Individual Based Population Simulation

Mehmet Hakan Sarı

Abstract:

Population dynamics is research until early 1900's. Models are created to show underlying mathematics. Lotka-Volterra is one of the population model. There are some improvements on Lotka-Volterra model considering diffusion and mean field approximations. By the computational power increased with computers, the model is supported by stochastic models such as Monte-Carlo simulations. However, introduced stochastic models also somewhat deterministic and do not count individual movements. By using computational power, individual based population simulation is developed highly depend on Monte-Carlo method.

Keywords: population dynamic, Lotka-Volterra model, Real time simulation

I. Introduction & Background

Lotka-Volterra model is prey-predator model to explain population dynamics in the environment. The model depends on simple deterministic differential equations. Two coupled differential equations that contains prey predator and constants the model predicts the relationship between predator and prey. The equations are given as:

$$\dot{a}(t) = \lambda a(t)b(t) - \mu a(t)$$

$$\dot{b}(t) = \sigma b(t) - \lambda a(t)b(t)$$

The parameters λ, μ, σ explains the populations growth rate and death rate etc. By changing parameters the dependency can also be changed [1]. The population over time can be seen in figure 1.

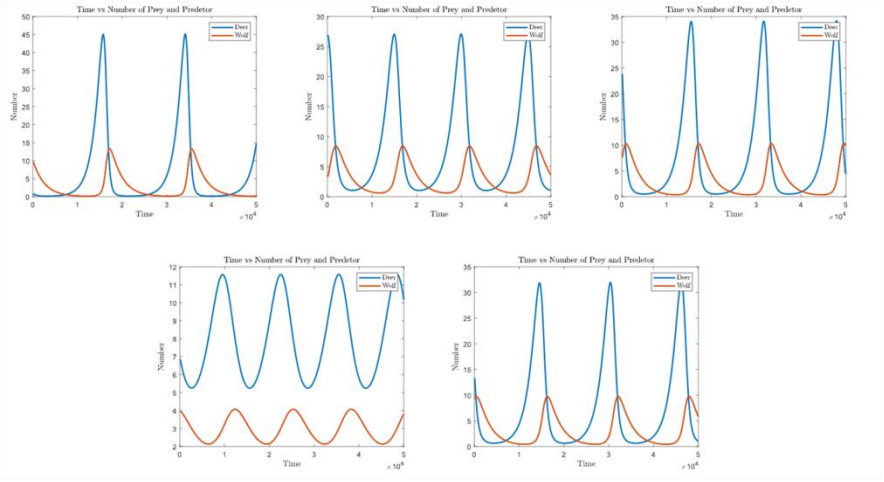

Fig1: *Deterministic Lotka Volterra Model with different parameters.*

However, in this model the population and the potential is very deterministic and predictable. As it can be seen from figure 1, the cycle are periodic depending on their initial conditions and continue in such a way. This model is discussed through the years and following the base equations new models are developed. These models are, "The Two-Species Lotka-Volterra Rate Equation", "The Zero-Dimensional Stochastic Lotka-Volterra Model", "Mean-Field Rate Equations With Finite Carrying Capacities", "The Deterministic Reaction-Diffusion Equations" and lastly "The stochastic lattice Lotka-Volterra Model" [2,3]. These models uses differently than the original LV model, averaging techniques, probability distributions etc. One important aspect of the models is that they focus on the spatial information of the predators and the capacity of the environment which is not cared by the original Lotka-Volterra. It is said that, new models are more suitable to model such population dynamics. Last model that I want to discuss is Stochastic Lattice Lotka Volterra Model (SLLVM).

