# Agent Based Modeling
# Price Determination with Evolutionary Strategies
# — Final Report —

Berk Can Özmen - 396694

February 17, 2025

## 1 Introduction

The model attempts to create and simulate a basic production-consumption-trade cycle, consisting of individuals that can decide using evolved strategies. The model provides an environment and a set of rules for agents, where agents can decide on their actions using an arbitrary function, which will be optimized through genetic algorithms. More specifically, the agents produce a set of goods, trade them, and consume them in order to survive

Since the environment and the specific behaviors of each agent creates the dynamics of the system, using agent based models is a necessity. With classical equilibrium approach, it is impossible to model the complex system behaviours of the environment.

At a low level, the model consists of individual agents with specific strategies that interacts with the environment. Each agent's decision-making process is governed by a set of neural networks that dictate actions related to production, trade, consumption and movement. Additionally, environmental variables, such as resource availability/distribution and seasonal changes, influence agent behavior. The emergent property of the model is a self-organizing pricing mechanism, trade patterns, and specialization emerge naturally from agent interactions without central coordination.

## 2 Literature

Unfortunately, I am not aware of an existing directly comparable literature, however related ideas that offer inspiration in fields like, agent-based computational economics, machine learning (for neural network and genetic algorithm parts), classical micro economics, and systems such as "Conway's Game of Life" can be found.

The main ideas/assumptions of the environment construction comes from classical microeconomics, including fundamental assumptions such as "economical rationality" and supply-demand model. Since these ideas are considered as "common knowledge" I am not going to directly use references to them.

The ideas regarding neural networks, usage of feed forward neural networks, also depends on a fundamental theorem of statistical learning theory (Vap95), namely universal approximation theorem (HSW89). For short, the universal approximation theorem states that given a feed forward network with enough hidden layer neurons, it can approximate a function with error $\epsilon > 0$. In other words, we assume that, functions that are encoded as neural networks, can approximate any arbitrary function, limited by their capacity and data that they have seen. See also the probably approximately correct learning, a fundamental theorem of statistical learning theory, from A Theory of the Learnable (Val84).

Even though statistical learning theory gives us bounds about what we can learn and what we cannot, it doesn't tell us about how to learn. Genetic algorithms is an established method for black box optimization. Regarding this project, the main challenge is that we cannot define a loss function, i.e we cannot define what strategy is good and what strategy is bad, rather we want this to be a function of environmental parameters. Thus, to optimize the neural networks used, a method similar to the proposed in (Yao99) is used. For the mutation method proposed by (Yao99), where a parallel genome encodes the mutation strength of each bit of gene, is found to be too unstable, thus a single bit mutation with possibility of sign swap is employed as in (ZAS20).

## 3 ODD

### 3.1 Purpose and patterns

Given survival criteria, based on natural resources that can be produced by agents and local market dynamics, and strategies that are evolved through genetic algorithms, creating simple market dynamics, where prices of goods are decided by the environment and the strategies of the agents

### 3.2 Entities, state variables and scales

#### Agent Variables

| **Variable** | **Description** | **Scale** |
|---|---|---|
| genome | A string that encodes 4 neural networks Float | |
| age | Age of the agent [0-25] | |
| charm | Sexual selection fitness of agents [0-1] | |
| tv-dist | A variable to quantify how diverse the agent eats | [0-1] |
| reputation | A variable to quantify the history of agents success in satisfying its needs | [0,1] |
| utility | A variable to quantify the trade success of an agent | [0-1] |
| produce-rate | production distribution of the agent 3x[0-1] | |
| inventory | inventory of the agent | list of 3 positive integers |
| prices | Agent's pricing of each item | list of 3 positive floats |
| trade-wish | Agents decision of how many items to buy,sell | 3 floats |
| trade-wish-plot | A helper variable to plot the trade-wish | 3 floats |
| needs | Needs of the agent | 3 positive floats |
| mutated? | If the agent has undergone a mutation | boolean |
| reproduce? | If the agent satisfy the condition to reproduce | boolean |
| activation | The activation function of the agent's neural networks | 0,1,2 |
| net-produce | Monitor variable to track agents production | |
| net-trade | Monitor variable to track agents net trade | |
| net-trade-volume | Monitor variable to track agents trade volume | |
| net-consume | Monitor variable to track agents consumption | |

#### Patch Variables

| **Variable** | **Description** | **Scale** |
|---|---|---|
| elevation | Elevation of the patch | [0-1] |
| total-production-wish | Variable to keep track of total production investments in the patch by all the turtles on this patch | 3x[0-inf] |

