Cyber-Physical Systems

Laura Nenzi

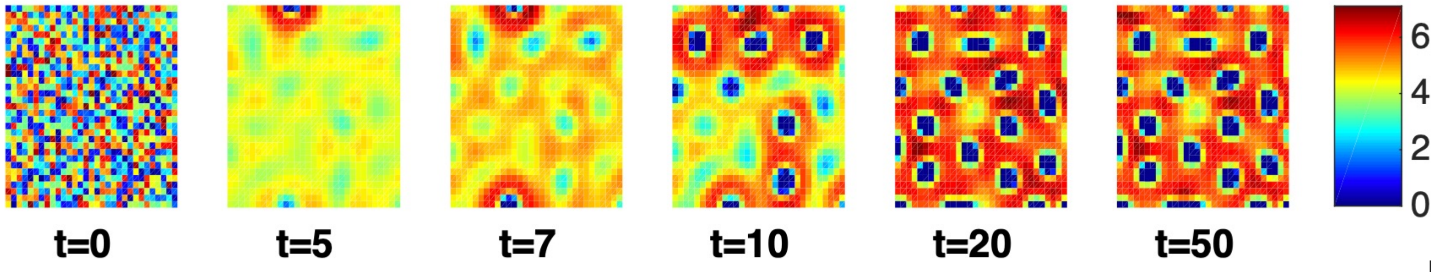
Università degli Studi di Trieste
II Semestre 2020

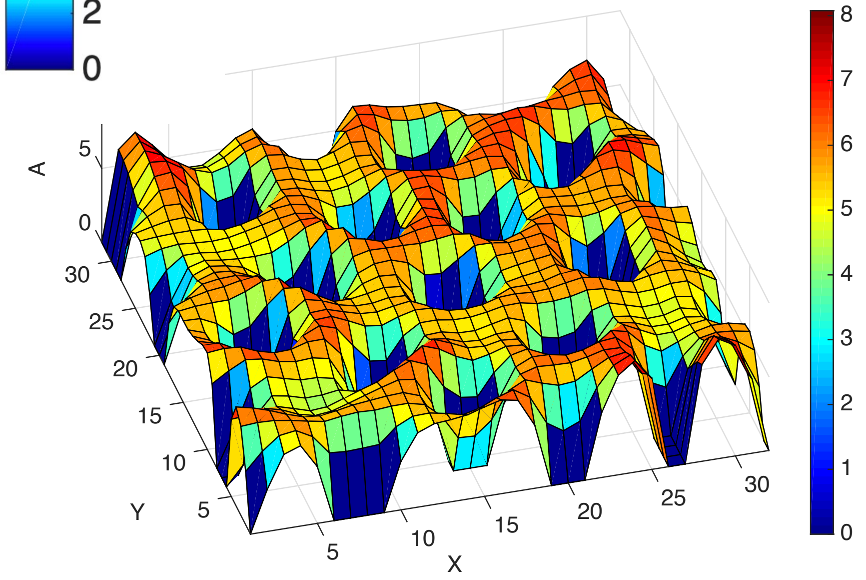
Lecture 17 (II) STREL : Spatio-Temporal Reach and Escape Logic

Static Space and Regular Grid

The formation of Patterns

The production of skin pigments that generate spots in animal furs:



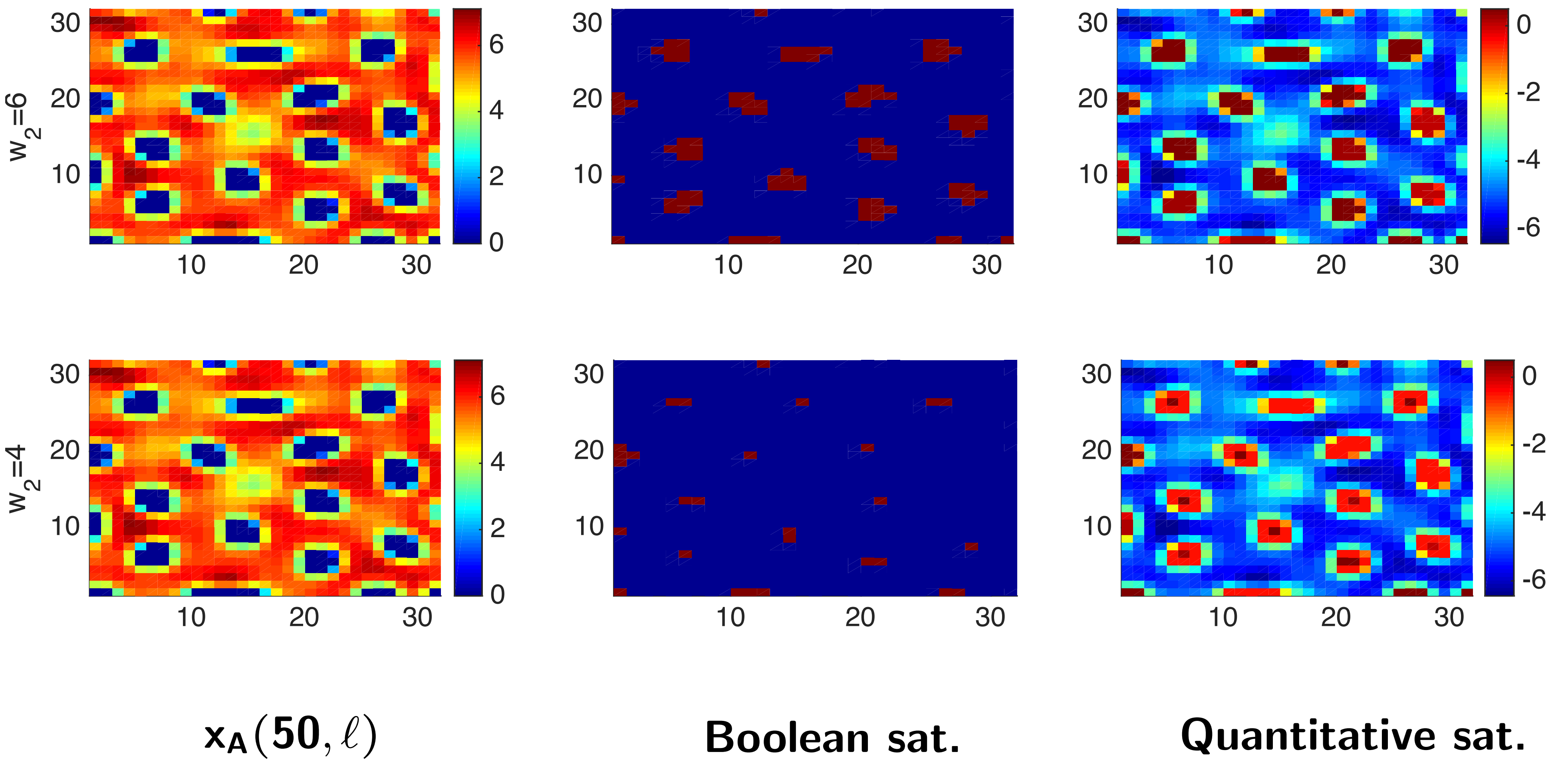


Space model: a K×K grid treated as a graph, cell$(i, j) \in L = \{1, \dots, K\} \times \{1, \dots, K\}$

Spatio-Temporal Trajectory: $x: L \mathrel{-\!>} \mathbb{T} \rightarrow \mathbb{R}^2$ s.t. $x(\ell) = (x_A, x_B)$

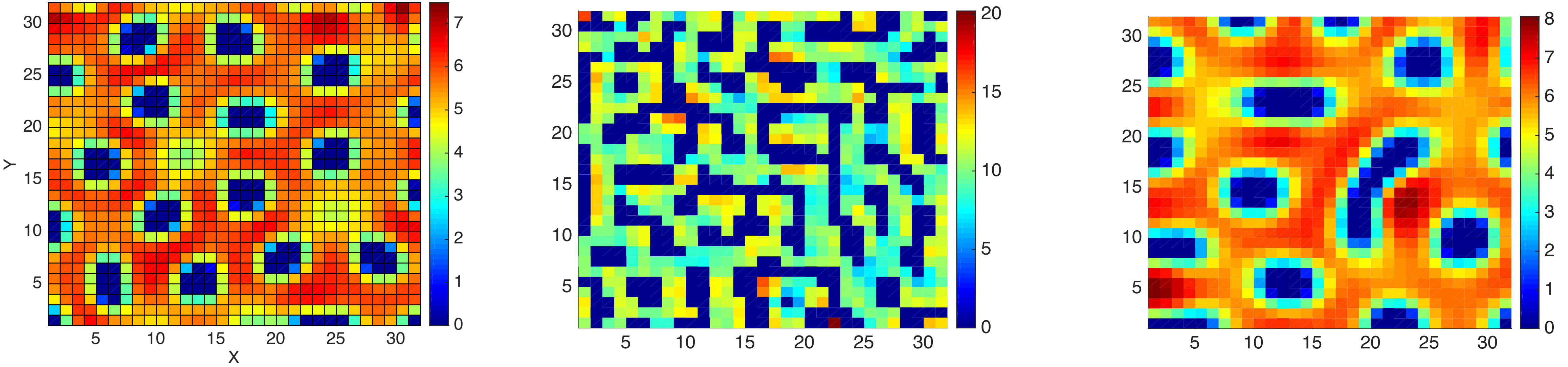
Spot formation property

$$\phi_{spot_{form}} = F_{[19,20]} G((A \leq 0.5) \circledcirc^{hops}_{[1,w_2]} (A > 0.5))$$



