1 - p_{max})^{n-x} \left(\frac{x}{n}\right) \left(\frac{p_i}{p_{max}}\right) \\
&= \sum_{x=1}^{n} \binom{n-1}{x-1} p_{max}^{x-1} (1 - p_{max})^{n-x} p_i \\
&= (p_{max} + 1 - p_{max})^{n-1} p_i = p_i
\end{aligned}$$

□

Lemma 2. *$EV_1 \ldots EV_n$ are mutually independent, where EV_i denotes the event of element i being included in a drawn sample; that is,*

$$Pr\left(\bigcap_{i \in R} (EV_i)\right) = \prod_{i \in R} Pr(EV_i), \text{ for all } R \subset \{1, \ldots, n\}.$$

Proof.

$$\begin{aligned}
&Pr\left(\bigcap_{i \in R} (EV_i)\right) \\
&= \sum_{x=|R|}^{n} Pr((k \leftarrow B(n, p_{max})) = x) Pr(R \subset \texttt{SWOR}(k, D)) \prod_{i \in R} Pr(B_i) \\
&= \sum_{x=|R|}^{n} \binom{n}{x} p_{max}^x (1 - p_{max})^{n-x} \frac{\binom{n-|R|}{x-|R|}}{\binom{n}{x}} \prod_{i \in R} \frac{Pr(EV_i)}{p_{max}} \\
&= \sum_{x=|R|}^{n} \binom{n-|R|}{x-|R|} p_{max}^{x-|R|} (1 - p_{max})^{n-x} \prod_{i \in R} Pr(EV_i) \\
&= (p_{max} + 1 - p_{max})^{n-|R|} \prod_{i \in R} Pr(EV_i) \\
&= \prod_{i \in R} Pr(EV_i)
\end{aligned}$$

□

Procedure `SWOR(`k, X`)` requires $O(n)$ time to initialize the built-in array E and $O(n)$ space to accommodate it. Each invocation of Procedure `SWOR(`k, X`)` executes a loop of k iterations(Lines 2-7). Then, the Procedure re-swaps the array E into the built-in state, where there are at most $2k$ differences between them(Line 8). Therefore, the time complexity of each invocation of Procedure `SWOR(`k, X`)` is $O(k)$.

The Sieve algorithm requires $O(n)$ time to calculate p_{max} (Line 1). This step is ignored in subsequent invocations under the same problem setting. For each invocation, a binomial variate k is generated(Line 2) and Procedure `SWOR(`k, D`)` is invoked once(Line 3), after which, a loop of k iterations is executed. Let $\mathfrak{C}(B(n,p))$ denote the cost of generating a variate that follows a binomial distribution $B(n,p)$. Then, the following lemma can be derived.

Lemma 3. *Solving the HSS problem with the Sieve algorithm takes $O(n)$ preprocessing time and requires $O(n)$ extra space; and the time complexity of drawing each sample is*

$$O(\mathfrak{C}(B(n, p_{max})) + k). \tag{1}$$

Several works focus on binomial random variate generation, e.g., [5] [6] [7] [8] [9]. In [9], the authors report the results of comparing a number of algorithms in experiments. For example, they show Algorithm BG in [6] needs $O(k)$ computation time for each sampling if the generated variate is k and Algorithm BALIAS in [9] is $O(1)$ fast. In addition, they conclude that their proposed Algorithm BTPE [9] is the most effective when μ is moderate or large, where μ is the product of the parameters n and p in $B(n,p)$. For cases where μ is small, Algorithm BINV [9] dominates. Since there is no difference between the binomial variate generation algorithms in our theoretical analysis when its complexity can be bounded by $O(k)$, we let $\mathfrak{C}(B(n,p))$ be $O(k)$ without specifying which efficient binomial variate generator should be used. In [9], the authors conclude that the combination of Algorithms BINV and BTPE is the fastest experimentally. Hence, we adopt this combination to run the experiments in Section 3.

By replacing $\mathfrak{C}(B(n, p_{max}))$ in Eq.(1) with $O(k)$, the time complexity of the Sieve algorithm becomes $O(k)$. Because the expected value of k is $n \cdot p_{max}$, the Sieve algorithm draws a sample in $O(n \cdot p_{max})$ time on average. When $1 - p_{min} < p_{max}$, where p_{min} is the minimum of all p_i's, it is easier to draw the complement of a sample. Hence, it can be extended with a dual version of Sieve algorithm, as shown in Algorithm 3.

