Space Relationship Modeler (SPAREMO)

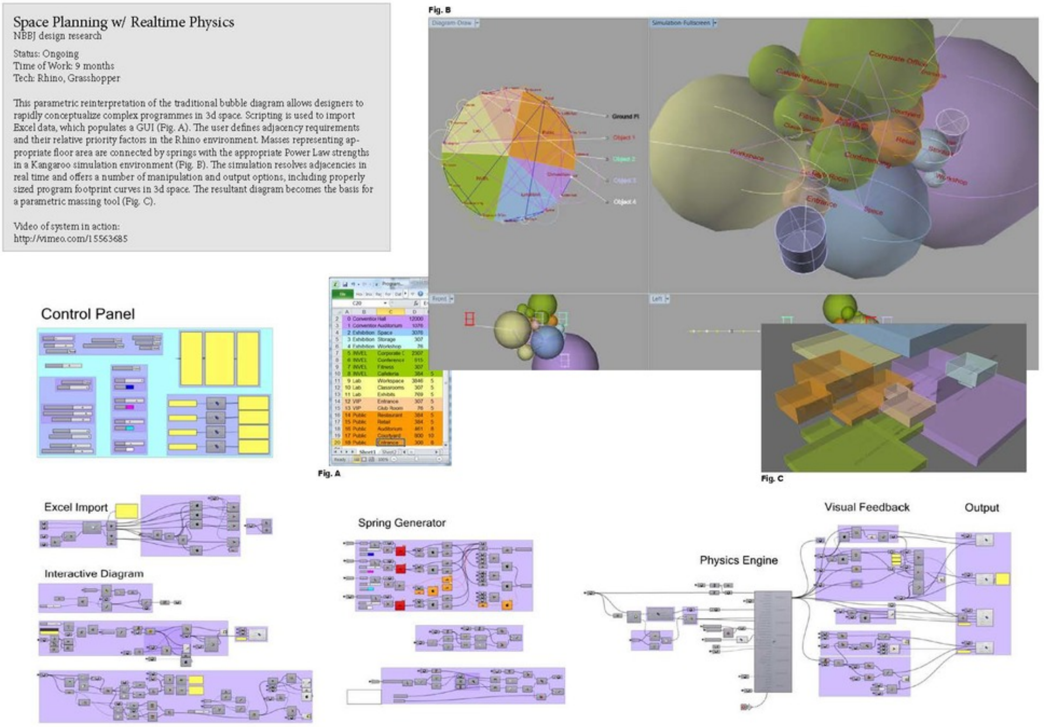
Marco Juliani
Morphogenetic Programming
ucbqjul@ucl.ac.uk

Overall Objectives

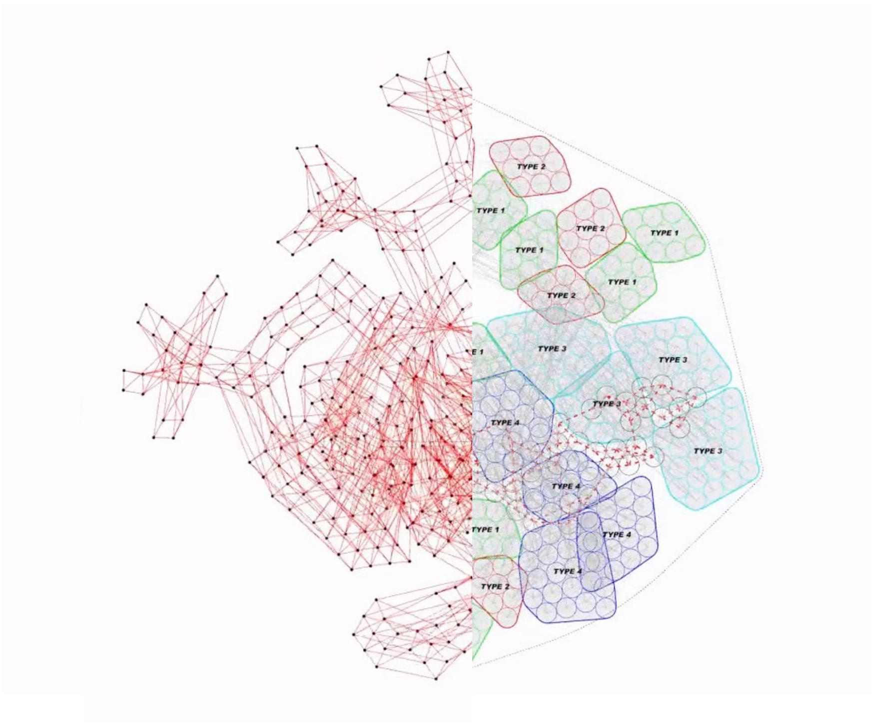
- ‘Emulate physics’ category
- Generative framework for deriving early space-planning solutions.
- A playful framework to visualize and rethink programmatic and spatial relationships.
- ‘Relationship maps’ produced by using relationship matrices combined with spring systems.
- Multiple ways of manipulating ‘hyper-parameters’ as representation options to guide the simulation to arrive at helpful solutions.