The formation of Patterns

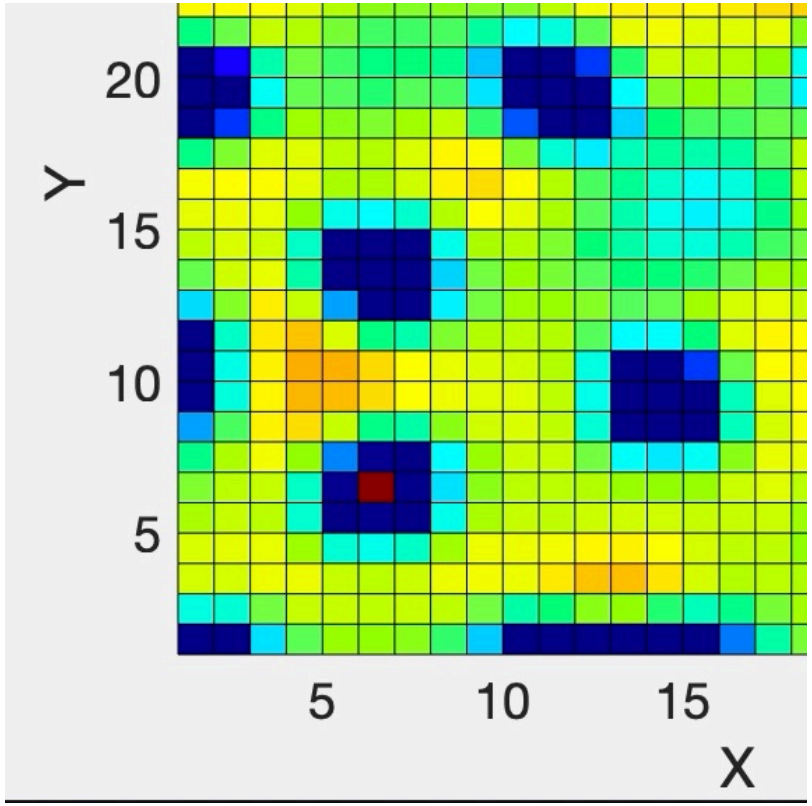
$$\phi_{pattern} := \boxdot^{hops} \Diamond^{hops}_{[0,15]} \phi_{spot_{form}}$$



Perturbation Property

$$\phi_{\mathrm{pert}} := (x_A \geq 10) \wedge (\phi_{absorb} \circledcirc^{hops}_{[1,2]} \phi_{no\_effect})$$

- $\phi_{absorb} = F_{[0,1]} G_{[0,10]} (x_A < 3);$
- $\phi_{noeffect} := G_{[0,20]} (x_A < 3)$



Static Space and Stochastic Systems

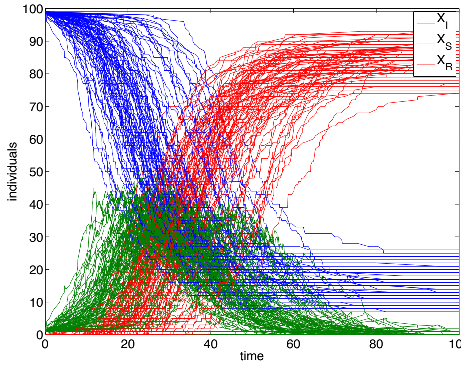
Application to Stochastic Systems

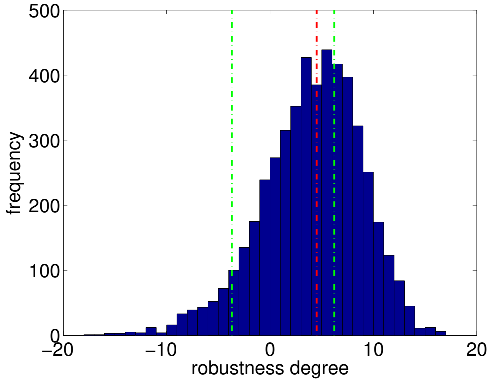
STREL can be applied on stochastic systems considering methodologies as Statistical Model Checking (SMC)

Stochastic process $\mathcal{M} = (\mathcal{T}, \mathcal{A}, \mu)$ where $\mathcal{T}$ is a trajectory space and $\boldsymbol{\mu}$ is a probability measure on a σ-algebra of $\mathcal{T}$

We approximate the satisfaction probability $S(\varphi, t)$, i.e. the probability that a trajectory generated by the stochastic process $\mathcal{M}$ satisfies the formula ϕ.

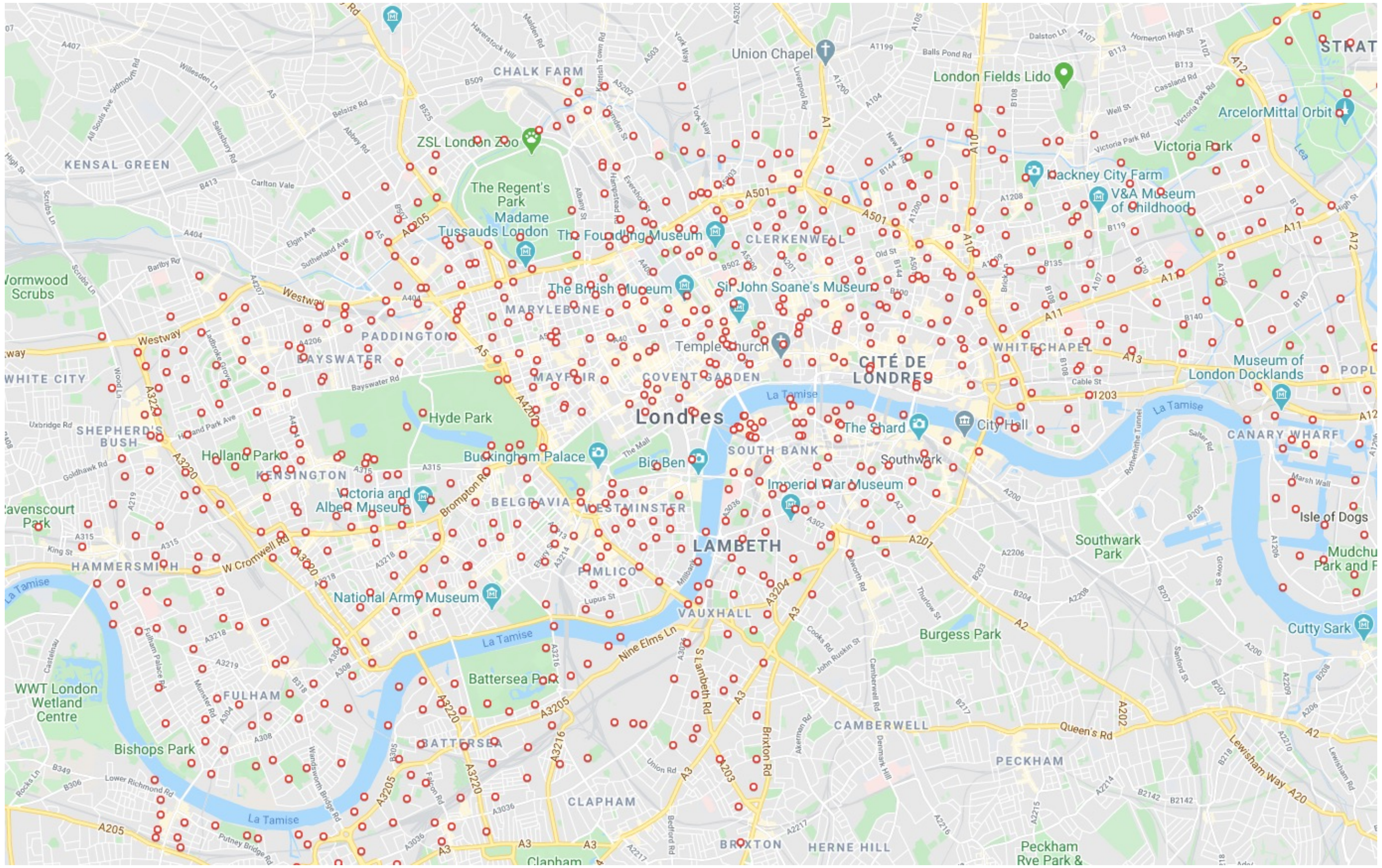
We can do something similar with the quantitative semantics computing the robustness distribution





Bike Sharing Systems (BSS)

London Santander Cycles Hire network



- 733 bike stations (each with 20-40 slots)
- a total population of 57,713 agents (users) picking up and returning bikes

We model it as a Population Continuous Time Markov Chain (PCTMC) with time-dependent rates, using historic journey and bike availability data.

