THE MATHEMATICS AND OPTIMIZATION OF OSRS COMBAT
An Application to the Nightmare Zone

Nawar Ismail

22nd February, 2020

Contents

1   Introduction

Oldschool Runescape is a MMORPG which features combat. In combat, players are awarded experience for damage dealt to their opponents (often called monsters). After a certain amount of experience is gained the player may level up, improving the abilities in combat. The mechanics of combat involve accuracy, maximum damage, attack speed, among other details. There have been several attempts to quantify the experience rates according to a player's, and their opponent's levels and equipment. Although they are quite accurate, there are additional corrections that can be applied. Additionally extensions to more complex fighting scenarios has not been performed. We will be specifically considering applications to the Nightmare Zone, a common method of training combat. This game is played with long term investments and considerations in mind, as many achievements are achieved over periods of months or even years. In this sense, even minor optimizations can result in real-world savings for the player. Furthermore, a codebase accompanies this article which allows for more advanced use of the derived formulas (such as in optimization problems). These are the central motivations for this study (and its fun!).

2   Damage

The official Wiki page provides a very good review of maximum hit mechanics. For completion we will provide a brief summary. For some reason, it does not contain information about accuracy calculations. These are can be found (unofficially) from osrsbox (which references DPS calculator by Bitterkoekje, the forum post of which has been archived), MachOSRS (reddit), and [deleted] (reddit). Following this, we will take care to accurately determine the number of hits required to kill an opponent, followed by experience rate calculations. Related calculations have been performed by Nukelawe, which discusses the impact of overkill on expected damage, which is often neglected in damage calculations. However we will justify that the application of this formula is not properly accounted for and we will instead provide an more accurate method.

2.1   Maximum Hit

There are three combat styles in the game: melee, ranged, and magic. The former two are quite similar, while the latter is slightly more involved. In general, there are 5 bonuses that a player can obtain to boost their maximum hit. These are bonuses from: potions, prayer, other, style, and special attack. These will be referred to as: B^{pot}, B^{pray}, B^{other}, B^{style}, B^{SA}, respectively. For non-magic styles, an effective damage level is defined as

$$L^{\text{eff}} \equiv \left\lfloor \left(L + B^{\text{pot}} \right) B^{\text{pray}} B^{\text{other}} \right\rfloor + B^{\text{style}}, \tag{1}$$

where L is either the ranged level or strength level. The base damage, D^{base} is then given by,

$$D^{\text{base}} = C_0 + C_1 S^{\text{eff}} + C_2 E^{\text{str}} + C_3 E^{\text{str}} S^{\text{eff}}, \tag{2}$$

$$\text{where } \{C\} = \left\{ 1.3, \frac{1}{10}, \frac{1}{80}, \frac{1}{640} \right\}, \tag{3}$$

and E^{str} is the ranged or melee strength bonus of the worn equipment. The max hit, M is then given by,

$$\boxed{M = \left\lfloor D^{\text{base}} B^{\text{SA}} \right\rfloor,} \tag{4}$$

where $B^{\mathrm{SA}} = 1$ if no special attack is being used.

For melee, the `Keras` and `Saradomin Sword` are exceptions to this (see the wiki). For ranged, it should be noted that `bolt` effects only have probabilities of activating (and therefore increasing M). For future considerations, this could be handled by defining an effective or average max hit: $\langle M \rangle = PM$, where P is the probability of activation. Additionally, some sources state that some equipment which would be categorized under "other" bonuses such as the `Slayer helmet` actually use $\lfloor D^{\mathrm{base}} B^{\mathrm{SA}} B^{\mathrm{other}} \rfloor$ and so act as a multiplier instead of its expected placement. This is not stated on the wiki, but appears to be correct.

2.1.1 Magic

The base damage for a magic spell is fixed and only multiplicative bonuses are obtainable. A single bonus yields a max hit of,

$$M = D^{\mathrm{base}} E^{\mathrm{str}}, \tag{5}$$

where now E^{str} is the multiplicative magic damage bonus (think "magic strength"). In general, these effects stack, but intermediate flooring calculations can impact the max hit (check the wiki for ordering). `Charge`, `Magic dart`, `Salamanders`, and `Trident of the seas/swamp` have alternate calculations.

2.2 Accuracy

A will denote the player's probability of a successful hit (called accuracy). This attribute depends on both the player and the opponents stats. For all three combat styles, the attacker rolls an attack role, while the defender rolls a defensive roll. First effective levels are calculated,

$$L_A^{\mathrm{eff}} \equiv \lfloor (L + B^{\mathrm{pot}}) B^{\mathrm{pray}} B^{\mathrm{other}} \rfloor + B^{\mathrm{style}} + 8, \tag{6}$$

where the subscript on L_A is used to denote the difference between the effective damage levels discussed above, and these effective accuracy levels. Then the maximum roll is given by,

$$R_{\max} = \lfloor L_A^{\mathrm{eff}}(E + 64) \rfloor, \tag{7}$$

where E is the equipment bonus for the respective attack or defense style. Now the accuracy is given according to,

$$\boxed{A = \begin{cases} 1 - \frac{1}{2}\frac{D_{\max}+2}{A_{\max}+1} & A_{\max} \geq D_{\max} \\ \frac{A_{\max}}{2D_{\max}+2} & \text{else,} \end{cases}} \tag{8}$$

where $A_{\max}$ and $D_{\max}$ represent the maximum attack and defense rolls ($R_{\max}$) respectively. This is plotted in Fig. 1a. No information suggests magic accuracy is any different. No other work / official statements show how special attacks factor into this. It is a fair assumption that $A^{\mathrm{eff}} = AB^{\mathrm{SA}}$ for special attacks that increase accuracy (since no flooring needs to be done for accuracy).

