

UNIVERSITY OF ATHENS
M.Sc. in BIOINFORMATICS
ATHENS 2005

Assignment in
COMPLEX ADAPTIVE SYSTEMS

Topic
Ecological food webs

Elpida Tzafestas

1 Objective of the project

The objective is experimentation with a basic food web of three species and variations thereof. More precisely, the student has to carry out studies of behavioral and ecological parameters and seek and compare various types of equilibria. See detailed instructions in section 4.

2 Ecosystem description

The ecosystem includes agents belonging to one of three organism species : carnivores, herbivores and plants. The behavioral model of the first two is uniform : they chase their prey in their vicinity and if they catch one, i.e. if they find themselves in the same place with it, they kill it and eat it, in which case they gain **Ef (Energy Food)** amount of energy.

If their energy (initially **E0**) raises above a level **Er (Energy Reproduction)**, then they reproduce themselves, that is they “give birth” to a new individual of the same species, and lose half their energy. Moreover, the agents lose consistently an amount **dE** of energy in every simulation cycle. If their energy falls below 0, then they die. The plants reproduce in every cycle (this is a quick model in order to accelerate the system). The ecological (non-behavioral) parameter **Reset Energies** defines whether a newborn agent’s energy is initialized to **E0** or whether it inherits the (lowered) energy of its parent, that is about **Er/2**. In the latter case, we may intuitively expect the ecological system to demonstrate relative energetic instability. The function implementing the above is given below :

```
public void execute()
{
    if ((type != CARNIVORE) && (!dead)) // HERBIVORE
    {
        chasePrey();
        energy -= dE;
        FoodWebAgent prey =
            simulation.herbivoreAt(position.x,position.y);
        // simulation.plantAt(position.x,position.y);
        if (prey != null)
        {
            energy += Ef;
            prey.die();
        }
        if (energy >= Er)
        {
            energy /= 2;
            simulation.divide(this);
        }
        else if (energy <= 0) die();
    }
    else if ((type == PLANT) && (!dead))
        simulation.divide(this);
}
```

The above ecosystem would lead to an uncontrollable (exponential) growth of the three populations. Ecological systems modeling has shown that a new ecological (non-behavioral) parameter has to be introduced, the so-called **carrying capacity** parameter, that denotes the maximum number of agents allowed in any single environmental position. With this setting, whenever an agent tries to reproduce, the reproduction (division) act is successful and takes place only if the total number of agents in its position is less than the corresponding carrying capacity. We use two different capacities, one for the “mobile” agents (carnivores and

herbivores) and one for the plants. In the case of the plants, we need to define two capacity parameters, the minimum and maximum plant capacity which allow plant reproduction. The function that implements reproduction according to the above is given below. The user may try to exclude capacity processing from the system to actually observe exponential population growth.

```
public void divide(FoodWebAgent motherAgent)
{
    Point pos = motherAgent.getPosition();
    if (motherAgent.isPlant())
    {
        int pl = getPlantsAround(pos.x,pos.y,1);
        if (((pl > MAX_PLANT_CAPACITY) || (pl < MIN_PLANT_CAPACITY))
            && usesPlantCarryingCapacities)
            return;
    }
    else // motherAgent == CARNIVORE or HERBIVORE
        if ((getAllMobileAgentsAt(pos.x,pos.y) >= CARRYING_CAPACITY)
            && usesCarryingCapacity)
            return;
    FoodWebAgent agt = new FoodWebAgent(this);
    agt.makeCloneOf(motherAgent);
    agt.setPosition(pos.x,pos.y);
    if (resetEnergyOnDivision()) agt.resetEnergy();
    else agt.setEnergy(motherAgent.getEnergy());
    addAgentAt(pos.x,pos.y,agt);
    agents.addElement(agt);
    agt.moveRandomly();
}
```