SLLVM employ numerical Monte Carlo simulations and field theoretic arguments in order to break deterministic cycle of the LVM. This model depends on the lattice information of the field. For certain rules and constant rates, the lattices are filled by preys or predators. These rules can be followed as:

- Death of predator at rate μ
- One random lattice diffusion at rate D
- Offspring generation by the prey at rate σ
- Predator hunts the prey and give offspring nearby at rate λ

While the capacity of the populations guaranteed by the λ, μ, σ's, the spatial information is also added through the diffusion rate. By using Monte-Carlo simulation the rules are employed in a stochastic way. The diffusion or offsprings location are determined by randomly that creates non-deterministic cycles but the population is controlled by one capacity either [3]. The population over time graph can be seen in figure 2.

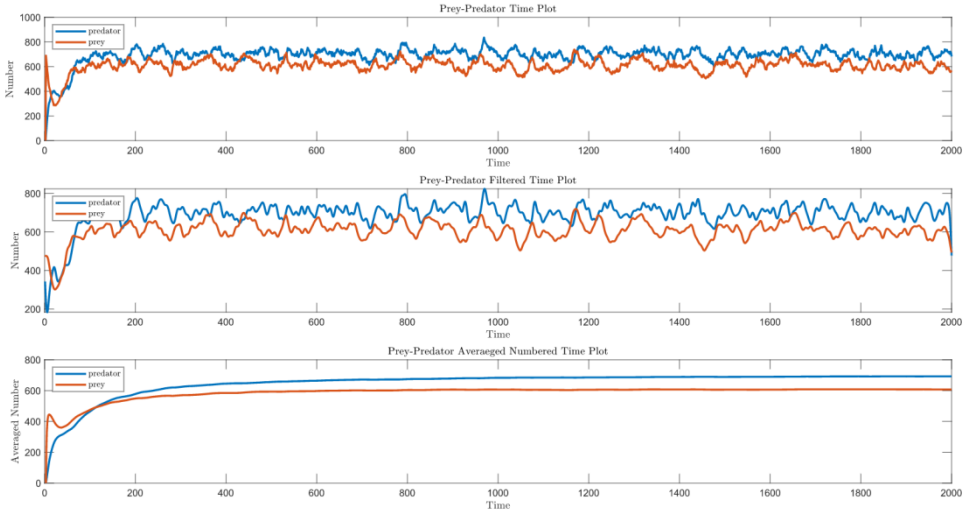

Fig 2: *SLLVM results for certain parameters*

However, SLLVM also do not consider individual movements inside the population. The rules for diffusion and hunting are very exact that individual behavior is not considered and the individuals are not tracked. This idea is supported by several papers that individuals in the population also carry huge importance [4]. Calling that by using computational power and data analytics I decide to employ randomwalk-longdistance diffusion based population simulation. In this work, I hope to observe individual primitive behavior would result in meaningful mathematically dependent population dynamic. My base idea was to employ what Nathan et. all suggested [4]. Thus I called my simulation trial as, individual based population simulation.

II.Simulation

Disclaimer: Whole simulation is written and designed by myself. Due to time limitation I could not employ what I really desire to see. By small improvements the simulation could really give a population dynamic. At the moment, it is just alpha version of the simulation.

When I think about the environment, I see that the landscape and terrain are the key aspect of one species life. Following this, food source is included in this list. Thus, before the creating lattice, I work on lattice itself. For larger environmental population dynamics (such as deer, wolf, fish etc.), one lattice could lay on the mountains, hills, step, lake. Each of the terrain has its own specification and offerings for the species. For instance, instead of exact randomwalk of the predator in every lattice, predator might not pass the mountains or it will spend more energy to pass that it will increase its death rate. Forests might give better protection to prey and hunt ratio would decrease. Therefore, the constants suggested by the other models λ, μ, σ, would also depend on the lattice or prey, predator population. In other words, constant will not be constant anymore but depend on spatial and maybe temporal information. After I decide that constant could depend on lattice, I designed lattice.

For generality, lattice is created by the user. By a drawing program one can draw a map that contains different terrain and landscapes (just water-grass are available in current version). By introducing this feature, we earn lots of degree of freedom to work. Whole diffusion process connected to the lattice information. For demonstration I drew “Trabzon” and “Baku” maps.