**Environmental Variables**

| **Variable** | **Description** | **Scale** |
|---|---|---|
| input-lengths | Input layer neurons of the networks | Positive integer |
| output-lengths | Output layer neurons of the networks | Positive integer |
| network-layers | Description of hidden layers | List of hidden layer neurons |
| networks | Network shapes | list of list of layers |
| genome-length | Integer length of the genome | Positive integer |
| gene-points | List of where each gene starts and ends | List of integers |
| start-inventory | starting inventory | list of 3 positive floats |
| min-needs | Minimum needs to survive | list of 3 positive floats |
| start-needs | 2x min-needs | Same as needs |
| max-needs | maximum needs that can be satisfied | Same as needs |
| gdp-per-capita | Estimation of number of items in inventories | List of positive floats |
| production-per-capita | Estimation of yearly production of items | List of positive floats |
| consume-per-capita | Estimation of yearly consumption of items | List of positive floats |
| trade-per-capita | Estimation of yearly trade volume | List of positive floats |
| death-by-starvation | Counter for how many people died of starvation in a year | Integer |
| death-by-age | Counter of how many people died of old age in a year | Integer |
| new-borns | Counter of how many people is born in a year | Integer |
| season | Season | [0-4] integer |
| max-product | Baseline production of each item, read by a .csv | Table of positive numbers |
| season-efficiency-table | Season efficiency table, read by a .csv | Table of positive numbers |
| good_table | Nutrition values of goods, read by a .csv | Table of positive numbers |
| need_table | Min-needs, read by a .csv | Table of positive numbers |
| max-age | Maximum age that an agent can survive up to | Integer |
| initial-population | Starting population size | Integer |
| fertility-start | The age that agents can start reproducing | Integer |
| fertility-end | The age that agents become infertile | Integer |
| nutrition-multiplyer | Each goods nutrition value will be multiplied by this value | Positive float |
| mutation-chance | Rate of mutation | [0-1] |
| fog-radius | The max interaction distance between agents | Positive float |
| market-size | Max number of exchanges each turn | Positive integer |
| nm-decay | The rate of decay of nutrition multiplier | [0-1] |
| nm-min | Minimum nutrition multiplier | Positive Float |
| nm-min-pop | Population limit, population over this will cause nutrition-multiplier to decay | Pos Integer |
| mutate? | If mutation is on | boolean |
| crossover? | If crossover is on | boolean |
| inherit? | If inheritance is on | boolean |
| inheritance | percentage of inheritance | boolean |
| pr-gaussian-a | Variable to control production limits of fish and meat | [0-1] |
| pg-gaussian-b | Variable to control production limit of wheat | |

## 3.3 Process overview and scheduling

```
Set season
Reset variables that needs reset
foreach people[
    produce
    for i in market-size[
        get-prices
        exchange
    ]
    consume
    move

    check-reproduction
    if reproduce?[
        sort people according to their charm
        respecting this order mate
    ]
    evaluate-survival
    set age age+1
    set reputation
]
```

## 3.4 Design concepts

### Basic principles

The mechanics of the environment are hard coded. Then genetic algorithms is used to characterize every agent. The genome encodes 4 feed forward neural networks (directly it's weights and biases with a predetermined and identical architecture for each agent). Then through the environmental mechanics, the agents are faced with a survival pressure, and are expected to adapt to their environment by natural selection, i.e adapting their neural network weights by mutation and crossover operators.

4 Neural networks represents the decision strategies for production, trade, consumption and movement. The agents must acquire goods (either through production or trade) and consume them in order to survive. The amount of consumption dictates survival, where the quality of consumption (how diverse it is and how fulfilling it is) and the profit from trade (utility) dictates the sexual success (i.e the right to choose it's mate or be attractive to the chooser). Through these mechanisms, agents are expected to evolve neural networks that is adapting to the environment.

### Emergence

The production, consumption, exchange and movement strategies are expected to emerge. These strategies are limited by the input to the neural networks, however since the neural networks can encode an arbitrary function, the structure of the strategies cannot be determined pre-simulation. Since optimization of neural networks doesn't depend on a loss function, we do not dictate a good-bad strategy. A suitable strategy is the one that can survive, while interacting with other agents, thus the "fitness" of a strategy is relative, i.e a strategy that is good in one configuration of agents, may not be successful in another. Thus through these strategies, a population dynamic is expected to emerge

The trade dynamics includes direct interaction between agents, where they offer each other agreements. Thus a "profitable" trade should improve the survivability of an agent. As a result, an agreement of prices (exchange rate of goods) and the amounts is supposed to emerge.

### Adaptation

Through natural selection, the unfit agents are expected to die, and the agents will be adapting to the environment indirectly. Through sensing their environment (season and elevation etc) they can also adapt their strategies.

### Fitness

The agents do not have any explicit goals. However agents that cannot satisfy their needs by consuming goods (which can be acquired either through trade or through production), will die and agents that eats a balanced diet are more likely to reproduce. Thus an indirect fitness can be defined as surviving / satisfying their needs.

### Learning

Agents do not explicitly learn, however since the strategies are encoded in their neural networks, through natural selections, their neural networks is expected to converge to a state, that allows them to survive.

### Prediction

The agents do not explicitly predict their environment. The prediction is also dictated by their genome, which dictates their strategies. Thus the strategies that are more fit to the environment (thus can implicitly predict) are expected to thrive in the population.

### Sensing

The agents can explicitly sense the season, their location, other agents and their own internal variables (inventory and needs). These variables are used as inputs to their neural networks. They will use this information to decide on their production, consumption, movement, exchange and mating strategies. They can also implicitly sense the behavior of other agents, since their survivability depends on the other agents (because of the trade). However this is also implicitly governed by the genetic algorithms.

### Interactions

Agents can trade with each other. Each agent will determine a price and amount to buy/sell for each type of goods. Then according to their inventories, they will offer an exchange to other agents. Agents can also reproduce with other agents, where they choose their mate depending on their success. During reproduction, agents will share their genome and create a new agent randomly sampled from both of the parents genome.

During production, the agents that are on the same patch compete for the resources. It means that, if two agents want to produce the same good on the same patch, they will share the maximum production capacity of the patch, thus interacting implicitely.

### Stochasticity

The agent's genome is initialized randomly, since the aim of the model is to come up with good strategies through evolution. During reproduction, the children's genome is randomly chosen from parents genome (to keep the good solution) and also mutated randomly (to allow exploration of new strategies).

### Collectives

There are no explicit collectives of individuals. They all have unique strategies (unique genomes), and apart from them they are the same. However, through strategies collectives of individuals can emerge, such as farmers, fishers or traders. Since producing only one good is more efficient, and each good can be produced efficiently in different areas and agents need different types of goods this type of specification is expected. For example, a viable configuration could be, fishermans in low altitudes, farmers in high altitudes and traders that carry goods in between and live off the trade they are making.

### Observation

In the current state, we track the production, amount of goods in each individual's inventory, traded goods and their prices, population, cause of death, new borns, average age of death and average number of children of each agent is tracked. These variables are tracked either directly from the global variables, or are computed through agents local variables.