The above ecosystem has the structure of an open food chain : carnivores chase herbivores that chase plants that chase no one. A more biologically realistic implementation defines a **closed ecological chain**, that is similar to the previous one but possesses the additional feature that the plants do not chase actively but consume passively some form of food found in their immediate surroundings. This food comes from the decomposition of dead organisms of all three species and we call it **remains**. To model this closed ecological model of populations, the three species need one additional energetic parameter, **dp (death product)**, that denotes the amount of food that is released to the environment in the agent’s position at death. Moreover, plants need two new behavioral parameters, **Rl (Root Length)** and **Rt (Root Threshold)**. The first denotes the length of the “root” of the plant, i.e. how far it can reach for food, whereas the second is used together with **Ef** and denotes the amount of food that has to be found in the vicinity of the plant’s position for it to gain energy (i.e. so that we may claim it “ate” something). The user may experiment with several values of the two new parameters, especially with the root length parameter (to date, neither God nor me have managed to achieve a stable ecosystem with a root length of 0, let’s see if anyone of you could do it!). The method implementing the above behavioral model, as well as the method implementing agent death and energy release to the environment, are given below. Next, a snapshot of the ecosystem is also given, with environmental remains visualized in the form of small dark squares.

```
public void execute()
{
    FoodWebAgent prey;

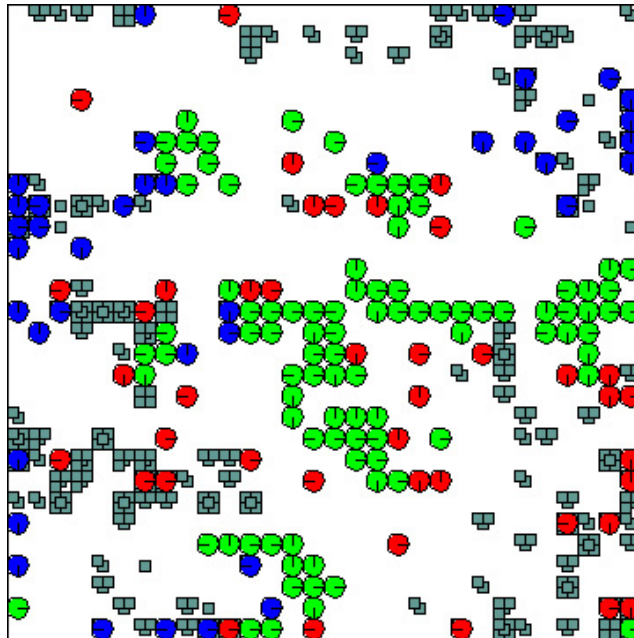
    if ((type != CARNIVORE) && (!dead)) // HERBIVORE
    {
        . . . . .
    }
}
```

```

else if ((type == PLANT) && (!dead))
{
    if (simulation.isClosedChain())
    {
        energy -= dE;
        int r = trackRemains();
        if (r >= rootThreshold)
        {
            energy += Ef;
            consumeRemains(rootThreshold);
        }
        if (energy >= Er)
        {
            energy /= 2;
            simulation.divide(this);
        }
        else if (energy <= 0) die();
    }
    else simulation.divide(this);
}
}

public void die()
{
    dead = true;
    int s = deathProduct;
    if (simulation.isClosedChain())
        simulation.addRemainsAt(position.x,position.y,s);
}

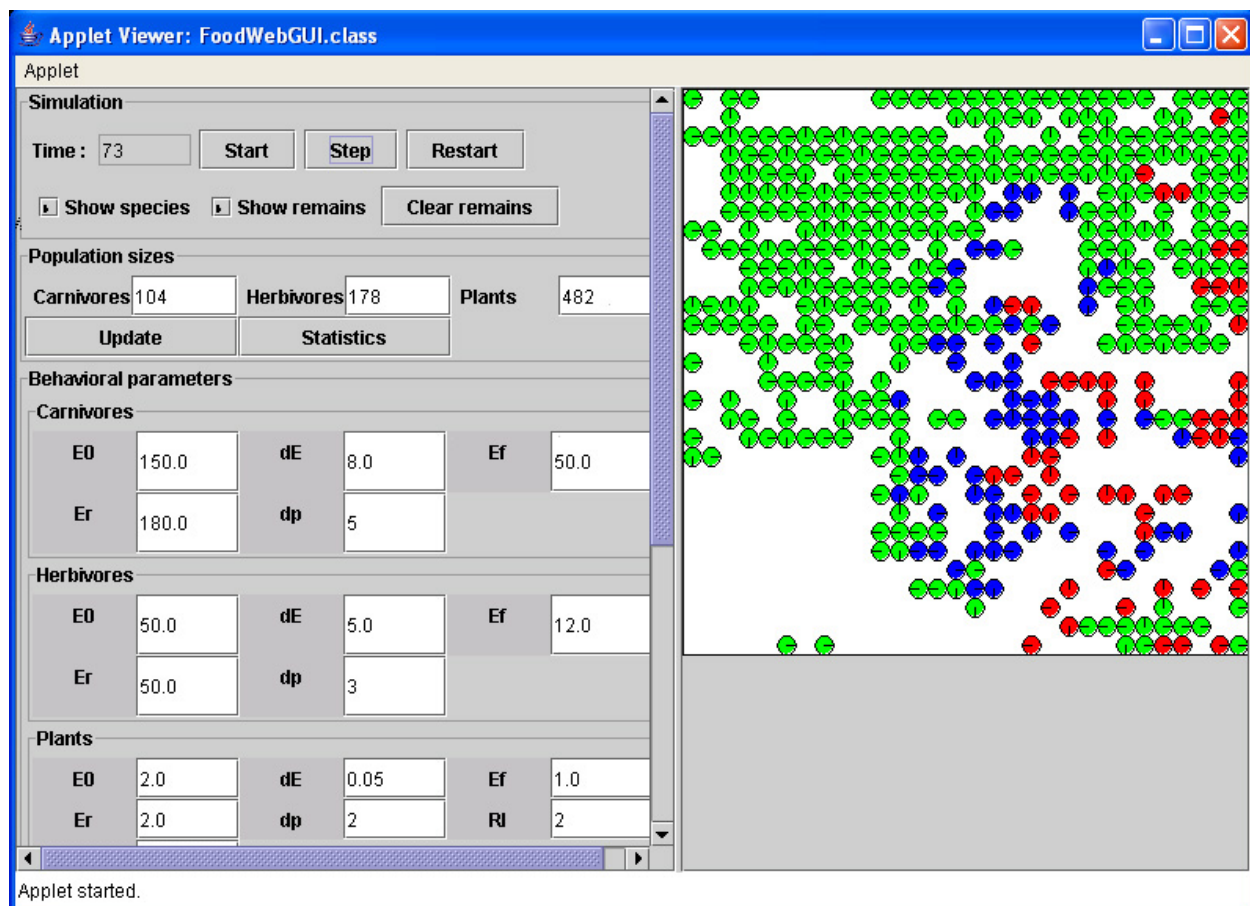
```



