Corrupted Blood: A Virtual Epidemic in World of Warcraft

***Abstract*—In this paper, we explore the application of Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) Algorithms to mitigate disease spread within virtual environments, using the infamous Corrupted Blood plague incident in World of Warcraft (WoW) as a case study. We provide a detailed overview of the pandemic's timeline, the characteristics of the Corrupted Blood spell, player behaviors, and the evolution of player health points and damage taken.**

For the GA application, we approximate player behaviors and model the virtual environment using a matrix-based map. We implement the genetic algorithm to simulate player movements, infection, and quarantine efforts, leading to the identification of optimal strategies to slow down the disease spread.

In the PSO application, we define problem parameters and a fitness function to evaluate different solutions. The PSO algorithm optimizes these parameters to minimize the number of infected players over iterations. Through simulation results, we demonstrate the effectiveness of both GA and PSO algorithms in controlling disease spread within virtual environments.

Overall, our study highlights the versatility of evolutionary algorithms in modeling complex systems and providing insights into disease dynamics within virtual worlds.

***Index Terms*—Algorithms, Corrupted Blood, Disease Outbreak, Genetic Algorithms, Optimization, Particle Swarm Optimization, Virtual Environments, World of Warcraft.**

I. Introduction

A. Quick Overview [1]

- **September 13, 2005:** World of Warcraft releases Patch 1.7, opening Zul'Gurub, where the Corrupted Blood plague is triggered.
- **September 16, 2005:** The first report of the plague is made on the WoW forum.
- **September 16-17, 2005:** Blizzard implements quarantines to stop the spread of the disease.
- **September 18, 2005:** Gaming blogs begin reporting on the plague.
- **September 22, 2005:** The BBC News reports on the plague.
- **October 5, 2005:** NPR reports on how the plague is causing scientists to consider how the virtual world can provide clues on what people would do in real pandemics.

Fig. 1: *Image of players afflicted by the Corrupted Blood disease.*

B. The spell: Corrupted Blood (2005) [2]

- **Effect:** Inflicts 263 to 337 damage every 2 seconds and spreads to nearby allies.
- **Duration:** 4 seconds
- **Range:** 100 yards
- **Schools:** Physical damage
- **Cooldown:** None
- **Periodic trigger:** Every 2 seconds
- **Classes affected:** Not specified

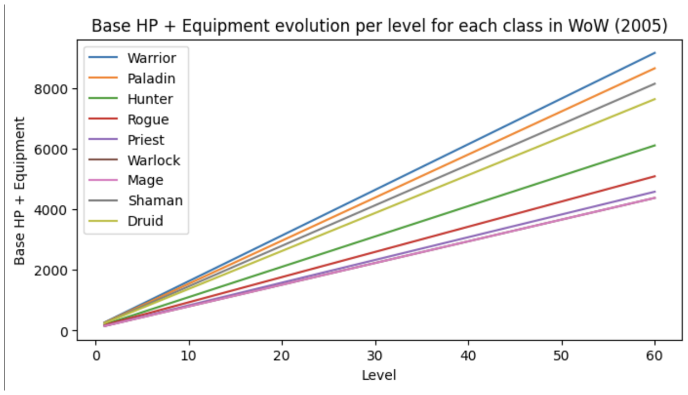
C. Data of the pandemic [3]

- About **4 million players** encountered the epidemic in WoW in September 2005.
- The **R0** of the disease in capital cities and transport hubs was around **10^2 per hour.**
- Quarantine efforts failed due to its high contagiousness and player resistance.
- Engaging in **PvP** in non-contested areas immunizes against the plague, but only from non-PvP flagged players.
- It spreads from **NPC to NPC**, **NPC to player**, and **player to NPC**, regardless of gender, race, class, level, or faction.

D. Players behaviours [3]

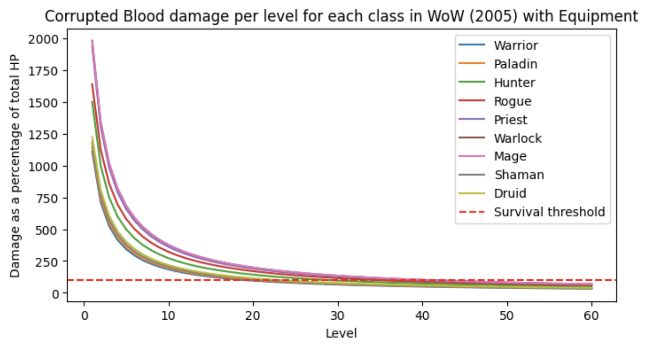
- Some players acted as **first responders**.
- **Pets** were used as **vectors** for the disease.
- **Teleportation** allowed the disease to spread over **long distances**.
- Some players avoided urban areas or left the game.
- **Non-player characters** acted as **asymptomatic carriers**.
- Players **resurrected** weaker characters, keeping a susceptible population present.
- Players resisted the idea of quarantine.
- Players were emotionally invested in the game.

