Examining the Scaling Relationship Between City Size and Import Energy Cost

New Mexico
Supercomputing Challenge
Final Report
April 3rd, 2013

Team Number 3
AIMS@UNM

Team Members:

Walid Hasan
Israel Montoya
Nico Ponder
Ronald Rosa
Roderick Van Why

Teachers:

Michael Harris
Andrew Hostetler

Table of Contents

Executive Summary

A pair of ecological studies, published by a Los Alamos ecologist, recently showed that the scaling relationship between a city's size and its innovation and wealth was superlinear, meaning that as cities grew in size, they created or attracted disproportionately more innovation and wealth. However, these papers did not examine the scaling relationship between a city's size and the energy cost of importing food into the city. This simulation was designed to make clear the relationship between a city's size and the impact on its food ecology. By examining how the city expanded its food collection efforts, our model examined the relationship between food ecology and city size.

A team of ecologists and computer scientists at the University of New Mexico assisted in the development of the program, which was written in NetLogo. Using their previous ecological models as guidelines and reference points, we created agents that followed an optimized search algorithm before returning a model city. By logging and charting the energy usage of the agents as they gathered and imported food into the city, the model was able to examine the scaling relationship between city size and the energy cost of food importation.

Problem Statement

In two analyses of human ecology in cities, titled “Invention in the city: Increasing returns to patenting as a scaling function on metropolitan size” (2006) and “Growth, innovation, scaling, and the pace of life in cities,” (2007) the author, Luis M.A. Bettencourt, discusses the effect of a city’s size on its ecology. Bettencourt first found that larger cities are much more likely to house innovators and inventors, and showed a super-linear relationship between city size and the amount of inventors. Bettencourt then relates patterns of wealth, behavior, and infrastructure to city growth in the same superlinear manner, and postulates that this may promote “urbanism as a way of life.” In essence, Bettencourt shows that larger cities have disproportionately more invention and wealth.

In these papers, however, the scaling relationship of food needs in city ecology is not mentioned. Given that other population-related qualities of the city follow a relationship of increasing returns, our team was curious about whether the ecological energy cost of importing food into a city would be affected by the city’s size in the same manner. This is especially important when discussing food production, gathering, and import into a metropolitan area. The goal of this project was to create a model that can accurately describe the scaling relationship between the two.

Methods

We examined the relationship between a city's size and its food ecology using an agent-based model programmed in NetLogo. The code accomplished the goal by going through several steps. The model generated three breeds of agents at a central city that was surrounded by food. The agents represented humans with wheelbarrows, horses with carts, and semi-trailer trucks. When the model was run, the agents spread out from the city to search for food. As the agents encountered a food source they 'harvested' it by changing the patch that held the food to the background color of the world. After the agents acquired the food, they switched behaviors and returned to the city while laying a trail for other agents to follow. As the agents returned to the city, they deposited their food. The agents then searched for food again, prioritizing trails left by themselves and other agents, and repeated this until there was no more food to find or until the model was stopped. The amount in Gigajoules that the agents exerted while searching for the food was logged for the duration of the model.

As these processes were running, the model generated three graphs in real time. The graphs showed the Gigajoules expended over time, the Gigajoules expended in relation to the size of the city's food network, and the Gigajoules expended per 1 kilometer of food in relation to the size of the food network. These graphs were used to compare the energy cost of importing food into the city in relation to the city's size.

Results and Conclusion

The model showed that the energy cost of importing additional food into the city increased at a much higher rate than the expansion of the city's food network. This can be seen in Figure 1, Figure 2, and Figure 3 below. These graphs show the relationship between the amount of Gigajoules that the agent breeds exerted while importing food and the size of the city's food network. The superlinear nature of the graphs demonstrated that as the city's food network expended, the Gigajoules required to import food into the city increased at an exponential rate.

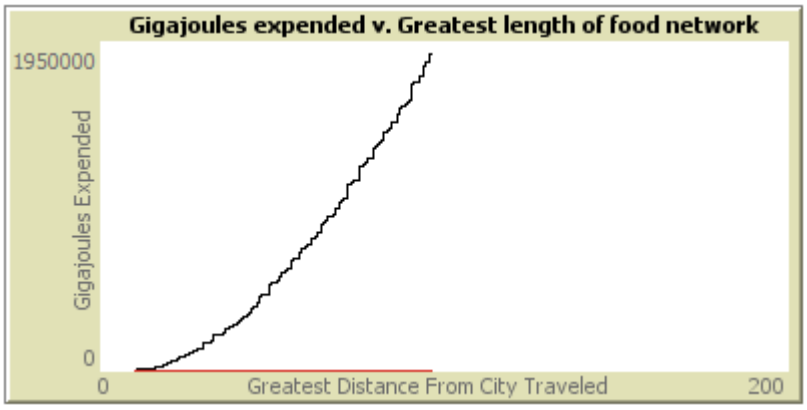


Fig. 1- The Gigajoules expended by the human agents in relation to the size of the food network

