**!! READ ME !!**

This file contains an explanation of the different procedures to be developed for the “Inhabitant-agent” in the CitySim ABM.

General explanation is in black. Netlogo code is in red. The procedures are numbered. These numbers correspond to those in the flowchart.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;

**;; 1 PEER CALCULATIONS**

;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Agents “interact” mainly with their peer network. Agents compare their experiences, decisions and preferences with their peers. The outcome determines their satisfaction level as well as the uncertainty they experience. Their relative satisfaction level and relative uncertainly then determine their behavioral strategy.

First, all agents select peers.

1. **To select peers**

Peers will be selected every time step (or before every relevant decision). Selecting peers is based on the following four factors:

1a*) difference in travel mode preference* (so if agents have the same preference hey are more likely to become peers) Travel mode preference is fixed throughout the simulation and will be set randomly at the start of the simulation)

*1b) difference in travel mode selected,* so if agents have the same travel mode they are more likely to become peers). Travel mode is variable and is calculated every time step (potentially even several times in one time step- every time an agent needs to decide on a new travel.

*1c) distance between my home and home of potential peer* (so if agents live near each other they are more likely to become peers) .

*1d) relative difference in home-work distanc*e. (so if agents travel similar distance too work they are more likely to become peers) .

These four factors will then we used to calculate a difference score expressed as the difference between the agent and a potential peer. The 5 agents with the lowest “difference” score will be selected as peers.

Travel mode (and travel mode preferences) are expressed as:

1 = bike;

2 = walk;

3 = Public Transport;

4 = car

The following code is from how I calculated things in Netlogo.

repeat (count inhabitant) [ ask one-of inhabitant with [ already\_selected\_peers? = false ] [ select\_peers ] ]

let prey other inhabitant

if prey != nobody

[ set already\_selected\_peers? true

ask prey [

**1a) Relative difference in travel mode preference.** This calculates the difference between travel-mode preferences.

ifelse (inhabitant\_travel\_mode\_preference - [travel\_mode \_preference] of myself ) = 0

[ set relative\_difference\_in\_travel\_mode \_preference 0 ]

[ ifelse ( abs ( inhabitant\_travel\_mode \_preference - [inhabitant \_travel\_mode \_preference] of myself )) = 1

[ set relative\_difference\_in\_travel\_mode \_preference 0.33 ]

[ifelse ( abs (inhabitant\_travel\_mode \_preference - [inhabitant\_travel\_mode \_preference] of myself ) ) = 2

[ set relative\_difference\_in\_travel\_mode \_preference 1 ]

[ set relative\_difference\_in\_travel\_mode \_preference 0.67 ]

**1b) Travel mode selected.** This calculates the difference between travel-mode.

ifelse (inhabitant\_travel\_mode - [travel\_mode] of myself ) = 0

[ set relative\_difference\_in\_travel\_mode 0 ]

[ ifelse ( abs ( inhabitant\_travel\_mode - [ inhabitant\_travel\_mode] of myself )) = 1

[ set relative\_difference\_in\_travel\_mode 0.33 ]

[ifelse ( abs (inhabitant\_travel\_mode - [inhabitant\_travel\_mode] of myself ) ) = 2

[ set relative\_difference\_in\_travel\_mode 1 ]

[ set relative\_difference\_in\_travel\_mode 0.67 ]

**1c) Distance between** inhabitant **home (here=myself) and home of potential (here=self) peer.** This calculates if agents live close to each other. The closer they, the smaller the difference between them and the more likely they will be to become peers.

The max distance that can still positively impact this factor needs to be checked / discussed.. Now set to 2 km. Basically if the distance is less than two it will positively affect the change to become a peer (the smaller the difference the larger the change)

( ifelse ( abs (([ xcor] of self ) - ([ xcor] of myself )) > 2 )

[ set relative\_xcor\_distance 1 ]

[ set relative\_xcor\_distance ( abs (([ xcor] of self ) - ([ xcor] of myself )) ) / 2 ] )

( ifelse ( abs (([ ycor] of self ) - ([ ycor] of myself )) > 2 )

[ set relative\_ycor\_distance 1 ]

[ set relative\_ycor\_distance ( abs (([ ycor] of self ) - ([ ycor] of myself )) ) / 2 ] )

**1d) relative difference in home-work distance.** This calculates the difference between travel distances from agent and a potential peer. The smaller the difference, the more likely they will be to become peers.