E. Evolution of Healths Points for each class [4] [5]



In 2005, the health points (HP) for each class in World of Warcraft (WoW) evolved significantly with equipment, ranging from an average of 80 HP to 9000 HP.

F. Evolution of Damage taken for each class [4] [5]



Based on our assumption of a player being infected five times consecutively, the average survival threshold level across the nine classes listed is approximately 30.22.

II. Geneticals Algorithms Application [6]

A. Approximation

For simplicity reasons, we have decided to make some approximations.

- Class distribution: Classes are generated randomly with equal probability $\frac{1}{9}$.
- When individuals heal themselves or others, they restore the entire health.
- Below these levels, players automatically die if they are infected:

```
dict_lvl_die = {
    'warrior': 19,
    'paladin': 20,
    'hunter': 28,
    'rogue': 34,
    'priest': 38,
    'warlock': 40,
    'mage': 40,
    'shaman': 21,
    'druid': 23
}
```

B. Model

1) Player: .

We will work with 100 players, 4 of whom are the initiators of the virus. We will also work with 8 NPCs who can catch the virus and become asymptomatic carriers.

To implement the players, we use a genotype containing 8 genes ranging from 0 to 7.

- **genotype[0]:** Tendency to flee from an infected area (0 = low to 1 = high)
- **genotype[1]:** Tendency to help other players (0 = low to 1 = high)
- **genotype[2]:** Tendency to spread the virus (0 = high to 1 = low)
- **genotype[3]:** Tendency to explore infected areas (0 = low, 1 = high)
- **genotype[4]:** Preference for self-treatment (0 = low, 1 = high)
- **genotype[5]:** Virus resistance/level (0 = low, 0.6 = high)
- **genotype[6]:** Movement speed (0 = slow, 1 = fast)
- **genotype[7]:** Location (0 = move towards populated areas to 1 = move towards deserted areas)

There are 7 classes, each with different levels of health and a position in a map.

```
import random

class Player:
    def __init__(self, genotype_length):
        self.genotype = [round(random.random(), 2)
    for _ in range(genotype_length)]
        # genotype[5]= 0    0.6
        self.genotype[5] = round(random.random() *
    0.6, 2)
        self.location = None
        self.infected = False

        self.classplayer = classPlayer[random.
    randint(0, 8)]
        self.health = self.calculate_health()
```

2) Map: .

We have decided to use a matrix to model the World of Warcraft map. At each step, we will see the players move around it.

```
class WoWMap:
    def __init__(self, num_zones, zone_size):
        self.num_zones = num_zones
        self.zone_size = zone_size
        self.zones = []
        self.grid = [[0 for _ in range(num_zones *
    zone_size)] for _ in range(num_zones * zone_size
    )]

        for i in range(num_zones):
            for j in range(num_zones):
                zone_center = (i * zone_size +
    zone_size // 2, j * zone_size + zone_size // 2)
                self.zones.append(Zone(zone_center,
    zone_size))
```

C. Implementation of the genetic algorithm

1) First and Second Steps: .

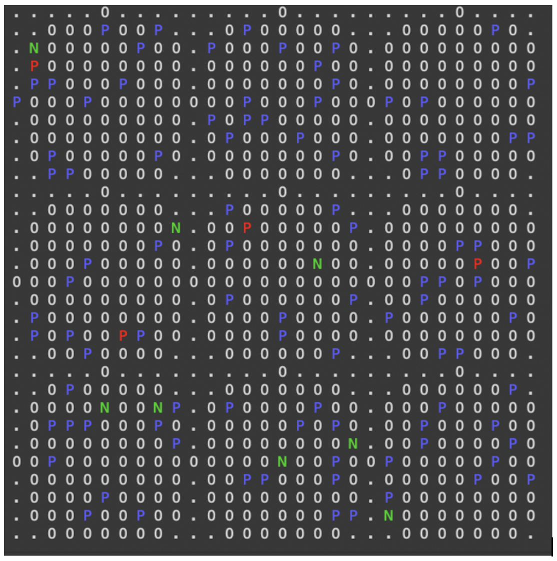
We will work on stages.
At each stage, an infected person can:

- Either lose HP and die if their health is below 0.
- Or die because they have not reached the required level.

The Paladin, Druid, Shaman, and Priest classes can heal themselves and others. And people can move around the map.

