Artificial Societies — Part Two

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

Part one of these notes was based on Chapters $1, 2, 3$ in the book, with suggestions for modifications and extensions.

This (part two) set of notes discusses:

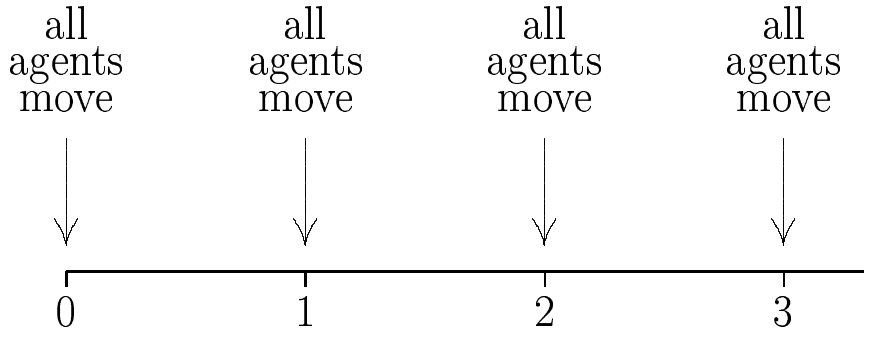
- asynchronous time evolution
- associated behavioral rule modifications
- implementation of asynchronous time
- event list considerations

Synchronous Time Evolution

Events are synchronized to a simulation clock that advances deterministically at a fixed rate.

Time is discrete, conventionally denoted $t = 0, 1, 2, \ldots, T$.

All events of interest occur precisely at these time steps.



Randomization is necessary to avoid artifacts.

Synchronous time is unnatural in many applications.

Synchronous Time Algorithm

```
/* initialize the landscape */
/* initialize the agents    */
t = 0;
while (t <= T) {
  /* move and reproduce the agents      */
  /* update the landscape               */
  /* update the agent list              */
  /* randomize (shuffle) the agent list */
  t++;
}
```

Ambiguity in the order of events:

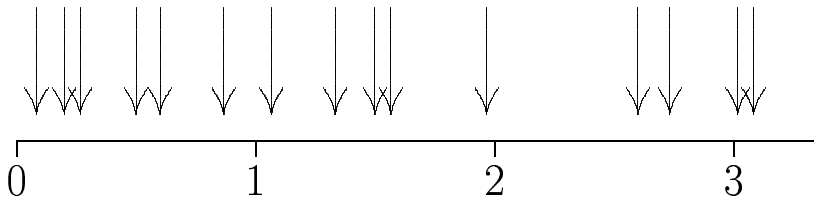
- Should the landscape or agent list be updated first?
- Should the landscape be updated before agents move?

Asynchronous Time Evolution

Each type of event occurs at random at its own characteristic rate.

Time is continuous (real-valued).

Individual agent events occur at distinct times.



Randomization of agent events is inherent.

Ambiguity in the order of events is removed.

The Model Revisited

Model attributes:

T : maximum simulated time for the model

X, Y : vertical and horizontal landscape dimensions

Model variables:

t : time $(0 < t < T)$

$A(t)$: number of agents at time t

Define $\mathcal{X} = \{0, 1, \ldots, X-1\}$ and $\mathcal{Y} = \{0, 1, \ldots, Y-1\}$.

We use subscript $a = 0, 1, \ldots, A(t)-1$ to distinguish agents.

The Landscape Revisited

Each cell in the $X \times Y$ landscape is distinguished by its (x, y) position with $(x, y) \in \mathcal{X} \times \mathcal{Y}$.

Landscape attributes:

$\gamma(x, y)$: resource capacity at cell (x, y)

$\rho(x, y)$: resource regrowth rate at cell (x, y)

Landscape states:

$r(x, y, t)$: level of resource at cell (x, y) at time t

$d(x, y)$: most recent time of resource depletion at cell (x, y)

$o(x, y, t)$: occupancy status of cell (x, y) at time t

The state $d(x, y)$ is new — used for resource regrowth.

The resource level and occupancy status at a cell (x, y) are functions of time t.

The attributes $\gamma(x, y)$ and $\rho(x, y)$ remain as previously defined.

The Agents Revisited

Agent attributes:

- ϕ : field of view (FOV) attribute
- μ : metabolic rate of resource consumption
- ϵ : initial wealth (endowment)
- σ : sex (male or female)
- β : time of birth
- λ : lifespan
- α : age when reproductive capability begins
- ω : age when reproductive capability ends ($\alpha < \omega$)
- η : gestation period (for females)

Agent states:

- $w(t)$: amount of resource wealth at time t
- (x, y) : landscape cell occupied by the agent
- $m(t)$: pointer to current mate (if any)

The attribute η and state $m(t)$ are new — used in a modified reproduction rule.

The Behavioral Rules Revisited

Resource regrowth and agent movement rules remain as previously defined.

Reproduction is modified to be more realistic, incorporating a gestation period.

For agent a to be fertile at time t

- the agent must be of childbearing age

$$\alpha \leq t - \beta < \omega$$

- $m_a(t)$ must be undefined
- the agent must not die naturally before the end of the gestation period,

$$\beta + \lambda > t + \eta$$

- the agent's wealth at the end of the gestation period must be at least as large as its initial endowment

$$w(t + \eta) \geq \epsilon$$

The Reproduction Rule Revisited

The selection of a candidate mate remains the same.

The remainder of the reproduction process is defined as follows.

- If a mate is found, the female agent becomes pregnant.
- Throughout the gestation period, neither parent can move or attempt to reproduce.
- At the time of birth $(t + \eta)$, the unoccupied cell with maximum resource is selected for the child from the union of each parent's set of adjacent cells.
- If no such cell is available, the child dies at time $t + \eta$.