The energetic behavioral parameters of the two above models (open and closed chain) are given comprehensively in the following table.

Parameter	Carnivores	Herbivores	Plants
E0	√	√	Closed world only
dE	√	√	Closed world only
Ef	√	√	Closed world only
Er	√	√	Closed world only
Death product (dp)	Closed world only	Closed world only	Closed world only
Root length (RI)	-	-	Closed world only
Root threshold (Rt)	-	-	Closed world only

3 Implementation of the testbed



3.1 User's manual

The applet's screen shows a white area in the right where lie the simulated organisms (red carnivores, blue herbivores and green plants). By double-clicking on a position, the user may examine in a popup dialog how many and which agents exist in this position. The left side of the applet includes a number of simulation control components.

- **The simulation clock.**

- **Three simulation control buttons.**

The **Start/Stop** button starts/stops the simulation and in every cycle executes all the agents. The **Step** button advances the simulation one cycle further. Finally, the **Restart** button restarts the simulation, i.e. reinitializes the clock to 0 and resets all agents and statistics observers.

- **Two checkboxes “Show species” and “Show remains”.**

Those define whether the agents or the environmental remains respectively will be visible during simulation.

- **Population sizes of the three species.**

The three textfields give the current population sizes. The user may manually type in new sizes for one or more of the species and reinitialize the overall ecosystem with the **Update** button. The **Statistics** button is used to retrieve the statistics of the three population sizes. The following figures give a few population examples : stable open ecosystem, ecosystem where all mobile agents die, exponentially growing ecosystem without carrying capacity for mobile agents, stable closed ecosystem.