### Initialization

```
to setup
    clear-all
    reset-ticks

    ;set global parameters like needs, genome basics etc
    set-parameters
    ;set elevation of each patch
    setup-map
    ;create the initial population
    create-population initial-population
end
```

## 3.5 Input Data

`good.csv`
Determines the nutrition value of items

| | **Protein** | **Fat** | **Luxury** |
|---|---|---|---|
| **Fish** | 10 | 2 | 0 |
| **Wheat** | 0.5 | 0.5 | 10 |
| **Meat** | 2 | 10 | 0 |

Table 1: Nutrition values of items

`need.csv`
Determines the min-needs of an agent $[30, 30, 0]$

`season-efficiency.csv`
Determines the efficiency of which item can be produced in which season

|  | **Winter** | **Spring** | **Summer** | **Fall** |
|---|---|---|---|---|
| **Fish** | 1.5 | 1.0 | 0.5 | 1.0 |
| **Wheat** | 2.0 | 0.0 | 2.0 | 0.0 |
| **Meat** | 0.5 | 1.0 | 1.5 | 1.0 |

Table 2: Season efficiency table

|  | **Fish** | **Wheat** | **Mean** |
|---|---|---|---|
| **Base Production** | 10 | 10 | 10 |

Table 3: Base production values of items

`max-product.csv`
Determines the baseline production of each item

## 3.6 Submodels

### Agents

Every agent contains 4 feed forward neural networks (with the same number of hidden layers and neurons). and a genome, that contains 4 genes and encodes the weights of the neural networks. Every NN decides on an action (in order of the genome encoding), i.e produce, exchange, consume, move.
**Networks**
Each network is a feed-forward network and contains 3 hidden layers with [32,16,8] neurons, with relu activation.
**Produce NN**
inputs : Normalized coordinates, normalized inventory, normalized needs, season, number of turtles here
length : 9
outputs : Softmax - 3 neurons that encode the percentage of production of each item
**Exchange NN**
inputs : Normalized coordinates, normalized inventory, normalized needs, season, Local prices from last tick
length : 11
outputs : First 3 neurons encodes how much to buy/sell. The last 3 neurons encode the prices of the items (last 3 neurons uses relu)
**Consume NN**
inputs : Normalized coordinates, normalized inventory, normalized needs, season
length : 8
output : Each neuron encodes how much of each item to be consumed
**Move NN**
inputs : Normalized coordinates, season, crowd info of 5 neighboring sectors (square with fog-radius)
length : 8
output : Heading and how much to walk, with relu activation

### Needs

Every agent needs 30 protein and 30 fat to survive a tick. The luxury need does not affect the survival. It can still be satisfied, and it effects the charm of the agent. Then $min\_needs = [30, 30]$, $start\_needs = [60, 60]$, $max\_needs = [90, 90]$

|  | **Protein** | **Fat** | **Lux** |
|---|---|---|---|
| **Fish** | 10 | 2 | 0 |
| **Wheat** | 0.5 | 0.5 | 10 |
| **Meat** | 2 | 10 | 0 |

Table 4: Nutrition values of items

Reproducing also costs *min_needs* amount of needs. Other than that, needs can boost/down production and the chance of reproducing.

|  | **Protein** | **Fat** | **Lux** |
|---|---|---|---|
| **Person** | 30 | 30 | 0 |

Table 5: Minimum needs to survive

### Production

Every agent makes a decision (pmf) to how much invest in each type of production by using the neural network. This is the base of their production ($p$). Then this base rate is multiplied by elevation efficiency ($e$), season efficiency ($s$), health efficiency ($h$) to produce produce-rate ($pr$)

Then $pr_i = h \times s_i \times e_i \times p_i$

Then each agent competes at the patch they are on with other agents. So for example if there are 2 agents $(i, j)$ that want to produce $p_1^i = 0.2$ rate fish and $p_1^j = 0.3$, $pr_1^i$ get $ps_1^i = 2/5$ and $pr_1^j$ gets $ps_1^j = 3/5$ production share ($ps$) of the total production cap $pc$.

Then item gets by the agent is calculated as $= pr_i \times ps_i \times pc_i$

1. Health ($h$)
Health is a efficiency constant that is decided on satisfied needs. It maps needs in term of 2*min-needs, by scaling linearly intervals $[0, 90] \rightarrow [0.5, 1.5]$ and means the result.

```
let health mean (map [ [a b] -> a / (2 * b)] needs min-needs)
```

2. Elevation Efficiency ($e$)
Elevation efficiency decides on how efficient can each product be produced, given the elevation of the patch. The functions:

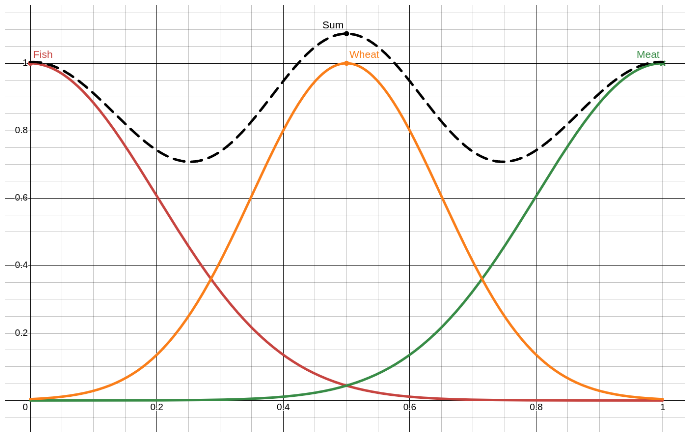


Figure 1: Elevation efficiency function

Season Efficiency ($s$)

|  | **Winter** | **Spring** | **Summer** | **Fall** |
|---|---|---|---|---|
| **Fish** | 1.0 | 0.5 | 0.5 | 0.75 |
| **Wheat** | 0.0 | 2.0 | 2.0 | 0.0 |
| **Meat** | 0.5 | 1.0 | 0.75 | 0.5 |