Prediction for 40 minutes.

Bike Sharing Systems (BSS)

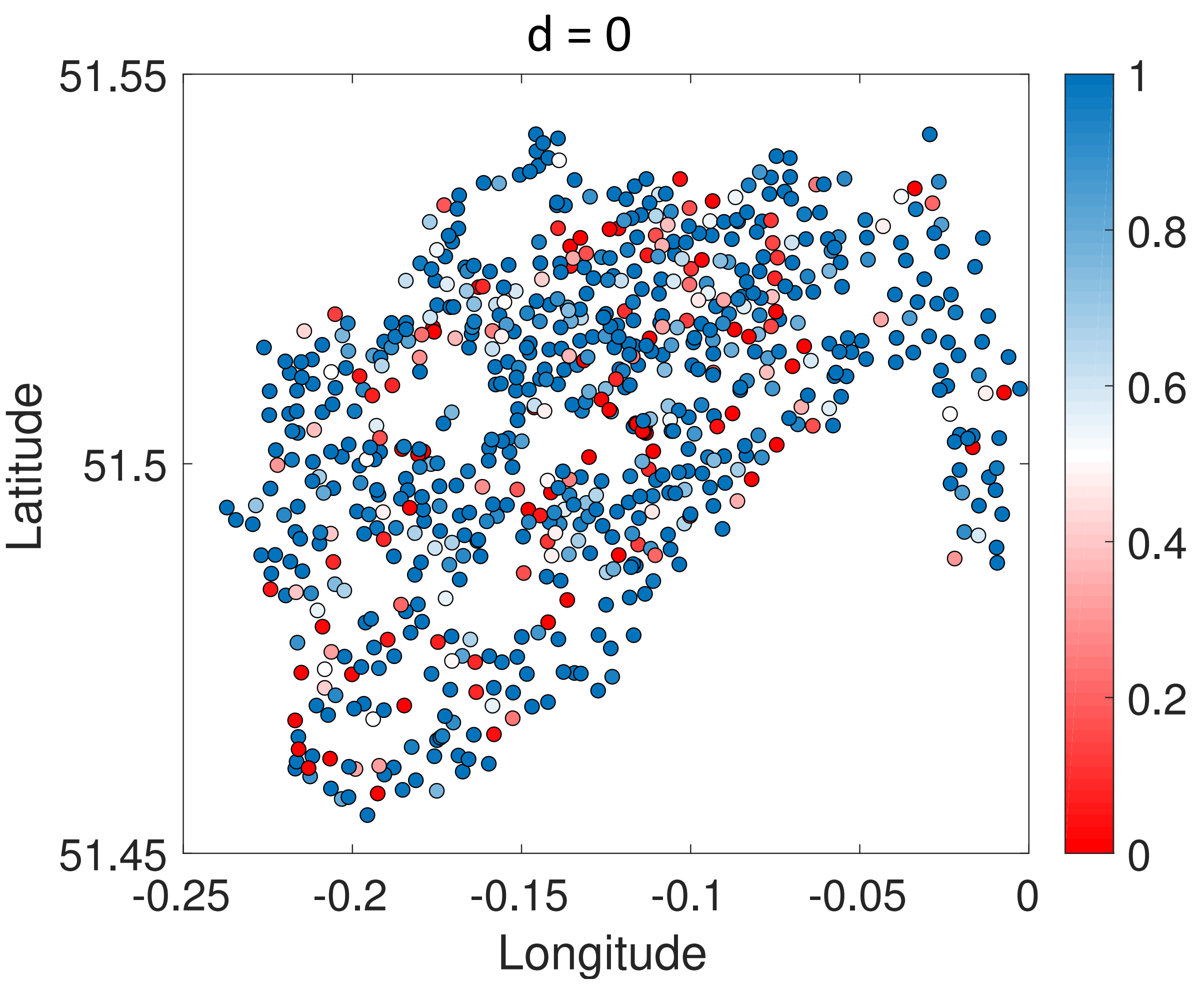
Spatio-Temporal Trajectory: $x: L \rightarrow \mathbb{T} \to \mathbb{Z}^2$ s.t. $x(i,t) = \left(B_i(t), S_i(t)\right)$

Space model

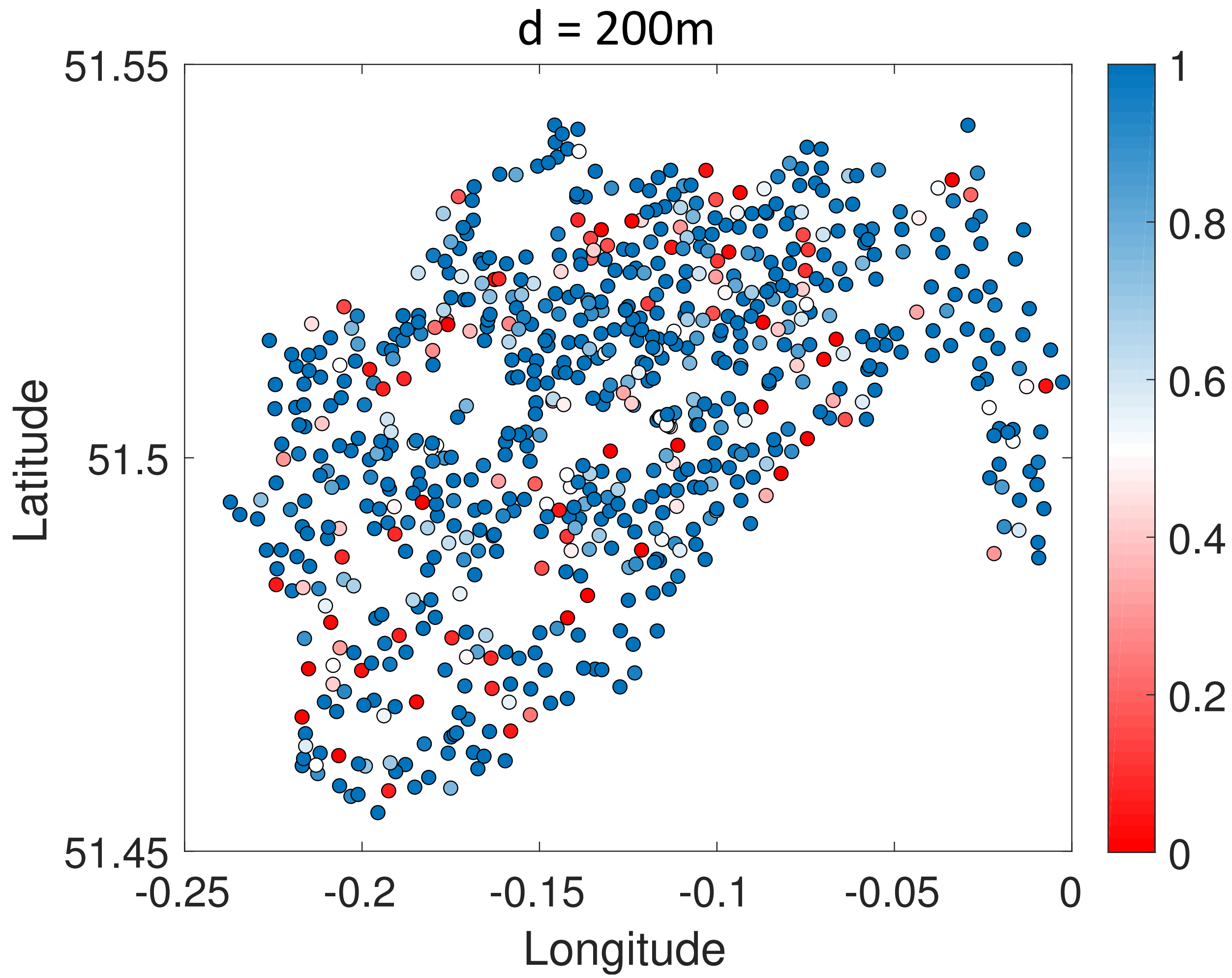
- Locations: $L = \{bike\ stations\}$,
- Edges: $\left(\ell_i, w, \ell_j\right) \in W$ iff $\mathrm{w} = \| \ell_i - \ell_j \| < 1\ kilometer$

Availability of Bikes

$$\phi_1 = \mathrm{G}\{\diamondsuit^{weight}_{[0,d]}(B > 0) \wedge \diamondsuit^{weight}_{[0,d]}(S > 0)\}$$



