On the Average Runtime of an Open Source Binomial Random Variate Generation Algorithm

Vincent A. Cicirello

Computer Science
Stockton University
Galloway, NJ 08205 USA
https://www.cicirello.org/

Technical Report ALG-24-007

March 2024

On the Average Runtime of an Open Source Binomial Random Variate Generation Algorithm

Vincent A. Cicirello[1]

Computer Science
Stockton University
Galloway, NJ 08205 USA
https://www.cicirello.org/

Technical Report ALG-24-007

Abstract

The BTPE algorithm (Binomial, Triangle, Parallelogram, Exponential) of Kachitvichyanukul and Schmeiser is one of the faster and more widely utilized algorithms for generating binomial random variates. Cicirello's open source Java library, $\rho\mu$, includes an implementation of BTPE as well as a variety of other random number related utilities. In this report, I explore the average case runtime of the BTPE algorithm when generating random values from binomial distribution $B(n, p)$. Beginning with Kachitvichyanukul and Schmeiser's formula for the expected number of acceptance-rejection sampling iterations, I analyze the limit behavior as n approaches infinity, and show that the average runtime of BTPE converges to a constant. I instrument the open source Java implementation from the $\rho\mu$ library to experimentally validate the analysis.

Keywords: binomial; BTPE; inverse transform; Java; open source; random variate; runtime analysis

ACM Classes: F.2.1; G.3; G.4; I.1.2

MSC Classes: 68Q25; 68Q87; 68W40; 60-04

1 Introduction

In this report, I explore the average runtime behavior of binomial random variate generation within the open source Java library $\rho\mu$ [5]. The $\rho\mu$ library provides a variety of random number related enhancements beyond what is provided by the Java API itself. The core functionality of $\rho\mu$ is provided through a hierarchy of wrapper classes. This set of $\rho\mu$'s wrapper classes directly corresponds to the hierarchy of random number generator interfaces introduced in Java 17 [19]. In some cases, $\rho\mu$ overrides Java's random number generators with faster algorithms, such as for random integers subject to a bound. In other cases, $\rho\mu$ adds functionality, such as additional distributions, i.e., the binomial, Cauchy, among others [5]. The initial motivation of the $\rho\mu$ library was to provide a source of enhanced and more efficient randomness for other libraries, such as JavaPermutationTools [3] and Chips-n-Salsa [4].

[1] https://orcid.org/0000-0003-1072-8559

Among the randomness enhancements of the $\rho\mu$ library is support for generating binomial random variates, which are important to many applications [18, 17, 9, 22, 15, 2, 12, 21, 14]. There are many algorithms for generating binomial random variates [10, 11, 16, 1, 14, 13]. The $\rho\mu$ [5] library implements the BTPE Algorithm (Binomial, Triangle, Parallelogram, Exponential) [10], and falls back upon the inverse transform [10, 13] for cases that cannot be handled by BTPE.

The runtime of many algorithms for generating random values from binomial distribution $B(n, p)$ grows with some function of n and p. For example, the runtime of the inverse transform method is $O(np)$ [10, 13]. One technique many binomial random variate algorithms utilize is acceptance–rejection sampling [8], and the iterations of such sampling is often what leads to such runtimes. BTPE does use acceptance–rejection sampling, but seems to require relatively few iterations. At the time of introduction, Kachitvichyanukul and Schmeiser determined the expected number of iterations as a function of n and p [10] (see Section 2.1). However, the aim of my present report is to provide simpler insight into the average case runtime than is provided by Kachitvichyanukul and Schmeiser's formula.

In this report, I explore the limit of Kachitvichyanukul and Schmeiser's formula as n approaches infinity (Section 3), for two cases of p, including the minimum p supported by BTPE (i.e., $p = \frac{10}{n}$), which maximizes the expected number of rejection sampling iterations, and $p = 0.5$, which is when the expected number of rejection sampling iterations is minimized. Both cases lead to average runtimes that are constants in the limit for large n. In Section 4, I validate the findings experimentally using the $\rho\mu$ library, and utilizing my prior empirical study [6]. I discuss the findings and offer conclusions in Section 5, including the main result that the average runtime of BTPE is $\Theta(1)$. I now begin in Section 2 with some required background.

2 Preliminaries

2.1 Rejection Sampling Iterations of BTPE

The BTPE [10] algorithm separates the binomial distribution $B(n, p)$ into four parts, using triangular functions in the middle portions, and exponential functions in the tails, and it uses acceptance–rejection sampling [8]. For complete details of BTPE, which are beyond the scope of this paper, I refer the reader to the article that introduced it [10].