Table 6: Season efficiency table

|  | **Protein** | **Fat** | **Luxury** |
|---|---|---|---|
| **Fish** | 10 | 2 | 0 |
| **Wheat** | 0.5 | 0.5 | 10 |
| **Meat** | 2 | 10 | 0 |

Table 7: Nutrition values of items

3. Production Cappacity ($pc$)

|  | **Fish** | **Wheat** | **Mean** |
|---|---|---|---|
| **Base Production** | 10 | 10 | 10 |

Table 8: Base production values of items

**Consumption - Needs - Charm**

Every agent has 2 basic needs and a luxary need that it must keep satisfying, by consuming goods. Every tick, agent decides how much of each item to consume to satisfy their needs. The nutrition is added to agent's needs, and needs are decreased by min-needs, and agent's satisfied needs cannot exceed max-needs (3 time min-needs).

Agents produce more if they satisfy their needs more. Agents get to choose their partner first, if they satisfy their needs more, and they consume min-needs amount of needs to reproduce. Lastly, consuming a more diverse diet makes agents more charming, which increases their chances of being chosen as a partner.

```
to consume
    ;subtract the min needs (each turn once)
    set needs (map - needs min-needs)

    ;decide on consumption
    let output relu (decide 2)
    ;consume min(consumption, inventory)
    let actual-consume (map [[c i] -> min list c i] output inventory)

    ;increase needs by amount of nutrition acquired
    set needs (map + needs (calculate-nutrition actual-consume))
    ;needs cannot exceed the max-needs
    set needs (map [ [a b] -> min list a b] needs max-needs)
    ;take the consumed items from inventory
    set inventory (map - inventory actual-consume)

    ;calculate charm of the agent
    ;calculate tv dist of their consumption to uniform distribution
    set tv-dist 1
     if sum actual-consume > 0[
       set tv-dist max ((map [[a] -> (3 / 2) * (abs ((a / sum actual-   consume) - (1 / 3)))] actual-consu
      ]
     set tv-dist (1 - tv-dist)

    ;calculate how much of their needs they could satisfy
    let need-dist min (map [[a b c] -> (a - c) / (b - c)] needs max-needs start-needs)
    ;set their chamr 0.25*tv-dist + 0.25*need-dist + 0.25*reputation + 0.25*utility
    if age > 0[
        set charm tv-dist * (1 / 4) + need-dist * (1 / 4) + (1 / 4) * ((reputation / (age * 4)) ^ 2)  + (1
    ]
end
```

**Exchange**

First each agent decide on their trade parameters (trade-wish and prices). Then they look at other people around them, in fog radius. For the item pair i,j (sell i, buy j), if both seller and buyer profits from the exchange, the exchange is carried on. This procedure is repeated market-size times

```
foreach range market-size[ix ->
    set trade-count ix
    ;decide on trade parameters
    ask people[
        get-prices
     ]
     ;trade
    foreach shuffle sort people[p ->
```

```
        ask p[exchange]
    ]
]

to get-prices
    ;Use the NN
    let output (decide 1)

    ;Process the output
    set trade-wish (sublist output 0 3)

    set trade-wish-plot trade-wish
    ;set trade-wish (map * trade-wish inventory)

    set trade-wish (map [[tw i] -> ifelse-value tw < 0 [-1 * min (list (abs tw) i)] [tw] ] trade-wish inve

    set prices  map [[a] -> a + 1] relu (sublist output 3 6)
    ;set prices (map [[a] -> (9 * a) + 1] prices)
end


to exchange
    ;get turtles in range
    let market-turtles other turtles in-radius fog-radius
    if count market-turtles = 0 [stop]

    ;for each it
    foreach shuffle (range 3)[i ->
    foreach shuffle (range 3)[j ->
        ;if the agent wants to sell i and buy j
        if (item i trade-wish < 0) and (item j trade-wish > 0)[
            ;calculate i per j price
            let b-exchange-rate (item j prices / item i prices)

            ;get other turtles that wants to buy i and sell j
            let market-ij market-turtles with [(item i trade-wish > 0) and (item j trade-wish < 0)]

            ;check if the trade is profitable for both parties
            set market-ij market-ij with [ ((item j prices) / (item i prices)) < b-exchange-rate ]

            ;sort them by most profit
            set market-ij sort-by [[t1 t2] -> ((item j [prices] of t1) / (item i [prices] of t1)) <  ((ite

            ;realise the exchange
            shop market-ij i j
      ]
    ]
    ]
end
```

### Move

In the output layer, relu function is used.

```
to move
    pen-down

    ;use the NN
    let output relu(decide 3)

    ;process the output
    ;let head heading + ((item 0 output) * 360)
    let head heading + (item 0 output mod 360)
    let forw min list 2 (item 1 output)

    ;execute the command
    set heading head
    fd forw

    pen-up
end
```

**Genetic Algorithm**

**Survival**

```
to evaluate-survival
    ;check for age
    if age > max-age[
        die
    ]
    ;Check if all needs are satisfied (element-wise)
    if (member? false (sublist bool-list 0 2 ))[
        let bool-list (map [[a b] -> a >= b] needs min-needs)
        if (member? false (sublist bool-list 0 2 ))[
        die
    ]
end
```

**Reproduction Fitness**

```
to check-reproduce
    ;Check if inventory consist bigger numbers than start-inventory (element-wise)
    let bool-list (map [[a b] -> a > 1.95 * b ] needs start-needs)
    ;set reproduce? true if all the elements of bool-list are true
    set reproduce? not (member? false (sublist bool-list 0 2 ))

    ;check if the agent is in fertility
    if age < fertility-start or age > fertility-end [set reproduce? false]
    set reproduce? (reproduce?)
end
```

