Masdar Reimagined

A Generative Approach to Designing a Walkable City

Urban Sentience UBLLV1-15-M

2022

Chukwuemeka Nwosu

Felix Mallinder

Abstract

This project uses evolutionary multi-objective optimization to generate urban fabrics with improved proximity and walking distances than the current Masdar model. Beginning with analyzing the outcomes of different urban design configurations from the orthogonal and non-orthogonal grid typologies. We have used specifically designed computational tools to measure the shortest physical distance between locations in Masdar city, also a computational multi-optimization framework has been developed that will enable the generation of optimal urban grid layouts. In our proposal, the definition of a street network has been reimagined using new rule-sets to create an adaptive and inorganic street network that is robust in finding in itself optimal configurations for improved proximity and shorter distance of travel to the required amenities.

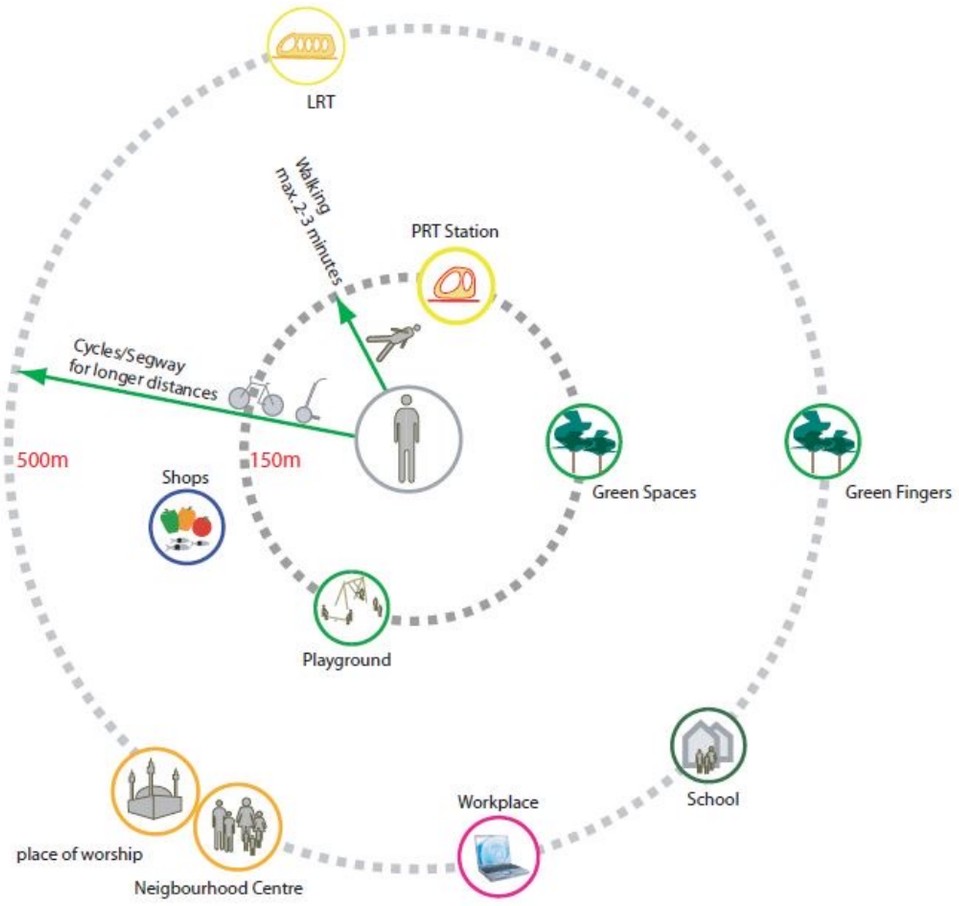
Contents

1. Introduction

Walkability in Urban Design

Walkability refers to the extent to which the built environment supports and encourages walking. This can be achieved by providing for pedestrian comfort and safety, connecting people with various destinations within a reasonable amount of time and effort, and offering visual interest during the journey. There is a general consensus that an area will encourage walking if it has a high degree of proximity (i.e. many functions are within walking distance) and connectivity (i.e. allowing easy and direct routes between destinations). Southworth (2005 p. 249) suggest that "Factors that contribute to walkability include a dense network of footpaths, good linkages with public transportation, a mix of land uses, safety (from crime and road accidents), and high environmental quality with good street design and visual interest". The grid street network in is a system of regularly spaced streets and intersections that form a rectangular pattern. This system is often used to lay out cities, towns, and neighborhoods and has a long history dating back to ancient civilizations. The grid pattern was first used in ancient Babylon and was later adopted by the Romans, who used it to lay out their military camps and towns. The grid pattern was also used in the design of many medieval European cities, such as Paris and London. A grid network can make it easier for pedestrians to navigate and find their way to their destination, as the regular pattern of streets and intersections allows for a clear and logical path. This can make walking more convenient and attractive for residents and visitors. However, the grid pattern has also been criticized for its lack of flexibility and for not taking into account the natural features of the landscape. A grid network can also create long blocks and intersections, which can be intimidating or unpleasant for pedestrians to cross, especially if there is heavy vehicle traffic, and can also result in a lack of visual interest and a monotonous streetscape, which may discourage walking. In general, the grid system is adequate for walkability, but its other draw-backs limits it from being an efficient network system. Computational optimization (CO) approaches are increasingly employed to address challenging design problems, even though CO applications at the urban design scale have been limited compared to architecture due to enhanced complexity and computation requirements. Genetic algorithms are a type of computational optimization algorithm that can be used to solve complex problems, including those related to urban design and walkability. In the context of walkability optimization, genetic algorithms could be used to find the most efficient arrangement of streets, sidewalks, and other pedestrian infrastructure in order to minimize the distance that people need to walk between destinations. To use a genetic algorithm, a set of "chromosomes" representing different city layouts would be created and evaluated based on fitness criteria related to walkability. The chromosomes that perform best according to the fitness criteria would be selected to "breed" and create a new generation of chromosomes, which would be evaluated in the same way. This process would be repeated over multiple generations until an optimal layout is found. Genetic algorithms are well-suited for solving multi-objective optimization problems, such as those related to walkability, as they can handle a large number of variables and constraints and can adapt to changing environments. Overall, genetic algorithms could be a useful tool for optimizing walkability in urban design by finding efficient and convenient arrangements of pedestrian infrastructure.

2. Background



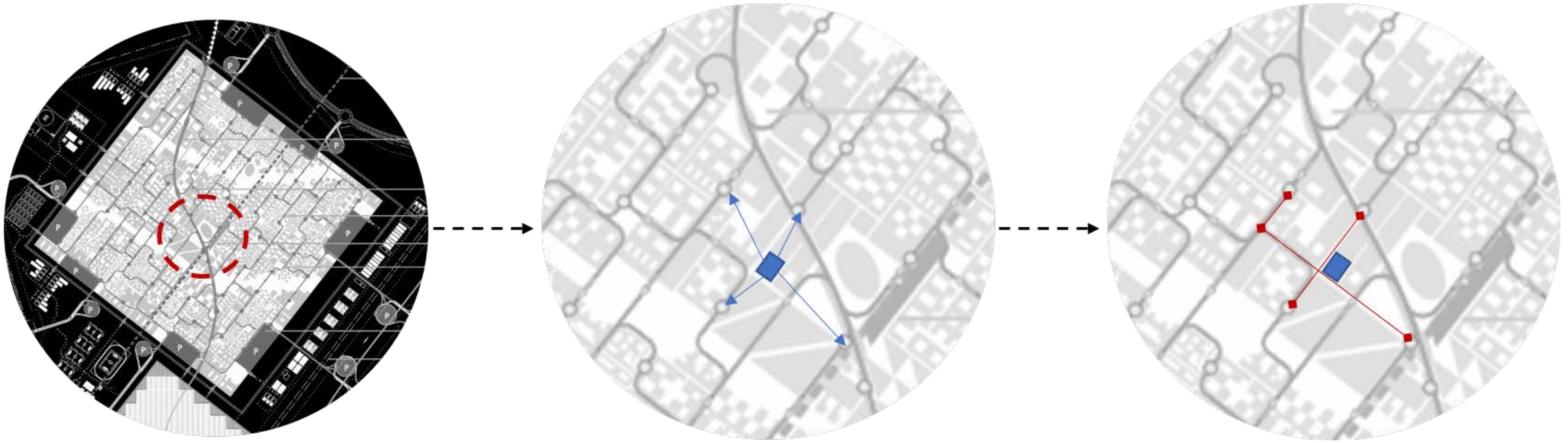
How can we improve Walkability in Masdar City?

The grid system was used in the spatial design of the existing Masdar development, which highlights improving walkability as a principal consideration for this choice, however the current Masdar proposal fails to tackle the other inherent issues of the grid system. Our approach to this brief is to design the city of Masdar as a walkable city and reimagine the grid as an inorganic network breaking away from the regular and the repetitive to create a highly dynamic and inter-connected organic network that will improve the issue of walkability in Masdar city and eliminate the evident drawbacks of the grid system that has been adopted. How would we design a grid for walkability? Looking at the existing Masdar plan we can observe that the system was not designed with walkability as the primary objective but rather bus transit stations have been introduced at 150m proximity from points to reduce walking distances from the point of travel to the transport stations, This 150m proximity doesn't take into consideration the distance based on the existing spatial layout in Masdar, upon testing the Masdar model using the shortest walk algorithm, the walking distances were discovered to be larger than 150m due to the existing spatial configuration. The Masdar model shows a weakness in that it does not directly address the issue of improving walkability, but places transport stations at within proximity to the pedestrians. The Masdar spatial layout also lacks the non-linearity, self-organisation and emergence that adaptive and evolutionary systems exhibit. Our project redesigns Masdar city, making walkability the goal and the driver for the whole design evolution. We have emptied the current Masdar city of all its existing features and reimagining it as a virgin site so we can re-define a new approach for what a walkable Masdar city is. Our design is derived from using multi-objective optimization to minimize the walking distances from any point in the city to the required ammenity by utilizing the shortest walk algorithm in the grasshopper environment to generate data for optimization.

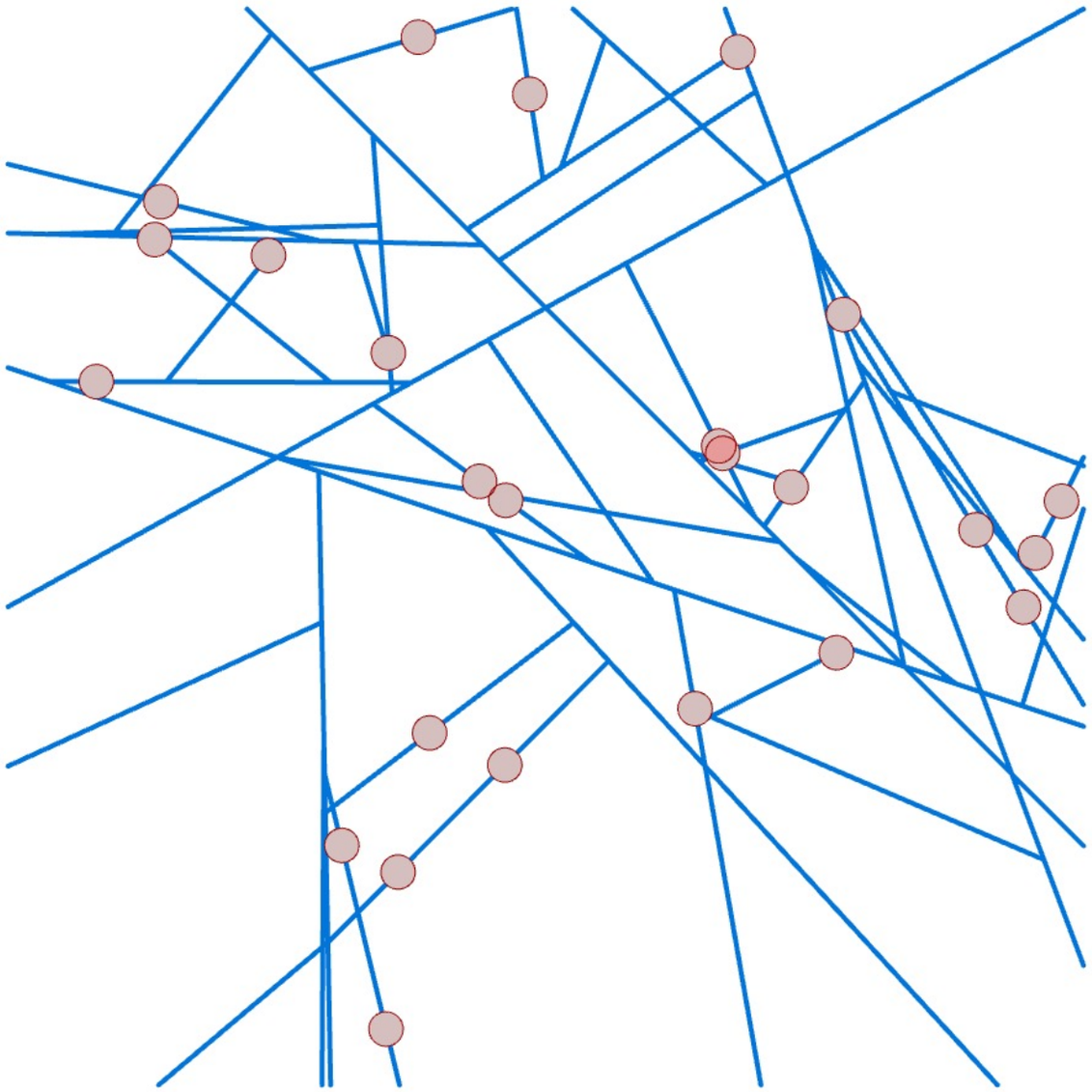
How can we improve Walkability in Masdar City?

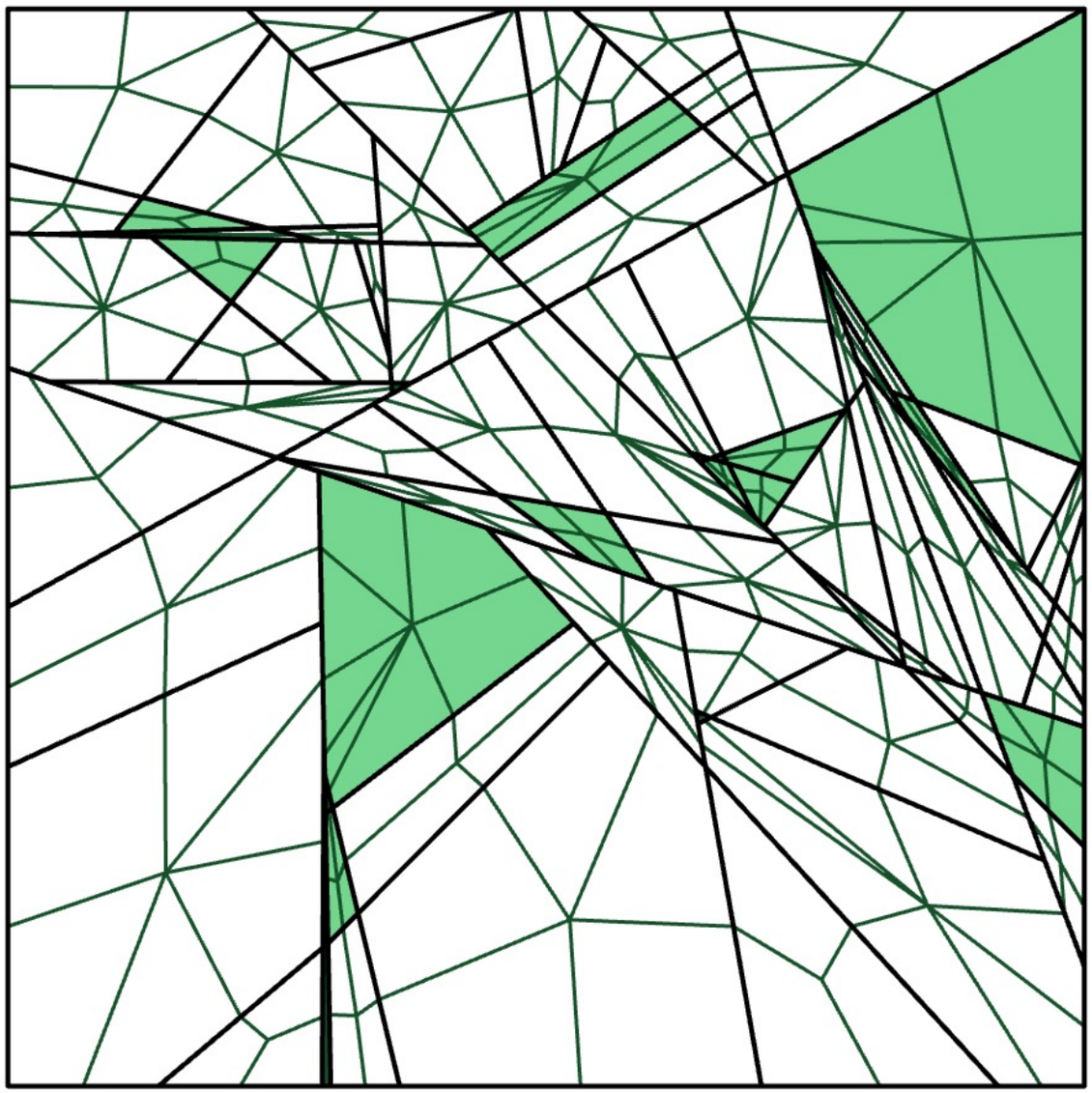
Proximity of 150m does not translate to shorter walking times.

Walking distances are increased due to spatial layout of city.

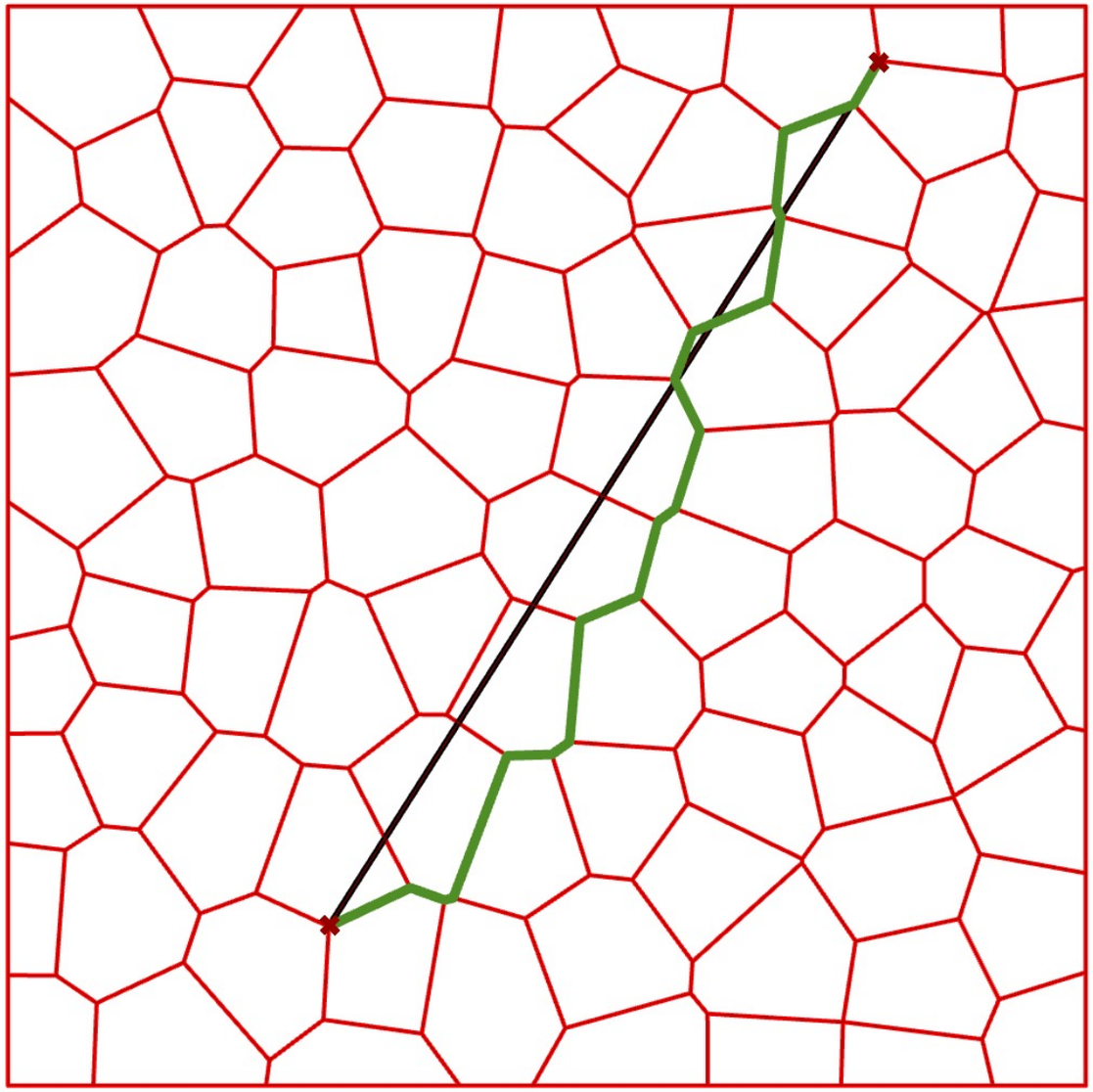


3. Methodology





3.1 Testing Criteria

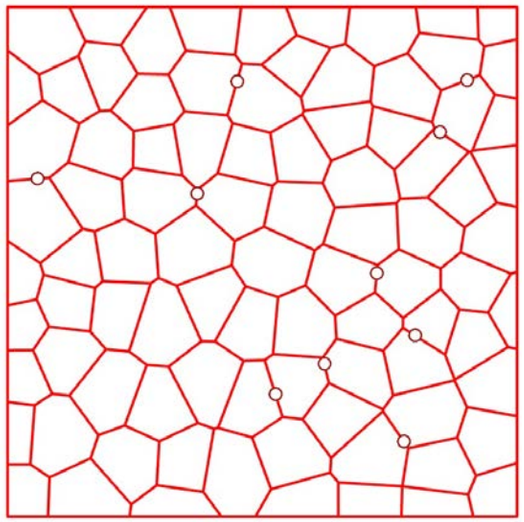


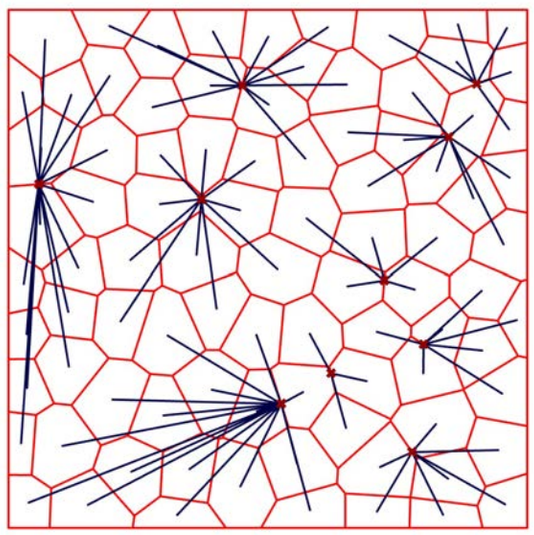
Building on the idea of walkability vs proximity in the urban environment. This study utilises a grasshopper plug in, shortest walk.