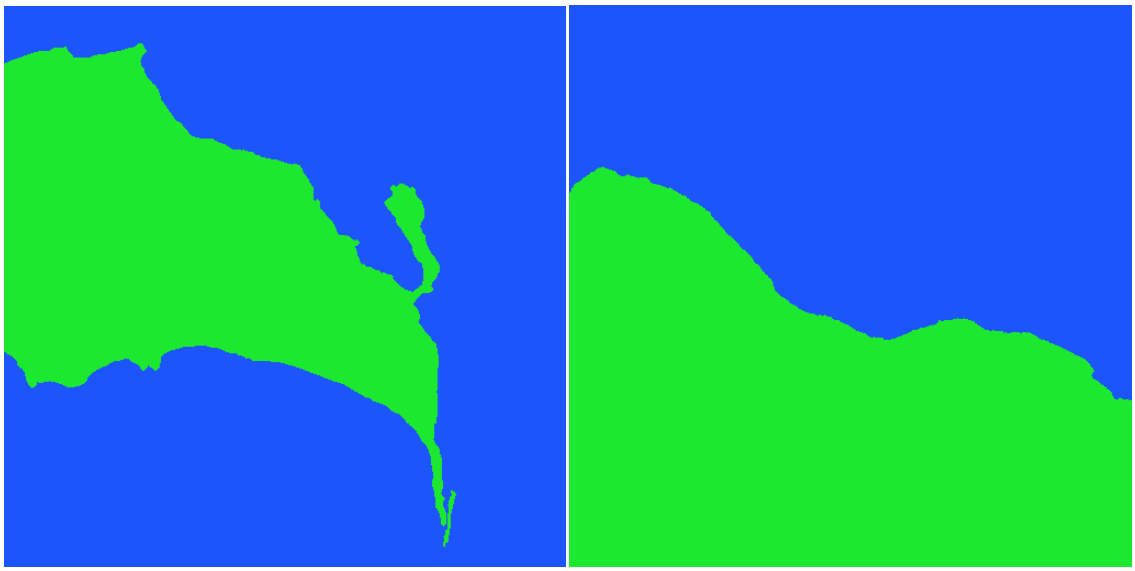

Fig 3: Baku and Trabzon as lattice information. The maps are 512x512 pixels.

The strong side of this is the user can draw any lattice that is wanted to observe. If whole map was land, it would be same lattice used in SLLVM. Besides, both different population characteristics can be examined at same time. The land and the sea could have different populations and different constant. It is great way to implement such things.

For this project, I select fish-fisherman population dynamic as I want to discover less numbered smart predator – large number dummy prey interaction. Before I introduce rules, I want to discuss the properties and methods of two species.

A. Fisherman < Unit

I want to fisherman to wander around the sea if there is no fish in its line of sight. If fish enters its line of sight, it should chase the closest fish. When its nearby the fish, it fishes. In next version of my simulations, they will dock, refuel and drop the source they collected. During this time fish can reproduce. So docking would be like death of the predator in other models. I did not implement unit superclass at the moment. But unit defines the class is individual.

Properties:

name: Name is for tracking the fisherman.

LOS : Line of sight defines how far fisherman can see the land.
source: Source counts how much fish a fisherman fished.
fuel: Each diffusion in the lattice consumes fuel. It counts movement.
location: Location of the fisherman in global coordinates. Global coordinates determined by the map defined by the user.
status: “legal” or “illegal”. Fisherman information should check in.

ID: ID is the fisherman ID.