To generate a random value from binomial distribution $B(n, p)$, each acceptance–rejection iteration of BTPE generates two random values from $U(0, 1)$, i.e., uniformly distributed over the interval $[0.0, 1.0)$. When they introduced BTPE, Kachitvichyanukul and Schmeiser determined that the expected number of iterations, $E[I]$ of BTPE is [10]:

$$E[I] = p_4 \binom{n}{M} r^M (1 - r)^{n-M}, \tag{1}$$

and since each iteration generates two random uniform values from $U(0, 1)$, the expected number of uniform variates, $E[V]$, required by BTPE is thus:

$$E[V] = 2p_4 \binom{n}{M} r^M (1 - r)^{n-M}. \tag{2}$$

Equations 1 and 2 involve r, M, and p_4, which in turn depend upon a few others, all of which are functions of n and p. Kachitvichyanukul and Schmeiser [10] define these as follows:

$$r = \min(p, 1 - p), \tag{3}$$

$$q = 1 - r, \tag{4}$$

$$f_M = nr + r, \tag{5}$$

$$M = \lfloor f_M \rfloor, \tag{6}$$

$$p_1 = \lfloor 2.195\sqrt{nrq} - 4.6q \rfloor + 0.5, \tag{7}$$

$$x_M = M + 0.5, \tag{8}$$

$$x_L = x_M - p_1, \tag{9}$$

$$x_R = x_M + p_1, \tag{10}$$

$$c = 0.134 + \frac{20.5}{15.3 + M}, \tag{11}$$

$$a_L = \frac{f_M - x_L}{f_M - x_L r}, \tag{12}$$

$$\lambda_L = a_L \left(1 + \frac{a_L}{2}\right), \tag{13}$$

$$a_R = \frac{x_R - f_M}{x_R q}, \tag{14}$$

$$\lambda_R = a_R \left(1 + \frac{a_R}{2}\right), \tag{15}$$

$$p_2 = p_1(1 + 2c), \tag{16}$$

$$p_3 = p_2 + \frac{c}{\lambda_L}, \tag{17}$$

$$p_4 = p_3 + \frac{c}{\lambda_R}. \tag{18}$$

Also note that BTPE is only relevant when: $\min(p, 1 - p) \geq \frac{10}{n}$. Kachitvichyanukul and Schmeiser recommend using the inverse transform method [13] as a fallback for cases when BTPE is not relevant. This is what is done in the implementation in the $\rho\mu$ library [5].

The expected number of acceptance–rejection sampling iterations, expressed in Equation 2, is maximized when $\min(p, 1 - p)$ is minimal. Thus, the expected number of rejection sampling iterations is maximized for the cases $p = \frac{10}{n}$ and $p = \frac{n-10}{n}$. Due to symmetry, these two cases lead to the same expected number of rejection sampling iterations. The nearer p is to 0.5, the fewer iterations BTPE requires on average. Table 1 shows how the expected number of required uniform variates changes as n increases for these two cases of p. It includes the minimum n supported by BTPE (e.g., $n = 20$), and then considers n at increasing powers of two to show how the number of uniform variates used by BTPE to generate one binomial random variate varies as n increases rapidly.

2.2 Stirling's Formula

While computing the limiting behavior of BTPE, we will encounter some factorials. Stirling's formula [7] will be useful in the analysis. Stirling's formula is as follows:

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\alpha_n}, \tag{19}$$

Table 1: Expected number of uniform variates to generate one binomial random variate as predicted by Equation 2 as n increases.

	$E[V]$ from Equation 2	
n	$p \in \left\{\frac{10}{n}, \frac{n-10}{n}\right\}$	$p = 0.5$
20	3.996	3.996
2^5	3.837	3.599
2^6	3.792	3.337
2^7	3.790	2.985
2^8	3.793	2.632
2^9	3.796	2.420
2^{10}	3.797	2.317
2^{11}	3.798	2.307
2^{12}	3.798	2.276
2^{13}	3.799	2.300
2^{14}	3.799	2.299
2^{15}	3.799	2.300
2^{16}	3.799	2.302
2^{17}	3.799	2.310
2^{18}	3.799	2.310
2^{19}	3.799	2.313
2^{20}	3.799	2.314

where

$$\frac{1}{12n+1} < \alpha_n < \frac{1}{12n}. \tag{20}$$

A consequence of using Stirling's formula is that it will introduce terms involving e^{α_n}, either generally for n or in other cases for specific values of n. The following observations will be useful:

$$\lim_{n\to\infty} \alpha_n = 0\,, \tag{21}$$

$$\lim_{n\to\infty} \alpha_{n/2} = 0\,, \tag{22}$$

$$\lim_{n\to\infty} \alpha_{n-10} = 0\,, \tag{23}$$

$$\lim_{n\to\infty} e^{\alpha_n} = 1\,, \tag{24}$$

$$\lim_{n\to\infty} e^{\alpha_{n/2}} = 1\,, \tag{25}$$

$$\lim_{n\to\infty} e^{\alpha_{n-10}} = 1\,. \tag{26}$$

3 Limit Analysis

Let's now proceed to compute the limit:

$$\lim_{n\to\infty} E[V] = \lim_{n\to\infty} 2p_4 \binom{n}{M} r^M (1-r)^{n-M}, \tag{27}$$

for two cases, when $p = \frac{10}{n}$ and when $p = 0.5$ in Sections 3.1 and 3.2, respectively.

