Probabilistic Fire Analysis

Muir the Mage

1 Introduction and Usage

This application is meant to compute the distribution of damage that a group of fire mages may expect to do. This may best be considered a mage team optimizer in that the output assumes that all mages have the stats. The innovation here is that a simulation at best can only sample this distribution rather than provide a full perspective as to the damage the mage team may do. Furthermore this application allows you to set a fight length determined by the number of casts.

The application was written using Matlab. In order to run it you do not need a Matlab license but will need to install the framework using the installer. Available on Github, simply download the project as a Zip, then use **FireAnalysis_web.app** on Mac OSX or **FireAnalysis_web.exe** on Windows to install both the application and Matlab framework. Additional files are not required but are provided for those interested in the implementation (and submitting bug reports).

2 Assumptions

The application works takes in the number of mages in the group and the intellect, bonus critical hit and bonus hit chance from gear then computes a distribution as if attacking a level 63 boss. Wowhead has a helpful tool for finding the stats for different gear setups. The application assumes that the mage is consumed and buffed minimally with the following

1. Greater Arcane Elixir
2. Greater Firepower
3. Greater Wizard Oil
4. Arcane Intellect
5. Mark of the Wild

It is also assumed that the mage has 3 talent points in both Elemental Precision and Critical Mass as well as 5 talent points in both Improved Fireball and Ignite and is using Fireball Rank 12. Options are included for world buffs which

increase your critical strike chance or multiplicative adjust your total intellect in the case of the Spirit of Zandalar buff.

Notable options which are intentionally not included are Gnome racial, Power Infusion, trinkets and improved scorches. These factors will scale up your damage proportionally for any gear load out that you select such that they can be ignored as constant multiples.

3 Under the Hood

In this section we will explain the math behind the calculations such that the reader may provide validation and feedback for improvement.

3.1 Direct Damage

The direct damage of fireballs is straight forward to compute. The damage distribution of a single Fireball can be expressed as a sum of box functions B_0.

$$B_0\left(x; a, b\right) = \begin{cases} \frac{1}{b-a} & x \in [a, b] \\ 0 & else \end{cases}$$

In this section we will use S, H, C, M, and N to denote the spellpower, bonus hit chance, bonus critical hit chance, number of mages, and number of casts respectively. Then the damage distribution of a single Fireball is given by

$$P^1\left[d = x; H, C\right] = (H - C) \cdot B_0(x; d_1, d_2) + C \cdot B_0(x; 1.5 \cdot d_1, 1.5 \cdot d_2) + \delta\left(\max(1 - H, .01) \cdot x\right)$$

where $d_1 = 596 + S$ is the lower bound damage and $d_2 = 760 + S$ is the upper bound damage for a non-critical hit, and δ is the Dirac delta function.

Then the probability of total damage x after two casts is given by the convolution of P^1, the sum over y that y damage is done on the first cast and $x - y$ on the second.

$$P^2[d = x] = \int_{y=0}^{\infty} P^1[x - y] \cdot P^1[y] = \left(P^1 * P^1\right)[x]$$

The trend holds such that the distribution damage after k casts P^k is given by the convolution of P^{k-1} and P^1, $P^k = P^{k-1} * P^1$. Note that in order to obtain the expected Damage Per Second from the damage distribution one simply divides by k times the cast time of Fireball.

3.2 Ignite Damage

If it were known the expected number of times ignite hits the 2 and 4 second mark (resulting in damage) for each number of stacks on the ignite then it is straight forward to compute the distribution of damage ignite alone gives. This is because the damage distribution of a k stack ignite P_i^k is given by the

distribution of k Fireballs when mages have 100% hit chance and critical hit chance multiplied multiplied by the 40% damage of ignite and divided by two ticks, $x/(.4 \cdot 2) = 1.25x$.

$$P_i^k\left[d = x\right] = P^k\left[d = 1.25x; H = 1, C = 1\right]$$

Then the damage distribution of an ignite with k stacks which is expected to tick t_k times is simply $P_i^k[x/t_k]$. The total ignite damage is then given by the convolution of these.

Thus we focus our attention on evaluating the number of ticks expected for each number of stacks, t_k. The positive side of the large batch window is that the state of ignite only changes every 200ms, meaning that there are 4 second/ignite $\cdot$ 1/200 states/second = 20 states/ignite for each of the 5 stacks, plus one state for no ignite. This means there are only 101 possible states for the ignite to be in.

The state of the ignite is altered in two way: a cast is completed or one batch window will execute. When cast is completed there is a probability C chance that the number of stacks on the ignite will increase by one (if below 5) and furthermore increase the remaining time on the ignite if less than two seconds, otherwise nothing will happen. The execution of a batch window will simply tick down the timer on the ignite, removing when it has exhausted.

We will track the state of the ignite through one cast cycle. This means we make the assumption that all M mages complete one cast in each window of 3 seconds = 15 batches (the cast time of Fireball). This means that each cycle of casts entails the mixing of N mage cast transition with 15 batch window changes. We will use a Markov chain to track the ignite state distribution. A brief review of Markov chains follows.

A Markov chain is determined by the matrix T of transition probabilities. $T(i, j)$ gives the probability of moving from state i to state j. Then note that the product of two transition matrices T_1 and T_2 gives entries which are the probability of moving one step according to the probabilities of T_1 then one step according to the probabilities of T_2, $T(i, j) = (T_1 \cdot T_2)(i, j) = \sum_k T(i, k) \cdot T(k, j)$. Then the state distribution can be measured by providing an initial state row vector t_0^t, which has a 1 in the position corresponding to having no ignite and 0s everywhere else, then finding the product $t_0^t T$.