Next-Event Simulation

Next-event simulation is used to facilitate asynchronous time evolution.

For a next-event simulation model, we must clearly define

1. the simulation clock
2. the system state
3. the event types
4. set of algorithms for each event type that define state changes

Simulation clock: real-valued time parameter t

System state: at any time $0 < t < T$, defined by

- $A(t)$
- $o(x, y, t)$ and $r(x, y, t)$ and $d(x, y)$ for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$
- $(x, y)_a$ and $w_a(t)$ and $m_a(t)$ for all $a = 0, 1, \ldots A(t) - 1$

Event Types

The four types of events that can change the system state and their associated changes:

1. Movement and resource consumption by agent a
   - change $(x, y)_a$ and $w_a(t)$ for the agent
   - change $o(x, y, t)$ and $r(x, y, t)$ for the departed and newly occupied cells
   - change $d(x, y)$ for the departed landscape cell
2. Death of agent a
   - decrement $A(t)$
   - change $o(x, y, t)$ and $d(x, y)$ for $(x, y)_a$
3. Successful mating of agent a with agent a'
   - change $m_a(t)$ and $m_{a'}(t)$ for the agents
4. Birth of new agent a'' from parent agents a and a'
   - change $w_a(t)$, $m_a(t)$ and $w_{a'}(t)$, $m_{a'}(t)$ for the parents

   and if there is a cell available for the child
   - increment $A(t)$
   - create $(x, y)_{a''}$ and $w_{a''}(t)$ for the child
   - change $o(x, y, t)$ and $r(x, y, t)$ for $(x, y)_{a''}$

Reproduction as Two Event Types

Because of the gestation period, the system state can change when agents mate and when a new agent is born.

Therefore, reproduction is modeled as two distinct types of events:

- A *mating* event initiates the gestation period if the female becomes pregnant (pregnancy not guaranteed).
- A *birth* event signals the end of the gestation period when the child is born (if possible).

Both of these event types can change the state of the system as previously described.

The Agent Data Structure

How should the agent data structure be modified for asynchronous time evolution?

Because there are four event types, include the four associated next-event times for an agent.

```
typedef struct {
  /* agent attributes */

  /* agent states     */

  t_movement;  /* next movement event time   */
  t_mating;    /* next mating event time     */
  t_birth;     /* next childbirth event time */
  t_death;     /* time of death              */
}
```

Asynchronous Inter-Event Times

Inter-event times are assumed to be *iid* $Exponential(1/\nu)$ where ν is the rate of occurrence of the associated event type.

- stationary Poisson process with rate ν
- consistent with synchronous inter-event times, use $\nu = 1.0$

The mating event type can either be coupled with movement or modeled as a separate stationary Poisson process.

If the two are coupled, a mating event occurs for an agent each time a movement/consumption event occurs for that agent.

Asynchronous Resource Regrowth

Resource regrowth is not considered an event, but rather an inherent part of the other events.

The resource level at a cell changes as follows:

- At $t = 0$, for each (x, y) cell, $r(x, y, t) = \gamma(x, y)$.
- As time evolves, resources are depleted only when an agent occupies a cell.
- While the cell is occupied, resources continue to grow but are gathered by the occupying agent.
- At the moment an agent departs, the cell has no resources.
- As time evolves, the cell resources can then regrow up to the resource capacity, provided the cell does not become occupied again.

Resource regrowth is included as part of the movement, mating and birth events.

At time $t > d(x, y)$, the current level of resource at an (x, y) cell is defined by

$$r(x, y, t) = \min\{\gamma(x, y),\ \rho(x, y)(t - d(x, y))\}$$

Asynchronous Time Algorithm

```
/* initialize the landscape */
/* initialize the agents    */
t = 0.0;
while (t <= T && A > 0) {
  event = GetNextEvent();
  t = event.time;
  switch (event.type) {
    case movement:
      /* vacate the current cell */
      /* select the (x,y) cell within the FOV with maximum resource */
      /* occupy the selected (x,y) cell */
      /* update the wealth of the agent */
      /* schedule the next movement event */
    case death:
      A(t)--;
      /* vacate the current cell */
    case mating:
      /* execute the mating algorithm */
      if ( /* the mating is successful */ ) {
        /* compute endowments and update the wealths of the parents */
        /* update the mate pointers of each parent */
        /* schedule a birth event */
        /* cancel any parent agent events to occur before the birth */
      }
    case birth:
      /* update the wealths of the parent agents */
      /* schedule the next event occurrences for each parent agent */
      if ( /* there is an (x,y) cell for the new agent */ ) {
        A(t)++;
        /* initialize the new agent */
        /* occupy the selected (x,y) cell with the new agent */
        /* update the wealth of the new agent */
        /* schedule the next event occurrences for the new agent */
      }
  }
}
```

The Event List

Next-event simulation is driven by an *event list*.

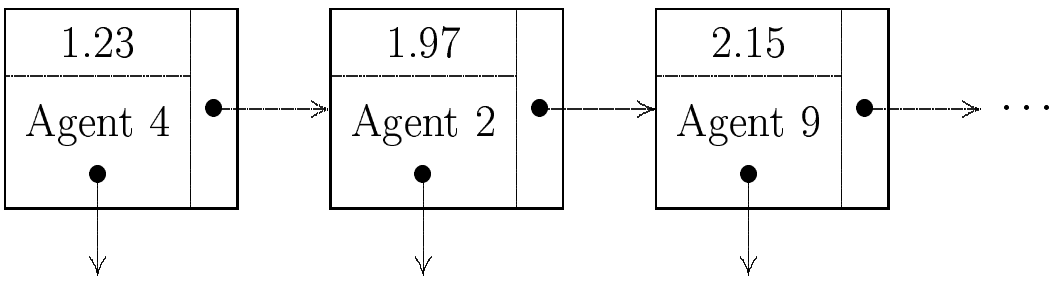
Each element is associated with the future occurrence of a single event (of one of the four event types defined) and consists of:

- the time the event is to occur;
- the type of event;
- a pointer to the corresponding agent.

