SJ PH

Fleeing from Terror:
Considering Safety When Designing Public Spaces in the Age of Mass Murder

Nic Fishman

Sometimes, the safest thing people can do is flee from danger. However, what if they are in a room with dozens, or hundreds, of other people when that danger arises? How do they get out? And when architects designed the room–whether an intimate theater, concert hall, or sprawling convention center–did they think about how to optimally allow people to exit in an emergency?

In the following paper we establish a standard for evaluating a given room's safeness: We use a model-predicted evacuation time resulting from a simulated terror crisis as our safety metric. Then, we develop algorithmic methodologies to optimize room design, and propose a regulatory framework for ensuring future construction projects protect building occupants.

Our safety-assessment algorithm first takes in a prospective floor-plan design. Then, we computationally run a series of randomized terror scenario simulations, keeping record of the amount of "time"–represented with a conceptual, particle physics-based proxy–that it takes for the population (e.g. a group of theater-goers) to get out. This "time" is averaged across 8 simulations to minimize variance, producing a metric we term "Time to Exit" (TTE). This TTE is then inputted into an iterative statistical process which adjusts the parameters of the floor-plan (e.g. exit-door placement) in order to create a modified design that better minimizes the room's TTE score. By evaluating and creating rooms that ensure short evacuation times, we address a societally relevant public health issue: Terror crisis response.

1 Introduction

In the modern era, it is rare for a theater in the United States to catch fire and cause hundreds of people to die. One hundred years ago, this particular tragedy was all too common, but after decades of legislation, updated fire codes, and improved building materials, we have largely put a stop to theater fires. All of this regulation has directly translated into huge public health gains: After all, designing better buildings makes people safer.

Today, we face a tragedy of similar scale: Mass shootings motivated by individuals who can do extraordinary damage. Hundreds of Americans die each year in gun-fueled rampages, an increasing proportion of them children when accounting for recent school-centered shootings [Cha17]. As this public health issue continues to escalate, it becomes clear that we need a policy solution that keeps people safe. Perhaps, the most obvious answer is gun control, but this is a highly contentious issue in the United States and is politically intractable. The next logical step may be to think about establishing mental health screening, identifying potentially unstable individuals, and providing them therapy to help them become more mentally healthy, thereby stunting atrocities at the perpetrator-level. This seems sensical, but experts have concluded "the sorts of individuals who commit mass murder often are either not mentally ill or do not recognize themselves as such." [Cha17]. Most budding, bloodthirsty killers would be resistant to therapy and other such interventions, because they see the outside world as the root of all their problems [Kha17]. In other words, while such a therapy-based approach is beneficial for the general public good–improving national mental health is always a great objective–it does not feasibly mitigate harms from terror-based events specifically.

A more practical solution is to think about how we design our public spaces, and to consider how we can — moving forward — design rooms that are intrinsically safer. Hence, we propose an objective framework for evaluating a theater's "Time to Exit." Note that in this analysis, we use the word "theater" to represent generalized enclosed places where people congregate, from convention centers to stadiums to movie theaters to classrooms. Directly defined, "Time to Exit" (TTE) is the time it would take everyone to evacuate a room in response to a sudden attack. If, after evaluating the TTE for a user-submitted theater design, we generate an improved room schematic that is modified with computationally optimized exit-placements and possesses a separate (hopefully lower) TTE, then we can evaluate how safe a theater currently is, and compare this to how safe it could be. The core of this algorithmic framework is establishing an effective model to simulate how people would flee a theater when a terrifying actor appears. To accomplish this simulation task, we take inspiration from chemistry, using a heavily modified version of the Metropolis algorithm used by chemists to simulate fluid dynamics. Our framework can model TTEs for any size of theater, any number of people, and any number of barriers (e.g. benches or tables) in the room. This framework can then be applied in a legislative setting, allowing regulatory bodies to set an acceptable threshold on how much a proposed theater design's safety score can deviate from its respective optimized design safety score, and subsequently, we can make more informed decisions when approving new construction projects.

2 Modeling Crowd Dynamics

Given the set up of a theater, its dimensions, location of obstacles, location of exits, the initial position of the people in the theater, and the position of the terror (e.g. a weapon-wielding terrorist, a bomb device, a fire, or any other stimuli that people would be repulsed from), how long does it take everyone to get out of the theater?

2.1 The Metropolis Method

Human behavior is notoriously multifaceted and difficult to predict, so to make this problem more approachable, we need a model: A system that will represent complicated crowd dynamics in a more computationally manageable way. At this point, we turn to chemistry for inspiration, invoking a model that predicts how randomized particles might move toward a lower potential equilibria along a potential field. A highly relevant topic in molecular-simulation sectors is fluid dynamics: How can we simulate the behavior of liquids at various temperatures, pressures, and volumes? The solution to this issue is called "Hard Sphere Simulation", where every molecule is assumed to be a sphere (or, in two dimensions, a circle), and all spheres are considered "hard" in that they are not allowed to overlap. Spheres are assumed to have no interaction unless they're overlapping, in which case they have infinite potential. The simulation permutes all the particles in one time-step, via a symmetrical distribution on every axis. We will call this a "move." To make these simulations accurately simulate reality, an acceptance criterion is applied to decide whether to accept a "move." The acceptance criterion is the probability that the system moves from the old state ***o*** to the new state ***n***. This is calculated using the Maxwell-Boltzmann distribution, which takes a change in potential energy for a system, and returns a probability of that change:

$$x \sim Uni(0,1) \tag{1}$$

$$acc(o \to n) = e^{-\frac{U(n)-U(o)}{kT}} > x \tag{2}$$

The acceptance criterion is a Bernoulli random variable that will accept a new state ***n***, with potential energy ***U(n)***, according to the PDF of the Maxwell-Boltzmann distribution, at temperature ***T*** (mathematically, temperature is multiplied by the Boltzmann constant ***k***, so in this paper, we will treat ***kT*** as one parameter).