2.3 Average Damage

Here begins the new calculations (if the previous sections were unclear, please read the cited references). The damage done during an attack is uniformly distributed on

$[0, M]$, so each integer between 0 and the max hit have an equal chance of $1/(M+1)$ of occurring. The average damage done during an attack is the expectation,

$$\langle D \rangle = \frac{1}{M+1} \sum_{i=0}^{M} i \tag{9}$$

$$= \frac{1}{\cancel{M+1}} \frac{M(\cancel{M+1})}{2} \tag{10}$$

$$= \frac{M}{2} \tag{11}$$

However, this makes the assumption that the player can always hit up to their max hit. This is not the case if the opponent has less health, h than the maximum hit. This effect is sometimes referred to as *overkill* damage.
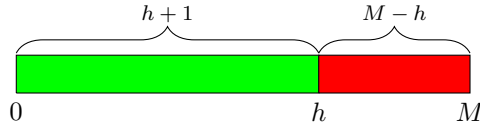
In this case, we could hit every integer below h each with a probability of $1/(M+1)$, and we could hit exactly h (since we're considering hits capped by h) with a probability of $(M-h)/(M+1)$. Averaging the expectations gives,

$$\langle D \rangle_{h<M} = \frac{1}{M+1} \sum_{i=0}^{h} i + \frac{M-h}{M+1} h \tag{12}$$

$$= \frac{1}{M+1} \frac{h(h+1)}{2} + \frac{M-h}{M+1} h \tag{13}$$

$$= \frac{h}{2} \frac{2M-h+1}{M+1} \tag{14}$$

$$= \frac{h}{2} \left(1 + \frac{M-h}{M+1} \right) \tag{15}$$

$$= \frac{h}{2} \left(2 - \frac{h+1}{M+1} \right) \tag{16}$$

Overall then, our expected damage on a successful hit is:

$$\langle D \rangle = \frac{1}{2} \begin{cases} M & \text{if } h \geq M \\ h \left(2 - \frac{h+1}{M+1} \right) & \text{if } h < M \end{cases} \tag{17}$$

This is plotted in Fig. 1b. Nukelawe presents a formula for the *overall* (i.e. not piecewise) expected hit, assuming each starting health is equally likely. We will justify that this isn't the correct approach for calculating turns to kill, or damage per second in the next section. Regardless, it is good to check that our equations agree:

$$\langle D \rangle_{\text{overall}} = \langle \langle D \rangle_{h<M} + \langle D \rangle_{h\geq M} \rangle \tag{18}$$
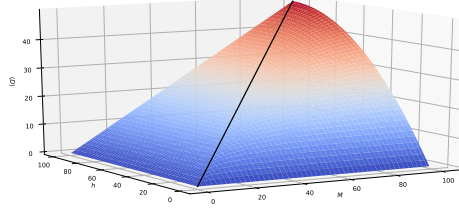
$$= \frac{1}{h_0} \left(\sum_{h<M} \langle D \rangle_{h<M} + \sum_{h\geq M} \langle D \rangle_{h\geq M} \right) \tag{19}$$

$$= \frac{1}{h_0} \left(\sum_{n=M+1}^{h_0} \frac{M}{2} + \sum_{n=1}^{y} \frac{n}{2} \left(2 - \frac{n+1}{M+1} \right) \right) \tag{20}$$

$$= \frac{y(y+1)}{h_0(M+1)} \left(\frac{1}{2}(M+h_0+1) - \frac{1}{3}(2y+1) \right), \tag{21}$$

4
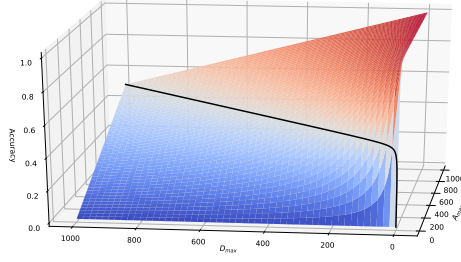
(a) Player accuracy as a function of the player's maximum accuracy roll and opponent's maximum defense roll. The black line is the piecewise boundary.

(b) The average damage of an attack with a maximum hit of M on an opponent with h health. The black line is the piecewise boundary.

where $y = \min(M, h_0)$. The last step is actually quite tricky and involved to show. For that reason it can be found in the Appendix. This agrees with Nukelawe's findings.