The distance between inhabitant-agents that can still positively impact this factor needs to be checked / discussed.. Now set to 3 km. Basically if the distance is less than two it will positively affect the change to become a peer (the smaller the difference the larger the change)

( ifelse ([home-work distance] of self ) - ([home-work distance] of myself ) > 3)

[ set relative\_difference\_in\_home-work distance 1 ]

[ set relative\_difference\_in\_home-work distance ( 1 - ( ([home-work distance] of self ) / ([home-work distance] of myself )) ) ] )

**Then to calculate /identify peers:**

Set difference\_to\_inhabitant\_to\_select\_peers

( relative\_difference\_in\_travel\_mode \_preference +

relative\_difference\_in\_travel\_mode +

( (relative\_xcor\_distance + relative\_ycor\_distance ) / 2) + relative\_difference\_in\_home-work distance )

set inhabitant\_current\_peers min-n-of 5 prey [difference\_to\_inhabitant\_to\_select\_peers ]

The following code is for the agents and the peers to keep track of their peers and from who they have been peers. This to allow for a network analysis at the end of the simulation.

ask inhabitant\_current\_peers [

set list\_of\_other\_inhabitants\_inhabitant\_is\_currently\_peer\_of lput ([who] of myself) list\_of\_other\_inhabitants\_ inhabitants\_inhabitant\_is\_currently\_peer\_of

set list\_of\_other\_ inhabitants\_inhabitant \_was\_peer\_of lput ([who] of myself) list\_of\_other\_ inhabitants\_inhabitant \_was\_peer\_of ]

set list\_of\_inhabitant\_current\_peers lput ([who] of inhabitant\_current\_peers) list\_of\_inhabitant\_current\_peers

set list\_of\_inhabitant\_overall\_peers lput ([who] of inhabitant\_current\_peers) list\_of\_inhabitant\_overall\_peers

repeat (count inhabitant\_current\_peers) [ let prey2 one-of inhabitant\_current\_peers with [with\_link\_to != ([who] of myself)]

if prey2 != nobody

[ if (Peer\_link = true) [ create-link-with prey2 ]

ask prey2 [ set with\_link\_to ([who] of myself) ]

] ] ]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;

**;; NEED CALCULATIONS**

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Agent calculate an need satisfaction which is build up out of:

* Social need

Social need has to do with satisfaction related to doing things similar and superior to peers/others

* Personal need

Personal satisfaction has to do with what one likes to do

* Existence need

Existence need satisfaction has to do with

These three needs are then joint in an **overall need satisfaction** calculation and compared to the “aspiration level” of the agent (to calculate the “relative overall need satisfaction”). The aspiration level is fixed throughout the simulation and randomly set per agent at the beginning. The value is between 0-1.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;

**;; 2 SOCIAL NEED CALCULATIONS**

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

**2 To calculate social need satisfaction**

An agent social need satisfaction relates to the level of satisfaction an agent gets from a specific choice in relation to the choices made by its peers. Agents generally feel more satisfied with their own choices when their choice is relatively similar to that of its peers, while at the same time they feel more satisfied when they are performing slightly better than their peers.

These two factors(similarity and superiority) are calculated to calculate the social need satisfaction. Similarity is based on selected travel mode, whereas superiority is based on travel-speed (so travel time/ travel distance). Values are between 0 and 1. With 0 being the best and 1 the worst.

**2a) Similarity among peers**

to calculate\_similarity\_to\_peers

ifelse (abs (inhabitant\_travel\_mode – [travel\_mode ] of myself)) = 0

[ set relative\_difference\_to\_current\_peers\_travel\_mode 0 ]

[ ask inhabitant\_current\_peers

[ set relative\_difference\_to\_current\_peers\_travel\_mode

( (abs (inhabitant\_travel\_mode – [travel\_mode ] of myself )) / max\_difference\_travel\_mode)

;;travel mode can be 1,2,3 or 4.. max difference is therefore 3; if A is 1 and B is 4

set similar\_to\_peers\_ratio (( sum [ relative\_difference\_to\_current\_peers travel\_mode] of inhabitant\_current\_peers) / (count inhabitant\_current\_peers) )