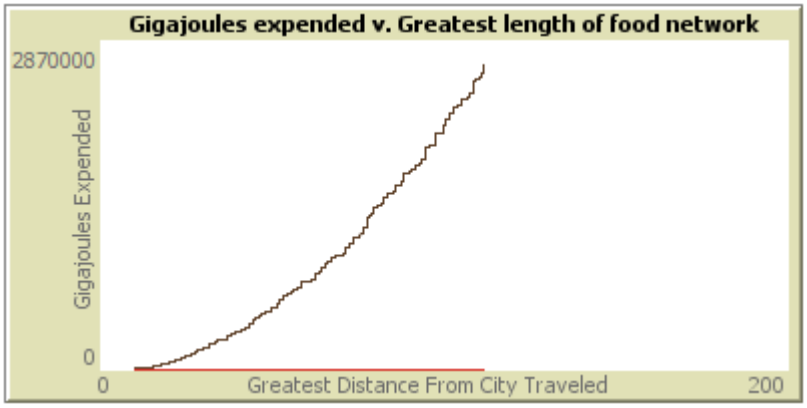
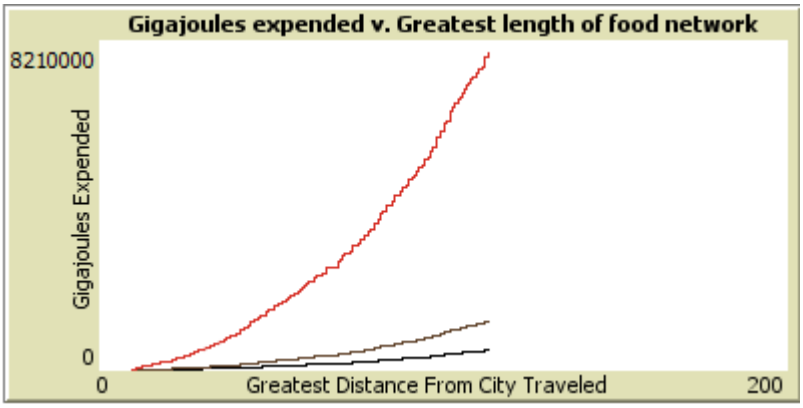


Fig. 2- The Gigajoules expended by the horse agents in relation to the size of the food



network

Fig. 3- The Gigajoules expended by the truck agents in relation to the size of the food network

The graph shown in Figure 4 supports this conclusion. The graph shows the amount of Gigajoules required to harvest one square kilometer increasing as the size of the food importation network increases, rather than remaining at a constant rate. This suggests that the increasing distance that the agents have to travel to import food into the city is the underlying cause in the superlinear increase seen in the previous graphs.

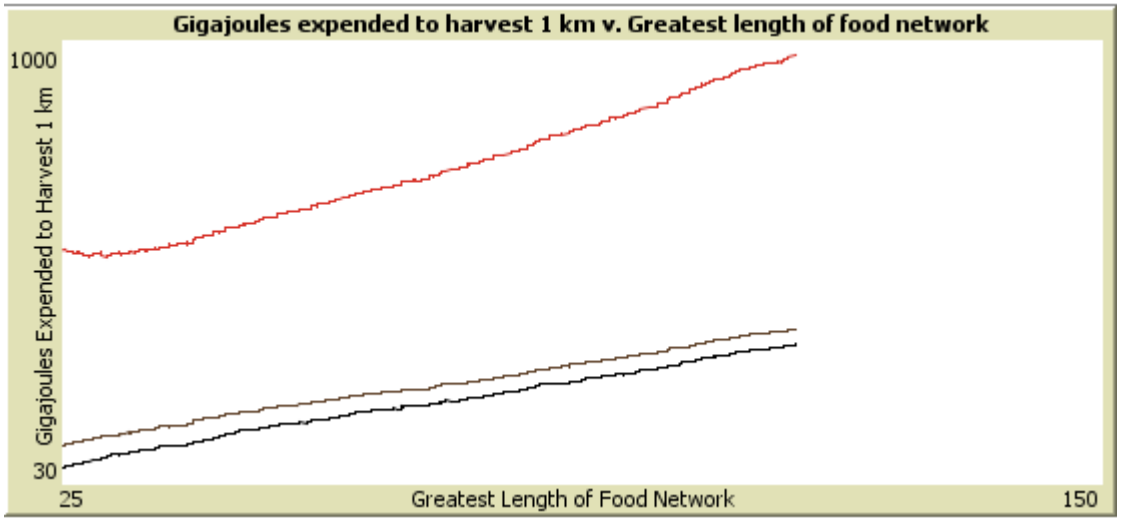


Fig. 4- The amount of Gigajoules expended to harvest one square kilometer of food in relation to the size of the city's food network

Our model successfully showed that the relationship between a city's size and the energy cost of importing food into a city is superlinear in nature. We consider this highly significant, given that the large cities that are described in the Bettencourt studies as potentially creating increasingly large amounts of innovation and wealth may also create an increasingly large energy cost when importing food into the city.

Credits and References

The original model was created by Daniel Washington and Dr. Kenneth Letendre in the lab of Dr. Melanie Moses, Departments of Computer Science and Biology, at the University of New Mexico. Funding was from the National Science Foundation's program in Advancing Theory in Biology (Grant #EF 1038682).

More information about the model and the ants upon which the model is based can be found in:

T.P. Flanagan, K. Letendre, W.R. Burnside, G.M. Fricke and M.E. Moses. (2011). "How Ants Turn Information into Food." Proceedings of the 2011 IEEE Conference on Artificial Life:178-185.

T.P. Flanagan, K. Letendre, W.R. Burnside, G.M. Fricke and M.E. Moses. (2012). "The Effect of Colony Size and Food Distribution on Harvester Ant Foraging." PLoS ONE (in press).