2.4   Health after n Attacks

Despite the above being an "average", it is not the average damage expected per hit over the life time of an opponent since certain healths are much more likely to appear than others. As an example, suppose there is an opponent with 100 health fighting an attacker with a max hit of 30. On the first hit, the most likely health is 85. This means that unlike the above calculation, each hit is *not* equally likely. Particularly, this over estimates the contribution in the overkill region, since (depending on the circumstances) let's say 1 or 2 hits is expected per kill. Those 1 or 2 hits are more likely to occur at specific h values, but the average sums across all h equally. In the non-overkill region, this has no impact due to it being constant. So either a Markov-Chain type analysis is required or the following: To determine the health of an opponent after a given number of *successful* attacks (we can include accuracy later), we note that the piecewise damage expectation gives a recursive function for the health, h_n of an opponent after n attacks,

$$h_{n+1} = h_n - \frac{1}{2} \begin{cases} m & \text{if } h_n \geq M \\ h_n \left(2 - \frac{h_n+1}{M+1} \right) & \text{if } h_n < M. \end{cases} \tag{22}$$

This however is too cumbersome to handle at once. Since the health is a decreasing monotonic function, we can consider it in two parts. While h is above or equal to M the solution to the above becomes,

$$h_{n+1} = h_n - \frac{M}{2} \tag{23}$$

$$\implies h_n = h_0 - n\frac{M}{2}, \tag{24}$$

where h_0 is the maximum / starting health. The solution to the other is a bit more complex. After a certain number of iterations the health will drop below M and the second case above will kick in. We'll say this occurs after L iterations (noting that this *average* quantity can be a non-integer),

$$M > h_0 - L\frac{M}{2} \tag{25}$$

$$\frac{2}{M}(h_0 - M) < L \tag{26}$$

$$\implies L = 2\left(\frac{h_0}{M} - 1 \right). \tag{27}$$

5

Now the expected health that the second condition starts at is given by,

$$\langle h_L \rangle = h_0 - 2 \left(\frac{h_0}{M} - 1 \right) \frac{M}{2} \tag{28}$$

$$= M. \tag{29}$$

Thus the second case is expected to starts on iteration L with an initial health of M. For simplicity, we will define $m \equiv n - L$. Returning to our recursive function,

$$h_{m+1} = h_m - \frac{h_m}{2} \left(2 - \frac{h_m + 1}{M + 1} \right) \tag{30}$$

$$= h_m - h_m + \frac{h_m}{2} \frac{h_m + 1}{M + 1} \tag{31}$$

$$= \frac{h_m^2 + h_m}{2(M + 1)} \tag{32}$$

$$h_{m+1} = \gamma(h_m^2 + h_m), \tag{33}$$

where $\gamma \equiv 1/2(M + 1)$. This type of recurrence relation is called a Quadratic Map, and unfortunately it has no closed form solution in general. Some work is shown in the appendix to attempt to simplify it, but no final form was realized. For the sake of completeness, we will call the solution to Eq. 33 $f(m; M, f_0)$. At this point we're left with,

$$h(n; h_0, M) = \begin{cases} h_0 - \frac{1}{2}nM & \text{else if } n \leq L \\ f(n - L; M, M) & \text{otherwise.} \end{cases} \tag{34}$$

In general, we are more interested in the number of iterations that is required to kill an opponent which is the inverse of this function, $n = h^{-1}(h; h_0, M)$. However, since the recurrence relation cannot be solved analytically, we cannot obtain a general expression for this. We could numerically compute this result. However remember that we are dealing with *expectation values* or averages. This means it is totally possible for the inverted function to say "The opponents health will be 18 in 4.5 iterations, on average". This is a problem because our recursive equation can only increment the iteration by one (and we can only start at f_0)! In the next section, we will look at an approximation that will allow us to handle non-integer expectations.

2.4.1 Approximate Solutions

Despite not being able to find a closed-form solution for Eq. 33, we can look for approximate solutions. Although it's rare, there are a few quadratic recurrence relations that **do** have solutions. There are two relevant cases, ones where a constant term is introduced, or ones where the linear term is removed. Since increasing n (and therefore iterations) results in smaller and smaller h_n, it suggests that adding constant terms will heavily skew the results in this region. Removing the linear term is still not ideal since its contribution increases relative to the quadratic term, in the low h_n limit (when this recursive case is required). However this should be less significant and we'll find that it does pretty well! In this case, our recursive relation becomes,

$$h_{m+1} = \gamma h_m^2. \tag{35}$$

Starting with $h_0 = M$, and looking at a few terms reveals,

$$h_1 = \gamma^1 M^2 \tag{36}$$

$$h_2 = \gamma^1 h_1^2 \tag{37}$$

$$= \gamma^1 \gamma^2 M^{(2 \cdot 2)} \tag{38}$$

$$h_3 = \gamma^1 h_2^2 \tag{39}$$

$$= \gamma^1 \gamma^2 \gamma^4 M^{(2 \cdot 2 \cdot 2)} \tag{40}$$

$$\implies h_m = \gamma^{\sum_{i=0}^{m-1} 2^i} M^{2^m} \tag{41}$$

$$= \gamma^{2^m - 1} M^{2^m} \tag{42}$$

$$= \frac{1}{\gamma} (\gamma M)^{2^m} \tag{43}$$

$$\boxed{h_m = \frac{1}{\gamma} \left(\frac{1}{2} - \gamma \right)^{2^m}} \tag{44}$$

