Artifact Sub Math

tooflesswulf

February 6, 2022

1 Introduction

I hate this game.

Related: Generating functions. Higher order derivatives. Bell's polynomials. Beta functions. Fourier transforms. Young's inequality.

1.1 Contribution 1: Roll probability calculator

Suppose you want the artifact in Figure 1 to have more than 10% Energy Recharge and more than 15% CRIT DMG after upgrading to +20. I give an efficient cache-based method for calculating this probability. See section 2.

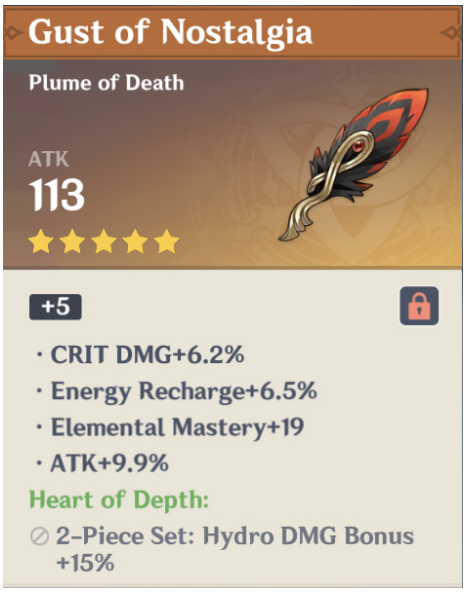

Figure 1: An example artifact

1.2 Contribution 2: Artifact replacement calculator

In Figure 2, I show my Beidou's current flower compared with a +0 artifact. Using a Beidou damage formula, I can compute the probability that upgrading the flower to +20 will improve Beidou's total damage or not.

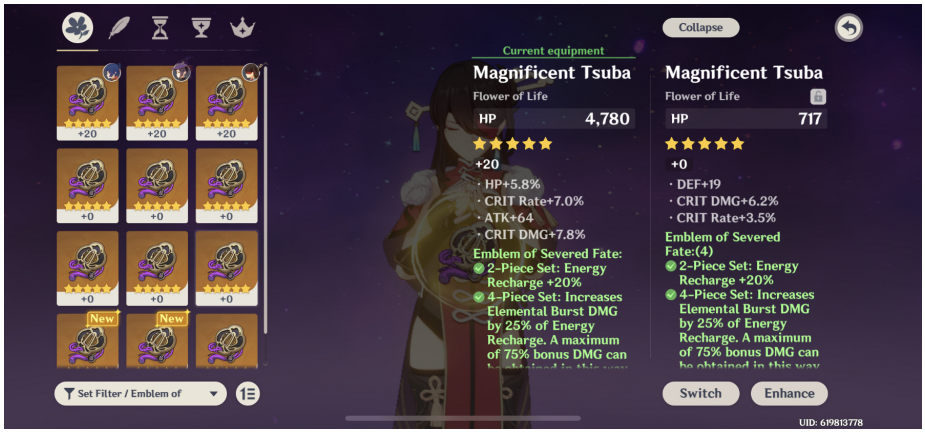

Figure 2: I would like to decide whether it's worth upgrading the +0 feather or not.

We can run the computation for every un-upgraded artifact in inventory; the 'best' artifact to upgrade is the one with highest probability of improving total damage. See Section 3.

1.3 Contribution 3: Artifact set completeness And DPS-to-resin

With a character, damage formula, and artifact set, we can evaluate the probability that a random +0 artifact, once upgraded to +20, will beat the current artifact set's total DPS. Then we can find the expected farming duration for any piece of the set to be improved. We also have a way to estimate the expected DPS-to-resin relation. See Section 4.

2 Roll probability calculation

Let A be an artifact. The artifact has 4 substats, with values A^i. It turns out the substat values can be written as an integer system, $A^i \in \alpha\mathbb{Z}$. The α depends on the substat; for example CritDmg has $\alpha = .78$.

Then the upgrade value of an artifact's substat can be written as a uniform choice $U = \{7\alpha, 8\alpha, 9\alpha, 10\alpha\}$[1]. When a substat is upgraded n times, its value follows the sum of n independent U distributions.

[1]Take for example CritDmg, which has upgrade values of $\{5.44, 6.22, 6.99, 7.77\}$. It turns out that $\{7, 8, 9, 10\} \times .777$ yields the same four values

2.1 Generating functions and Algebra

The generating function of U is:

$$\phi = G_U(z) = \mathbb{E}[z^U] = \frac{1}{4}\left(z^7 + z^8 + z^9 + z^{10}\right) \tag{1}$$

Then a substat upgraded n times follows the distribution:

$$G_{U+\cdots+U} = \phi^n$$

Generating functions of integer-value random variables follow a nice rule with derivatives:

$$\begin{aligned} P(A^i = a|n) &= \frac{1}{a!} G^{(a)}_{U+\cdots+U}(0) \\ &= \frac{1}{a!}\frac{\mathrm{d}^a}{\mathrm{d}z^a}\phi^n\big|_{z=0} \end{aligned} \tag{2}$$

2.1.1 Evaluating derivatives

To evaluate the derivative in (2), we proceed via an application of Faa di Bruno's formula (18). Let:

$$f(x) = x^n$$

$$\phi = \frac{1}{4}\left(z^7 + z^8 + z^9 + z^{10}\right)$$

Then with $\phi^n = f(\phi(z))$, we apply (18) to get:

$$\frac{\mathrm{d}^a}{\mathrm{d}z^a} f(\phi(z)) = \sum_{j=1}^{a} f^{(j)}(\phi) B_{a,j}\left(\phi', \cdots, \phi^{(a)}\right) \tag{3}$$

Where $B_{a,j}(\cdot)$ is the (a,j)th partial Bell polynomial (15). Due to the construction of ϕ, we have $\phi^{(j)}(0) = j!/4$ if $j \in \{7,8,9,10\}$ and zero otherwise. This gives a special form for the Bell polynomials, which we can evaluate using (16).