Original simulation URL: https://sites.google.com/site/unmantbot/?pli=1

Acknowledgements

UNM Team

Robbie Burger

Sean T. Hammond

Tatiana P. Flanagan

Mike Chang

Trevor Fristoe

Neal Holtschulte

Jeff Nekola

Melanie Moss

AIMS@UNM

Andrew Hostetler

Michael Harris

Frederick Bobberts

Appendix A: The Model’s Code

```
globals [
  road_ahead new_distance curr_distance dist pile_radius GAtrail GAsite GAexpand
lfood_counter1
  food_total cal_per_min_kg kg_per_individual kilojoule_conversion sec_per_tick
movement_total
  gigajoules_total time_ticks seconds_per_hour max_food_in_a_patch
ticks_to_wait_after_harvesting
  movement_humans movement_horses movement_trucks gigajoules_humans
gigajoules_horses gigajoules_trucks
  cal_per_hour_horses cal_per_gallon kilometers_per_gallon size_patch
carry_volume_humans carry_volume_horses
  carry_volume_trucks squad_size_humans squad_size_horses squad_size_trucks
greatest_distance
  food_humans food_horses food_trucks food_joule_ratio_humans
food_joule_ratio_horses food_joule_ratio_trucks
  ]
;first line are vars kept from original code
;all other lines are vars we added
breed [humans human]
breed [horses horse]
breed [trucks truck]
humans-own [trail_ahead behavior cityx boundary? has_food? speed_humans
dist_to_move_humans patches_to_move_humans foodx foody fidelity recruit]
horses-own [trail_ahead behavior cityx boundary? has_food? speed_horses
dist_to_move_horses patches_to_move_horses foodx foody fidelity recruit]
trucks-own [trail_ahead behavior cityx boundary? has_food? speed_trucks
dist_to_move_trucks patches_to_move_trucks foodx foody fidelity recruit]
patches-own [lfood? trail? ticks_since_harvesting]
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;main setup function;;;;;;;;;main setup function;;;;;;;;;main setup function;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to setup_world
__clear-all-and-reset-ticks
 setup_breeds
 setup_vars
 ask patches [
 set pcolor 62
 ]
 go_density_recruit
;  do-plotting-left
 if ticks > 100 [
    ask humans [
```

```
    set behavior 3
            ]
 ]
end
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;creates locations for food piles;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to setup_breeds
 set-default-shape humans "person" ;human shape
 ask patch 0 0 [
   sprout-humans pop_humans
 ]
 ask humans [
   set color black ;human color
   set ycor (-7 + random 14) ;human location
   set xcor ((xcor - 7) + random 14)
   set heading random 360 ;human heading
   set behavior 0 ;sets the humans to their initial behavior condition
   set size 1
 ]
 set-default-shape horses "cow" ;horse shape
 ask patch 0 0 [
   sprout-horses pop_horses
 ]
 ask horses [
   set color brown ;horse color
   set ycor (-7 + random 14) ;horse location
   set xcor ((xcor - 7) + random 14)
   set heading random 360 ;horse heading
   set behavior 0 ;sets the horses to their initial behavior condition
   set size 1
 ]
  set-default-shape trucks "car" ;truck shape
 ask patch 0 0 [
  sprout-trucks pop_trucks
 ]
 ask trucks [
   set color red ;truck color
   set ycor (-7 + random 14) ;truck location
   set xcor ((xcor - 7) + random 14)
   set heading random 360 ;truck heading
   set behavior 0 ;sets the trucks to their initial behavior condition
   set size 1
 ]
end
to setup_vars ;;executed by setup
```

```
 ;;used for general calculation
 set size_patch 1000; in meters per side.
 set max_food_in_a_patch 76602;expected yield of 1 sq km of land
 set kilojoule_conversion 4.184  ;4.184 kilojoules per Calorie.
 set sec_per_tick 60; based on program measurements
 set seconds_per_hour 3600
 set greatest_distance 0
 set food_humans 1
 set food_horses 1
 set food_trucks 1
 ;;for calculating kjoules for human movement
 set cal_per_min_kg 0.06 ;in units of Cal * 1/(sec*kg)
 set kg_per_individual 62 ;average human body mass
 set carry_volume_humans 2.2 ;in cubic m
 set squad_size_humans (max_food_in_a_patch / carry_volume_humans)
 ;;for calculating kjoules for horse movement
 set cal_per_hour_horses 2500 ;From National Academy's NRH's 2007 study
 set carry_volume_horses 10 ;in cubic m
 set squad_size_horses (ceiling(max_food_in_a_patch / carry_volume_horses))
 ;;for calculating kjoules for truck movement
 set cal_per_gallon 35000 ;kcal in a gallon of diesel
 set kilometers_per_gallon 6; avg fuel efficiency of a highly effecient loaded semi
 set carry_volume_trucks 114.8 ;in cubic m
 set squad_size_trucks (ceiling(max_food_in_a_patch / carry_volume_trucks))
 ask trucks [
    set speed_trucks 64.37; in km/h, originally 40 mph
    set dist_to_move_trucks  (speed_trucks * (sec_per_tick / seconds_per_hour)) ; km/h
* (1h/3600sec * 60 sec/tick)
    set patches_to_move_trucks (dist_to_move_trucks / (size_patch / 1000))
 ]
 ask horses [
    set speed_horses 13; in km/h, from Sean's data for a trot speed
    ;lowered this speed b/c data was unavailale for calories burned by a canter, but is
available for a trot
    set dist_to_move_horses (speed_horses * (sec_per_tick / seconds_per_hour))
    set patches_to_move_horses (dist_to_move_horses / (size_patch / 1000))
 ]
 ask humans [
    set speed_humans 4;
    set dist_to_move_humans (speed_humans * (sec_per_tick / seconds_per_hour))
    set patches_to_move_humans (dist_to_move_humans / (size_patch / 1000));
 ]
 ;; NOTE:
 ;; dist to move and patches to move are currently the same because the patch size is a
kilometer
 ;; this will change when/if the size is changed
```

```
end