3.1 Case 1: Maximum Rejection Sampling Iterations

Consider the limit from Equation 27 when $p = \frac{10}{n}$, which is the lowest p (in terms of n) supported by BTPE. Begin by computing the values of the various constants from Section 2.1 in terms of n and $p = \frac{10}{n}$ as follows:

$$r = \min(p, 1-p) = \frac{10}{n}\,, \tag{28}$$

$$q = 1 - r = \frac{n-10}{n}\,, \tag{29}$$

$$f_M = nr + r = n\left(\frac{10}{n}\right) + \frac{10}{n} = 10 + \frac{10}{n}\,, \tag{30}$$

$$M = \lfloor f_M \rfloor = \left\lfloor 10 + \frac{10}{n} \right\rfloor = 10\,, \tag{31}$$

$$c = 0.134 + \frac{20.5}{15.3 + M} = 0.134 + \frac{20.5}{15.3 + 10} \approx 0.944\,, \tag{32}$$

$$x_M = M + 0.5 = 10 + 0.5 = 10.5\,. \tag{33}$$

From Equation (20), find the following, which we need later:

$$\frac{1}{121} < \alpha_{10} < \frac{1}{120}\,, \tag{34}$$

$$\alpha_{10} \approx 0.0083\,, \tag{35}$$

$$e^{\alpha_{10}} \approx 1.008\,. \tag{36}$$

We will also need the following limit:

$$\begin{aligned}
\lim_{n\to\infty} p_1 &= \lim_{n\to\infty} \lfloor 2.195\sqrt{nrq} - 4.6q \rfloor + 0.5 \\
&= \lim_{n\to\infty} \left\lfloor 2.195\sqrt{n\left(\frac{10}{n}\right)\left(\frac{n-10}{n}\right)} - 4.6\left(\frac{n-10}{n}\right) \right\rfloor + 0.5 \\
&= \lim_{n\to\infty} \left\lfloor 2.195\sqrt{10\left(\frac{n-10}{n}\right)} - 4.6\left(\frac{n-10}{n}\right) \right\rfloor + 0.5 \\
&= \left\lfloor 2.195\sqrt{10} - 4.6 \right\rfloor + 0.5 \\
&= 2.5\,.
\end{aligned} \tag{37}$$

We will also need the limits of λ_L and λ_R as n approaches infinity, as follows:

$$\begin{aligned}
\lim_{n\to\infty} \lambda_L &= \lim_{n\to\infty} a_L \left(1 + \frac{a_L}{2}\right) \\
&= \lim_{n\to\infty} \left(\frac{f_M - x_L}{f_M - x_L r}\right)\left(1 + \frac{f_M - x_L}{2(f_M - x_L r)}\right) \\
&= \lim_{n\to\infty} \left(\frac{f_M - x_M + p_1}{f_M - r(x_M - p_1)}\right)\left(1 + \frac{f_M - x_M + p_1}{2(f_M - r(x_M - p_1))}\right) \\
&= \lim_{n\to\infty} \left(\frac{f_M - 8}{f_M - 8r}\right)\left(1 + \frac{f_M - 8}{2(f_M - 8r)}\right) \\
&= \lim_{n\to\infty} \left(\frac{\frac{10n+10}{n} - 8}{\frac{10n+10}{n} - \frac{80}{n}}\right)\left(1 + \frac{\frac{10n+10}{n} - 8}{2(\frac{10n+10}{n} - \frac{80}{n})}\right) \\
&= \left(\frac{10-8}{10}\right)\left(1 + \frac{10-8}{20}\right) \\
&= 0.22
\end{aligned} \tag{38}$$

and

$$\begin{aligned}
\lim_{n\to\infty} \lambda_R &= \lim_{n\to\infty} a_R \left(1 + \frac{a_R}{2}\right) \\
&= \lim_{n\to\infty} \left(\frac{x_R - f_M}{x_R q}\right)\left(1 + \frac{x_R - f_M}{2x_R q}\right) \\
&= \lim_{n\to\infty} \left(\frac{x_M + p_1 - f_M}{q(x_M + p_1)}\right)\left(1 + \frac{x_M + p_1 - f_M}{2q(x_M + p_1)}\right) \\
&= \lim_{n\to\infty} \left(\frac{13 - f_M}{13q}\right)\left(1 + \frac{13 - f_M}{26q}\right) \\
&= \lim_{n\to\infty} \left(\frac{13 - \frac{10n+10}{n}}{\frac{13n-130}{n}}\right)\left(1 + \frac{13 - \frac{10n+10}{n}}{\frac{26n-260}{n}}\right) \\
&= \left(\frac{13-10}{13}\right)\left(1 + \frac{13-10}{26}\right) \\
&\approx 0.257.
\end{aligned} \tag{39}$$