Methods:

wander()

Fisherman wanders by randomwalk. It always try to be away boundary. (End of the map and land in this case)

hunt()

Fisherman fishes the fish around lattice. The radius of hunting is 3x3 matrix at the moment. The fish is caught at 0.8 probability.

chase()

Fisherman chases the closest fish. If there is no fish it continuous to wander. The hunt function is initiated from this function. If the chased fish is nearby, it hunts.

radar()
To avoid boundaries and fish detection radar function is used. Radar function calls map functions. Map function provides, landscape and population maps for the fisherman.

initialcheck()

For random distributed initial conditions, the function checks if fisherman is on land or another fisherman. It also register the fisherman.

register()

Register function register the fisherman to the map. Thus map knows where the fisherman is located.

B. *Fish < Cluster*

Fish usually do not wander around by itself. It is going with its school. Thus small fish is not an individual itself like fisherman. Thus, I count fish as cluster. Each fish cluster has independent size and configuration. Cluster can be used later for forests for this setup. Though, if one want to work bacteria population, it is also logical to use cluster superclass. For time-being I could not add the individual interactions in a cluster as it is cumbersome to implement.

SuperClass Properties

size: The number of the individuals that create one cluster.

configuration: It is a matrix that contains how individuals are formed. Each configuration is randomly formed.

birthrate: It defines the probability of giving birth out of ten.

compactness: It defines how much one configuration be compact. If compactness large, the individuals can be more separate.

solitariness: It defines how much one cluster be close to one another. It is useful for forest patches.

CM: Center of mass of the configuration. It is used to attach configuration to global coordinates.

Class Properties

name: not available at the moment. It can used for tracking fishes.

LOS: Line of sight the fish. It is used for detecting land.

location: Location information in global coordinates.

ID: Fish ID that appears on map.

SuperClass Methods

generate():

The cluster method that generate random looking configurations. The generated cluster is stored in a matrix.

divide():

I did not implement yet. But it is crucial to divide clusters after some threshold values. For instance each fish can have capacity. After that capacity, cluster should divide.

update():

It updates cluster through the map. Thus, the updated cluster is formed by the map. After cluster is updated configuration is also updated.

move():

Cluster moves for given direction. Move function can be very complicated for considering each individual movement and it will be specific for each configuration. But I just move location due to simplicity.

birth():

Birth function increase the size of the cluster. If one cluster is larger, it has more chance of giving birth. The location of birth selected randomly and configuration is updated accordingly. But, currently my birth algorithm does not satisfy compactness.

Class Methods

wander():

Fish does randomwalk by its configuration. The boundaries are known by the sense function.

migrate():

It is not implemented yet. Sometimes fish schools should migrate to find food. It will create more dynamic population map.

sense():

It does same thing with fisherman radar function. According to fish LOS, it sense around it to locate other things.

initialcheck():

It is used for randomly distributed fish schools. If a school in top of another school or land, new location is selected.

save():

The fish population is saved to population map.

The methods and properties are summarized as such. In the following section results of the simulation will be shown. Figure 4 and 5 shows initial conditions for simulation. The fisherman is the small orange pixel. Light blue is fish cluster.