**Mate Selection**
Agents are sorted by their charm. And starting from the most charmful agent, agents selects their partners. A charm score is calculated, based on the diversity of the agent's diet (TV distance to uniform distribution). Every agent selects the most charming partner in radius fog-radius. After mating, both parents lose [min-need] amount need. Then they can again be eligible to reproduce if they are chosen by another agent as partner and still satisfy the reproduction condition.
**Crossover**
The crossover is based on 4 genes. Genes are never mixed (not to destroy the Neural Networks). Each child gets one gene from one parent with %50 of chance.
**Mutation**

```
to mutate
    ;Get how many bits will be mutated from a poisson distribution
    let n random-poisson 0.5 + 1

    ;For each mutation
    foreach range n[
        ;select a random bit
        let index random (length genome)
        ;multiply the bit by a random float [0.5-1.5]
        let rand ((random-float 1) + 0.5)
        ;randomly swap the sign
        let sign ((random 2) * 2 - 1)
        ;set the new genome
        set genome (replace-item index genome (item index genome * rand * sign))
    ]
end
```

**Child**
The child inherits a percentage (inheritance variable) of one of their parents inventory. They are hatched from this parent, randomly in a circle with radius fog-radius. The rest of the variables are set to 0, and updated during the simulation

## 4 Model Verification

### 4.1 Unit tests

Testing of individual functions is done, to control if they work as expected. The standard methods of unit tests are employed, where a set of input values is supplied to functions and the output is compared with the expected output

### 4.2 Dynamics and Evolution

The simulation is run with different initial conditions (The relative nutrition of goods, initial nutrition-multiplier, initial population, nutrition-multiplier decrease rate), to see which parameters selections drives evolution and supports survival. However these tests are conducted to get an idea of the environment, and as a result a set of initial conditions are configured, in a way that it makes trading profitable.

## 5 Peer Review

Peer review of the project is done by Robert Gebhardt. To answer the suggested improvements regarding the smoothness of the simulation, the reason the initial population is high, is to start with a bigger and diverse search space, and possibly, from the begging, already find average solutions. The model fundamentally has the computation-exploration trade off, where the increase in population comes with a faster and completer exploration of the search space, while increase the required computation.

Regarding the visualization, it is acknowledged that coming up with meaningful sumamry statistics is hard. Therefore, I have also used python to further analyze the behaviour of the neural networks (see results)

## 6 Model Parameters

#### Nutrition Multiplier (nm)

Acts as a coefficient to scale the nutrition of items matrix. Effects and properties of the parameters is going to be discussed in results.

| | **Protein** | **Fat** | **Luxury** |
|---|---|---|---|
| **Fish** | 10*nm | 2*nm | 0*nm |
| **Wheat** | 0.5*nm | 0.5*nm | 10*nm |
| **Meat** | 2*nm | 10*nm | 0*nm |

Table 9: Nutrition values of items

#### $\sigma$ of the Production Gaussians

The standard deviation of the Gaussian distributions in Figure:1. Fish and meat uses the variable gaussian-pr-a = 0.3 while wheat uses gaussian-pr-b = 0.3. These parameters are kept constant during an evolution. The change in parameter has a strong effect of natural boundaries that a population extend to. Thus, for high values of gaussian-pr-a, fish and meat producers can produce the complementary product on their own, thus decreasing the value of trade. The parameters are chosen to be large enough to support two independent fish and meat producer population that are close, but small enough that they will need a mediator (possibly a wheat producer or mixed producer population) to trade.

#### Other parameters

Other parameters that are not investigated, due to lack of time and lack of computational resources, are mutation chance, fertility window, max-age, fog-radius, market-size and initial population (likely not a very important factor).

## 7 Results

It is hard to come up with a quantitative analysis of the results. Thus I will provide some intuitions and a qualitative analysis.

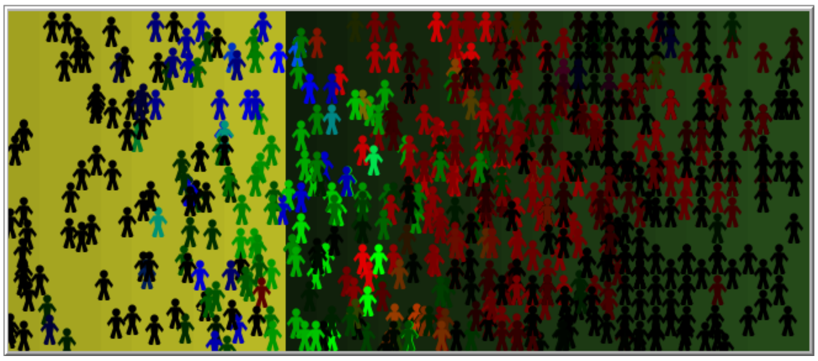
## 7.1 Nutrition Multiplier

Since the starting strategies are random, the random agents are not very successful in survival. Thus, I have implemented a "nutrition-multiplier" variable, where the nutrition of goods are artificially decreased throught the evolution.

Since the survival of an agent depends on it's relation to other agents, individual strategies are meaningful in relation to population strategies. Even though competition in production, trade and sexual selection provides an incentive to improve agent's strategies, the populations might stuck in a local optima and become conservative. This means that, the population is stabilized in a way, where they do not allow agent's with new strategies to survive and actually resist evolution, i.e keeping their population stable, and allowing only the new-borns with "normative" strategies to survive

These populations require an external shock to drive evolution of new strategies. This can either happen naturally, by contacting another population (either their borders grow into each other, or individuals travel), or artificially by decreasing the nutrition multiplier.

For example, in this case, fish is traded from left to right. The meat producers in the middle, and right gets access to fish through trade. However part of the population in further right cannot get access to this trade. Decreasing the nutrition multiplier might cause them to die faster, where they will be replaced by agents that trade or agents that come up with better pricing, thus driving the evolution faster.