We finally can compute the limit of Equation (27):

$$\begin{aligned}
\lim_{n\to\infty} E[V] &= \lim_{n\to\infty} 2p_4 \binom{n}{M} r^M (1-r)^{n-M} \\
&= \lim_{n\to\infty} 2p_4 \binom{n}{10} \left(\frac{10}{n}\right)^{10} \left(\frac{n-10}{n}\right)^{n-10} \\
&= \lim_{n\to\infty} 2p_4 \left(\frac{n!}{10!(n-10)!}\right) \left(\frac{10}{n}\right)^{10} \left(\frac{n-10}{n}\right)^{n-10} \\
&= \lim_{n\to\infty} 2p_4 \left(\frac{\sqrt{\frac{n}{(n-10)}}\, n^n\, e^{\alpha_n}}{\sqrt{20\pi}\, 10^{10}(n-10)^{n-10}\, e^{\alpha_{n-10}}\, e^{\alpha_{10}}}\right) \left(\frac{10}{n}\right)^{10} \left(\frac{n-10}{n}\right)^{n-10} \\
&= \lim_{n\to\infty} 2p_4 \left(\frac{\sqrt{\frac{n}{(n-10)}}}{\sqrt{20\pi}\, e^{\alpha_{10}}}\right) \approx \lim_{n\to\infty} 2p_4 \left(\frac{\sqrt{\frac{n}{(n-10)}}}{1.008\sqrt{20\pi}}\right) \approx \lim_{n\to\infty} 0.2503 p_4 \sqrt{\frac{n}{(n-10)}} \qquad (40)\\
&\approx \lim_{n\to\infty} 0.2503 p_4 \\
&\approx \lim_{n\to\infty} 0.2503 \left(p_1(1+2c) + \frac{c}{\lambda_L} + \frac{c}{\lambda_R}\right) \\
&\approx \lim_{n\to\infty} 0.2503 \left(2.888\, p_1 + \frac{0.944}{\lambda_L} + \frac{0.944}{\lambda_R}\right) \\
&\approx \lim_{n\to\infty} 0.2503 \left(7.22 + \frac{0.944}{\lambda_L} + \frac{0.944}{\lambda_R}\right) \\
&\approx 0.2503 \left(7.22 + \frac{0.944}{0.22} + \frac{0.944}{0.257}\right) \approx 0.2503\,(7.22 + 4.291 + 3.673) \\
&\approx 3.801.
\end{aligned}$$

Thus, the expected number of uniform random variates required by BTPE to generate one binomial random variate converges to approximately 3.801 as n grows large (approximately 1.90 rejection sampling iterations on average) for the case of minimum supported p. Observe in Table 1 that BTPE approaches this limit case rapidly.

3.2 Case 2: Central Case

Now consider the limit from Equation 27 for the case of $p = 0.5$.

Let us begin by computing some of the constants from Section 2.1 for this case. When computing M, assume that n is even, without loss of generality:

$$r = \min(p, 1-p) = \frac{1}{2}, \qquad (41)$$

$$q = 1 - r = \frac{1}{2}, \qquad (42)$$

$$f_M = nr + r = \frac{n+1}{2}, \qquad (43)$$

$$M = \lfloor f_M \rfloor = \frac{n}{2}\,. \qquad (44)$$

Now substitute these, and other Section 2.1 definitions, into Equation 27 and simplify:

