Universiteit Utrecht

Faculty of Geosciences

Integrated field-agent based modelling using the LUE scientific data base

Oliver Schmitz, Kor de Jong, Derek Karssenberg

Utrecht University, Faculty of Geosciences, Physical Geography, Utrecht, The Netherlands

Vienna, 4 May 2020

The LUE data model

Multi-paradigm modelling

Coupling fields and agents requires to represent:

- Continuous phenomena (e.g. plant biomass)
- Varying set(s) of individuals (e.g. deer)
- Interactions

The LUE data model

Requirements

Capability to represent diverse environmental data, e.g.:

- Located in (3D) space and time
- Varying continuously through space and time
- Different spatial and temporal discretisations
- Linked data: relations, networks

Allow for an efficient implementation

The LUE data model

The conceptual data model

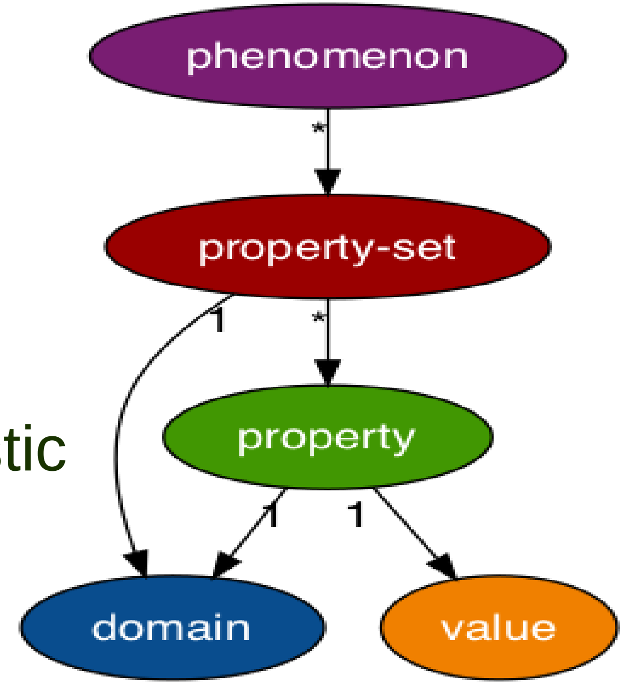
Phenomenon: Collection of related property-sets, e.g. properties of individual Zebras

Property-set: Collection of properties sharing a space/time domain

Domain: when and where something is, eg. Location of a Zebra through time

Property: Location and variation of a characteristic through time and space

Value: e.g. body temperature of a Zebra



The LUE data model

The physical data model

The LUE model is implemented in HDF5:

- All model data in a single, portable file
- C++14 API and Python API (with NumPy support)
- Open format
- Open source, participation welcome:
  - https://github.com/pcraster/lue/
  - https://lue.readthedocs.io/en/latest/index.html

Modelling environment

LUE embedded in a Python framework

Providing high level access to the LUE database

Extending Map Algebra concept to the agent-based modelling

Access using the Python dot notation, e.g.:

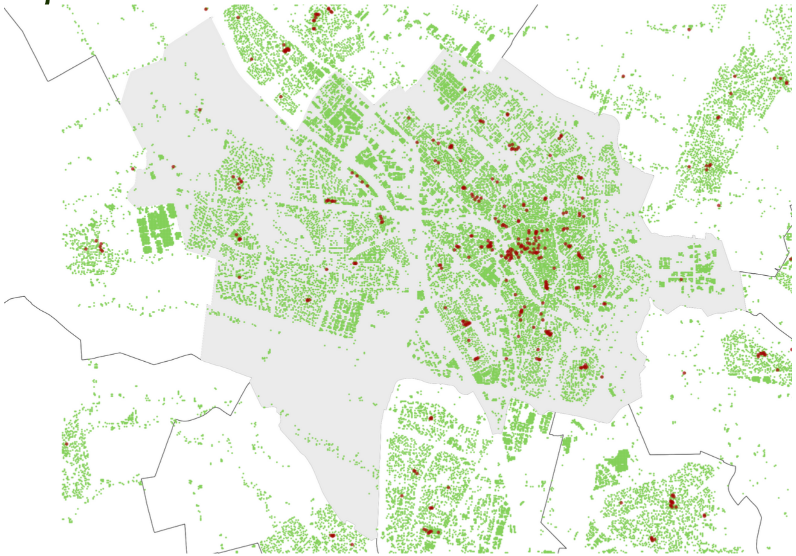
phenomenon.propertyset.property = ...