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;main runtime function;;;;;;;;;;;;;;;;;;;;main runtime function;;;;;;;;;;;;;;main runtime
function;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to go_density_recruit ;;function executed by the "run" button
 set GAtrail .92328
 set GAsite 1.0
 set GAexpand 0.9836
 evaporate_trail   ;;;decrements all trails trail value (ctrl+F "evaporate trail" for details)
 ask humans [ ;splits the humans behavior into halves based on location.
    ;humans on the left will bring food to the left city and humans on the right will bring
food to the right city.
     if  cityx != 0
[
    set cityx 0
]




    if not can-move? 1 or pcolor = black[ ;if humans are near the world boundary, will turn
180 degrees and move away 1 unit
     rt 180
    fd patches_to_move_humans
    set movement_humans (movement_humans + 1)
  ]

    ;;At each tick, humans decide on an individual basis to execute one of six behviors.
    ;;Different stimuli such as the presence of trails or food will cause humans to change
their behaviors
    ;;on an individual bases.


    stop_search? ;function determining random chance of giving up (ctrl+F
"stop_search?" for details)


    if behavior = 0 [ ;function for initial expansion of the humans (ctrl+F "move_away" for
details)
     move_away
    ]

    if behavior = 1 [  ;function for random walking and food searching behavior (crtl+F
"random_walk" or "check_food" for details)
```

```
     random_walk
     check_food
    ]

    if behavior = 2 [  ;function for trail following behavior (ctrl+F "scan_trail" for details)
     scan_trail
    ]


   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   ;;;;;;;;;returning to the city;;;;;;;;;;
   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    if behavior = 3 [  ;function for when the humans choose to return home without
marking a trail (ctrl+F "return_home" for details)
     return_home
    ]

    if behavior = 4 [ ;function when humans choose to return home while marking a trail
or incrementing existing trail (ctrl+F "color_trail" for details)
     color_trail
     return_home

    ]
   if behavior = 6 [ ;function where humans use site fidelity to return to the last known
food location (ctrl+F "find_food" for details)
     find_food
    ]
 ]
    ask horses [ ;splits the humans behavior into halves based on location.
    ;humans on the left will bring food to the left city and humans on the right will bring
food to the right city.
     if  cityx != 0
[
    set cityx 0
]



    if not can-move? 1 or pcolor = black[ ;if humans are near the world boundary, will turn
180 degrees and move away 1 unit
     rt 180
    fd patches_to_move_horses
    set movement_horses (movement_horses + 1)
  ]
```

```
    ;;At each tick, humans decide on an individual basis to execute one of six behviors.
    ;;Different stimuli such as the presence of trails or food will cause humans to change
their behaviors
    ;;on an individual bases.


    stop_search? ;function determining random chance of giving up (ctrl+F
"stop_search?" for details)


    if behavior = 0 [ ;function for initial expansion of the humans (ctrl+F "move_away" for
details)
      move_away
    ]

    if behavior = 1 [  ;function for random walking and food searching behavior (crtl+F
"random_walk" or "check_food" for details)
      random_walk_horses
      check_food
    ]

    if behavior = 2 [  ;function for trail following behavior (ctrl+F "scan_trail" for details)
      scan_trail_horses
    ]


   ..........................................
   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   ;;;;;;;;;returning to the city;;;;;;;;;;;
   ..........................................
   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    if behavior = 3 [  ;function for when the humans choose to return home without
marking a trail (ctrl+F "return_home" for details)
      return_home_horses
    ]

    if behavior = 4 [ ;function when humans choose to return home while marking a trail
or incrementing existing trail (ctrl+F "color_trail" for details)
      color_trail
      return_home_horses

    ]
    if behavior = 6 [ ;function where humans use site fidelity to return to the last known
food location (ctrl+F "find_food" for details)
      find_food_horses
    ]
```

```
 ]

     ask trucks [ ;splits the humans behavior into halves based on location.
    ;humans on the left will bring food to the left city and humans on the right will bring
food to the right city.
     if  cityx != 0
[
    set cityx 0
]




    if not can-move? 1 or pcolor = black[ ;if humans are near the world boundary, will turn
180 degrees and move away 1 unit
     rt 180
    fd patches_to_move_trucks
    set movement_trucks (movement_trucks + 1)
  ]

    ;;At each tick, humans decide on an individual basis to execute one of six behviors.
    ;;Different stimuli such as the presence of trails or food will cause humans to change
their behaviors
    ;;on an individual bases.


    stop_search? ;function determining random chance of giving up (ctrl+F
"stop_search?" for details)


    if behavior = 0 [ ;function for initial expansion of the humans (ctrl+F "move_away" for
details)
     move_away
    ]

    if behavior = 1 [  ;function for random walking and food searching behavior (crtl+F
"random_walk" or "check_food" for details)
     random_walk_trucks
     check_food
    ]

    if behavior = 2 [  ;function for trail following behavior (ctrl+F "scan_trail" for details)