$$
\begin{aligned}
\lim_{n\to\infty} E[V] &= \lim_{n\to\infty} 2p_4 \binom{n}{M} r^M (1-r)^{n-M} \\
&= \lim_{n\to\infty} 2p_4 \binom{n}{\frac{n}{2}} \left(\frac{1}{2}\right)^{\frac{n}{2}} \left(1-\frac{1}{2}\right)^{n-\frac{n}{2}} \\
&= \lim_{n\to\infty} 2p_4 \left(\frac{n!}{\left(\frac{n}{2}\right)!\left(\frac{n}{2}\right)!}\right)\left(\frac{1}{2}\right)^n \\
&= \lim_{n\to\infty} 2p_4 \left(\frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\alpha_n}}{\sqrt{\pi n}\left(\frac{n}{2e}\right)^{n/2} e^{\alpha_{n/2}}\sqrt{\pi n}\left(\frac{n}{2e}\right)^{n/2} e^{\alpha_{n/2}}}\right)\left(\frac{1}{2}\right)^n \\
&= \lim_{n\to\infty} 2p_4 \left(\frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\alpha_n}}{\pi n\left(\frac{n}{e}\right)^n\left(\frac{1}{2}\right)^n e^{2\alpha_{n/2}}}\right)\left(\frac{1}{2}\right)^n \\
&= \lim_{n\to\infty} 2p_4 \left(\frac{\sqrt{2/\pi}}{\sqrt{n}}\right)\left(\frac{e^{\alpha_n}}{e^{2\alpha_{n/2}}}\right) \\
&\approx \lim_{n\to\infty} 1.596\left(\frac{p_4}{\sqrt{n}}\right) \\
&\approx \lim_{n\to\infty} 1.596\left(\frac{1}{\sqrt{n}}\right)\left(p_1(1+2c)+\frac{c}{\lambda_L}+\frac{c}{\lambda_R}\right) \\
&\approx \lim_{n\to\infty} 1.596\left(\frac{1}{\sqrt{n}}\right)\left(p_1(1+2c)+\frac{2c}{\lambda_R}\right) \\
&\approx \lim_{n\to\infty} 1.596\left(\frac{p_1(1+2c)}{\sqrt{n}}+\frac{2c}{\sqrt{n}\lambda_R}\right).
\end{aligned}
\tag{45}
$$

The next-to-last step above utilizes the fact that $\lambda_R = \lambda_L$ when $p = 0.5$.

Before continuing to simplify Equation 45, compute the following limits. First:

$$
\lim_{n\to\infty} c = \lim_{n\to\infty} 0.134 + \frac{20.5}{15.3+M} = \lim_{n\to\infty} 0.134 + \frac{20.5}{15.3+\frac{n}{2}} = 0.134. \tag{46}
$$

Now, compute the limit $\lim_{n\to\infty}\frac{p_1}{\sqrt{n}}$. Since p_1 involves a floor, compute bounds on this limit. Although we will see that the upper and lower bounds are equal, and thus via the squeeze theorem is also equal to our target limit.

$$
\begin{aligned}
\lim_{n\to\infty}\frac{\lfloor 2.195\sqrt{nrq}-4.6q\rfloor+0.5}{\sqrt{n}} &\le \lim_{n\to\infty}\frac{p_1}{\sqrt{n}} \le \lim_{n\to\infty}\frac{\lfloor 2.195\sqrt{nrq}-4.6q\rfloor+0.5}{\sqrt{n}} \\
\lim_{n\to\infty}\frac{\left\lfloor 2.195\sqrt{n/4}-2.3\right\rfloor+0.5}{\sqrt{n}} &\le \lim_{n\to\infty}\frac{p_1}{\sqrt{n}} \le \lim_{n\to\infty}\frac{\left\lfloor 2.195\sqrt{n/4}-2.3\right\rfloor+0.5}{\sqrt{n}} \\
\lim_{n\to\infty}\frac{\lfloor 1.0975\sqrt{n}-2.3\rfloor+0.5}{\sqrt{n}} &\le \lim_{n\to\infty}\frac{p_1}{\sqrt{n}} \le \lim_{n\to\infty}\frac{\lfloor 1.0975\sqrt{n}-2.3\rfloor+0.5}{\sqrt{n}} \\
\lim_{n\to\infty}\frac{1.0975\sqrt{n}-2.8}{\sqrt{n}} &\le \lim_{n\to\infty}\frac{p_1}{\sqrt{n}} \le \lim_{n\to\infty}\frac{1.0975\sqrt{n}-1.8}{\sqrt{n}} \\
1.0975 &\le \lim_{n\to\infty}\frac{p_1}{\sqrt{n}} \le 1.0975
\end{aligned}
\tag{47}
$$

The next limit that we need is $\lim_{n\to\infty} \frac{2}{\sqrt{n}\lambda_R}$ as follows:

$$\begin{aligned}
\lim_{n\to\infty} \frac{2}{\sqrt{n}\lambda_R} &= \lim_{n\to\infty} \frac{2}{\sqrt{n}\left(a_R\left(1+\frac{a_R}{2}\right)\right)} \\
&= \lim_{n\to\infty} \frac{2x_R q}{\sqrt{n}(x_R - f_M)\left(1+\frac{x_R-f_M}{2x_R q}\right)} \\
&= \lim_{n\to\infty} \frac{x_R}{\sqrt{n}(x_R - f_M)\left(1+\frac{x_R-f_M}{x_R}\right)}
\end{aligned} \tag{48}$$