Combining all these components, the Metropolis algorithm is born [FS01]. See Algorithm 1.

Algorithm 1 This simple set of algorithms gives a high level appreciation for how simple the Metropolis algorithm can be. All that happens is X, the initial position of the particles, is moved n times, each particle in X given a random displacement according to a normal distribution with $\mu = 0$ and $\sigma = \sigma_x$. A move is accepted if the particles do not overlap, otherwise it is rejected.

```
1:  procedure METROPOLIS(n, X, kT, σ_x)
2:      U ← ∞
3:      for i=1:n do
4:          X, U ← move(X, en, kT, σ_x)
5:      return X, U
6:  procedure MOVE(X, en, kT, σ_x)
7:      X_n ← normrnd(μ = 0, σ = σ_x, n = len(X))
8:      U_n ← energy(X)
9:      if exp(−(U_n − U)/kT) > rand then
10:         return X_n, U_n
11:     return X, U
12: procedure ENERGY(X)
13:     for x_i in X do
14:         for x_j in X do
15:             if overlap(x_i, x_j) then
16:                 return ∞
17:     return 0
```

2.2 Extending to the Human Case

How do we connect the movement of particles in a flowing liquid to people in a movie theater? The key insight is that humans

are like particles. The ***kT*** value can be thought of as the level of panic in the theater: How likely people are to be confused and go the wrong way. The σ_x value is the distance people can move in a single time-step: The larger it is, the closer the simulation is to real time. There is, of course, a major item missing from the model we have so far described: There is no motivating force. Thus, the essential leap in logic is to conceive all the particles as being negatively charged. With this modification, the model prods the people to move towards the exit. Elaborating further, the exit will be simulated by a positive point charge and the terror by a negative point charge. Accordingly, the negatively charged people will have lower potential (i.e. be in a more favorable state) when they are closer to the exit and farther away from the terror object. A negatively charged particle will be at its lowest potential when closest to the positively charged particle.

This requires some minimal modification to the algorithms above. Namely, the energy function will now perform the following calculation–if ***e*** is the vector position of the exit and ***t*** is the vector position of the terror:

$$U = \sum_{x=X} \frac{kQ_t}{||x-t||} - \frac{kQ_e}{||x-e||} \tag{3}$$

With this substitution, the Maxwell-Boltzmann acceptance criterion (Equation 2) will over time force the simulation to its lowest potential equilibria. In layman's terms, the people will leave the theater. To make this model more realistic, we can account for in-room barriers like seats, railings, walls, and columns with some further modifications. Thus, we have an system that can effectively simulate any theater set-up.

2.3 Extending to Multiple Exits

Our model can be extended one step further. So far, it only allows for a theater to have one exit. If we added a second or a third exit, there is a logical incongruity: There would be an optimal point in-between the two exits, where moving in either direction would cause an increase in potential (i.e. be unfavorable) that the acceptance criterion is unlikely to accept. Think about how if an electron had a proton on either side of it at equal distances away, the electron (here, the theoretical person) would be immobile. Thus, our model system would likely achieve a steady state equilibrium without everyone evacuating the theater. Some theoretical people would be paralyzed between the two exits. This is not realistic. The simplest, most efficient solution to this problem is to say that people are "smart." They will know what exit is closest to them, and only experience the attractive force of that exit. This greatly simplifies the calculation for multiple exits, and elegantly solves the steady-state equilibria reality incongruity. To incorporate this adaption, we make a slight modification to the energy function. See Algorithm 2.

Algorithm 2 This iteration of the energy algorithm takes X, the current position of the people, B, the position of all barriers, E the position of all exits, t, the position of the terror, and l and w, the dimensions of the theater. The overlap functions are in reality specific to the shapes being compared and involve a bit of interesting geometry, but have been excluded here for succinctness.

```
procedure ENERGY(X, B, E, t, l, w)
    ▷ Check if any people are out of the bounds of the simulation
    if any_out_of_bounds(X, l, w) then
        return ∞
    for i ← 1 : len(X) do
        ▷ Check if person overlaps any barriers
        if any_overlap(X(i), B) then
            return ∞
        ▷ Check if person overlaps any people
        if any_overlap(X(i), X(i + 1 : end)) then
            return ∞
    U_n ← 0
    for x in X do
        ▷ Get closest exit, do potential calculation, and add to summation
        e ← get_closest_exit(x, E)
        U_n ← U_n + 1/||x−t|| − 1/||x−e||
    return U_n
```

2.4 Summary

In summary, to establish the safety-evaluation framework presented in this paper, we need the ability to simulate how people would flee a theater in response to a terrifying actor. To do so, we take inspiration from chemistry by using a heavily modified version of the Metropolis fluid dynamics algorithm. We arrive at a model that estimates TTE for a theater of any size, any number of people, and any amount of interior barriers.

3 Runtime Optimization

Even though this algorithm is theoretically fast, there are some factors that can create extensively prolonged computational run-times. Currently, our algorithm runs in about 13 seconds for our test theater design (described later). For industry and government usage, a shorter runtime would be optimal. The improved algorithm is for the number of people in the theater, as the energy calculation checks whether all people are in overlap with all other people. Runtime optimizations were implemented, specifically cell lists and parallelization, outlined within the supplemental appendix Section 8 at sub-header 8.3.

As Table 1 shows, these optimizations cut the runtime of a simulation by 37%.

Algorithm 2	Algorithm 3	Parallelized Algorithm 3
12.35 s	10.04 s	7.78 s

Table 1: These are the average runtime calculated after 100 simulations using each algorithm.