Each agent can execute only one type of event at a time, so include only the most imminent event for each agent.

- One event per agent in the event list.
- The type of the event can be determined from the agent data structure.

A sample event list using a singly-linked list data structure sorted by increasing event time:

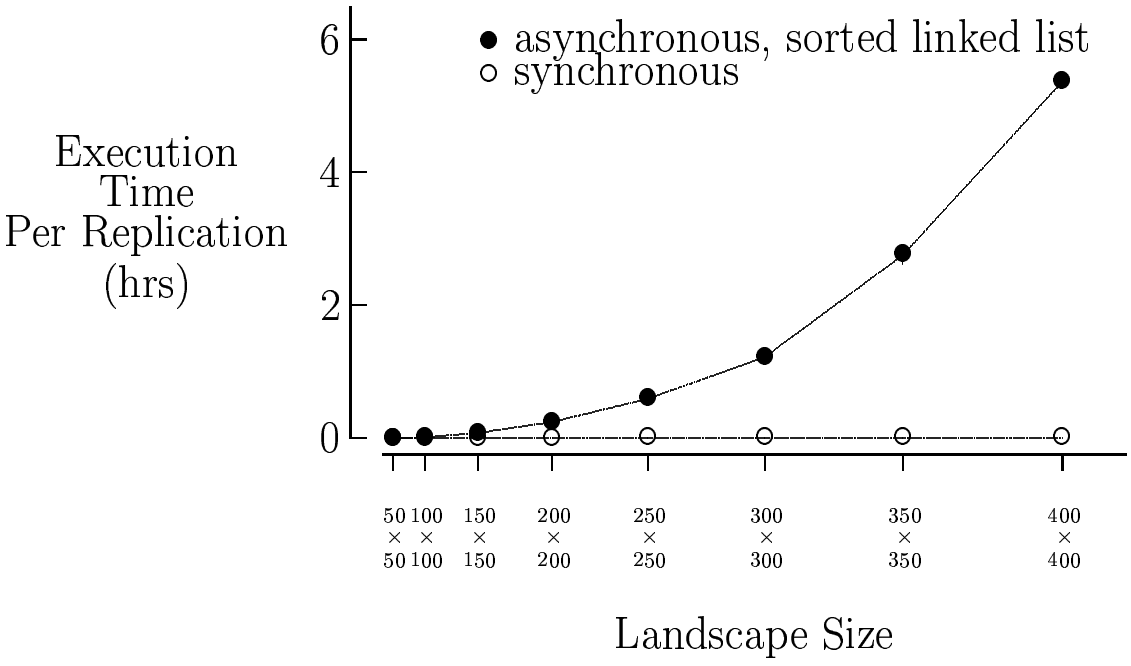


Sorted Linked-List Event List Performance

Event list operations:

- `GetNextEvent()` returns the head of the list.
- Event insertion requires a linear search from the beginning.
- Event cancellation also requires a linear search.

With $T = 500$, 0.16 initial population, and no reproduction:



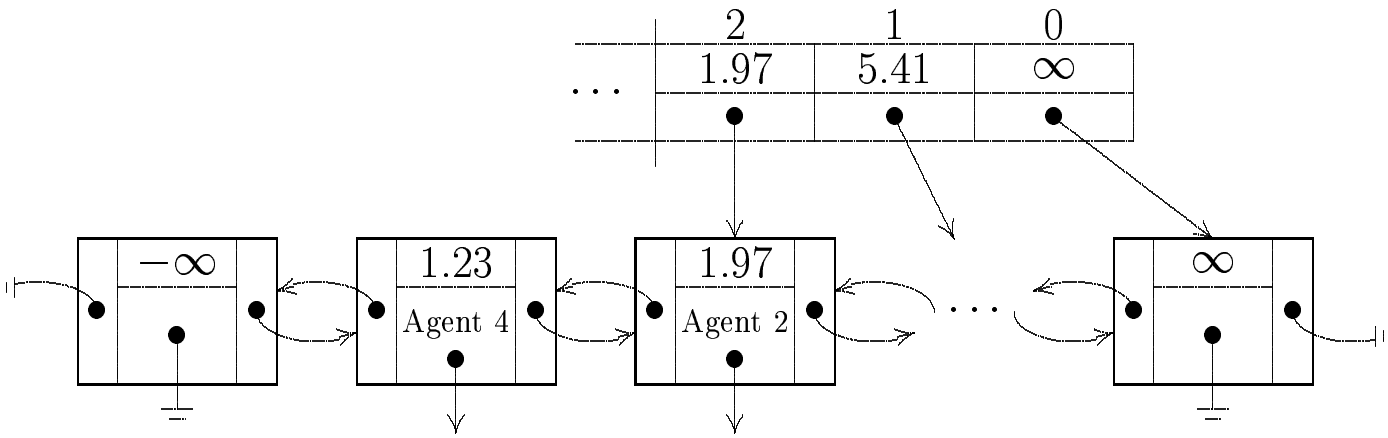
"Fat-dot accuracy" — 95% confidence intervals

The effect of landscape size on execution time is negligible.

An Improved Event List Approach

Henriksen's algorithm naturally extends the sorted linked list.