This equation is not only a closed form expression, but it can also be inverted!

$$\log_2 (\log_{\frac{1}{2} - \gamma} (\gamma h_m)) = m. \tag{45}$$

This equation asymptotically reaches 0, but the opponent dying occurs when h_m drops below 1 yielding an average kill on iteration,

$$\log_2 (\log_{\frac{1}{2} - \gamma} (\gamma)) = m. \tag{46}$$

How can we make use of this approximation to improve our more accurate calculation? Remember that we are trying to solve the issue of non-integer iterations. Instead of beginning on iteration 0 for the iterative procedure, we can start m at any real number in (0, 1), and iterate upwards on that. For example to get the health at iteration 4.87, we could use our analytic approximation to calculate 0.87 (which uses *one* approximation), then iterate 4 times using the correct equation to get to the final result. So this defines our new best approximation,

$$\boxed{\begin{aligned} h(n; h_0, M) &= \begin{cases} h_0 - \frac{1}{2} n M & \text{if } n \leq L \\ \frac{1}{\gamma} \left(\frac{1}{2} - \gamma \right)^{2^{n-L}} & \text{if } n - L < 1 \\ f \left(n - L; M, \frac{1}{\gamma} \left(\frac{1}{2} - \gamma \right)^{2^{n-L}} \right) & \text{otherwise} \end{cases} \\ \text{where, } \gamma &= \frac{1}{2(M+1)} \text{ and } L = 2 \left(\frac{h_0}{M} - 1 \right). \end{aligned}} \tag{47}$$

To solve for when the opponents health drops below one, $n = h^{-1}(1)$, we must use a numerical root finding algorithm to solve for the zero of $(1 - h(n))$. It is possible that $h^{-1}(1/2)$ is more representative of death than $h^{-1}(1)$. This should be discussed.

2.4.2 Comparisons

We have seen several methods for calculating expected kill iterations. We can now compare these models to numerically simulated fights as in Fig. (2). Regardless of the model, the turns to kill grows quickly when the max hit is low compared to the initial health of the opponent (as expected). The models deviate the most from the simulations when the initial health is low, and when the max hit is high. Both the crude method and the Bitt-Nuke method have unbounded errors, while the Recursive method actually has a maximum near $(h, M) \approx (25, 80)$. Thus the recursive method is not only more accurate (less error), it also holds in more cases.
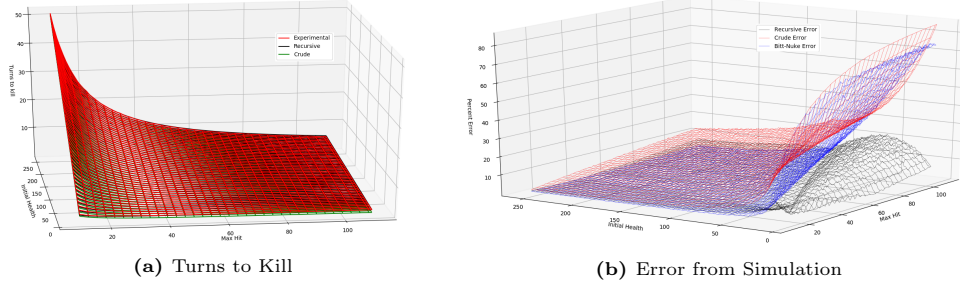
7

(a) Turns to Kill

(b) Error from Simulation

Figure 2: A comparison between different models. The crude model does not consider the effects of overkill. The recursive model is that described in Eq. 47 which best accounts for overkill. The Bitt-Nuke model averages the effect of overkill. The true value can be evaluated by simulating many fights and averaging the results. To get high precision this is a very expensive evaluation. On the left, the expected number of successful hits required for a kill is shown. All three models *appear* quite similar, however this is an artifact of scale. On the right, the percent relative error with respect to the simulations are shown. The noise is evident and occurs due to the difficulty of efficiently simulating the required number of fights. The peak error is the region in which overkill is expected to occur. The average error of the recursive method is 4.94% with a standard deviation of 4.8%, while the Bitt-Nuke method has an average of 10.57% and a standard deviation of 17.4%.

2.5 Experience and Damage Per Second

We saw that by inverting the damage per attack equation, Eq. (47), we could determine the average number of successful attacks required to kill an opponent based on the max hit and starting health. To extend this to number of *total* attacks we have to introduce accuracy, $a \in [0, 1]$. For the health per turn, $h(n)$, we simply note that $h(n) \to h(na)$ will appropriately reduce the number of "effective turns". Similarly, for turns to kill, $h^{-1}(1) = n \to n/a$ appropriately extends the number of turns expected. So multiplying and dividing easily accounts for accuracy.

This can further be extended to time to kill by considering the time it takes for each attack. Let s be the attack speed (in attacks per second), then $n/(as)$ is the required time to kill an opponent. While the health at a given time, t is $h(tas)$. This is a good time to note that the error shown in Fig. (2) is unchanged since these multiplicative factors cancel out in the relative error calculation.