**2b) Superiority to peers**

to calculate\_superiority\_to\_peers

ifelse ( inhabitant\_relative\_travel\_time > (( sum [inhabitant\_relative\_travel\_time ] of inhabitant\_current\_peers) / (count inhabitant\_current\_peers)))

[ set superior\_to\_peers\_ratio 0 ]

[ set superior\_to\_peers\_ratio ( inhabitant\_relative\_travel\_time / (( sum [inhabitant\_relative\_travel\_speed ] of inhabitant\_current\_peers) / (count inhabitant\_current\_peers)) ) ]

**to calculate\_social\_need\_satisfaction**

The total social need satisfaction is then calculated as the average of the two values.

set inhabitant\_social\_need\_satisfaction ((superior\_to\_peers\_ratio + similar\_to\_peers\_ratio) / 2)

;; being similar and superior to peers are equally important. The ratio will be between 0-1

**end**

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;

**;; 3 PERSONAL NEED CALCULATIONS**

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

**to calculate\_personal\_need\_satisfaction**

The personal satisfaction an agent experiences relates to doing what one loves. In general, agents feel more satisfied with their choices if they are (more) similar to their preferred activities.

Here, the personal need satisfaction relates to travel mode and preferred travel mode.

ifelse (abs ([inhabitant\_travel\_mode] of myself – [inhabitant\_travel\_mode\_preference ] of myself)) = 0

[ set relative\_difference\_to\_current\_peers\_travel\_mode 0 ]

[ set relative\_difference\_to\_current\_peers\_travel\_mode

( (abs ([ inhabitant\_travel\_mode ] of myself – [inhabitant\_travel\_mode\_preference ] of myself )) / max\_difference\_travel\_mode)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;

**;; 4 EXISTENCE NEED CALCULATIONS**

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

The existence need satisfaction an agent experiences relates to the economic means of existence. Here, this is interpreted as being efficient with one time and is therefore expressed as relative speed with which on travels to work/home.

**to calculate\_existence\_need\_satisfaction**

ifelse ( inhabitant\_relative\_travel\_speed =< relative\_speed\_per\_travel\_mode )

[ set inhabitant\_existence\_need\_satisfaction 0 ]

[ set inhabitant\_existence\_need\_satisfaction (inhabitant\_relative\_travel\_speed / speed\_per\_travel\_mode )]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;

**;; 5 TOTAL NEED CALCULATIONS & RATIO**

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Agents then calculate the overall need satisfaction level as a combination of the three need satisfaction levels times their respective relative importance. The relative importance per need are set at the beginning of the simulation and are the same for all agents (they are global variables) and do not change throughout the simulation. For this case, I think it makes sense that the relative importance of the needs are equal, so all 0.33.

Ps: remember that for all need satisfaction levels 0 = the best, and 1 = the worst

**to calculate\_overall\_need\_satisfaction**

set inhabitants\_overall\_need\_satisfaction

( ( inhabitant\_relative\_importance\_for\_existence\_need \* inhabitant\_existence\_need\_satisfaction ) +

( inhabitant\_relative\_importance\_for\_social\_need \* inhabitant\_social\_need\_satisfaction ) +

( inhabitant\_relative\_importance\_for\_personal\_need \* inhabitant\_personal\_need\_satisfaction ) )

**end**

The overall need satisfaction level is then related to the agent aspiration level. The aspiration level is an agent variable which fixed throughout the simulation and differs per agent. The aspiration level is randomly set between 0 and 1.

In order to calculate the relative overall need satisfaction in an intuitive manner, the overall need satisfaction is subtracted from 1.

**to calculate\_relative\_overall\_need\_satisfaction**

ifelse ( ( 1 - inhabitant\_overall\_need\_satisfaction ) > inhabitant\_aspiration\_level)

[ set inhabitant\_overall\_need\_satisfaction\_aspiration\_level\_ratio 0.75 ]

[ set inhabitant\_overall\_need\_satisfaction\_aspiration\_level\_ratio 0.25 ]

**end**

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

**;; 6 UNCERTAINTY CALCULATIONS & RATIO**

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Agents experience uncertainty from two factors: 1) difference between their travel speed expectation from that of their peers, and 2) difference between their current travel mode and that of their peers. Basically, the higher the similarity with their peers in relation to their expectations and their choices, the less uncertain they feel.