As shown in model (left) the plug in takes a straight line between points and computes the shortest walkable path through the network. This plug in allows us to test the truth of Masdar's accessibility through the context of its spatial layout. Further, we can use this function to develop a consistent testing methodology to be applied in future design models.

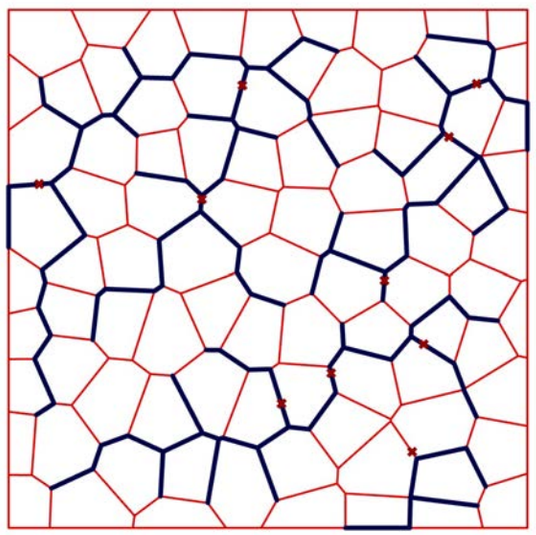
First, the shortest walk plug in is used to develop a tool that tests the walkable distances to the closest placed node with n a network, for all cells. This is developed, parametrically such that Galapagos , a machine learning optimization tool, can control the placement of nodes to seek the optimal results for standard deviation, average or maximum walking distances.

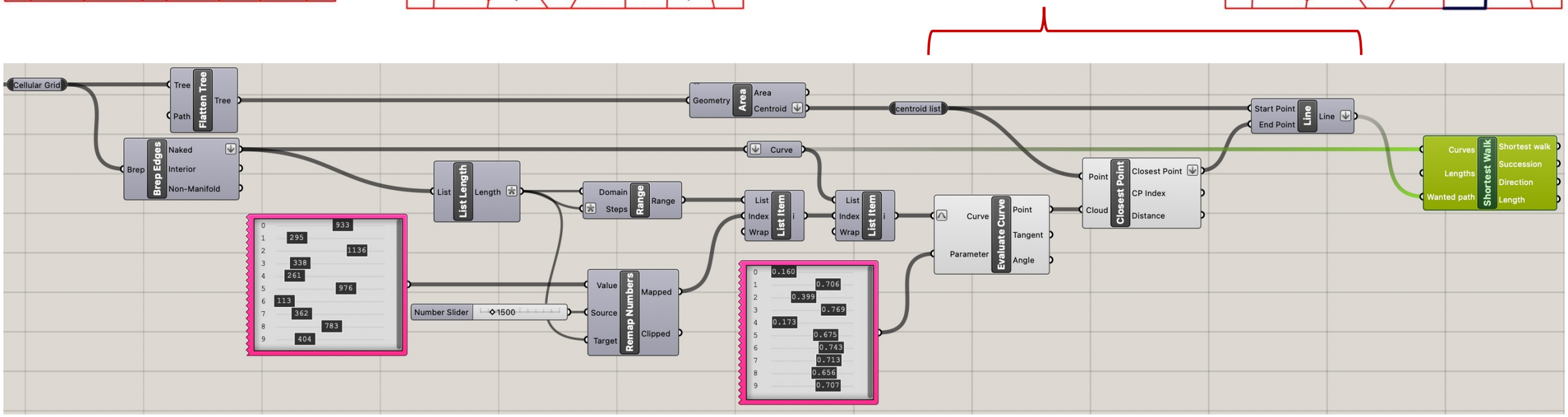
Transport Proximity





3 Here, the code highlights the proximity vs walkability concept. proximity lines are calculated by drawing lines between each centroid and the evaluated closest transport node. The resultant collection of lines is fed into the shortest walk plug-in tool and generates the following modelled shortest walk paths





1 This section of the code uses a gene pool to select the list by index in the list of grid lines. Remapping the large range of gene pool to the length of the list means that the algorithm is robust for generative grid designs.

In this instance 10 genomes are selected representing 10 lines for transport nodes, as shown in the model images.

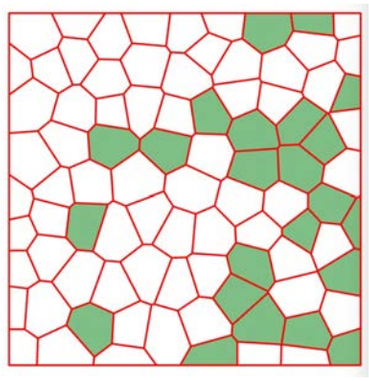
2 This section again uses a gene pool this time the genomes exist in the range 0 to 1 with 3 d.p. the curves are reparametrized and the geneome is value is used to select the location of transport nodes along selected lines.

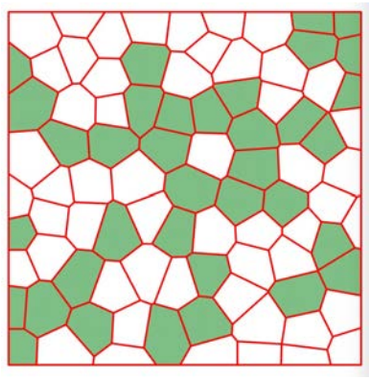
Thus, allowing up to 1000 permutations of node location per line.

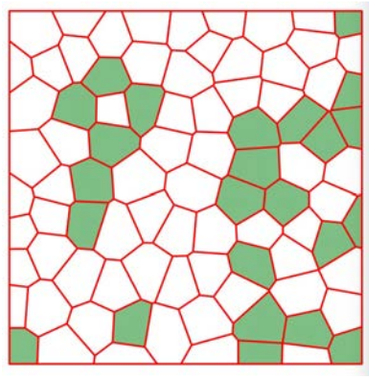
Green Space Proximity

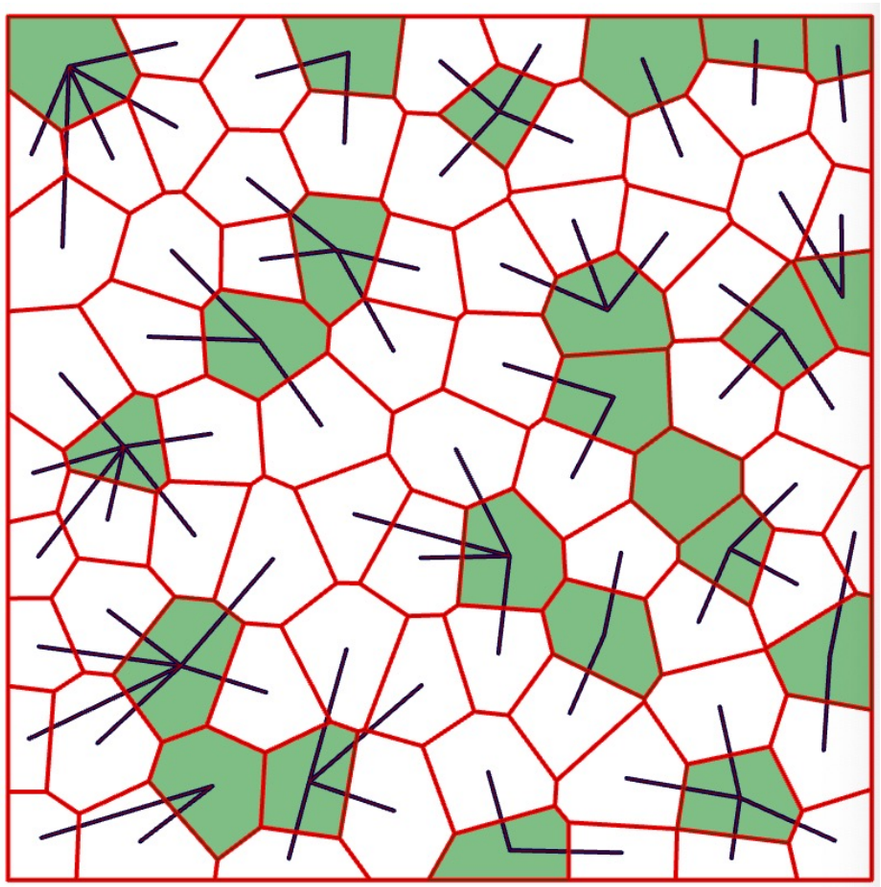
In this instance the the shortest walk plug in is again utilised to develop a quantification of green space accessibility.

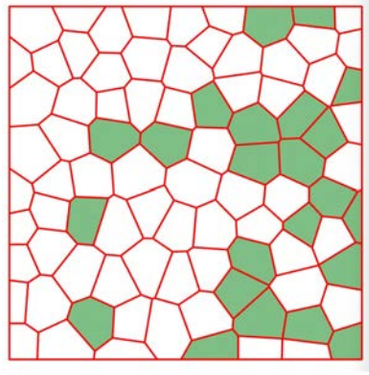
Again the calculations of this model are utilised as fitness criteria for design optimisation, with the genomes being locational parameters of chosen green space cells.

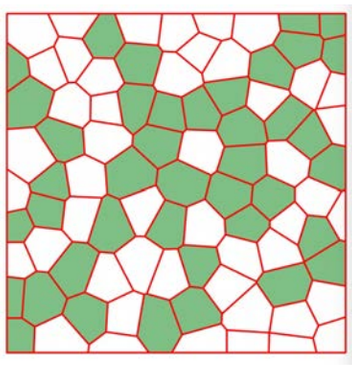






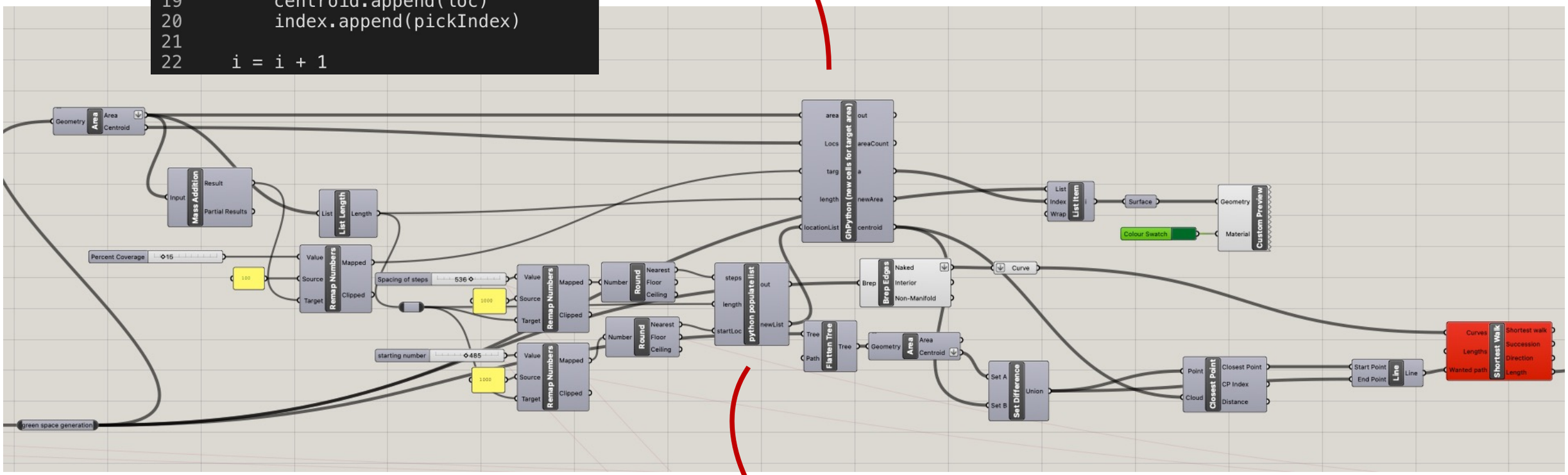






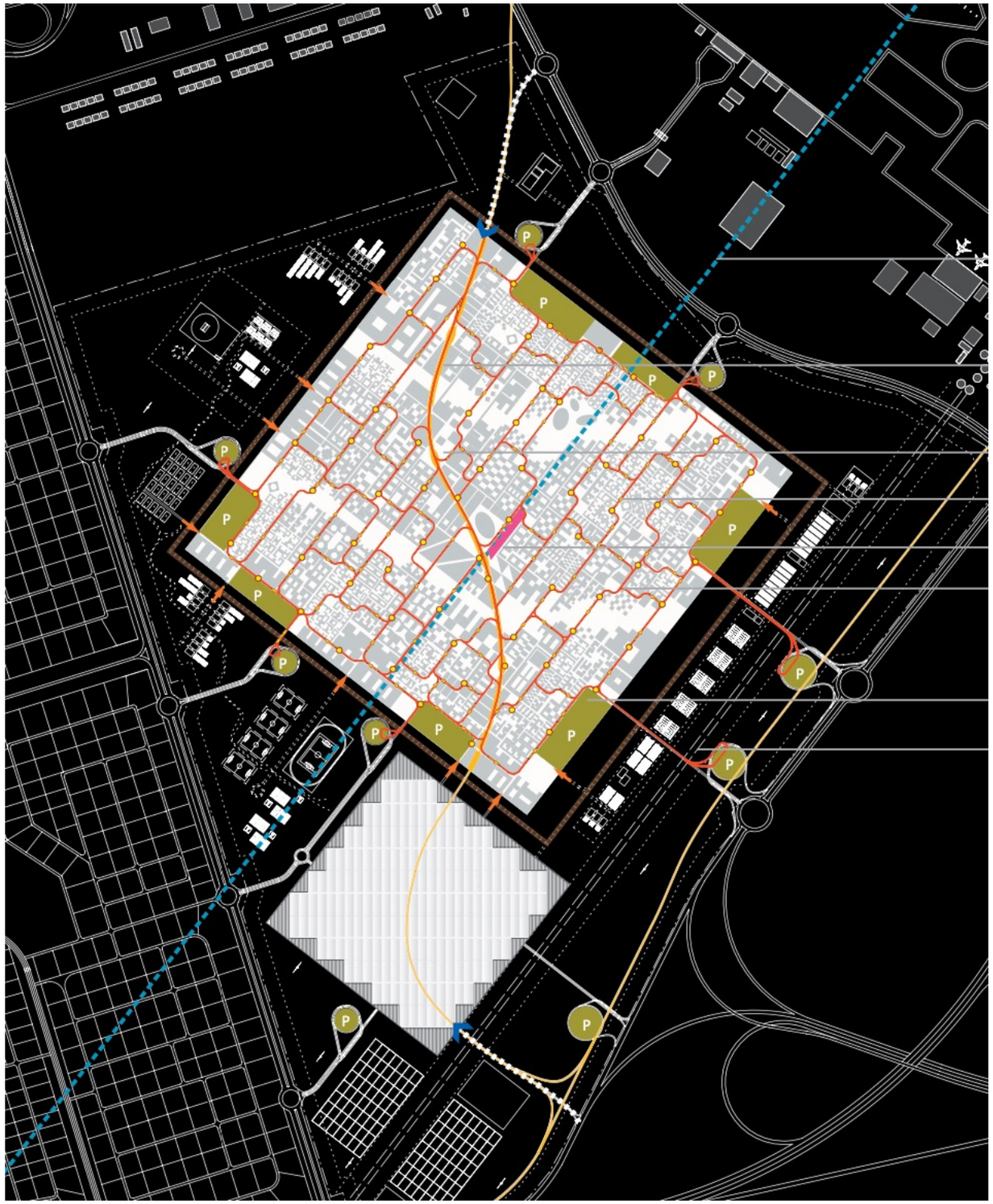
```python
import rhinoscriptsyntax as rs

centroid = []
index = []
areaCount = 0
i = 0

while i < length :

    pickIndex = locationList[i]
    pick = area[pickIndex]
    loc = Locs[pickIndex]

    if (pick + areaCount) < targ :

        areaCount = areaCount + pick
        centroid.append(loc)
        index.append(pickIndex)

    i = i + 1
```

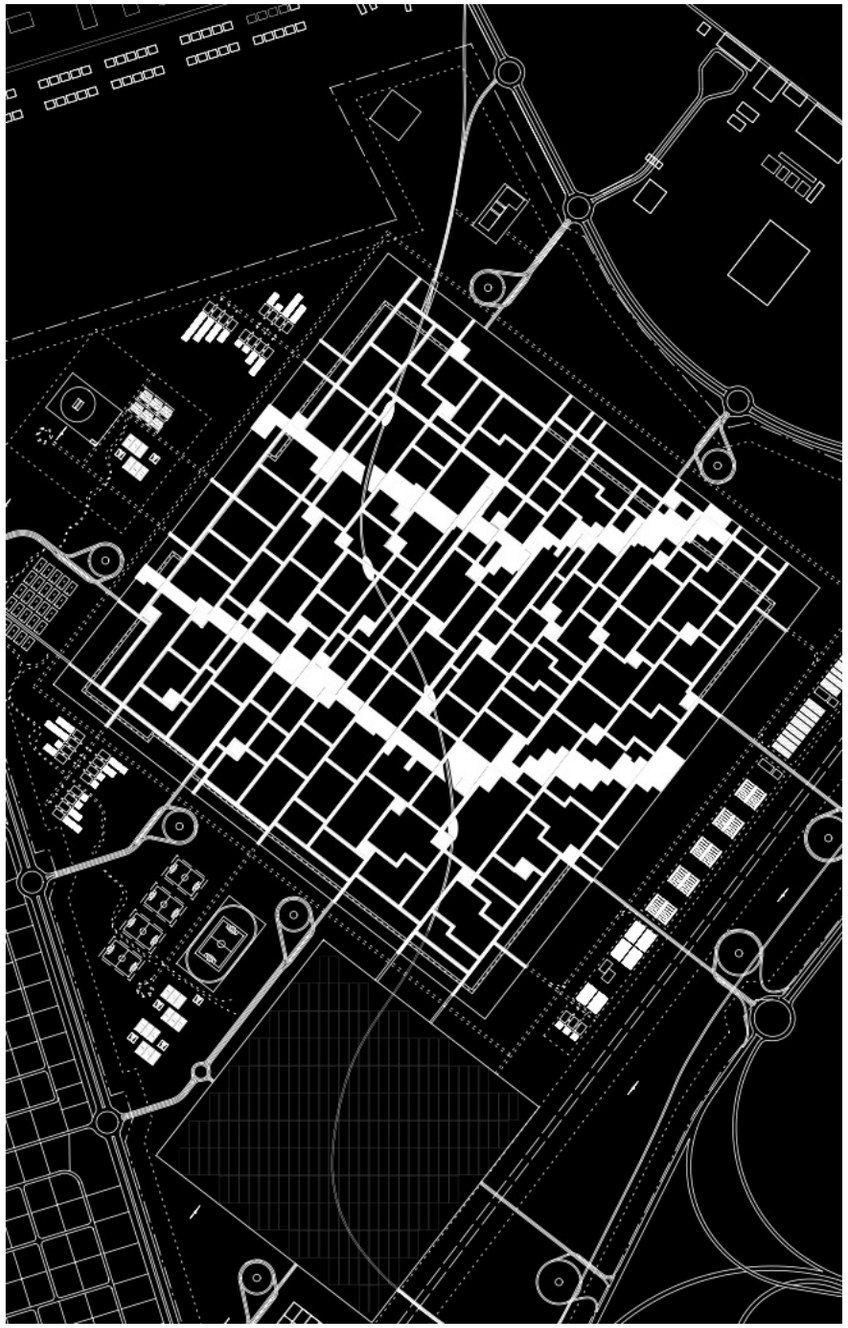
2 This Python component utilizes the list permutations from the function below. Doing so, it picks the cell of each index and adding the area to a total area, continually check if the cumulative are is below the target. Producing a series of cells of which the total area is as close to the percentage coverage required.



1 Applying a ‘shuffling’ algorithm to the list index, allows for control over list variations. This algorithm has two parameters, starting index and step values, both values are remapped to within the length of the index list, thus allowing for the number of possible permutations to = $(list\ Length)^2$

```python
import rhinoscriptsyntax as rs

count = startLoc
newList = []

a=0
i=0


while i < length:

    if (count + steps) > length:
        count = (count + steps) - length
    else:
        count = count + steps

    newList.append(count-1)
    i = i+1

```



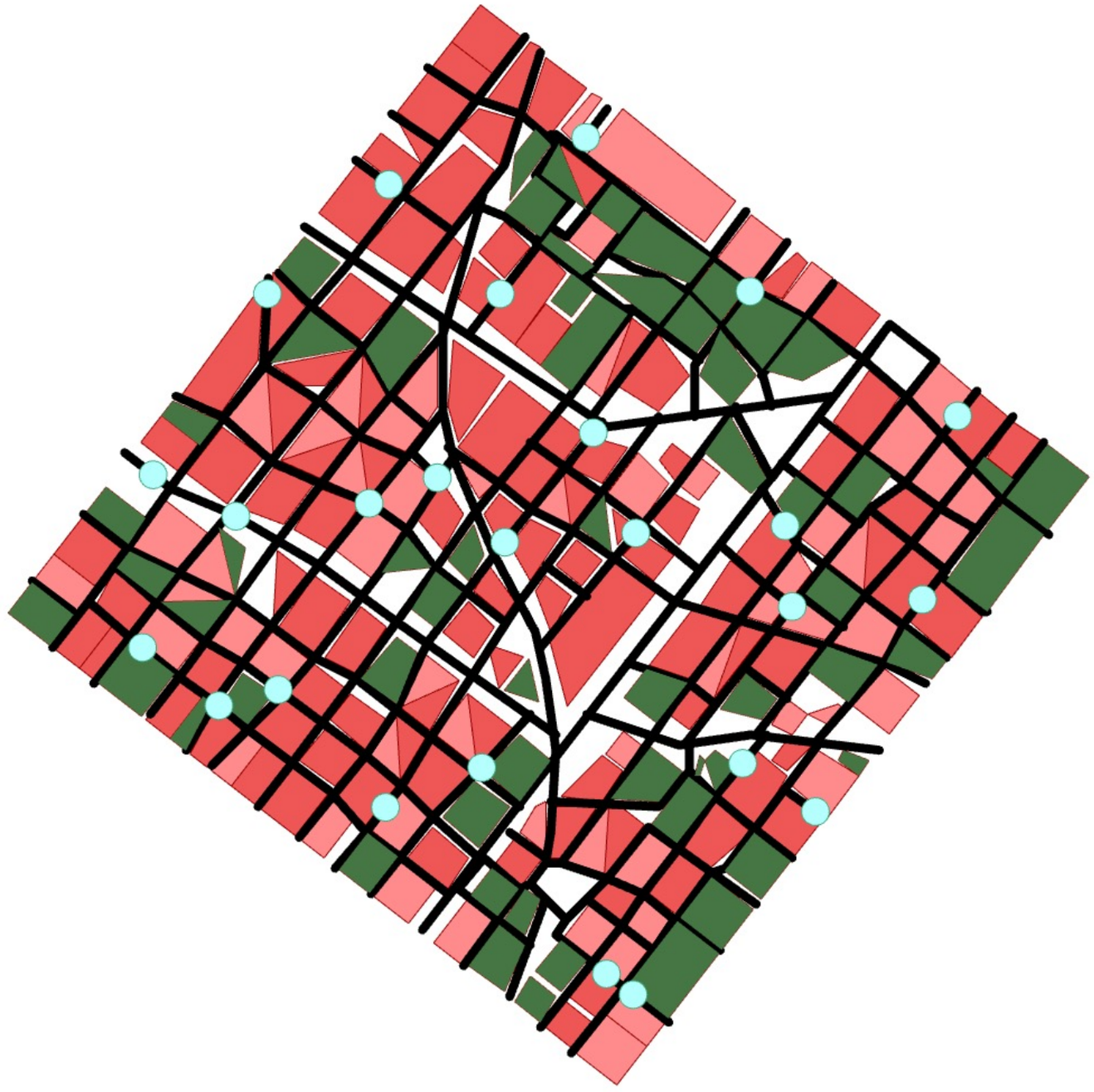


3.2. Testing Masdar

The previous models represent the existing spatial and transport networks of the Masdar proposal. Computing this data with the quantification tools developed for walkability and accessibility we can critique the claims of the Masdar masterplan.