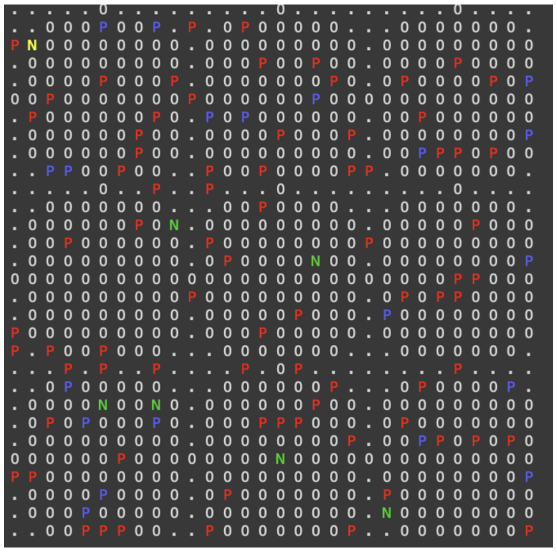
Here is the map with the different players and the NPCs.

- P in red means it is an infected player,
- P in white means it is a non-infected player,
- N in green means it is a non-infected NPC,
- N in yellow means it is an infected NPC.



```
. . . . . 0 . . . . . . . . . 0 . . . . . . . . . 0 . . . .
. . 0 0 0 P 0 0 P . . . 0 P 0 0 0 0 0 . . . 0 0 0 0 0 P 0 .
. N 0 0 0 0 0 P 0 0 . P 0 0 0 P 0 0 P 0 . 0 0 0 0 0 0 0 0 0
. P 0 0 0 0 0 0 0 0 . 0 0 0 0 0 0 P 0 0 . 0 0 0 0 0 0 0 0 0
. P P 0 0 0 P 0 0 0 . 0 0 0 0 0 0 0 P 0 . 0 0 0 0 0 0 0 0 P
P 0 0 0 P 0 0 0 0 0 0 0 0 P 0 0 0 P 0 0 0 P 0 P 0 0 0 0 0 0
. 0 0 0 0 0 0 0 0 0 . P 0 P P 0 0 0 0 0 . 0 0 0 0 0 0 0 0 0
. 0 0 0 0 0 0 0 0 0 . 0 P 0 0 0 P 0 0 0 . 0 0 0 0 0 0 0 P P
. 0 P 0 0 0 0 0 P 0 . 0 0 0 0 0 0 0 P 0 . 0 0 P P 0 0 0 0 0
. . P P 0 0 0 0 0 . . . 0 0 0 0 0 0 0 . . . 0 P P 0 0 0 0 .
. . . . . 0 . . . . . . . . . 0 . . . . . . . . . 0 . . . .
. . 0 0 0 0 0 0 0 . . . P 0 0 0 0 0 P . . . 0 0 0 0 0 0 0 .
. 0 0 0 0 0 0 0 0 N . 0 0 P 0 0 0 0 0 P . 0 0 0 0 0 0 0 0 0
. 0 0 0 0 0 0 0 P 0 . 0 P 0 0 0 0 0 0 0 . 0 0 0 0 P P 0 0 0
. 0 0 0 P 0 0 0 0 0 . 0 0 0 0 0 0 N 0 0 . 0 0 0 0 0 P 0 0 P
0 0 0 P 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 P P 0 P 0 0 0
. 0 0 0 0 0 0 0 0 0 . 0 P 0 0 0 0 0 0 P . 0 0 P 0 0 0 0 0 0
. P 0 0 0 0 0 0 0 0 . 0 0 0 0 P 0 0 0 0 . P 0 0 0 0 0 0 P 0
. P 0 P 0 0 P P 0 0 . 0 0 0 0 P 0 0 0 0 . 0 0 0 0 0 0 0 0 0
. . 0 0 P 0 0 0 0 . . . 0 0 0 0 0 0 P . . . 0 0 P P 0 0 0 .
. . . . . 0 . . . . . . . . . 0 . . . . . . . . . 0 . . . .
. . 0 P 0 0 0 0 0 . . . 0 0 0 0 0 0 0 . . . 0 0 0 0 0 0 P .
. 0 0 0 0 N 0 0 N P . 0 P 0 0 0 0 P 0 0 . 0 0 0 P 0 0 0 0 0
. 0 P P P 0 0 0 P 0 . 0 0 0 0 0 P 0 P 0 . 0 0 P 0 0 0 P 0 0
. 0 0 0 0 0 0 0 0 P . 0 0 0 0 0 0 0 0 N . 0 0 P 0 0 0 0 P 0
0 0 P 0 0 0 0 0 0 0 0 0 0 0 0 N 0 0 P 0 0 P 0 0 0 0 0 P 0 0
. 0 0 0 0 0 0 0 0 0 . 0 0 P P 0 0 0 P 0 . 0 0 0 0 0 P 0 0 P
. 0 0 0 0 P 0 0 0 0 . 0 0 0 0 0 0 0 0 0 . P 0 0 0 0 0 0 0 0
. 0 0 0 P 0 0 P 0 0 . 0 0 0 0 0 0 0 P P . N 0 0 0 0 0 0 0 0
. . 0 0 0 0 0 0 0 . . . 0 0 0 0 0 0 0 . . . 0 0 0 0 0 0 0 .
```