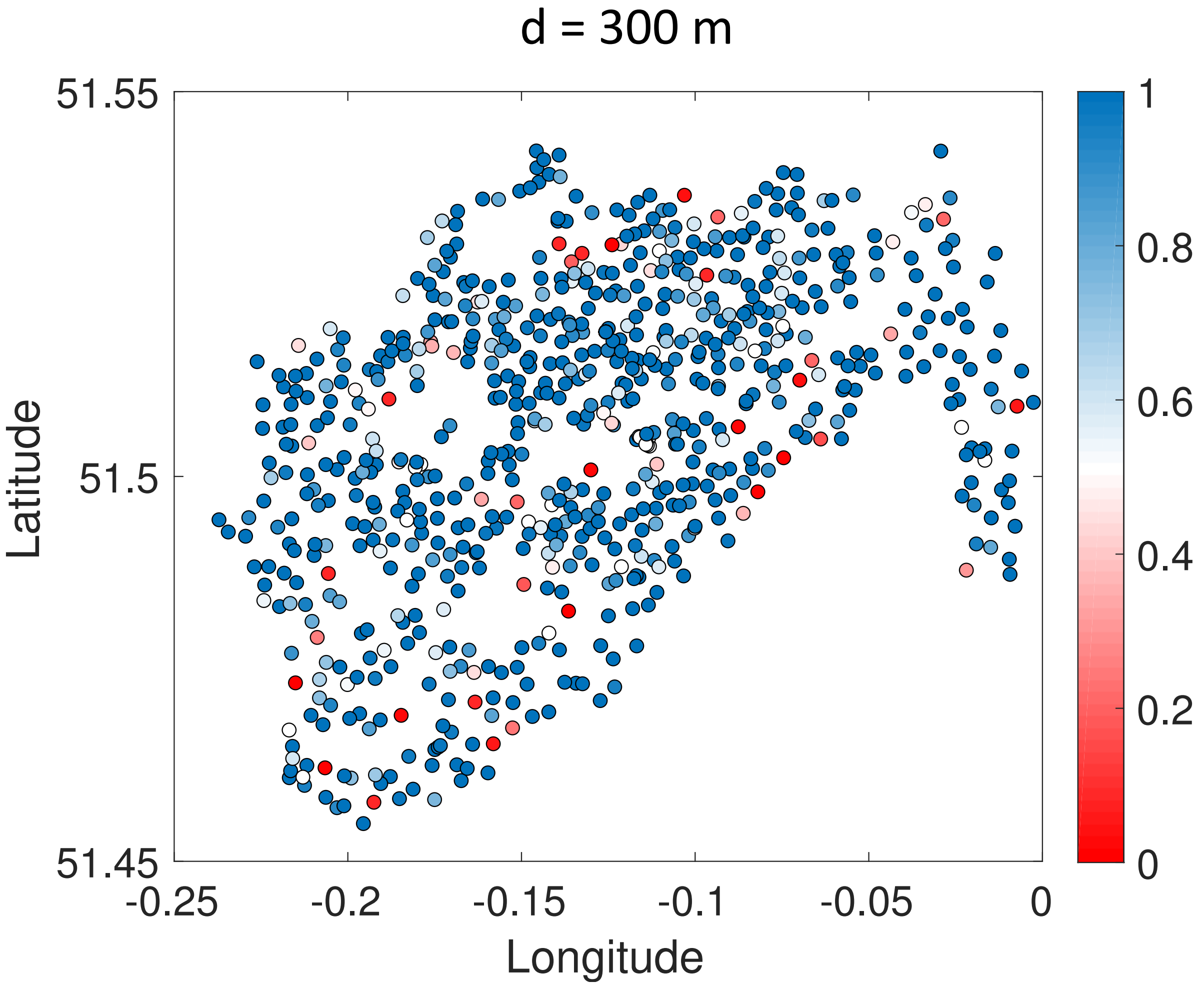
std in [0, 0.0158] , mean std = 0.0053.



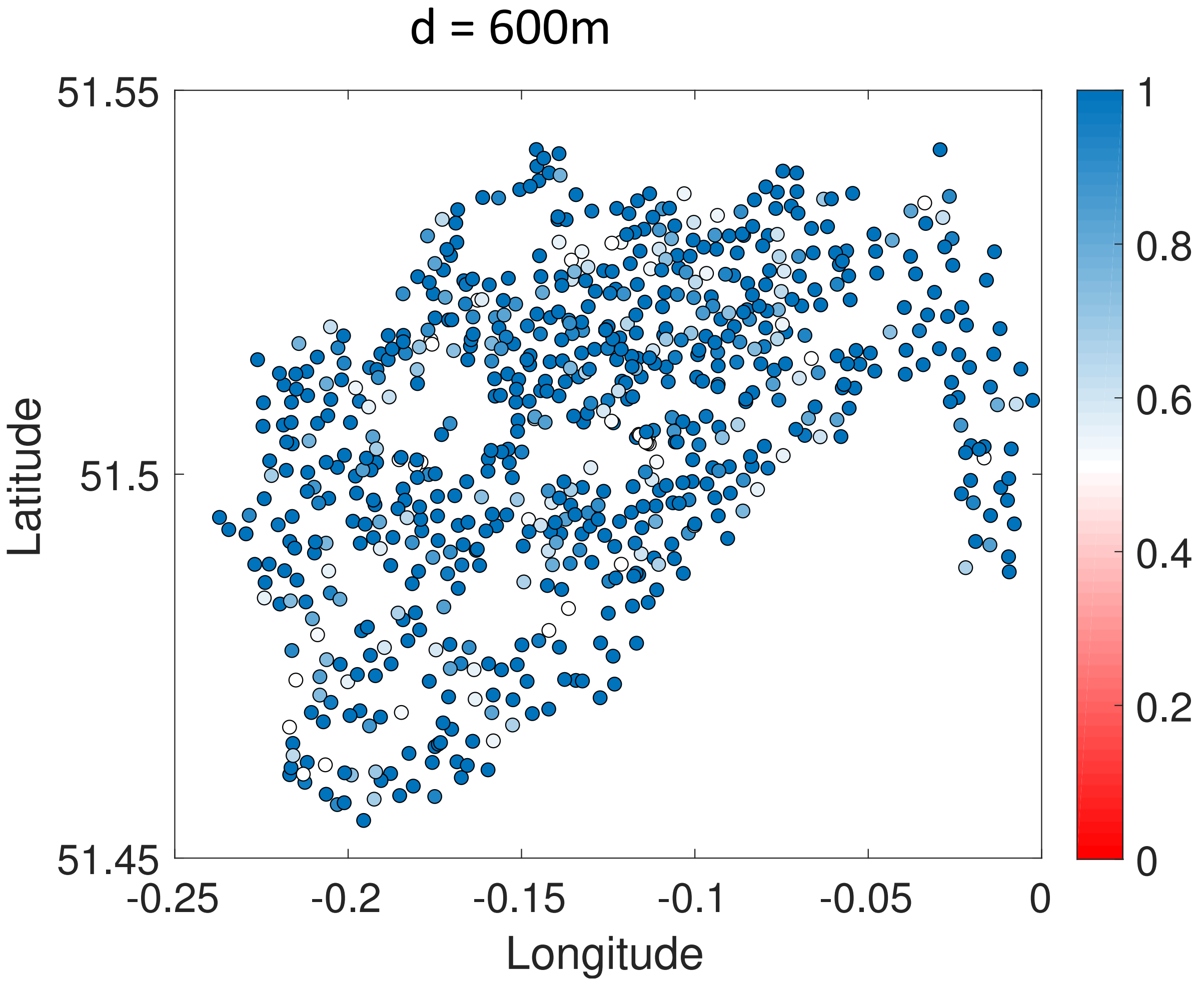
std in [0, 0.0158] , mean std = 0.0039.

Availability of Bikes

$$\phi_1 = \mathrm{G}\{\Diamond^{weight}_{[0,d]}(B > 0) \wedge \Diamond^{weight}_{[0,d]}(S > 0)\}$$



std in [0, 0.0151] , mean std = 0.0015.