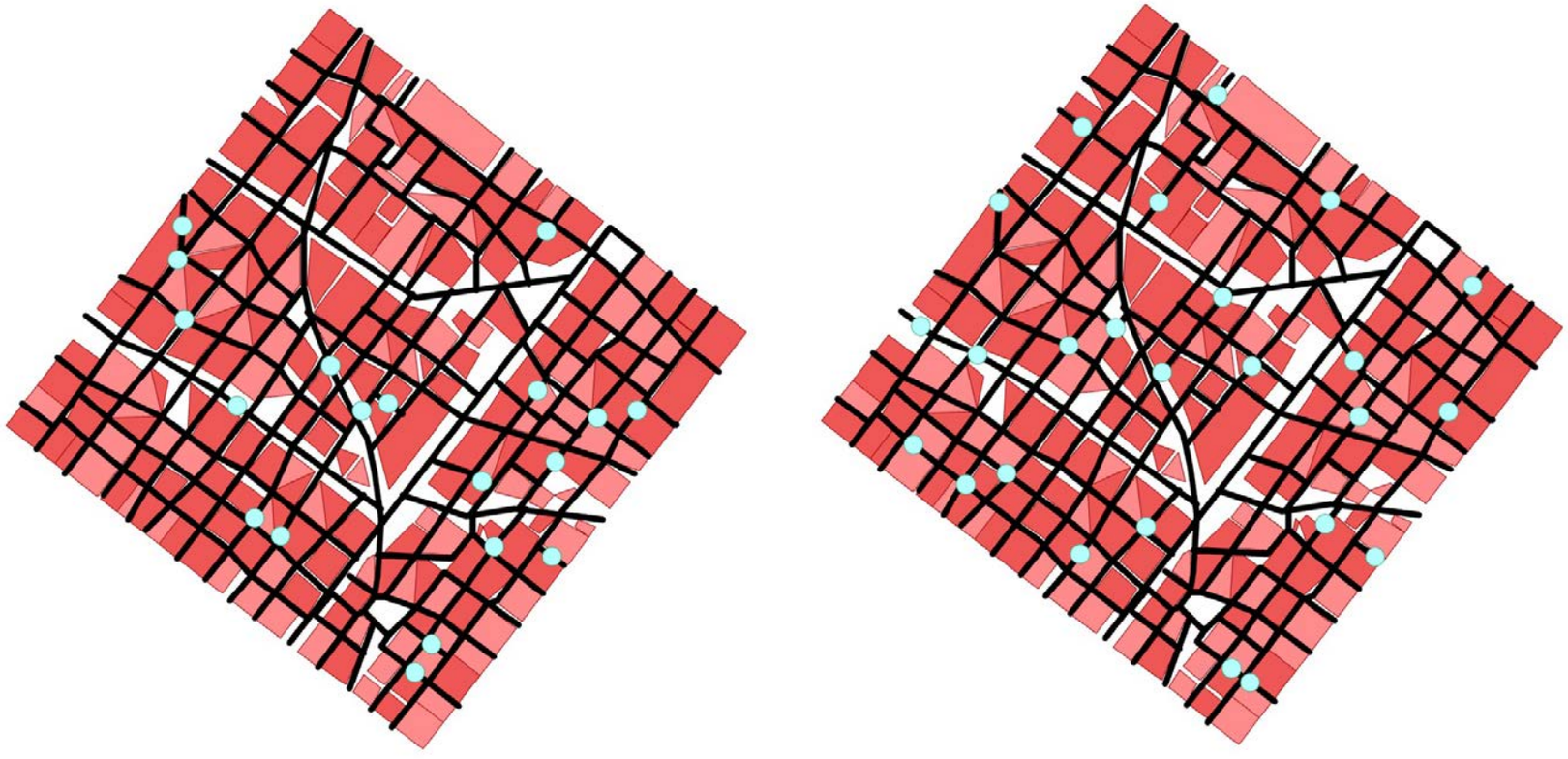
Further, applying Galapagos machine learning optimisation we can seek improvements the results of the proposed Masdar design, by making use of the parametric design solutions. The following pages showcase the visual implications when applying this testing criteria.

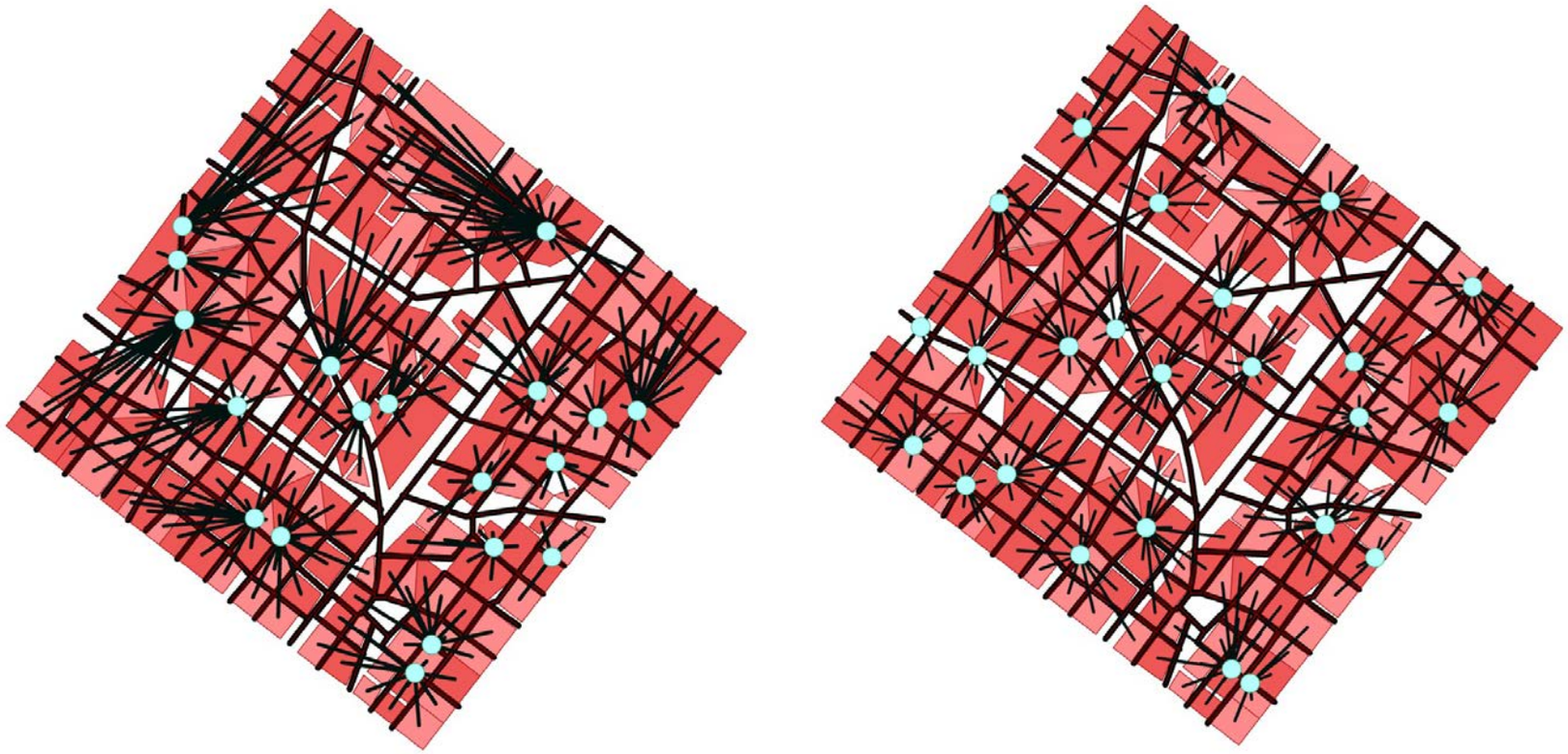
On the right, shows a vision for an improved Masdar masterplan with optimally reduced walking distances.

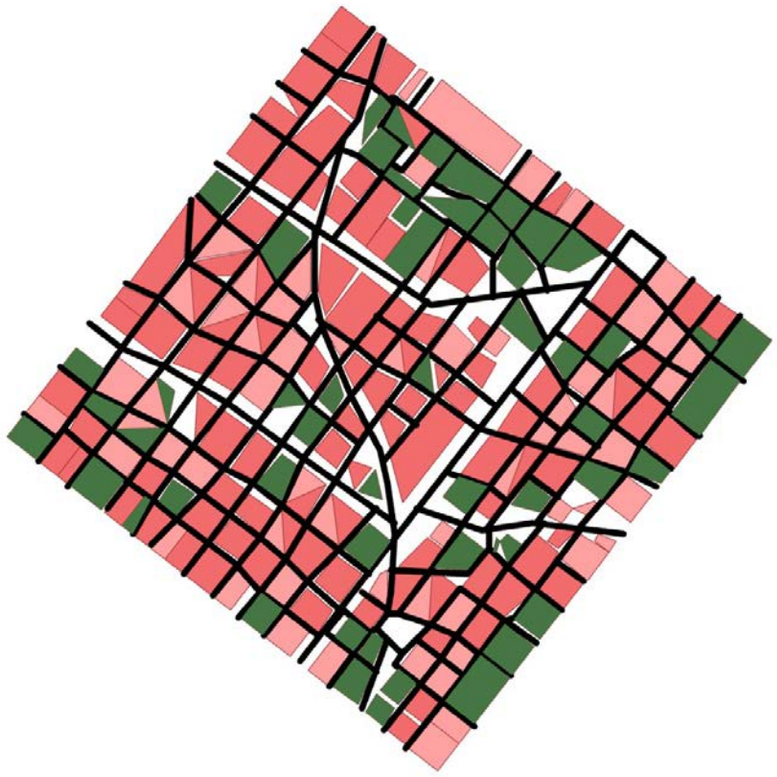


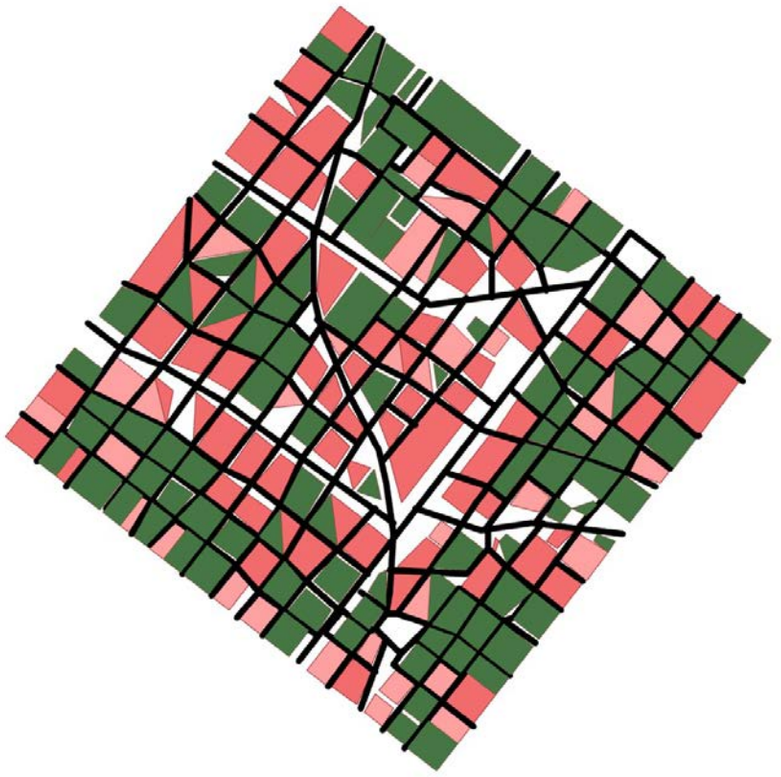
Applying the previously defined transport node modelling, we can seek optimal improvements in the node location in Masdar for reduced walking distances and reduced standard deviation of results.

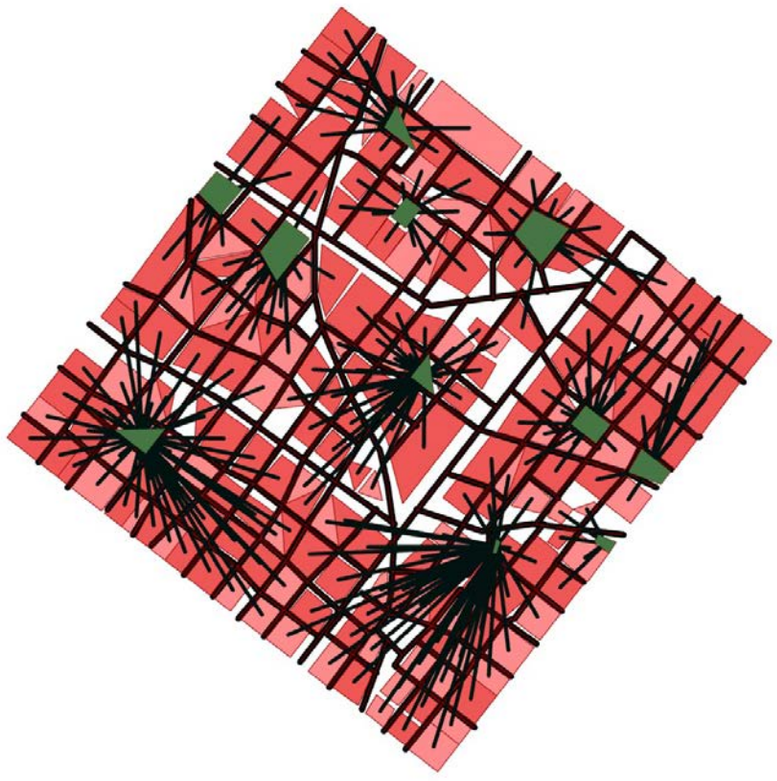
The comparison models showing the cells ‘served’ by each transport node. It is evident that the optimised model (right) has an improved functionality of proximity based locations.

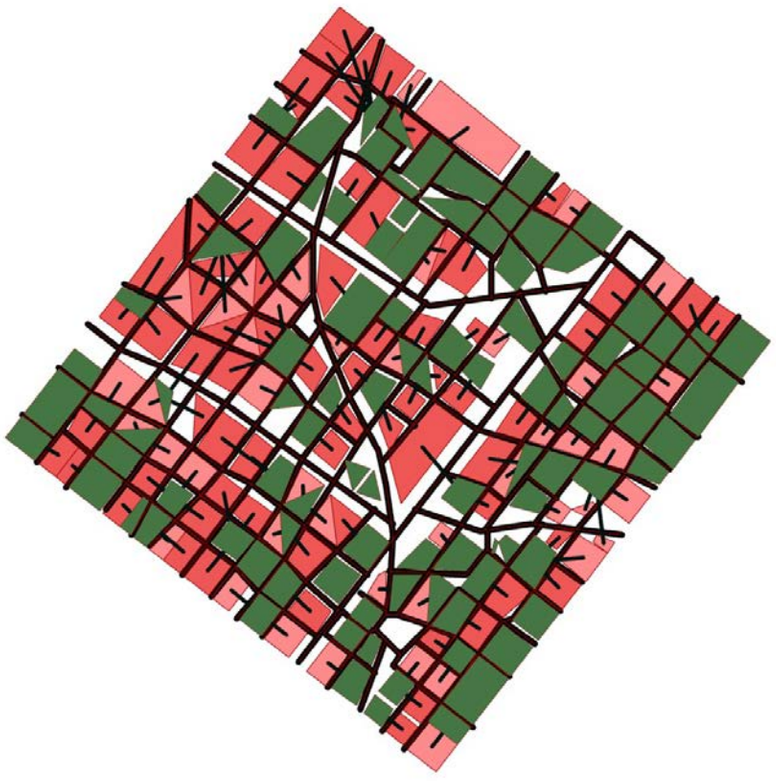












Here the model iterations show the reduction in green space proximity for design coverages of 10% and 40% - Its intuitive that the greater coverage will have a reduced proximity/

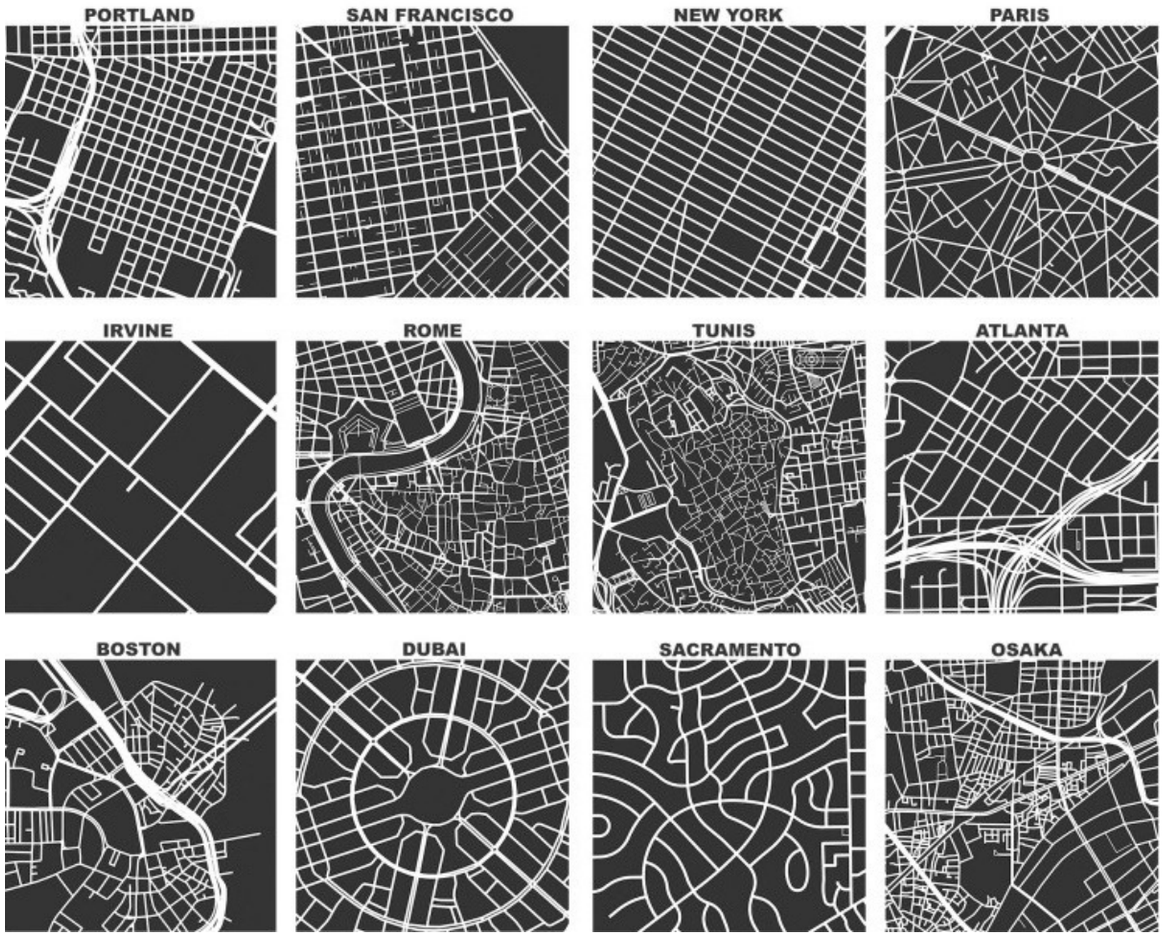
Although this modelling also highlights the importance of placement of green space. The model bottom left shows a poor deviation of proximity, with the majority of Masdar being served by just two cells in the south west.

3.3. Designing a grid

The next step in improving the walkability of Masdar is to rethink and redesign the spatial layout. To do so we must develop a generative network design, such that it can can morph and conform to optimal walkability. If we wish to use the same testing methods the grid design must be equally parametric, so how can we parameterise an organic design ?

Historical grids

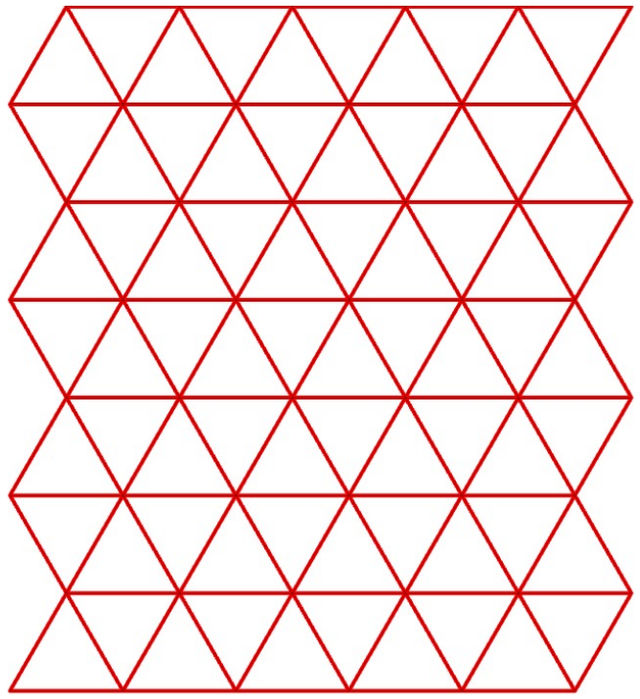
Grid systems exist in many permutations through age and global location. Walkability varies across different designs, but almost none are primed for pedestrianisation. Repetitive, cost effective architecture and automotive transport control the narrative. So how do we start again? What does a user centric design consist of and how can we make it organic?