Review

Space Planning w/ Real Time Physics



NBBJ – Physics-based space planning
http://www.marcsyp.com/space-planning

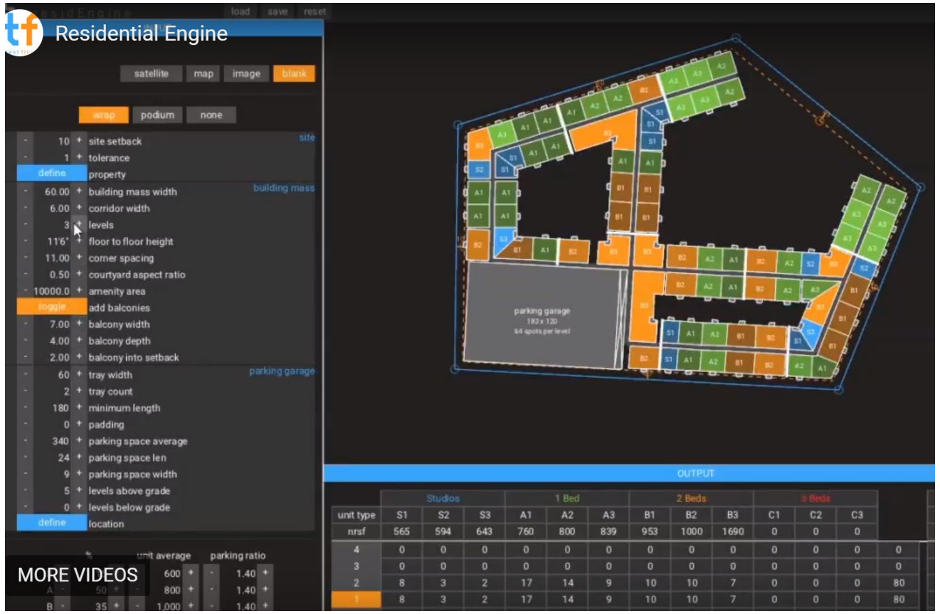


Yazdani Studio – Parametric Plan Tool
https://vimeo.com/146462913

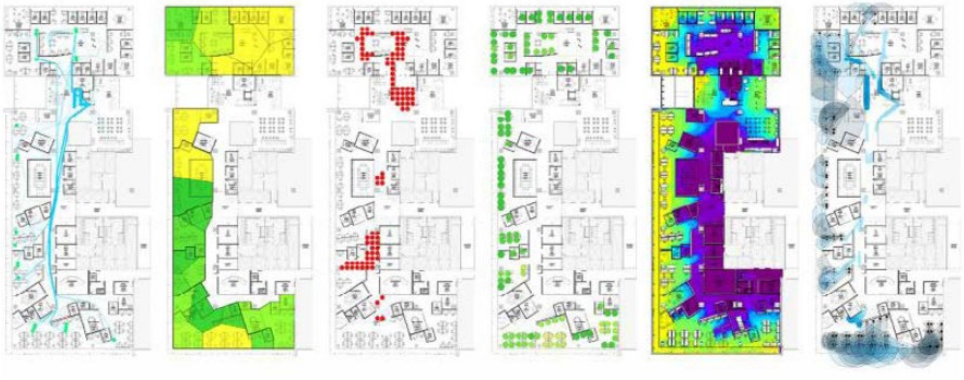
ucbqjul@ucl.ac.uk

Review

Plan Optimization



TestFitIO– Parametric Residential Planner
https://testfit.io/

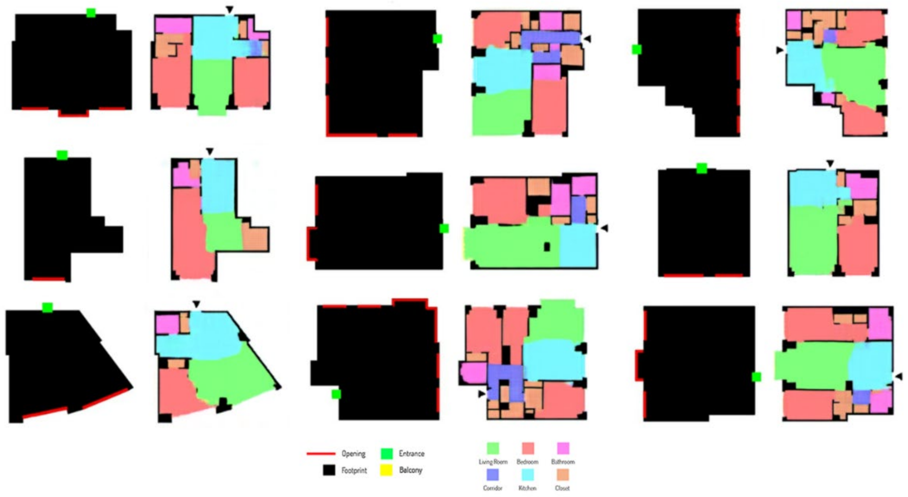


The Living – Generative Space Planning
https://www.autodeskresearch.com/sites/default/files/Project-Discover-Generative-Design.pdf