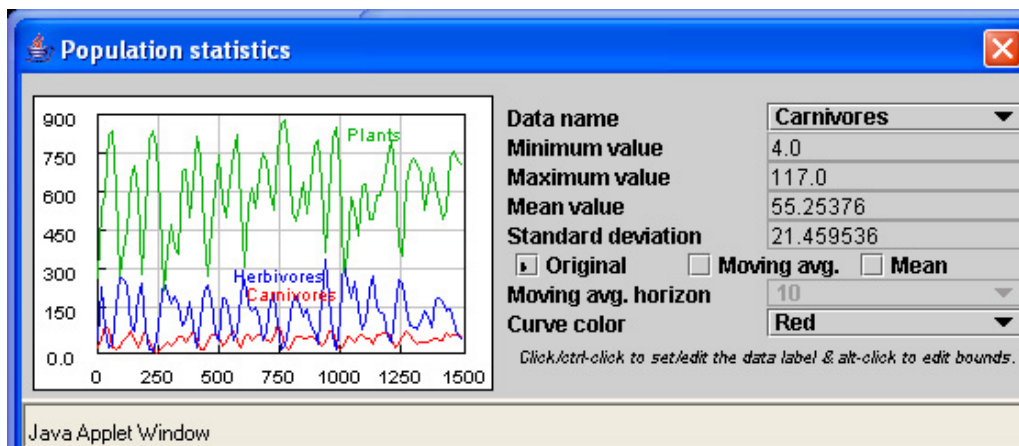
- **Behavioral (energetic) parameters for the three species.**

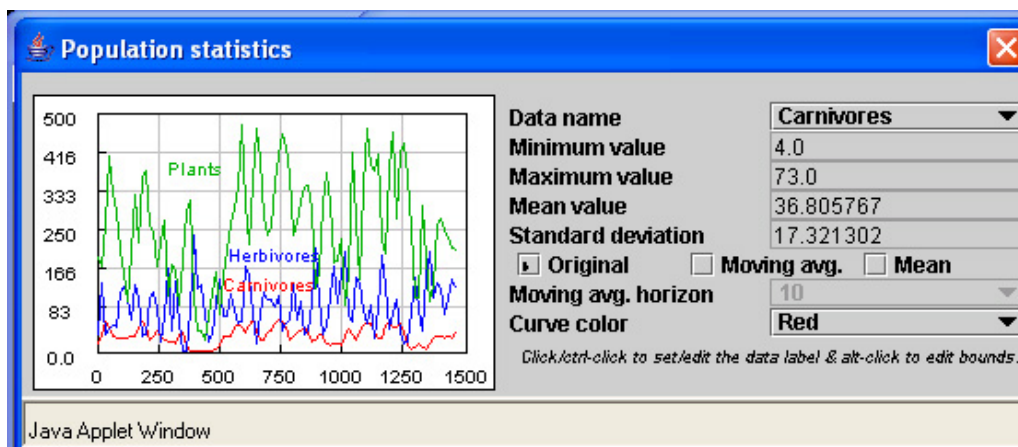
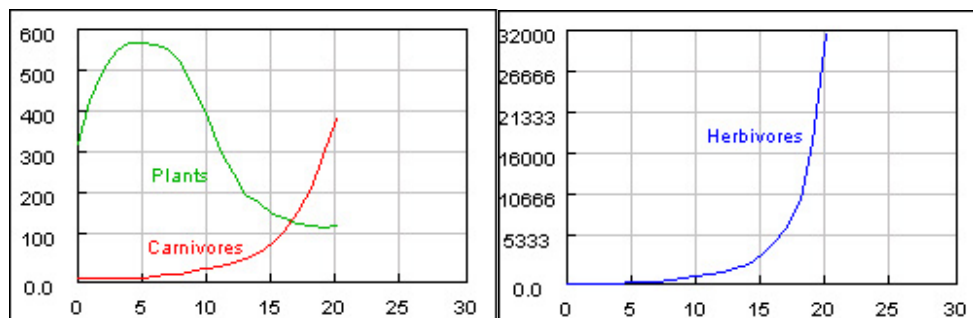
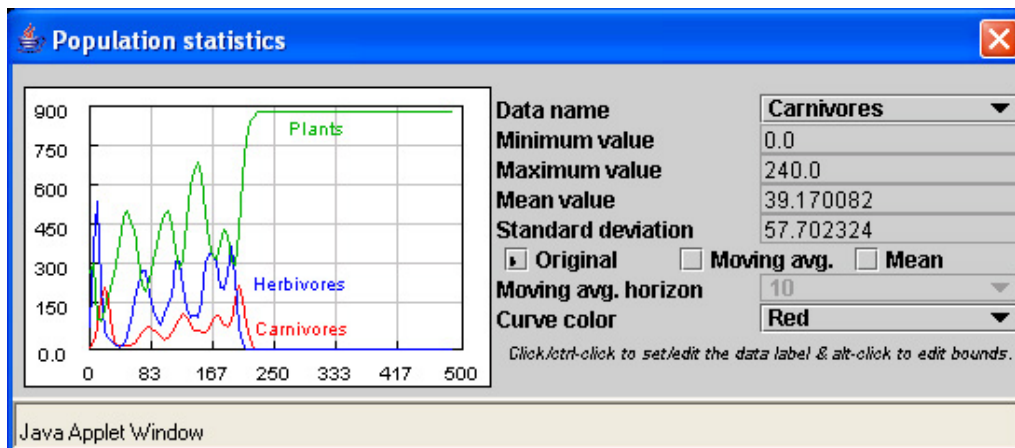
The five textfields per species give the corresponding values for **E0**, **dE**, **Ef**, **Er**, **dp** parameters, as described above. Especially for the plants, two additional textfields are given for the values of the **Rt** and **Rl** parameters. The user may manually type in new values for one or more of those parameters and reinitialize the overall system with the **Update** button.