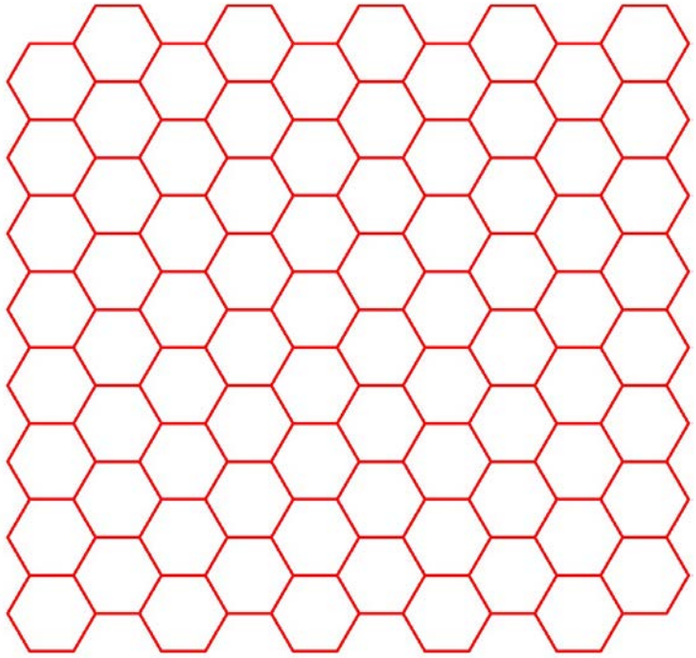


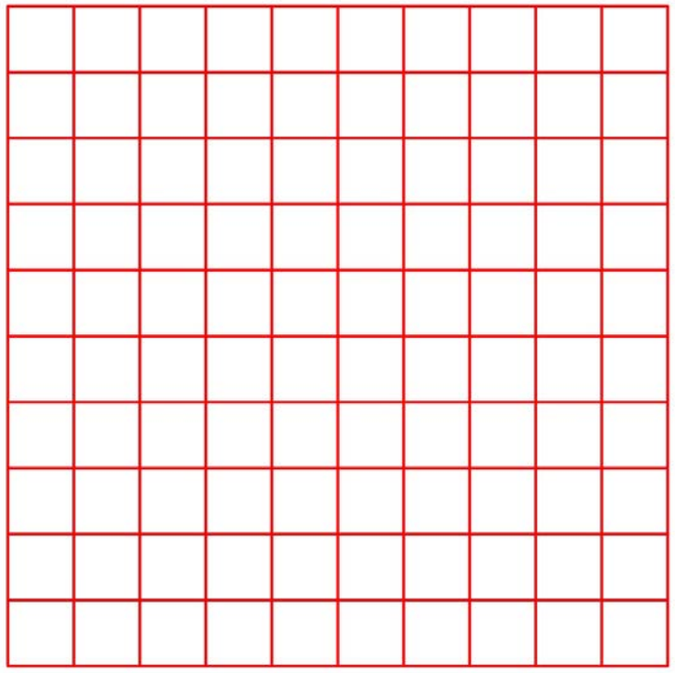
Repeat pattern grid

To follow the existing grid methodology, we can explore geometrical influence by using alternative repeat patterns. In this case the parameters would be control over shape, number a recursion pattern

This is a cellular approach to a network system, allowing for permutations and generative morphing.





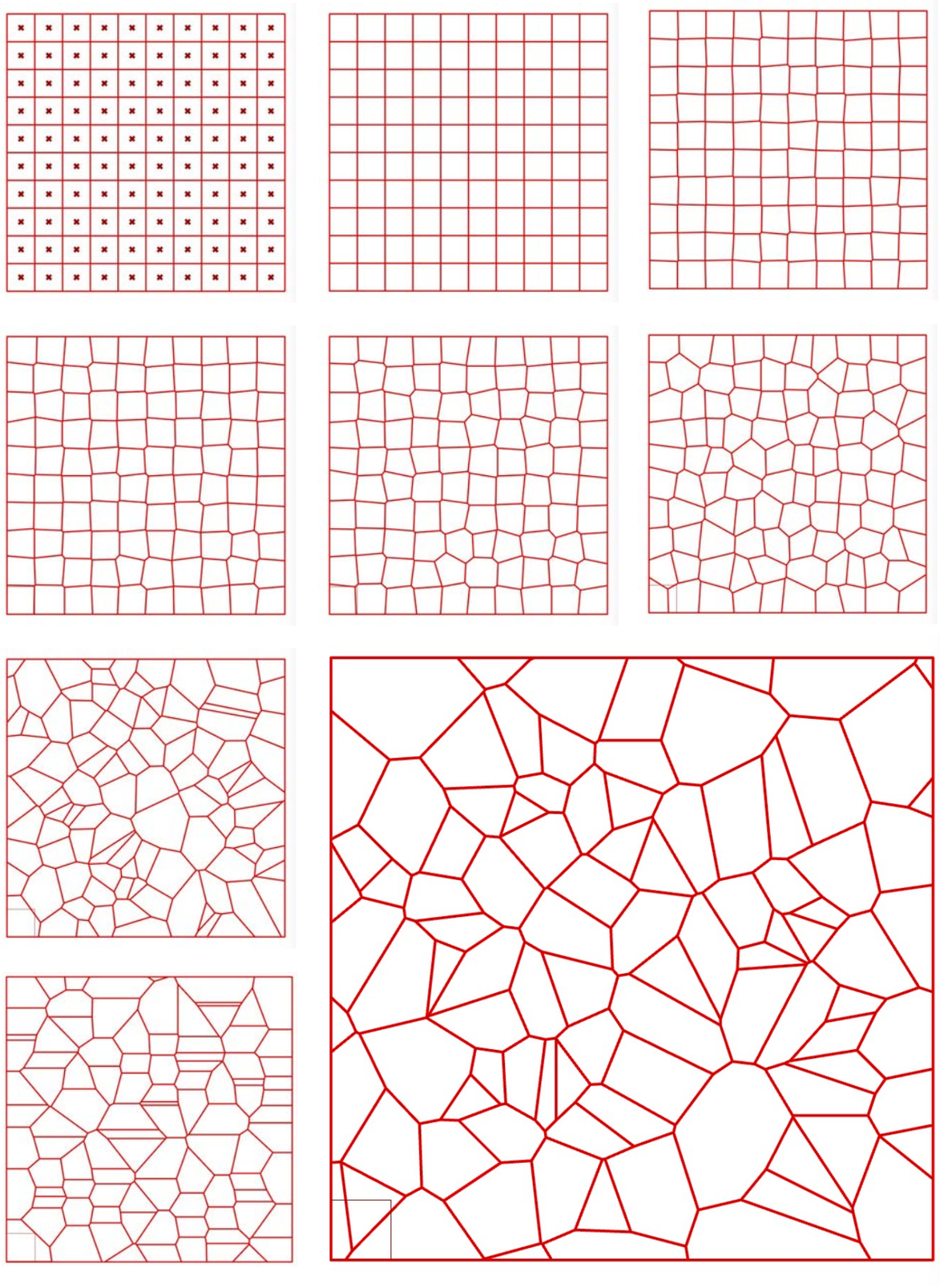


3.4. Organic Network

Developing a control function for cellular network could look like the following:

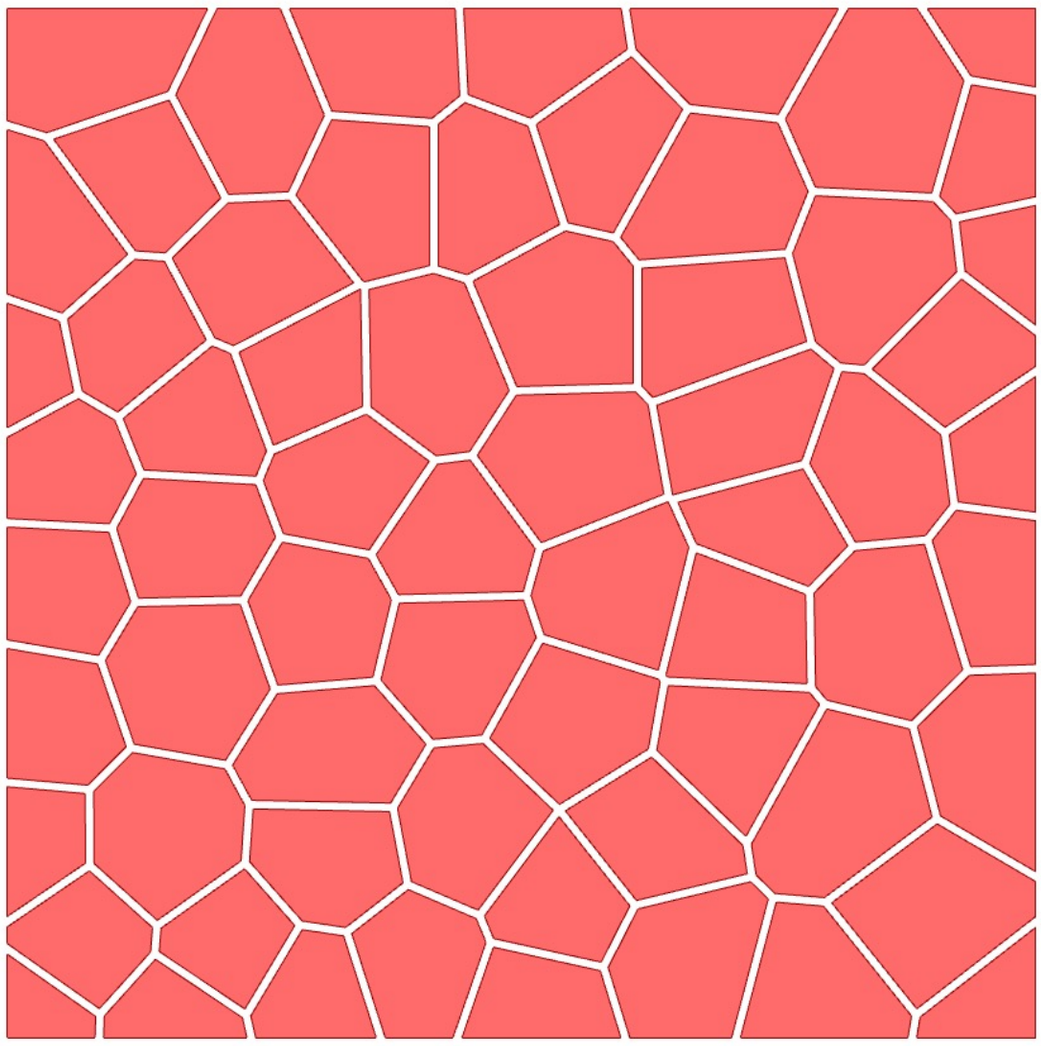
In this case, the control genomes are the x,y movement applied to the centroids of a basic cell (image 1). As we recursively apply further movement the geometry of each cell, the network starts to develop a more unique form. Finally developing into an organic network, primed by testing criteria.

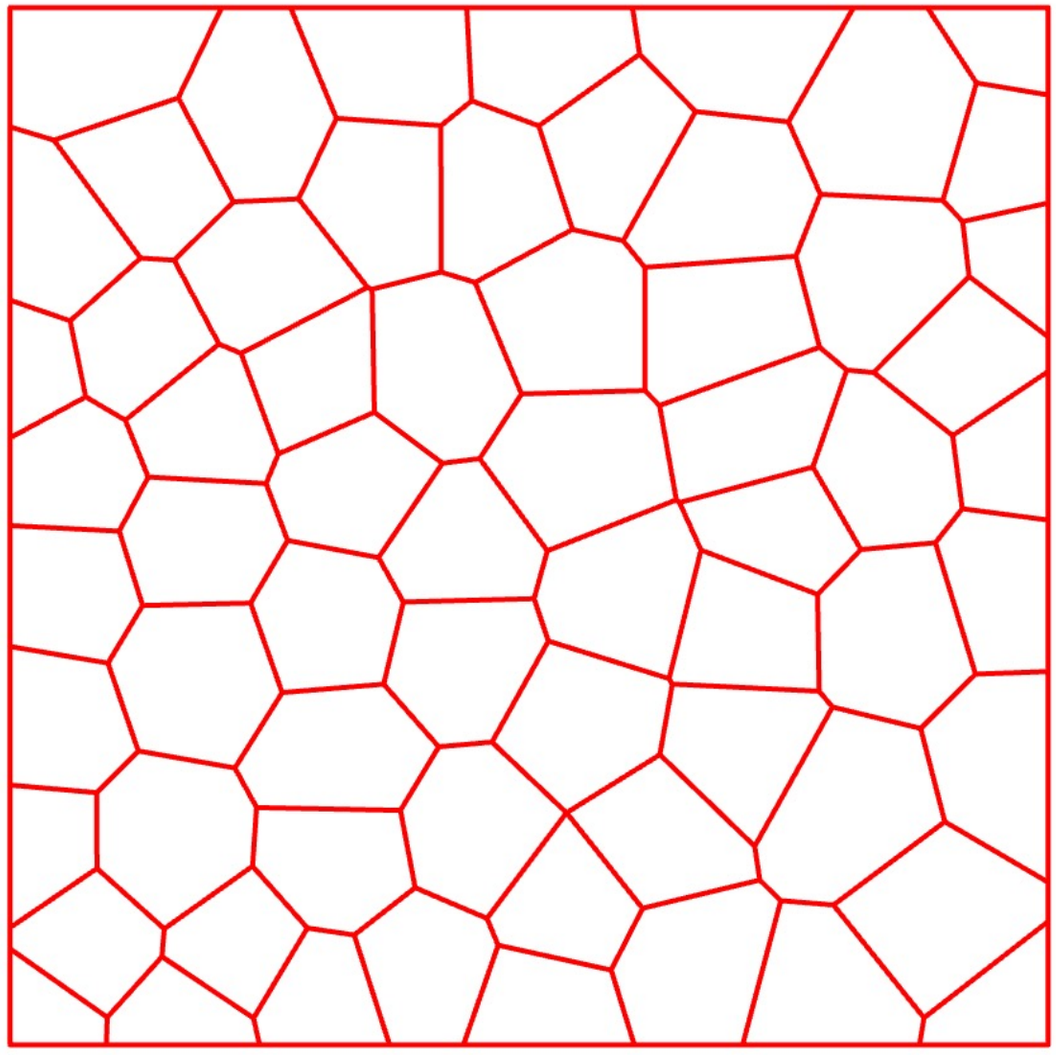
Although this method allows for unique generation and removes a lot of the repetition issues of the modern grid system. It is still not free of control, the number of cells in this series is pre set meaning the truly organic form we seek is unattainable with this method.



Cells and Lines

Seeking organic design, we must look beyond the cellular approach of the grid system and consider the form that cells depend upon. In the case of urban design, grid systems are restricted and controlled by their paved networks. If we seek to parameterize the lines that form such networks we can provide greater generative control for an optimised solution.



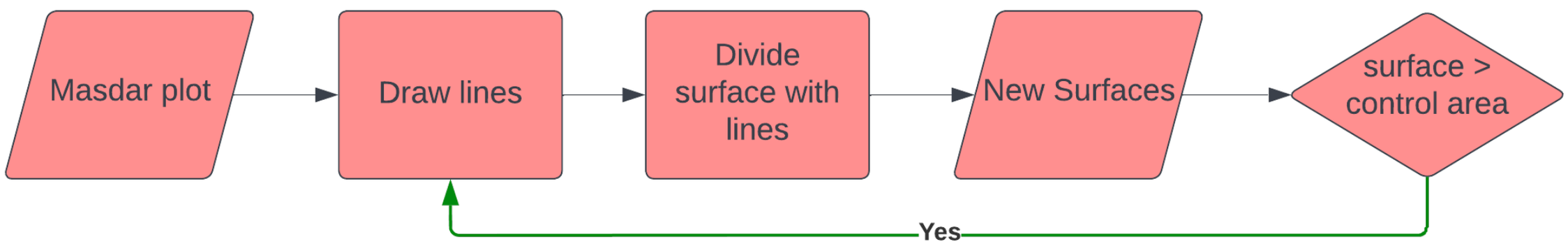


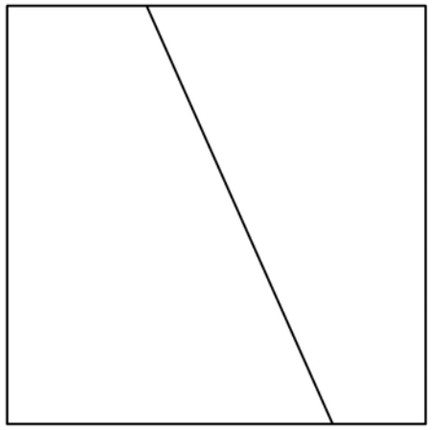
Recursive Line Approach

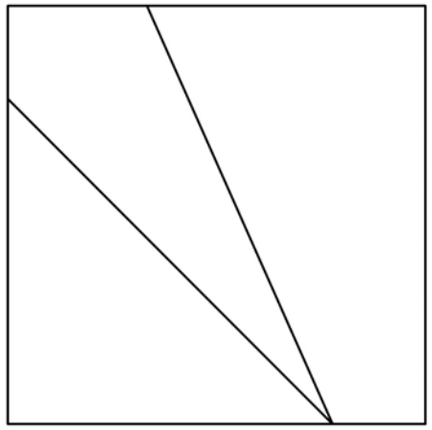
The recursive line approach was chosen as an alternative approach to cellular networks. In this case a number of lines are generated crossing the site, with the start and end point parametrised as geonomes. Thus allowing minimal influence when seeking organic design. These generated lines divide the surface area into small surfaces, of which the same process is carried out

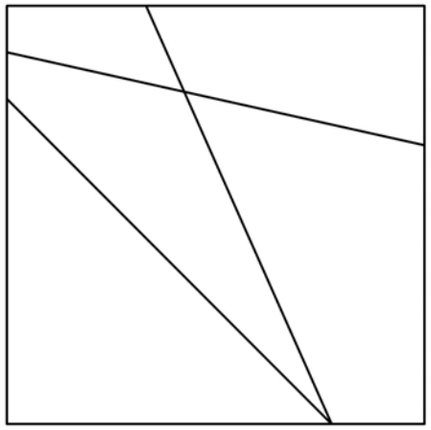
As shown this approach aims to be a recursive one, continually repeating the simplistic line generation and surface division. Developing the network in the context of an urban environment, control for minimal surface size prevents further division for unbuildable plot sizes.

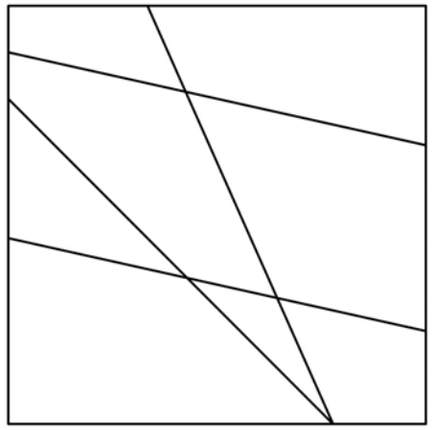
Final plots add pathways, connecting the centroids of each plot directly to the bordering lines.