Now consider the following:

$$x_R - f_M = x_M + p_1 - \frac{n+1}{2} = M + \frac{1}{2} + p_1 - \frac{n+1}{2} = \frac{n}{2} + \frac{1}{2} + p_1 - \frac{n+1}{2} = p_1. \tag{49}$$

Now continue simplifying Equation 48, keeping Equation 47 in mind as well as that $p_1 \in \Theta(\sqrt{n})$ as follows:

$$\begin{aligned}
\lim_{n\to\infty} \frac{2}{\sqrt{n}\lambda_R} &= \lim_{n\to\infty} \frac{x_R}{\sqrt{n}(x_R - f_M)\left(1+\frac{x_R-f_M}{x_R}\right)} \\
&= \lim_{n\to\infty} \frac{x_M + p_1}{p_1\sqrt{n}\left(1+\frac{p_1}{x_M+p_1}\right)} \\
&= \lim_{n\to\infty} \frac{\frac{n+1}{2} + p_1}{p_1\sqrt{n}\left(1+\frac{p_1}{\frac{n+1}{2}+p_1}\right)} \\
&= \lim_{n\to\infty} \frac{n+1+2p_1}{2p_1\sqrt{n}\left(1+\frac{2p_1}{n+1+2p_1}\right)} \\
&= \lim_{n\to\infty} \frac{n+1+2p_1}{2p_1\sqrt{n}} \\
&= \lim_{n\to\infty} \frac{\sqrt{n} + \frac{1}{\sqrt{n}} + 2\frac{p_1}{\sqrt{n}}}{2p_1} \\
&= \lim_{n\to\infty} \frac{2.195 + \sqrt{n}}{2p_1} \\
&= \lim_{n\to\infty} \frac{\sqrt{n}}{2p_1} = \frac{1}{2(1.0975)} = \frac{1}{2.195} \approx 0.456\ .
\end{aligned} \tag{50}$$

Now continue simplifying Equation 45, utilizing Equations 46, 47, and 50 as follows:

$$\begin{aligned}
\lim_{n\to\infty} E[V] &\approx \lim_{n\to\infty} 1.596\left(\frac{p_1(1+2c)}{\sqrt{n}} + \frac{2c}{\sqrt{n}\lambda_R}\right) \\
&\approx \lim_{n\to\infty} 1.596\left(1.0975(1+2c) + 0.456c\right) \\
&\approx 1.596\left(1.0975 \cdot 1.268 + 0.456 \cdot 0.134\right) \\
&\approx 2.319.
\end{aligned} \tag{51}$$

Thus, when $p = 0.5$ the expected number of uniform random variates required by BTPE to generate one binomial random variate converges to approximately 2.319 as n grows large (approximately 1.16 rejection sampling iterations on average). Observe in Table 1, that this limit case is reached quickly.

Table 2: Average number of calls to $U(0,1)$ by $\rho\mu$'s BTPE implementation compared to the prediction of Equation 2. 95% confidence intervals are shown, as well as the p-values from t-tests.

	$B(n, \frac{10}{n})$			$B(n, 0.5)$		
n	predicted	mean	p-value	predicted	mean	p-value
2^5	3.837	3.829 ± 0.052	0.76	3.599	3.582 ± 0.046	0.46
2^6	3.792	3.786 ± 0.050	0.80	3.337	3.334 ± 0.042	0.89
2^7	3.790	3.810 ± 0.051	0.45	2.985	3.004 ± 0.034	0.27
2^8	3.793	3.765 ± 0.051	0.27	2.632	2.637 ± 0.025	0.66
2^9	3.796	3.796 ± 0.051	0.99	2.420	2.412 ± 0.019	0.41
2^{10}	3.797	3.818 ± 0.051	0.42	2.317	2.304 ± 0.016	0.11
2^{11}	3.798	3.875 ± 0.052	0.00	2.307	2.308 ± 0.017	0.85
2^{12}	3.798	3.808 ± 0.051	0.72	2.276	2.276 ± 0.015	0.98
2^{13}	3.799	3.809 ± 0.051	0.70	2.300	2.288 ± 0.016	0.13
2^{14}	3.799	3.798 ± 0.051	0.97	2.299	2.304 ± 0.017	0.55
2^{15}	3.799	3.808 ± 0.052	0.73	2.300	2.286 ± 0.016	0.08
2^{16}	3.799	3.828 ± 0.052	0.27	2.302	2.306 ± 0.017	0.65
2^{17}	3.799	3.825 ± 0.052	0.32	2.310	2.301 ± 0.016	0.29
2^{18}	3.799	3.818 ± 0.051	0.46	2.310	2.313 ± 0.017	0.73
2^{19}	3.799	3.792 ± 0.050	0.80	2.313	2.314 ± 0.017	0.91
2^{20}	3.799	3.807 ± 0.051	0.74	2.314	2.309 ± 0.017	0.58

4 Experimental Validation

In earlier work, I used the open source Java library $\rho\mu$ to empirically explore the behavior of my BTPE implementation [6]. I wrapped an instance of Java's SplittableRandom class, which implements the splitmix [20] pseudorandom number generator (PRNG). The purpose was to instrument the PRNG in order to count the number of uniform random variates, $U(0,1)$, generated (i.e., calls to the nextDouble() method) while generating a binomial random variate. I then supply this wrapped PRNG instance as the source of randomness for $\rho\mu$'s implementation of BTPE.