     scan_trail_trucks
    ]
```

```
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ;;;;;;;;;returning to the city;;;;;;;;;;;
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    if behavior = 3 [  ;function for when the humans choose to return home without
marking a trail (ctrl+F "return_home" for details)
     return_home_trucks
    ]

    if behavior = 4 [ ;function when humans choose to return home while marking a trail
or incrementing existing trail (ctrl+F "color_trail" for details)
     color_trail
     return_home_trucks

    ]
    if behavior = 6 [ ;function where humans use site fidelity to return to the last known
food location (ctrl+F "find_food" for details)
     find_food_trucks
    ]
 ]
 ;regrow_patches
 set time_ticks (time_ticks + 1)
 set gigajoules_humans (cal_per_min_kg * kg_per_individual * kilojoule_conversion / 60
* sec_per_tick * pop_humans * squad_size_humans * time_ticks / 1000 / 1000)
 set gigajoules_horses (cal_per_hour_horses / 60 * kilojoule_conversion * pop_horses *
squad_size_horses * time_ticks / 1000 / 1000)
 set gigajoules_trucks (cal_per_gallon / kilometers_per_gallon * kilojoule_conversion *
squad_size_trucks * movement_trucks / 1000 / 1000)
 set food_joule_ratio_humans (gigajoules_humans / food_humans)
 set food_joule_ratio_horses ( gigajoules_horses / food_horses)
 set food_joule_ratio_trucks (gigajoules_trucks / food_trucks)
 set gigajoules_total (gigajoules_humans + gigajoules_horses + gigajoules_trucks)
 set food_total (food_humans + food_horses + food_trucks)
 set movement_total (movement_humans + movement_horses + movement_trucks)
 tick ;next time step
end
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;human behavioral fuctions;;;;;;;;;;;;;;human behavioral fuctions;;;;;;;;;;;;;;human
behavioral fuctions;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to return_home  ;function for human behavior when returning to the city
 let max_road 0
 let x -1
 let y -1
 let trail_follow? 0
 let x1 cityx
```

```
 ifelse (distancexy cityx 0) > 0 [  ;if human is not at city, execute movement
    facexy cityx 0
    set curr_distance (distancexy cityx 0) ; registers current distance from city

    if curr_distance > greatest_distance [
    set greatest_distance curr_distance
    ]


    rt random-normal 0 20  ;random wiggle movement
    ask patch-ahead 1 [
     set new_distance (distancexy x1 0)  ;registers future position from city
    ]
    if new_distance < curr_distance[ ;only moves if future position from city is closer than
current position
     face patch-ahead 1
     forward distance patch-ahead .51 ;moves to the patch 1 unit ahead
     set movement_humans (movement_humans + 1)
     setxy  pxcor pycor ;centers on the patch after moving
    ]
 ]
 [ if has_food? = 1 [ ;increments the food collection counters on the respective side of
the simulation upon returing food to the city
     set food_humans (food_humans + 1)

     ]
 set has_food? 0
 while [x < 2] [
    set y -1
    while [y < 2] [ ;while loops used to ask all surrounding patches for trail
     if x != cityx or y != 0 [ ;does not look for trail on the city
       ask patch-at x y [
         if trail? > max_road [ ;sets maximum trail value to highest surrounding trail value
           set max_road trail?
         ]
       ]
     ]
     set y (y + 1)
    ]
    set x (x + 1)
 ]
 ifelse recruit = 1 [
    ifelse max_road > 0 [
     set xcor cityx
     set ycor 0
```

```
    while [xcor = cityx and ycor = 0 and behavior != 2] [ ;If human enters trail follow
behavior, executes loop until human moves away from the city
      set heading (random 360)
      scan_ahead   ;scans 1 patch at every random heading
      if road_ahead >= (random-float max_road) [  ;random chance based on the
maximum trail to follow current trail trail
        face patch-ahead 1
        fd distance patch-ahead .51 ;moves onto 1 patch ahead
        set movement_humans ((movement_humans + 1) )
        setxy pxcor pycor ;centers on current patch
        set behavior 2
      ]
    ]
   ]
   [ set heading (random 360) ;if no trail is present, human enters search mode
    set behavior 1
   ]
 ]
 [
   ifelse fidelity = 1 [
    facexy foodx foody
    set behavior 6
   ]
   [ set heading (random 360)
    set behavior 1
   ]
 ]
 ]
end
to return_home_horses  ;function for human behavior when returning to the city
 let max_road 0
 let x -1
 let y -1
 let trail_follow? 0
 let x1 cityx
 ifelse (distancexy cityx 0) > 0 [  ;if human is not at city, execute movement
   facexy cityx 0
   set curr_distance (distancexy cityx 0) ; registers current distance from city

   if curr_distance > greatest_distance [
    set greatest_distance curr_distance
   ]

   rt random-normal 0 20  ;random wiggle movement
   ask patch-ahead 1 [
    set new_distance (distancexy x1 0)  ;registers future position from city
```

```
    ]

    if new_distance < curr_distance[ ;only moves if future position from city is closer than
current position
      face patch-ahead 1
      forward distance patch-ahead .8 ;moves to the patch 1 unit ahead
      setxy  pxcor pycor ;centers on the patch after moving
    ]
 ]
 [ if has_food? = 1 [ ;increments the food collection counters on the respective side of
the simulation upon returing food to the city
     set food_horses (food_horses + 1)

     ]
 set has_food? 0
 while [x < 2] [
    set y -1
    while [y < 2] [ ;while loops used to ask all surrounding patches for trail