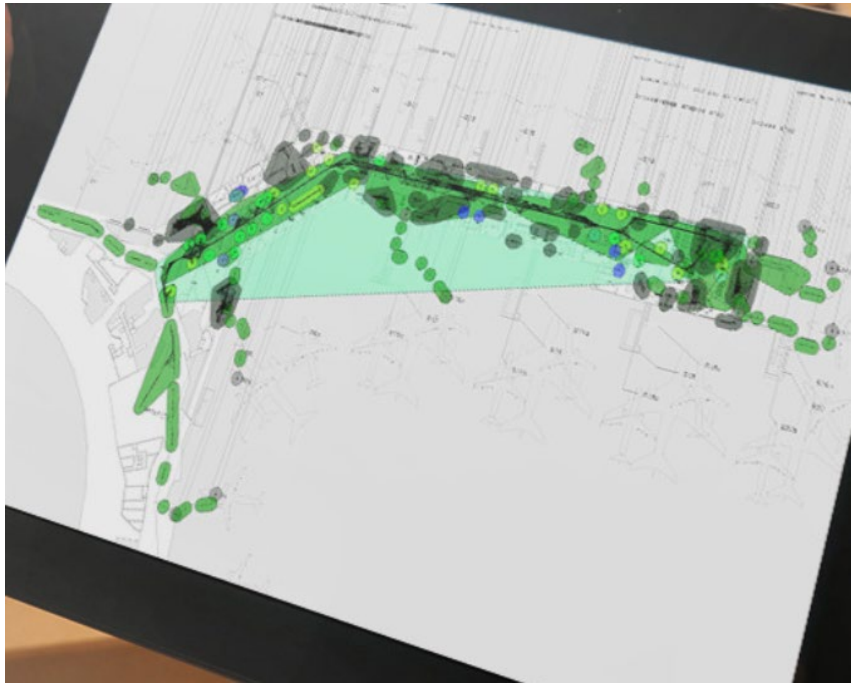
ucbqjul@ucl.ac.uk

Review

Msc Algorithms



Stanislas Chaillou – GAN Generated Plans
https://towardsdatascience.com/ai-architecture-f9d78c6958e0