4 Score Functions

4.1 Minimum Simulations for Accurate Approximations

At this point, we have two more major problems to solve. These issues are parallel to each other, and the solution to the first is necessary to solve the second:

1. First: How to set the number of iterations $\boldsymbol{n}$ and the other simulation parameters, $\boldsymbol{kT}$ and $\boldsymbol{\sigma_{x'}}$ for optimal runtime.
2. Second: How to find the optimal exit position given a theater (dimensions, barriers, initial position of people).

On a deeper level, these problems are conceptually parallel. In the following two subsections, "Runtime Score" and "Exit Score," one will notice many similarities. After all, both problems involve running the simulation and getting an accurate, stable value (i.e. the runtime for problem 1, and the number of accepted iterations it takes for the theater to empty out for problem 2) from the output. Because the simulation is a Monte Carlo algorithm, the values it produces will vary as random variables. So the task is to figure out how few times the simulation can be run to get a result with at least a 95% certainty of being within 99% of the true expected value of the value of interest. We can apply a method developed by Psutka and Psutka [PP15] (the co-authors share the same last name) to test the accuracy of sampling from a function like this to see that an average of 8 samples will give the 95% certainty of being within 99% of the true expected value. The mathematical derivation of this "8" value is elucidated in our supplemental appendix Section 8 at sub-header 8.2.

4.2 Runtime Score

We must start by optimizing the amount of time the program itself takes to run. Having the minimum simulations, 8, for an accurate estimate allows us to write our objective function for runtime. This function is incredibly simple: Given a $\boldsymbol{kT}$ and $\boldsymbol{\sigma_{x'}}$ it will hold all other aspects of the simulation constant, run the simulation 8 times (each time collecting the runtime), fit a distribution with the maximum likelihood estimators, and finally, return the expected runtime from that distribution.

4.3 Exit Score

Similar to above, we can write an objective function for an exit vector. This is the most important part of the framework, as here, we determine how to best position the exits to minimize evacuation time, so that people can safely flee in an emergency. The public health benefits are self-explanatory: Saving lives is the ultimate good. Given a vector of exit positions $\boldsymbol{e}$, the function will hold all other aspects of the simulation constant, run the simulation 8 times (each time collecting the number of frames until the theater is empty), then fit a distribution with the maximum likelihood estimators, and finally, return the expected number from

that distribution. The output of this function is the Time To Exit (TTE) metric.

5 Optimization

Everything described up to this point has been independent of the theater being examined. Now, the theoretical algorithms described above will have to be run for each theater to optimize the ***n***, ***kT*** and $\boldsymbol{\sigma_x}$ for the given theater, and then, we will use these parameters to minimize the runtime. Subsequently, we optimize the exit positions and get the ideal TTE for a given theater. The main complication surrounding this real-life application is actually performing these optimizations. Unlike the above theoretical scenarios in Section 4 where it was relatively simple to derive a likelihood function, the objective functions defined in Sections 4.2 and 4.3 for runtime and exit score seem impossible to elucidate in this more reality-centric scenario. This means the framework requires a different type of optimization, gradient free optimization, which can work without being able to take the derivative of the factor being optimized. The framework utilizes two different forms of gradient free optimization, the Nelder-Mead method and Bayesian Optimization, which are used to optimize runtime and exit position, respectively. To delve into the heavy mathematics of this method, see supplement Section 8.4.

6 Results

To understand how our framework could be used in a regulatory setting to determine a theater design's safety rating (in terms of optimal exit placements to minimize TTE), we examine a simple theater as an example.

The theater shown in Figure 1 contains all the information the framework needs to evaluate the exit position: The dimensions of the theater, the positioning of all interior barriers, and an initial exit position (assumed to be the "reasonable" exit position necessary for the runtime optimization).

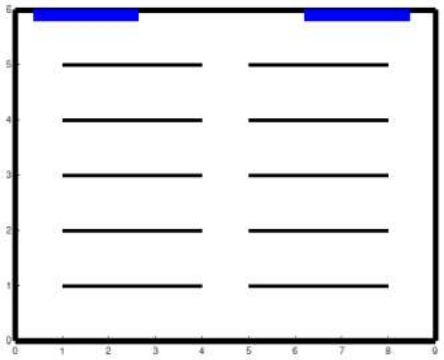

Figure 1. This shows our simple theater example: There are 8 rows each seating three people, with two exits at the back of the theater. It might repre-

6.1 Runtime Optimization

The framework starts by setting the maximum number of iterations for each simulation ***n***. It does this by simulating the initial exit position 8 times, fitting a distribution, and selecting a minimum number of iterations that will mean 95% of good simulations will complete. For this theater, that ***n=346055*** .

Pre-optimization	Post-optimization
$kT = 0.05, \sigma_x = 0.05$ 7.78 s	$kT = 0.0033, \sigma_x = 0.04$ 3.36 s

Table 2: These are the runtime scores with the initial and optimized values of $kT = 0.05$ and $\sigma_x = 0.05$.

Using this value of ***n***, the framework then optimizes runtime by running the simplex algorithm on the runtime score function. The initial simplex is set with ***kT = 0.05*** and $\boldsymbol{\sigma_x}$***=0.05.*** These initialization values were determined experimentally to promote good simplex convergence for this problem. The process of optimization takes approximately one hour, and cuts the simulation runtime by 57%.

6.2 Exit Position Optimization

The framework then moves on to optimizing the exit position. For an in-depth mathematical view of how this process works and how the Bayesian Optimization algorithm explores the space of possible exit positions, see supplemental Section 8.3. The final exit position the Bayesian process achieves is illustrated in Figure 2.