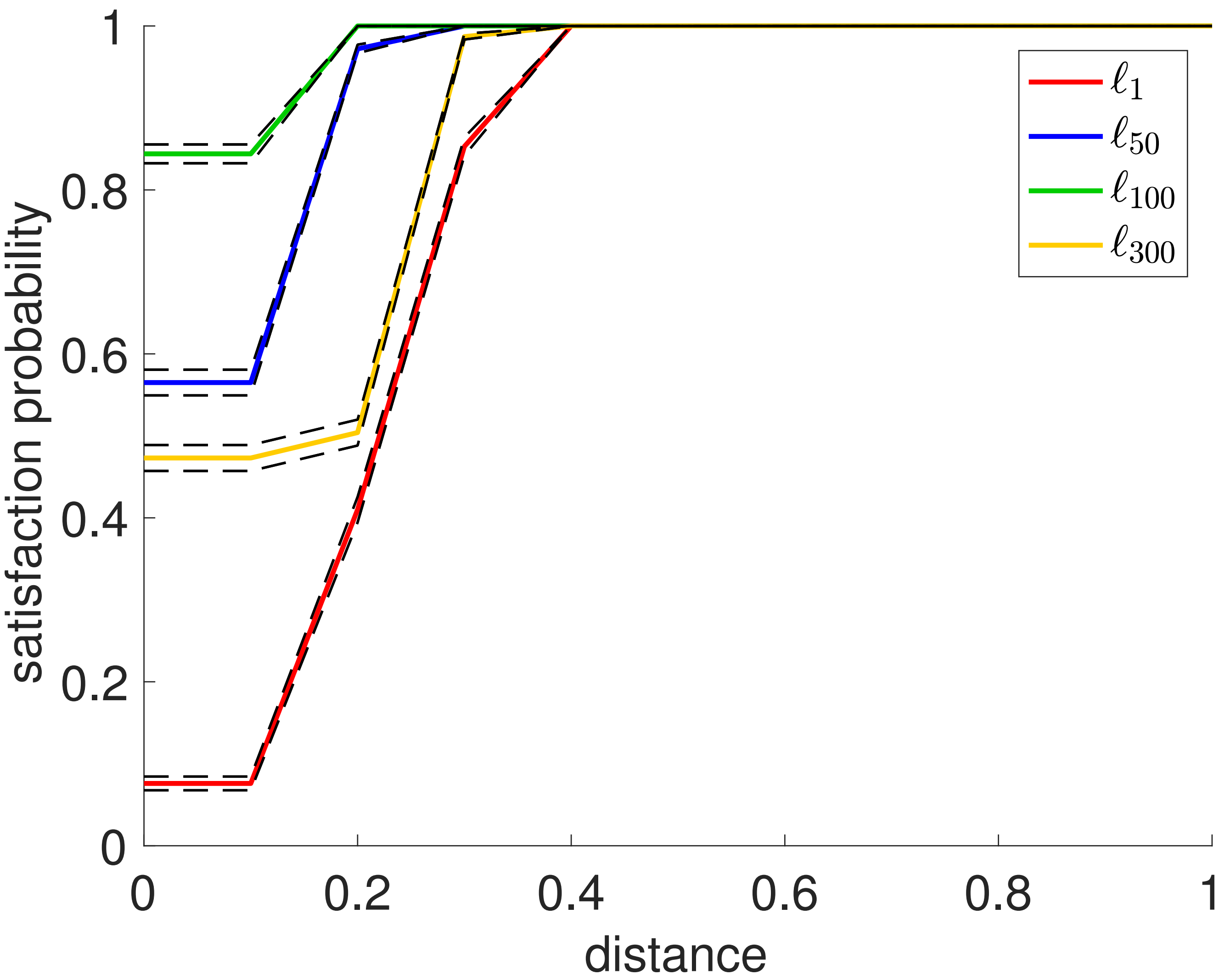


std in [0, 0.0142] , mean std = 0.0002.

Availability of Bikes

$$\phi_1 = G\{\diamondsuit^{weight}_{[0,d]}(B > 0) \wedge \diamondsuit^{weight}_{[0,d]}(S > 0)\}$$

Satisfaction probability of some BBS stations vs distance d=[0,1.0]



Bike Sharing Systems (BSS)

$$\psi_1 = \mathrm{G}\{\diamondsuit_{[0,d]}^{weight}(\mathrm{F}_{[t_w,t_w]}B > 0) \wedge \diamondsuit_{[0,d]}^{weight}(\mathrm{F}_{[t_w,t_w]}S > 0)\}$$

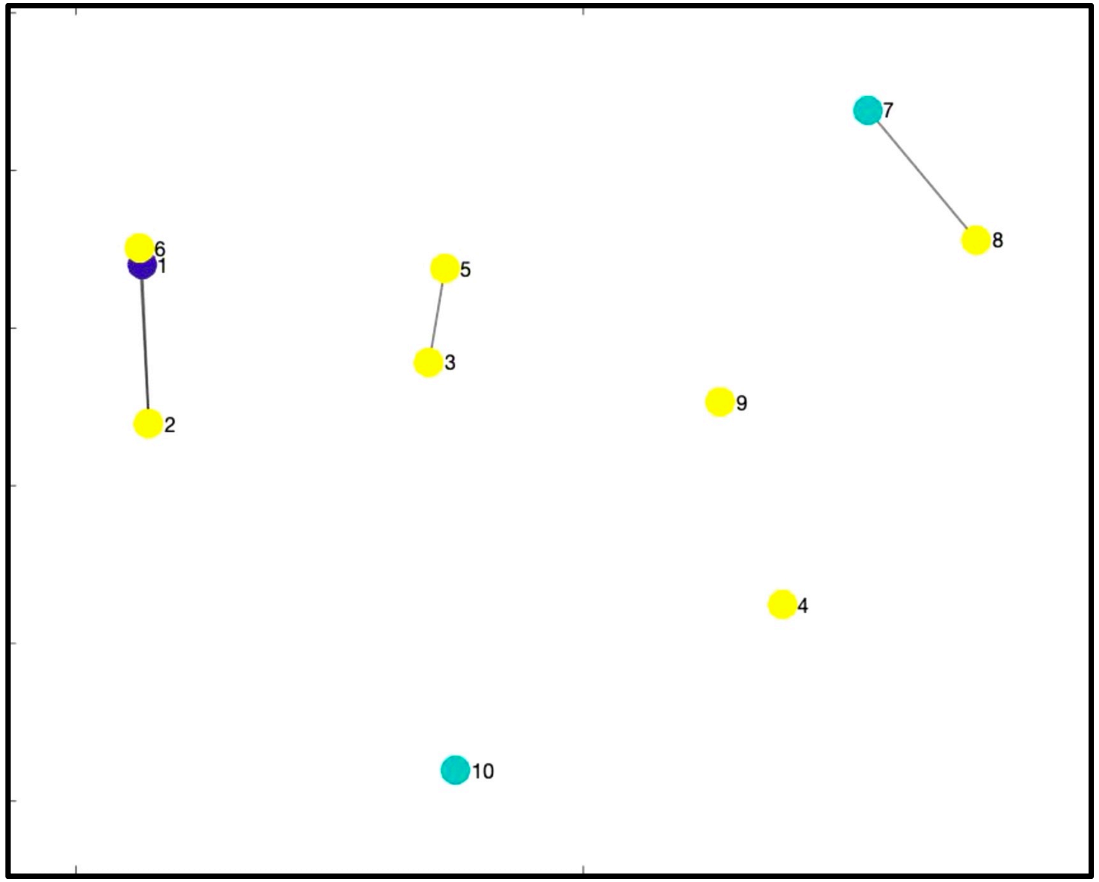
Average walking speed of 6.0 km/h, e.g. d = 0.5 km -> t_w = 6 minutes

The results similar to the results of previous property

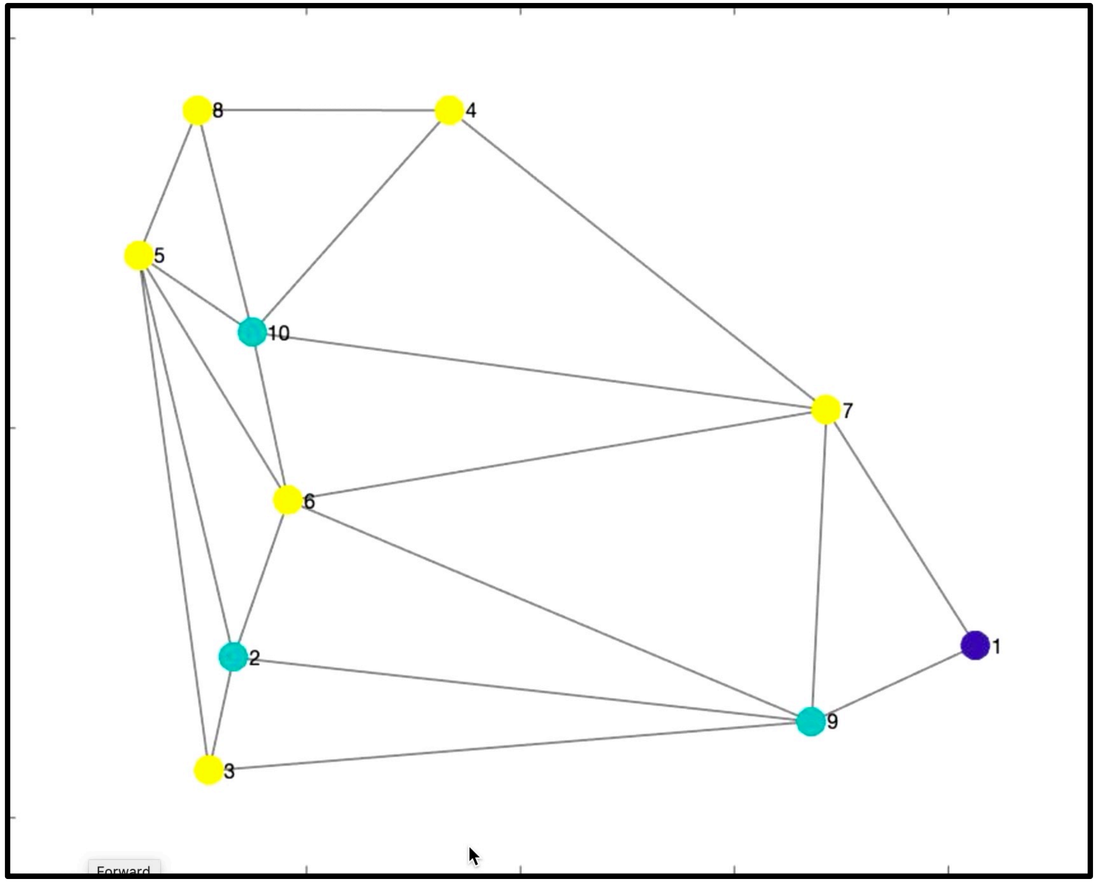
Dynamic Space

Mobile Ad-hoc sensor NETwork (MANET)