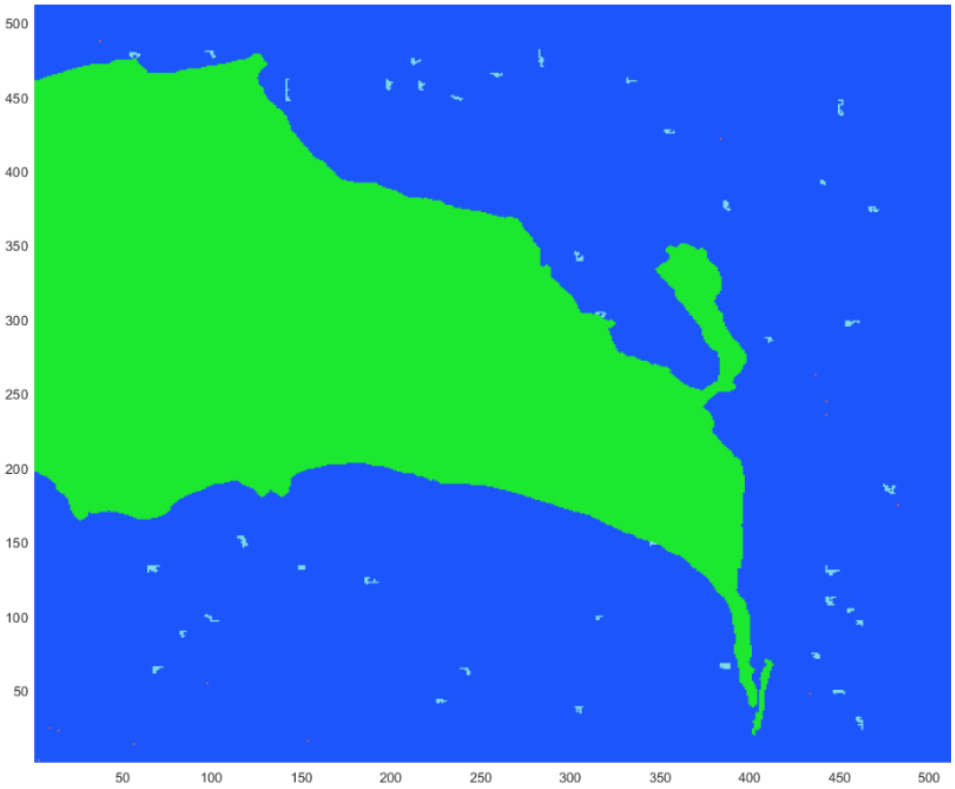

Fig 4: *Baku with 40 fish schools and 14 fisherman ships.*

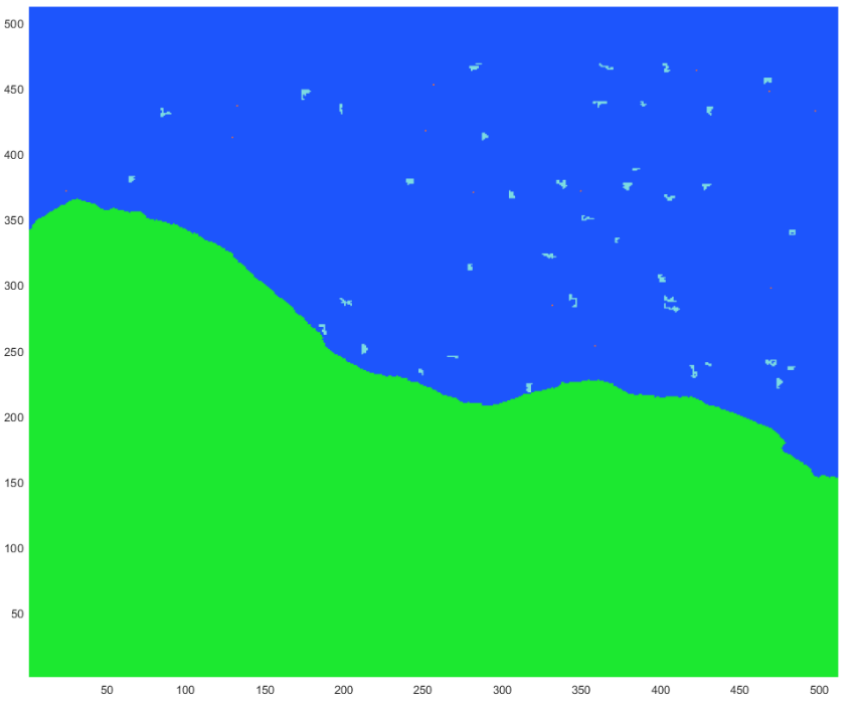

Fig 5: *Trabzon with 40 fish schools and 14 fisherman ships.*

III.Results

The simulation is run for each fisherman and cluster at the same time step. Fish schools do randomwalk in each instance and give offspring in each 10 time step, whereas fisherman chases fish if it is in its LOS, otherwise it also do randomwalk. The measured quantities are the fish population, resource that fisherman gathered and fuel that is spent. As I did not include docking function for the fisherman, the simulation cannot capture the population dynamics. Also, **divide()** function should be written for clusters in order to increase school number after it exceeds threshold value. Nonetheless, the simulation works for some extend. There are known errors and bugs in this version which interrupts the simulation.

