Artificial Societies — Part One

Growing Artificial Societies
Epstein and Axtell, Brookings Institution Press, 1996

Why is the *new* social science a model-based simulation approach? It provides a methodology for studying

- complex processes *together*;
- *heterogeneous* populations and *non-equilibrium* dynamics;
- *rational actors* to represent humans as members of society;
- how heterogeneous behavior of *individuals* generates the macroscopic behavior of *society*.

This methodology departs dramatically from traditional approaches

- in the *way* types of social behavior are treated (combat, trade, cultural transmission);
- in the way types of social behavior are *combined*.

Artificial Society

An "agent-based" model used to study social processes

- interaction with environment
- population dynamics (reproduction, birth, death)
- trade (economic interaction)
- migration
- group formation
- combat
- transmission of culture
- propagation of disease

A *computational laboratory*

- "societies" grown in simulation environment
- attempt to discover *micro*-scale mechanisms sufficient to generate *macro*-scale behavior

Agents

Agents are the "people" in an artificial society

- each agent has *attributes* and *states*
- the attributes are *static*, fixed for the lifetime of the model (*e.g.*, metabolic rate, vision, gender)
- the states are *dynamic*, changing in time (*e.g.*, location, wealth, health)

Landscape

The environment *in* which agents operate, *with* which they interact

- a 2D grid of cells containing renewable resources
- each cell has *attributes* and *states*
- the attributes are *static*, fixed for the lifetime of the model (*e.g.*, capacity, regrowth rate)
- the states are *dynamic*, changing in time (*e.g.*, resource level)

Behavioral Rules

Three types of behavioral rules govern the agents and landscape

- *Agent-Agent (a-a)*: governs interaction of one agent with another (*e.g.*, mating, combat, trade)
- *Environment-Environment (e-e)*: governs interaction of one cell with neighboring cells (*e.g.*, heterogeneous resource regrowth)
- *Agent-Environment (a-e)*: governs interaction of agent with one or more landscape cells (*e.g.*, gathering resources, creating pollution)

In the book, *e-e* rules define a *cellular automaton*

- a questionable model assumption
- potential to produce *quantization artifacts*

Building the Model

Models are built at three levels:

- conceptual (in your mind)
- specification (on paper)
- implementation (in software)

Chapter 1 of the book characterizes artificial societies conceptually

Chapter 2 moves the model toward the specification level

Temporal Coordinate System

A t coordinate system denotes time evolution of the artificial society

- t is *integer-valued* so that $t = 0, 1, 2, \ldots$
- forces all agents to act simultaneously (in parallel)
- some (perhaps artificial) conflict resolution rule must be imposed
- forced synchronous movement can produce simulation artifacts

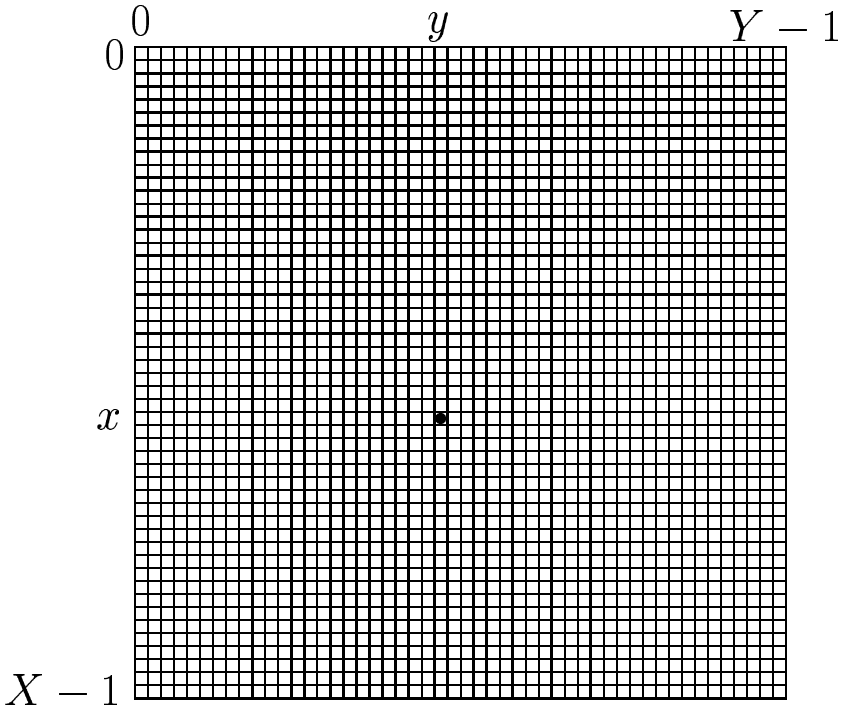
The parameter T denotes the maximum number of time steps