7 Conclusion

In conclusion, we examine how this framework can be integrated into a legislative setting, utilizing it to create regulations that address one of the greatest public health harms currently faced by the United States: Mass shootings and other similar terror events. Given the exit score function and the optimal exit position, we can use a easily understandable comparison fraction (below) to

evaluate the original design's exit positioning with respect to the optimal design's exit positioning, and subsequently, we can decide whether to proceed with construction. For our comparison fraction, we have:

$$\frac{TTE_i - TTE_o}{TTE_o} \leq \Delta \tag{4}$$

Where TTE_i is the TTE for the initially submitted plan, and TTE_o is the TTE for the optimal solution derived through the Bayesian Optimization. Δ is the acceptability threshold set by a regulatory body. A reasonable Δ might be 0.50, which would allow construction companies to at most reduce the safety score of a theater by 50% of the ideal (perhaps for cost-cutting or aesthetic reasons) and still move forwards with the project. In this case, the $TTE_i = 2730$, and the $TTE_o = 1261$, so the fraction evaluates to 1.165, which would likely be deemed an unacceptably high level of divergence from the room's optimally safe exit-placement design. Accordingly, we would give the architects this feedback, along with the optimal, computationally-derived exit positions. Subsequently, they could reevaluate their design to be better at keeping people safe from terror events.

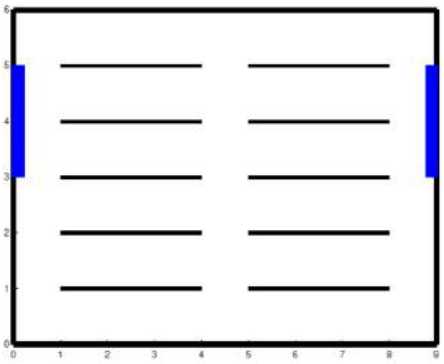

Figure 2: This is the optimal value of the exit position, given that the terror is at the front center of the theater. Logically, it makes sense: It skews to the back, further from where the terror is, but is otherwise reasonably symmetric which is congruous with the overall symmetry of our example theater. Also, notice that the exits in our optimized design are in-line with the seating rows, thereby alleviating the deleterious people bunching issues caused by moving the exits completely to the back wall.

Thus far, we have very specifically examined the case of optimizing exit position to minimize evacuation time, but the beauty of this framework is that it could be extended to adjust every quantifiable aspect about the design of the theater: The spacing of the seats, the angle of the rows of seats, the number and spacing of aisles, etc. These all can be evaluated, modeled, optimized, and re-evaluated to generate a TTE-based safety-score evaluation. Obviously, in this paper, we do not intend to belittle the training and expertise of professional architects: There are many economic, aesthetic, and safety factors that go into the design of buildings. However, now we have the ability to specifically and quantitatively assess whether a room is easy-to-exit in the case of danger, and perhaps more importantly, generate a modified room design that minimizes the "Time to Exit" using a rigorous, mathematics-based model. Considering the clear-cut nature of this TTE score regulatory test, incorporating our threshold into the construction-focused legislative branches seems like a rational step in building safer new buildings. By saving lives in crisis scenarios, the ultimate goal of any harm mitigation initiative, our TTE-based optimization of room construction is a great boon for the field of design-based public health.

8 Supplemental Mathematical Appendix

8.1 Runtime Optimization

8.1.1 Cell Lists

Cell lists are a way of breaking up the simulated space into equally sized blocks, and then only computing interactions between particles in adjacent cells. This means that especially for simulations with many particles, the number of required calculations is significantly lower. See Figure 3. The framework can make use of this idea, even though the situation is not exactly the same as the one Allen describes.[AT89]

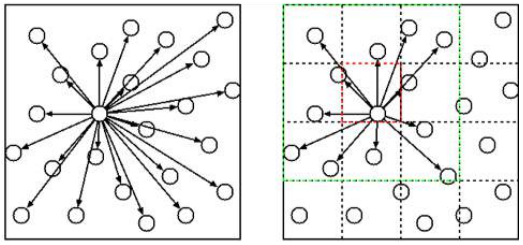

Figure 3: (Left) This diagram is what molecular dynamics seeks to simulate: The interaction between the particle of interest and all other particles. (Right) This shows what cell lists simulate: The interaction between the particle of interest, all particles in the same cell, and all particles in neighboring cells. [16]

The framework will similarly use cell lists to break up the simulated theater, but because our methodology assumes particles have no interaction with each other, it doesn't go as far the molecular dynamics cell list implementations. All our cell list strategy has to account for is a given particle's interactions with other "hard" objects in the the cell(s)–noting that particles can be in multiple cells–that the given particle is occupying.

There's a lot that can be pre-computed using this implementation. First, note that the barriers in the respective cells will not change: After all, barriers are intrinsically in a fixed position for a given room. Secondly, the cells that neighbor any given cell will be constant, so this can also be pre-computed. Finally, we can use pre-computation with regards to the static perimeter cells. The only variables within cell lists that will change are what people are occupying what cells. Refer to Algorithm 3 to see the implementation of the energy function using the framework's modified cell list strategy.