$$B_{a,j}\left(0, \cdots, \frac{7!}{4}, \frac{8!}{4}, \frac{9!}{4}, \frac{10!}{4}\right) = 4^{-j}\frac{a!}{j!}\binom{j}{a-7j}_3. \tag{4}$$

Then the jth derivative of $f(\phi(0))$ evaluates to

$$f^{(j)}(\phi(0)) = \frac{n!}{(n-j)!}[\phi(0)]^{n-j} \tag{5}$$

which is zero for all but $j = n$. Plugging equations (3-5) into (2), we have:

$$\begin{aligned} P(A^i = a|n) &= \frac{1}{a!}\sum_{j=1}^{a} f^{(j)}(\phi) B_{a,j}\left(\phi', \cdots, \phi^{(a)}\right) \\ &= 4^{-n}\binom{n}{a-7n}_3 \end{aligned}$$

2.1.2 Multivariate conditional independence

We are typically interested in more than one artifact substat at once. Fortunately, artifact substat values are conditionally independent given their number of rolls, because all the upgrade rolls are independent.

$$P(A^1 = a_1 \wedge A^2 = a_2 \wedge \cdots | n_1, n_2, \cdots) = P(a_1|n_1) \cdot P(a_2|n_2) \cdots$$

Therefore we can efficiently evaluate the roll probability with:

$$\begin{aligned} X &= [A^1 \geq a_1 \wedge A^2 \geq a_2 \wedge A^3 \geq a_3 \wedge A^4 \geq a_4] \\ P(X|n_1, \cdots, n_4) &= P(A^1 \geq a_1|n_1) \cdots P(A^4 \geq a_4|n_4) \\ P(X) &= \sum_{n_1, \cdots, n_4} P(X|n_1, \cdots, n_4) P(n_1, \cdots, n_4) \end{aligned}$$

Where σ is the multinomial distribution on 4 uniform random choices.

$$\sigma(n_1, \cdots, n_4) = P(n_1, \cdots, n_4) = P(n_1 + \cdots + n_4 = N)$$

2.2 Total artifact query & Caching

Let's define:

$$\begin{aligned} {}^b\xi(a, n) &= P(A^i = a|n + b) = 4^{-(n+b)} \binom{n+b}{a - 7(n+b)}_3 \\ {}^b\mu(a, n) &= P(A^i \geq a|n + b) = \sum_{a'=a}^{\infty} {}^b\xi(a', n) \end{aligned}$$

The parameter b toggles whether we consider the initial value as random or not ($b \in \{0, 1\}$). ${}^b\xi(a, n)$ gives the probability that a substat reaches the value a at the nth upgrade. ${}^b\mu(a, n)$ gives the probability the substat reaches or exceeds the value.

With the results in section 2.1.2, the roll probability query is a function that runs on a sum of σ and μ values; this sum contains at most (5 upgrades/4 categories $\rightarrow \binom{5+4-1}{4} = 70$) values.

The σ and μ functions also operate on integer arguments, making them very easy to cache (about 1kb of nontrivial values).

2.2.1 Caching σ

The function depends on the number of arguments. Four arguments is the simplest case.

Because we have $N = 5$ upgrades, if we sort the inputs there are only 5 possible inputs. The values are listed in Table 1. We can use the same approach to tabulate all possible values for $N \in [1, 5]$ as well as 1-3 input σ functions.

$\sigma(0,0,0,5)$	$1/1024$
$\sigma(0,0,1,4)$	$5/1024$
$\sigma(0,1,1,3)$	$20/256$
$\sigma(0,1,2,2)$	$30/1024$
$\sigma(1,1,1,2)$	$60/1024$
else	0

Table 1: Table of all σ_5 values for four inputs

Evaluate according to these formula:

$$\sigma(a) = \binom{N}{a, N-a} \left(\frac{1}{4}\right)^a \left(\frac{3}{4}\right)^{N-a}$$
$$\sigma(a,b) = \binom{N}{a, b, N-a-b} \left(\frac{1}{4}\right)^{a+b} \left(\frac{2}{4}\right)^{N-a-b}$$
$$\sigma(a,b,c) = \sigma(a,b,c,N-a-b-c)$$

In total, we can cache 89 values to capture the entire behavior of the function. See Appendix C.

2.2.2 Caching μ

Using $k = n+b$, we can cache ${}^b\mu(a,n) = \mu(a,k)$. The trivial values occur when $a \leq 7k$ or $a > 10k$. And any substat can be upgraded a maximum of 6 times, so we have:

$$\sum_{k=0}^{6} (10k - 7k) = 63$$

A total of 63 cached values to fully describe μ. See Appendix C.

2.3 Desired main and sub stats

I haven't found an efficient way to get the probability of combinations of substats yet. Instead, I found the probability of every 4-tuple of substats given some main stat, and saved them in an excel sheet as exact rational numbers. See Appendix D.

2.4 Summary

Given a query on substats, we can evaluate the probability a new artifact would fulfill those requirements once upgraded.

$$P(A^1 \geq a_1, \ldots, A^4 \geq a_4) = \sum_{n^*} \sigma(n^*) \prod_i {}^1\mu(a_i, n_i)$$

If we already have an artifact, we can evaluate the probability that we get a_i *increases* in the substats, with a reduced total N number of upgrades.

$$P(A^1 \geq a_1, \ldots, A^4 \geq a_4) = \sum_{n^*} \sigma(n^*) \prod_i {}^0\mu(a_i, n_i)$$

3 Artifact replacement

Let D be a damage formula, evaluating on a character C and 5 artifacts $A_1 - A_5$. We also have an un-upgraded artifact Z; let $Up(Z)$ be the distribution of random artifacts that arises from upgrading A^*. We can compare the initial damage d to the possible resultant damage d'.

$$d = D(C, A^{1-4}, A^5)$$
$$d' = D(C, A^{1-4}, A'), \quad A' \in Up(Z)$$

The goal of this section is to estimate $P(d' > d)$, the probability that upgrading Z will give us a net damage increase.