All inhabitants start with a memory of travel speed per travel mode, this can be based on average speed per travel mode. The memory of agents relates to their cognitive effort and differs between agents. The range is from 1-10 items. Every time an agent travels to work, the memory is updated by deleting the oldest memorised event and adding the newest memory. The expected travel speed for the next trip is then calculated through a linear function of the memorised travel speeds.

Both factors are calculated as follows:

**6a To calculate difference between agent speed expectation from that of its peers**

First, calculate the **expected travel speed for next trip** (per travel mode) using a simple linear function. Here I quickly put is, but should be a longer ifelse

Ifelse ( travel mode = 1 )

[ set expected\_travel\_speed (matrix:forecast-linear-growth memorized\_speed\_travel\_speed\_travel\_mode\_1)

blabla

Then the agent lists the average expected travel speeds (per travel mode) of itself and its peers. The uncertainty is then expressed through the **coefficient of variation** (standard deviation / mean).

[ set list\_of\_peer\_expected\_travel\_speed sentence expected\_travel\_speed [expected\_travel\_speed ] of inhabitant\_current\_peers

set list\_of\_peer\_and\_self\_expected\_travel\_speed lput expected\_travel\_speed of myself

set standard\_deviation\_of\_expected\_travel\_speed standard-deviation of list\_of\_peer\_and\_self\_expected\_travel\_speed

set average\_expected\_travel\_speed ( (reduce [?1 + ?2] list\_of\_peer\_and\_self\_expected\_travel\_speed ) / (count peers + 1) )

set uncertainty\_travel\_speed ( standard\_deviation\_of\_expected\_travel\_speed\_ / average\_expected\_travel\_speed) ]

;; if the uncertainty an agent experiences is larger than its uncertainty tolerance level, the agent feels uncertain, whereas if the uncertainty experienced by the agent is less than its uncertainty tolerance level the agent does not feel uncertain.

to calculate\_ratio\_uncertainty\_uncertainty\_tolerance\_level

ifelse ( inhabitant\_uncertainty > inhabitant\_uncertainty\_tolerance\_level)

[ set inhabitant\_uncertainty\_'uncertainty\_tolerance\_level'\_ratio 0.75 ]

[ set inhabitant\_uncertainty\_'uncertainty\_tolerance\_level'\_ratio 0.25 ]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

**;; 7 BEHAVIORAL STRATEGY CHOICE**

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Depending on the relative satisfaction (=inhabitant\_overall\_need\_satisfaction\_aspiration\_level\_ratio) and the relative uncertainty (=inhabitant\_uncertainty\_'uncertainty\_tolerance\_level'\_ratio) agents experience, they follow one of the four bahavioral options: *repeat, imitate, optimise, inquire.* The ratios range between 0-1. The divide between the four strategies was made with a hard cut at 0.5. To clearly show the strategies of the agents, agents have a specific color showing their strategy.

if (( inhabitant\_overall\_need\_satisfaction\_aspiration\_level\_ratio > 0.5) and (inhabitant\_uncertainty\_'uncertainty\_tolerance\_level'\_ratio <= 0.5))

[ repeate

set color black

set inhabitant\_behavioral\_option "repeat" ]

if ((inhabitant\_overall\_need\_satisfaction\_aspiration\_level\_ratio > 0.5) and (inhabitant\_uncertainty\_'uncertainty\_tolerance\_level'\_ratio > 0.5))

[ imitate

set color sky

set inhabitant\_behavioral\_option "imitate" ]

if (( inhabitant\_overall\_need\_satisfaction\_aspiration\_level\_ratio <= 0.5) and (inhabitant\_uncertainty\_'uncertainty\_tolerance\_level'\_ratio <= 0.5))

[ optimise

set color magenta

set inhabitant\_behavioral\_option "optimise" ]

if (( inhabitant\_overall\_need\_satisfaction\_aspiration\_level\_ratio <= 0.5) and (inhabitant\_uncertainty\_'uncertainty\_tolerance\_level'\_ratio > 0.5))

[ inquire

set color yellow

set inhabitant\_behavioral\_option "inquire" ]



Relative behavioral options calculated in order to make graphs etc.

set relative\_repeat\_behaviour ( (count inhabitant with [ inhabitant\_behavioral\_option = "repeat" ]) / (count inhabitant))

set relative\_imitate\_behaviour ( (count inhabitant with [ inhabitant\_behavioral\_option = "imitate" ]) / (count inhabitant))

set relative\_inquire\_behaviour ( (count inhabitant with [ inhabitant\_behavioral\_option = "inquire" ]) / (count inhabitant))

set relative\_optimise\_behaviour ( (count inhabitant with [ inhabitant\_behavioral\_option = "optimise" ]) / (count inhabitant))

**;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;**

**;;**

**;; BEHAVIORAL STRATEGY CALCULATIONS**

**;;**

**;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;**

To **repeat** means to have the same travel mode as the trip before.

to repeate

set next\_trip\_­travel\_mode travel\_mode

end

To **imitate** means doing what the majority of the agent peers does.