Algorithm 3 This iteration of the energy function is identical to Algorithm 2, except using a cell lists to limit the number of interaction calculations that must be done at every step.

```
1: procedure ENERGY(X, B, E, t, cells, l, w)
2:     ▷ Check if any people who were in perimeter cells before the move
3:     ▷ are now out of the bounds of the simulation
4:     if any_out_of_bounds(cells.perimeter_cells.X, l, w) then
5:         return ∞
6:     ▷ Update cells people are occupying
7:     for i ← 1 : len(X) do
8:         ▷ Get cells X_i is in, and neighbors of those cells
9:         possible ← cells.X(i).cells.neighbors
10:        ▷ Set cells X_i is in to cells it overlaps
11:        cells.X(i).cells ← overlap(X(i), possible)
12:    for c in cells do
13:        for i ← 1 : len(c.X) do
14:            ▷ Check if person in current cell overlaps barriers in cell
15:            if any_overlap(c.X(i), c.B) then
16:                return ∞
17:            ▷ Check if person in current cell overlaps people in cell
18:            if any_overlap(c.X(i), c.X(i + 1 : end)) then
19:                return ∞
20:    U_n ← 0
21:    for x in X do
22:        ▷ Get closest exit, do potential calculation, and add to summation
23:        e ← get_closest_exit(x, E)
24:        U_n ← U_n + 1/||x−t|| − 1/||x−e||
25:    return U_n
```

8.2 Minimum Simulations for Accurate Approximations

8.2.1 Determining Appropriate Distributions

The two variables of interest here are the runtime and the number of frames until the theater is empty. The first step to providing the accuracy above is understanding how these two variables vary.

From Figure 4, it is clear that both of these values vary as Log-Normal distributions.

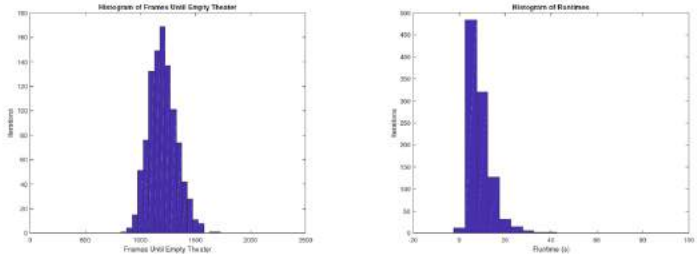

Figure 4: Running the simulation 1000 times, and taking these variables,

8.2.2 MLE for the Log-Normal Distribution

The way the framework is going to get these proven bounds on the value of interest is by fitting the Log-Normal distribution for some number of samples. The next step is to derive the $\boldsymbol{\mu}$ and $\boldsymbol{\sigma^2}$ that maximize the log-likelihood function of the Log-Normal distribution

$$X_i \sim LogNormal(\mu, \sigma^2) \tag{5}$$

$$f_{X_i}(x|\mu, \sigma^2) = \frac{1}{x\sqrt{2\pi\sigma^2}} e^{-\frac{(log(x)-\mu)^2}{2\sigma^2}} \tag{6}$$

First, the likelihood function, given $\boldsymbol{X=\{X_1, X_2 \dots X_n\}}$, is the product of the probability densities of each $\boldsymbol{X_i}$:

$$L(\mu, \sigma | X) = \prod_{i=1}^{n} f(X_i|\mu, \sigma^2) \tag{7}$$

$$= \prod_{i=1}^{n} \frac{1}{X_i\sqrt{2\pi\sigma^2}} e^{-\frac{(log(X_i)-\mu)^2}{2\sigma^2}} \tag{8}$$

$$= (2\pi\sigma^2)^{-\frac{n}{2}} \prod_{i=1}^{n} \frac{1}{X_i} e^{-\frac{(log(X_i)-\mu)^2}{2\sigma^2}} \tag{9}$$

Next, take the log of the Eq. (8) to get $\boldsymbol{L(\mu, \sigma^2 | X)}$:

$$\mathcal{L}(\mu, \sigma^2|X) = log\left((2\pi\sigma^2)^{-\frac{n}{2}} \prod_{i=1}^{n} \frac{1}{X_i} e^{-\frac{(log(X_i)-\mu)^2}{2\sigma^2}}\right) \tag{10}$$

$$= -\frac{n}{2} log(2\pi\sigma^2) - \sum_{i=1}^{n} \left[log(X_i) + \frac{(log(X_i)-\mu)^2}{2\sigma^2}\right] \tag{11}$$

$$= -\frac{n}{2} log(2\pi\sigma^2) - \sum_{i=1}^{n} \left[log(X_i) + \frac{log(X_i)^2 - 2log(X_i)\mu + \mu^2}{2\sigma^2}\right] \tag{12}$$

$$= -\frac{n}{2} log(2\pi\sigma^2) - \sum_{i=1}^{n} \left[log(X_i) + \frac{log(X_i)^2}{2\sigma^2} - \frac{log(X_i)\mu}{\sigma^2} + \frac{\mu^2}{2\sigma^2}\right] \tag{13}$$

$$= -\frac{n}{2} log(2\pi\sigma^2) - \frac{n\mu^2}{2\sigma^2} - \sum_{i=1}^{n} \left[log(X_i) + \frac{log(X_i)^2}{2\sigma^2} - \frac{log(X_i)\mu}{\sigma^2}\right] \tag{14}$$

Now taking the partials, first with respect to $\boldsymbol{\mu}$:

$$\frac{\partial \mathcal{L}}{\partial \mu} = -\frac{n\mu}{\sigma^2} \sum_{i=1}^{n} \frac{log(X_i)\mu}{\sigma^2} \tag{15}$$

Setting this to zero and solving, then:

$$0 = -\frac{n\hat{\mu}}{\hat{\sigma}^2} + \sum_{i=1}^{n} \frac{log(X_i)}{\hat{\sigma}^2} \tag{16}$$

$$n\hat{\mu} = \sum_{i=1}^{n} log(X_i) \tag{17}$$

$$\hat{\mu} = \frac{\sum_{i=1}^{n} log(X_i)}{n} \tag{18}$$

Next taking the partial with respect to $\boldsymbol{\sigma^2}$:

$$\frac{\partial \mathcal{L}}{\partial \sigma^2} = -\frac{n}{2\sigma^2} - \sum_{i=1}^{n} \frac{(log(X_i)-\mu)^2}{2(\sigma^2)^2} \tag{19}$$