Fig. 2: *Map after the beginning*



```
. . . . . 0 . . . . . . . . . 0 . . . . . . . . . 0 . . . .
. . 0 0 0 P 0 0 P . P . 0 P 0 0 0 0 0 . . . 0 0 0 0 0 0 0 .
P N 0 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0 0
. 0 0 0 0 0 0 0 0 0 . 0 0 0 P 0 0 P 0 0 . 0 0 0 0 P 0 0 0 0
. 0 0 0 0 P 0 0 0 P . 0 0 0 0 0 0 0 P 0 . 0 P 0 0 0 0 P 0 P
0 0 P 0 0 0 0 0 0 0 P 0 0 0 0 0 0 P 0 0 0 0 0 0 0 0 0 0 0 0
. P 0 0 0 0 0 0 P 0 . P 0 P 0 0 0 0 0 0 . 0 0 P 0 0 0 0 0 0
. 0 0 0 0 0 0 P 0 0 . 0 0 0 0 P 0 0 0 P . 0 0 0 0 0 0 0 0 P
. 0 0 0 0 0 0 P 0 0 . 0 0 0 0 0 0 0 0 0 . 0 0 P P P 0 P 0 0
. . P P 0 0 P 0 0 . . P 0 0 P 0 0 0 0 P P . 0 0 0 0 0 0 0 .
. . . . . 0 . . P . . P . . . 0 . . . . . . . . . 0 . . . .
. . 0 0 0 0 0 0 0 . . . 0 0 P 0 0 0 0 . . . 0 0 0 0 0 0 0 .
. 0 0 0 0 0 0 P 0 N . 0 0 0 0 0 0 0 0 0 . 0 0 0 0 0 P 0 0 0
. 0 0 P 0 0 0 0 0 0 . P 0 0 0 0 0 0 0 0 P 0 0 0 0 0 0 0 0 0
. 0 0 0 0 0 0 0 0 0 . 0 P 0 0 0 0 N 0 0 . 0 0 0 0 0 0 0 0 P
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 P P 0 0 0
. 0 0 0 0 0 0 0 0 0 P 0 0 0 0 0 0 0 0 0 . 0 P 0 P P 0 0 0 0
. 0 0 0 0 0 0 0 0 0 . 0 0 0 0 0 P 0 0 0 . P 0 0 0 0 0 0 0 0
P 0 0 0 0 0 0 0 0 0 . 0 0 0 P 0 0 0 0 0 . 0 0 0 0 0 0 0 0 0
P . P 0 0 P 0 0 0 . . . 0 0 0 0 0 0 0 . . . 0 0 0 0 0 0 0 .
. . . P . P . . P . . . . P . 0 P . . . . . . . . P . . . .
. . 0 P 0 0 0 0 0 . . . 0 0 0 0 0 0 P . . . 0 P 0 0 0 0 P .
. 0 0 0 0 N 0 0 N 0 . 0 0 0 0 0 0 P 0 0 . 0 0 0 0 0 0 0 0 0
. 0 P 0 P 0 0 0 P 0 . 0 0 0 P P P 0 0 0 . 0 P 0 0 0 0 0 0 0
. 0 0 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0 P . 0 0 P P 0 P 0 P 0
0 0 0 0 0 0 P 0 0 0 0 0 0 0 0 N 0 0 0 0 0 0 0 0 0 0 0 0 0 0
P P 0 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0 P
. 0 0 0 0 P 0 0 0 0 . 0 P 0 0 0 0 0 0 0 . P 0 0 0 0 0 0 0 0
. 0 0 0 P 0 0 0 0 0 . 0 0 0 0 0 0 0 0 0 . N 0 0 0 0 0 0 0 0
. . 0 0 P P P 0 0 . . P 0 0 0 0 0 0 0 P . . 0 0 0 0 0 0 0 P
```