Experience is calculated as a multiple, e of the damage done. So if the opponent has h_0 health, and they are killed in $n/(as)$ seconds, then the experience per second is,

$$\boxed{E = eash_0/n} \tag{48}$$

noting that n is also a function of h_0, so E does *not* increase linearly with h_0. For most opponents, $e = 4$ however there are exceptions.

3 Nightmare Zone

We can now attempt to extend this into an useful application. For practical purposes, the Nightmare Zone (NMZ) is a combat training ground minigame that offers a low intensity, but efficient method of training combat skills. The player enters an arena with several bosses they have previously fought in the main game. At any given time, four (randomly chosen - with replacement) are in combat with the player. Before beginning, the player can decide a minimum of five bosses they would like to re-fight.

This can be seen as a first a check, and second an optimization problem (to maximize experience). At this point we have calculated the experience per hour in terms of: accuracy, max hit, initial opponent health, and attack speed. The accuracy and max hit

Figure 3: A player (bottom-right) fighting against four bosses in the NMZ. Three of the bosses at this time happened to be the same, however each boss is expected to occur with equal probability.

are involved yet straightforward calculations, while the other parameters are generally in a database. We have only made two assumptions so far, no health regeneration, and constant combat, along with 1 approximation in the turns to kill formula. 1 health is regenerated every minute the opponent is under their maximum health. [Pretty sure I've seen exceptions] Thus, kills under on minute shouldn't be affected by this, while kills on the order of a few minutes only have a very small influence (since the duration of the fight typically implies high health). The lower the initial health of the opponent, the lower the impact of health regeneration.

3.1 Experimental Considerations

The health regeneration has a small effect of increasing damage per second by extending the non-overkill region and increasing the damage cap in the overkill region). Calculations are performed assuming continuous combat but due to random spawning, they may line up in a way that delays combat. Additionally, larger opponents would increase the probability of delay. Maintaining the player's own health requires some sort of consumption which may introduce delays. Finally, the bosses may spawn in unequal proportions (on average this should wash away, but may either increase or decrease the experience rates). These effects are summarized in Table. 1.

3.2 Validation

To increase the probability that the player will remain in combat, only opponents who occupy a single tile will be used (that way more can stack up on the player, and they are more likely to retaliate against someone). For the sake of accuracy and reducing

Property	Regeneration	Intermittent Combat	Sustainability	Equal Counts
Impact	Higher	Lower	Lower	Random
Multiplicative	No	Yes	Yes	Yes

Method	Overkill Consideration	Experience Prediction
Crude	Over Estimate	Loose Lower Bound
Recursive	Approximate	Upper Bound
Simulation	Approximate	Upper Bound
Bitter	Over Estimate	Loose Lower Bound

Table 1: The top table shows the effect on experience rates when certain factors are properly considered. Multiplicative factors means that that the ratio between two different rates are independent of these quantities, non-multiplicative factors appear inside the particular method which in general would not cancel. The bottom table relates how overkill is considered to the over/under prediction of experience rates. The properties with the largest influence decrease the experience rate. So the methods which well-approximate overkill are upper bounds. Those that over estimate it are lose lower bounds since otherwise they would be upper bounds so this additional decrease may or may not put it below the true expectation.

human error, no prayers, or special attack will be used. Additionally, tank gear, and absorption potions will be used to minimize the influence of sustaining the player's health. Runelite's built-in experience rate tracker will be used to collect data.

Figure 4: The main setup used for validation. Due to the nature of testing, some health, strength, and combat levels were gained, but were properly taken into account.

3.2.1 Time-Dependent Boosts

There are several time-dependent boosts in the game, most notably potions. Additionally, Dharok is a set of equipment which provides increased damage based on the health missing from the player. Since health recovers at a rate of one health per minute, and the boosted level from potions decreases one level per minute, these introduce time-dependent experience rates. Thankfully, the treatment is relatively simple. Let $a = a(t)$, and $M = M(t)$. At each boosted state, we can calculate the expected experience rate,

and average over the boosted states:

$$\langle E \rangle = \frac{1}{N} \sum_{i=0}^{N} E(a(t_i), M(t_i)) \qquad (49)$$

For example, if the player re-drinks a potion after $N = 10$ levels have dropped, then the above determines the average experience. Things are a little bit more complex when both potions and Dharok are used since the timings might not line up, but for large times the detail should get averaged out.

3.2.2 Comparisons

Settings	Crude	Recursive	Simulation	Bitt-Nuke	GamePlay
Easy No Potions $M \in [26]$ 8.36h	66.9 +15.19%	59.1 +1.83%	58.5 +0.78%	56.7 −2.42%	58.1±0.6 [1.0%]
Hard No Potions $M \in [27]$ 8.19h	68.7 +6.52%	65.6 +1.71%	65.5 +1.49%	65.0 +0.75%	64.5±0.4 [0.7%]
Easy Potions $M \in [27..32]$ 6.84h	79.8 +18.16%	68.6 +1.62%	68.3 +1.21%	65.7 −2.69%	67.5±1.0 [1.5%]
Hard Potions $M \in [27..32]$ 6.81h	79.6 +6.25%	75.3 +0.55%	75.2 +0.45%	74.7 −0.23%	74.9±0.9 [1.2%]

Table 2: Experience per hour predictions from different methods. Potions and Easy or Hard bosses which correspond to the boss difficulty (which only increases health and offensive capability) are varied. Each method reports two values, the xp/h [in kilo-xp] and the signed percent difference from the game play. Green coloring indicates there is agreement within error, red is used otherwise. The bracketed quantities are the relative error of the reported rate. Max hits are given as a range when potions are used. Data collection periods are also given in hours.