Landscape Coordinate System

An (x, y) integer-valued spatial coordinate system identifies cells

Rotate the coordinate system 90 degrees relative to convention

- x iterates rows, $\downarrow$
- y iterates columns, $\rightarrow$



Characterizing the Landscape

The landscape is an $X \times Y$ array so that

$$x = 0, 1, 2, \ldots, X-1 \qquad \text{and} \qquad y = 0, 1, 2, \ldots, Y-1$$

(For all figures in Chapter 2, $X = Y = 50$)

Each (x, y) landscape cell is characterized by:

- $o(x, y)$: *occupancy status* of the cell — 1 if occupied, 0 otherwise
- $r(x, y)$: current *resource level* at the cell
- $\rho(x, y)$: resource *regrowth rate* of the cell
- $\gamma(x, y)$: resource *capacity* of the cell

State variables: $o(x, y)$ and $r(x, y)$
Attributes: $\rho(x, y)$ and $\gamma(x, y)$

In the book, $r(x, y)$, $\rho(x, y)$, and $\gamma(x, y)$ are integer-valued — why?

Characterizing the Landscape (Cont.)

Each (x, y) landscape cell is characterized by four states/attributes

- $o(x, y)$: naturally Boolean (0 or 1)
- $r(x, y)$: real-valued, non-negative
- $\rho(x, y)$: real-valued, non-negative
- $\gamma(x, y)$: real-valued, non-negative

In the book, $r(x, y)$ is integer-valued with no real justification

$\rho(x, y)$ and $\gamma(x, y)$ should agree in type with $r(x, y)$

Living on a Doughnut

"Mmmm, doughnuts. Is there *anything* they can't do?"

— Homer Simpson

"...the [landscape] wraps around from right to left ...and from top to bottom, forming a doughnut — technically a torus."

- equivalently, periodic boundary conditions are used
- any (x, y) cell beyond the boundaries is equivalenced to a cell within the boundaries via $(x \bmod X, y \bmod Y)$
- *e.g.*, if $X = Y = 50$, then

$$(51, 50) \Longleftrightarrow (1, 0) \quad \text{and} \quad (-1, 5) \Longleftrightarrow (49, 5)$$

In ANSI C, $x \,\%\, X$ may be inconsistent with math definition if $x < 0$

$$\text{use } (x + X) \,\%\, X \text{ in place of } x \,\%\, X$$
$$\text{use } (y + Y) \,\%\, Y \text{ in place of } y \,\%\, Y$$

Landscape Resource Capacity

The book uses only one capacity model, but never defines that model

The model appears to be the two-peak Gaussian

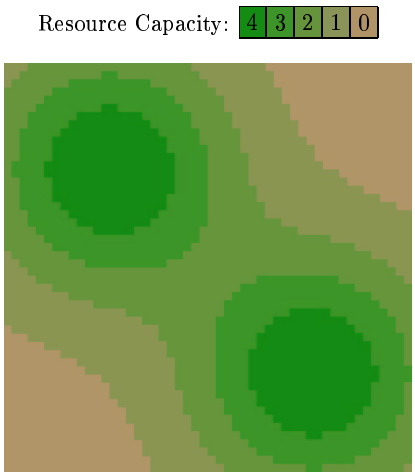
$$\gamma(x, y) = f(x - X/4,\, y - Y/4) + f(x - 3X/4,\, y - 3Y/4)$$

where

$$f(x, y) = \psi \exp\left(-(x/\theta_x)^2 - (y/\theta_y)^2\right)$$

with $\psi = 4.0$, $\theta_x = 0.3X$ and $\theta_y = 0.3Y$

- Is this equation correct?
- If so, are θ_x and θ_y correct?
- Note $\gamma(x, y)$ is integer-valued in the book



Characterizing the Agents

In accordance with behavioral rules, agents move about the landscape gathering and consuming resources

At any time t, each agent is characterized by

- $x(t)$: *vertical* location at time t
- $y(t)$: *horizontal* location at time t
- $w(t)$: resource wealth at time t
- μ: metabolic rate
- ϕ: vision

State variables: $x(t)$, $y(t)$, and $w(t)$
Attributes: μ and ϕ

In the book, $w(t)$ and μ are integer-valued — why?