That prior paper [6] considered many combinations of n and p. Here I focus on the same values of $n \in \{2^5, 2^6, \ldots, 2^{20}\}$. But, I only focus on two specific cases of p, the two cases utilized earlier in the paper: $p = \frac{10}{n}$ from Section 3.1 and $p = 0.5$ from Section 3.2. For each binomial distribution $B(n,p)$, I generate 10,000 binomial random variates, and I compute the average number of uniform variates per binomial, with 95% confidence intervals. I compare the experimental mean to the prediction of the number of uniform variates from Equation 2, and test significance with a t-test. The experiments used OpenJDK 17 on a Windows 10 PC with a 3.4 GHz AMD A10-5700 CPU and 8 GB RAM. I used $\rho\mu$ 3.1.1. Both the source code for the experiments (`https://github.com/cicirello/btpe-iterations`), as well as for $\rho\mu$ (`https://github.com/cicirello/rho-mu`) is maintained on GitHub. The API documentation for the $\rho\mu$ library is available on the web: `https://rho-mu.cicirello.org/`.

Table 2 shows the results. The raw and processed data are available on GitHub (`https://github.com/cicirello/btpe-iterations`). The empirical results confirm the analytical prediction of Equation (2). Observe that there is no significant difference between the analytical prediction and the empirically computed means. T-test p-values are above 0.05 in almost all cases (well above in most cases). There is only a single case, $B(2^{11}, 10/2^{11})$, in Table 2 where a t-test p-value is less than 0.05, seemingly suggesting that the implementation behaves differently than theory predicts in this case. However, this case is explainable by random chance. There are 32 cases represented in Table 2, so this one case is approximately 3% of the cases tested. My more in depth empirical study explored a much larger number of cases experimentally, finding no significant difference between predicted and observed behavior in all but 4% of cases [6]. At significance level 0.05, we should expect false-positives in approximately 5% of cases.

5 Discussion and Conclusions

In this report, I analyzed the runtime behavior of the BTPE algorithm for binomial random variate generation, and observed the following concerning average performance:

- The average number of acceptance–rejection sampling iterations, $E[I]$, and average number of uniform random numbers generated while generating a binomial random variate, $E[V]$, are as follows:

$$1 \leq E[I] < 2, \tag{52}$$

$$2 \leq E[V] < 4. \tag{53}$$

  At least one iteration is necessary (the above lower bound), and the above upper bounds derive from the case of minimum supported $n = 20$ for $p = 0.5$, which is the case requiring the maximum number of rejection sampling iterations on average (Table 1).

- For large n and the case of $p = \frac{10}{n}$, which maximizes the expected number of rejection sampling iterations, we find that the expected number of uniform random numbers required to generate one binomial random variate is a constant:

$$\lim_{n \to \infty} E[V] \approx 3.801. \tag{54}$$

- For large n and the case of $p = 0.5$, the case that minimizes the expected number of rejection sampling iterations, we find that the expected number of uniform random numbers required to generate one binomial random variate is also a constant:

$$\lim_{n \to \infty} E[V] \approx 2.319. \tag{55}$$

Thus, the average runtime of BTPE is $\Theta(1)$, while the average runtime of many of the alternative algorithms for generating binomial random variates is worse than a constant. For example, the average runtime of the inverse transform method is $O(np)$ [10, 13].

In this report, I also experimentally validated my limit analysis of Kachitvichyanukul and Schmeiser's formula, using my implementation of BTPE in the $\rho\mu$ Java library [5]. This validates: (a) Kachitvichyanukul and Schmeiser's formula for the expected number of rejection sampling iterations, (b) my analysis of that formula, and (c) that $\rho\mu$'s implementation of BTPE behaves as theory predicts. One limitation of this study is that it focuses on average case performance. It is possible that BTPE may exhibit many more rejection sampling iterations on occasion. The maximum number of uniform random variates generated while generating a single binomial during the course of my previous experimentation [6] was 38 (19 rejection sampling iterations). Such instances were very rare in that study, which generated approximately 2.88 million binomial random variates.