     if x != cityx or y != 0 [ ;does not look for trail on the city
       ask patch-at x y [
         if trail? > max_road [ ;sets maximum trail value to highest surrounding trail value
           set max_road trail?
         ]
       ]
     ]
     set y (y + 1)
    ]
    set x (x + 1)
 ]
 ifelse recruit = 1 [
    ifelse max_road > 0 [
     set xcor cityx
     set ycor 0
     while [xcor = cityx and ycor = 0 and behavior != 2] [ ;If human enters trail follow
behavior, executes loop until human moves away from the city
       set heading (random 360)
       scan_ahead_horses   ;scans 1 patch at every random heading
       if road_ahead >= (random-float max_road) [  ;random chance based on the
maximum trail to follow current trail trail
         face patch-ahead 1
         fd distance patch-ahead .8 ;moves onto 1 patch ahead
         set movement_horses (movement_horses + 1)
         setxy pxcor pycor ;centers on current patch
         set behavior 2
       ]
     ]
```

```
    ]
    [ set heading (random 360) ;if no trail is present, human enters search mode
     set behavior 1
    ]
 ]
 [
    ifelse fidelity = 1 [
     facexy foodx foody
     set behavior 6
    ]
    [ set heading (random 360)
     set behavior 1
    ]
 ]
 ]
end
to return_home_trucks  ;function for human behavior when returning to the city
 let max_road 0
 let x -1
 let y -1
 let trail_follow? 0
 let x1 cityx
 ifelse (distancexy cityx 0) > 0 [  ;if human is not at city, execute movement
    facexy cityx 0
    set curr_distance (distancexy cityx 0) ; registers current distance from city

    if curr_distance > greatest_distance [
     set greatest_distance curr_distance
    ]

    rt random-normal 0 20  ;random wiggle movement
    ask patch-ahead 1 [
     set new_distance (distancexy x1 0)  ;registers future position from city
    ]
    if new_distance < curr_distance[ ;only moves if future position from city is closer than
current position
     face patch-ahead 1
     forward distance patch-ahead patches_to_move_trucks ;moves to the patch 1 unit
ahead
     setxy  pxcor pycor ;centers on the patch after moving
    ]
 ]
 [ if has_food? = 1 [ ;increments the food collection counters on the respective side of
the simulation upon returing food to the city
     set food_trucks (food_trucks + 1)
```

```
    ]
 set has_food? 0
 while [x < 2] [
   set y -1
   while [y < 2] [ ;while loops used to ask all surrounding patches for trail
    if x != cityx or y != 0 [ ;does not look for trail on the city
      ask patch-at x y [
        if trail? > max_road [ ;sets maximum trail value to highest surrounding trail value
          set max_road trail?
        ]
      ]
    ]
    set y (y + 1)
   ]
   set x (x + 1)
 ]
 ifelse recruit = 1 [
   ifelse max_road > 0 [
    set xcor cityx
    set ycor 0
    while [xcor = cityx and ycor = 0 and behavior != 2] [ ;If human enters trail follow
behavior, executes loop until human moves away from the city
      set heading (random 360)
      scan_ahead_trucks   ;scans 1 patch at every random heading
      if road_ahead >= (random-float max_road) [  ;random chance based on the
maximum trail to follow current trail trail
        face patch-ahead 1
        fd distance patch-ahead patches_to_move_trucks ;moves onto 1 patch ahead
        set movement_trucks (movement_trucks + 1)
        setxy pxcor pycor ;centers on current patch
        set behavior 2
      ]
    ]
   ]
   [ set heading (random 360) ;if no trail is present, human enters search mode
    set behavior 1
   ]
 ]
 [
   ifelse fidelity = 1 [
    facexy foodx foody
    set behavior 6
   ]
   [ set heading (random 360)
    set behavior 1
   ]
```

```
 ]
 ]
end
to random_walk
 set color black
 let st_dev 0
 ;behavior executed during behavior 1
 if ticks mod 4 = 0 [
    rt random-normal 0 (st_dev * 180 / pi)
 ]
 fd .25   ;turns up to 30 degrees off of current heading and moves forward 1/4 of
maximum speed
 set movement_humans (movement_humans + .25)
end
to random_walk_horses
 set color brown
 let st_dev 0
 ;behavior executed during behavior 1
 if ticks mod 4 = 0 [
    rt random-normal 0 (st_dev * 180 / pi)
 ]
 fd .25   ;turns up to 30 degrees off of current heading and moves forward 1/4 of
maximum speed
 set movement_horses (movement_horses + 1)
end
to random_walk_trucks
 set color red
 let st_dev 0
 ;behavior executed during behavior 1
 if ticks mod 4 = 0 [
    rt random-normal 0 (st_dev * 180 / pi)
 ]
 fd .25   ;turns up to 30 degrees off of current heading and moves forward 1/4 of
maximum speed
 set movement_trucks (movement_trucks + 1)
end
to evaporate_trail
 ask patches with [trail? > 0] [
    let evapo 0

  set evapo Evaporation_rate

    set trail? (trail? * (1 - evapo))  ;trail evaporation function
    if trail? < .001 [set trail? 0]   ;if trail becomes almost undetectable, sets value to 0
    if pcolor != 62 [   ;trail is only visually represented on pixels without food
     if trail? >= 6 [set pcolor 99]
```

```
     if trail? >= 5 and trail? < 6 [set pcolor 98]   ;color gets darker as trail gets weaker
     if trail? >= 4 and trail? < 5 [set pcolor 97]
     if trail? >= 3 and trail? < 4 [set pcolor 96]
     if trail? >= 2 and trail? < 3 [set pcolor 95]
     if trail? >= 1 and trail? < 2 [set pcolor 94]
     if trail? >= .1  and trail? < 1 [set pcolor 93]
     if trail? >= .01 and trail? < .1 [set pcolor 92]
     if trail? = 0 and pcolor = 92 [set pcolor grey]
    ]
 ]
end
to find_food
 set color blue ;humans turn blue while executing site fidelity