Coordinator Router End-devices



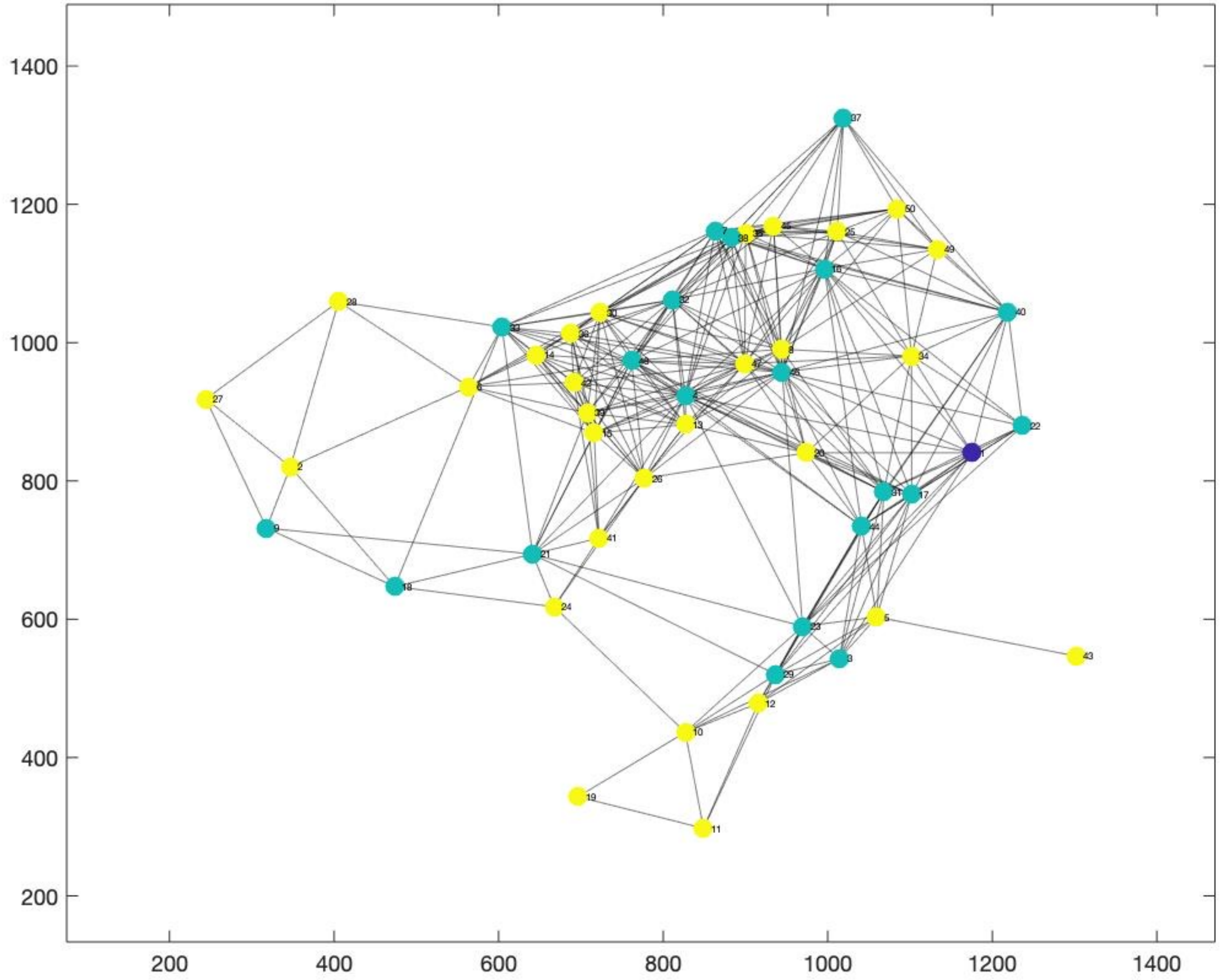
Connectivity Graph



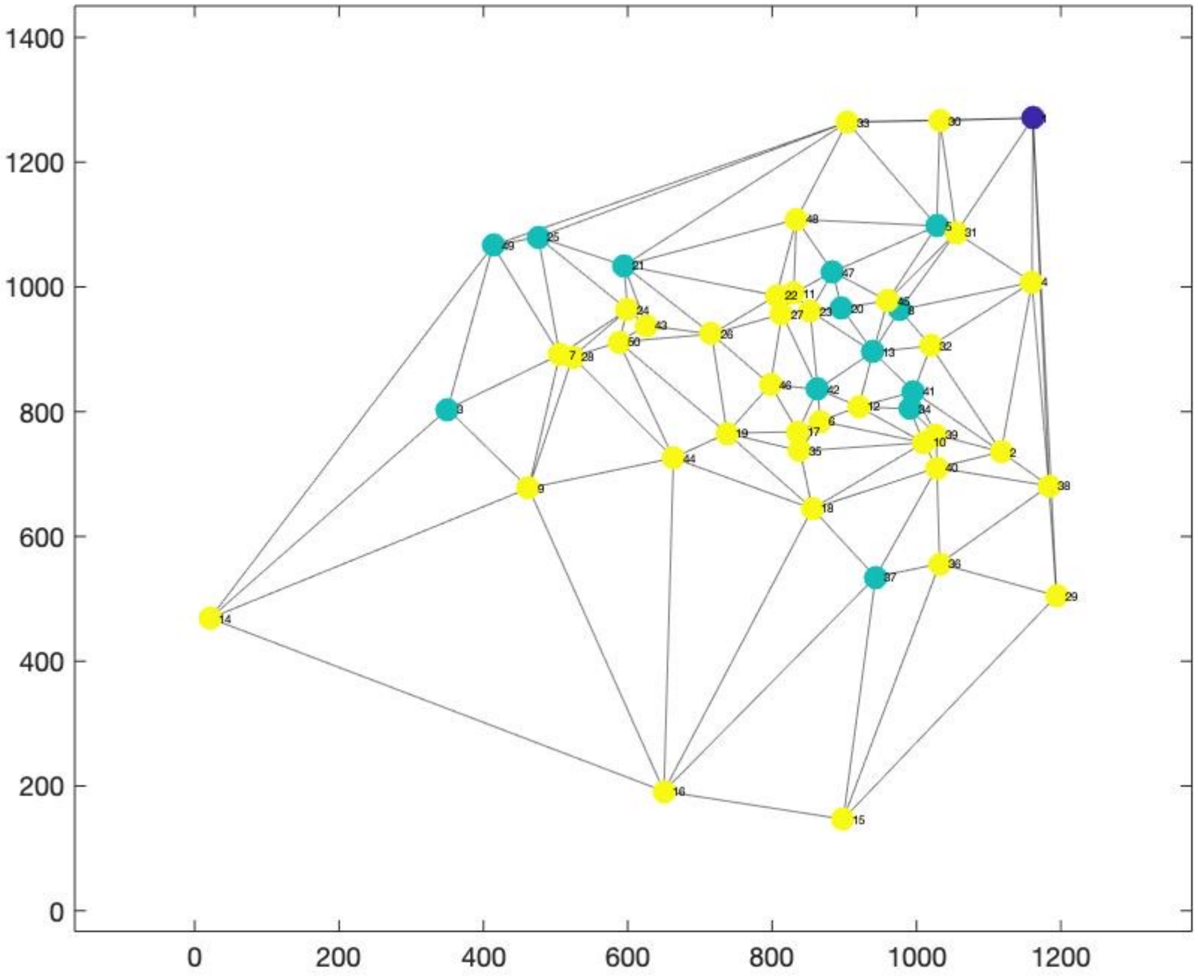
Proximity Graph

Mobile Ad-hoc sensor NETwork (MANET)

Coordinator Router End-devices



Connectivity Graph



Proximity Graph

Mobile Ad-hoc sensor NETwork (MANET)

Space model $S(t)$

- Locations: $L = \{devices\}$,
- Edges: $(\ell_i, w, \ell_j) \in W$ iff $\mathrm{w} = \| \ell_i - \ell_j \| < \min(r_i, r_j)$

Spatio-Temporal Trajectory: $x: L -> \mathbb{T} \to \mathbb{Z} \times \mathbb{R}^2$ s.t.