Setting this to zero and solving, then:

$$0 = -\frac{n}{2\hat{\sigma}^2} + \sum_{i=1}^{n} \frac{(log(X_i)-\hat{\mu})^2}{2(\hat{\sigma}^2)^2} \tag{20}$$

$$\frac{n}{2\hat{\sigma}^2} = \sum_{i=1}^{n} \frac{(log(X_i)-\hat{\mu})^2}{2(\hat{\sigma}^2)^2} \tag{21}$$

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{n} (log(X_i)-\hat{\mu})^2}{n} \tag{22}$$

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{n} (log(X_i) - \frac{\sum_{i=1}^{n} log(X_i)}{n})^2}{n} \tag{23}$$

So the maximum likelihood estimators for the Log-Normal distribution are:

$$\hat{\mu} = \frac{\sum_{i=1}^{n} log(X_i)}{n} \tag{24}$$

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{n} (log(X_i) - \frac{\sum_{i=1}^{n} log(X_i)}{n})^2}{n} \tag{25}$$

8.2.3 Minimum Samples for Accurate Log-Normal MLE

Using the methodology developed by Psutka and Psutka [PP15] for the normal distribution, lower bounds can be derived for the number of samples for a given certainty of a given confidence of accurate maximum likelihood estimates. It is relatively easy to adapt their methods to the Log-Normal distribution.

Expected Log-Likelihood

First, we need a way to evaluate the accuracy of a likelihood estimate. The best way to evaluate this accuracy would be to ask: Given the true parameters of and , what would the value of the log-likelihood function be? In short, the answer is the expected log-likelihood function. The expected log-likelihood function is defined as:

$$E[\mathcal{L}] = \lim_{N\to\infty} \left(\frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\mu, \sigma^2 | X)\right) \tag{26}$$

$$= \lim_{N\to\infty} \left(\frac{1}{N} \sum_{i=1}^{N} \left(-\frac{n}{2} log(2\pi\sigma^2) - \frac{n\mu^2}{2\sigma^2} - \sum_{i=1}^{n} \left[log(X_i) + \frac{log(X_i)^2}{2\sigma^2} - \frac{log(X_i)\mu}{\sigma^2}\right]\right)\right) \tag{27}$$

$$= -\frac{n}{2} log(2\pi\sigma^2) - \frac{n\mu^2}{2\sigma^2} - \lim_{N\to\infty} \left(\frac{1}{N} \sum_{i=1}^{N} \left(\sum_{i=1}^{n} \left[log(X_i) + \frac{log(X_i)^2}{2\sigma^2} - \frac{log(X_i)\mu}{\sigma^2}\right]\right)\right) \tag{28}$$

$$= -\frac{n}{2} log(2\pi\sigma^2) - \frac{n\mu^2}{2\sigma^2} - \sum_{i=1}^{n} \left[\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} log(X_i) + \frac{\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} log(X_i)^2}{2\sigma^2} - \frac{\mu \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} log(X_i)}{\sigma^2}\right] \tag{29}$$

$$= -\frac{n}{2} log(2\pi\sigma^2) - \frac{n\mu^2}{2\sigma^2} - \sum_{i=1}^{n} \left[E[log(X_i)] + \frac{E[log(X_i)^2]}{2\sigma^2} - \frac{\mu E[log(X_i)]}{\sigma^2}\right] \tag{30}$$

Now with the expected log-likelihood in terms of the expectation of various functions of , those expectations must be solved. Starting with the expectation of $log(X_i)$, which is trivial, by the definition of the Log-Normal distribution:

$$log(X_i) = Y \qquad \sim N(\mu, \sigma^2) \tag{31}$$

$$E[log(X_i)] = E[Y] \qquad = \mu \tag{32}$$

Similar logic is applicable to the derivation of the expectation of $log(X_i)^2$:

$$Var(Y) = E[Y^2] - (E[Y])^2 \tag{33}$$

$$E[Y^2] = Var(Y) + (E[Y])^2 \tag{34}$$

$$E[Y^2] = \sigma^2 + \mu^2 \tag{35}$$

$$E[log(X_i)^2] = \sigma^2 + \mu^2 \tag{36}$$

Returning to the expected log-likelihood function:

$$E[\mathcal{L}] = -\frac{n}{2}log(2\pi\sigma^2) - \frac{n\mu^2}{2\sigma^2} - \sum_{i=1}^{n}\left[E[log(X_i)] + \frac{E[log(X_i)^2]}{2\sigma^2} - \frac{\mu E[log(X_i)]}{\sigma^2}\right] \quad (37)$$

$$= -\frac{n}{2}log(2\pi\sigma^2) - \frac{n\mu^2}{2\sigma^2} - \sum_{i=1}^{n}\left[\mu + \frac{\sigma^2+\mu^2}{2\sigma^2} - \frac{\mu^2}{\sigma^2}\right] \quad (38)$$

$$= -\frac{n}{2}log(2\pi\sigma^2) - \frac{n\mu^2}{2\sigma^2} - n\mu - \frac{n(\sigma^2-\mu^2)}{2\sigma^2} + \frac{n\mu^2}{\sigma^2} \quad (39)$$

$$= \frac{n}{2}\left(\frac{2\mu^2}{\sigma^2} - 2\mu - log(2\pi\sigma^2) - 1\right) \quad (40)$$

Evaluating Likelihood Accuracy

Now setting aside for a moment this expected value for log-likelihood, the fraction below will motivate a means for evaluating the accuracy of any estimated $\hat{\mu}$ and $\hat{\sigma}^2$:

$$\frac{L(\hat{\mu}, \hat{\sigma}^2|X)}{E[L(\mu, \sigma^2|X)]} \geq \beta \quad (41)$$

This fraction puts the likelihood using the maximum likelihood estimators for the Log-Normal over the expected value of the log-likelihood function, given the true $\boldsymbol{\mu}$ and $\boldsymbol{\sigma^2}$. This is useful, because we now have a fraction that will range between 0 and 1, but that will asymptotically approach 1 as the number of samples n used in the MLE of$\boldsymbol{\mu}$ and $\boldsymbol{\sigma^2}$ pproaches infinity. This means the value $\boldsymbol{\beta}$ is a proxy for estimation accuracy. It is this value we will set to ensure we get within 99% of the true expected value of the value of interest.