Characterizing the Agents (Cont.)

Each agent is characterized by five states/attributes:

- $x(t)$ (row index): naturally integer-valued, non-negative
- $y(t)$ (column index): naturally integer-valued, non-negative
- $w(t)$: real-valued, non-negative
- μ: real-valued, non-negative
- ϕ: naturally integer-valued, positive

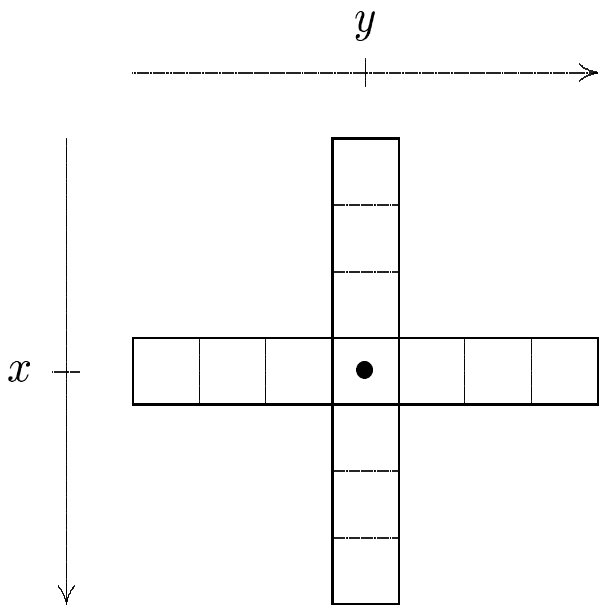
$w(t)$ and μ should agree in type with $r(x, y)$

Typically, $\phi \ll \min\{X, Y\}$

Agent Field of View (FOV)

An agent with vision ϕ can see ϕ cells in the four primary directions

This collection of 4ϕ cells defines an agent's *field of view* (FOV)



e.g., if $\phi = 3$ the agent can move *at most* 3 units N, S, E, or W

Landscape Regrowth Rule

Rule **G**: at each (x, y) cell, resource regrows at a rate of $\rho(x, y)$ units per time interval up to cell capacity

- an *e*-*e* rule
- regrowth rate $\rho(x, y)$ is an attribute
- no inherent reason why $\rho(x, y)$ can't depend on cell location (like $\gamma(x, y)$)

In other words, the resource level at cell (x, y) at time $t + 1$ is

$$\min \{\gamma(x, y),\ r(x, y) + \rho(x, y))\}$$

where $r(x, y)$ is the resource level at time t

$O(XY)$ time complexity at *each* time step

Agent Movement Rule

Rule **M**: an *a-e* rule

- Look at all cells within FOV defined by ϕ
- Select closest unoccupied cell with maximum resource (break ties randomly)
- Move to the selected cell and collect all the resource

If at time t

- agent is at cell (x, y)
- agent possesses resource wealth $w(t)$
- closest unoccupied cell with max resource is (x', y')

then at time $t + 1$

- agent moves to (x', y')
- $w(t+1) = \max\{0,\ w(t) + r(t+1) - \mu\}$
- if $w(t+1) = 0$, the agent dies

$O(\phi A)$ complexity at *each* time step, where A is number of agents

Movement Rule Considerations

Agents move simultaneously (in parallel)

- each agent moves only once per time step

At most one agent can occupy a cell at any time

What if two agents try to move to the same cell simultaneously?

- destroy the parallel structure by forcing a sequential process
- agent order must be randomized to ensure against artifacts

Random Number Generation

Random number generation is based on the function `Random()`:

- provided in the library `rngs.c` in ~`bglaws/umsa/c`
- returns a real-valued number u
- u is equally likely to be any value in the interval $0.0 < u < 1.0$

Discrete states/attributes are modeled as $Equilikely(a, b)$

- provided in `rvgs.c` as `Equilikely(long a, long b)`
- returns an integer-valued number (`long`)
- equally likely to be any integer in the interval $[a, b]$
- each integer value has probability $1/(b - a + 1)$

Continuous states/attributes are modeled as $Uniform(a, b)$

- provided in `rvgs.c` as `Uniform(double a, double b)`
- returns a real-valued number (`double`)
- equally likely to be any real value in the interval (a, b)
- each real value has probability density $1/(b - a)$

Data Structures

What data structures are needed at the implementation level?

