## 10.4 Movement of Ants

## *R* Quick Review Questions

*Introduction to Computational Science:*

*Modeling and Simulation for the Sciences, 2nd Edition*

Angela B. Shiflet and George W. Shiflet

Wofford College

© 2014 by Princeton University Press

This file contains system-dependent Quick Review Questions and answers in *R* for Module 10.4 on "Movement of Ants." Complete all code development in *R*.

Grid Initializations

**Quick Review Question 1** This question refers to the initialization of the grid for ant movement. In an M-file, we define a function *gridInit* to return an initial grid, *grid*. The function begins as follows:

initAntGrid<- function(n, probAnt)

% INITANTGRID - initialization of n+2-by-n+2 array for ant simulation

% probAnt is the probability that a site has an ant.

global EMPTY NORTH EAST SOUTH WEST STAY BORDER

EMPTY = 0

NORTH = 1

EAST = 2

SOUTH = 3

WEST = 4

STAY = 5

BORDER = 6

**a.** Initialize *grid* to be an *n*+2-by-*n*+2 array of value *BORDER*.

**b.** Write a statement to return a random integer between 1 and 4 representing the four directions.

**c.** Complete the code to assign values to the interior of the two-dimensional array *grid*. With a probability of *probAnt*, a site contains an ant that faces in a random direction. Otherwise, the site does not contain an ant.

for (i in 2:n+1){

for (j in 2:n+1){

if( ){

grid[ ] = floor(runif(1,1,5))

else

grid[ ] = ;

}

}

}

**Quick Review Question 2**

**a.** Assign to *pherGrid* a *n*+2-by-*n*+2 array of value grid of zeros.

**b.** Complete the loop to generate a chemical trail with no ants on the middle row of the grid, *grid*. The maximum amount of chemical, *MAXPHER*, which is a global constant, occurs in column *n*, and for column *i*+1 the amount of chemical is a fraction, *i / n*, expressed as an integer, of the maximum, *MAXPHER*. For example, if *MAXPHER* is 50, *i* is 10, and *n* is 17, then the amount of chemical in column 10 is the integer 29 because (50)(10)/17 = 29.41.

i = round(n/2);

for i = 1:n

grid[mid, i+1] = ;

end

Applying Diffusion

**Quick Review Question 3** Complete the code for *applyDiffusionExtended* to apply the *diffusion* function to each internal cell and returns an (*n* + 2)-by-(*n* + 2) pheromone grid, keeping the border intact.

applyDiffusionExtended = function(matExt, diffusionRate)

n = ncol(matExt) - 2

pherGrid = matExt

for(i in : ){

for (j in : ){

site = matExt[i, j];

N = matExt[i-1, j];

NE = matExt[i-1, j+1];

E = matExt[i, j+1];

SE = matExt[i+1, j+1];

S = matExt[i+1, j];

SW = matExt[i+1, j-1];

W = matExt[i, j-1];

NW = matExt[i-1, j-1];

pherGrid[i, j] = diffusionPher(diffusionRate,

site, N, NE, E, SE, S, SW, W, NW);

}

return (phergrid)

}

Sensing

**Quick Review Question 4**

**a.** Complete the first line of the function M-file *sense.m*.

<- sense(site, na, ea, sa, wa, np, ep, sp, wp)

**b.** Complete the command in the function definition to indicate that *STAY* and *EMPTY* are not local.

utils::globalVariables( )

**c.** Complete the command to indicate that an empty site does not sense anything. Thus, its value remains the same.

if (site == )

direction =

}

**d.** Complete the assignment to *lst* of a list of the amounts of pheromone, *np*, *ep*, *sp*, *wp*, in neighboring cells.

lst =

**e.** This part refers to the second *sense* rule that an ant does not turn to a cell from which the creature just came. Suppose *site* is the direction from which the ant came. Write the statement to do the following: If *site* is less than or equal to 4 (i.e., not *STAY*), make the corresponding value of *lst* be a small number, -2.

**f.** This question refers to the fourth *sense* rule: An ant does not turn to a location that currently contains an ant. Suppose *na* is the ant grid value for the neighbor to the north. Assuming *neighbors* is defined as [*na*, *ea*, *sa*, *wa*], write the loop to do the following: If the neighbor in a direction contains an ant, make the corresponding element of *lst* be -2.

**e.** Complete the statement to assign the maximum level from *lst* to *mx*.

mx =

**f.** According to rule 6, if no neighboring cell is available, the ant will not plan to move. Start the *if* statement to test that the ant cannot (i.e., *mx* is negative), and if so the returned value is *STAY*

( ){

direction =

}

**g.** Parts g-j refer to Rule 5, which says that an ant turns in the direction of the neighboring available (not the previous, an occupied, or a border cell) with the greatest amount of chemical. Complete the statement to assign to *posList* a list of indices in *lst* where the maximum level, *mx*, occurs.

posList = ( )

**h.** Complete the statement to assign to *lng* the number of elements in *posList*.

lng = (posList)

**i.** Complete the statement to assign to *rndPos* a random integer between 1 and *lng*, inclusively, that is a possible index of *posList*.

rndPos =

**j.** Give the code to return the *posList* element with index *rndPos*.

**k.** Give this entire *sense* rule.

**Quick Review Question 5** *applySenseExtended* is similar to *applyPherExtended*, except we change the value of the *antGrid* cell only if the cell contains an ant (i.e. is not *EMPTY*). Start the *if* statement to test that the *ij* element of *antGrid* is not *EMPTY*.