Taking the log of Eq. (40) and rearranging so that we have a less than inequality we have:

$$log\left(\frac{L(\hat{\mu}, \hat{\sigma}^2|X)}{E[L(\mu, \sigma^2|X)]}\right) \geq log(\beta) \quad (42)$$

$$\mathcal{L}(\hat{\mu}, \hat{\sigma}^2|X) - E[\mathcal{L}(\mu, \sigma^2|X)] \geq log(\beta) \quad (43)$$

$$E[\mathcal{L}(\mu, \sigma^2|X)] - \mathcal{L}(\hat{\mu}, \hat{\sigma}^2|X) \leq -log(\beta) \quad (44)$$

Minimum Samples for Accurate MLE

We will start by defining $\hat{\mu}$ and $\hat{\sigma}^2$ as the estimates resulting from i random variables in $\boldsymbol{Y = \{Y_1, Y_2, \ldots Y_n\}}$. We will also define $\boldsymbol{Z = \{Z_1, Z_2, \ldots Z_n\}}$ where $\boldsymbol{k}$ will be a large integer. $\boldsymbol{Y}$ and $\boldsymbol{Z}$ are drawn from the same distribution with parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma^2}$. This allows us to define:

$$i^*_\beta = \min_i\{E[\mathcal{L}(\mu, \sigma^2|Z)] - \mathcal{L}(\hat{\mu}_i, \hat{\sigma}_i^2|Z) \leq -log(\beta)\} \quad (45)$$

The estimate $\boldsymbol{i^*_\beta}$ is by simulating the above procedure. This can be seen in Algorithm 4.

Algorithm 4 This algorithm takes a two large integers, t and k, the number of iterations and the size of the test data vector respectively, and an accuracy β. It returns the minimum number of iterations required to achieve the given accuracy for each of the t simulations.

procedure NSAMPLES(t, k, β)
  ▷ Initialize i^*_β as a vector of 2's
  $i^*_\beta \leftarrow twos(t, 1)$;
  for $i \leftarrow 1 : t$ **do**
    ▷ Generate two random parameters μ and σ
    $\mu \leftarrow 100 * rand$
    $\sigma \leftarrow rand$
    ▷ Draw k samples from the lognormal with parameters μ and σ
    $Z \leftarrow lognrnd(\mu, \sigma, k, 1)$
    ▷ Repeat the loop until the desired accuracy is reached
    while $E[\mathcal{L}(\mu, \sigma^2|Z)] - \mathcal{L}(\hat{\mu}_i, \hat{\sigma}^2_i|Z) \leq -log(\beta)$ **do**
      ▷ Draw $i^*_\beta(i)$ samples from the lognormal with parameters μ and σ
      $Y \leftarrow lognrnd(mu, \sigma, i^*_\beta(i), 1)$
      ▷ Estimate μ and σ with $\hat{\mu}$ and $\hat{\sigma}$
      $\hat{\mu} \leftarrow mean(log(Y))$
      $\hat{\sigma} \leftarrow \sqrt{var(log(Y))}$
      $i^*_\beta(i) \leftarrow i^*_\beta(i) + 1$
  return i^*_β

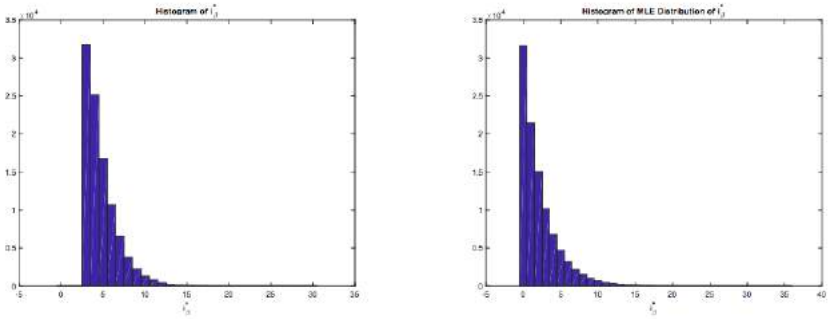

Figure 5: (Left) This is a histogram of i^*_β; the result of running Algorithm 4 with ***t=100000***, ***k=100000***, ***β=0.99***. (Right) This is the histogram of 100,000 values drawn from the distribution fit to, using $\hat{p}$.

It is clear from Figure 5 that ***i*$_\beta$ - 3: Geo(p)***. This also makes sense. There is some probability that the MLE estimate fits the distribution well with only ***i*** samples: Each iteration we increase the number of samples, running the same experiment and stopping after getting a success. A geometric random variable is defined as the number of experiments before a success, so it makes sense that it would accurately capture this phenomenon. Here, too, we need to fit this distribution to get the value of ***p***. It is therefore necessary to derive the maximum likelihood estimator for the Geometric distribution:

$$L(p|X) = \prod_{i=1}^{n}(1-p)^{X_i-1}p \tag{46}$$

$$L(p|X) = p^n(1-p)^{\sum_{i=1}^{n} X_i - n} \tag{47}$$

$$\mathcal{L}(p|X) = nlog(p) + (\sum_{i=1}^{n} X_i - n)log(1-p) \tag{48}$$

$$\frac{\partial \mathcal{L}}{\partial p} = \frac{n}{p} - \frac{\sum_{i=1}^{n} X_i - n}{1-p} \tag{49}$$