- "Dummy" events are added to the ends of the linked list.
- An auxiliary search array contains sample times from the event list, sorted by decreasing time, and pointers to the associated events.
- Complete linear searches of the event list are avoided:
  1. A binary search of the auxiliary array finds the smallest time greater than the event time sought.
  2. Beginning with the event pointed to by the selected auxiliary array element, a linear search of the event list in descending order finds the desired event time.
- "Pull" technique reduces the length of the linear search.

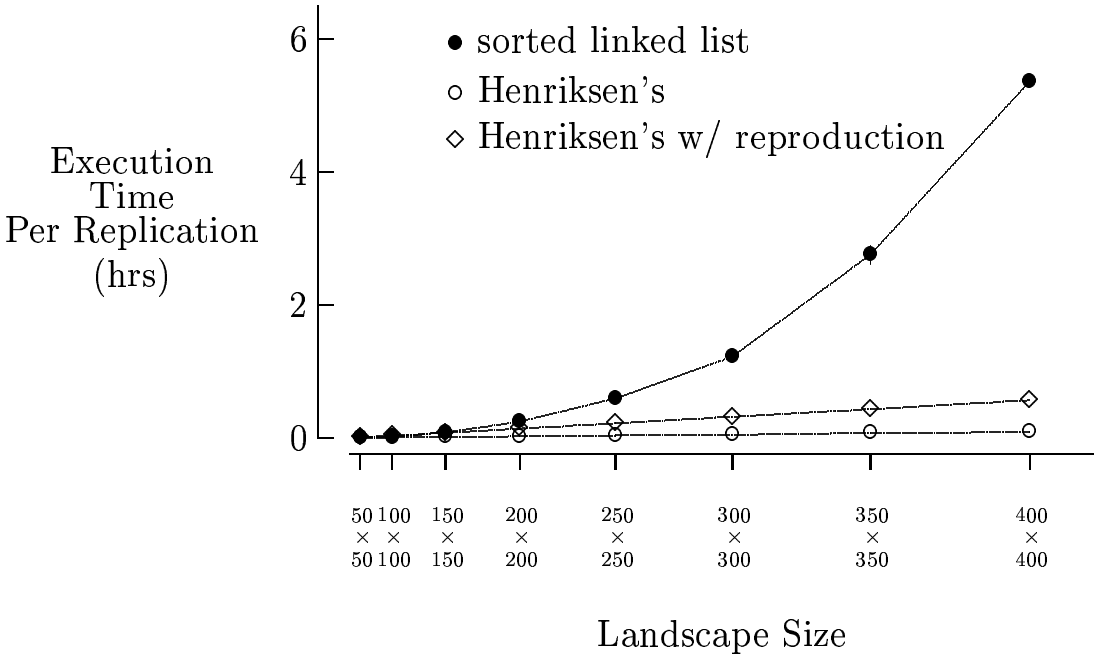


Improved Event List Performance

Event list operations:

- `GetNextEvent()` returns the second item in the event list.
- Event insertion and cancellation require a binary search of the auxiliary array, followed by a partial linear search of the event list.

With $T = 500$ and 0.16 initial population:



"Fat-dot accuracy" — 95% confidence intervals

Other approaches: heap, splay tree, calendar queue, ...

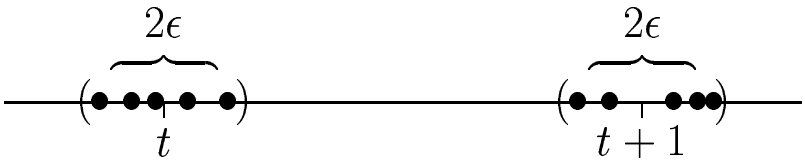
Synchronous Time Via Next-Event Simulation

How do we compare synchronous and asynchronous results?

The simulation clock, system state, event types and set of algorithms described for asynchronous time evolution remain the same.

Different event types no longer run as Poisson processes with their own characteristic rates.

Force agent events to occur within a small $\pm\epsilon$ window at times $t, t+1, \ldots$ as shown.

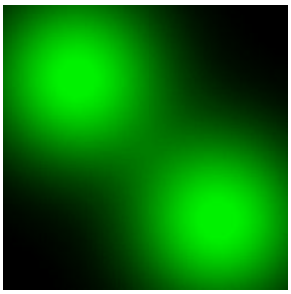


Movement and mating events for an agent occur simultaneously.

Birth and/or death events, if any, also take place at this time.

Parameters for Async vs. Sync Results

Initial values for the following results:



- $\gamma(x, y)$ is the two-peak Gaussian
- $\rho(x, y) = 1$ for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$
- 0.16 initial proportion of agents

with agent attributes initialized as follows:

ϕ : *Equilikely*(1, 6)

μ : *Uniform*(1.0, 4.0)

ϵ : 10.0

λ : *Uniform*(60.0, 100.0)

α : *Uniform*(12.0, 15.0)