Walking

**Quick Review Question 6**

**a.** Complete the assignment so that the new amount in *newPherGrid*[*i*, *j*] is the maximum of 0 and the current amount minus *EVAPORATE* (rule 7).

newPherGrid[i, j] =

**b.** Give the ant-grid element to the north of *antGrid*[*i*, *j*].

Simulation

**Quick Review Question 7**

**a.** Write a statement to assign to *antGrids* a page containing *antGrid*.

**b.** Write a statement to append *antGrid* to *antGrids* as the *i* + 1 element.

Visualizing the Simulation

**Quick Review Question 8** Suppose *antGrids* is a list of ant grids, *pherGrids* is a list of pheromone grids, and *maxp* is the maximum amount of chemical in any cell of any grid in *pherGrids*.

**a.** Write a statement to assign to *m* the number of grids in *antGrids*.

**b.** Write a statement to assign to *n* the number of rows (or columns) on the interior of an individual grid.

**c.** Write a statement to initialize *gr* to be an *n*× *n* matrix of zeros.

**d.** Assign to *a* the *k*th element of *antGrids*.

**e.** Assign to *map* the appending of the *rgb* value for red and the gray scale sequence from 0 to 1 with *maxp* number of values.

**f.** Suppose *p* is the *k*th element of *pherGrids*. Complete the nested loops to assign to *gr*[*i* - 1, *j* - 1] an appropriately scaled pheromone value from *p*[*i*, *j*] if *a*[*i*, *j*] is *EMPTY* and 0 otherwise, when an ant is present. We subtract one from the indices of *gr* because we are not including the border. To scale, we first add 1 to *p*[*i*, *j*] to eliminate the possibility of zero, and then we divide by (*maxp* + 1) to obtain a value greater than zero but less than or equal to one. Finally, to have the larger amounts of chemical be darker, we subtract the result from one.

for (i in 2:(n+1)){

for (j in 2:(n+1)) {

if (a[i, j] == EMPTY) {

gr[i-1, j-1] = # most chem->black

}

else {

gr[i-1, j-1] = # make ant lowest value, -> red

}

}

}

**g.** Write statements to produce an image of *gr* using the color map *map*, no axes, and a box around the image.

Answers to Quick Review Questions

**1. a.** grid = BORDER\*matrix(rep(1,(n+2)\*(n+2)),ncol=n+2)

**b.** floor(runif(1,1,5))

**c.** for (i in 2:n+1){

for( j in 2:n+1){

if (runif(1) < probAnt){

grid[i, j] = floor(runif(1,1,5))

else

grid[i, j] = EMPTY

}

}

}

**2. a.** grid = matrix(rep(0,(n+2)\*(n+2)),ncol=n+2)

**b.** i = floor(n/2) + 1

for (j in 1:n){

grid[mid, i+1] = i/n\*MAXPHER;

}

**3.**

applyDiffusionExtended<-function (matExt, diffusionRate){

n = ncol(matExt) - 2

pherGrid = matExt

for (i in 1:n+1){

for (j in 2:n+1){

site = matExt[i, j]

N = matExt[i-1,j]

NE = matExt[i-1, j+1]

E = matExt[i, j+1]

SE = matExt[i+1, j+1]

S = matExt[i+1, j]

SW = matExt[i+1, j-1]

W = matExt[i, j-1]

NW = matExt[i-1, j-1]

pherGrid[i, j] = diffusionPher(diffusionRate,

site, N, NE, E, SE, S, SW, W, NW)

}

}

return(pherGrid)

}

**2. a.** sense <- function(site, na, ea, sa, wa, np, ep, sp, wp)

**b.** utils::globalVariables(c("STAY", "EMPTY"))

**c.** if (site == EMPTY){

direction = EMPTY

}

**d.** lst = c(np, ep, sp, wp)

**e.** if (site < STAY){

lst(site) = -2

}

**f.** for (i in 1:4){

if (neighbors[i] > 0){

lst[i] = -2

}

}

**e.** mx = max(lst)

**f.** if (mx < 0){

direction = STAY

}

**g.** posList = which(lst == mx)

**h.** lng = length(posList)

**i.** rndPos = ceiling (runif(1,0,lng))

**j.** posList[rndPos]

**k.**

sense = function(site, na, ea, sa, wa, np, ep, sp, wp)

global STAY EMPTY

if (site == EMPTY){

direction = EMPTY

return

}

lst = c(np, ep, sp, wp)

# don't allow ant to turn to previous cell, so make value artificially

# small

if (site < STAY){

lst(site) = -2

}

# don't allow ant to turn to cell with another ant, so make value

# artificially small

neighbors = c(na, ea, sa, wa)

for (i in 1:4){

if (neighbors[i] > 0){

lst(i) = -2

}

}

mx = max(lst)

if (mx < 0){

direction = STAY

}else{

posList = which(lst == mx)

lng = length(posList)

rndPos = ceiling(runif(1,0,lng))

direction = posList[rndPos]

}

**5.** if (antGrid[i, j] != EMPTY)…

**6. a.** newPherGrid[i, j] = max((newPherGrid[i, j] - EVAPORATE, 0));

**b.** newPherGrid[i - 1, j]

**7. a.** antGrids[, , 1] = antGrid

**b.** antGrids[, , i+1] = antGrid

**8. a.** m = length(antGrids[1,1,])

**b.** n = length(antGrids[,1,1]) - 2

**c.** gr = matrix(rep(0,n\*n), nrow=n)

**d.** a = antGrids[, , k]

**e.** map = append(rgb(1,0,0),gray(seq(0,1,length=maxp)))

**f.**

for (i in 2:(n+1)){

for (j in 2:(n+1)) {

if (a[i, j] == EMPTY) {

gr[i-1, j-1] = 1 - (p[i, j]+1)/(maxp+1) # most chem->black

}

else {

gr[i-1, j-1] = 0 # make ant lowest value, -> red

}

}

}

**g.**

image(gr, col=map, axes=FALSE)

box()