Lemma 4. *Applying Algorithm 3 to the case where $1 - p_{min} < p_{max}$, the Sieve algorithm can draw a sample in $O(n \cdot min(p_{max}, 1 - p_{min}))$ time on average.*

Remark 1. *The Sieve algorithm is much more effective than the Naive algorithm when p_{max} or $1 - p_{min}$ deviates from 1 significantly. Although the algorithms have*

the same asymptotic behavior when both p_{max} and $1-p_{min}$ are close to 1, the Sieve algorithm is less effective because the constant factor hidden in the asymptotic time complexity analysis of the Naive algorithm is smaller.

Algorithm 3: Dual Sieve algorithm

input : a domain set $D = \{1, \ldots, n\}$ in which element i is associated with an inclusion probability p_i

output : a drawn sample S

```
1  p_min ← min_{i∈D} p_i
2  k ← B(|D|, 1 − p_min)
3  R ← SWOR(k, D)
4  S̄ ← φ
5  foreach i ∈ R do
6      t ← U(0, 1)
7      if t < (1 − p_i)/(1 − p_min) then
8          S̄ ← S̄ ∪ {i}
9      end
10 end
11 return D − S̄
```

2.3 Hybrid Algorithm

To leverage the advantages of the Naive and Sieve algorithms, Algorithm 4, the Hybrid algorithm, divides the original domain set D into two disjoint domain sets X and Y. The division is made so that X contains all elements whose inclusion probability is less than a calculated threshold $p_{thres} = \sqrt{p_\mu}$, and Y contains the remaining elements. Then, the Hybrid algorithm applies the Sieve algorithm and the Naive algorithm to X and Y respectively. Therefore, the equivalence of the Hybrid algorithm and the Naive algorithm is verified by the equivalence of the Sieve algorithm and the Naive algorithm, as shown in the previous section. Let $sieve(D)$ and $naive(D)$ represent solving the HSS problem in the domain set D with the Sieve algorithm and the Naive algorithm.

The Hybrid algorithm requires $O(n)$ preprocessing time to execute Lines 1-4. This step is only performed once, i.e., during the first invocation. However, it is not necessary to allocate extra space for sets X and Y, which can be done by swapping the elements in the domain set D. To draw a sample, the Hybrid algorithm invokes the Sieve and Naive algorithms in Line 5. As a result, the following lemma is derived.

Algorithm 4: Hybrid algorithm

input : a domain set $D = \{1, \ldots, n\}$ in which element i is associated with an inclusion probability p_i

output : S, a drawn sample

$p_\mu \leftarrow \sum_{i \in D} p_i / n$
$p_{thres} \leftarrow \sqrt{p_\mu}$
$X \leftarrow \{i \in D \,|\, p_i \leq p_{thres}\}$
$Y \leftarrow D - X$
$S \leftarrow sieve(X) \cup naive(Y)$
return S

Lemma 5. *Solving the HSS problem with the Hybrid algorithm takes $O(n)$ preprocessing time and requires $O(n)$ extra space; the time complexity of drawing each sample is*

$$O(\mathfrak{C}(B(n, p_{thres})) + k + |Y|). \tag{2}$$

$|Y|$ is bounded by $O(n\sqrt{p_\mu})$ because $\sum_{i \in Y} p_i \leq \sum_{i \in D} p_i = n \cdot p_\mu$ and $\min_{i \in Y} p_i$ is greater than $\sqrt{p_\mu}$. Similar to the analysis of the Sieve algorithm, Eq. (2) can be rewritten as $O(n\sqrt{p_\mu})$ on average.

Lemma 6. *Similar to the idea of applying a dual Sieve algorithm, the case of $1/2 < p_\mu$ can be handled more effectively by a dual version of the Hybrid algorithm. Thus, it takes $O(n\sqrt{p^*})$ time on average to draw a sample with the Hybrid algorithm, where p^* is $\min(p_\mu, 1 - p_\mu)$.*

We compare the complexity of the proposed algorithms in Table 1,

ALGO\CPLX	Time(Preprocessing)	Time(Draw Each Sample)	Space
Naive algorithm	$O(1)$	$O(n)$	$O(1)$
Sieve algorithm	$O(n)$	$O(n \cdot \min(p_{max}, 1 - p_{min}))$ on average	$O(n)$
Hybrid algorithm	$O(n)$	$O(n\sqrt{p^*})$ on average	$O(n)$

Table 1: Complexity comparison.

3 Experiment Results

In this section, we compare the performance of the Naive algorithm and Hybrid algorithm via experiments. Then, we demonstrate how the Hybrid algorithm to

substantially improves the performance of a practical application. The implementation of the combined BTPE and BINV algorithms [9] was downloaded from *GSL*(GNU Scientific Library). All experiments were run on a workstation with Intel Xeon 3.2GHz processors.