3.1 Damage formula approximation

We estimate the damage formula by using a linear approximation. With the exception of EnergyRecharge and ElementalMastery, the damage formula is actually linear (affine) in the substats. So for sufficiently small substat values, a linear approximation actually works quite well.

Technically, we should find the best uniform approximation of the damage formula. However, I don't know how to do that, so we will have to settle for the best least-squares approximation. An artifact has only 4 substats, so we can parameterize the damage formula in terms of those substats.

We integrate over region $\mathcal{R} = \{(a_1, a_2, a_3, a_4) \mid 7N \leq a_1 + a_2 + a_3 + a_4 \leq 10N\}$, with the damage formula re-parameterized $D(a_1, a_2, a_3, a_4)$. Our new damage formula becomes $D'(a_1, a_2, a_3, a_4)$.

$$D'(a_1, a_2, a_3, a_4) = \vec{w} \cdot [a_1, a_2, a_3, a_4] + w_0$$
$$R = \int_{\mathcal{R}} (D - D')^2 da_i$$
$$D' \leftarrow \arg\min_{D'} \left[\int_{\mathcal{R}} (D - D')^2 da_i \right]$$

The global minimum of the residual R occurs when the derivatives with $\vec{w}$ and w_0 are zero.

$$\begin{aligned} \nabla_{\vec{w}} R &= \nabla_{\vec{w}} \int_{\mathcal{R}} (D - D')^2 da_i \\ &= \int_{\mathcal{R}} \nabla_{\vec{w}} (D - D')^2 da_i \\ &= 2 \int_{\mathcal{R}} (D - D') \nabla_{\vec{w}} D' da_i = 0 \end{aligned} \tag{6}$$

Evaluating this integral is difficult, to say the least. However, if D can be expressed as a finite polynomial, then I do have a solution. See Appendix E.

$$k_1 + k_2 + k_3 + k_4 = K$$
$$\int_{\mathcal{R}} a_1^{k_1} \cdots a_4^{k_4} da_i = \left[(10N)^{4+K} - (7N)^{4+K}\right] \frac{k_1!k_2!k_3!k_4!}{(4+K)!} \tag{7}$$

3.1.1 Polynomial approximation of D

Equation (7) shows that if D is a polynomial form, we can very easily evaluate (6) to find a linear approximation of the damage formula. So how do we guarantee that D is a polynomial form?

My current method is to take a local 2nd order Taylor approximation. It's not ideal, but it seems to work well enough. (I'm open to suggestions though). One benefit from using the Taylor approximation is that the Taylor approximation can be built as a composition of linear terms.

Specifically let's take the 2nd order expansion. Let ∂_i be the partial derivative with respect to the ith artifact substat. In case I want to generalize to higher orders, I start using Einstein summation notation here.

$$D \approx D^* = D_0 + (\partial_i D)a^i + \frac{1}{2}(\partial_i \partial_j D)a^i a^j$$

3.1.2 Linear approximation of D

Now to find the least squares linear approximation by evaluating the integral in eq (6). We can first break the integral apart into:

$$\int_{\mathcal{R}} D' \nabla_{\vec{w}} D' da_i = \int_{\mathcal{R}} D^* \nabla_{\vec{w}} D' da_i$$

The left side we can evaluate as a linear equation in $\vec{w}$:

$$\nabla_{\vec{w}} D' = \begin{bmatrix} 1 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}$$
$$\gamma_k = \frac{(10N)^k - (7N)^k}{k!}$$
$$\int_{\mathcal{R}} D' \frac{\partial D'}{\partial w} d\vec{a} = \begin{bmatrix} \gamma_4 & \gamma_5 & \gamma_5 & \gamma_5 & \gamma_5 \\ \gamma_5 & 2\gamma_6 & \gamma_6 & \gamma_6 & \gamma_6 \\ \gamma_5 & \gamma_6 & 2\gamma_6 & \gamma_6 & \gamma_6 \\ \gamma_5 & \gamma_6 & \gamma_6 & 2\gamma_6 & \gamma_6 \\ \gamma_5 & \gamma_6 & \gamma_6 & \gamma_6 & 2\gamma_6 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}$$

And the right side is a (5×1) column vector. So we can find $\vec{w}$ by solving a linear equation.

3.2 The other integral

$$\int_{\mathcal{R}} D^* \nabla_{\vec{w}} D' da_i$$

This part's actually quite easy, simply follow Equation (7) for each index. I can't figure out how to write it concisely in a nice notation though. Any tensor calculus experts pls help :)

3.3 Computing $P(d' > d)$

Moving on to the main goal, we know that d' is approximated using D', a function linear in the substat values.

$$\begin{gathered} X = w_1 a_1 + \cdots + w_4 a_4 \\ x^* = d - w_0 \\ P(d' > d) \approx P(\vec{w} \cdot \vec{a} + w_0 > d) = P(X > x^*) \end{gathered} \tag{8}$$

3.3.1 Weighted sums of artifact substats

Following a similar approach to section 2.1.2, we can define:

$$P(d' > d) \approx P(X > x^*) = \sum_{n_1, \cdots, n_4} P(X > x^* | \cdots) P(n_1, \cdots) \tag{9}$$

Which is convenient because the a_i are all conditionally independent given their upgrade numbers $n_1 \cdots n_4$. This allows us to take their sum as a convolution. Conveniently, Fourier transforms make convolutions very easy to take.