Case study
Food environments

Objective: to identify the influence of social network and existence of grocery stores on individual’s propensity on healthy food

Applied to the municipality of Utrecht, NL

Locations of ~40000 households,
 ~400 stores



Case study

Simplified model description

$$\frac{d x_i}{d t} = f(x_i) + g(x_{group})$$

x_i propensity of individual (-1,1)

t time (years)

x_{group} average propensity in group (spatial or social network)

f, g functions

Case study

Excerpt of a model script

```
def initial(self):
  self.sim = LueMemory(self.nrTimeSteps())

  # Spatial domain of frontdoor locations
  locations = Points(mobile=False)
  locations.read('houses.csv')

  # Phenomenon
  self.household = self.sim.add_phenomenon('household', locations.nr_items)

  # Property set
  self.household.add_property_set('frontdoor', locations, fame.TimeDomain.dynamic)

  # Properties
  self.household.frontdoor.add_property('propensity')
  self.household.frontdoor.add_property('default_propensity')
  self.household.frontdoor.add_property('alpha')
  self.household.frontdoor.add_property('beta')
  self.household.frontdoor.add_property('gamma')
  self.household.frontdoor.add_property('buffersize')
  self.household.frontdoor.add_property('social_neighbours', dtype=numpy.int64)
  self.household.frontdoor.add_property('neighboured_foodstores', dtype=numpy.int16)

  self.household.frontdoor.alpha = 0.15
  self.household.frontdoor.beta = 0.5
  self.household.frontdoor.gamma = 0.0
  self.household.frontdoor.buffersize = 500
  self.household.frontdoor.default_propensity = 0.4

  # Assign initial propensity around -0.17
  self.household.frontdoor.propensity = uniform(self.household.frontdoor, -0.18, -0.16)
  # Assign Watz-Strogatz network
  self.household.frontdoor.social_neighbours = neighbour_network(self.household.nr_objects, 2, 0.1, seed)
```

Model initialisation phase, defining the structure of a phenomenon and assigning initial values

Case study

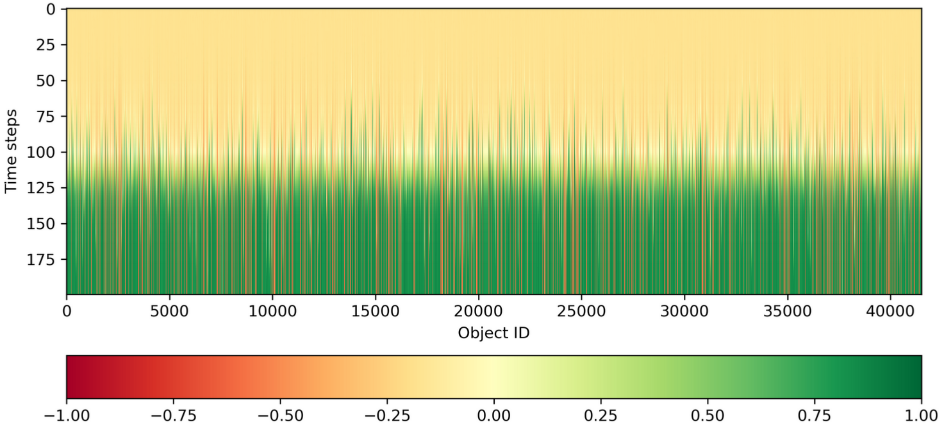
Excerpt of a model script

```
# For each time step
def dynamic(self):
  print('dynamic {}'.format(self.currentTimeStep()))

  # Operation performed for each agent in a phenomenon
  term1 = self.household.frontdoor.alpha * (self.household.frontdoor.default_propensity - self.household.frontdoor.propensity)

  # Effect of neighboured stores

  # Calculate the potential default value for households in case no food is in buffer
  total_average = property_average(self.foodstore.frontdoor.propensity)
  neighboured_store = network_average(self.household.frontdoor.neighboured_foodstores, self.foodstore.frontdoor.propensity, total_average)
  term2 =  self.household.frontdoor.beta * (neighboured_store * (1.0 - abs(self.household.frontdoor.propensity)))

  # Effect of social network
  social_neighbours = network_average(self.household.frontdoor.social_neighbours, self.household.frontdoor.propensity)
  term3 =  self.household.frontdoor.gamma * (social_neighbours_prop * (1.0 - abs(self.household.frontdoor.propensity)))

  self.household.frontdoor.propensity += self.timestep * (term1 + term2 + term3)
```