Known Common Errors:

Fish clusters usually hit the boundary. Searching possible errors.

Error in fish/wander (line 68)

```
map_pop(y-CMy+1:y-CMy+m, x-CMx+1:x-CMx+n) = obj.configuration;  %probably this one is correct.
```

Error in OOP_project (line 75)

```
[school(i),z.population] = school(i).wander(z.population, land_dis,bound_dis);
```

Population map extends beyond the lattice. Visualization can give errors or fisherman, fish goes beyond lattice.

There might be more bugs that I have not encountered yet. I am keeping track of the errors to fix them in future. Although these issues exist I can provide some results to comment on.

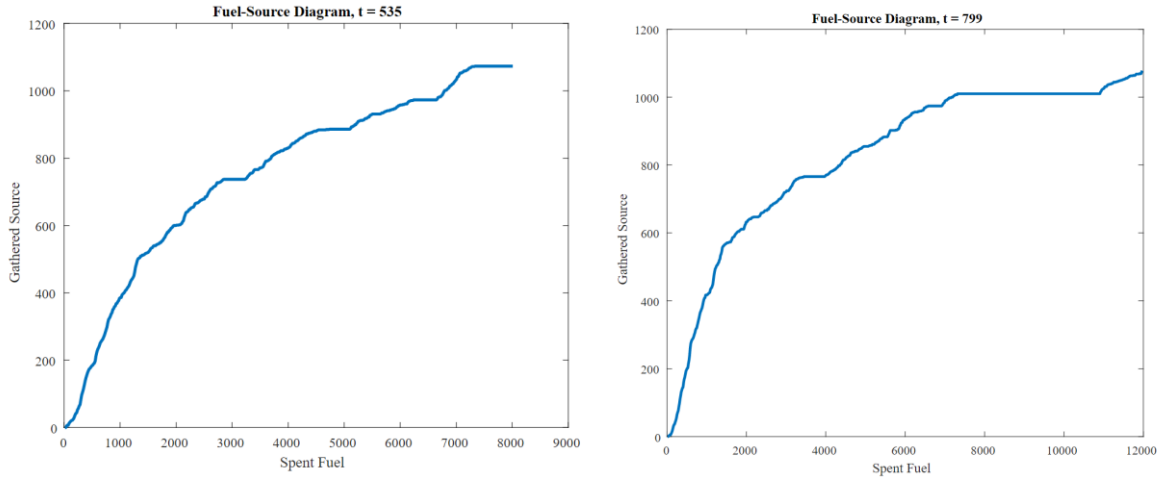

Fig 6: *Fuel-Source graph for Baku map. The simulations are run until it crashed.*