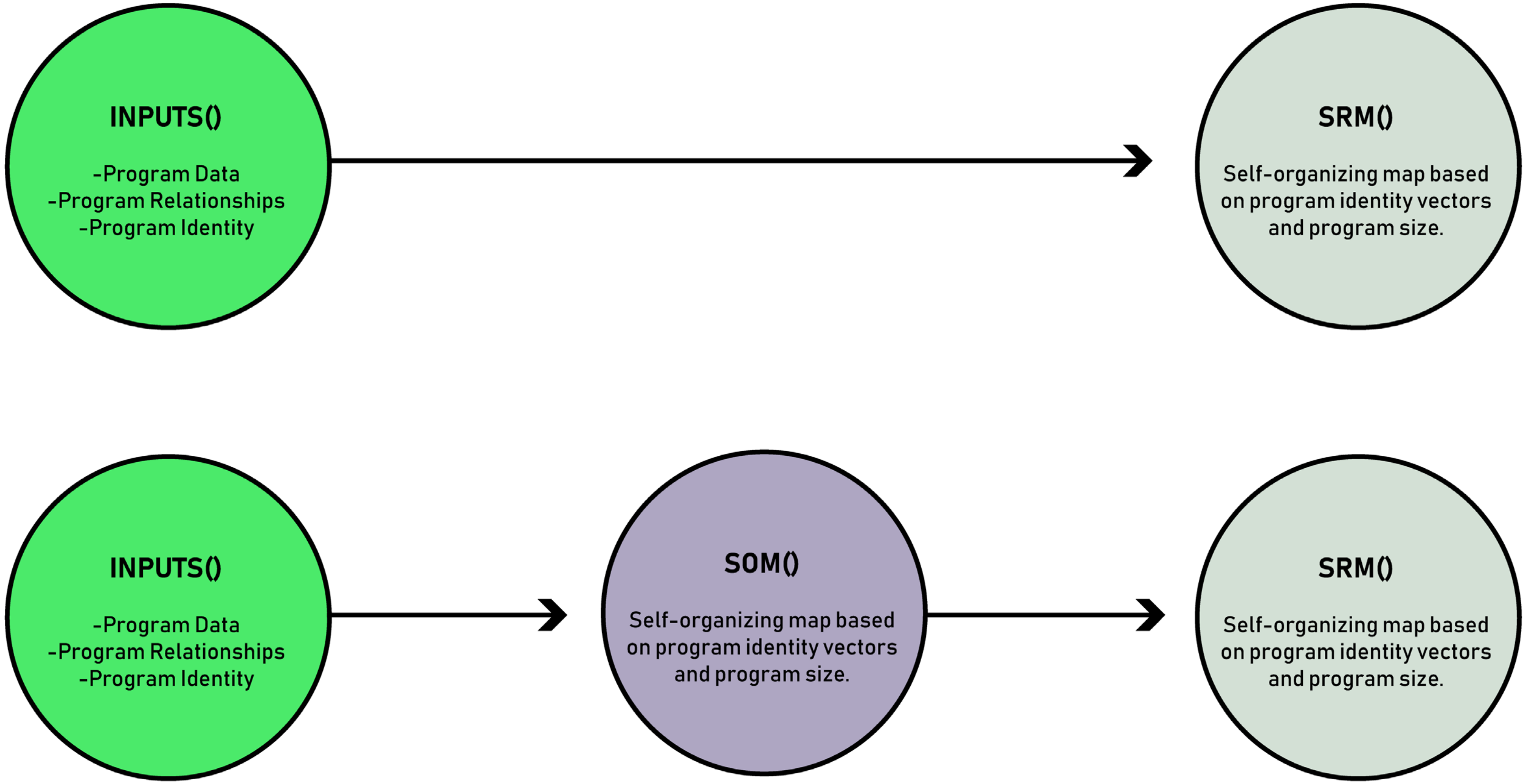
Woods Bagot- Program Relationship Tool

ucbqjul@ucl.ac.uk

Workflow Overall

Method 1: Global_Program

Method 2: Kohonen_Program



-Parsed .csv matrices that describe program size/ relationship/ identitiy

-Program broken down to bite size modules.
-Program starting point is laid out according to how closely it resmebles other program, on a gradient.

-Springs corresponding to all program pairs are brought to equilibrium while allowing for hyperparameter manipulation and multiple levels of representation.

ucbqjul@ucl.ac.uk

Workflow Inputs

Program/ Areas

quickAreas

File Edit View Insert Format D

Program	Area
Open Office	10000
Storage	800
Kitchen	4000
Meeting	2000
Reception	1000
Private Working	2000
Fire-Safe Circula	3800

Program Relationships

quickSprings Test

File Edit View Insert Format Data To

Program 1	Strength to	Program 2
Open	1	Storage
Open	5	Kitchen
Open	8	Meeting
Open	3	Reception
Open	7	Private
Open	6	Circ
Storage	8	Kitchen
Storage	4	Meeting
Storage	8	Reception
Storage	2	Private
Storage	3	Circ
Kitchen	6	Meeting
Kitchen	7	Reception
Kitchen	1	Private
Kitchen	8	Circ
Meeting	3	Reception
Meeting	2	Private
Meeting	4	Circ

Program Identity Vectors

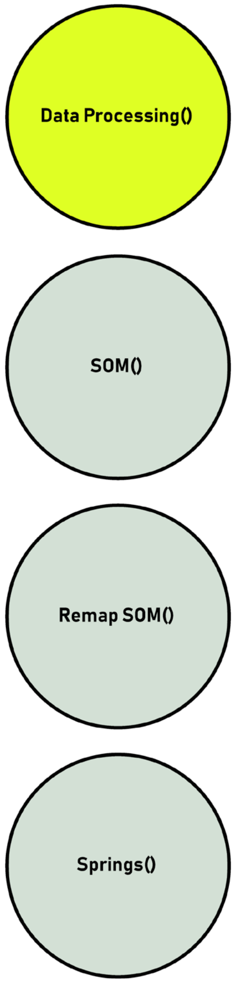
Kohonen Net Sample

File Edit View Insert Format Data Tools Add-o

A	B	C
Open Office	null	0
Natural Light	0.4	1
Noise Allowed	0.2	2
Amenity Access	0.85	3
Openness	0.8	4
Pure function	0.7	5
Storage	null	6
Natural Light	0	7
Noise Allowed	1	8
Amenity Access	0.4	9
Openness	0	10
Pure function	1	11
Kitchen	null	12
Natural Light	0.7	13
Noise Allowed	1	14
Amenity Access	0.9	15
Openness	0.9	16
Pure function	0.5	17
Meeting	null	18
Natural Light	0.15	19
Noise Allowed	0.6	20
Amenity Access	0.14	21
Openness	0.12	22

Marco Juliani ucbqjul@ucl.ac.uk

Pseudocode$_1$



Data Processing()

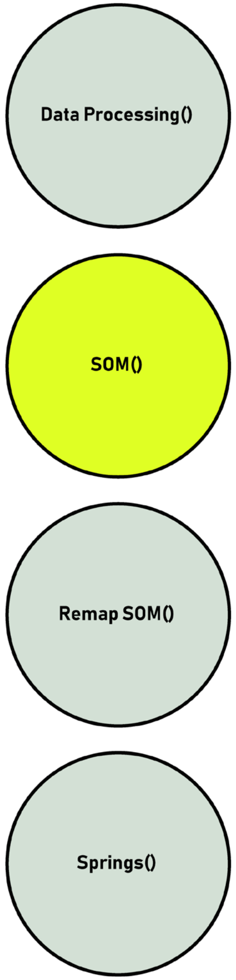
Feed Program Names and Areas();

Feed Program Numerical Interrelationships();

Feed Program Identity Vectors();

ucbqjul@ucl.ac.uk

Pseudocode$_2$



SOM()

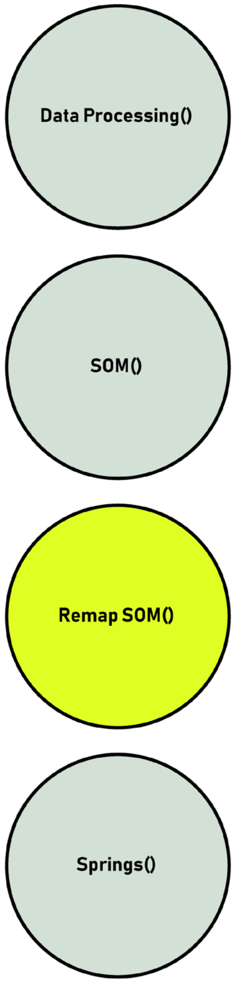
//Vanilla SOM adapted from Morphogenetic class

//Allows for a flexible matrix of variable subdivisions keeping with the total plan area required.