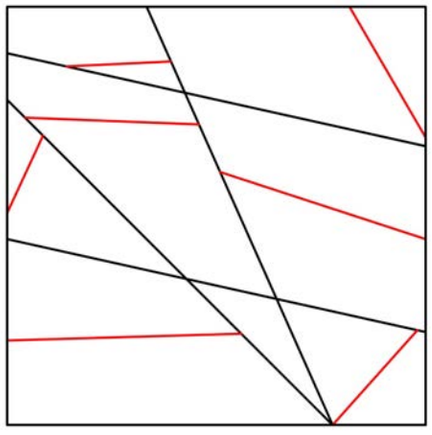


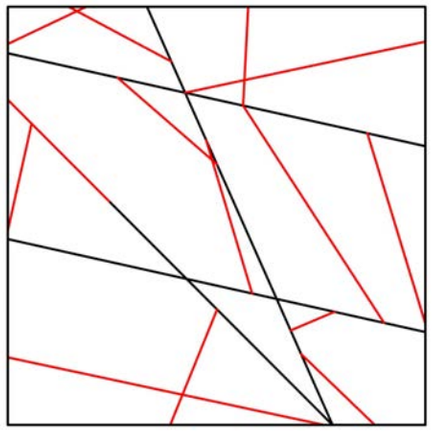


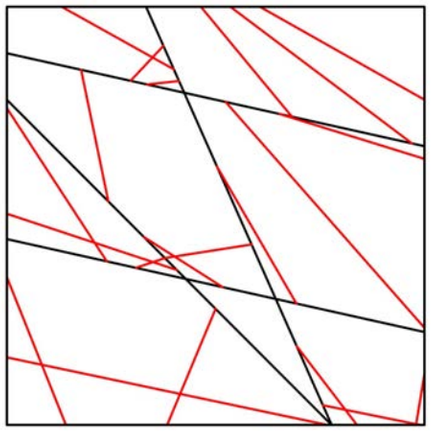


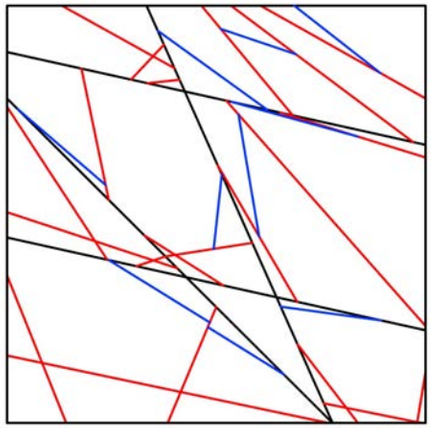


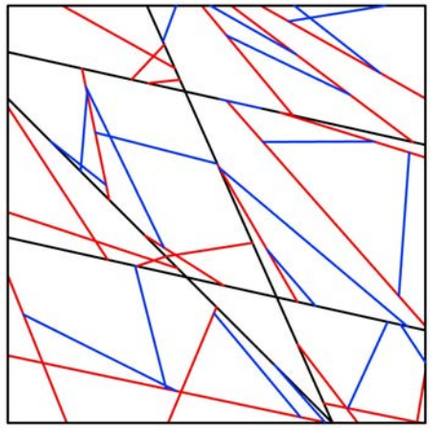


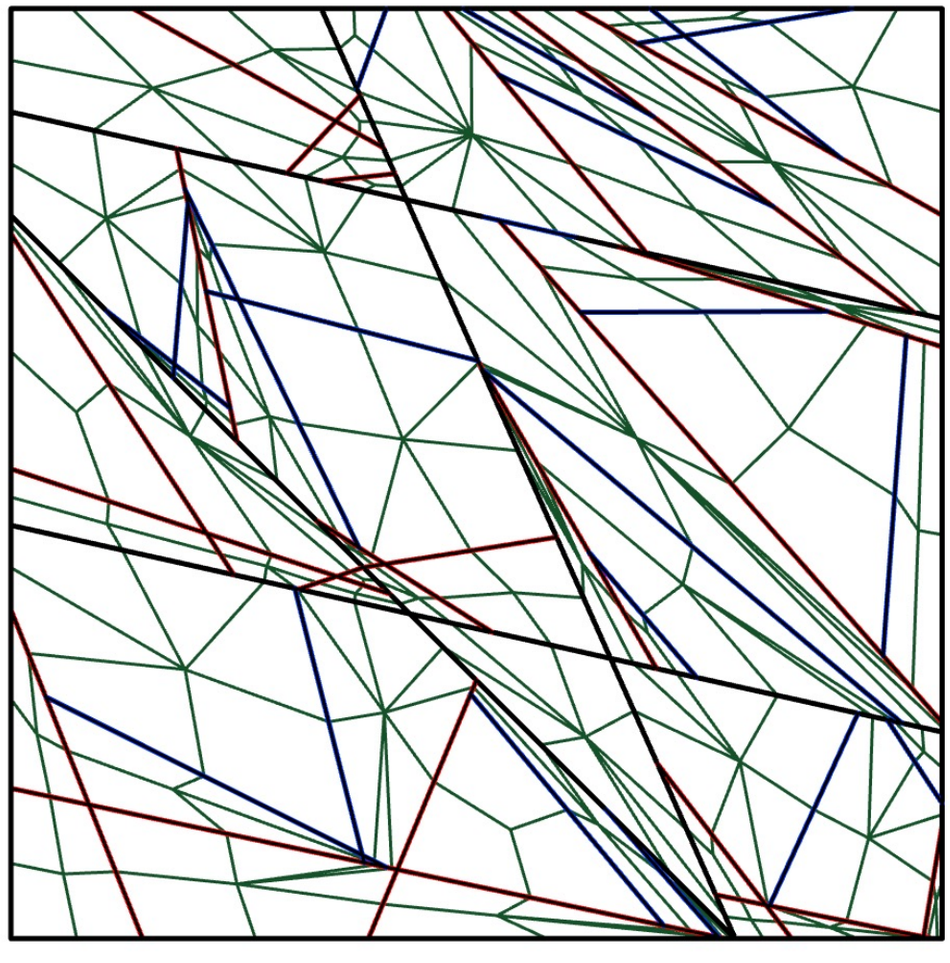






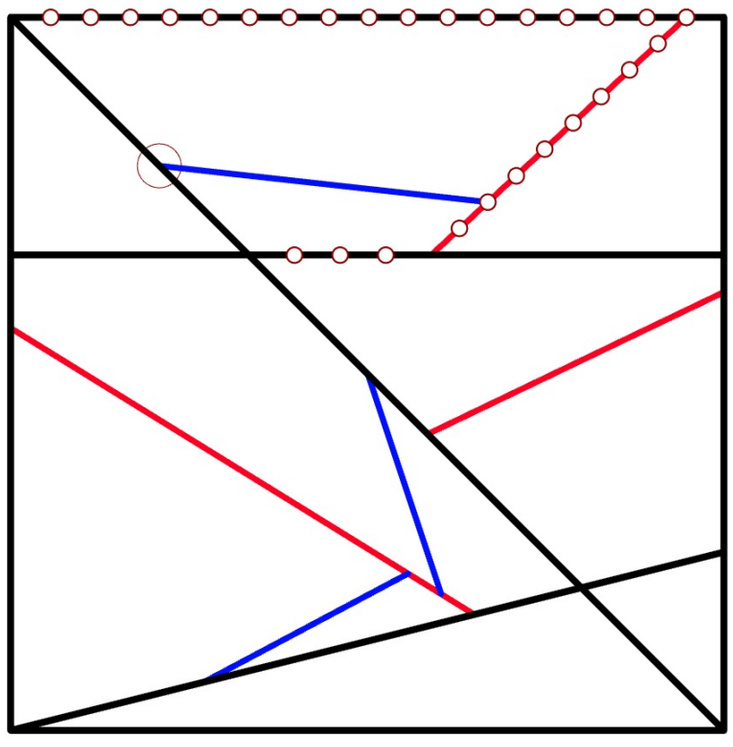


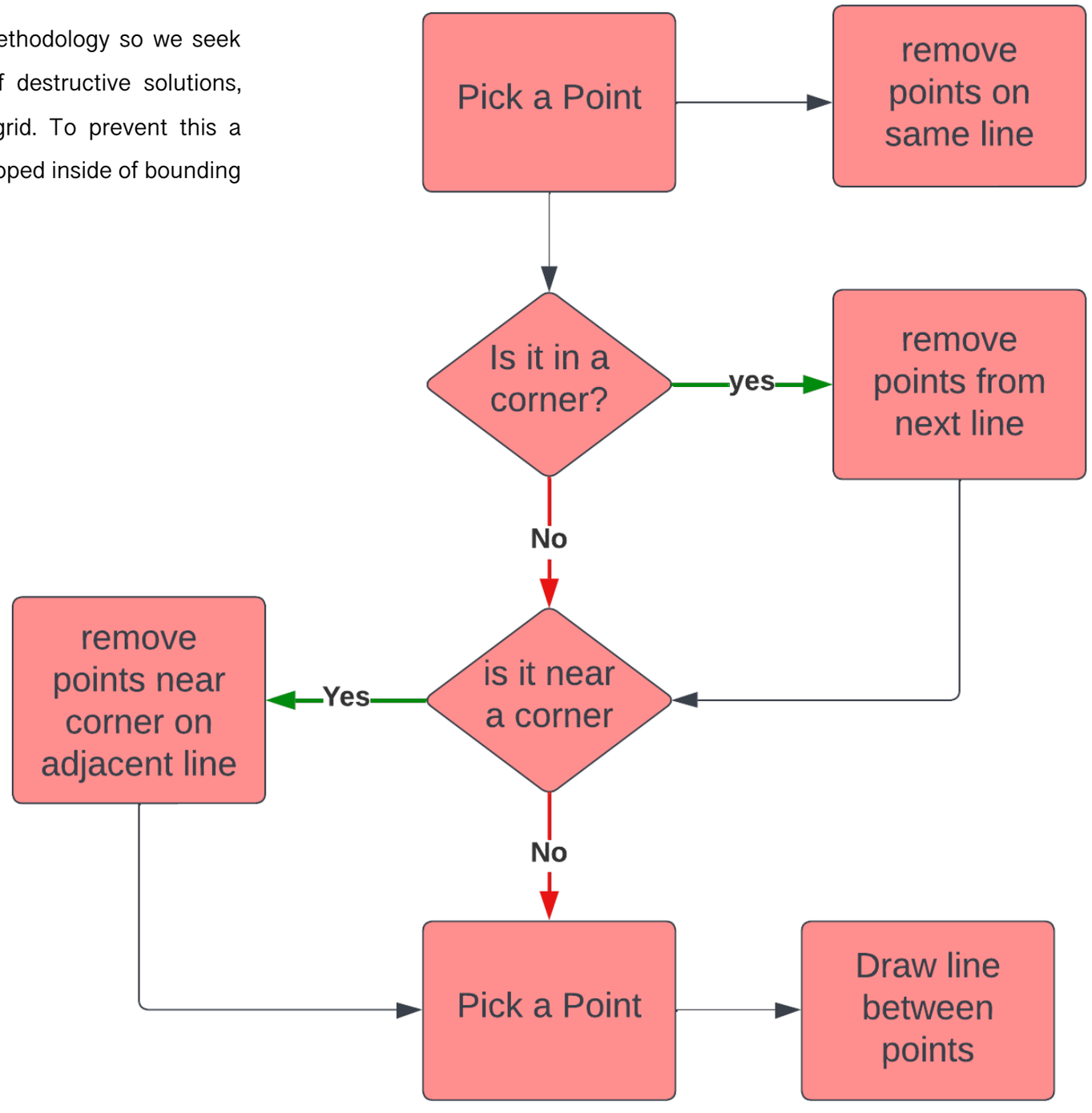


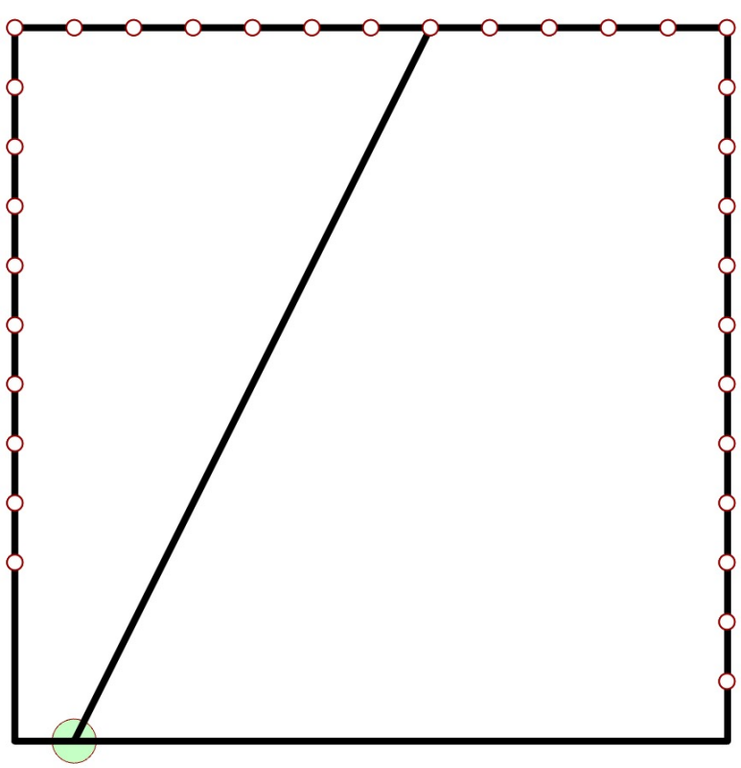


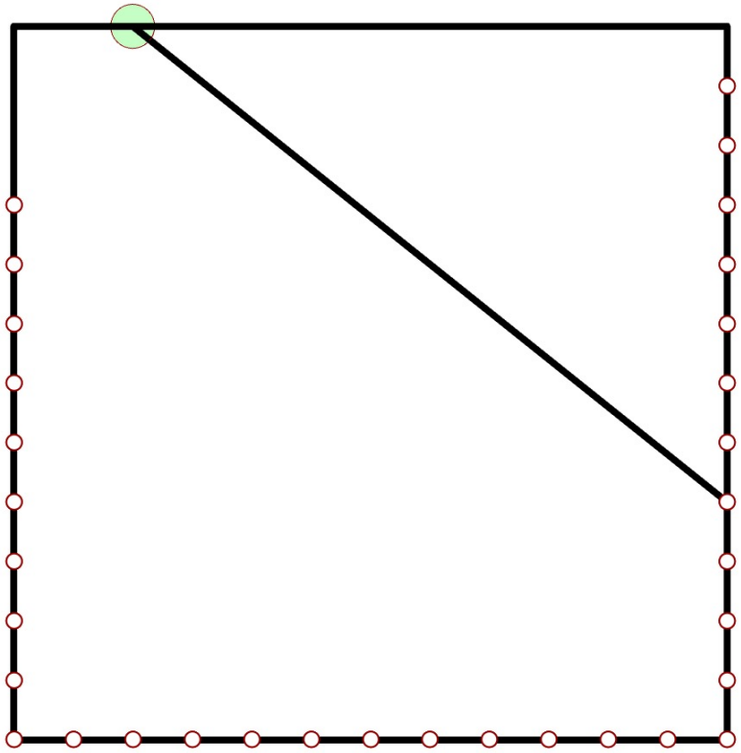
Control

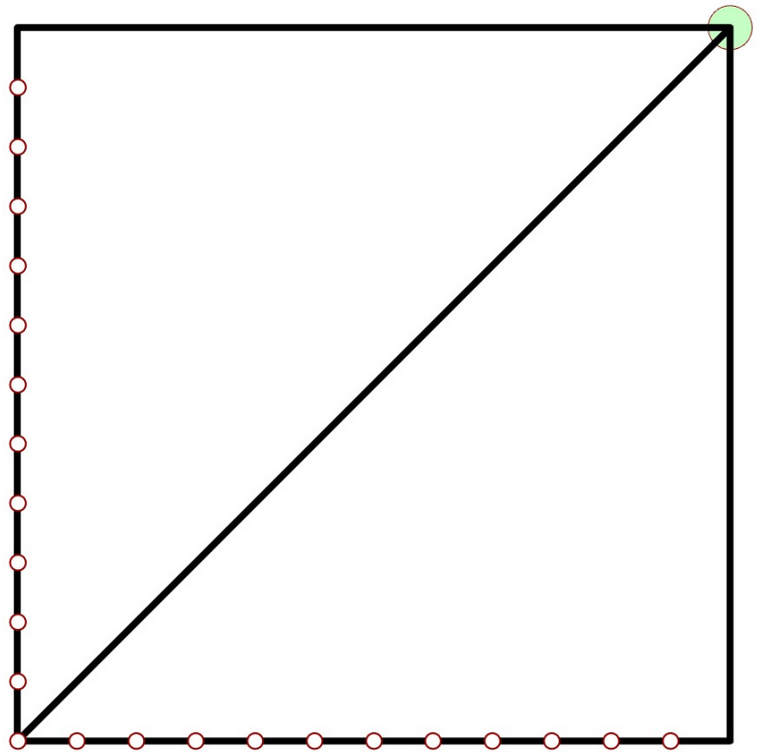
Allowing generative freedom is a major part of this designs methodology so we seek minimal control. Although complete freedom can lead to self destructive solutions, solutions in which no lines are present, representing a null grid. To prevent this a simplistic yet robust logic system ensures that every line is developed inside of bounding lines. The logic flow is as follows:

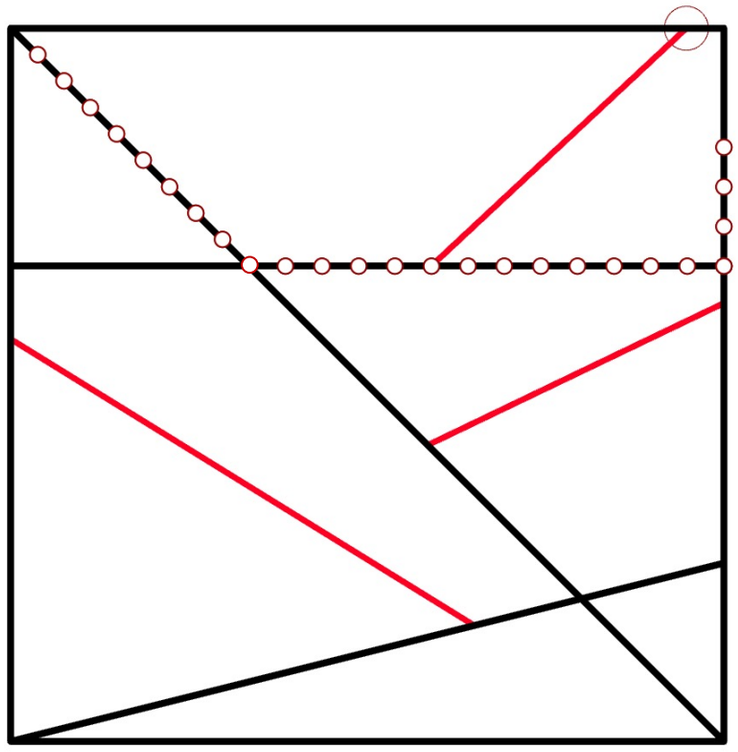


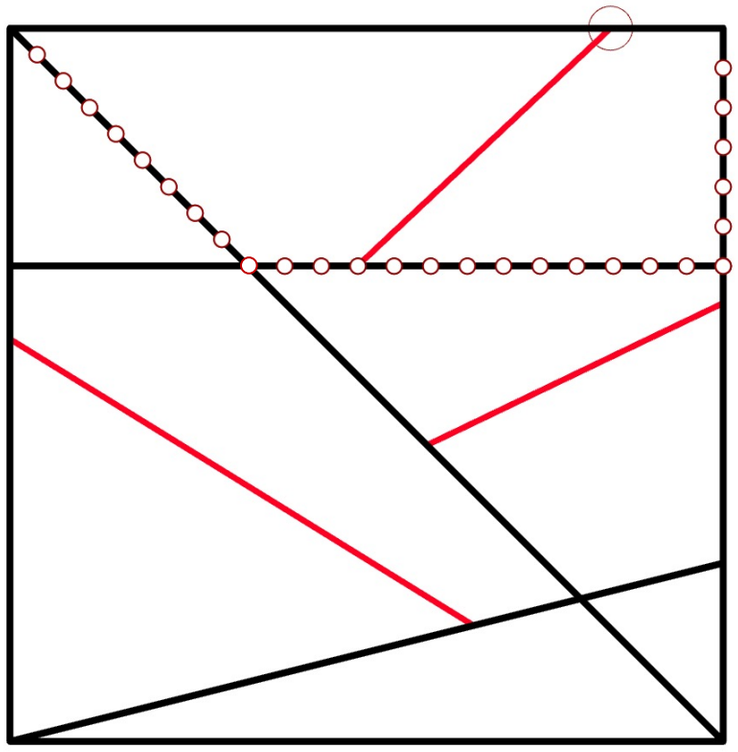


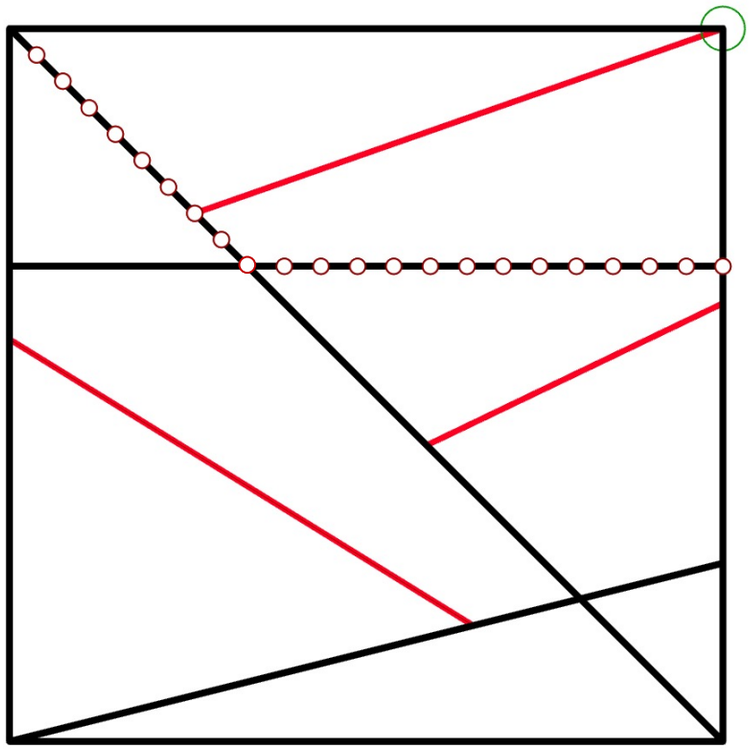




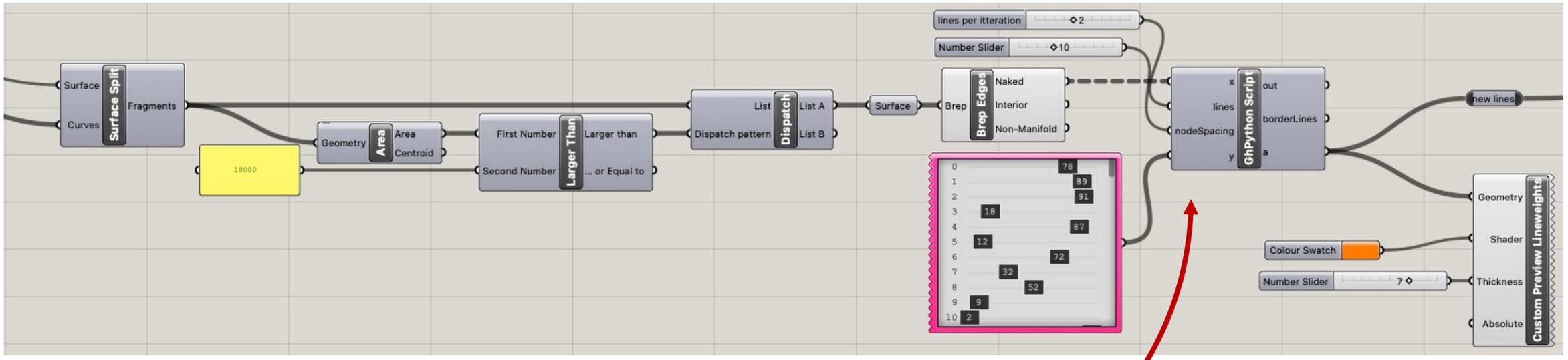








```python
__author__ = "felixmallinder"

import ghpythonlib.treehelpers as th
import Rhino.Geometry as geo
import scriptcontext
import rhinoscriptsyntax as rs

def makeNewList(p):
    for i in range(0,b.Count):

        if p in b[i]:
            ext = b[i].Count
            removePoints(b[i]) #remove all in same line

            #remove 0 loc in next line
            if i == b.Count-1:
                int = 0
            else: int = i + 1
            pointZero = b[int][0]
            removeList.append(pointZero)

            if p == b[i][0]:
            #if in corner remove line before

                if i == 0:
                    int = b.Count -1
                else: int = i -1
                removePoints(b[int])

            elif p == b[i][1] or p == b[i][2]:
            # if in first 2 remove last and second last from previous list

                if i == 0:
                    int = b.Count -1
                else: int = i - 1

                ext = b[int].Count  #needs new extent as line before

                pointExt1 = b[int][ext-1]
                pointExt2 = b[int][ext-2]
                removeList.append(pointExt1)
                removeList.append(pointExt2)

            elif p == b[i][ext-1] or p == b[i][ext-2]:
            # if in last 2 remove 0th first and second from next list

                if i == b.Count-1:
                    int = 0
                else: int = i + 1

                pointf0 = b[int][0]
                pointf1 = b[int][1]
                pointf2 = b[int][2]
                removeList.append(pointf0)
                removeList.append(pointf1)
                removeList.append(pointf2)

    #makes new list of points to pick from
    for value in removeList:
        if value in newList:
            newList.remove(value)

    return newList

def removePoints(points):
    for point in points:
        removeList.append(point)

#----------run----------------

genePick = 0
lineList = []

#---create matrix of points----

for i in range(x.BranchCount):
    branchList = x.Branch(i)

    allPoints = []
    b = []

    for j in range(branchList.Count):
        s = str(branchList[j])
        length = rs.CurveLength(s)

        if length > nodeSpacing*2:
            subList = []
            points = rs.DivideCurve(s,length/nodeSpacing) #divide line


            for p in points: #puts points of each line into matrix
                subList.append(p)
                allPoints.append(p)

            subList.pop()
            allPoints.pop()
            b.append(subList)

    linepoints = []