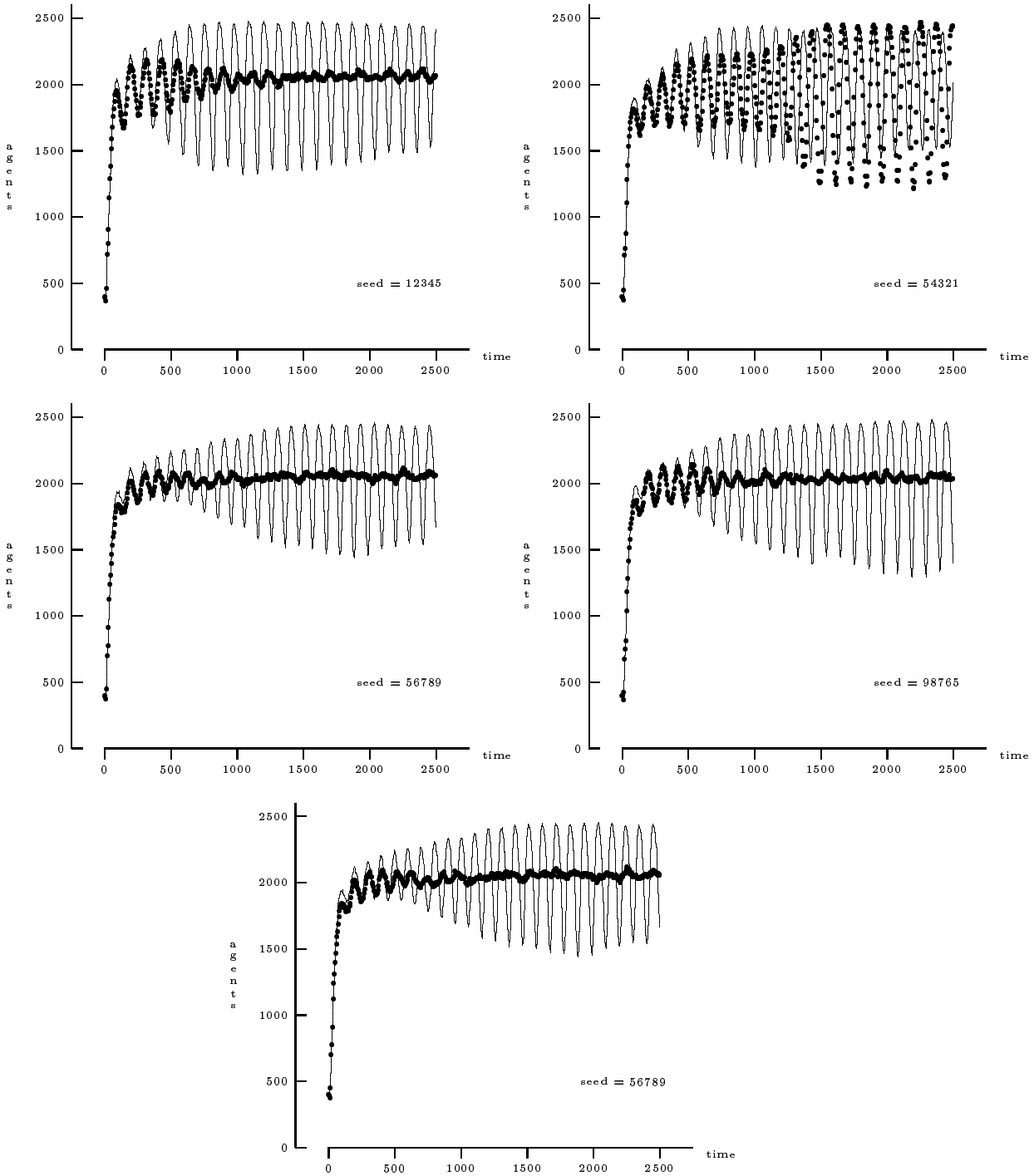
ω : *Uniform*(40.0, 50.0) for females

Uniform(50.0, 60.0) for males

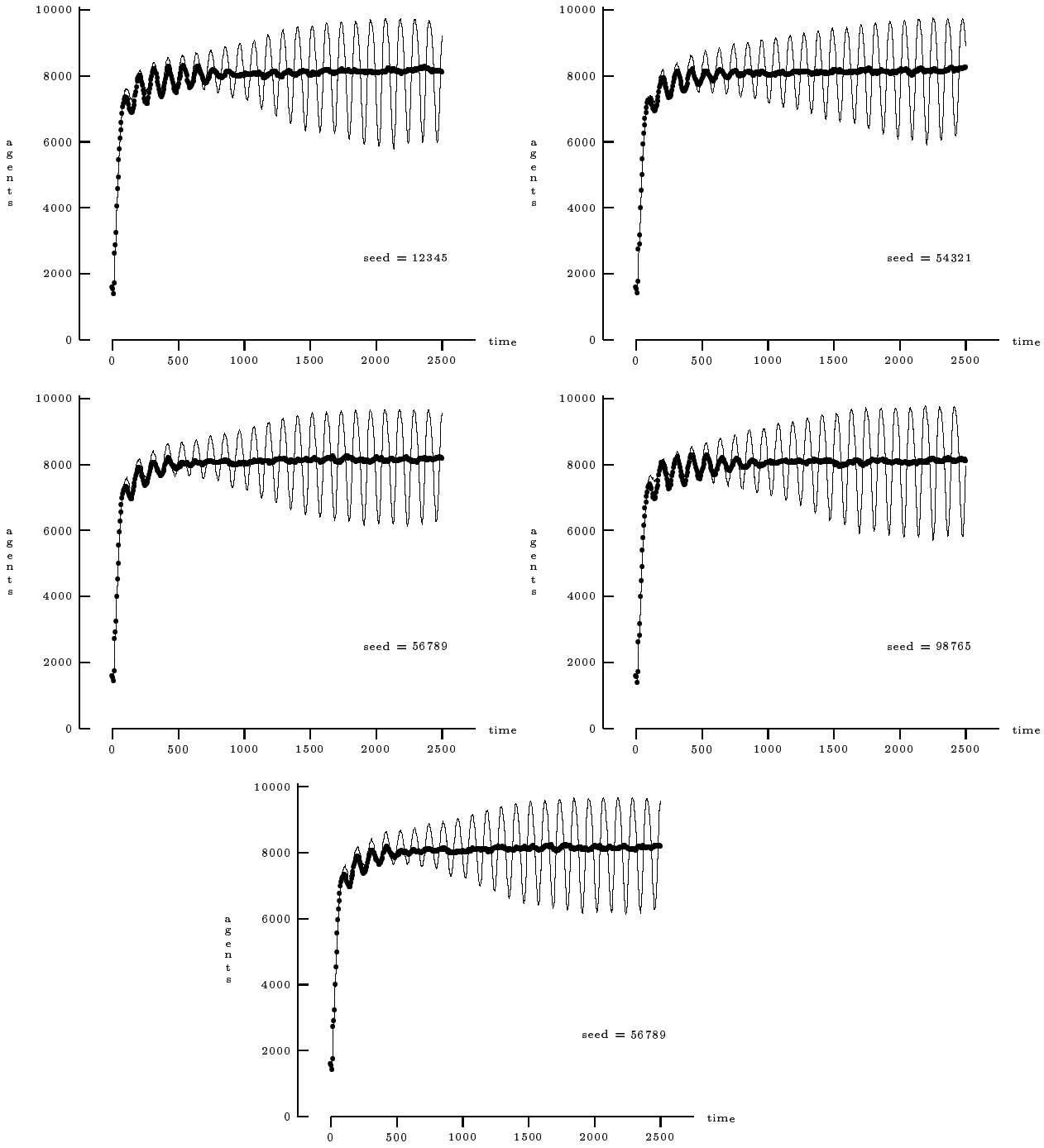
Agents execute the movement and reproduction rules ($\eta = 0$).