 if (abs xcor) < ( abs foodx + 2) and (abs xcor) > ( abs foodx - 2) [ ;if the human is within
2 pixels of last known food location, exits site fidelity behavior
    if (abs ycor) < ( abs foody + 2) and (abs ycor) > ( abs foody - 2) [
     set behavior 1
    ]
 ]
 facexy foodx foody ;moves towards last known food location
 rt (-20 + random 40)
 fd patches_to_move_humans
 set movement_humans (movement_humans + 1)
 setxy xcor ycor
end
to find_food_horses
 set color blue ;humans turn blue while executing site fidelity
 if (abs xcor) < ( abs foodx + 2) and (abs xcor) > ( abs foodx - 2) [ ;if the human is within
2 pixels of last known food location, exits site fidelity behavior
    if (abs ycor) < ( abs foody + 2) and (abs ycor) > ( abs foody - 2) [
     set behavior 1
    ]
 ]
 facexy foodx foody ;moves towards last known food location
 rt (-20 + random 40)
 fd patches_to_move_horses
 set movement_horses (movement_horses + 1)
 setxy xcor ycor
end
to find_food_trucks
 set color blue ;humans turn blue while executing site fidelity
 if (abs xcor) < ( abs foodx + 2) and (abs xcor) > ( abs foodx - 2) [ ;if the human is within
2 pixels of last known food location, exits site fidelity behavior
    if (abs ycor) < ( abs foody + 2) and (abs ycor) > ( abs foody - 2) [
     set behavior 1
    ]
```

```
 ]
 facexy foodx foody ;moves towards last known food location
 rt (-20 + random 40)
 fd patches_to_move_trucks
 set movement_trucks (movement_trucks + 1)
 setxy xcor ycor
end
to stop_search?
 if random 10000 = 1 [  ;determines the percentage chance for humans to give up and
return to the city
    set behavior 3
 ]
end
to move_away
 ifelse not can-move? 1
 [ set behavior 1 ]
 [ ifelse (xcor < 0)
    [ ;probability to begin search is determined by GA and user parameters
     ifelse (random 10000 / 10000 < 1 - initial_expansion)
     [ set behavior 1 ]
     [ set behavior 1 ] ;if search mode is not engaged, moves outward at full speed
    ]
    [  ifelse (random 10000 / 10000 < 1 - GAexpand)
     [ set behavior 1 ]
     [ set behavior 1 ];if search mode is not engaged, moves outward at full speed
    ]
 ]

end
to check_food
 let x -1
 let y -1
 let seed_count 0
 let food? 0
 let p 0
 let rec_factor 0
 set recruit 0
 set fidelity 0
 if xcor = 0 [  ;defines trail creation probabiity based on Ga and user inputs
    set rec_factor lay_a_trail
 ]
 ask patch-here [ ;collects food on a patch
    if pcolor = 62[
     if count turtles-here >= 1  [
       set food? 1
       set ticks_since_harvesting 0
```

```
    ]
   ]
]
 if behavior = 1 or behavior = 2 and food? = 1[ ;executes once food has been collected
   set has_food? 1
   set seed_count 0
   set pcolor grey ;removes visual represenation of food from the patch
   while [x < 2] [ ;uses while loops to scan surrounding eight patches for food
    set y -1
    while [y < 2] [
      if (pxcor + x) > (- world-width / 2 + 1) and (pxcor + x) < (world-width / 2 - 1) [
        if (pycor + y) > (- world-height / 2 + 1) and (pycor + y) < (world-height / 2 - 1) [
          ask patch-at x y [
            if pcolor = 62 [
              set seed_count (seed_count + 1) ;number of uncollected food nearby
            ]
          ]
        ]
      ]
      set y (y + 1)
    ]
    set x (x + 1)
   ]
   set foodx xcor
   set foody ycor
   set p (random 100 / 100)
   ifelse p <=  seed_count + rec_factor [ ;function to determine wether to lay a trail
    set behavior 4
   ]
   [set behavior 3
   ]
]
 ifelse xcor = 0 [ ;probability to use site fidelity or density recruitment upon finding a seed
is determined by GA and user parameters
   if random-float 1 < ((Site_fidelity / 100) + seed_count) [
    set fidelity 1
   ]
   if random-float 1 < ((Density_recruit / 100) - seed_count) [
    set recruit 1
   ]
]
 [  if random-float 1 < (GAsite + seed_count) [
   set fidelity 1
]
 if random-float 1 < (GAtrail - seed_count) [
   set recruit 1
```

```
 ]
 ]
end
to scan_ahead ;humans look for trail directly infront of themselves
 let nx cityx
 if can-move? 1 [ ;checks for the world boundaries
    ask patch-ahead 1 [
     ifelse trail? > 0 [ ;if trail ahead, return true
       set road_ahead trail?
       set dist distancexy nx 0
       ask humans [
         set trail_ahead 1
       ]
     ]
     [ set road_ahead 0
       ask humans [
         set trail_ahead 0
       ]
     ]
    ]
 ]
end
to scan_ahead_horses ;humans look for trail directly infront of themselves
 let nx cityx
 if can-move? 1 [ ;checks for the world boundaries
    ask patch-ahead 1 [
     ifelse trail? > 0 [ ;if trail ahead, return true
       set road_ahead trail?
       set dist distancexy nx 0
       ask horses [
         set trail_ahead 1
       ]
     ]
     [ set road_ahead 0
       ask horses [
         set trail_ahead 0
       ]
     ]
    ]
 ]
end
to scan_ahead_trucks ;humans look for trail directly infront of themselves
 let nx cityx
 if can-move? 1 [ ;checks for the world boundaries
    ask patch-ahead 1 [
     ifelse trail? > 0 [ ;if trail ahead, return true
```

```
      set road_ahead trail?
      set dist distancexy nx 0
      ask trucks [
        set trail_ahead 1
      ]
    ]
    [ set road_ahead 0
      ask trucks [
        set trail_ahead 0
      ]
    ]
  ]
]
end
to scan_trail ;function to follow trail trails
 let max_road 0
 let x2 xcor
 let y2 ycor
 let nx cityx
 let tdrop 0
 set curr_distance (distancexy cityx 0) ;will not follow trails that get closer to the city
 ask neighbors [