3.1 Experiments

We conducted a number of experiments to assess the speedup and relative error of the Hybrid algorithm for solving the HSS problem. Let $EX_{alg}(n, p_\mu)$ be an experiment of drawing 100 samples with an algorithm alg, where $|D| = n$ and $\sum_{i \in D} p_i/n = p_\mu$. In addition, let $SP(n, p_\mu)$ denote the ratio of the running time of $EX_{naive}(n, p_\mu)$ to that of $EX_{hybrid}(n, p_\mu)$; note that the preprocessing time is included. Let $AS_{alg}(n, p_\mu)$ be the average size of the samples drawn in the experiment $EX_{alg}(n, p_\mu)$; and let $RE(n, p_\mu)$ denote the relative error $(AS_{hybrid}(n, p_\mu) - AS_{naive}(n, p_\mu))/AS_{naive}(n, p_\mu)$. The Figure 1(a) illustrates the speedup factor. Clearly, the speedup is substantial when p_μ deviates from 1/2 significantly. The difference between the curves of $SP(10^5, *)$ and $SP(10^6, *)$ is caused by the preprocessing time. The Figure 1(b) illustrates the relative error of the two algorithms. The distributions of the algorithms are similar because, in both scenarios, the relative error is bounded in the interval $[-0.4\%, 1\%]$ and converges to the interval $[-0.2\%, 0.2\%]$. Since $AS_{naive}(n, p_\mu)$ is not fixed, the relative error declines as p_μ increases.

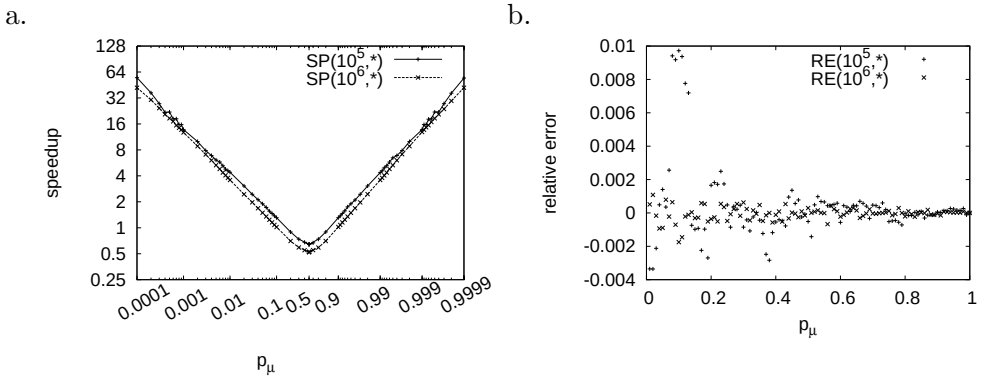

Fig. 1: Comparison of the Hybrid algorithm and the Naive algorithm.

3.2 An Practical Application of the Hybrid Algorithm

Particle-based simulation models are now being used in computational epidemiology [3] in addition to traditional SIR (Susceptible, Infector, and Remove/Recovered)

differential-equation-based approaches, such as [10] [11] [12] and [13]. There are two key reasons for this development [3]. The first is that the SIR model can best describe the dynamics of an epidemic when the number of infected persons is large, rather than in the initial or final stages of a disease outbreak. However, the initial stage is crucial because some intervention methods could be applied to prevent or slow down the transmission of the disease at this point. The second reason is that a particle-based simulation model provides more opportunities to fine tune the values of the features considered by the model.

Algorithm 5: Particle-based simulation model.

```
foreach time period T do
    foreach infector I do
        foreach susceptible individual S that had contact with I during T
        do
            if the infection between I and S takes place then
                change the susceptible status of S
            end
        end
    end
end
```

Algorithm 5 provides a high-level description of the particle-based simulation model of disease transmission. The most inner loop can be thought as a case of the HSS problem. The most time-consuming part is determining which possible infections have taken place. For each possible infection between an infector I and a susceptible individual S, the occurrence rate p may vary because of a number of factors, such as the nature of the disease, the intervention methods applied to it, and the closeness of I and S. In other words, p varies between different (I, S) pairs and also in the same pair when time period changes.

According to the work [3], we construct a simulation model based on Taiwan census data. As shown in Figure 2, the value of p is usually quite small in the simulation model of disease transmission, as a person seldom has close contact with a large number of family members or friends, compared to the number of casual contacts encountered daily. The figure also shows that setting $\sqrt{p_\mu}$ as p_{thres} is not appropriate. In practice, to maximize the speedup, p_{thres} can be adjusted dynamically based on the distribution of inclusion probabilities. Figure 3 compares the simulated results, attack rates, and the differences between the computational cost of two algorithms. Based on the figure, we conclude that the Hybrid algorithm improve the performance of particle-based disease simulation substantially, without changing the behavior of the model.