Reproduction and movement are coupled.

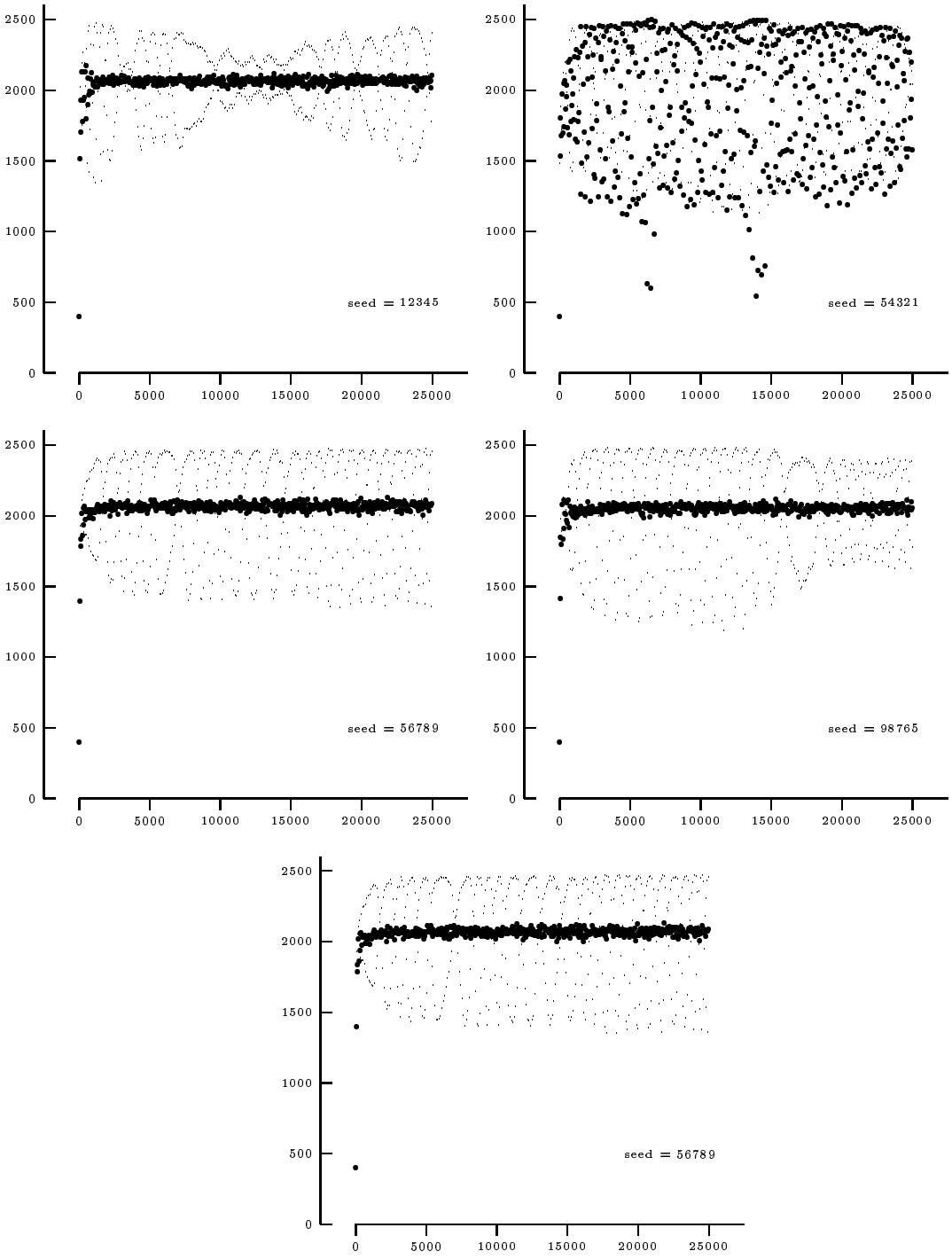
50×50 **landscape,** $T = 2500$



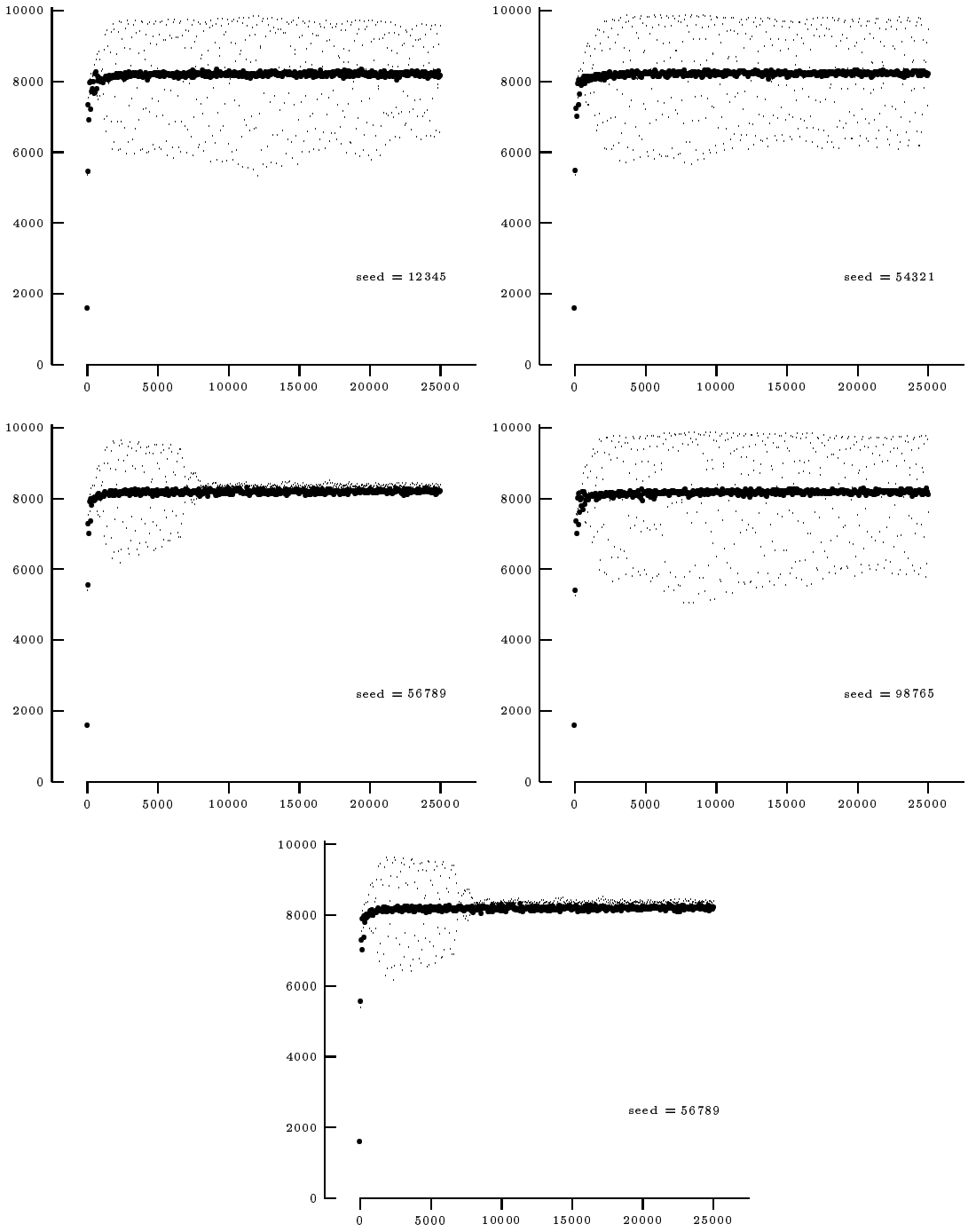
100×100 **landscape,** $T = 2500$