Fig. 3: *Map after the first step*

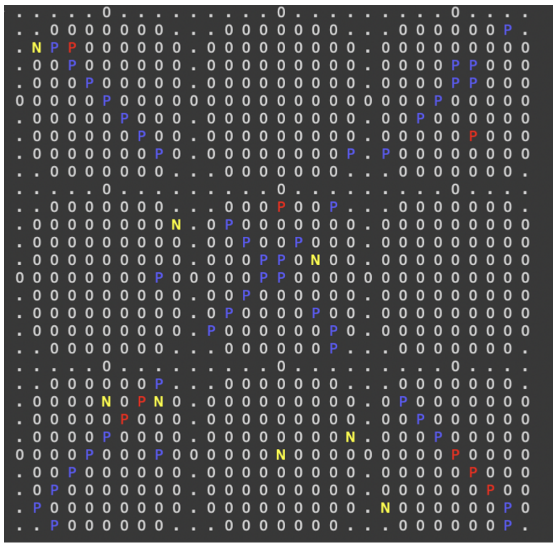
2) *Several Steps:* .

After a certain number of steps, we will leverage genetic algorithm methods.
Firstly, we will sort the population according to the fitness function, which will sum the elements of a genotype.
Then, we will create three groups within this sorting.

- The first group will be the top of the sorted list, which we will keep as it is.
- The second group will undergo a **two-point crossover**.
- Finally, the third group will undergo a **mutation** where each gene has a probability of 0.2 of being modified.

We fill the population with random individuals, and then we repeat the cycle to obtain a population where there are few or no infected individuals, thus reducing the spread of the virus.

After 100 sets of 20 steps and evolutions, we end up with a population where there are not too many infected individuals: the infection has been slowed down.



```
. . . . . 0 . . . . . . . . . 0 . . . . . . . . . 0 . . . .
. . 0 0 0 0 0 0 0 . . . 0 0 0 0 0 0 0 . . . 0 0 0 0 0 0 P .
. N P P 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0 0
. 0 0 P 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0 0 . 0 0 0 0 P P 0 0 0
. 0 0 0 P 0 0 0 0 0 . 0 0 0 0 0 0 0 0 0 . 0 0 0 0 P P 0 0 0
0 0 0 0 0 P 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 P 0 0 0 0 0
. 0 0 0 0 0 P 0 0 0 . 0 0 0 0 0 0 0 0 0 . 0 0 P 0 0 0 0 0 0
. 0 0 0 0 0 0 P 0 0 . 0 0 0 0 0 0 0 0 0 . 0 0 0 0 0 P 0 0 0
. 0 0 0 0 0 0 0 P 0 . 0 0 0 0 0 0 0 0 P . P 0 0 0 0 0 0 0 0
. . 0 0 0 0 0 0 0 . . . 0 0 0 0 0 0 0 . . . 0 0 0 0 0 0 0 .
. . . . . 0 . . . . . . . . . 0 . . . . . . . . . 0 . . . .
. . 0 0 0 0 0 0 0 . . . 0 0 0 P 0 0 P . . . 0 0 0 0 0 0 0 .
. 0 0 0 0 0 0 0 0 N . 0 P 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0 0
. 0 0 0 0 0 0 0 0 0 . 0 0 P 0 0 P 0 0 0 . 0 0 0 0 0 0 0 0 0
. 0 0 0 0 0 0 0 0 0 . 0 0 0 P P 0 N 0 0 . 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 P 0 0 0 0 0 P P 0 0 0 0 0 0 0 0 0 0 0 0 0 0
. 0 0 0 0 0 0 0 0 0 . 0 0 P 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0 0
. 0 0 0 0 0 0 0 0 0 . 0 P 0 0 0 0 P 0 0 . 0 0 0 0 0 0 0 0 0
. 0 0 0 0 0 0 0 0 0 . P 0 0 0 0 0 0 P 0 . 0 0 0 0 0 0 0 0 0
. . 0 0 0 0 0 0 0 . . . 0 0 0 0 0 0 P . . . 0 0 0 0 0 0 0 .
. . . . . 0 . . . . . . . . . 0 . . . . . . . . . 0 . . . .
. . 0 0 0 0 0 0 P . . . 0 0 0 0 0 0 0 . . . 0 0 0 0 0 0 0 .
. 0 0 0 0 N 0 P N 0 . 0 0 0 0 0 0 0 0 0 . 0 P 0 0 0 0 0 0 0
. 0 0 0 0 0 P 0 0 0 . 0 0 0 0 0 0 0 0 0 . 0 0 P 0 0 0 0 0 0
. 0 0 0 0 P 0 0 0 0 . 0 0 0 0 0 0 0 0 N . 0 0 0 P 0 0 0 0 0
0 0 0 0 P 0 0 0 P 0 0 0 0 0 0 N 0 0 0 0 0 0 0 0 0 P 0 0 0 0
. 0 0 P 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0 0 . 0 0 0 0 0 P 0 0 0
. 0 P 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0 0 . 0 0 0 0 0 0 P 0 0
. P 0 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0 0 . N 0 0 0 0 0 0 P 0
. . P 0 0 0 0 0 0 . . . 0 0 0 0 0 0 0 . . . 0 0 0 0 0 0 P .
```

Fig. 4: *Map after 100 sets of 20 steps*

After the pandemic, we now have the average survival rate of the remaining players: $[0.73, 0.67, 0.58, 0.62, 0.63, 0.49, 0.55, 0.61]$. The average survival rate exceeds **49 out of 60**, indicating that having a high skill level correlates with longer survival.