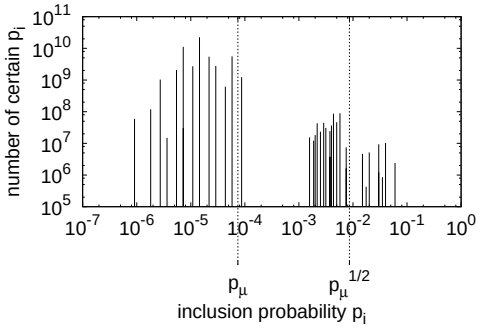

Fig. 2: Histogram of p_i in the disease transmission model.

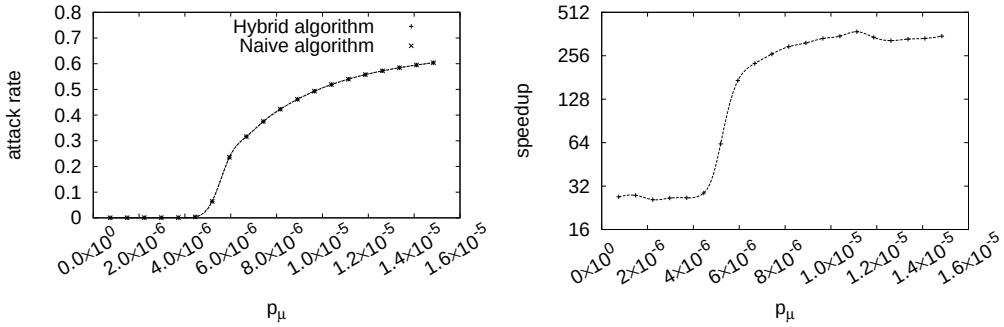

Fig. 3: Simulation results of the Hybrid and Naive algorithms.

4 Concluding Remarks

In this paper, we propose an algorithm, called the Hybrid algorithm, to solve the problem of heterogeneous subset sampling. We prove the correctness of the algorithm and show that time complexity is $O(n\sqrt{p^*})$. In addition, we evaluated the performance via experiments and demonstrated its efficacy by a practical application. The experiment results show the substantial speedup when p_μ deviates from 1/2 a great deal. Moreover, the exemplar application demonstrates that the Hybrid algorithm can be applied to simulate the particle-based simulation of the spread of a disease effectively and without changing the behavior of the model.

In addition to the application, we believe that the Hybrid algorithm can be applied to many other problems. In practice, $\sqrt{p^*}$ may not be an appropriate choice for p_{thres}. To maximize the performance in a future work, we will analyze the distribution of the inclusion probabilities to adjust p_{thres} dynamically.

References

1. R. J. Serfling. Some elementary results on poisson approximation in a sequence of bernoulli trials. *Society for Industrial and Applied Mathematics*, 20(3):567–579, 1978.
2. Carl-Erik Sarndal, Bengt Swensson, and Jan Wretman. Model assisted survey sampling (springer series in statistics). 2003.
3. Timothy C. Germann, Kai Kadau, Ira M. Longini, and Catherine A. Macken. Mitigation strategies for pandemic influenza in the united states. *PNAS*, 103(15):5935–5940, April 2006.
4. Donald E. Knuth. *The Art of Computer Programming, Volume II: Seminumerical Algorithms*. Addison-Wesley, 1969.
5. Daniel A. Relles. A simple algorithm for generating binomial random variables when n is large. *Journal of the American Statistical Association*, 67(339):612–613, 1972.
6. Devroye. L. Generating the maximum of independent identically distributed random variables. *Computers and Mathematics with Applications*, 6:305–315, 1980.
7. C.D. Kemp. A modal method for generating binomial variables. *Communications in Statistics Theory and Methods*, 15:805–813, 1986.
8. Luc Devroye. *Non-Uniform Random Variate Generation*. Springer, April 1986.
9. Voratas Kachitvichyanukul and Bruce W. Schmeiser. Binomial random variate generation. *Commun. ACM*, 31(2):216–222, 1988.
10. W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. *Proc. R. Soc. London Ser. A*, 115:700–721, 1927.
11. L. A. Evachev and I. M. Longini. A mathematical model for the global spread of influenza. *Math. Biosci.*, 75:3–22, 1985.
12. Roy M. Anderson and Robert M. May. *Infectious diseases of humans*. Oxford Press, 1991.
13. L. Hufnagel, D. Brockmann, and T. Geisel. Forecast and control of epidemics in a globalized world. *Proc. Natl. Acad. Sci. USA*, 101:15124–15129, 2004.