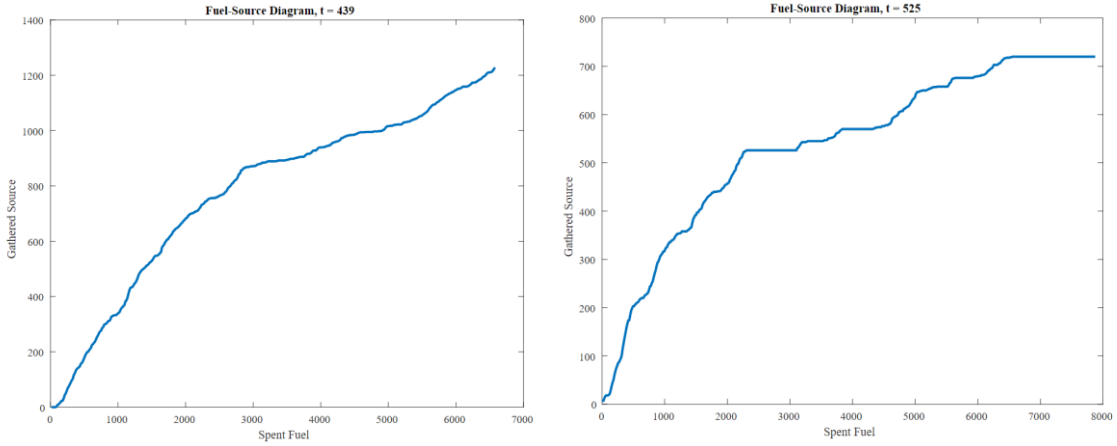

Fig 7: *Fuel-Source graph for Trabzon map. The simulations are run until it crashed.*

In figure 6 and 7, the source gathered are compared with the fuel that is spent. It can be meaningful data for searching fuel efficiency. I run 2 simulations for each map to observe this relationship. In some time instances, the source stays constant. During these time intervals, fish reproduce and do not go extinction.

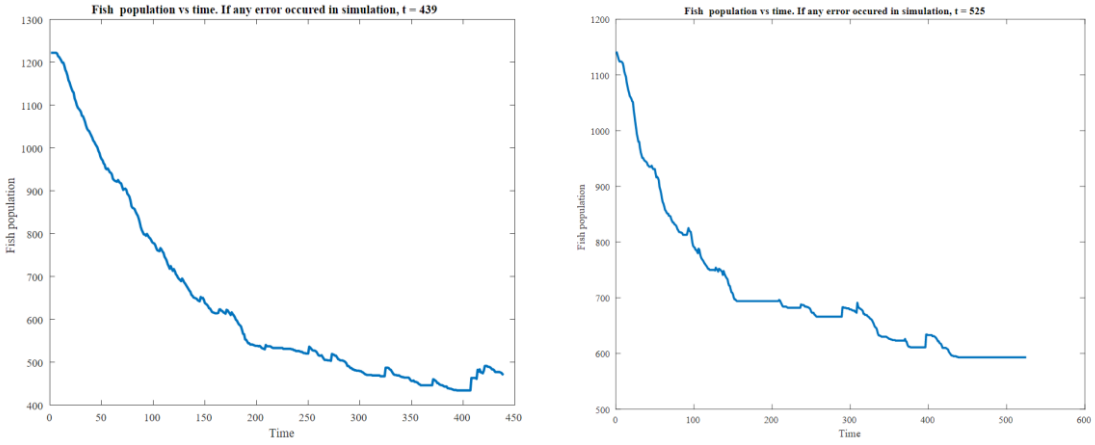

Fig 8: *Fish population graph for Trabzon map. The simulations are run until it crashed.*

In figure 8 right hand side graph, together with figure 7, we can see that there is a time interval that collected source and fish population stays constant. Even if fish entered nearly 10 times offspring period (fish give birth in each t = 10 with some probability), the population remain constant. Thus, I conclude that fish cluster reached its stagnation point and cannot reproduce anymore. This shows that **divide()** function is necessary for fish reproduction.