With $f_{a|n}$ being the probability density function $p(a|n)$, $\mathcal{F}$ denoting Fourier transform, and $\mathcal{F}\{f\} = \widehat{f}$ being the Fourier transform of f:

$$\begin{gathered} f_{a|n=1} = \frac{1}{4}\left(\delta_7 + \delta_8 + \delta_9 + \delta_{10}\right) \\ \widehat{f_{a|n}} = \left[e^{8.5i\omega} \cos\frac{\omega}{2} \cos\omega\right]^n \\ \widehat{f_{\alpha a|n}} = \left[e^{8.5i\alpha\omega} \cos\frac{\alpha\omega}{2} \cos\alpha\omega\right]^n \end{gathered}$$

Then the density function of X can be determined as the following:

$$\begin{gathered} f_{X|n^*} = f_{w_1 a_1|n_1} \star \cdots \star f_{w_4 a_4|n_4} \\ f_{X|n^*} = \mathcal{F}^{-1}\left\{\widehat{f_{w_1 a_1|n_1}} \cdots \widehat{f_{w_4 a_4|n_4}}\right\} \\ F_{X|n^*} = \int_{-\infty}^{x} f(x)dx = \mathcal{F}^{-1}\left\{\frac{1}{i\omega}\widehat{f_{w_1 a_1|n_1}} \cdots \widehat{f_{w_4 a_4|n_4}} + \pi\delta(\omega)\right\} \\ P(X > x^*|n^*) = 1 - P(X \leq x^*|n^*) = 1 - F_{X|n^*}(x^*) \end{gathered}$$

And we have an exact solution for $P(X > x^*)$. If you're a psychopath, you should start looking for NFTs (numerical Fourier Transform/inverse-Fourier transform).

3.3.2 Approximate weighted sums of artifact substats

Because eq (8) approximates $P(d' > d)$ using $P(X > x^*)$ in the first place, let's simplify computation by approximating $P(X > x^*)$ via the central limit theorem.

By the central limit theorem, X should be a roughly Normal distribution. So if we know the mean and variance, we should have a decent idea of the distribution of X.

$$\mu_X = \sum_i w_i \frac{17}{2} n_i$$
$$\sigma_X^2 = \sum_i w_i^2 \frac{5}{4} n_i$$
$$X' \sim \mathcal{N}(\mu; \sigma)$$

Normal distributions are very nice. This gives us a very easy way to estimate expression (8).

$$\begin{aligned} P(X > x^* | n^*) &\approx P(X' > x^* | n^*) \\ &= \frac{1}{2} \text{Erfc}\left(\frac{x^* - \mu_X}{\sigma_X \sqrt{2}}\right) \end{aligned} \tag{10}$$

3.3.3 Estimation Error

See appendix F for my derivation of the error of our approximation. This is important to guarantee that our approximation is always valid, but is less important for those interested in implementation.

3.4 Normal approximation of d'

We can further simplify evaluating the query $P(d' > d)$ by applying a Gaussian approximation over the entire distribution of d', though this turns out to be quite a bad approximation sometimes. (See figure 3) Call our approximation η. Starting from eq (9), we have:

$$\begin{aligned} \mu_\eta = \mathbb{E}[d'] &= \sum_{n^*} \sigma(n^*) \mathbb{E}[X | n^*] \\ &= \sum_{n^*} \sigma(n^*) \frac{17}{2} \sum_i w_i (n_i + b) \\ &= \frac{17}{8} (N + 4b) \sum_i w_i \end{aligned}$$

$$\begin{aligned}
\mathbb{E}[(d')^2] &= \sum_{n^*} \sigma(n^*)\mathbb{E}[X^2|n^*] \\
&= \sum_{n^*} \sigma(n^*) \left[\frac{5}{4} \sum_i w_i^2 (n_i + b) + \left(\frac{17}{2} \sum_i w_i (n_i + b) \right)^2 \right] \\
&= ((N + 4b)^2 - N) \left(\frac{17}{8} \sum_i w_i \right)^2 + \left(\frac{5}{4} \frac{(N + 4b)}{4} + \frac{17^2 N}{4^2} \right) \sum_i w_i^2
\end{aligned}$$

$$\begin{aligned}
\sigma_\eta^2 &= \mathbb{E}[(d')^2] - (\mathbb{E}[d'])^2 \\
&= \left[\left(\frac{5}{4} \frac{(N + 4b)}{4} + \frac{17^2 N}{4^2} \right) \sum_i w_i^2 - N \left(\frac{17}{8} \sum_i w_i \right)^2 \right]
\end{aligned}$$

And now we can very quickly estimate:

$$P(d' > d) \approx P(\eta > d) = \frac{1}{2} \text{Erfc} \left(\frac{d - \mu_\eta}{\sigma_\eta \sqrt{2}} \right)$$

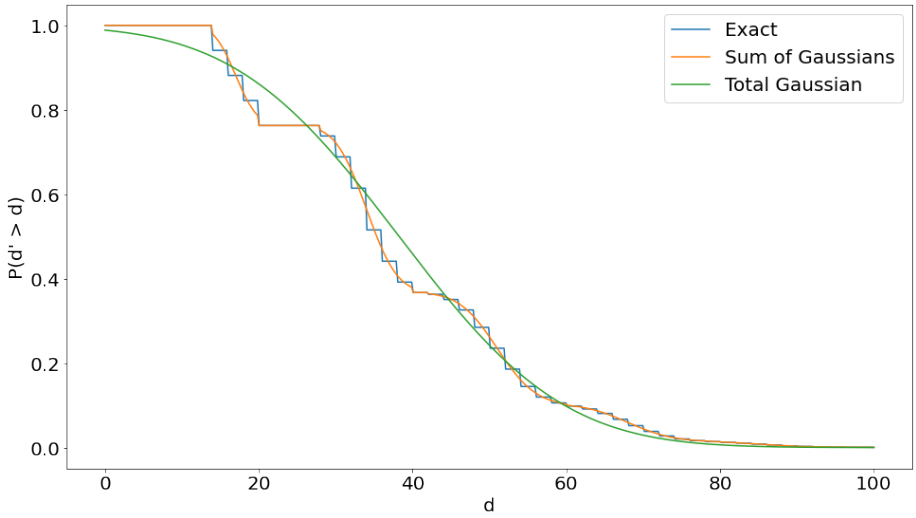

Figure 3: Comparison between exact and approximate methods. The fastest method, TotalGaussian, is a very poor approximation in this circumstance.

