Estimated TBP for Improvement

IceDynamix

Abstract

A common question for the player is whether it's worth grinding for better pieces considering the current build. A possible metric for that would be the leaderboard and the relative position of the player, but that incurs issues such as going for a non-standard optimization target or other biases from the community such as recency bias (new character does not have many builds, prefarming). A more absolute approach would be to calculate the estimated amount of trailblaze power (TBP) in order to obtain a relic that improves the build.

Contents

1 Prerequisites

We need to determine the approach, the target and the desired way of displaying results in order to start the calculations.

1.1 Brute Force vs. Analytical

An easy way to get a rough estimate for investment is to run a sufficient number of simulations and thus brute force the answer. However, with a bit of clever tinkering, it should be possible to derive an analytical calculation that computes the chances based on estimated main stat probabilities and substat probabilities collected by the community, for example [1]. **This is the method that this document will be focusing on.**

1.2 Incremental Build vs. Full Recalculation

The easiest way to find a way to improve on is to consider finding a relic that replaces one of the relics in the current build. However, this is a short-term goal that is likely to lead to a local maximum rather than a global maximum. Replacing a relic with one of a different set bonus would likely lower the optimization target and would be discarded short-term, but switching to a different, provably more optimal set altogether would lead to larger gains long-term. Which one should the optimizer aim for?

Opinion: Aiming for replacing single relics with the same main stat and set should be sufficient for our use case.

1.3 Displaying results

TODO

Separate numbers for every relic slot, farming that particular domain/SU World.

2 Probabilities

Our goal is to arrive at the probability p of rolling any particular relic, which we specifically split up into $p = p_{\text{main}} \cdot p_{\text{sub}}$ which are the main stat probability and the substat probability respectively.

The goal is to consider a set of relics that has been filtered to a given condition and adding their probabilities to give a single probability of rolling any relic of a set of relics. We use that probability to aggregate the number of runs and TBP later in Section 3.

2.1 Main Stat Probability p_{main}

The main stat probability can be partially or fully bypassed using synthesis with or without Self-Modeling Resin.

Every domain or SU world offers two different relic sets, of which one is desired. Every relic has a uniform chance to be in one of either 2 relic slots if the set is a planar ornament set, or 4 slots for regular sets. Every relic in a given slot has different probabilities of dropping the correct main stat, documented in [1].

The total probability of getting a relic with the correct set, slot and main stat is the product of the probabilities $p_{\text{main}} := p_{\text{set}} \cdot p_{\text{slot}} \cdot p_{\text{stat}}$

Event	Probability
p_{set}	$\frac{1}{2}$
p_{slot}	$\begin{cases} \frac{1}{2} & \text{if planar} \\ \frac{1}{4} & \text{else} \end{cases}$
p_{stat}	Refer to [1]

2.2 Substat Probability p_{sub}

Substat probability is split into two parts that are closely related to each other, which is the distribution of initial substats until all 4 substat slots have been filled, and the distribution of upgrades into desirable substats.

2.2.1 General methodology

Rarity	**Initial substats n_s**	**Max. Level l**
2*	0	+6
3*	1-2	+9
4*	2-3	+12
5*	3-4	+15

Table 1: Stats regarding relic rarity

A relic with a given rarity has n_s initial substats to start with, colloquially referred to being a n_s-liner, eg. *3-liner*. A relic upgrade happens every +3 levels up until the maximum level l for a given relic rarity. Every upgrade adds a new substat not already on the relic (including main stat), until all 4 substat slots have been filled $(4 - n)$, after which each upgrade improves one of the 4 existing substats. **All relics are assumed to be 4-liners with the number of substat slot fills deducted from the total number of upgrades.** For example, a 5* 3-liner is equivalent to a 5* 4-liner with 1 less upgrade.

For example, a 5* relic can drop either as a 3-liner or a 4-liner. The estimated distribution for each is estimated to be around 80/20. A 5* relic can be enhanced up to +15, which is a total of 5 upgrades. If the relic is a 3-liner, then it requires $4 - n = 1$ upgrades to fill all substat slots and has 4 upgrades left to go into the 4 existing substats. If the relic is a 4-liner then no upgrades are required to fill the substat slots and there are 5 upgrades left to go into the 4 existing substats.