To calculate this, my Netlogo code is a rather indirect.. I makes lists to first identify the travel mode used by the majority of the agent peers in the previous round

and then to set the next trips travel mode to that travel mode.

to imitate

set list\_of\_travel\_mode\_options [ 1 2 3 4 5..etc ] ;; make a list with all travel-modes

set relative\_peers\_travel\_mode\_1 (count peers with [ travel\_mode = 1] / ( count peers))

set relative\_peers\_travel\_mode\_2 (count peers with [ travel\_mode = 2] / ( count peers))

set relative\_peers\_ravel\_mode\_3 (count peers with [ travel\_mode = 3] / ( count peers)) ;; do this for all travel-modes

set list\_of\_relative\_peers\_travel\_modes lput relative\_ peers\_travel\_mode\_1 relative\_ peers\_travel\_mode\_2..etc ;; make a list of all relative travel modes

let x ( map [ (list ?1 ?2) ] list\_of\_relative\_peers\_travel\_modes list\_of\_travel\_mode\_options ) ;; join the relative travel mode list with the list of travel mode option

set list\_of\_relative\_peers\_travel\_modes \_&\_connected\_travel\_mode\_options sort-by [ first ?1 > first ?2 ] x ;; sort the list on the largest proportion of peers

set next\_trip\_pre\_travel\_mode item 0 list\_of\_relative\_ peers\_travel\_modes \_&\_connected\_travel\_mode\_options ;; set the next rounds “pre” travel mode to the first item on that list

set next\_trip\_travel\_mode last next\_trip\_pre\_travel\_mode ;; set the actual next rounds travel mode to the last item on that list

end

To **inquire** here means assessing which travel-modes are currently selected by any of the other agents and then evaluating the potential satisfaction levels of all these travel-modes.

**to inquire**

If (count inhabitant with [ travel\_mode = 1 ] > 0

[ calculate\_potential\_personal\_need\_satisfaction\_travel\_mode\_1 ;; these procedures are explained below!!

calculate\_potential\_social\_need\_satisfaction\_travel\_mode\_1

calculate\_potential\_existence\_need\_satisfaction\_travel\_mode\_1

calculate\_potential\_overall\_need\_satisfaction\_travel\_mode\_1

set list\_of\_travel\_mode\_options\_inquire lput travel\_mode\_1

set list\_of\_potential\_overall\_need\_satisfaction lput potential\_overall\_need\_satisfaction\_travel\_mode\_1 ]

If (count inhabitant with [ travel\_mode = 2 ] > 0

[ calculate\_potential\_personal\_need\_satisfaction\_travel\_mode\_2 ;; these procedures are explained below!!

calculate\_potential\_social\_need\_satisfaction\_travel\_mode\_2

calculate\_potential\_existence\_need\_satisfaction\_travel\_mode\_2

calculate\_potential\_overall\_need\_satisfaction\_travel\_mode\_2

set list\_of\_travel\_mode\_options\_inquire lput travel\_mode\_2

set list\_of\_potential\_overall\_need\_satisfaction lput potential\_overall\_need\_satisfaction\_travel\_mode\_2 ]

etc

**ADD all other travel modes**

let x ( map [ (list ?1 ?2) ] list\_of\_potential\_overall\_need\_satisfaction list\_of\_travel\_mode\_options\_inquire ) ;; join the relative travel mode list with the list of travel mode option

set list\_of\_potential\_overall\_need\_satisfaction \_&\_connected\_travel\_mode\_options sort-by [ first ?1 > first ?2 ] x ;; sort the list on the largest proportion of peers

set next\_trip\_pre\_travel\_mode item 0 list\_of\_potential\_overall\_need\_satisfaction \_&\_connected\_travel\_mode\_options ;; set the next rounds “pre” travel mode to the first item on that list

set next\_trip\_travel\_mode last next\_trip\_pre\_travel\_mode ;; set the actual next rounds travel mode to the last item on that list