- **Ecological parameters of the system.**

Three textfields are given for the carrying capacities described before, as well as two checkboxes to show whether carrying capacities are used in the current ecological system. The **Closed food chain** checkbox shows whether the ecosystem is closed or not, as described above. The **Reset energies** checkbox shows if there is energy reinitialization on agent reproduction (“division”) or not, as described above. The user may manually change one or more of these values and reinitialize the overall ecosystem with the **Update** button.

The above features of the testbed allow the user to experiment systematically with several variants of the trophic chain, in energetic and/or ecological settings of his choice.





4 Instructions

The testbed includes four types of classes :

- **FoodWebGUI class.** This is the class implementing the user interface (applet) and the manipulation features for the simulated ecosystem. The user does not need to know the internals of this class, but rather it uses it as such.
- **FoodWeb class.** This is the class implementing the simulated ecosystem itself and connects the interface to the agents (this class is responsible for executing the simulation, gathering and visualizing statistics, manipulating behavioral and ecological parameters and updating the interface). The user does not need to know the internals of this class, but rather it uses it as such.
- **FoodWebAgent class.** All simulated organisms are instances of this class.

- **Auxiliary classes.** Those are the classes Observer, History, Curve, CurveViewer, ListModifierDialog, ListModifierListener and TextAreaDialog, that implement the statistical data recording as a time series that may be translated into a curve and visualized as a graph. Another auxiliary class is the ListDialog class that is used to visualize the list of organisms in a particular environmental position. The user does not need to know the internals of these classes, but rather it uses them as such. Only the classfiles are given for these classes.

What to do

1. Experiment to find a set of behavioral parameters for the three species that yields equilibrium (successive ecological cycles) for at least 1500 time steps in the default ecological environment (open chain, default carrying capacities). Also experiment to find a second set that yields equilibrium for at least 1500 cycles and compare the equilibrium values for the two populations. Can you derive general conclusions on the relative equilibrium values of the populations based on only the energetic parameters sets ?

Hint. Starting from an initial behavioral parameter set that makes the chain disintegrate (at least one species goes extinct), study which species dies out (first) and which parameter you need to modify and how so that this species could manage to survive. Note that in some cases it is possible to control such a system by manipulating a parameter that adds to the aggressivity of the species that chases the sensitive species under consideration.

Deliverable. Description text including tables for the behavioral parameters used and result curves.