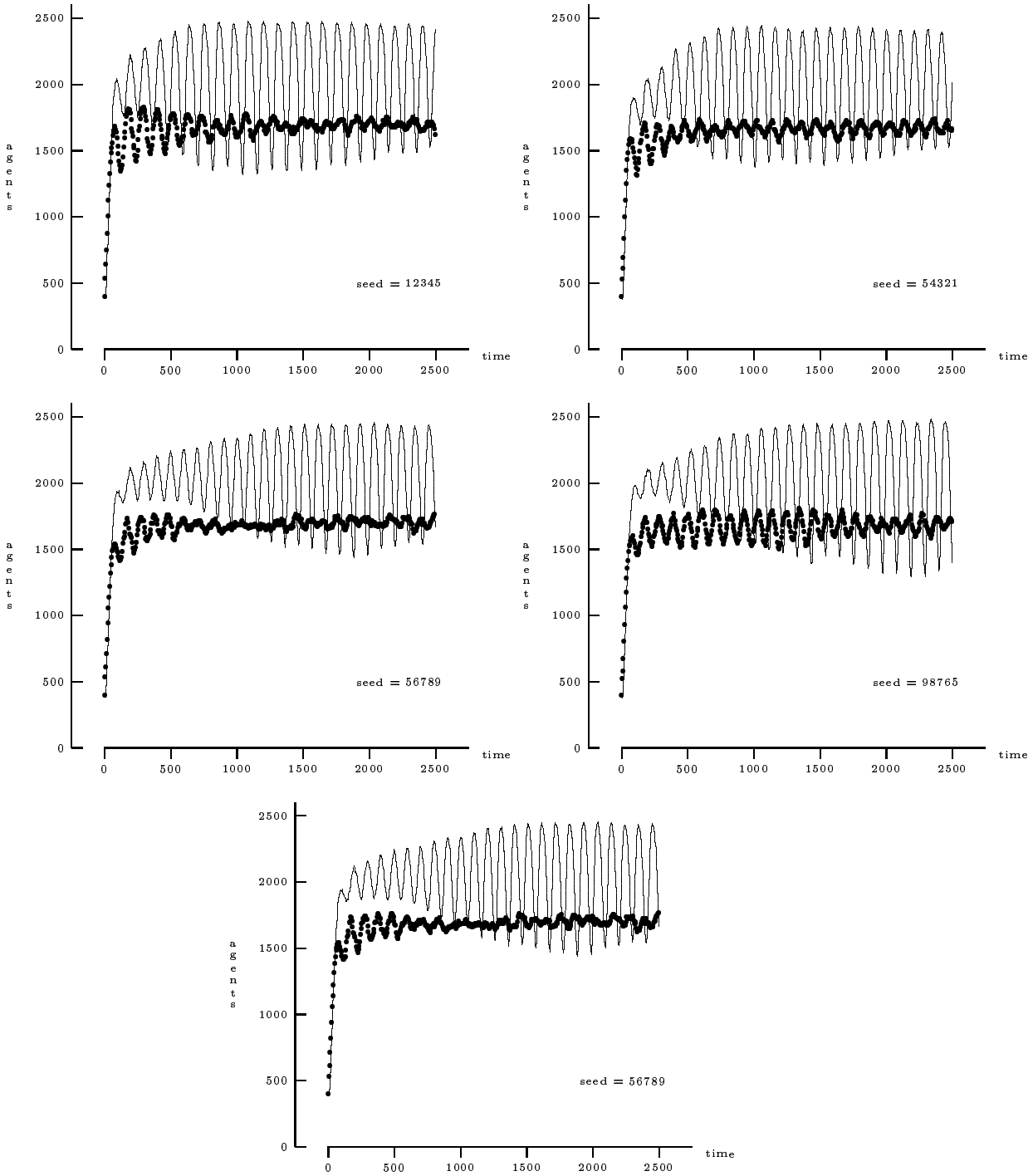
50×50 **landscape,** $T = 25000$



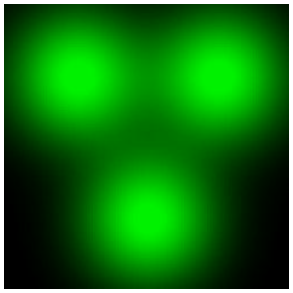
100×100 **landscape,** $T = 25000$

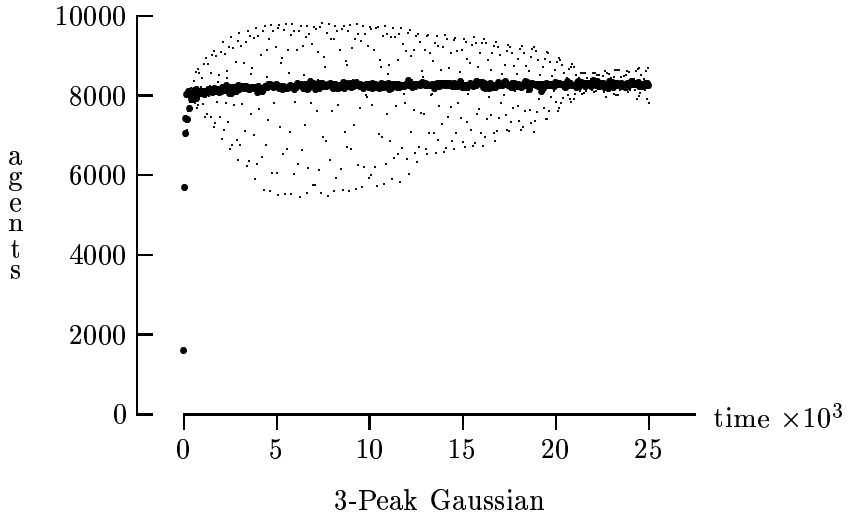