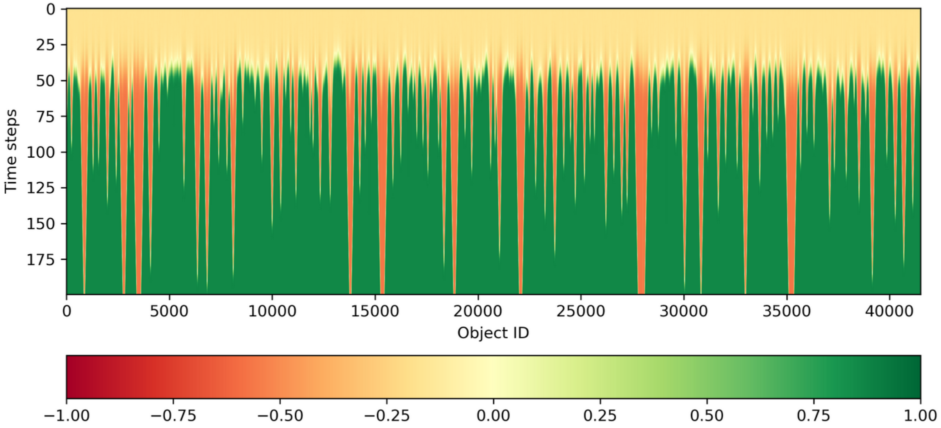
Process description executed for each time step. Each of the algebraic operations or network operations is performed on every object in a phenomenon

Case study
Exemplary output

Household propensities over time:



Neighbourhood effect of food outlets



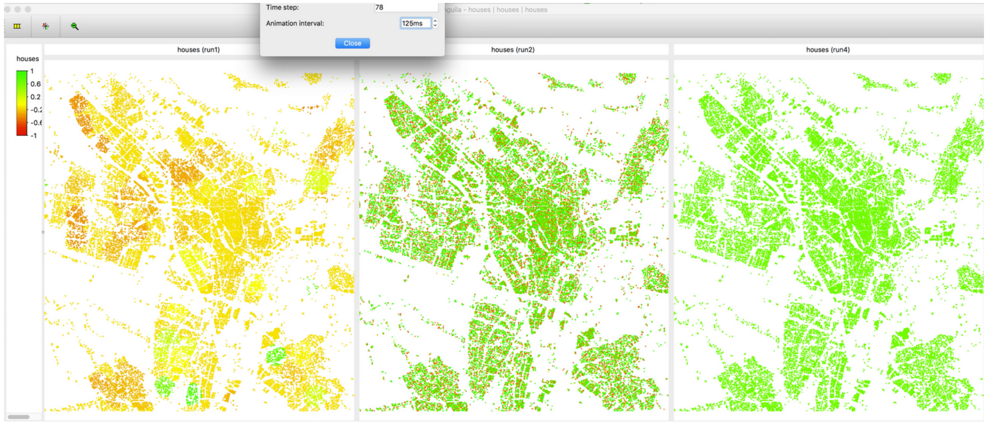
Effects of social neighbourhood included

Case study

Exemplary output

The animation (uploaded separately) shows the changes of household propensities for the municipality of Utrecht.

Left: spatial neighbourhood effects of stores
Centre: higher influence of social networks
Right: mixed influence of stores and social network



https://surfdrive.surf.nl/files/index.php/s/MqK0n6WQeeatWyK

Modelling environment
LUE embedded in a Python framework

Framework is work in progress

More operations need to be added, including spatial operations by binding PCRaster operations

Further information
LUE, framework, PCRaster

Oliver Schmitz, o.schmitz@uu.nl

http://pcraster.geo.uu.nl/
https://github.com/pcraster/lue
https://github.com/pcraster/fame
https://github.com/pcraster

de Bakker, M. P., de Jong, K., Schmitz, O., & Karssenberg, D. (2017). Design and demonstration of a data model to integrate agent-based and field-based modelling. Environmental Modelling & Software, 89, 172–189. https://doi.org/10.1016/j.envsoft.2016.11.016

de Jong, K., & Karssenberg, D. (2019). A physical data model for spatio-temporal objects. Environmental Modelling & Software. https://doi.org/10.1016/j.envsoft.2019.104553