$x(i, t) = (nodeType, battery, temperature\ )$

$nodeType = 1, 2, 3$ for coordinator, rooter, and end_device

Connectivity in a MANET

"an end device is either connected to the coordinator or can reach it via a chain of at most of 5 routers"

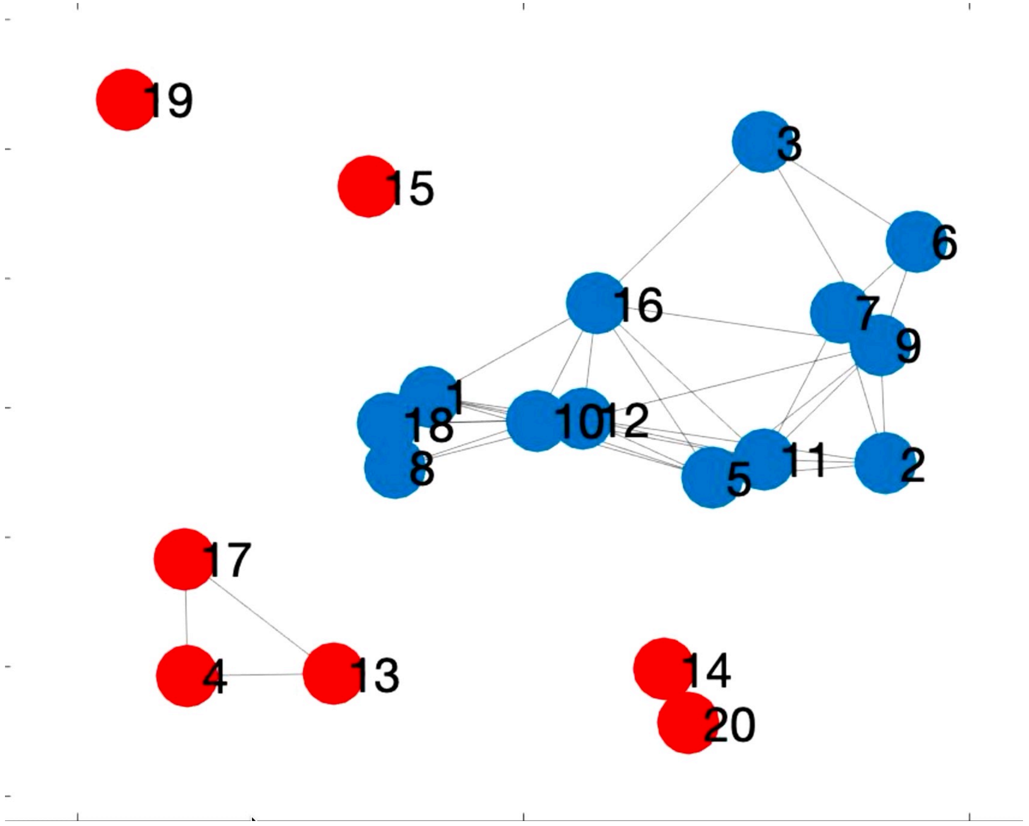
$$\phi_{connect} = device\mathcal{R}^{hops}_{\leq 1}(router\mathcal{R}^{hops}_{\leq 5} coord)$$

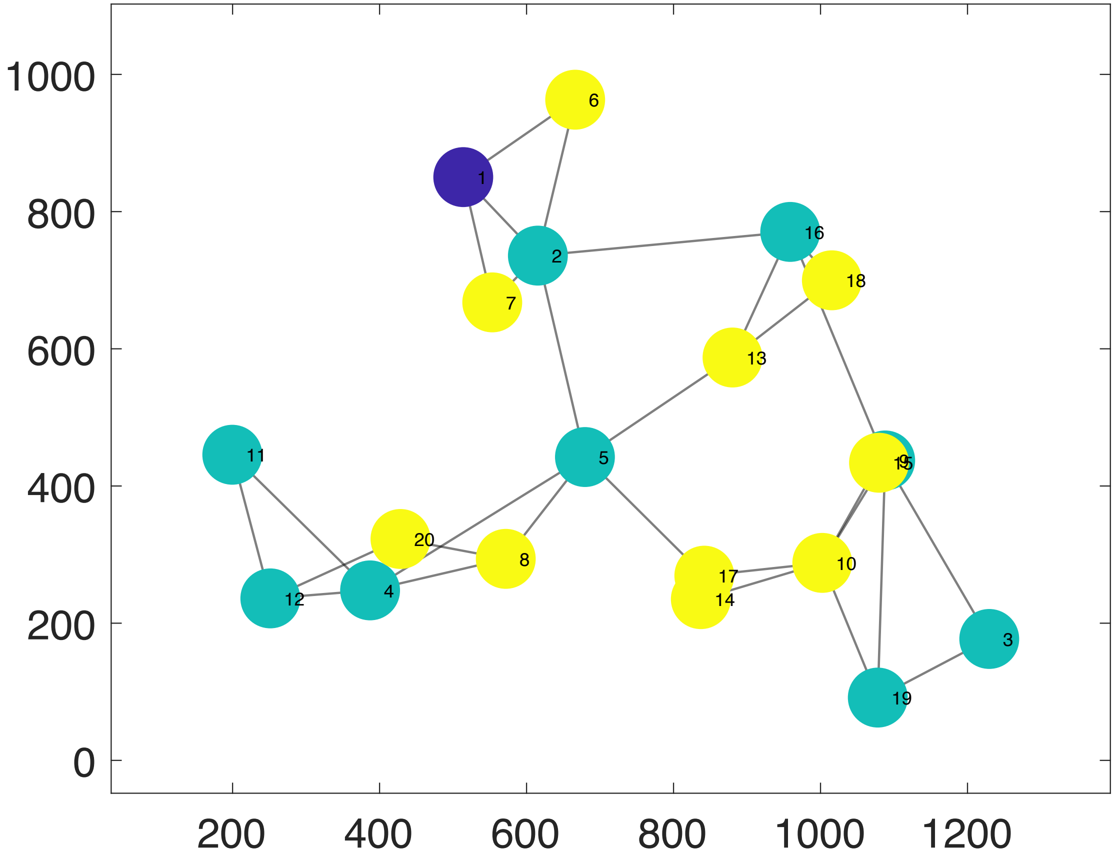
"broken connection is restored within h time units"

$$\phi_{connect\_restore} = \mathrm{G}(\neg\phi_{connect} \rightarrow \mathrm{F}_{[0,h]}\phi_{connect})$$

Boolean Satisfaction at each time step

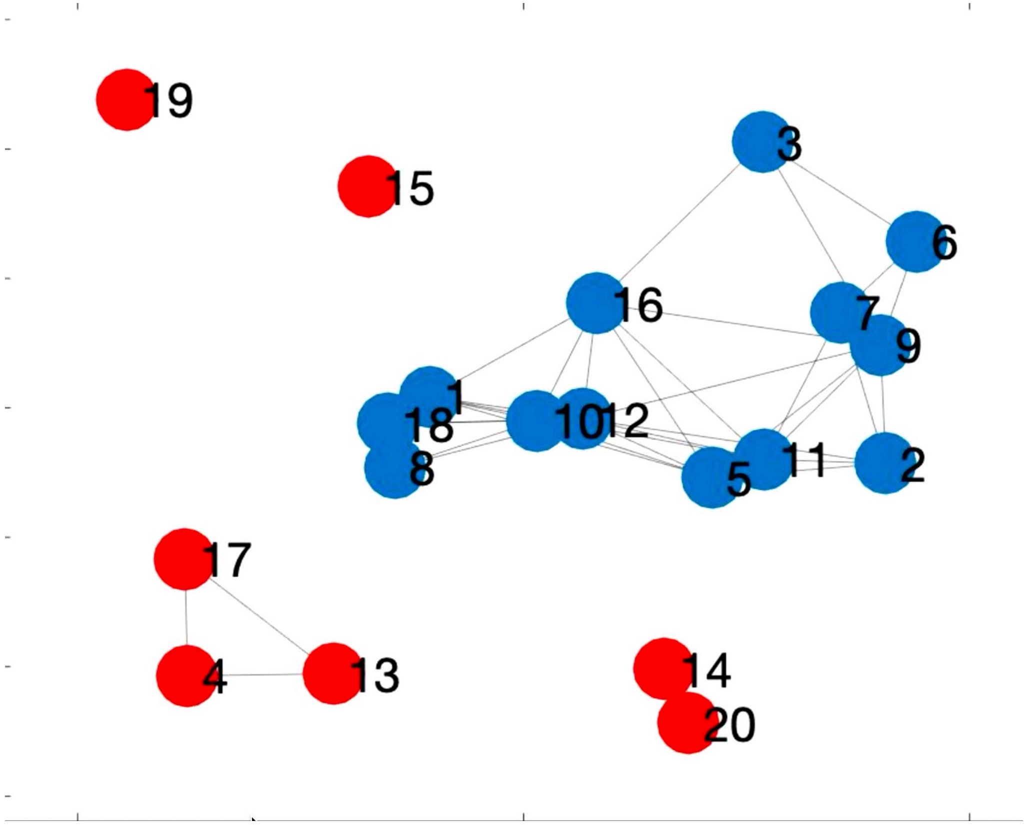
$$\phi_{connect} = device\mathcal{R}^{hops}_{\leq 1}(router\mathcal{R}^{hops}_{<\infty} coord)$$