(a) Visualization of net trade (buy). Agent's color is encoded in RGB, where R = fish, G = wheat, B = meat

Indeed, it is many times observed that, a (especially sharp) decrease in nutrition-multiplier is followed by increase in trade-per-capita.

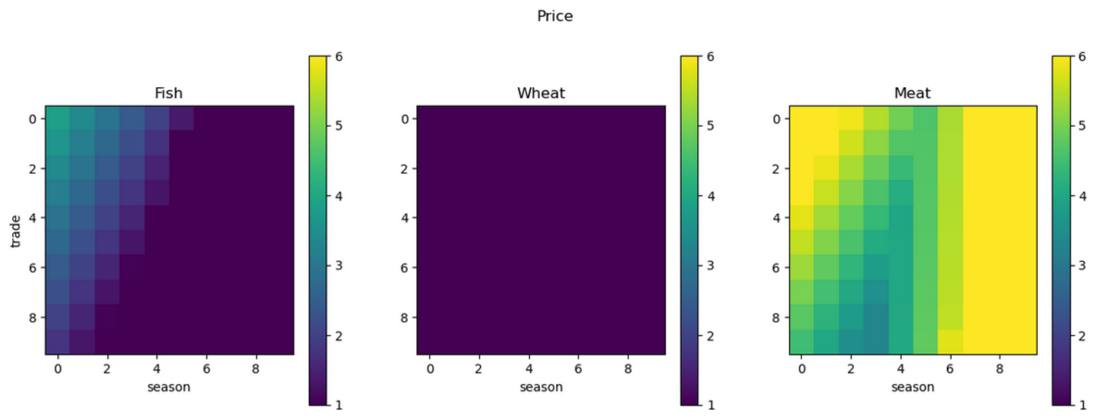
One more phenomenon that is observed, is that some populations will close themselves off to trading. I could speculate, that in the beginning, with random strategies that leads to bad pricing of items might cause agents who are willing to trade to make very bad deals and get extinct, thus only the agents that do not want to trade survive. Since the environment parameters are constructed in a way that, as the nutrition multiplayer decreases, it is impossible to survive without trade, such populations tend to extinct if they cannot adopt different strategies.

## 7.2 Luxury Goods

At the start, I designed three complementary goods that can be produced in different locations on the map. Each good primarily fulfills one need while also partially addressing the others. The idea was to create an environment, where specializing on production of one good and trading for the other good will obviously be better.

However this creates a situation, where each agent satisfies some of their needs at the minimum, because they don't have easy access to other goods. When an external shock happens, they might easily lose the bare minimum and causes the extinction of whole populations.

Thus I have changed one of the needs into a "luxury" needs, where satisfaction of it doesn't affect the survival, but it increases the chances of reproduction for the agent. Shocks to this good has less consequences, so it provides a tool to control the behavior of the agents. Also, this possibly makes this good a better medium of exchange (money like item), since spending it doesn't directly effect the survivability. It's not clear if this is the reason, but in multiple run it is observed that, the price of the luxury good (wheat) stays at 1, where the prices of other goods is changed.



(a) Visualization of price decision of one agent, with varying season and trade number (the number of trade out of market-size)

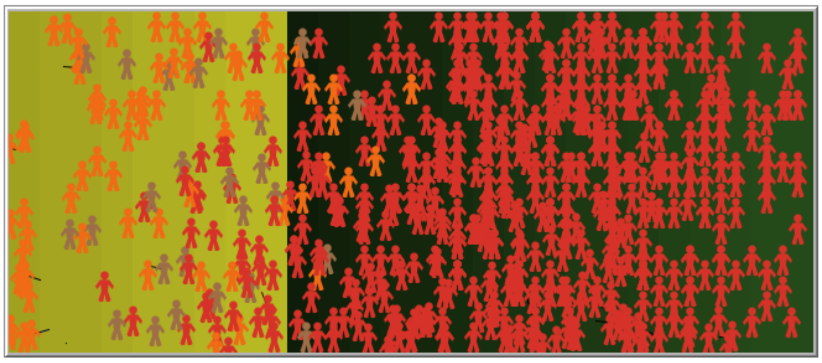
## 7.3 Population Genetics

As expected, in the beginning we end up with a couple of clusters of population (each agent has a random genome). First these clusters stabilizes, and their genome converges. Then, they start increasing their population and spreading around, either until they meet another population or the production variables change too much that they couldn't adapt. They do also adopt new strategies as they spread across the map, which creates a variation in the population genome.

Yet, we can still observe that, the variation in inter-population genome is much higher than intra-population, thus allowing us to recognize them by K-means clustering.

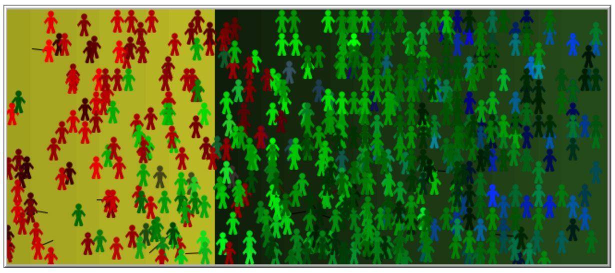
When two different population meets each-other, two things might happen. They can both take a conservative stance, where a mixture of their genome cannot survive so they do not mix into each-other and stay as different population with different strategies. These populations can be trading with each other, however non-trading populations are also observed. Or they can start mixing with each other, and this can result in the two population merging into a third and different population.

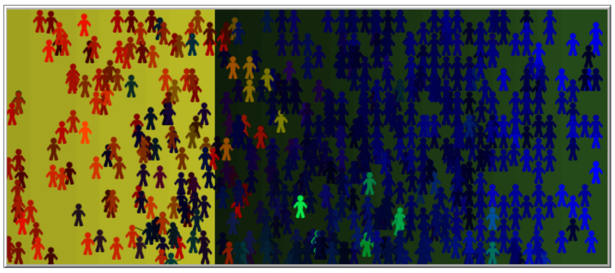
Through this population mechanics, a new implicit survival pressure emerges, (apart from survival and sexual selection), where potentially two population compete over the trade of an item, and drive each other to extinction.