```



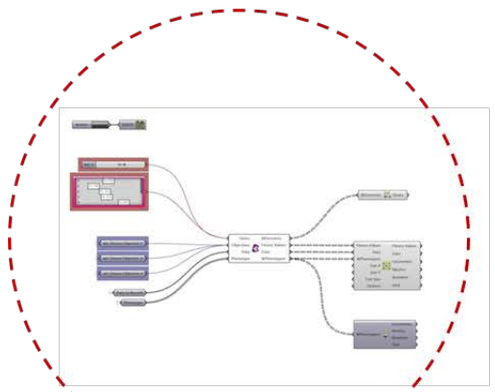

```
    # ---- for number of lines ----
    for k in range(0,lines):
        removeList  = []
        newList = list(allPoints)

        #pick first point using gene pool and all points
        pickPoint = y[genePick]
        genePick = genePick +1
        remapV = round((pickPoint*(len(allPoints)-1))/100) #remaps gene pool to list length
        point1 = newList[int(remapV)]

        #make new list of points, removing corners, same line etc
        newList = makeNewList(point1)

        #pick first point using gene pool and reduced points
        pickPoint = y[genePick]
        genePick = genePick +1

        remapV = round((pickPoint*(len(newList)))/100) #100 is genepool maximum {0,100}
        print int(remapV), newList, point1
        point2 = newList[int(remapV)]

        #create line from both points
        start = geo.Point3d(point1)
        end = geo.Point3d(point2)
        line = rs.AddLine(start, end)
        lineList.append(line)


a = lineList
borderLines = allPoints
```

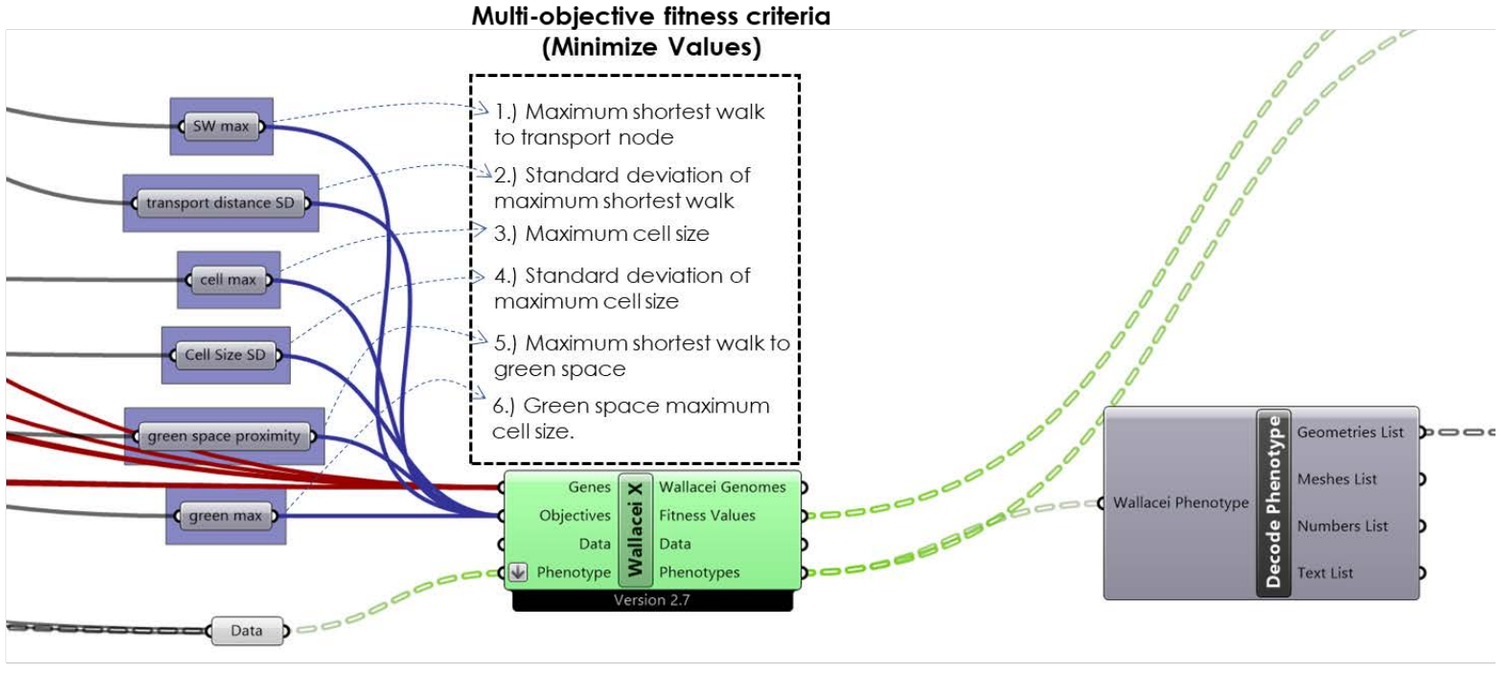
This block of grasshopper is repeated in series for recursion of nested lines. The ‘new lines’ output fed into the curves input of ‘surface split’ in the next generation and the surface input is always the Masdar plot.

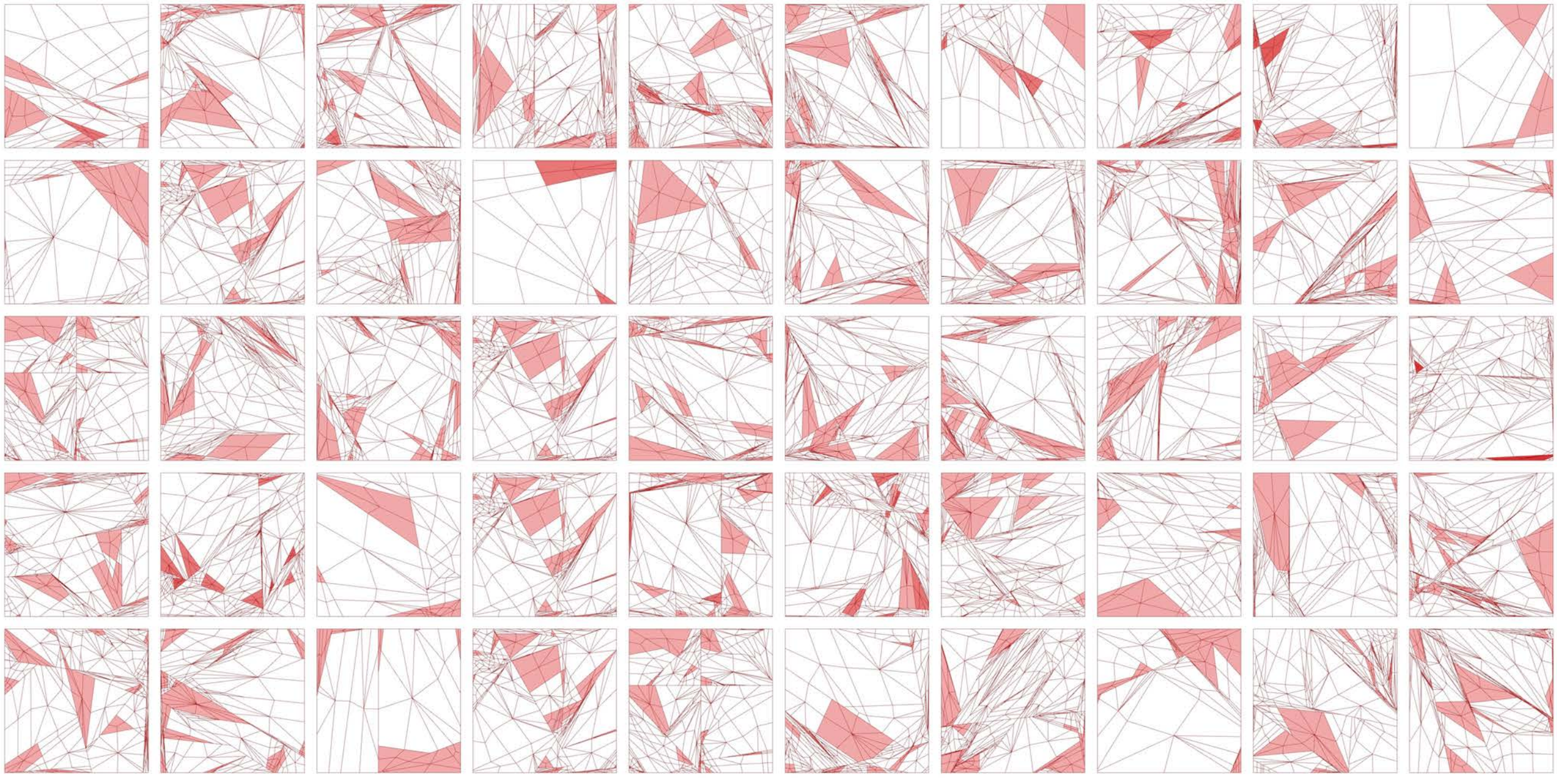
Due to processing power limitations, the current code uses a maximum of 4 recursions. Similarly, the code and computing power limits the generative freedom – the ‘lines per iteration’ slider isn’t independently chosen for each cell in a subdivision. Ie. The 2nd generation cells will all have 2 dividing lines in this case. For a more organic design, complete freedom would programmed. The final mode uses a 4-3-2-1 design, 4 lines for the first recursion, down to 1 line for the final recursion.

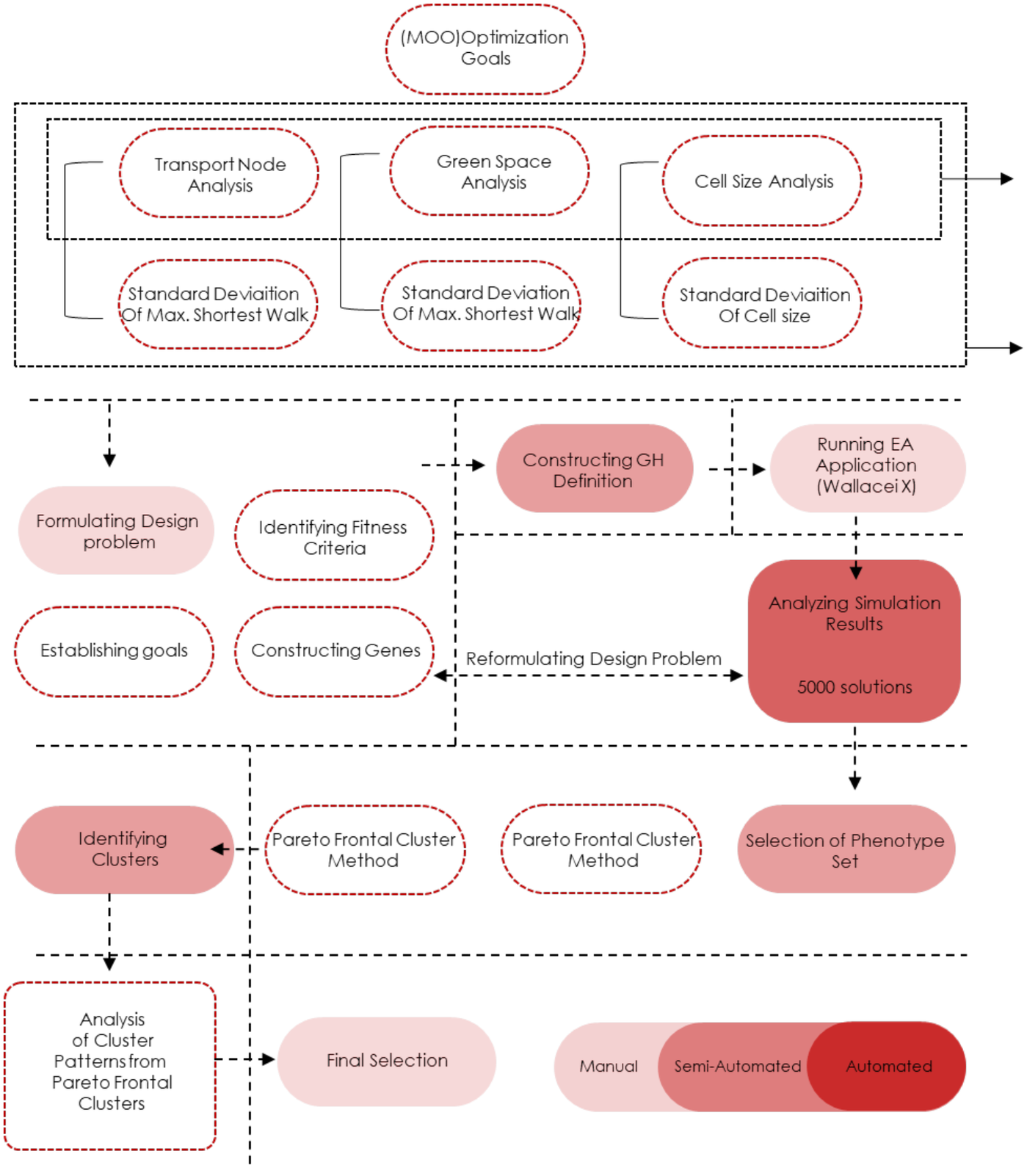
3.5. Multi Objective Optimisation

A multi-objective optimization problem is one in which it is not possible to optimize all objective functions simultaneously, so a Pareto optimal or non-dominated solution is one where no objective function can be improved without worsening the others. Multi-objective optimization was used in exploring the performance of our organic grid network with respect to three objectives: the shortest walk to transport nodes, the shortest walk to green open spaces and the maximum cell size. We considered two different multi-objective optimization tools for this study: Octopus, which uses the Strength Pareto Evolutionary Algorithm 2 (SPEA-2), and fast hypervolume-based many objective optimization algorithm (HypE); and Wallacei X, which uses the Nondominated Sorting Genetic Algorithm II (NSGA-II). We ultimately chose to use Wallacei X because it does not depend on initial solutions to converge to optimal solutions, has good computational time, and is less prone to getting stuck in suboptimal solutions. Additionally, the Wallacei interface and data access capabilities provided us with more flexibility in our analysis. Wallacei X optimization works with minimization formula; it tries to bring all fitness values towards zero, In order to maximize the fitness values, an inverse of the number will need to be computed before it is fed to the Wallacei X engine.









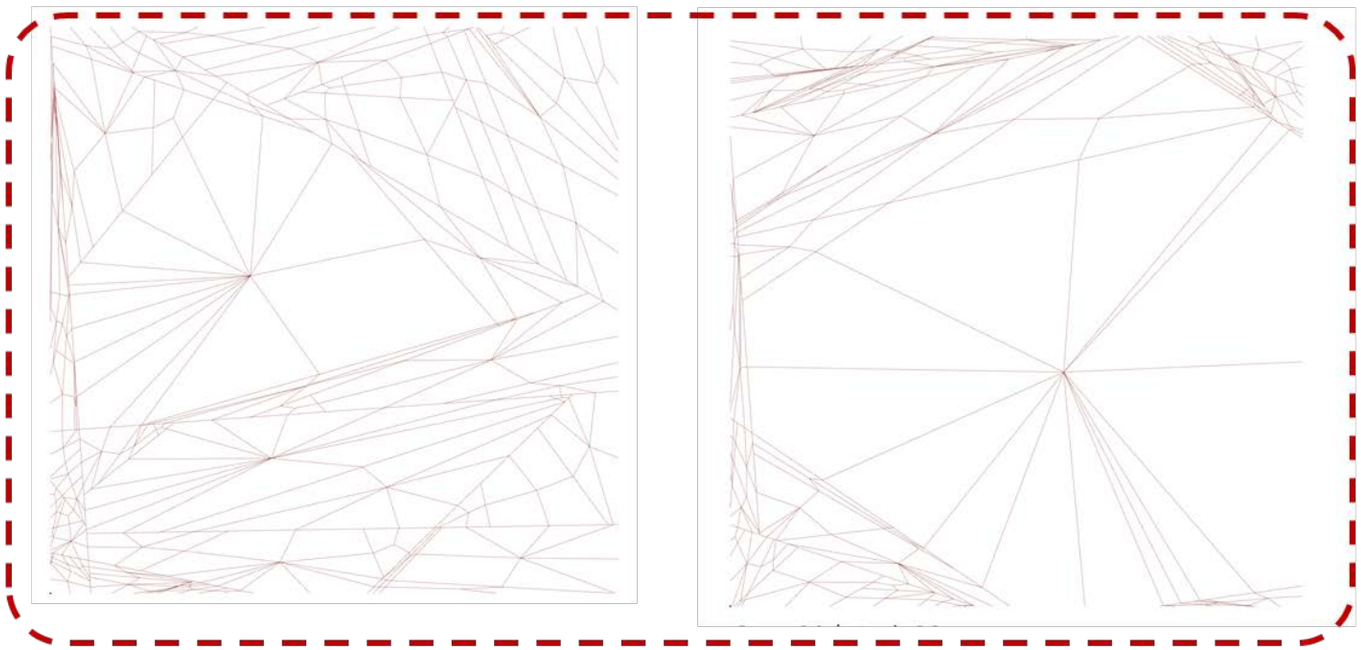
(MO Optimization 1)

The first optimization ran for 2:00hrs. on observation of results, the street network seemed to crash within itself and generated Pareto frontal solutions with Large and very small cells. This was not a desirable result.

(MO Optimization 2)

The second optimization ran for 2:30hrs and took into consideration the standard deviation values. The standard deviation of the fitness values were derived and minimized for each fitness goal. This helped us maintain an even distribution in the proportion of cell size and a good average distribution of the shortest walk for transport nodes and green spaces.

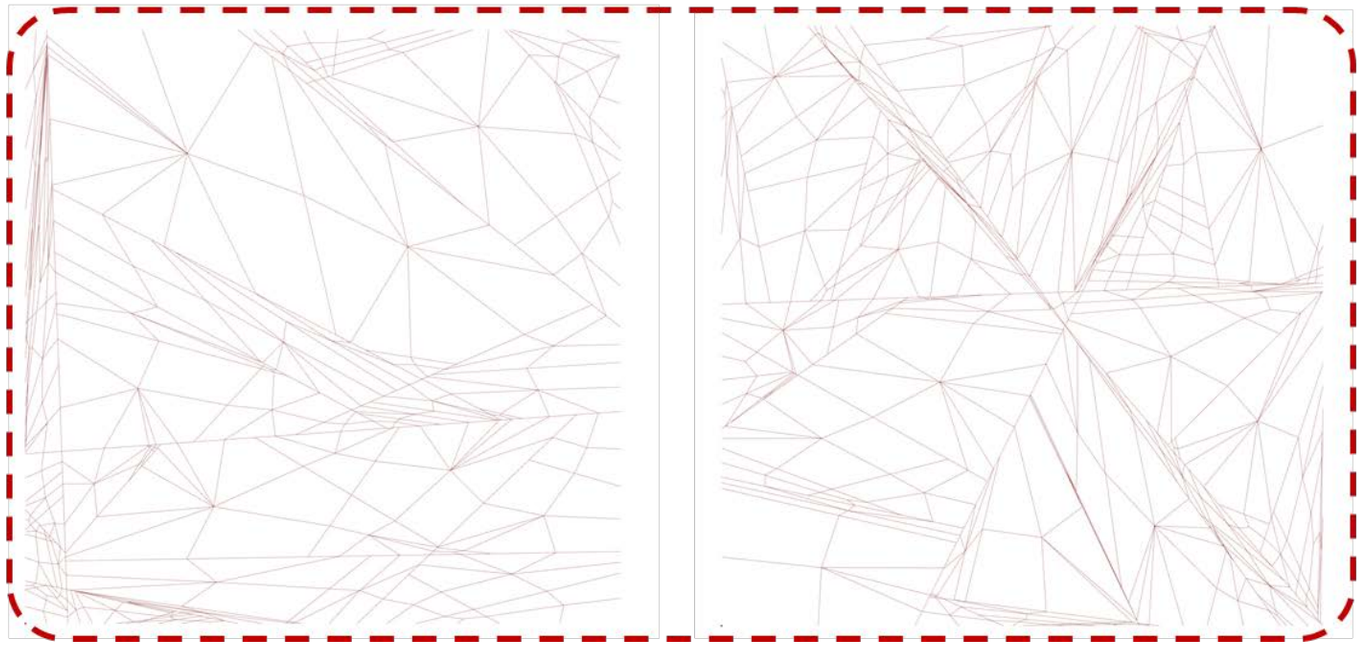
MO Optimization 1 Phenotypes (Fitness Values)



Disproportionate Large cell

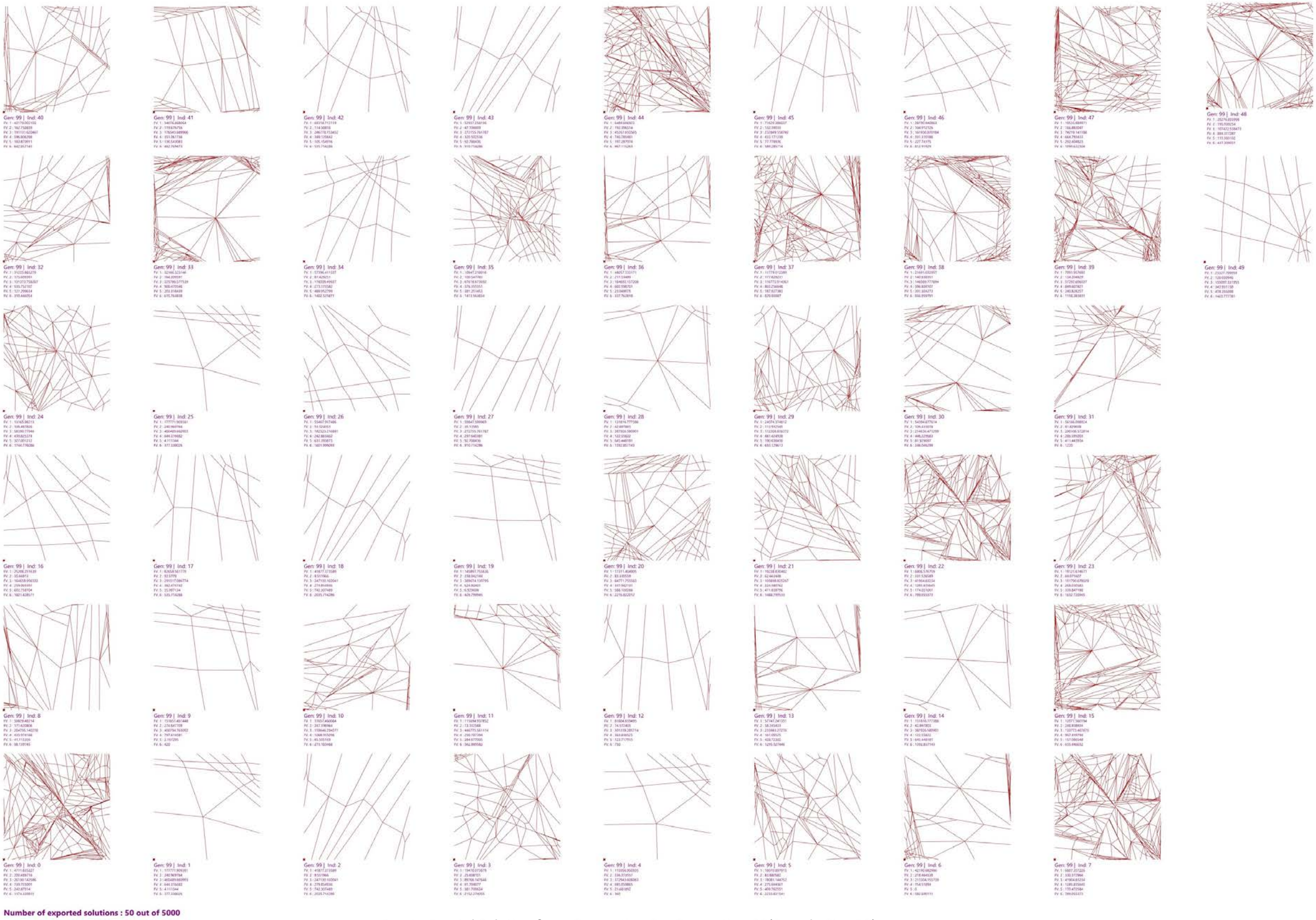
Results indicated a concentration of Large cells at the center of the neighborhood and a spread of smaller cell on the borders. This was not a desirable result as with a street network we desire to have a more even distribution of cell blocks.