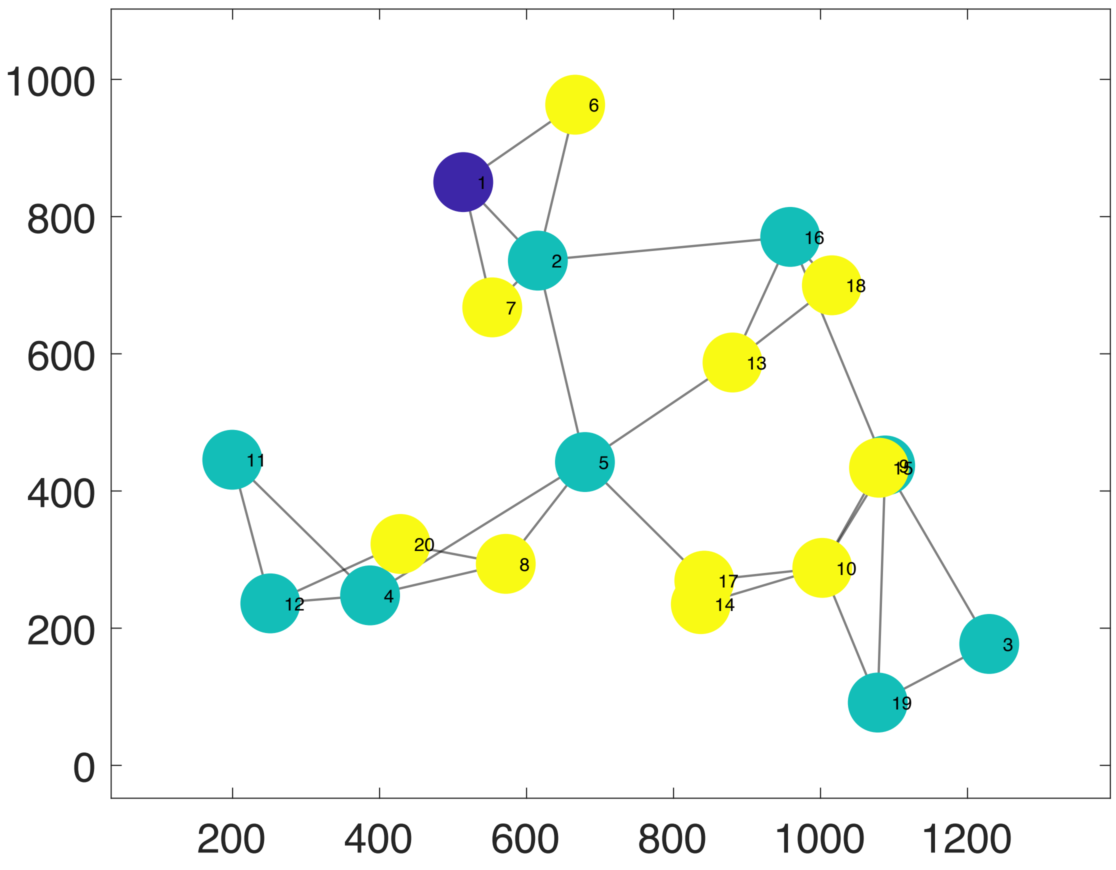




Boolean Satisfaction at each time step

$$\phi_{connect} = device\mathcal{R}^{hops}_{\leq 1}(router\mathcal{R}^{hops}_{<\infty} coord)$$





Delivery in a MANET

“from a given location, we can find a path of (hops) length at least 5 such that all nodes along the path have a battery level greater than 0.5”

$$\psi_3 = \mathcal{E}^{hops}_{[5,\infty]}(battery > 0.5)$$

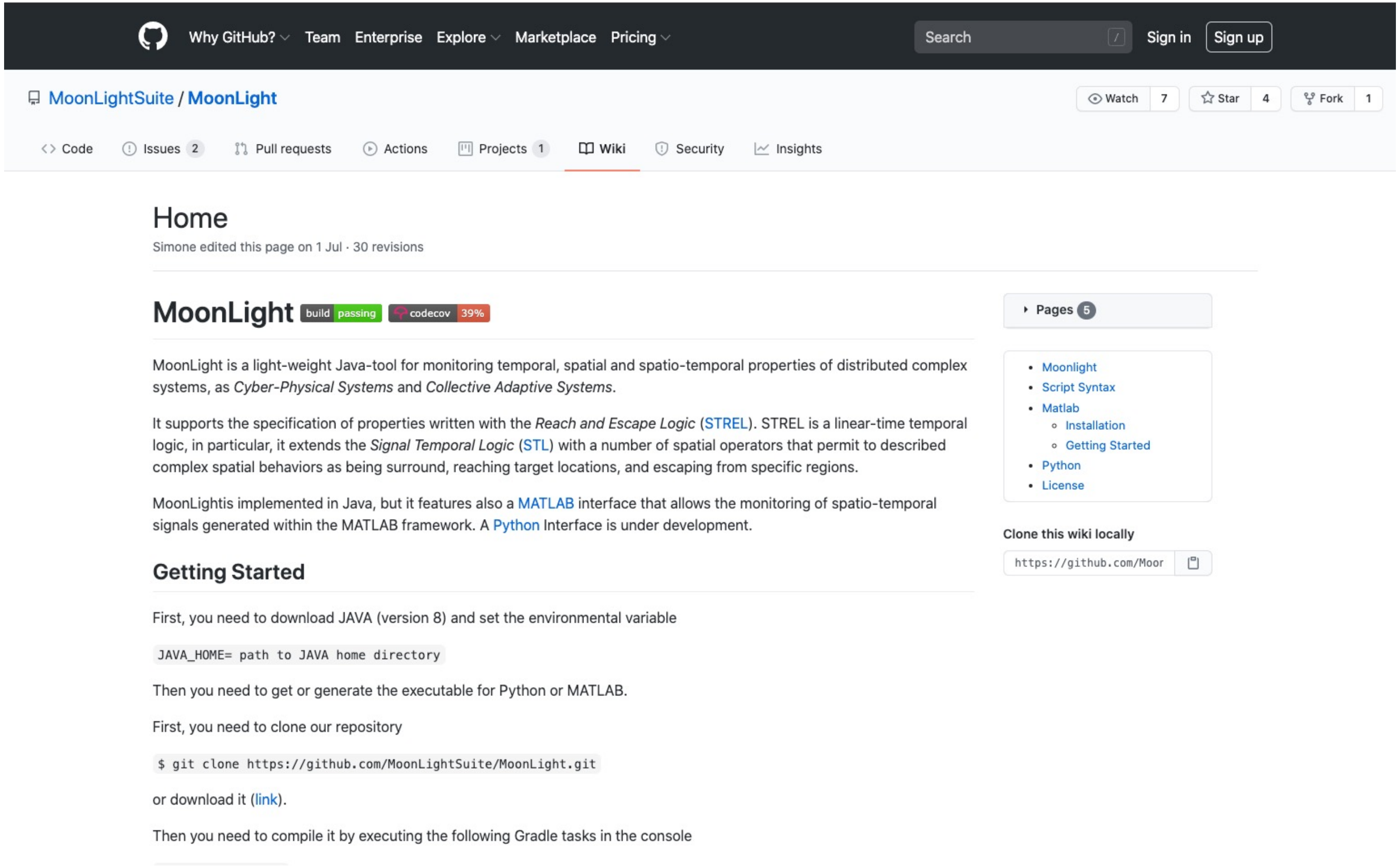
Reliability in a MANET

“reliability in terms of battery levels, e.g. battery level above 0.5

$$\phi_{reliable\_router} = ((battery > 0.5) \wedge router)\mathcal{R}^{hops}_{<\infty} coord$$

$$\phi_{reliable\_connect} = device\mathcal{R}^{hops}_{\leq 1}(\phi_{reliable\_router})$$

Moonlight: https://github.com/MoonLightSuite/MoonLight/wiki



Home

Simone edited this page on 1 Jul · 30 revisions

MoonLight build passing codecov 39%

MoonLight is a light-weight Java-tool for monitoring temporal, spatial and spatio-temporal properties of distributed complex systems, as *Cyber-Physical Systems* and *Collective Adaptive Systems*.

It supports the specification of properties written with the *Reach and Escape Logic* (STREL). STREL is a linear-time temporal logic, in particular, it extends the *Signal Temporal Logic* (STL) with a number of spatial operators that permit to described complex spatial behaviors as being surround, reaching target locations, and escaping from specific regions.

MoonLightis implemented in Java, but it features also a MATLAB interface that allows the monitoring of spatio-temporal signals generated within the MATLAB framework. A Python Interface is under development.

Getting Started

First, you need to download JAVA (version 8) and set the environmental variable

```
JAVA_HOME= path to JAVA home directory
```

Then you need to get or generate the executable for Python or MATLAB.

First, you need to clone our repository

```
$ git clone https://github.com/MoonLightSuite/MoonLight.git
```

or download it (link).

Then you need to compile it by executing the following Gradle tasks in the console

Clone this wiki locally

```
https://github.com/Moor
```

```
(atomicExpression)
| ! Formula
| Formula & Formula
| Formula | Formula
| Formula => Formula
| Formula until [a b] Formula
| Formula since [a b] Formula
| eventually [a b] Formula
| globally [a b] Formula
| once [a b] Formula
| historically [a b] Formula
| escape(distanceExpression)[a b] Formula
| Formula reach (distanceExpression)[a b] Formula
| somewhere(distanceExpression) [a b] Formula
| everywhere (distanceExpression) [a b] Formula
| {Formula}
```