Since the largest (still small) source of errors in all models (constant combat, and health sustaining (absorptions potions)) negatively impact experience rates, we expect the models to be tight upper bounds on the actual gameplay experience. However, it's possible that the bosses don't spawn in equal amounts, which may bias the results in either direction (Runelite doesn't properly record kill counts to check this). For no potions, only the max hit matters, but for potions, the strength level also determines what the boost range. To prevent needing to account for this, when potions are used, runs are only carried out for a single strength level. For no potions, as long as the max hit is

unchanged any strength level can be used.

Bitt-Nuke over-considers low end which lowers estimate, but it can't be known if this lowering beats the unaccounted for non-constant combat. Which may explain why it was occasionally under-estimated results, while the Recursive method was always expected to be a upper-bound. This means that with additional corrections, the Recursive model will perform better than Bitt-Nuke, although as reported this seems effect isn't strong.

3.3 Optimization

In the NMZ, we can consider the player's opponents, and equipment as parameters to an optimization problem. This could also consider equipment cost.

4 Improvements

Health regeneration. Analytic solution? Constant Combat (acts as reduced weapon speed so multiplicative). Health Sustainability. The last two shouldn't impact rankings since they are multiplicative and independent of the calculation method. Based on this, we can say that exact rankings is more important than exact approximations for practical purposes. Is the treatment $h = 1$ for a kill valid for all methods? Since crude and Bitterkoekje can get to 0? I've seen (enemy) hp go up in under a minute. There must be a global timer, not just every 60s after you drop below max health.

5 Conclusion

We summarized the combat mechanics behind effective levels, accuracy, and maximum hits. We performed detailed calculations to determine piecewise average damage, based on whether overkill was present. In contrast to the Bitt-Nuke model, we maintain this piecewise relation and evaluate an recursive scheme to determine a more accurate form for the average damage. An analytic approximation was used to handle non-integer turns to kill an opponent, which can ultimately be used to calculate experience per hour. We discuss the effects which are not considered like health regeneration, and constant combat. Finally, experience per hour models are compared. The Recursive model is the best model due to its accuracy, and since it satisfies the expected upper bound property. Future work may look into forming an optimization problem.

Appendix A

Average Damage

The average damage described in Section 2.3 can be expanded to give,

$$\langle D \rangle_{\text{overall}} = \frac{1}{h_0} \left(\sum_{n=M+1}^{h_0} \frac{M}{2} + \sum_{n=1}^{y} \frac{n}{2} \left(2 - \frac{n+1}{M+1} \right) \right) \tag{50}$$

$$= \frac{1}{h_0} \left(\frac{M}{2}(h_0 - y) + \sum_{n=1}^{y} n - \frac{1}{2} \sum_{n=1}^{y} n \frac{n+1}{M+1} \right) \tag{51}$$

$$= \frac{1}{2h_0} \left(Mh_0 - My + y(y+1) - \frac{1}{2(M+1)} \sum_{n=1}^{y} (n^2 + n) \right) \tag{52}$$

$$= \frac{1}{2h_0} \left(Mh_0 - My + y(y+1) - \frac{y(y+1)}{2(M+1)} - \frac{y(n+1)(2y+1)}{6(M+1)} \right) \tag{53}$$

$$= \frac{1}{2h_0(M+1)} \left(M^2 h_0 - M^2 y + My^2 + yM + Mh_0 - My - \frac{y^3 - y}{3} \right) \tag{54}$$

$$= \frac{y(y+1)}{h_0(M+1)} \left(\frac{M(M+1)h_0}{2y(y+1)} + \frac{(y-M)M}{2(y+1)} - \frac{y-1}{6} \right) \tag{55}$$

$$= \frac{y(y+1)}{h_0(M+1)} \left(\frac{M(M+1)h_0}{2y(y+1)} + \frac{(y-M)M}{2(y+1)} + \frac{y+1}{2} - \frac{1}{3}(2y+1) \right) \tag{56}$$

where $y = \min(M, h_0)$. In the second line, we used:

$$\sum_{a+1}^{b} 1 = \begin{cases} b - a & \text{if } b > a \\ 0 & \text{else} \end{cases} \tag{57}$$

$$= b - \begin{cases} a & \text{if } b > a \\ 0 & \text{else} \end{cases} \tag{58}$$

$$= b - \min(a, b) \tag{59}$$

To finish, let's focus on,

$$\frac{M(M+1)h_0}{2y(y+1)} + \frac{(y-M)M}{2(y+1)} + \frac{y+1}{2} \tag{60}$$

$$= \frac{1}{2y(y+1)} \left[M(M+1)h_0 + y(y-M)M + y(y+1)^2 \right] \tag{61}$$

$$= \frac{1}{2y(y+1)} \left[M^2 h_0 + Mh_0 + My^2 - M^2 y + y^3 + 2y^2 + y \right]. \tag{62}$$

This is a hard equation to simplify since the M's and h_0's are implicitly embedded in the y's, but if you play with it long enough you can "discover" a way to simplify it - a form *power reduction* that relies on getting rid of as many y's as possible.