In terms of percentage of classes, we still have:

- 15% of hunter
- 17% of warrior
- 21% of druid
- 30% of paladin
- 12% of shaman
- 2% of mage
- 1% of priest
- 2% of rogue
- 0% of warlock

To conclude, we will say that to resist the virus, one needs to be on average **level 49** and avoid choosing these classes: mages, priests, rogues, and warlocks.

III. PSO Algorithm Application

A. Simulation Parameters

1) *Spell::*

- **Duration**: 4 seconds
- **Damage**: 263 to 337 every 2 seconds
- **Trigger_Period**: Every 2 seconds
- **Range**: 100 yards

2) *Player Behavior Data::*

- **Initial_Population**: 100 players
- **R0**: 10

3) *Health Classes (HP)::* `calculate_hp_with_equip` computes player health points (HP) with equipment based on class and level following thoses equations:

$$\begin{aligned}
\text{Warrior: HP} &= 150.51 \times \text{level} + 120 \\
\text{Paladin: HP} &= 142.03 \times \text{level} + 120 \\
\text{Hunter: HP} &= 100.00 \times \text{level} + 100 \\
\text{Rogue: HP} &= 83.05 \times \text{level} + 100 \\
\text{Priest: HP} &= 74.92 \times \text{level} + 80 \\
\text{Warlock or Mage: HP} &= 71.53 \times \text{level} + 80 \\
\text{Shaman: HP} &= 133.56 \times \text{level} + 120 \\
\text{Druid: HP} &= 125.08 \times \text{level} + 120
\end{aligned}$$

B. Problem and fitness function

To implement the players, we use a problem function containing 8 ”solutions” ranging from 0 to 7.

- **Initial Number of Infected (`initial_infected`)**: This is the initial number of infected players at the beginning of the simulation.
- **Quarantine Effectiveness (`quarantine_effectiveness`)**: Measures how effective quarantine measures are in reducing the spread of the infection.
- **Familiar Effect (`familiar_effect`)**: Represents the impact of players’ familiars on disease transmission.
- **Teleportation Effect (`teleportation_effect`)**: Indicates how players’ teleportation ability affects disease spread.
- **Urban Avoidance Effect (`urban_avoidance_effect`)**: Explains how players’ avoidance of urban areas affects disease spread.
- **Quarantine Effort Intensity (`quarantine_effort`)**: Measures how much quarantine efforts are being implemented to reduce infection spread.
- **PvP Immunity (`pvp_immunity`)**: Represents players’ immunity in player versus player (PvP) areas.
- **NPC Asymptomatic Effect (`npc_asymptomatic`)**: Indicates the extent to which asymptomatic non-player characters (NPCs) contribute to disease spread.

Then, the transmission rate of the disease is calculated based on the provided parameters.

`assess_fitness` evaluates a solution’s fitness by calculating the total infected players across different player classes

C. Process

1) Particle update algorithm [7]: .

Algorithm 1 Particle update algorithm

```
0: procedure ParticleUpdate(Problem, velocity, position, index)
0:   Initialize Particle(problem, velocity, position, index)
0:   while not at stopping criterion do
0:     Assess fitness
0:     Update position and velocity using PSO algorithm
0:     if current fitness is better than previous fitness then
0:       Update fittest position
0:     end if
0:     Update previous fitness
0:   end while
0: end procedure=0
```

2) PSO Algorithm: .

Algorithm 2 Particle Swarm Optimization (PSO) Algorithm [7]

```
0: procedure PSO(problem, swarm_size, vector_length, num_informants)
0:   Initialize the PSO algorithm with problem, swarm_size, and vector_length
0:   Initialize the swarm with random particles and select a global fittest particle
0:   while not at stopping criterion do
0:     Update each particle in the swarm
0:     Update the global fittest particle
0:   end while
0: end procedure
0: procedure UPDATE_SWARM(follow_current, follow_personal_best, follow_social_best, follow_global_best, scale_update_step)
0:   for each particle in the swarm do
0:     Select informants for the particle
0:     Update the particle’s position and velocity
0:   end for
0: end procedure
0: procedure UPDATE_GLOBAL_FITTEST
0:   Find the fittest particle in the swarm
0:   Update the global fittest particle if necessary
0: end procedure=0
```

Here are two algorithms: the Particle Update Algorithm and the Particle Swarm Optimization (PSO) Algorithm. The Particle Update Algorithm iteratively adjusts particle positions and velocities based on fitness evaluations. The PSO Algorithm manages the swarm, updates particles, and selects the global fittest particle.