- the landscape is an $X \times Y$ *array* of structs of the form

```
typedef struct {
  bool o;          /* occupancy (state)          */
  double r;        /* resource level (state)     */
  double γ;        /* capacity (attribute)       */
  double ρ;        /* regrowth rate (attribute)  */
} Cell_Type;
```

- the agents are stored as a *list* of structs of the form

```
typedef struct {
  int x;           /* row index (state)          */
  int y;           /* column index (state)       */
  double w;        /* resource wealth (state)    */
  double μ;        /* metabolic rate (attribute) */
  int φ;           /* vision (attribute)         */
} Agent_Type;
```

Initializing the Landscape

The landscape is initialized as follows:

- the resource capacity $\gamma(x, y)$ is specified
- for each (x, y) the initial resource level is set to $\gamma(x, y)$
- for each (x, y) the regrowth rate $\rho(x, y)$ is specified

In Chapter 2, $\rho(x, y)$ is the same for all (x, y) ($\rho = \infty$ or $\rho = 1$)

The following algorithm initializes the landscape

```
Cell_Type cell[X][Y];  /* landscape */

int x, y;
for (x = 0; x < X; x++)
  for (y = 0; y < Y; y++) {
    cell[x][y].o = 0;
    cell[x][y].ρ = 1.0;
    cell[x][y].γ = Capacity(x,y);
    cell[x][y].r = cell[x][y].γ;
  }
```

where `Capacity(`x`,`y`)` returns the resource capacity at (x, y)

Additional heterogeneity *could* be added to the model by creating $\rho(x, y)$ randomly for each (x, y)

Initializing the Agents

The agents are initialized as follows:

- an initial number of agents A is specified
- the initial distribution of A agents on the landscape is random
- each agent is given an initial endowment $w(t) = \mathit{Uniform}(5.0,\ 25.0)$
- each agent is given a metabolic rate $\mu = \mathit{Uniform}(1, 4)$
- each agent is given a vision $\phi = \mathit{Equilikely}(1, 6)$

We can initialize the agent list

```
Agent_Type agent[A];  /* agent list */
```

with exactly one call to `Random()` for each agent

- Draw A integers from $\{0, 1, 2, \ldots, XY - 1\}$ at random without replacement
- Map each integer a drawn to a unique (x, y) cell via

```
x = a / Y;      /* 0 <= x < X */
y = a % Y;      /* 0 <= y < Y */
```

What happens if XY is huge?

What data structure is best for the agent list, especially for large A?

Initializing the Agents (Cont.)

Although not ideal, *acceptance/rejection* is appropriate here

```
int x, y;
int a = 0;      /* agent index */
while (a < A) {
  x = Equilikely(0, X - 1);
  y = Equilikely(0, Y - 1);
  if (cell[x][y].o == 0) {
    cell[x][y].o = 1;
    cell[x][y].r = 0.0;     /* convention */
    agent[a].x   = x;
    agent[a].y   = y;
    agent[a].w   = Uniform(5.0, 25.0);
    agent[a].μ   = Uniform(1.0, 4.0);
    agent[a].φ   = Equilikely(1, 6);
    a++;
  }
}
```

What is the expected number of calls to `Random()` per agent?

How does this expected number depend on agent density A/XY?

More Algorithms

What other algorithms are needed at the implementation level?

For each $t = 1, 2, \ldots$, four algorithms are used in the following order:

- Move the agents consistent with rule **M**
- Update the landscape consistent with rule **G**
- Update the agent list, check for any agent deaths, and modify the agent list accordingly
- Shuffle the agent list to randomize the *sequential* order in which agents will move in the next time step

Carrying Capacity

If it exists, the statistic

$$\lim_{t \to \infty} E[A(T)]$$

is called the *carrying capacity* of the landscape

- based on the ecological principal that an environment can support only a finite population
- a steady-state (infinite horizon) statistic

How large does T need to be to estimate this statistic?