2. For the above behavioral parameter sets, study the effect of the **carrying capacities** parameters. For which values of carrying capacities does the system converge to a new equilibrium and how should the behavioral parameters be tuned so as to achieve stability ? Also, give an interpretation of the relative equilibrium values for the populations that you studied and compared so far.

Deliverable. Description text including tables for the behavioral parameters used and result curves, as before.

3. Study the effect of the **reset energies** parameter. Is there a difference in the potential of an ecosystem to achieve equilibrium ? For instance, can a behavioral parameter set, that could not yield stability before, now become stable if reset energies = true or a former stable system now destabilize ? Compare respectively the relative equilibrium values of the populations for the two parameter set values.

Deliverable. Description text including tables for the behavioral parameters used and result curves, as before.

4. Respectively, study the above system in a closed world setting (**closed food chain = true**). Especially study and try to interpret the relative equilibrium values of the populations in an open and a closed world, as well as the type of obtained emergent formations (stable patterns, waves or what else ?). Do the closed world formations depend on the remains formations and, if yes, how ?

Hint. Because a closed world is much more computationally intensive than an open one (the environmental remains should be updated in every cycle), the simulation should be better operated with **show remains=off**. You can however suspend the simulation at regular intervals and visualize the remains. Also, do not forget at every parameter or population reinitialization to clear the previous environmental remains (with the **Clear**

remains button) so as not to bias the new results.

Deliverable. Description text including tables for the behavioral parameters used and result curves, as before. Also comparative snapshots of formations obtained.

5. **Optionally (judgement question).** How can you interpret the relation between behavioral and ecological parameters and equilibrium potential or other features of the ecosystem ? How do you think that such an ecosystem could self-regulate its behavioral parameters and actively seek equilibrium ?

Deliverable. Text.

4.1 Description of the *FoodWebAgent* class

FoodWebAgent class defines the following primary data:

- **Point position**
- **Point heading**
The position and the heading (motion direction) of the agent in the environment.
- **int type**
The organism type. One of the following.
- **final static int CARNIVORE**
- **final static int HERBIVORE**
- **final static int PLANT**
The three ecological species described above.
- **double energy**
The agent's energy. It is updated in every simulation cycle.
- **boolean dead**
Variable showing if the organism died in the current cycle (so that it can be removed from the environment).
- **double E0**
- **double dE**
- **double Er**
- **double Ef**
- **int deathProduct**
- **int rootLength**
- **int rootThreshold**
The energetic (behavioral) agent parameters as described above.

The most important methods that the *FoodWebAgent* class defines are the following :

- **public boolean isCarnivore()**
- **public boolean isHerbivore()**
- **public boolean isPlant()**
They probe the type of the organism.

- **public boolean isDead()**
It checks whether the organism died in the current cycle (so that it can be removed from the environment).
- **public void resetEnergy()**
It initializes the organism energy to **E0**.
- **public void makeCloneOf(FoodWebAgent)**
It initializes an organism with the same behavioral parameters as its parent, which is passed as an argument.
- **public void reinitializeOn(Vector)**
- **public void reinitializeOn(double,double,double,double,int,int,int)**
It initializes the organism with the set of behavioral parameters passed as arguments.
- **public void execute()**
It executes the behavioral model of the organism. See description above.
- **protected void chasePrey()**
It implements the prey hunting of “mobile” agents (the –mobile– carnivore chases –mobile– herbivores that chase –immobile– plants).
- **public void die()**
It implements an organism’s death. In a closed ecological chain, it releases food remains in the environment as described above.
- **public void moveRandomly()**
- **public void moveRandomlyTail(Point)**
They implement the random motion of agents.
- **private int trackRemains()**
- **private void consumeRemains(int)**
They implement respectively the search for and consumption of food (environmental remains) by the plants. In the latter case, the amount to be consumed is passed as an argument.

See also ...

Across Trophic Level System Simulation Project (ATLSS), “Spatially Explicit Species Index (SESI)” models @ Univ.of Tennessee

<http://atlss.org/>

Simon Levin’s publications on biological dynamics, diversity and complexity

<http://www.eeb.princeton.edu/~simon/Abstracts.html>

Individual based modeling resources, by Craiug Reynolds. some putdated links.

<http://www.red3d.com/cwr/ibm.html>