D. Results

1) PSO Parameters:

- **Swarm size:** 100 #Number of particles in the swarm
- **Vector length:** 8 #Length of the particle’s position vector
- **Number of informants:** 6 #Number of informants each particle considers
- **Number of iterations:** 50 #Number of iterations for PSO algorithm

2) PSO Coefficients:

- **Follow current:** 0.7 #Weight for following current velocity
- **Follow personal best:** 2.0 #Weight for following personal best position
- **Follow social best:** 0.9 #Weight for following social best position
- **Follow global best:** 0.0 #Weight for following global best position
- **Scale update step:** 0.7 #Scaling factor for updating particle velocity

3) Simulation Process: .

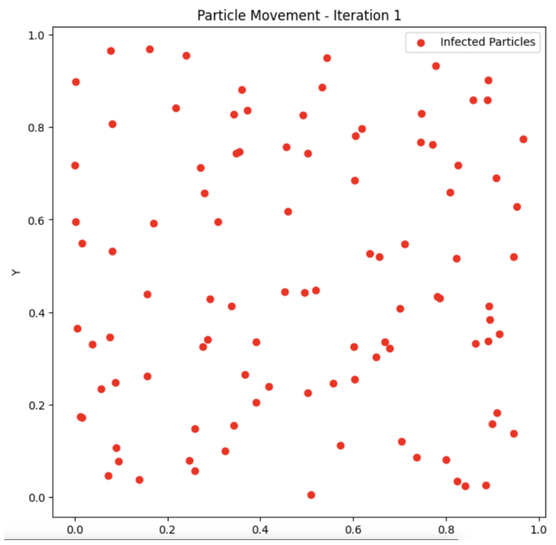


Fig. 5: *First Iteration – all particles infected*

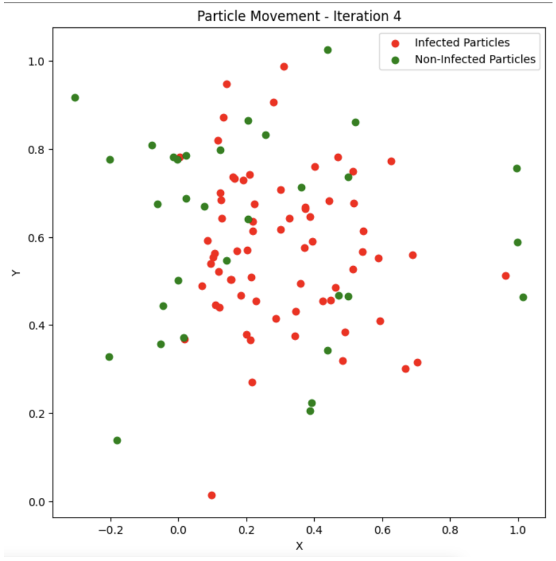


Fig. 6: *Fourth Iteration – particles infected and particles non-infected*

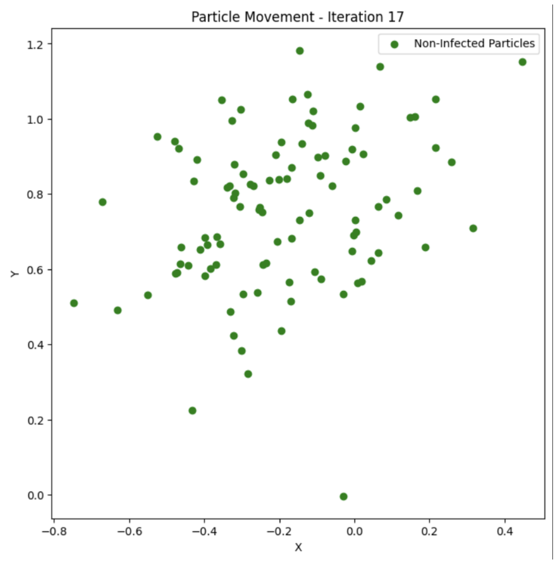


Fig. 7: *First Iteration – all particles non-infected*

4) Fitness Function [8]: .