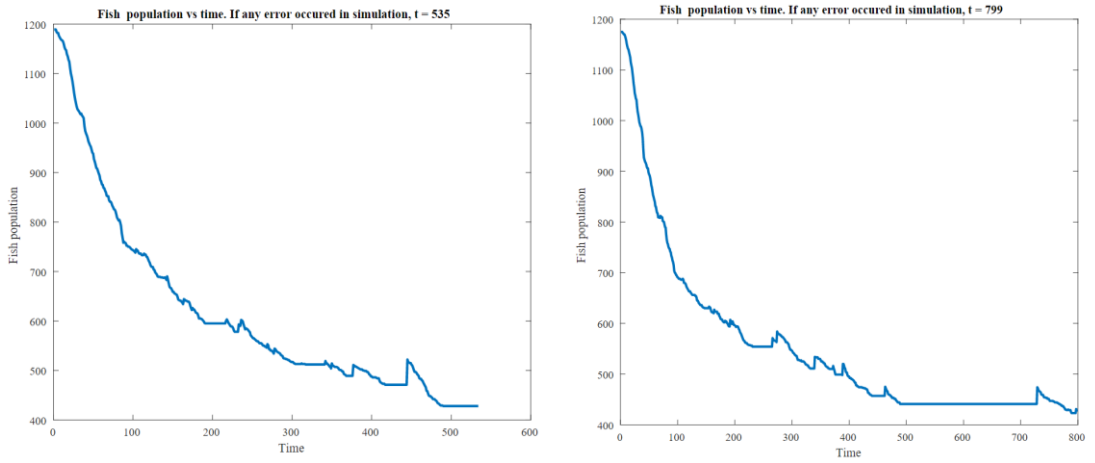

Fig 9: *Fish population graph for Baku map. The simulations are run until it crashed.*

Baku map shows this stagnation more explicitly. In figure 9 right hand side, we see that fish cannot reproduce over 20 offspring periods. During this time period their number also does not decrease because fisherman cannot locate fish population.

In order to quantify the output, I tabulated the parameters that I used in this simulation. There are so many other parameters that can be changed and observed to understand interaction between less numbered smart predator – high numbered dummy prey.

Map	BirthRate	HuntRate	FuelConsumption	LOS	Off-spring period	Threshold to divide
Trabzon	0.4	0.8	-1	30	10	N/A
Baku	0.4	0.8	-1	30	10	N/A

Table 1: *Parameters used in the simulation*

IV.Conclusion

In this project I try to develop a population dynamic simulation depending on previous work ideas [1,2,3,4] to observe more realistic, more stochastic and more individual based population dynamics. By using this simulation, user inputs can be easily given and the parameters can easily be changed. Thus, the simulation does not just provide what it intend to do but user defines its usage. Thus a SLLVM model or diffusion based LVM can also be conducted by arranging rules accordingly.

Even if there are many bugs and unfinished work for this simulation, by more work, it can be implemented on more complex systems simulation. Because the program is fully written in object oriented and the specification can be easily increased for the purpose of the study. The cluster superclass can be improved and individual interaction can be provided. A unit super class can be employed to create individual based species. This can also be supported by team superclass that will arrange interaction between the units. Besides, map class can be supported by terrain class to introduce grid dependent constants. There are endless possibilities to improve the simulation and add new features. However, bugs, errors always should be tracked.

Lastly, about the performance of the simulation, one time step took 0.2 seconds for 15 fisherman and 70 fish clusters. In 0.2 seconds, 85 times random walk runs and maps are updated. This performance can obviously be improved by constructing better and efficient algorithms.

V.References

[1] Lotka AJ (1920) Analytical note on certain rhythmic relations in organic systems. *Proc Natl Acad Sci USA* 6:410–415.

[2] Dannemanna, T., Boyer, D., & Miramontes, O. (2018, March 26). *Lévy flight movements prevent extinctions and maximize population abundances in fragile Lotka–Volterra systems*. pnas.org. Retrieved February 27, 2022, from https://www.pnas.org/doi/10.1073/pnas.1719889115

[3] Morales JM, Haydon DT, Frair J, Holsinger KE, Fryxell JM (2004) Extracting more out of relocation data: Building movement models as mixtures of random walks. Ecology 85:2436–2445.

[4] Nathan R, et al. (2008) A movement ecology paradigm for unifying organismal movement research. Proc Natl Acad Sci USA 105:19052–19059.

VI.Appendix

If you want to test map by yourself, you can draw a square painting in paint. You need to use:

Grass = [R = 28, G = 232, B = 48];

Water = [R = 29, G = 85, B = 252];

Be sure that you use “filled with color” or simple pen for unity in pixels. Brushes do not provide unity in RBG unless spray.