Marco Juliani

ucbqjul@ucl.ac.uk

Pseudocode$_3$



SOM Remap()

For each program

-> take an average of identity vector as summary vector

Find range of all summary vectors()

Divide range space into # intervals corresponding to number of program types();

Foreach particle

-> Find correct interval and deviation from the interval midpoint (discretize)

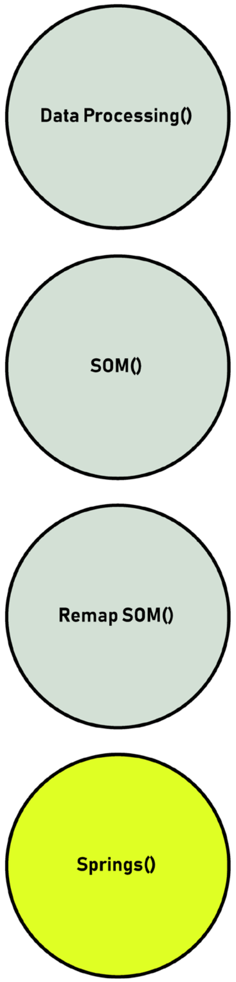
-> Store this multiplier value (instance) to be multiplied with a corresponding springConstant (type) to make up a weighted ‘springConstant’ for any given pair.

-> This performs a sort by how close each particle is from its corresponding BMU. This sort is visible in the SOM preview as a suffix of each particle’s name . This gets carried forward to the spring system.

Marco Juliani

ucbqjul@ucl.ac.uk

Pseudocode$_4$



Spring System()

Ref Dictionary - [Dictionary containing all program pair combinations (type) and corresponding springConstant for each]

Construct all participating particles (instances) using their ranked name, their instance multiplier (how close they were to BMU), their starting location()

Generate a spring-chain of aforementioned particles (instances)

Iterate through ‘ChainTension’ which is a double for loop which iterates through all pairs:

-> Take particle-pair instance name and strip it of numbers/symbols to determine the ‘type’ pair’. ie. *Office05_ Storage11* would become *Office_Storage*. If the type pair is accepted by the dictionary, then the type springConstant is multiplied by the instance springConstant. The radii of each instance pair is also taken into for restLength purposes.

//When ‘attraction equilibrium’ is reached, enable ‘fitting’ mode:

//Double for loop (similar to ChainReaction())to iterate through all pairs:

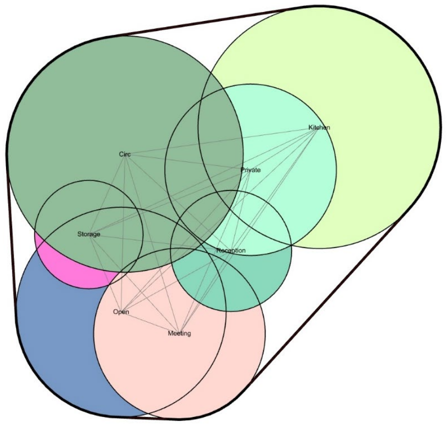
Find the restLength delta and vector between pairs to derive a new motion vector which in turn spurs a negative and equal motion vector on the other particle of the pair. Tally the number of interactions each particle has had with other particles.

Loop through the chain and add a new derived (average) position from above double loop, and divide by the number of interactions. This yields a motion that ends up separating all the particles while maintaining them at their desired restLength.