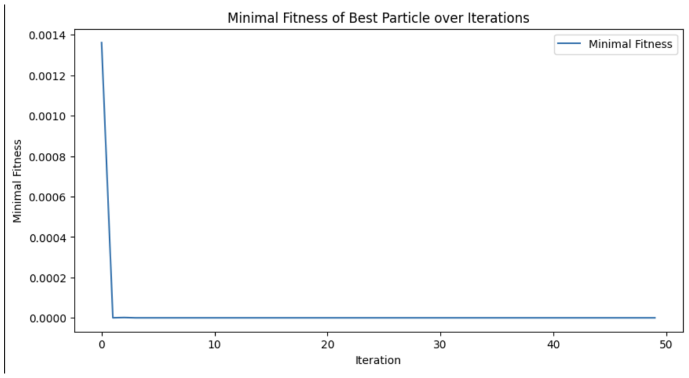


Fig. 8: *Minimal Fitness of Best Particle over Iterations*

5) Optimals Positions: .

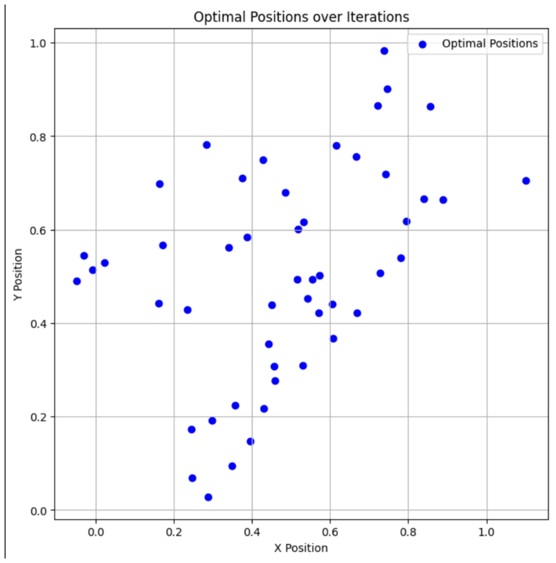


Fig. 9: *Optimals Positions over iterations*

6) Simulation Results: .

Optimal position:

$$[0.28310013, 0.56340775, 0.73298826, 1.1266275, 0.87363575, 1.18829493, 0.58170003, 0.96220079]$$

Minimal number of infections: 0.0 (drops close to 0 after 4 iterations.)

All particles are non-infected after **17 iterations.**

Percentage of remaining players by class:

- Paladin: 14.49%
- Hunter: 14.49%
- Mage: 14.49%
- Warrior: 14.49%
- Priest: 10.14%
- Rogue: 10.14%
- Druid: 7.25%
- Shaman: 7.25%
- Warlock: 7.25%

Average survival level: 28.90

To conclude, to resist the virus, one needs to be on average **level 28.90** and avoid choosing these classes: **Druid, Shaman, Warlock.**

IV. Conclusion

In summary, both the Genetic Algorithm (GA) and the Particle Swarm Optimization (PSO) Algorithm offer effective strategies for mitigating disease spread within virtual environments. The GA, with its iterative approach, provides a comprehensive solution by evolving player behaviors over multiple generations. On the other hand, the PSO algorithm optimizes parameters directly, leading to faster convergence and precise optimization. The choice between the two methods depends on specific requirements such as time constraints and modeling preferences.

References

[1] Jessica Smith: *Guide to the Corrupted Blood Plague Documentation Collection*, `https://searchworks.stanford.edu/view/xy157wz5444`

[2] Thottbot: *Corrupted Blood*, `https://web.archive.org/web/20051220020002/http://www.thottbot.com/index.cgi?sp=24328`

[3] Eric T Lofgren, Nina H Fefferman: *The untapped potential of virtual game worlds to shed light on real world epidemics*, `https://www.thelancet.com/journals/laninf/article/PIIS1473-3099(07)70212-8/fulltext`

[4] GameFAQs: *How much HP did level 60 raiding tanks have?*, `https://gamefaqs.gamespot.com/boards/534914-world-of-warcraft/52709466`

[5] Blizzard: *HP cap in Classic/Vanilla*, `https://us.forums.blizzard.com/en/wow/t/hp-cap-in-classic-vanilla/150800`

[6] Alba, Le Data Scientist: *Algorithme Génétique (french)*, `https://ledatascientist.com/algorithme-genetique/`

[7] Scott Condron's Blog: *Interactive Particle Swarm Optimisation Dashboard from Scratch in Python*, `https://www.scottcondron.com/jupyter/optimisation/visualisation/2020/08/02/interactive-particle-swarm-optimisation-from-scratch-in-python.html`

[8] Aleksei Rozanov: *Particle Swarm Optimization (PSO) from scratch. Simplest explanation in python*, `https://towardsdatascience.com/what-the-hell-is-particle-swarm-optimization-pso-simplest-explanation-in-python-be296fc3b1ab`