MO Optimization 2 Phenotypes (Fitness + Standard deviation of Fitness Values)



Compact Organic Grid system

On the addition of standard deviation of fitness values as a multi-objective fitness criteria, the results of the organic grid system were mostly compact and well distributed cells were achieved which was acceptable for further analysis

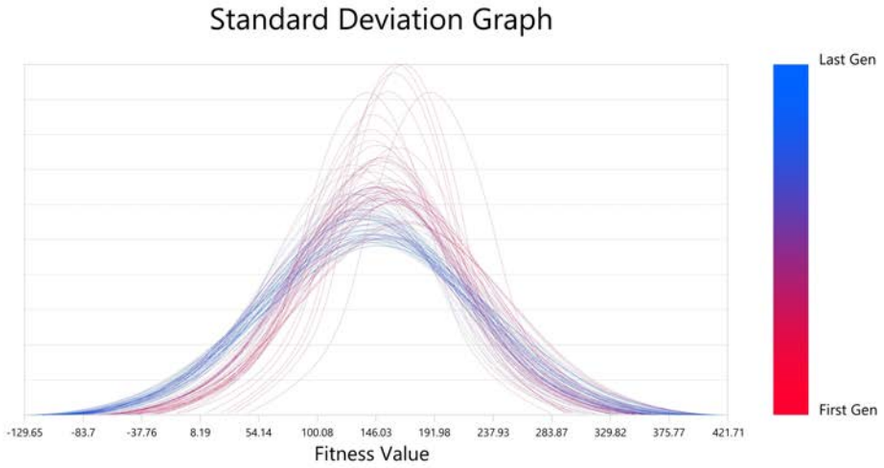


Number of exported solutions : 50 out of 5000

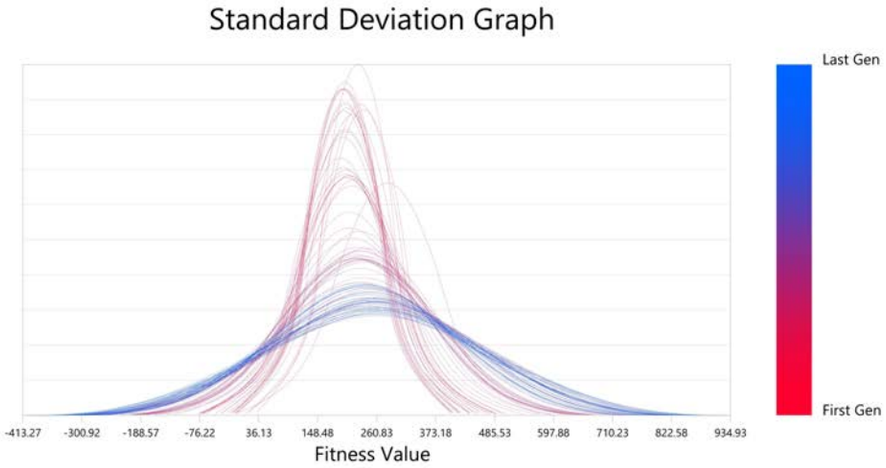
Pareto Frontal Solution from Generation 0 – Generation 99(Sample Size:50)

4. Results

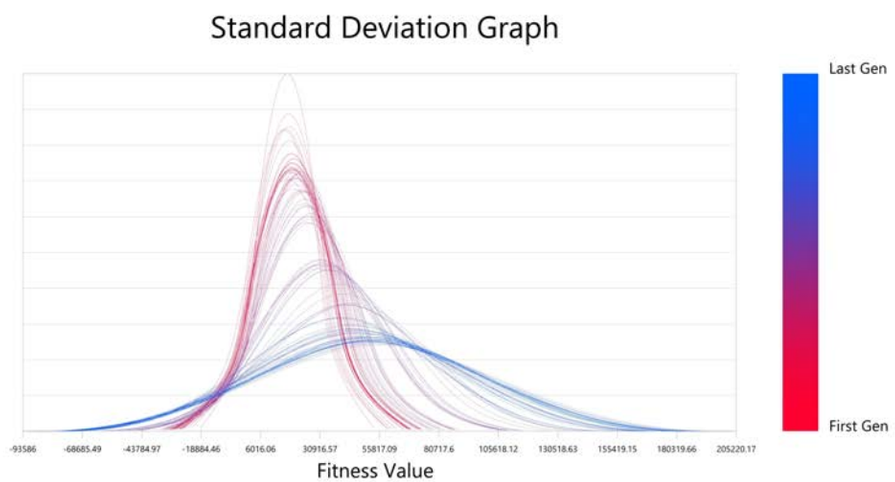
The standard deviation represents the distribution of a set of values from the mean. A low standard deviation factor indicates that most values are clustered around the mean (less variation within the population), while a high standard deviation factor indicates that the values are spread out farther from the mean (more variation within the population). The aim of the chart is to present and analyse the levels of variation/convergence for each generation in the population, as well as whether the generations are getting fitter throughout the simulation. Increased variation is represented through a 'flat ' curve, while increased convergence is represented through a 'narrow' curve. A shift in the curve to the left indicates better mean performance.



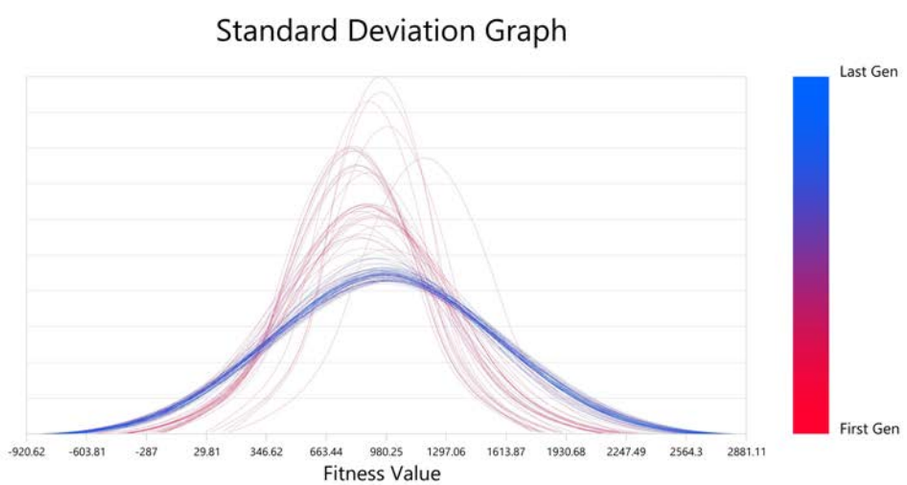
Maximum Shortest Walk to Transport Node



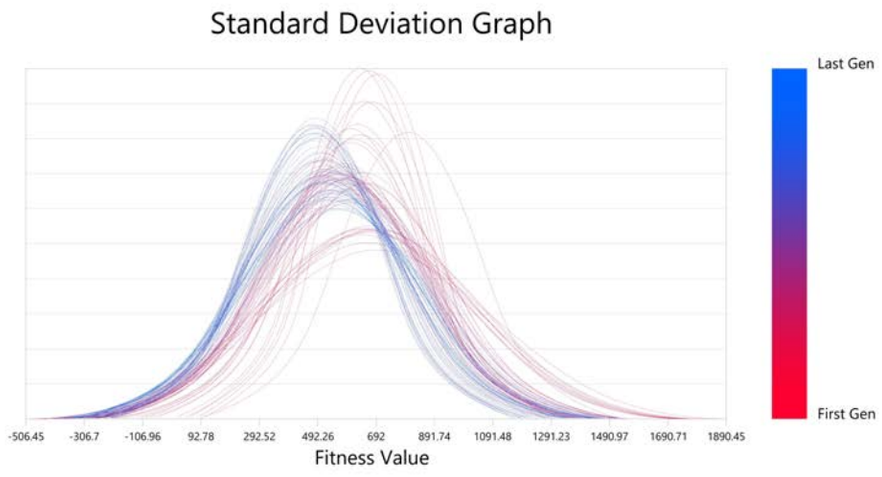
Maximum Shortest Walk to Green Space



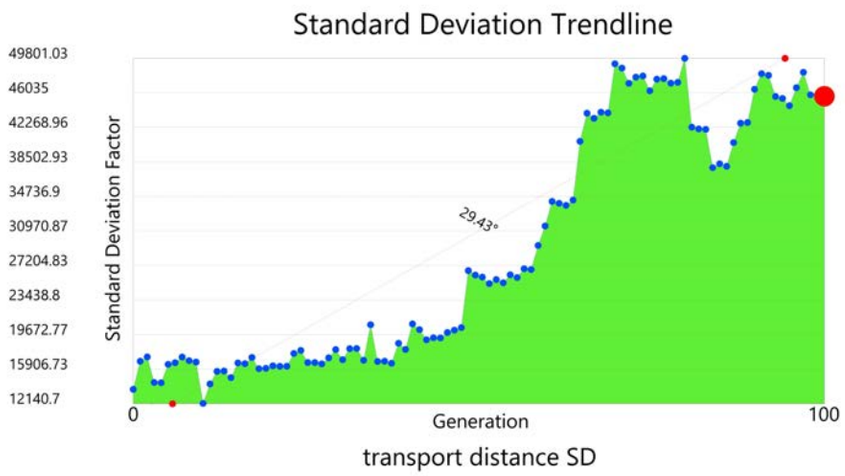
Standard Deviation of Maximum shortest walk to Transport Node

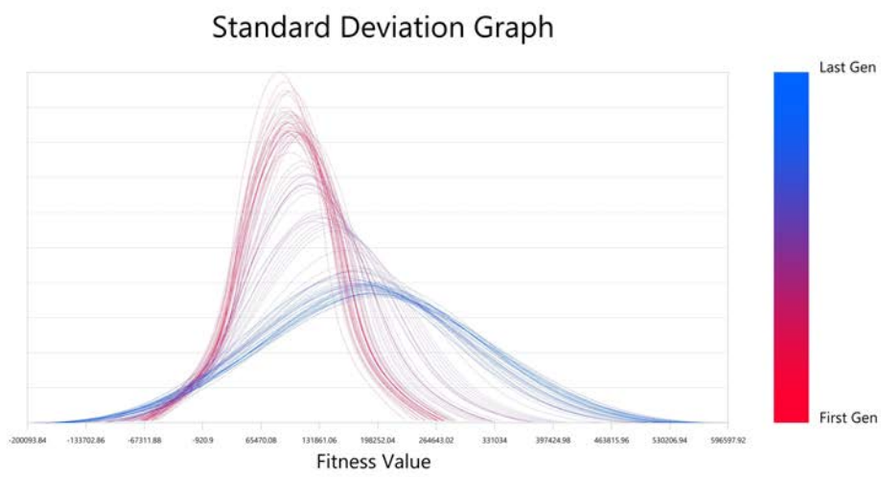


Standard Deviation of Maximum Shortest Walk to Green Space

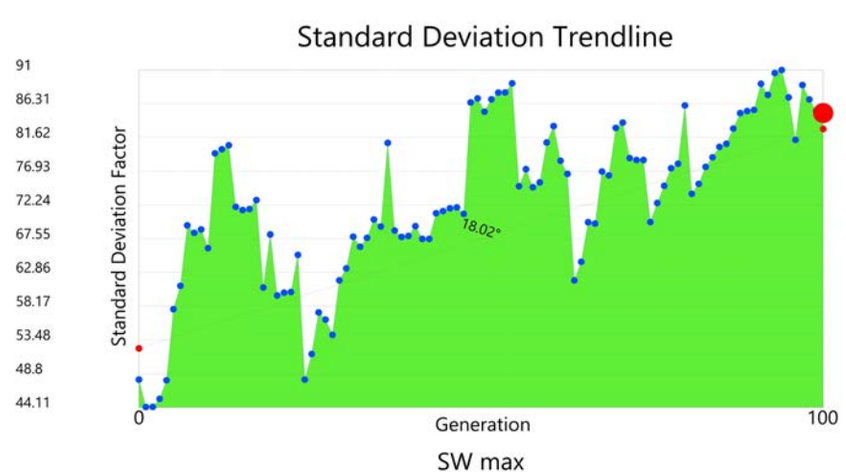


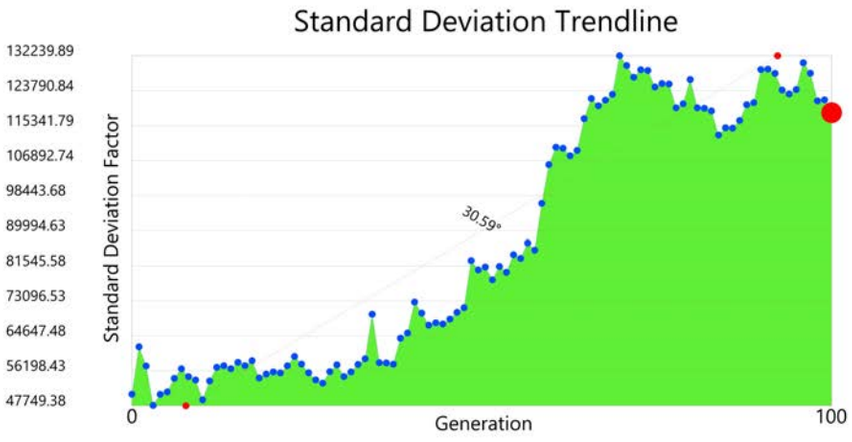
Maximum Cell Size



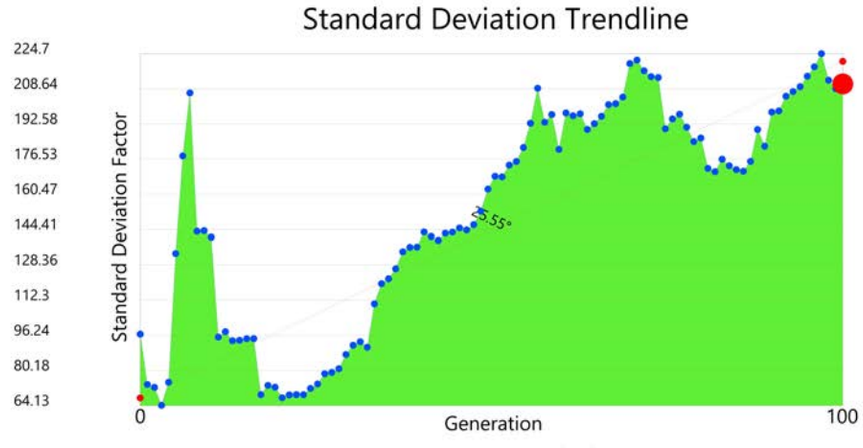


Standard Deviation of Maximum Cell Size

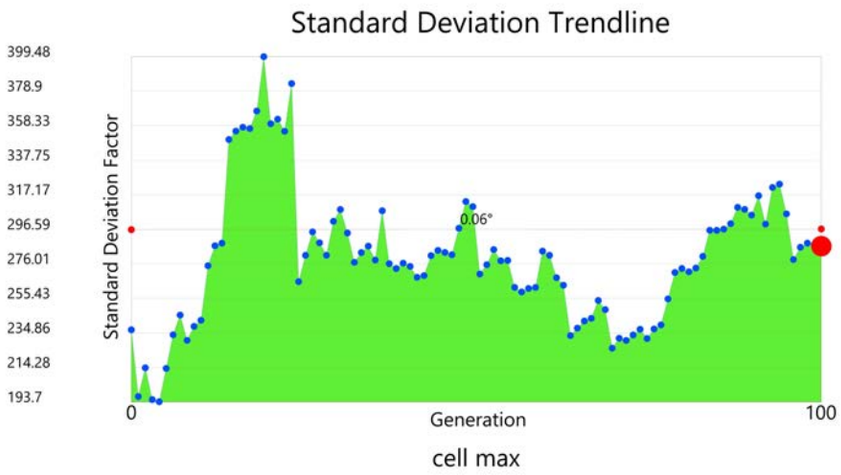




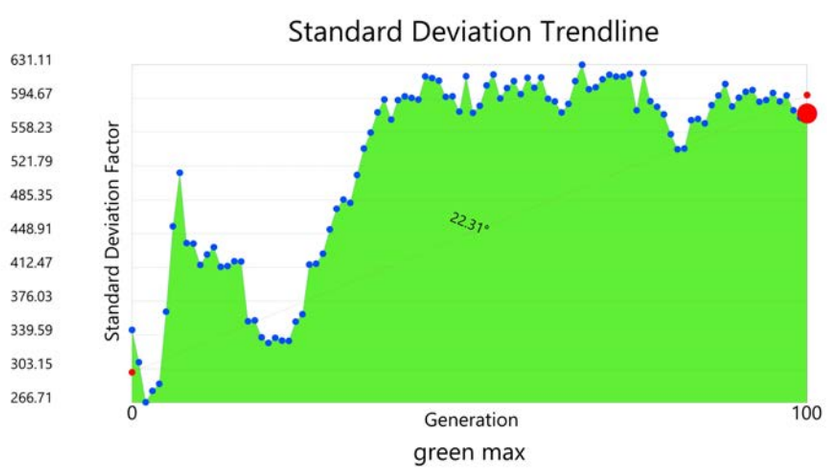
Cell Size SD



green space proximity



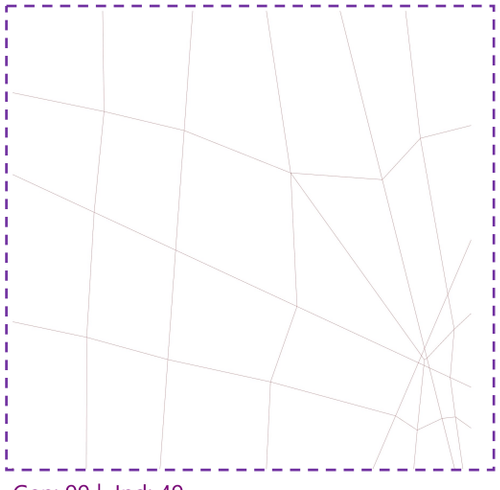
cell max



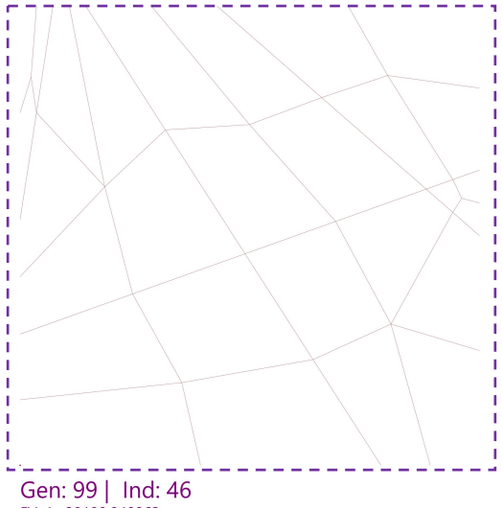
green max

Gen: 99 | Ind: 34
FV. 1 : 57396.411337
FV. 2 : 87.429251
FV. 3 : 178309.49927
FV. 4 : 273.173582
FV. 5 : 489.952799
FV. 6 : 1402.525871

Gen: 99 | Ind: 26
FV. 1 : 55467.917486
FV. 2 : 53.324053
FV. 3 : 182323.216891
FV. 4 : 242.883402
FV. 5 : 631.393873
FV. 6 : 1601.999093



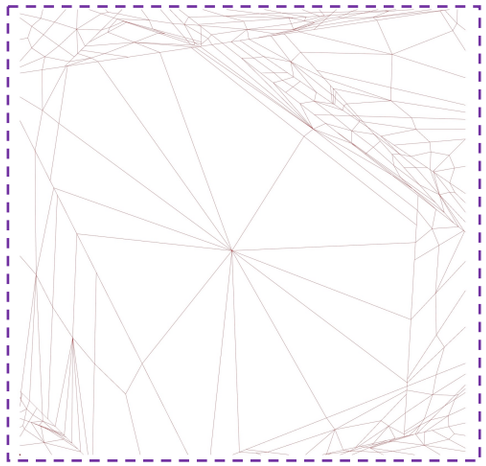
Gen: 99 | Ind: 49
FV. 1 : 23327.709959
FV. 2 : 126.030946
FV. 3 : 155097.331955
FV. 4 : 343.931158
FV. 5 : 478.355098
FV. 6 : 1463.777781



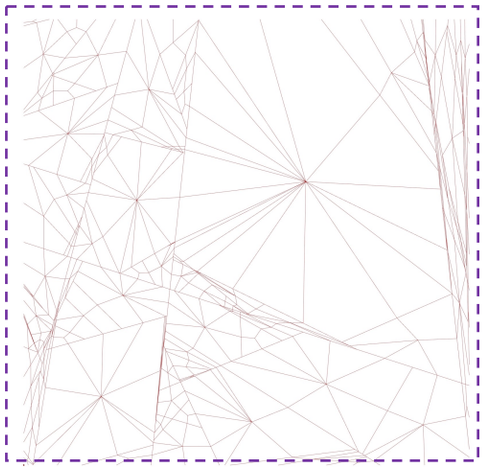
Gen: 99 | Ind: 46
FV. 1 : 28190.940863
FV. 2 : 104.912126
FV. 3 : 161936.870184
FV. 4 : 331.310186
FV. 5 : 227.74175
FV. 6 : 812.91929

Minimal Grid Pattern

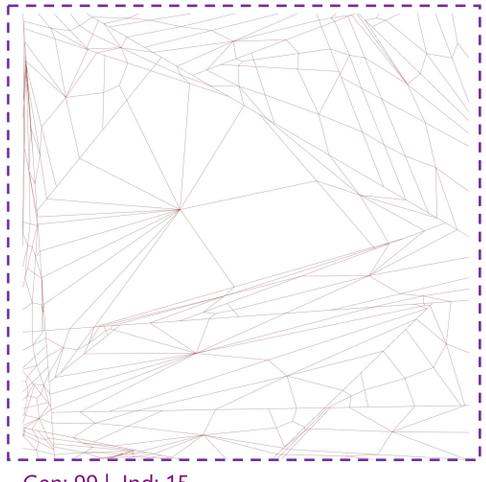
The minimal grid evolves because of two occurrences; the first scenario occurs when the lines picked by the algorithm at the third stage are in the same exact position, as a result no further division is possible by the cell system. The second scenario happens when the lines flow parallel and are non-intersecting resulting in less subdivisions. This limitations were an oversight from the rules developed by the street network generator, but due to our aim of exploring a truly organic system without over-control, we allowed the system to continue to explore these iterations.