3.5 Summary

Given a damage formula D, approximate it:

$$D \to D' = w_0 + w_1 a_1 + w_2 a_2 + w_3 a_3 + w_4 a_4$$

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \int_{\mathcal{R}} da_i D \begin{bmatrix} \gamma_4 & \gamma_5 & \gamma_5 & \gamma_5 & \gamma_5 \\ \gamma_5 & 2\gamma_6 & \gamma_6 & \gamma_6 & \gamma_6 \\ \gamma_5 & \gamma_6 & 2\gamma_6 & \gamma_6 & \gamma_6 \\ \gamma_5 & \gamma_6 & \gamma_6 & 2\gamma_6 & \gamma_6 \\ \gamma_5 & \gamma_6 & \gamma_6 & \gamma_6 & 2\gamma_6 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}$$

$$x^* = d - w_0$$

$$\mu_X = \frac{17}{2} \sum_i w_i n_i$$

$$\sigma_X = \sqrt{\frac{5}{4} \sum_i w_i^2 n_i}$$

Then we can evaluate the probability an existing or new artifact would beat some damage threshold.

$$P(d' > d) \approx \sum_{n^*} P(X' > x^* | n^*) P(n_1, n_2, n_3, n_4)$$

$$P(X' > x^* | n^*) = \frac{1}{2} \text{Erfc}\left(\frac{x^* - \mu_X}{\sigma_X \sqrt{2}} \right)$$

If we care for speed more than accuracy,

4 Artifact Set Completeness

We can measure the quality of an artifact set by how difficult it would be to improve it in terms of damage or some other criteria. Using equation (10), we can estimate the probability that a newly farmed artifact with some given main stat and sub stats, once upgraded to +20, will improve the damage formula. Then we can use the results of section 2.3 to sum across all possible main- and sub-stat combinations to get the total probability that a newly farmed artifact will improve the set.

Let $\mathfrak{M}$ be the set of main stats, $\mathfrak{S}$ be the set of sub stats. The allowed substat combinations are $s_{1-4} \in \mathfrak{S}^4$.

$$p = \sum_{m \in \mathfrak{M}} \sum_{s_{1-4} \in \mathfrak{S}^4} P(s_{1-4}|m) P(m) P(d' > d | m, s_{1-4}) \tag{11}$$

Note that the linearization of the damage formula $P(d' > d | m, s_{1-4})$ changes for different main stats and sub stats, so it must be re-computed for every iteration of the sum.

4.1 Expected farming duration for Improvement

The probability calculated by eq (11) gives the probability a newly farmed artifact of some slot will improve the overall damage.

$$p_{tot} = \frac{1}{5}\left[p_{\text{Flower}} + p_{\text{Feather}} + p_{\text{Sands}} + p_{\text{Cup}} + p_{\text{Hat}}\right]$$

Then the expected number N of artifacts that must be farmed is:

$$\mathbb{E}[N] = \frac{1}{p_{tot}}$$

4.2 Expected increase in damage

Suppose you farm a new random artifact and upgrade it to +20. Let d' be a random variable denoting the new damage number. Now suppose you have a base damage number d that you're trying to beat, and keep farming artifacts until you have a random +20 artifact whose damage d' is greater than d. Call this damage value ρ (its an upside-down d). Mathematically, ρ is a truncated distribution of d' on the left at d.

$$\rho = d'|d' > d$$
$$F_\rho(x) = \frac{F_{d'}(x)}{1 - F_{d'}(d)}$$

Second, we can treat d' as a Mixture Model with weights w_i:

$$\begin{aligned} f_{d'} &= \sum_{\text{slot}} \sum_{m \in \mathfrak{M}} \sum_{s_{1-4} \in \mathfrak{S}} P(\text{slot}) P(s_{1-4}|m) P(m|\text{slot}) f_{d'|m,s} \\ &= \sum_i \lambda_i f_{d'|m_i,s_i} \end{aligned}$$

Then we can truncate this mixture distribution, and take the mean as follows. Let $X_i = d'|m_i, s_i$

$$\begin{aligned} p_i = P(d' > d|m_i, s_i) = P(X_i > d) = 1 - F_{X_i}(d) \\ \mu_i = \mathbb{E}[X_i|X_i > d] = \mu_{X_i} + \frac{f_{X_i}(d)}{1 - F_{X_i}(d)}\sigma_{X_i} \\ \mathbb{E}[\rho] = \mathbb{E}[d'|d' > d] = \frac{\sum_i \lambda_i p_i \mu_i}{\sum_i \lambda_i p_i} \end{aligned} \tag{12}$$

Thus we derive an expression for the expected increase in damage ρ.

4.3 Expected damage-to-resin equivalence

We can now evaluate the expected increase in damage per resin spent ratio (in expectation). Let $d' > d$ be the damage of new artifact, and let N be the

number of farming runs it took to find this artifact. Then we can estimate the damage-to-resin ratio of this particular artifact to be:

$$\omega = \frac{d'}{cN}$$

Where c is the artifact-to-resin ratio. Then we can take the expectation:

$$\mathbb{E}[\omega] = \mathbb{E}\left[\frac{\rho}{cN}\right] = \frac{1}{c}\mathbb{E}\left[\rho\right]\mathbb{E}\left[\frac{1}{N}\right]$$

We can split the expectation because d' and N are independent random variables. $\mathbb{E}[\rho]$ is already from eq (12). We can find the expected value of $1/N$ because we know N is an geometric distribution.

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{N}\right] &= \sum_{n=1}^{\infty} \frac{1}{n} p(1-p)^{n-1} \\
&= \frac{p}{1-p} \sum_{n=1}^{\infty} \frac{(1-p)^n}{n} && (13) \\
&= \frac{p}{1-p} \left(-\ln[1-(1-p)]\right) && (14) \\
&= -\frac{p \ln p}{1-p}
\end{aligned}$$

The jump from line (13) to (14) is simply the infinite Taylor expansion of the function $-\ln(1-x)$. This gives us all the tools we need to evaluate $\mathbb{E}[\omega]$, giving us the expected damage per resin spent ratio.