50×50, Mating Modeled Separately

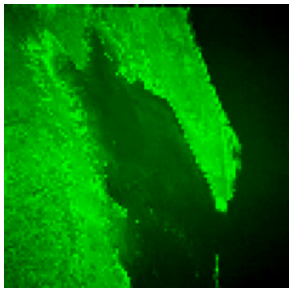


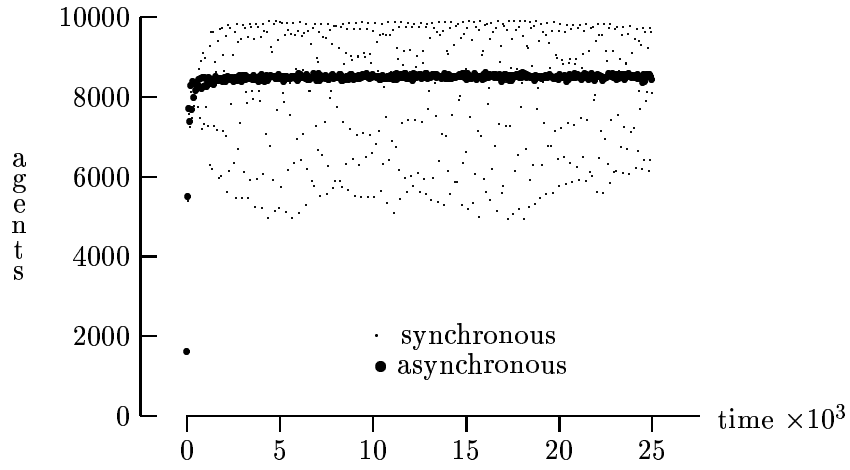
3-Peak and Digital Image Landscapes





3-Peak Gaussian





Australia Digital Image