**end**

To **optimise**

calculate\_potential\_personal\_need\_satisfaction\_travel\_mode\_1

calculate\_potential\_social\_need\_satisfaction\_travel\_mode\_1

calculate\_potential\_existence\_need\_satisfaction\_travel\_mode\_1

calculate\_potential\_overall\_need\_satisfaction\_travel\_mode\_1

calculate\_potential\_personal\_need\_satisfaction\_travel\_mode\_2 ;; these procedures are explained below!!

calculate\_potential\_social\_need\_satisfaction\_travel\_mode\_2

calculate\_potential\_existence\_need\_satisfaction\_travel\_mode\_2

calculate\_potential\_overall\_need\_satisfaction\_travel\_mode\_2

**etc for all travel modes**

set list\_of\_potential\_overall\_need\_satisfaction lput potential\_overall\_need\_satisfaction\_travel\_mode\_1 potential\_overall\_need\_satisfaction\_travel\_mode\_2 etc. ;; add all potential overall need satisfaction of all travel modes

let x ( map [ (list ?1 ?2) ] list\_of\_relative\_peers\_travel\_modes list\_of\_travel\_mode\_options ) ;; join the relative travel mode list with the list of travel mode option

set list\_of\_potential\_overall\_need\_satisfaction \_&\_connected\_travel\_mode\_options sort-by [ first ?1 > first ?2 ] x ;; sort the list on the largest proportion of peers

set next\_trip\_pre\_travel\_mode item 0 list\_of\_potential\_overall\_need\_satisfaction \_&\_connected\_travel\_mode\_options ;; set the next rounds “pre” travel mode to the first item on that list

set next\_trip\_travel\_mode last next\_trip\_pre\_travel\_mode ;; set the actual next rounds travel mode to the last item on that list

**end**

The following sub-procedure are needed for the inquire and optimize functions. Here, I only wrote the code for travel\_mode\_1. The same codes need to be done for all travel modes..:

**to calculate\_potential\_personal\_need\_satisfaction\_travel\_mode\_1**

ifelse (abs ( 1 – [inhabitant\_travel\_mode\_preference ] of myself)) = 0

[ set relative\_difference\_to\_current\_peers\_travel\_mode 0 ]

[ set relative\_difference\_to\_current\_peers\_travel\_mode

( (abs ( 1 – [inhabitant\_travel\_mode\_preference ] of myself )) / max\_difference\_travel\_mode)

**end**

**to calculate\_potential\_social\_need\_satisfaction\_travel\_mode\_1**

set potential\_similarity\_with\_travel\_mode\_1 ( count (inhabitant\_current\_peers with [travel\_mode = 1 ]) / ( count inhabitant\_current\_peers))

set potential\_social\_need\_satisfaction\_travel\_mode\_1 (( potential\_similarity\_with\_travel\_mode\_1 + superior\_to\_peers\_ratio) / 2)

**end**

**to calculate\_potential\_existence\_need\_satisfaction\_travel\_mode\_1**

set inhabitant\_expected\_relative\_travel\_speed\_travel\_mode\_1 (matrix:forecast-linear-growth remerbered\_speed\_travel\_mode\_1 )

ifelse ( inhabitant\_expected\_relative\_travel\_speed\_travel\_mode\_1 =< relative\_speed\_per\_travel\_mode )

[ set inhabitant\_potential\_existence\_need\_satisfaction 0 ]

[ set inhabitant\_potential\_existence\_need\_satisfaction ( inhabitant\_expected\_relative\_travel\_speed\_travel\_mode\_1 / speed\_per\_travel\_mode )]

end

**to calculate\_potential\_overall\_need\_satisfaction\_travel\_mode\_1**

set inhabitants\_potential\_overall\_need\_satisfaction\_travel\_mode\_1

( ( inhabitant\_relative\_importance\_for\_existence\_need \* inhabitant\_potential\_existence\_need\_satisfaction ) +

( inhabitant\_relative\_importance\_for\_social\_need \* inhabitant\_ potential\_social\_need\_satisfaction ) +

( inhabitant\_relative\_importance\_for\_personal\_need \* inhabitant\_ potential\_personal\_need\_satisfaction ) )

**end**