4.4 Summary

The formula lets you calculate ω, the resin-to-damage ratio.

$$\mathbb{E}[\omega] = -\frac{1}{c}\frac{p \ln p}{1-p}\frac{1}{5}\sum_{\text{slot}}\sum_{m \in \mathfrak{M}}\sum_{s_{1-4} \in \mathfrak{S}} P(s_{1-4}|m)P(m)\mathbb{E}[d'|d' > d, m, s]$$

$$\mathbb{E}[d'|d' > d, m, s] = \mu'_X + \frac{f_{X'}(d)}{1 - F_{X'}(d)}\sigma_{X'}$$

A Bell's polynomials

Bell's Polynomials (incomplete exponential Bell polynomials):

$$B_{n,k}(x_1, x_2, \cdots) = \sum \frac{n!}{j_1! j_2! \cdots j_{n-k+1}!} \left(\frac{x_1}{1!}\right)^{j_1} \cdots \left(\frac{x_{n-k+1}}{(n-k+1)!}\right)^{j_{n-k+1}} \tag{15}$$

Where the sum is taken over all sequences $j_1, \cdots, j_{n-k+1}$ such that:

$$j_1 + \cdots + j_{n-k+1} = k$$
$$j_1 + 2j_2 + \cdots + (n-k+1)j_{n-k+1} = n$$

A formula for describing and computing the ordinary multinomials (Belbachir, et al. 2008):

$$\sum_{a \geq 0} \binom{L}{a}_q t^a = (1 + t + \cdots + t^q)^L$$
$$\binom{L}{a}_q = \sum_{j=0}^{\lfloor a/(q+1) \rfloor} (-1)^j \binom{L}{j} \binom{a - j(q+1) + L - 1}{L-1}$$

A special identity for Bell's polynomials, also from Belbachir et al, 2008.

$$B_{n,L}(1!, 2!, \cdots, (q+1)!, 0, \cdots) = \frac{n!}{L!} \binom{L}{n-L}_q$$

Using the same proof technique as Belbachir et al, I have created a new version of above special identity for use in this document.

$$B_{n,L}(0, \cdots, a!, (a+1)!, \cdots, b!, 0, \cdots) = \frac{n!}{L!} \binom{L}{n-aL}_{b-a} \tag{16}$$

From a user on OEIS A008287 (Jean-François Alcover), we have the following special formula for a particular multinomial:

$$\binom{n}{k}_3 = \sum_{i=0}^{k/2} \binom{n}{i} \binom{n}{k-2i} \tag{17}$$

B Faa di Bruno formula

The Faa di Bruno formula describes the nth derivative of a composition of functions. Let $f, g : \mathbb{R} \to \mathbb{R}$ be infinitely differentiable functions. Let $h(x) = f(g(x))$. The first derivative of h is commonly known as the chain rule.

$$h'(x) = f'(g(x))g'(x)$$

For higher derivatives, this becomes much more obscure. It is related to the Bell polynomials.

$$h^{(n)} = \sum_{k=1}^{n} f^{(k)}(g(x)) \cdot B_{n,k}\left(g'(x), g''(x), \cdots, g^{(n-k+1)}(x)\right) \tag{18}$$

C Tables of μ and σ values

C.1 Table for μ

$\mu(a,n)$		$\mu(a,n)$	
(8, 1)	3/4	(36, 5)	1023/1024
(9, 1)	2/4	(37, 5)	1018/1024
(10, 1)	1/4	(38, 5)	1003/1024
		(39, 5)	968/1024
(15, 2)	15/16	(40, 5)	903/1024
(16, 2)	13/16	(41, 5)	802/1024
(17, 2)	10/16	(42, 5)	667/1024
(18, 2)	6/16	(43, 5)	512/1024
(19, 2)	3/16	(44, 5)	357/1024
(20, 2)	1/16	(45, 5)	222/1024
		(46, 5)	121/1024
(22, 3)	63/64	(47, 5)	56/1024
(23, 3)	60/64	(48, 5)	21/1024
(24, 3)	54/64	(49, 5)	6/1024
(25, 3)	44/64	(50, 5)	1/1024
(26, 3)	32/64		
(27, 3)	20/64	(43, 6)	4095/4096
(28, 3)	10/64	(44, 6)	4089/4096
(29, 3)	4/64	(45, 6)	4068/4096
(30, 3)	1/64	(46, 6)	4012/4096
		(47, 6)	3892/4096
(29, 4)	255/256	(48, 6)	3676/4096
(30, 4)	251/256	(49, 6)	3340/4096
(31, 4)	241/256	(50, 6)	2884/4096
(32, 4)	221/256	(51, 6)	2338/4096
(33, 4)	190/256	(52, 6)	1758/4096
(34, 4)	150/256	(53, 6)	1212/4096
(35, 4)	106/256	(54, 6)	756/4096
(36, 4)	66/256	(55, 6)	420/4096
(37, 4)	35/256	(56, 6)	204/4096
(38, 4)	15/256	(57, 6)	84/4096
(39, 4)	5/256	(58, 6)	28/4096
(40, 4)	1/256	(59, 6)	7/4096
		(60, 6)	1/4096