References

[1] J. H. Ahrens and U. Dieter. Computer methods for sampling from gamma, beta, poisson and bionomial distributions. *Computing*, 12(3):223–246, 1974. doi:10.1007/BF02293108.

[2] Habiba Arshad, Muhammad Attique Khan, Muhammad Sharif, Mussarat Yasmin, and Muhammad Younus Javed. Multi-level features fusion and selection for human gait recognition: an optimized framework of bayesian model and binomial distribution. *International Journal of Machine Learning and Cybernetics*, 10(12):3601–3618, 2019. doi:10.1007/s13042-019-00947-0.

[3] Vincent A. Cicirello. JavaPermutationTools: A java library of permutation distance metrics. *Journal of Open Source Software*, 3(31):950, November 2018. doi:10.21105/joss.00950.

[4] Vincent A. Cicirello. Chips-n-salsa: A java library of customizable, hybridizable, iterative, parallel, stochastic, and self-adaptive local search algorithms. *Journal of Open Source Software*, 5(52):2448, August 2020. doi:10.21105/joss.02448.

[5] Vincent A. Cicirello. $\rho\mu$: A java library of randomization enhancements and other math utilities. *Journal of Open Source Software*, 7(76):4663, August 2022. doi:10.21105/joss.04663.

[6] Vincent A. Cicirello. An analysis of an open source binomial random variate generation algorithm. *Engineering Proceedings*, 56(1):86, October 2023. doi:10.3390/ASEC2023-15349.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 4th edition, 2022.

[8] Bernard D. Flury. Acceptance–rejection sampling made easy. *SIAM Review*, 32(3):474–476, 1990. doi:10.1137/1032082.

[9] Jaime Israel García-García, Nicolás Alonso Fernández Coronado, Elizabeth H. Arredondo, and Isaac Alejandro Imilpán Rivera. The binomial distribution: Historical origin and evolution of its problem situations. *Mathematics*, 10(15):2680, 2022. doi:10.3390/math10152680.

[10] Voratas Kachitvichyanukul and Bruce W. Schmeiser. Binomial random variate generation. *Communications of the ACM*, 31(2):216–222, 1988. doi:10.1145/42372.42381.

[11] Voratas Kachitvichyanukul and Bruce W. Schmeiser. Algorithm 678: Btpec: Sampling from the binomial distribution. *ACM Transactions on Mathematical Software*, 15(4):394–397, 1989. doi:10.1145/76909.76916.

[12] Manzoor Khan and Jake Olivier. Regression to the mean for the bivariate binomial distribution. *Statistics in Medicine*, 38(13):2391–2412, 2019. doi:10.1002/sim.8115.

[13] Donald E. Knuth. *The Art of Computer Programming, Volume 2, Seminumerical Algorithms*. Addison Wesley, 3rd edition, 1998.

[14] Michael E. Kuhl. History of random variate generation. In *Proceedings of the 2017 Winter Simulation Conference*, pages 231–242. IEEE Press, 2017. doi:10.1109/WSC.2017.8247791.

[15] Ashley I Naimi and Brian W Whitcomb. Estimating risk ratios and risk differences using regression. *American Journal of Epidemiology*, 189(6):508–510, 2020. doi:10.1093/aje/kwaa044.

[16] 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. doi:10.1080/01621459.1972.10481259.

[17] Sayed Qaiser Ali Shah, Farrukh Zeeshan Khan, and Muneer Ahmad. Mitigating tcp syn flooding based edos attack in cloud computing environment using binomial distribution in sdn. *Computer Communications*, 182:198–211, 2022. doi:10.1016/j.comcom.2021.11.008.

[18] Sunny Singh, Muskaan Chawla, Devendra Prasad, Divya Anand, Abdullah Alharbi, and Wael Alosaimi. An improved binomial distribution-based trust management algorithm for remote patient monitoring in wbans. *Sustainability*, 14(4):2141, 2022. doi:10.3390/su14042141.

[19] Guy Steele. Jep 356: Enhanced pseudo-random number generators. JEP 356, OpenJDK, 2017. URL `https://openjdk.org/jeps/356`.

[20] Guy L. Steele, Doug Lea, and Christine H. Flood. Fast splittable pseudorandom number generators. In *Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications*, pages 453–472, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2660193.2660195.

[21] Guantao Wang and Jingjing Pei. Macro risk: A versatile and universal strategy for measuring the overall safety of hazardous industrial installations in china. *International Journal of Environmental Research and Public Health*, 16(10):1680, 2019. doi:10.3390/ijerph16101680.

[22] Xiaoxian Zhang and Zhifen Lin. Hormesis-induced gap between the guidelines and reality in ecological risk assessment. *Chemosphere*, 243:125348, 2020. doi:10.1016/j.chemosphere.2019.125348.