Letting T_c be the transition matrix for a cast being completed and T_b being the transition matrix for the batch window, the one-cycle transition matrix is given by a product of 15 matrices T_b and M matrices T_c in some order. Likewise the distribution after k batches is given by a product of k matrices T_b and some number $m \in \{0, \ldots, M\}$ matrices T_c, as any number of mages may have actually completed a cast in the first window. A description of T_b and T_c is included in the appendix.

The goal here is to make weak assumptions about the number of mages that complete a cast in each batch window, so we make the assumption that any ordering is equally likely and compute the expected transition after i batches $\bar{T}_i$ over this distribution. Consider the matrix-valued polynomial $P_i(x) = (I +$

$T_c + x \cdot T_b)^(M + i)$, and note that if T_c and T_b were to commute we would have the following identity.

$$P_{i,j}(x) = (T_c + x \cdot T_b)^{i+j} = \sum_k T_c^{i+j-k} T_b^k x^k$$

From the above we can see that even when T_c and T_b do not commute we have $P_{i,j}(x) = \sum_k \alpha_{k,j} x^k$ where the coefficient $\alpha_{i,j}$ is the sum of products of matrices with i matrices T_b and j matrices T_c. This means that expected transition after i batches is given as

$$\bar{T_i} = \frac{1}{n_i} \sum_{j=0}^{M} \alpha_{i-1,j} \cdot T_b$$

where n_i is a normalization constant $n_i = \sum_j \binom{i-1+j}{j}$. An explanation of one method of computing $\alpha_{i,j}$ is provided in the appendix.

Then the probability of each state after batch i is given by $t_0^t \bar{T_i}$. The expected number of ticks after one 15 batch cast cycle may be approximated by

$$t_0^t \bar{T_s} = \sum_{i=1}^{15} t_0^t \bar{T_i} = t_0^t \left(\sum_{i=1}^{15} \bar{T_i} \right)$$

Then the expected number of batches in each state after k cast cycles may be approximated by $\bar{T_s} \cdot \bar{T}_{15}^k$ such that the total expected number of batches spent in state number i is given by the i-th element of the vector

$$v_N = t_0^t \sum_{k=1}^{N} \bar{T_s} \cdot \bar{T}_{15}^k$$

The reason this is only an approximation is that this expression considers a possibility where the ordering of casts changes between cast cycles; such that it factors in a chance that all casts hit in the last batch within the first cycle and within the first batch on the second despite only 200ms elapsing. However, this these rare impossible scenarios play a very minor role because they are very rare and in a way highly unlikely scenarios which are favorable to ignite remaining up are countered by highly unlikely scenarios which are unfavorable.

4 Appendix

4.1 Transition Matrices

The state space will be enumerated such that

$$\mathit{index}(\mathit{stacks}, \mathit{time}) = \begin{cases} 20 * (\mathit{stacks} - 1) + \lfloor \mathit{time}/.2 \rfloor & \mathit{stacks} \neq 0 \\ 101 & \mathit{stacks} = 0 \end{cases}$$

We first define a matrix which will be useful in compressing our description of transition matrices. Let S_k be the $k \times k$ non-periodic shift matrix (which is a slight alternative to the k-th root of the identity), $e_{i,k}$ be the ith standard basis vector in $\mathbb{R}^k$ and I_k is the $k \times k$ identity matrix. Also let $e_k = \sum_i e_{i,k}$

$$S_k(i,j) = \begin{cases} 1 & j = i+1, i < k \\ 0 & else \end{cases} \qquad e_{i,k}(j) = \begin{cases} 1 & j = i \\ 0 & else \end{cases}$$

This means that we can write the spell transition matrix T_c for a spell interacting with the attack table (hitting, missing, critical hitting).

$$T_c = \begin{bmatrix} C \cdot (S_5 + e_{5,5}e_{5,5}^T) \otimes \begin{bmatrix} I_{10} & 0_{10\times 10} \\ I_{10} & 0_{10\times 10} \end{bmatrix} + & (1-C)\cdot I & 0_{100,1} \\ C & 0_{99,1} & 1-C \end{bmatrix}$$

The second type of transition is caused simply be time moving forward. At the end of a batch window the time left on the ignite decreases by .2 or one state regardless of the number of stacks on the ignite. Thus the time change transition matrix T_b is given by

$$T_b = I_5 \otimes S_{20} + \begin{bmatrix} e_5 \otimes e_{20,20} & 1 \end{bmatrix} e_{101,101}^T$$

Note that both T_b and T_c are row stochastic such that as an initial distribution is applied on the left we again obtain a distribution.

4.2 Computing Coefficients

In order to compute the k-th coefficient a polynomial $P(x) = \sum_i^N \alpha_i x^i$ we can use a root of unity filter. Let $\omega = e^{2\pi i/(N+1)}$ then

$$\alpha_k = \frac{1}{N+1} \sum_{i=0}^{N} \omega^{-k\cdot i} P(\omega^i)$$

This is because for all but α_k the terms of the summation run over the finite group, summing to zero. For the term α_k the coefficients are each 1 such that the sum is $(N+1)\alpha$ before being divided out.