Table 2: All nontrivial values for μ

C.2 Table for σ

$\sigma(\cdot)$	N		$\sigma(\cdot)$	N		$\sigma(\cdot)$	N	
(0)	1	3/4	(0, 0)	1	2/4	(0, 0, 0, 1)	1	1/4
(1)	1	1/4	(0, 1)	1	1/4			
(0)	2	9/16	(0, 0)	2	4/16	(0, 0, 0, 2)	2	1/16
(1)	2	6/16	(0, 1)	2	4/16	(0, 0, 1, 1)	2	2/16
(2)	2	1/16	(0, 2)	2	1/16			
			(1, 1)	2	2/16			
(0)	3	27/64	(0, 0)	3	8/64	(0, 0, 0, 3)	3	1/64
(1)	3	27/64	(0, 1)	3	12/64	(0, 0, 1, 2)	3	3/64
(2)	3	9/64	(0, 2)	3	6/64	(0, 1, 1, 1)	3	6/64
(3)	3	1/64	(0, 3)	3	1/64			
			(1, 1)	3	12/64			
			(1, 2)	3	3/64			
(0)	4	81/256	(0, 0)	4	16/256	(0, 0, 0, 4)	4	1/256
(1)	4	108/256	(0, 1)	4	32/256	(0, 0, 1, 3)	4	4/256
(2)	4	54/256	(0, 2)	4	24/256	(0, 0, 2, 2)	4	6/256
(3)	4	12/256	(0, 3)	4	8/256	(0, 1, 1, 2)	4	12/256
(4)	4	1/256	(0, 4)	4	1/256	(1, 1, 1, 1)	4	24/256
			(1, 1)	4	48/256			
			(1, 2)	4	24/256			
			(1, 3)	4	4/256			
			(2, 2)	4	6/256			
(0)	5	243/1024	(0, 0)	5	32/1024	(0, 0, 0, 5)	5	1/1024
(1)	5	405/1024	(0, 1)	5	80/1024	(0, 0, 1, 4)	5	5/1024
(2)	5	270/1024	(0, 2)	5	80/1024	(0, 0, 2, 3)	5	10/1024
(3)	5	90/1024	(0, 3)	5	40/1024	(0, 1, 1, 3)	5	20/1024
(4)	5	15/1024	(0, 4)	5	10/1024	(0, 1, 2, 2)	5	30/1024
(5)	5	1/1024	(0, 5)	5	1/1024	(1, 1, 1, 2)	5	60/1024
			(1, 1)	5	160/1024			
			(1, 2)	5	120/1024			
			(1, 3)	5	40/1024			
			(1, 4)	5	5/1024			
			(2, 2)	5	60/1024			
			(2, 3)	5	10/1024			

Table 3: Table of nontrivial values of σ

D Program to compute substat probabilities

A Python program.

```
from fractions import Fraction

def prob(distr, subs, depth=4):
    if len(subs) == 0:
        return 1
    elif depth == 0:
        return 0

    denom = sum(distr.values())
    pps = []
    for k, v in distr.items():
        dnext = distr.copy()
        del dnext[k]
        p = Fraction(v, denom)
        # p = v/denom
        nxtsub = subs.copy()
        if k in subs:
            nxtsub.remove(k)
        pz = prob (dnext, nxtsub, depth-1)
        pps.append(p * pz)

    return sum(pps)
```

Usage (in python script)

```
dz = {'HP': 6, 'DEF': 6, 'ATK': 6,
      'HP%': 4, 'DEF%': 4, 'ATK%': 4,
      'ER': 4, 'EM': 4,
      'CR': 3, 'CD': 3}
print(prob(dz, {'CR', 'HP'}))
# 887203013 4703/7055078574544 0 = 0.1258
```

E Derivation of integral over the Standard Orthogonal Simplex

Define the standard orthogonal simplex:

$$\Delta_c^n = \left\{ (x_1, \cdots, x_n) \in \mathbb{R}^n \middle| \sum_{i=1}^{n} x_i \leq 1 \text{ and } x_i \geq 0 \right\}$$

Then our region of integration can be defined as:

$$\mathcal{R} = \left\{ (x_1, \cdots, x_4) \in \mathbb{R}^4 \middle| 7N \leq \sum_i x_i \leq 10N \right\} = 10N\Delta_c^n \setminus 7N\Delta_c^n$$

So we can write:

$$\int_{\mathcal{R}} f dx_i = \int_{10N\Delta_c^4} f dx_i - \int_{7N\Delta_c^4} f dx_i$$

Let f be some polynomial function in x_i. Then the following formula holds:

$$\int_{\lambda\Delta_c^n} x_1^{k_1} \cdots x_n^{k_n} dx_i = \frac{\lambda^{K+n}}{(K+n)!} k_1! \cdots k_n!$$

where $K = k_1 + \cdots + k_n$.

Proof. By induction.

Base case $n = 1$. We have the trivial integral

$$\begin{aligned} \int_0^\lambda x^k dx &= \frac{1}{k+1} x^{k+1} \Big|_{x=0}^{\lambda} \\ &= \frac{\lambda^{k+1}}{k+1} = \frac{\lambda^{k+1}}{(k+1)!} k! \end{aligned}$$

Induction step. Suppose the formula holds true for all $n' < n$. Then:

$$\begin{aligned} \int_{\lambda\Delta_c^n} x_1^{k_1} \cdots x_n^{k_n} dx_i &= \int_0^\lambda \int_0^{\lambda - x_1} \cdots dx_2 dx_1 \\ &= \int_0^\lambda x_1^{k_1} \int_{(\lambda - x_1)\Delta_c^{n-1}} \cdots dx_1 \\ &= \int_0^\lambda x_1^{k_1} \frac{(\lambda - x_1)^{K - k_1 + n - 1}}{(K - k_1 + n - 1)!} k_2! \cdots k_n! dx_1 \\ &= \frac{k_2! \cdots k_n!}{(K - k_1 + n - 1)!} \int_0^\lambda x^{k_1} (\lambda - x_1)^{K - k_1 + n - 1} dx_1 \end{aligned}$$

Perform a simple u-sub with $x_1 = \lambda u$ on the integral:

$$\begin{aligned}\int_0^\lambda x^{k_1}(\lambda - x_1)^{K-k_1+n-1}dx_1 &= \int_0^1 \lambda^{k_1}u^{k_1}[\lambda(1-u)]^{K-k_1+n-1}\lambda du \\ &= \lambda^{K+n}\int_0^1 u^{k_1}(1-u)^{K-k_1+n-1}du\end{aligned}$$

Now this integral is easily recognizable as a Beta function.

$$B(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1}dt = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$$
$$\int_0^1 t^x(1-t)^y dt = B(x-1, y-1) = \frac{x!y!}{(x+y+1)!}$$