Setting the partial to zero, and solving for $\hat{p}$:

$$0 = \frac{n}{\hat{p}} - \frac{\sum_{i=1}^{n} X_i - n}{1 - \hat{p}} \tag{50}$$

$$\hat{p} = \frac{n}{\sum_{i=1}^{n} X_i} \tag{51}$$

This estimates the distribution of i^*_β - 3 very well, as is clear from Figure 5, meaning that $\hat{p}$ =0.5719 is a good estimation. Using this value, it is now possible to derive the minimum number of samples that will guarantee 95% certainty of being within 99% of the true expected value:

$$P(i_\beta \leq i_{min}) \geq 0.95 \tag{52}$$

$$1 - (1-p)^{i_{min}-1} \geq 0.95 \tag{53}$$

$$(1-p)^{i_{min}-1} \geq \frac{1}{20} \tag{54}$$

$$i_{min} = log(\frac{1-p}{20} + 1 - p) \tag{55}$$

$$= 4.531 \tag{56}$$

This value is, of course, 3 short of the actual value, so adding three and rounding up, we conclude that 8 samples is the minimum required to have 95% certainty of our MLE estimates being 99% accurate to the true distribution.

8.3 Optimization

8.3.1 Runtime

There are several assumptions that underpin the optimization of runtime. First, the optimal values for , , and are at least relatively robust if not entirely independent of the exit position. Were this not true, it would be impossible to optimize these runtime variables. The second assumption is that we have a "reasonable" solution with which to do the optimization. This will likely be a human selected exit vector that does have to be completely optimal, but cannot be atrociously bad (there will always be some exit positions where people never leave, or would take so long to escape that it isn't worth the simulation time). This is necessary, or else, it would be relatively impossible to do the runtime optimization.

Setting the number of iterations

The first value to set is , because it will not require any gradient free optimization. It is, at this point, important to draw the distinction between this value, the runtime, and the frames until we have an empty theater. The runtime is the number of real-time seconds the algorithm requires to run. The frames until we have an empty theater is the number of accepted moves before all people have evacuated the theater. The number of iterations is the total number of moves attempted, both accepted and rejected. The value is the maximum number of iterations before the simulation stops. This cutoff is necessary especially when dealing with non-"reasonable" solutions.

How will the framework optimize ? It will use the now-familiar technique outlined throughout Section 4: Eight runs will be conducted and the total number of moves required before all people escape will be collected. The iterations vary as a Log-Normal, the same as runtime and frames. A distribution will be fit to this data using MLE.

Nelder-Mead

The Nelder-Mead method attempts to find a global minimum for some real-valued function over some number of variables. In this case, it will be minimizing the runtime score function with respect to and . For higher dimensional problems Nelder-Mead never really converges, but for the simple case of 2 variable optimization (assuming a mono-modal function), it is a very efficient solution.[NM65]

8.3.2 Exit Position

For optimizing exit position, the Nelder-Mead algorithm will not work. If we are optimizing exit position for a stadium that holds thousands or tens of thousands, then there will be dozens of exits and Nelder-Mead will never converge. Even for a small number of exits, it is very unlikely exit position is a mono-modal surface: Consider a simple, symmetrical theater with two exits– the exit score for this function is at least bi-modal, as at the optimal positions, the exits can be flipped to achieve the same peak at a different input point.

This means the framework must use some other gradient-free optimization algorithm, preferably one that requires minimal iterations, as the exit score is computationally expensive to evaluate. This leads us to Bayesian Optimization.

The way Bayesian Optimization works is by treating the objective function as a random function (note: the exit score is a random function). If the objective is a random function, then we can place a prior over the domain of the random function. This prior is designed to capture our belief about the behavior of the function. The function is then evaluated for some number of initial randomly chosen points, and these points are used to update the prior in Bayesian fashion, generating a posterior distribution. This posterior distribution is then used to maximize the acquisition function in order to select the next point to evaluate.[SLA12]

References

[NM65] John A. Nelder and Roger Mead. "A simplex method for function minimization". In: Computer Journal 7 (1965), pp. 308–313.

[AT89] M. P. Allen and D. J. Tildesley. Computer Simulation of Liquids. New York, NY, USA: Clarendon Press, 1989. isbn: 0-19-855645-4.

[FS01] Daan Frenkel and Berend Smit. Understanding Molecular Simulation. 2nd. Orlando, FL, USA: Academic Press, Inc., 2001. isbn: 0122673514.

[SLA12] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Prac- tical Bayesian Optimization of Machine Learning Algo- rithms". In: Advances in Neural Information Processing Systems 25. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 2951–2959. url:http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf.

[PP15] Josef V. Psutka and Josef Psutka. "Sample Size for Maxi- mum Likelihood Estimates of Gaussian Model". In: Com- puter Analysis of Images and Patterns Lecture Notes in Computer Science (2015), pp. 462–469. doi: 10 . 1007 / 978-3-319-23117-4_40

[16] Cell lists. Nov. 2016. url: https://en.wikipedia.org/ wiki/Cell_lists.

[Cha17] Mona Chalabi. Not your imagination: mass shootings now happen more frequently in the US. Nov. 2017. url: https://www.theguardian.com/us- news/datablog/2017/nov/06/not-your-imagination-mass-shootings-now-happen-more-frequently-in-the-us.

[Kha17] Olga Khazan. Why Better Mental-Health Care Won't Stop Mass Shootings. Oct. 2017. url: https://www.theatlantic.com/health/archive/2017/10/why- better- mental-health-care-wont-stop-mass-shootings/541965/.