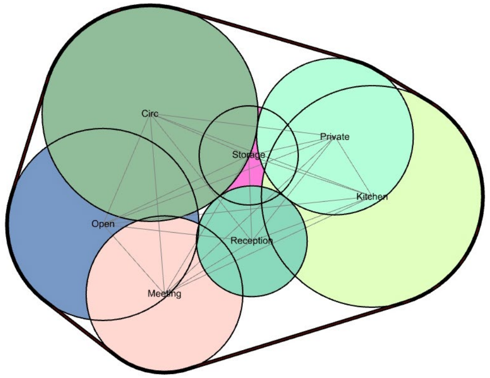
Marco Juliani

ucbqjul@ucl.ac.uk

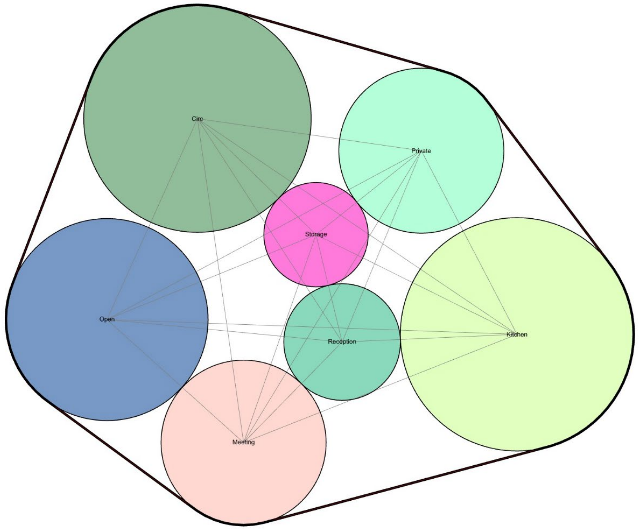
Workflow *Method 1 – Global_Program*



Random initial state

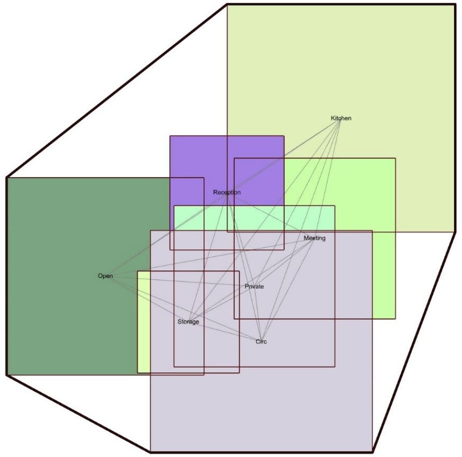


Attraction ‘ equilibrium’

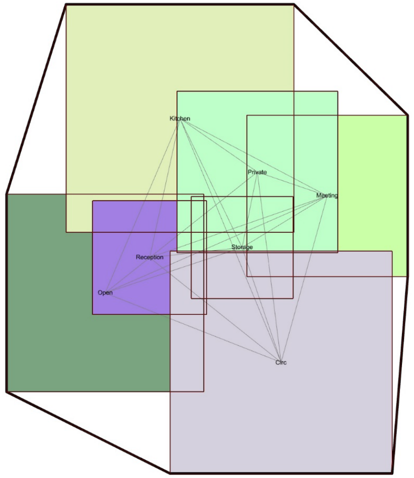


Repulsion ‘equilibrium

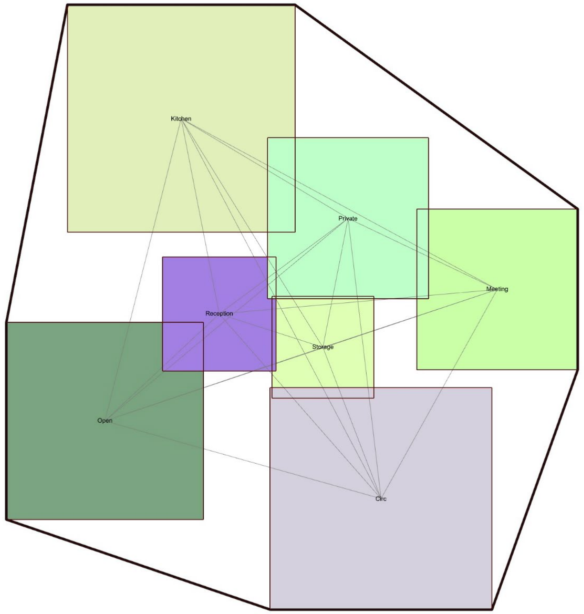
Workflow *Method 1 - Global_Program*



Random initial state

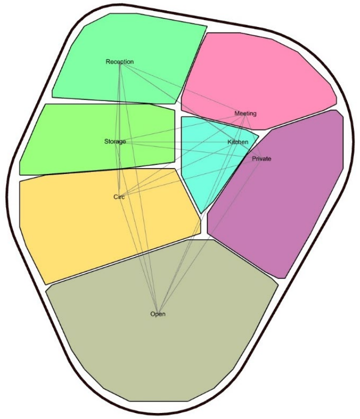


Attraction ' equilibrium'