A.1 Power Reduction

I'd like to preface the next part by saying the final result can easily be determined by plugging in m as the min, and h_0 as the min and combining the result. In this instance it works out nicely, but we will focus on general machinery to solve these problems assuming the solution was not so nice. Our goal here is to pull the m's and h_0's out of y.

To do this, let's see if there is a way to construct y^2 from the other variables, specifically only using y^1. We know that if $M < h_0$, we need a term like M^2, and in the opposite case, we need a term like h_0^2,

$$y^2 \sim M^2 \text{ or } h_0^2. \tag{63}$$

Based on this, we should be able to use min to switch between these two. So if we write the first term using y, we'd have something like My, which is true when M is the minimum. If it isn't the minimum, there should be a second term which cancels the now Mh_0 term plus the required h_0^2 term:

$$y^2 = My + h_0(y - M) \quad (!) \tag{64}$$

Using the same logic, we can inductively deduce,

$$\boxed{y^{n+1} = My^n + h_0^n(y - M) = My^n + h_0^n y - Mh_0^n.} \tag{65}$$

(and as an identity for the math people, with $\gamma = \min(a, b)$):

$$\boxed{\gamma^{n+1} = a\gamma^n + b^n(\gamma - a) = a\gamma^n + b^n\gamma - ab^n} \tag{66}$$

In fact, this holds for max as well, or any *similar* piece-wise function. Writing this as a recursive sequence by letting $g(n) = \gamma^n$ yields,

$$g(n + 1) = ag(n) + b^n(g(1) - a), \tag{67}$$

Under the initial condition $g(1) = \gamma$, WolframAlpha gives the general solution as,

$$g(n) = a^n + (\gamma - a)\frac{a^n - b^n}{a - b} \tag{68}$$

$$\boxed{g(n) = a^n + (\gamma - a)\sum_{i=0}^{n-1} a^{n-i-1} b^j,} \tag{69}$$

where the second line uses the difference of powers formula. This could have been solved by hand, but we've had enough fun with recursion in the other sections! This yields,

$$g(1) = \gamma \tag{70}$$
$$g(2) = a^2 + (\gamma - a)(a + b) \tag{71}$$
$$= a^2 + a\gamma - a^2 + b\gamma - ab \tag{72}$$
$$= a\gamma + b(\gamma - a) \tag{73}$$
$$g(3) = a^3 + (\gamma - a)\frac{a^3 - b^3}{a - b} \tag{74}$$
$$= a^3 + (\gamma - a)(a^2 + ab + b^2) \tag{75}$$
$$= a^3 + \gamma a^2 + \gamma ab + \gamma b^2 - a^3 - a^2 b - ab^2 \tag{76}$$
$$= \gamma a^2 + \gamma ab + \gamma b^2 - a^2 b - ab^2. \tag{77}$$

These agree with the original iterative equation. Okay, so this is a bit overkill since at most y^3 appears, so having general powers isn't too helpful. Nonetheless, we can now reduce the powers of y in the original equation, and see how that simplifies things.

A.2 Simplifying

We can now reduce the bracketed term in Eq. 62:

$$M^2 h_0 + M h_0 + M y^2 - M^2 y + y^3 + 2y^2 + y \tag{78}$$

$$= M^2 h_0 + M h_0 + M^2 y + h_0 y M - h_0 M^2 - M^2 y + y M^2 + y M h_0 + y h_0^2 + \tag{79}$$

$$- M^2 h_0 - h_0^2 M + 2My + 2h_0 y - 2h_0 M + y \tag{80}$$

$$= \left(-M^2 y + y M^2 + M^2 y + y M h_0 + y h_0^2 + 2My + 2h_0 y + y + h_0 y M \right) + \tag{81}$$

$$\left(M^2 h_0 + M h_0 - h_0 M^2 + -M^2 h_0 - h_0^2 M - 2h_0 M \right) \tag{82}$$

$$= \left(2My + 2h_0 y + y + 2y M h_0 + M^2 y + y h_0^2 \right) + \left(-M^2 h_0 - h_0^2 M - h_0 M \right) \tag{83}$$