How many initial agents are required to estimate without bias?

Can some *flow balance* equation explain the carrying capacity?

Characterizing the Population Size

$A(T)$ denotes the number of agents at time T

The best characterization of $A(T)$ uses *replication*

- if the number of replications is small, compute a 95% confidence interval estimate for $E[A(T)]$
- if the number of replications is large, a histogram can be used to estimate the pdf of $A(T)$

Evolution of the Model

Given the parameter T and state variable `alive` where

- T denotes the number of simulated time steps
- `alive` counts the number of living agents

the artificial society evolves in time according to

```
/* initialize the landscape */
/* initialize the agents    */
alive = A;
t = 0;
while (t < T) {
  /* move the agents                      */
  /* update the landscape                 */
  /* update the agent list                */
  /* randomize (shuffle) the agent list */
  t++;
}
/* output the value of 'alive' */
```

For each update of the agent list, `alive` is decreased appropriately

Agent Reproduction

Is it meaningful to study a fixed population of agents that live forever (unless they starve)?

We need agents that are born, mature, reproduce, and die of old age

The following agent attributes are needed

- ϵ: initial wealth (endowment)
- σ: gender (either male or female)
- β: time of birth
- λ: lifespan
- α: age when reproductive capability begins
- ω: age when reproductive capability ends

Death occurs naturally at $t = \beta + \lambda$ unless starvation precedes

Characterizing the Agent (Cont.)

Each agent is now characterized by the following six attributes (in addition to the original five states/attributes)

- ϵ: real-valued, positive
- σ: Boolean
- β: integer-valued, non-negative
- λ: integer-valued, positive
- α: integer-valued, positive
- ω: integer-valued, positive

ϵ must agree in type with $w(t)$

β, λ, α, and ω must agree in type with t

Agent Reproduction Rule

At time t, an agent is *fertile* if the following are true

- the agent is of childbearing age, i.e., $\alpha \leq t - \beta < \omega$
- the agent's wealth $w(t)$ is at least as large as its initial wealth ϵ

Rule **S**: at *each* time step, a fertile agent will attempt to reproduce as follows

- select one of the four nearest-neighbor agents (if any) of the opposite sex from the von Neumann neighborhood
- the selected neighbor agent is a *candidate* for mating if
  - the selected neighbor agent is fertile
  - there is an unoccupied cell (for the child) within the von Neumann neighborhood of either agent
- repeat until the candidate with maximum wealth is found or the search is exhausted

If a mate is found, mating and subsequent birth occur immediately

Neither parent agent will attempt to reproduce again during the current time step

Inheritance

A child agent acquires state/attribute values from its parents

- σ is determined via a fair coin flip
- ϵ is the average of its parents' initial wealths
- $w(\beta) = \epsilon$
- except for ω, which is gender specific, the remaining attributes $(\mu, \sigma, \lambda, \alpha)$ are equally likely to be inherited from either parent

Accordingly, each parent agent's wealth decreases by half of its own initial wealth

Are other inheritance rules more appropriate?

Initializing the Agents

The book suggests the following stochastic models for initializing the agent list at time $t = 0$

- for all agents, ϵ is *Uniform*(50.0, 100.0)
- for all agents, σ is *Equilikely*(0,1)
- for all agents, λ is *Equilikely*(60, 100)
- for all agents, α is *Equilikely*(12, 15)
- for female agents, ω is *Equilikely*(40, 50)
- for male agents, ω is *Equilikely*(50, 60)
- for all agents, μ is *Uniform*(1.0, 4.0)
- for all agents, ϕ is *Equilikely*(1, 6)

Evolution of the Model (With Reproduction)

Given the parameter T and state variable `alive` where

- T denotes the number of simulated time steps
- `alive` counts the number of living agents

the artificial society evolves in time according to

```
/* initialize the landscape */
/* initialize the agents    */
alive = A;
t = 0;
while (t < T) {
  /* move the agents                      */
  /* reproduce the agents     <=== NEW    */
  /* update the landscape                 */
  /* update the agent list                */
  /* randomize (shuffle) the agent list */
  t++;
}
/* output the value of 'alive' */
```

For each update of the agent list, `alive` is increased or decreased appropriately as agents are born and/or die