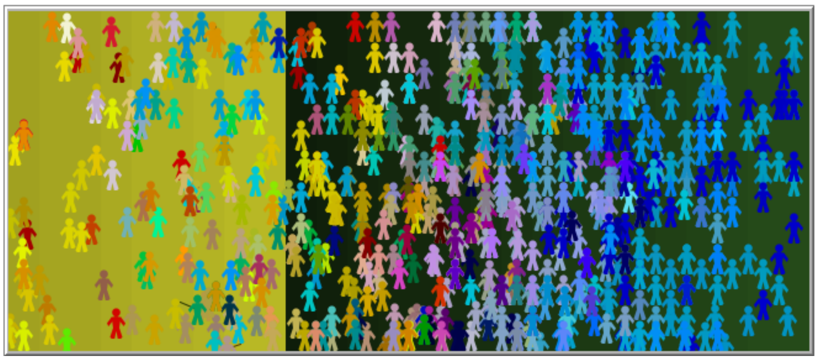
(a) K means clustering of populations, where 3 different population can be identified.

## 7.4 Production, Consumption and Trade

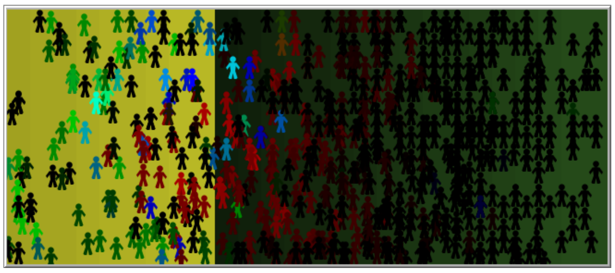

(a) Visualization of production in winter season. . Agent's color is encoded in RGB, where R = fish, G = wheat, B = meat

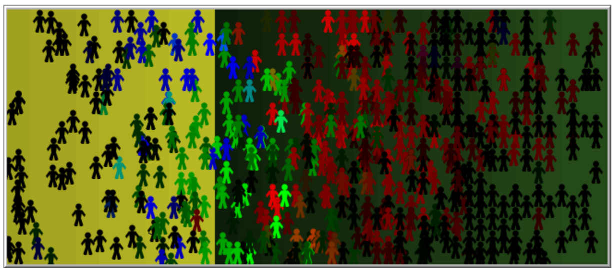

(b) Visualization of production in summer season. . Agent's color is encoded in RGB, where R = fish, G = wheat, B = meat

As mentioned before, at different location, different products can be produced. Thus, in the visualization of production above, we see a clear distinction between fish (red), wheat (green), and meat (blue) produced areas. Remember that wheat (green) is the luxury good and can only be produced in winter and summer seasons. In this case, we see that agent's prefer to produce wheat only in the winter season, and produce meat in the summer season even it's inefficient to produce it in the middle section.


(a) Visualization of consumption. . Agent's color is encoded in RGB, where R = fish, G = wheat, B = meat

However in the visualization of consumption, we see that there are yellow (fish and wheat) and blue (meat) colors in the fish region, and light blue (wheat and meat) in the meat region. The consumption is more uniform (which is preferred by the environment dynamics for both efficiency and sexual selection). This is only possible through trading.

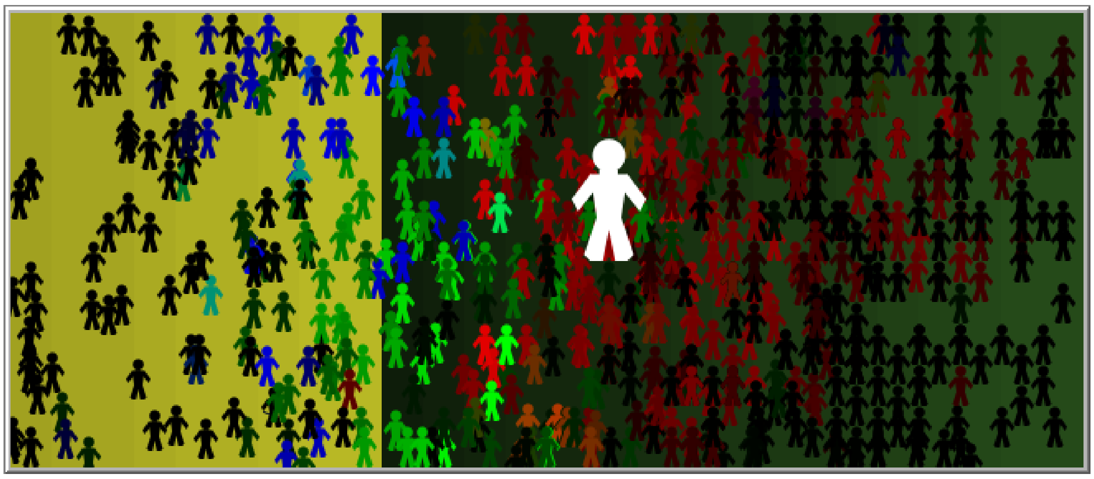

(a) Visualization of net trade buy in spring. . Agent's color is encoded in RGB, where R = fish, G = wheat, B = meat


(b) Visualization of net trade buy in autumn. . Agent's color is encoded in RGB, where R = fish, G = wheat, B = meat