Having eliminated the "hidden" variables, let's try to re-group into powers of y:

$$= y^2 + \left(My + h_0 y + y + 2y M h_0 + M^2 y + y h_0^2 \right) + \left(-M^2 h_0 - h_0^2 M \right) \tag{84}$$

$$= 2y^2 + y + 2y M h_0 + M^2 y + y h_0^2 + -M^2 h_0 - h_0^2 M + M h_0 \tag{85}$$

$$= y^2 M + y + y^2 + y^2 h_0 + y^2 + M h_0 \tag{86}$$

$$= y \left(yM + y h_0 + y + M + h_0 + 1 \right) \tag{87}$$

$$= y \left(y + 1 \right) \left(M + h_0 + 1 \right) \tag{88}$$

Putting this into the corresponding term in Eq. 56 gives

$$\frac{1}{2y(y+1)} y(y+1) \left(M + h_0 + 1 \right) = \frac{1}{2} \left(M + h_0 + 1 \right) \tag{89}$$

and so finally we arrive at,

$$\boxed{\langle D \rangle_{\text{overall}} = \frac{y(y+1)}{h_0(M+1)} \left[\frac{1}{2} \left(M + h_0 + 1 \right) - \frac{1}{3}(2y+1) \right].} \tag{90}$$

Appendix B

Analytic Form Attempt

Here we will attempt to derive a more useful form of Eq. 33. We will start with a slightly simpler relation,

$$h_{n+1} = h_n^2 + h_n. \tag{91}$$

We note that in general, this will be an expansion in terms of h_0,

$$h_n = \sum_{i=1}^{m} a_i h_0^i. \tag{92}$$

Our goal then, is to determine the set of coefficients a_i. These coefficients implicitly depend on n. Our next iteration will look like,

$$h_{n+1} = \left(\sum_{i=1}^{m} a_i h_0^i \right)^2 + \sum_{i=1}^{m} a_i h_0^i. \tag{93}$$

The squared term is actually very messy to deal with, requires the use of the Multinomial Theorem which (for the specific case $p = 2$) states,

$$\left(\sum_{i=1}^{m} x_i \right)^2 = 2 \sum_{\{k_m\}=2} \prod_{t=1}^{m} \frac{x_t^{k_t}}{k_t!}, \tag{94}$$

where $\{k_m\} = 2$ denotes the sets of m integers which add to two. For $m = 4$ one example might look like: $(1, 0, 0, 1)$. For those of you familiar with this, these sets arise from integer partitioning, but we won't go over how these are generated, but just know there are algorithms for generating them. We will drop the $= 2$ portion for brevity. Our form can be obtained with the transformation $x_i \to a_i h_0^i$,

$$\left(\sum_{i=1}^{m} a_i h_0^i \right)^2 = 2 \sum_{\{k_m\}} \prod_{t=1}^{m} \frac{a_t^{k_t} h_0^{t k_t}}{k_t!} \tag{95}$$

$$= 2 \sum_{\{k_m\}} \prod_{t=1}^{m} \frac{a_t^{k_t}}{k_t!} h_0^{\sum_{t=1}^{m} t k_t}. \tag{96}$$

Now let's return to the general n'th term (Eq. 93),

$$h_n = 2 \sum_{\{k_m\}} \prod_{t=1}^{m} \frac{a_t^{k_t}}{k_t!} h_0^{\sum_{t=1}^{m} t k_t} + \sum_{i=1}^{m} a_i h_0^i. \tag{97}$$

As-is, these cannot be combined. To handle this, we impose an ordering on $\{k_m\}$ so they they are given in increasing values of $\sum t k_t$, we will denote this as $\{k_m\}_i^L \uparrow$, where L indicates the sum's value for the current set, and i is the index of the current set. Now,

$$h_n = 2 \sum_{\{k_m\}_i^L \uparrow} \prod_{t=1}^{m} \frac{a_t^{k_t}}{k_t!} h_0^{L_m} + \sum_{i=1}^{m} a_i h_0^i. \tag{98}$$

Our only hope of combining these terms is to match up the powers. The good news is that now each term on the left, and right correspond to a single power.

$$h_n = 2 \sum_{\{k_m\}_i^L \uparrow} \left(\prod_{t=1}^m \frac{a_t^{k_t}}{k_t!} + a_i \right) h_0^{L_m}, \tag{99}$$

Now, we should start from the bottom $(n = 0)$ and see if we can build up. For $n = 0$, $h_1 = h_0^2 + h_0$. So the coefficients are $\{a\}^1 = (1, 1, 0, 0, 0, ...)$, where the zero padding accounts for the fact that the powers on the linear sum are less than the powers on the quadratic sum.

We can read off the coefficient is we sort-of invert the sum notation. Let's instead sum across i.

$$h_n = 2 \sum_i \left(\prod_{\{k_t\}_i}^m \frac{a_t^{k_t}}{k_t!} + a_i \right) h_0^{L_i}. \tag{100}$$

Now the coefficient for the L_i'th power for the next iteration is

$$a_{L_i} = 2 \left(\prod_{\{k_t\}_i}^m \frac{a_t^{k_t}}{k_t!} + a_i \right). \tag{101}$$

This actually works out really well since the coefficients for our original problem simply get multiplied by γ! Let's make another simplification by letting k_t be represented by a vector, $\vec{k}$ and constructing respective operations,

$$a_{i,n+1} = 2 \frac{\boldsymbol{a}_n^{\boldsymbol{k}_i}}{\boldsymbol{k}_i!} + 2 a_{i,n}. \tag{102}$$

Here a vector to the power of a vector is the product of the element-wise powers, and the factorial of a vector is the product of the element-wise factorials. This isn't very useful since it still requires an iterative scheme.