   if distancexy nx 0 > curr_distance[
    if trail? > 0 [
      if trail? > max_road [
        set max_road trail?
      ]
    ]
   ]
]
 ifelse max_road > 0 [
   while [xcor = x2 and ycor = y2] [
    set heading (random 360)
    set color white ;turn white while following a trail for visual effect
    scan_ahead
    if road_ahead > random max_road [
      if dist >= distancexy cityx 0 [
        face patch-ahead 1
        fd distance patch-ahead 1
        set movement_humans (movement_humans + 1)
        setxy pxcor pycor
      ]
    ]
   ]
```

```
    if (random 10000) / 10000 < tdrop [ ;small chance to abandon a trail and begin
searching
      set behavior 1
      set color black
      set heading random 360
    ]
 ]
 [
    set behavior 1 ;when the trail is gone, humans revert to search behavior
    set color black
    set heading random 360
 ]
end
to scan_trail_horses ;function to follow trail trails
 let max_road 0
 let x2 xcor
 let y2 ycor
 let nx cityx
 let tdrop 0
 set curr_distance (distancexy cityx 0) ;will not follow trails that get closer to the city
 ask neighbors [
    if distancexy nx 0 > curr_distance[
     if trail? > 0 [
       if trail? > max_road [
         set max_road trail?
       ]
     ]
    ]
 ]
 ifelse max_road > 0 [
    while [xcor = x2 and ycor = y2] [
     set heading (random 360)
     set color white ;turn white while following a trail for visual effect
     scan_ahead_horses
     if road_ahead > random max_road [
       if dist >= distancexy cityx 0 [
         face patch-ahead 1
         fd distance patch-ahead 1
         set movement_horses (movement_horses + 1)
         setxy pxcor pycor
       ]
     ]
    ]

    if (random 10000) / 10000 < tdrop [ ;small chance to abandon a trail and begin
searching
```

```
     set behavior 1
     set color brown
     set heading random 360
    ]
 ]
 [
    set behavior 1 ;when the trail is gone, humans revert to search behavior
    set color brown
    set heading random 360
 ]
end
to scan_trail_trucks ;function to follow trail trails
 let max_road 0
 let x2 xcor
 let y2 ycor
 let nx cityx
 let tdrop 0
 set curr_distance (distancexy cityx 0) ;will not follow trails that get closer to the city
 ask neighbors [
    if distancexy nx 0 > curr_distance[
     if trail? > 0 [
       if trail? > max_road [
         set max_road trail?
       ]
     ]
    ]
 ]
 ifelse max_road > 0 [
    while [xcor = x2 and ycor = y2] [
     set heading (random 360)
     set color white ;turn white while following a trail for visual effect
     scan_ahead_trucks
     if road_ahead > random max_road [
       if dist >= distancexy cityx 0 [
         face patch-ahead 1
         fd distance patch-ahead 1
         set movement_trucks (movement_trucks + 1)
         setxy pxcor pycor
       ]
     ]
   ]

    if (random 10000) / 10000 < tdrop [ ;small chance to abandon a trail and begin
searching
     set behavior 1
     set color red
```

```
     set heading random 360
    ]
 ]
 [
    set behavior 1 ;when the trail is gone, humans revert to search behavior
    set color red
    set heading random 360
 ]
end
to color_trail
 ask patch-here [
    if ((pycor != 0) or (pxcor != -100)) and ((pycor != 0) or (pxcor != 100)) [ ;will not lay trail
ontop of the city
     ;function to lay down trail during behvaior 3
     set trail? (trail? + 1)  ;increments trail by 1 every tick
     if pcolor != 62 [   ;only draws trail on pixels without food
       if trail? >= 6 [set pcolor 99]     ;gradual progression from dark blue to white based
on trail strength
       if trail? >= 5 and trail? < 6 [set pcolor 98]
       if trail? >= 4 and trail? < 5 [set pcolor 97]
       if trail? >= 3 and trail? < 4 [set pcolor 96]
       if trail? >= 2 and trail? < 3 [set pcolor 95]
       if trail? >= 1 and trail? < 2 [set pcolor 94]
       if trail? < 1  and pcolor = grey [set pcolor 93]   ;will not draw trail over food, only
grey space
                                                              ; set trail_evaporation  trail_evaporation
     ]
    ]
 ]
end
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;plotting and data-exporting;;;;;;;;plotting and data-exporting;;;;;;;;;;;;;;;;plotting and
data-exporting;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to plot_joules_vs_time
   set-current-plot "Gigajoules expended v. Time" ;plot name
   set-current-plot-pen "humans"
    plot-pen-down
    plotxy ticks gigajoules_humans
   set-current-plot-pen "horses"
    plot-pen-down
    plotxy ticks gigajoules_horses
   set-current-plot-pen "trucks"
    plot-pen-down
    plotxy ticks gigajoules_trucks
```

```
end
to plot_joules_vs_distance
   set-current-plot "Gigajoules expended v. Greatest length of food network" ;plot name
   set-current-plot-pen "humans"
    plot-pen-down
    plotxy greatest_distance gigajoules_humans
   set-current-plot-pen "horses"
    plot-pen-down
    plotxy greatest_distance gigajoules_horses
   set-current-plot-pen "trucks"
    plot-pen-down
    plotxy greatest_distance gigajoules_trucks

end
to plot_joules_per_food_vs_distance
   set-current-plot "Gigajoules expended to harvest 1 km v. Greatest length of food
network" ;plot name
   set-current-plot-pen "humans"
    plot-pen-down
    plotxy greatest_distance food_joule_ratio_humans
    set-current-plot-pen "horses"
    plot-pen-down
    plotxy greatest_distance food_joule_ratio_horses
   set-current-plot-pen "trucks"
    plot-pen-down
    plotxy greatest_distance food_joule_ratio_trucks

end
```