Therefore we can evaluate:

$$\int_0^\lambda x^{k_1}(\lambda - x_1)^{K-k_1+n-1}dx_1$$

$$\begin{aligned}\int_{\lambda\Delta_c^n} x_1^{k_1}\cdots x_n^{k_n}dx_i &= \frac{k_2!\cdots k_n!}{(K-k_1+n-1)!}\lambda^{K+n}\int_0^1 u^{k_1}(1-u)^{K-k_1+n-1}du \\ &= \frac{k_2!\cdots k_n!}{(K-k_1+n-1)!}\lambda^{K+n}\frac{(K-k_1+n-1)!k_1!}{(K+n)!} \\ &= \frac{\lambda^{K+n}}{(K+n)!}k_1!k_2!\cdots k_n!\end{aligned}$$

□

F Gaussian approximation Error Analysis

Let f and F be the true probability distribution and cdf, respectively. Let g and G be that Gaussian approximations of f and F. With $f_i = k_i a_i$, we have $f_X = f_1 \star f_2 \star f_3 \star f_4$ and $F_X = F_1 \star f_2 \star \cdots = f_1 \star F_2 \star \cdots$.

Then the error of our approximation is:

$$||F - G||_\infty = \sup_x |F(x) - G(x)|$$

We know that due to the integer nature of sub-stat upgrades, F must be a piece-wise constant function. In other words, $F(x) = F_k$ is constant for all x in the interval $x \in [q_k, q_{k+1})$.

Then we can use Taylor remainder theorem to find the error in terms of the derivative, treating F_k as the 0th order approximation, and using the first

derivative as the remainder.

$$\begin{aligned}
||F - G||_\infty &= \sup_x |F(x) - G(x)| \\
&= \max_k \sup_{x \in [q_k, q_{k+1})} |F_k - G(x)| \\
&\leq \max_k \sup_{x \in [q_k, q_{k+1})} \sup_\xi G'(\xi)(x - x_k) \qquad (19)
\end{aligned}$$

We can evaluate this in parts. First, the maximum of the derivative of a Gaussian cumulative distribution function.

$$\begin{aligned}
\sup_\xi G'(\xi) &= \sup_\xi g(\xi) \\
&= \frac{1}{\sigma^* \sqrt{2\pi}}
\end{aligned}$$

We know $G' = g$, which is just a Gaussian density function. The maximum is known and easy to compute.

As for estimating an upper bound for $(x - x_k)$, I will argue that it is bounded by $\max_i w_i/2$ in part F.1. Anyhow, we can then say:

$$\begin{aligned}
||F - G||_\infty &\leq \max_k \sup_{x \in [q_k, q_{k+1})} \sup_\xi G'(\xi)(x - x_k) \\
&\leq \max_k \sup_{x \in [q_k, q_{k+1})} \frac{1}{\sigma^* \sqrt{2\pi}} \max_i \frac{w_i}{2} \\
&= \max_i \frac{w_i}{2\sigma^* \sqrt{2\pi}}
\end{aligned}$$

F.1 Upper bound for $(x - x_k)$

For simplicity, consider $f = f_1 \star f_*$ and $g = g_1 \star g_*$. Then we can actually decompose $F - G$.

$$\begin{aligned}
F - G &= F_1 \star f_* - G_1 \star g_* \\
&= F_1 \star f_* - G_1 \star f_* + G_1 \star f_* - G_1 \star g_* \\
&= (F_1 - G_1) \star f_* + g_1 \star (F_* - G_*)
\end{aligned}$$

Then using the triangle inequality, we have:

$$||F - G||_\infty \leq ||(F_1 - G_1) \star f_*||_\infty + ||g_1 \star (F_* - G_*)||_\infty \qquad (20)$$

$$||(F_1 - G_1) \star f_*||_\infty \leq ||F - G||_\infty + ||g_1 \star (F_* - G_*)||_\infty \qquad (21)$$

Now for some handwaving. We know $(F - G)$ and $(F_* - G_*)$ look vaguely sawtooth-like, because we are subtracting a continuous function G from a piecewise constant function F.

Let w_1 be the largest w associated with any f_i or g_i. Then because g_1 is continuous and is wider than the pattern of $(F_* - G_*)$, convolving $g_1 \star (F_* - G_*)$ will result in a nearly zero function. In other words, assume

$$||g_1 \star (F_* - G_*)||_\infty \ll 1$$

Using this assumption in equations (20) and (21, we get

$$||F - G||_\infty \approx ||(F_1 - G_1) \star f_*||_\infty$$

Next, I cast Young's inequality for convolutions.

$$\begin{aligned} ||F - G||_\infty &\approx ||(F_1 - G_1) \star f_*||_\infty \\ &\leq ||F_1 - G_1||_\infty ||f_*||_1 \end{aligned}$$

Because f_* is always positive, the L_1 norm is just the integral, which is unity. And so we have that the maximum discrepancy is approximately upper bounded by $||F_1 - G_1||_\infty$.

We can compute the upper bound with the exact same Taylor remainder logic as in eq (19). Only this time, because F_1 consists of constant values with uniform spacing, it's much easier to claim that the max of $(x - x_k)$ is $w_1/2$.

So by analogue, the max of $(x - x_k)$ in eq (19) should be $w_1/2$, where w_1 is the largest such w_i.

F.2 Mixed approximation

If for whatever reason the error estimate is too large, we can try to reduce the error by mixing exact and approximate portions.

Suppose we have the approximation with substat weights

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \end{bmatrix} = \begin{bmatrix} 1 & 20 & 0.7 & 0 \end{bmatrix}$$

What we can do is approximate a Normal distribution using the weights (w_1, w_3, w_4) to reduce the upper bound of $(x - x_k)$ to $w_1/2$. Then we evaluate F_2 exactly, and numerically convolve f_2 with G_{134} to perform the mixed convolution.

This operation is computable because f_2 is a discrete distribution, so rather than using an integral, we can just sum over the possible values. However, the runtime suffers depending on how big the sum is.