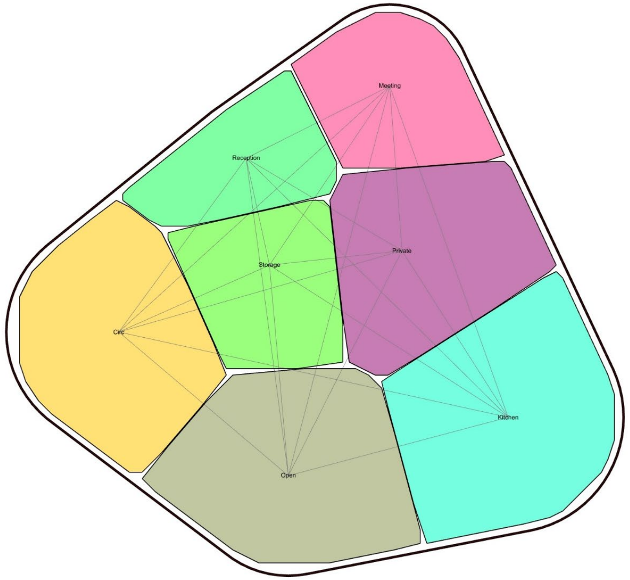
Repulsion 'equilibrium

Workflow *Method 1 - Global_Program*



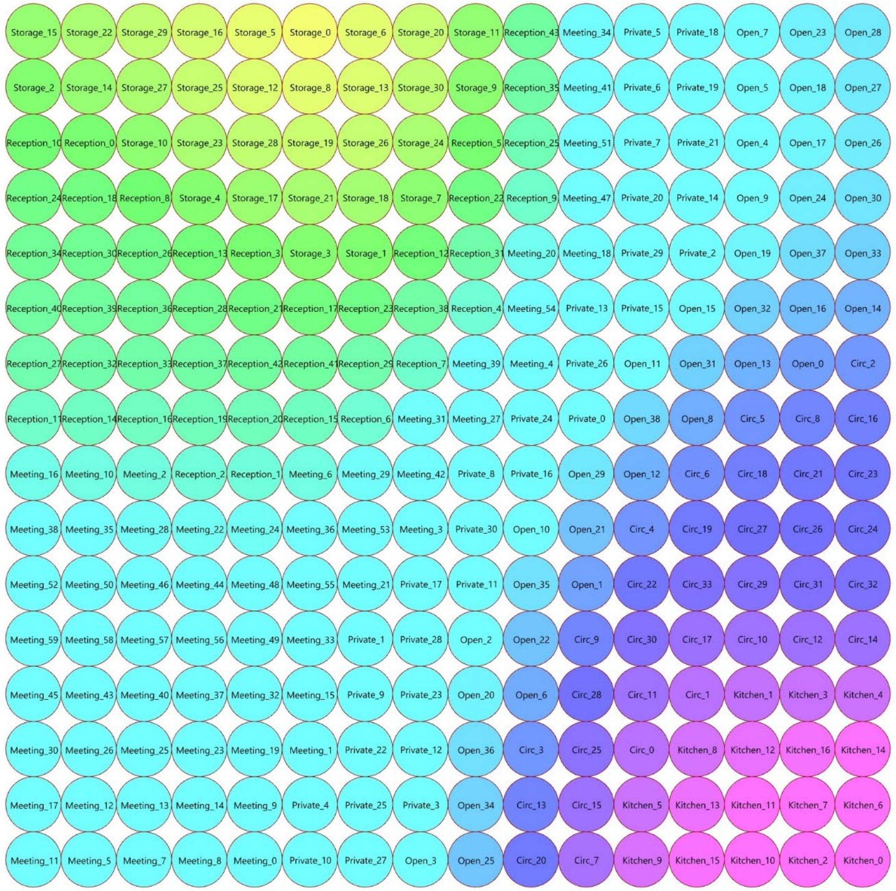
Random initial state

Attraction ‘ equilibrium’