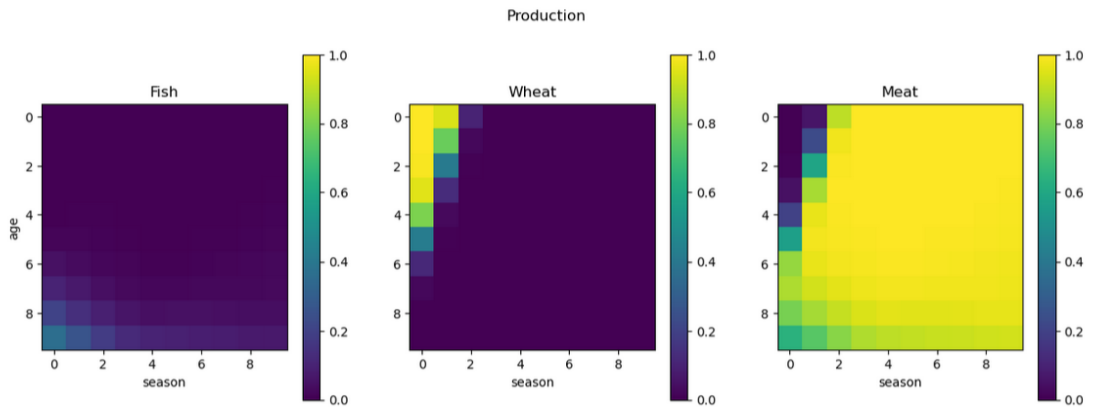
In this case, we see a yearly cyclic behavior. Where through the seasons fish is carried to right (first middle section buys it from left section, then right section buys it from the middle section) and meat is carried to the left section of the map. This is the result of the emergent behavior we were expecting to see, where agents synchronise their behavior, and an unspoken agreement about which season to trade which item is reached.

## 7.5 Individual Decision

The above mentioned behavior can also be observed in the decision boundaries of individual agents. In this case, the agent is the white individual marked at the graphics below, a meat producer, that also produces a little bit of wheat in the early season and when it's young.

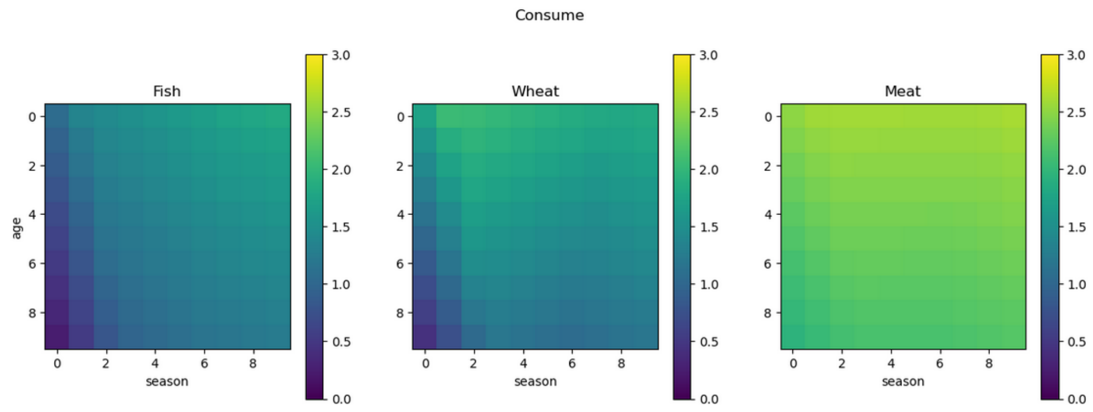


(a) Visualization of the individual in the net trade plot



(a) Visualization of production decision boundaries of an agent, plotted with varying season and age index

The agent's consumption is uniform which is prefered by the environment

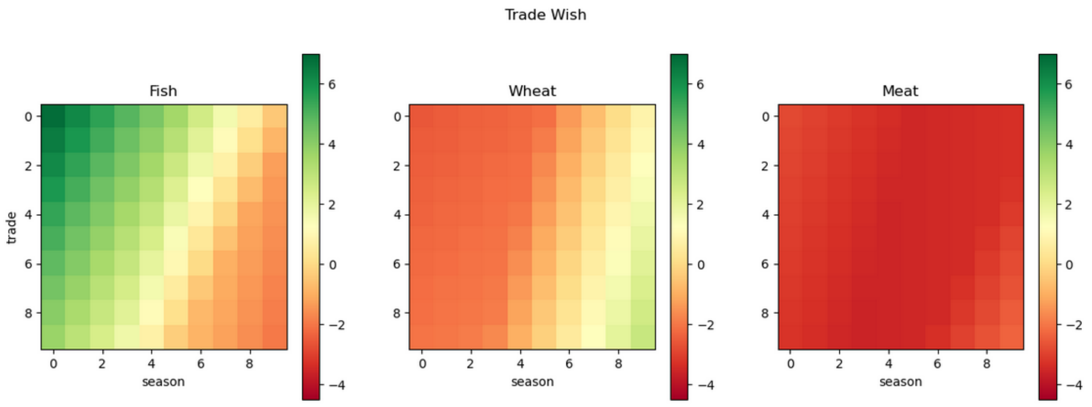


(a) Visualization of consumption decision boundaries of an agent, plotted with varying season and trade index. Positive values means buy, and negative values means sell

And lastly, we can observe the cyclic trading behavior. The agent always want to sell meat (a meat producer with excess meat). However, the decision boundaries for fish and wheat is split into two regions by season, where the first

2 seasons the agents wants to buy up to 5 fishes in exchange for up to 2 wheats or 4 meats

The last two seasons, the agent wants to buy up to 4 wheat in exchange for either up to 2 fish or 4 meats. Thus, aiming a trade surplus of fish and wheat depending on the season



(a) Visualization of net trade decision boundaries of an agent, plotted with varying season and trade index. Positive values means buy, and negative values means sell

## References

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[Val84] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984.

[Vap95] Vladimir N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., 1995.

[Yao99] Xin Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[ZAS20] Farah Zainuddin and Md Fahmi Abd Samad. A review of crossover methods and problem representation of genetic algorithm in recent engineering applications. 29:759–769, 01 2020.