Let S be the set of all substats with $|S| = 12$.

Let $t \subset S \setminus \{m\}$ be a set of 4 *initial substats* excluding the main stat m. Let T be the set of all sets of initial substats, which is specifically the set of all combinations of $S \setminus \{m\}$.

Let $n_u = \left\lfloor \frac{l}{3} \right\rfloor - 4 + n_s$ be the number of *substat upgrades* for a given relic, $u \subset \{0, 1, 2, 3\}$ a n_u-multiset of a given substat upgrade distribution and U the set of all substat upgrade distributions, which is specifically the set of all combinations with replacement, where $\{0, 1, 2, 3\}$ refers to the index of the 0-indexed initial substats.

A **roll result** $r = (t, u) \in T \times U$ is a tuple of a initial substat set and an upgrade distribution.

This suffices to find an optimization target that can be filtered for. We **generate all roll results** for a given rarity and **filter them based on an optimization target**. Every roll result has a probability p_{sub} of occuring, which is the product of the probability of the line probability p_l, the probability of the initial substat set p_t and the probability of the upgrade distribution p_u.

Note that roll results consider combinations rather than permutations, so the order of initial substats and substat upgrades are ignored. This decreases the amount of computation by a factor of roughly $4! \cdot 5! = 2880$. Additionally, substat upgrade quality is not considered. Every upgrade has a uniform chance to be 80%, 90% or 100% of the full upgrade value, which is ignored for the calculation in this document, but can be added to the calculation of p_u for an additional computation factor of about $3^5 = 243$.

2.2.2 3-liner vs. 4-liner p_l

Any given relic follows the 80/20 distribution of having an additional initial substat. This means that roll results with the additional substat upgrade have a lower probability of occuring.

$$p_l = \begin{cases} 20\% \text{ if additional initial substat} \\ 80\% \text{ else} \end{cases}$$

2.2.3 Initial substat set probability p_t

Let $r = (s, u)$ be a given roll result. The goal is to compute p_t from t. Every substat has a weight w_t of being picked (see Table 2).

Substat s	Weight w_t
HP(%), ATK(%), DEF(%)	10
Effect Hit Rate, Effect RES, Break Effect	8
CRIT Rate, CRIT Dmg	6
Speed	4

Table 2: Substat probability weights, sourced from [1]

The probability of a substat s being picked is following:

$$p_s = \frac{w_s}{\sum w}$$

However, the probability changes after the first substat, because the substat that was picked cannot be picked anymore and the relative weights change. We solve this by summing the total probability of every permutation of s.

```
let weights = {
  "ATK": 10,
  /**/
};

let subs = [/**/];
let main_stat = /**/;
let total_probability = 0;

for (const perm of permutations(subs)) {
  let p = 1;
  // Main stat cannot be rolled, so we exclude the weight
  let total_weight = weights.values().sum() - weights[main_stat];
  for (const sub of perm) {
    p *= w[sub] / tw;
    tw -= w[sub];
  }
  total_probability += p;
}

log(p); // this contains p_s
```

2.2.4 Upgrades probability p_u

We assume the probability of upgrading any single substat to be uniform (read: everything has the same chance). Let n be the number of upgrades and $k = 4$ the number of available substats. This is a classic problem in combinatorics called the k-multicombination. The number of any given upgrade distribution ignoring order is as following:

$$n_u = \binom{n+k-1}{k} = \frac{(n+k-1)!}{k!(n-1)!}$$

$$p_u = \frac{1}{n_u}$$

2.2.5 Total substat probability

The total substat probability comes out to following:

$$p_{\text{sub}} = p_l \cdot p_t \cdot p_u$$

2.3 Total relic probability

The probability p of a given piece is

$$p := p_{\text{main}} \cdot p_{\text{sub}}$$

.

The probability of rolling one of a set of relics is their sum. We can use this property by filtering the permutation of the substats

3 Aggregation into Trailblaze Power

TODO

Bibliography

[1] "HSR CN Mainstat Substat Distribution". [Online]. Available: https://docs.google.com/spreadsheets/d/1-MgfpwtN0PwR04eiqJp-PcaPB12oIkeaz-icEp-61b0/edit#gid=2001147117