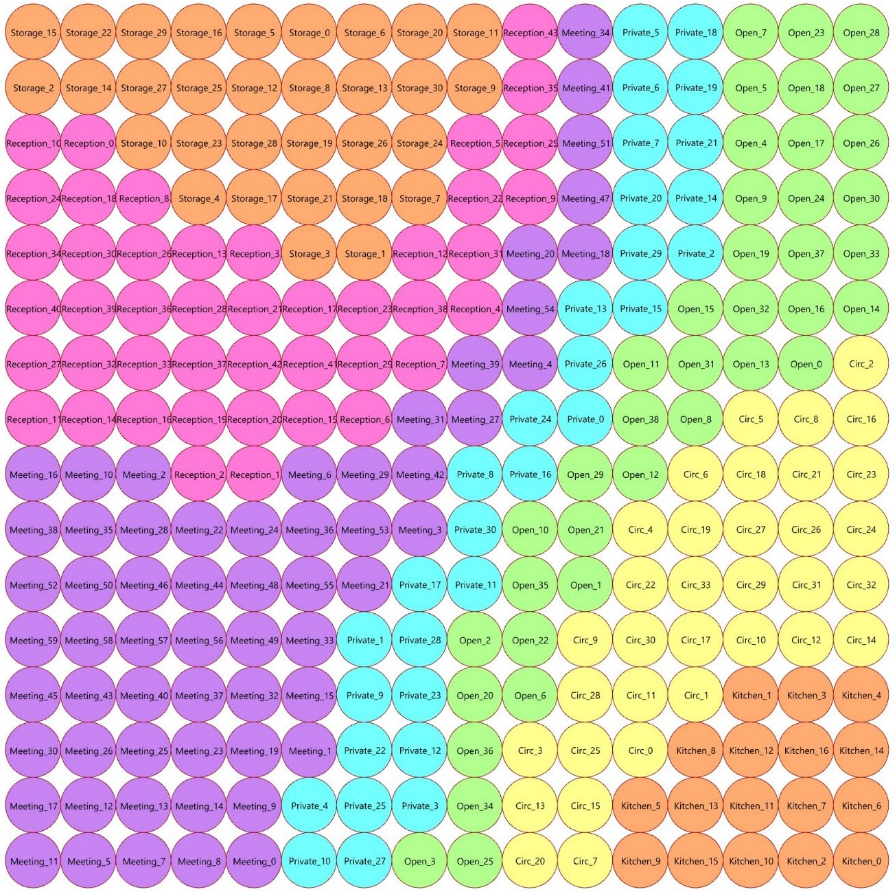


Repulsion ‘equilibrium

Workflow *Method 2 – Kohonen_Program*



Program Identity Gradient

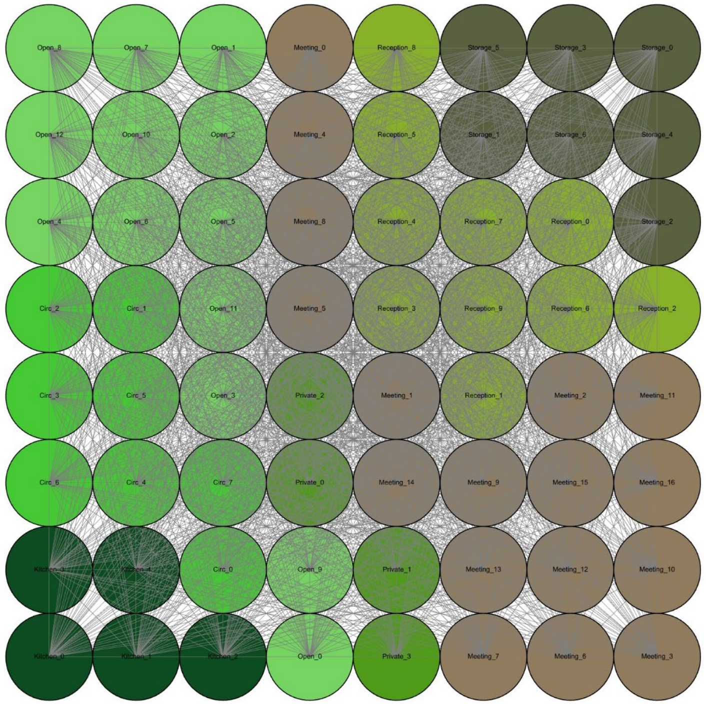


Discretized Identity Gradient

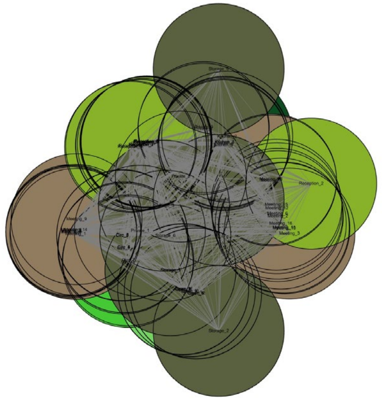
Marco Juliani

ucbqjul@ucl.ac.uk

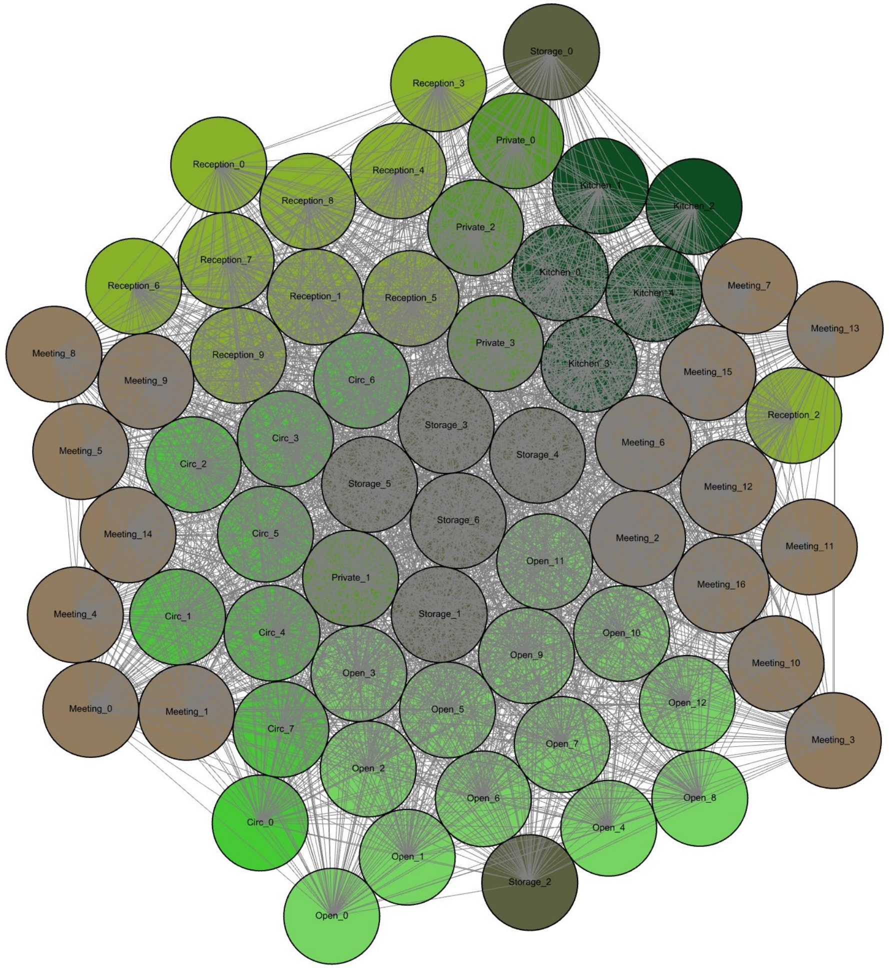
Workflow *Method 2 – Kohonen_Program*



1) Initial State – all springs connected and weighted differently (similar to ANN)



2) Attraction 'equilibrium



3) Final 'fitted' separation state/ Repulsion equilibrium

Marco Juliani

ucbqjul@ucl.ac.uk