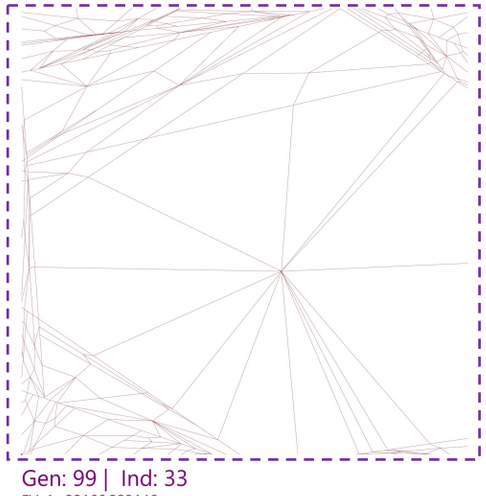
Gen: 99 | Ind: 38
FV. 1 : 21691.032957
FV. 2 : 140.838351
FV. 3 : 146569.777894
FV. 4 : 596.836107
FV. 5 : 201.304273
FV. 6 : 936.959791



Gen: 99 | Ind: 37
FV. 1 : 11779.012389
FV. 2 : 177.829231
FV. 3 : 116772.914361
FV. 4 : 803.256648
FV. 5 : 187.927383
FV. 6 : 829.85087



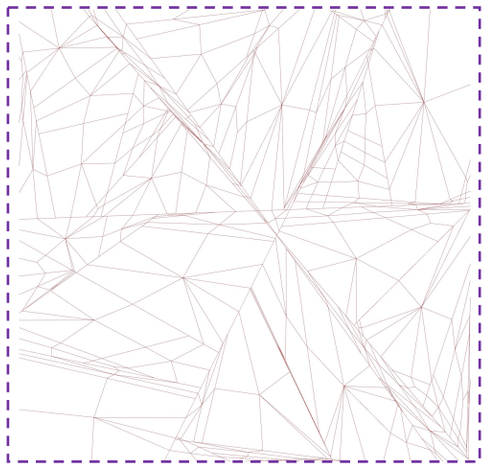
Gen: 99 | Ind: 15
FV. 1 : 12077.560194
FV. 2 : 248.938934
FV. 3 : 133773.467675
FV. 4 : 967.439794
FV. 5 : 157.086548
FV. 6 : 635.446652



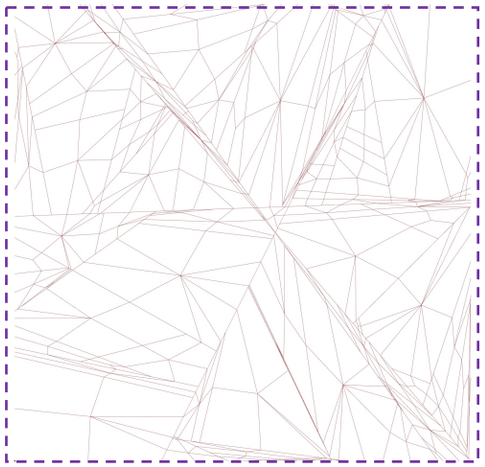
Gen: 99 | Ind: 33
FV. 1 : 32166.323146
FV. 2 : 164.209591
FV. 3 : 225786.577539
FV. 4 : 900.470546
FV. 5 : 203.318439
FV. 6 : 615.764838

Large Cell Phenomenon

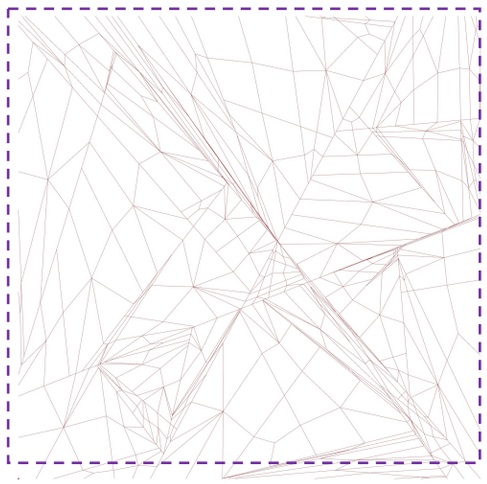
The Large cell pattern occurs when the genetic optimization identifies one large cell and a lot of minimal cells and tries to optimize the fitness values majorly for the large cell. This results in a cell which is a major outlier but overall good average results. The results are good for the large cells but not good for the entire cell divisions.



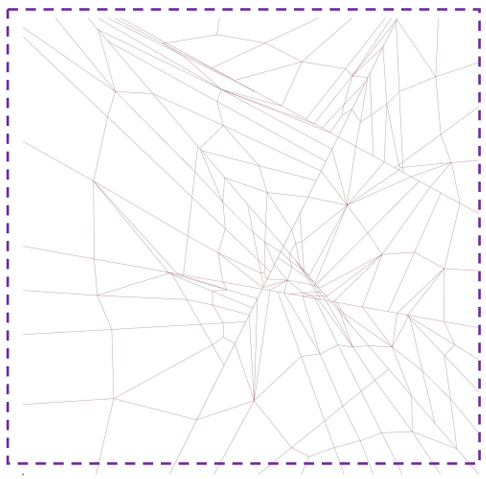
Gen: 99 | Ind: 7
FV. 1 : 6807.357226
FV. 2 : 330.372964
FV. 3 : 41904.83234
FV. 4 : 1285.835645
FV. 5 : 170.472584
FV. 6 : 789.055373



Gen: 99 | Ind: 22
FV. 1 : 6806.576759
FV. 2 : 331.526589
FV. 3 : 41904.83234
FV. 4 : 1285.835645
FV. 5 : 174.021061
FV. 6 : 789.055373



Gen: 99 | Ind: 0
FV. 1 : 4711.835227
FV. 2 : 200.489716
FV. 3 : 26130.142586
FV. 4 : 739.755001
FV. 5 : 242.87914
FV. 6 : 1174.330815

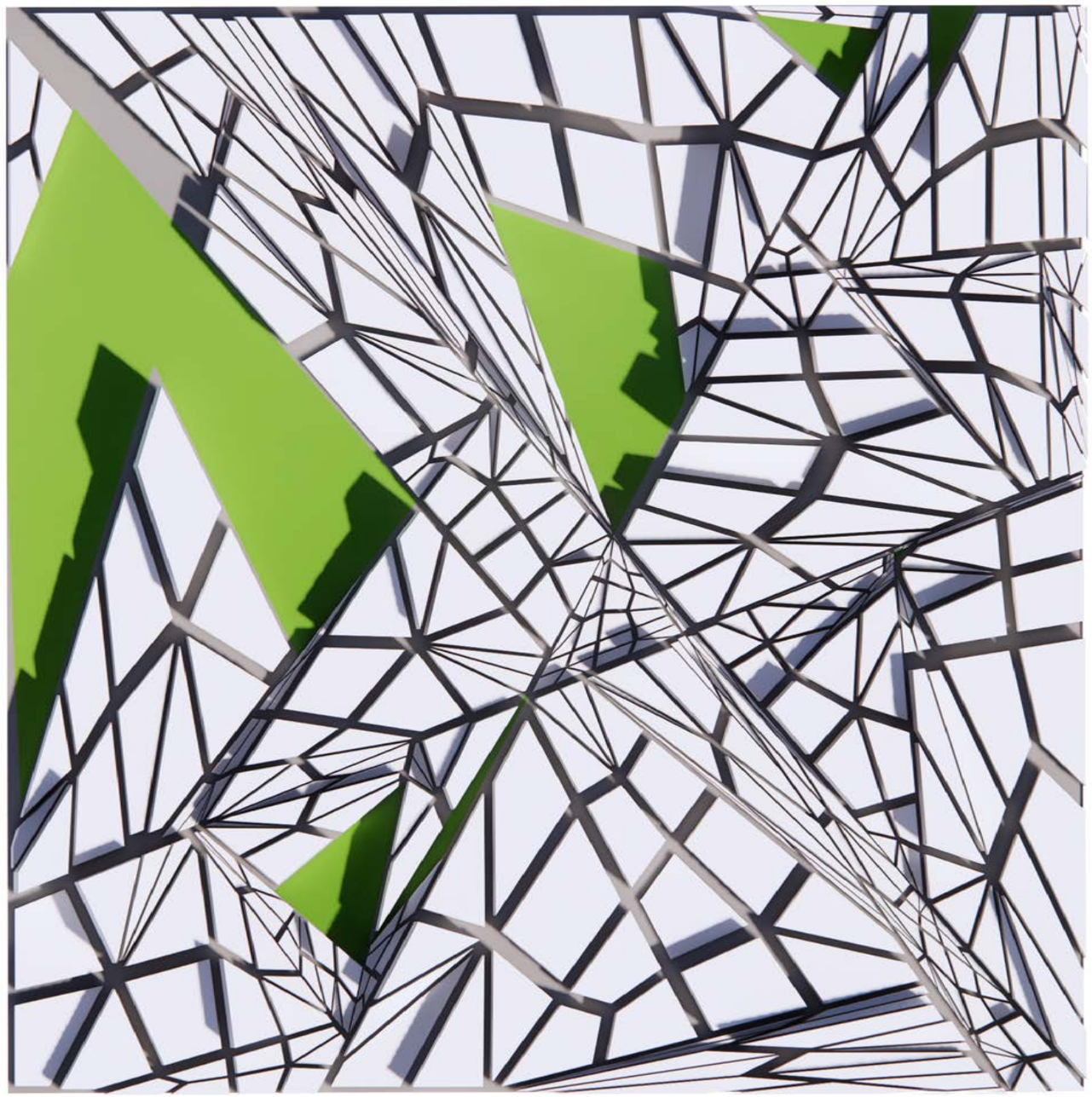


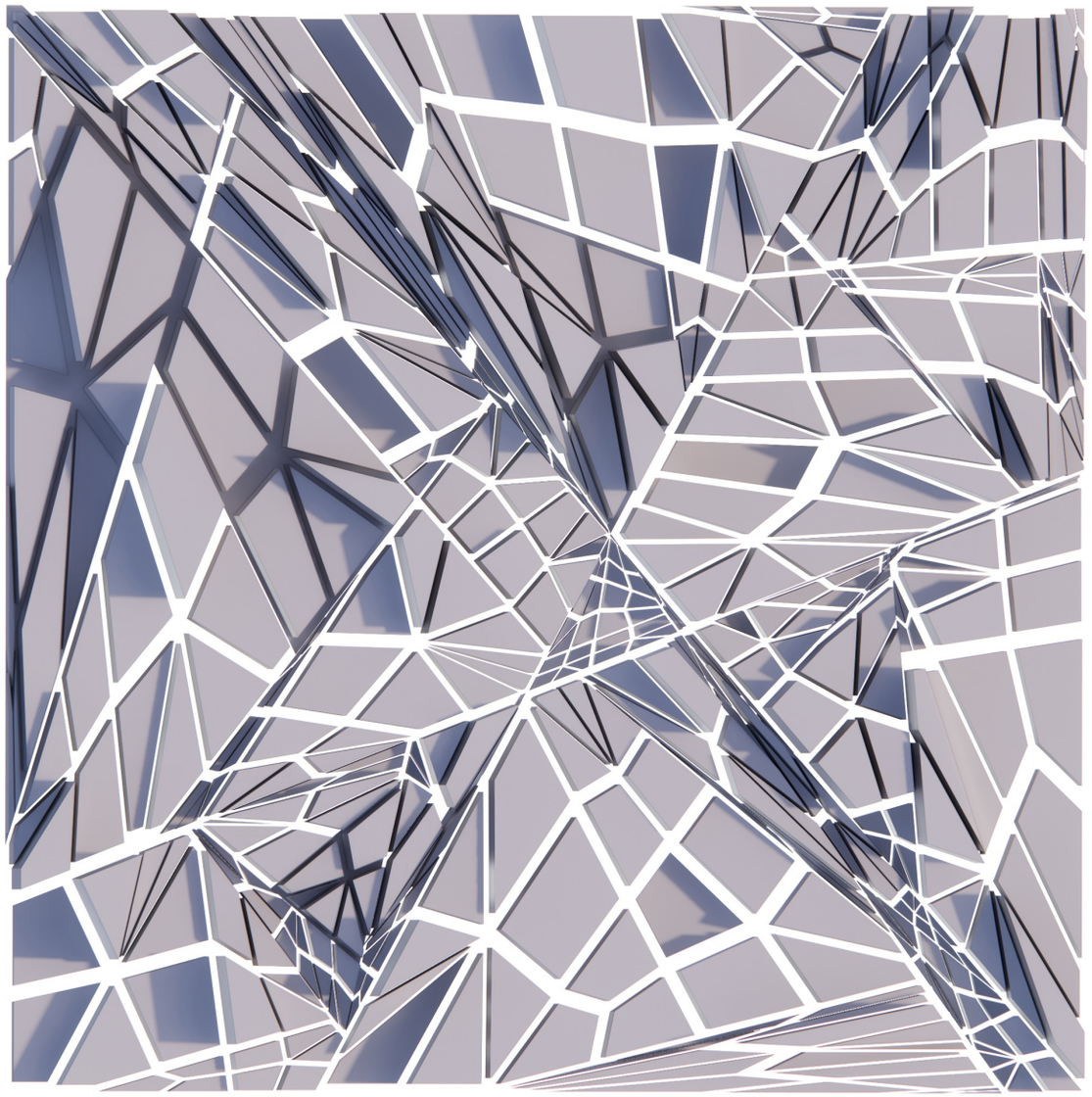
Gen: 99 | Ind: 35
FV. 1 : 10947.218916
FV. 2 : 100.547783
FV. 3 : 67618.673692
FV. 4 : 376.355551
FV. 5 : 381.251453
FV. 6 : 1413.563834

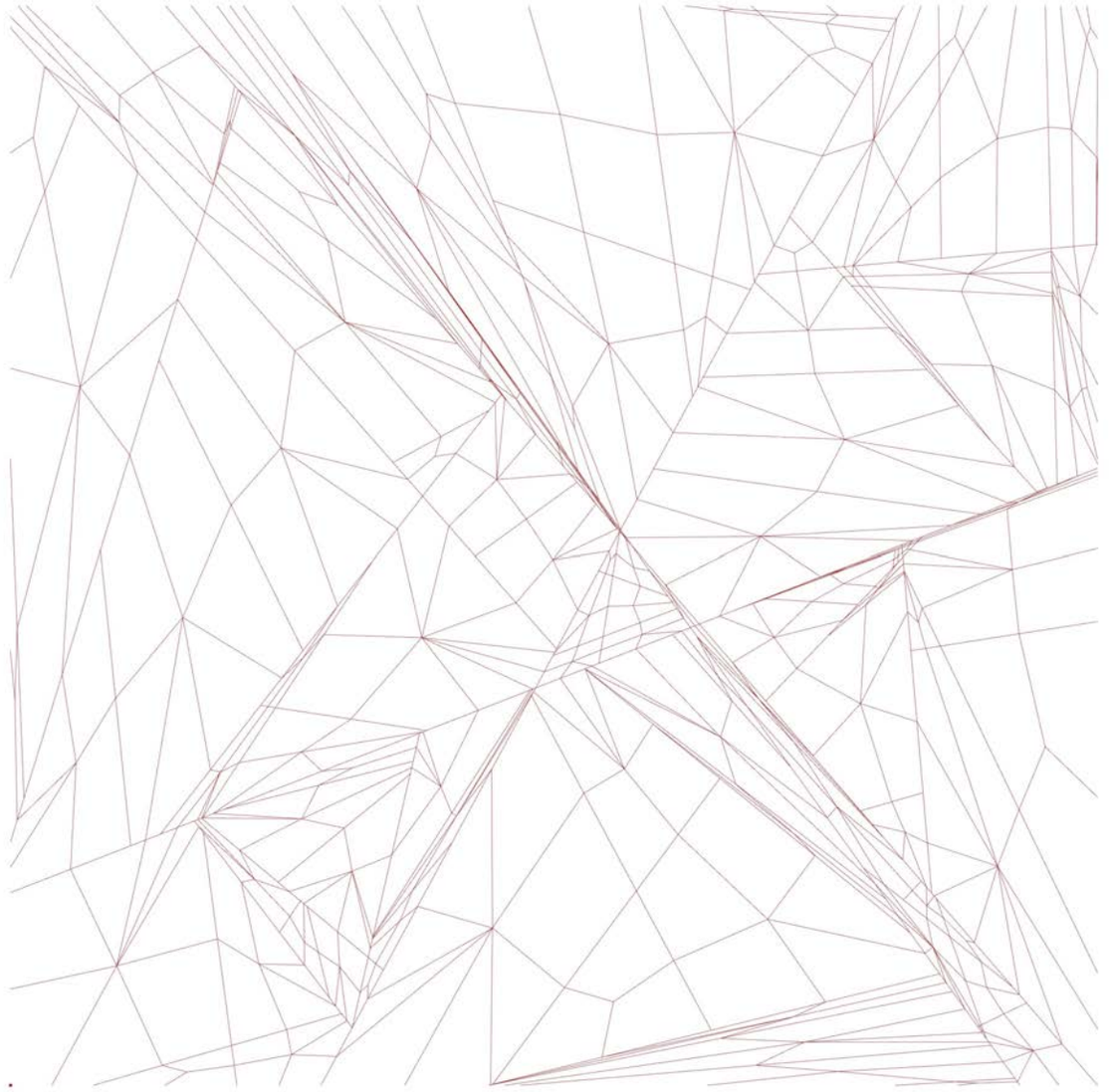
Dispersed Pattern

The dispersed pattern occurs when the cells are divided, and the area threshold allows for the third-degree division stage. The genetic optimization divides most cells into the third-degree division stage and optimizes the network based on each individual cell. This results in an even distribution of average walking distances for each cell and average similar area for each cell. This is the desirable result and would be discussed further in the section.

6. Final Solution







Gen: 99 | Ind: 0

FV. 1 : 4711.835227

FV. 2 : 200.489716

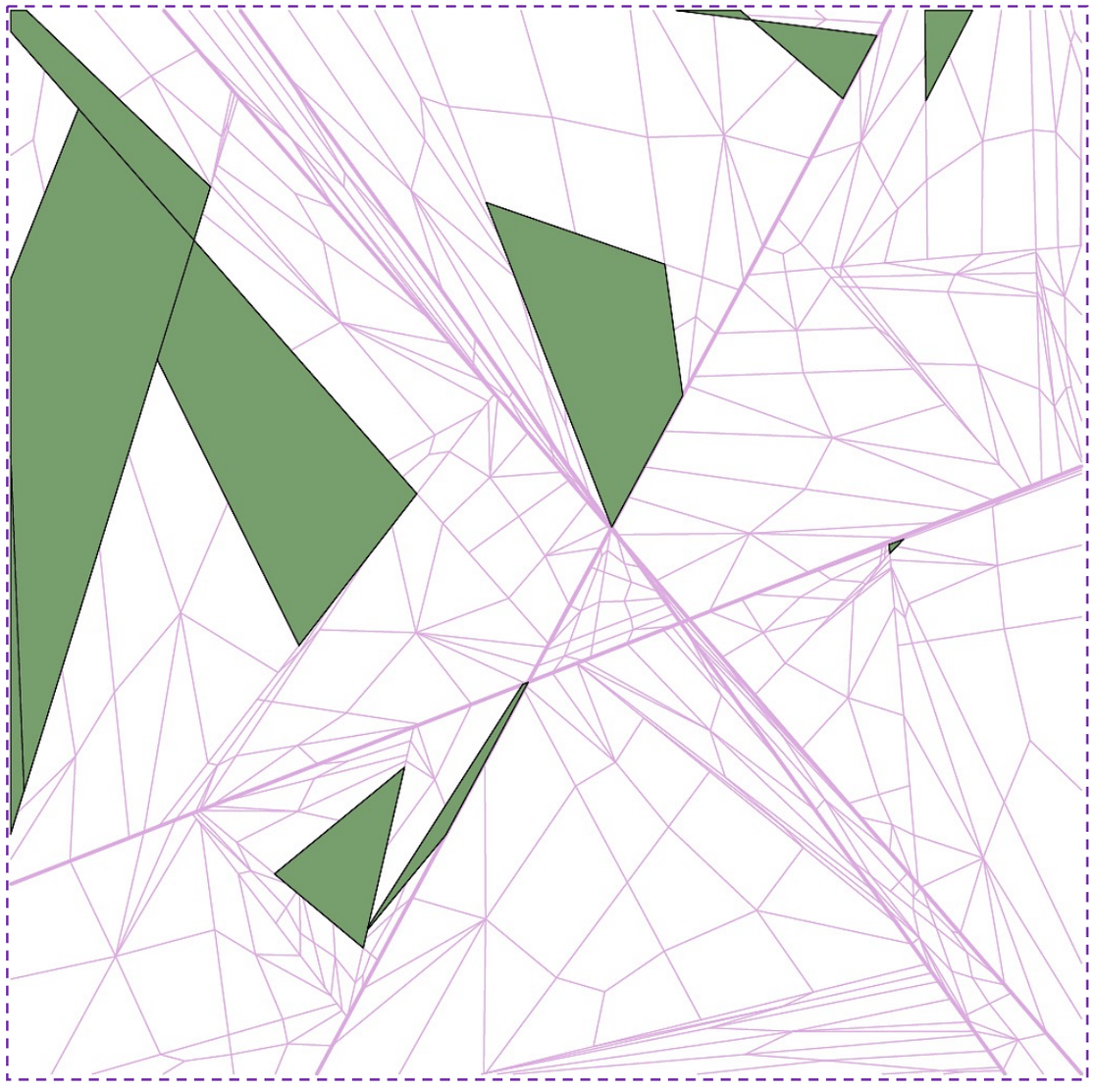
FV. 3 : 26130.142586

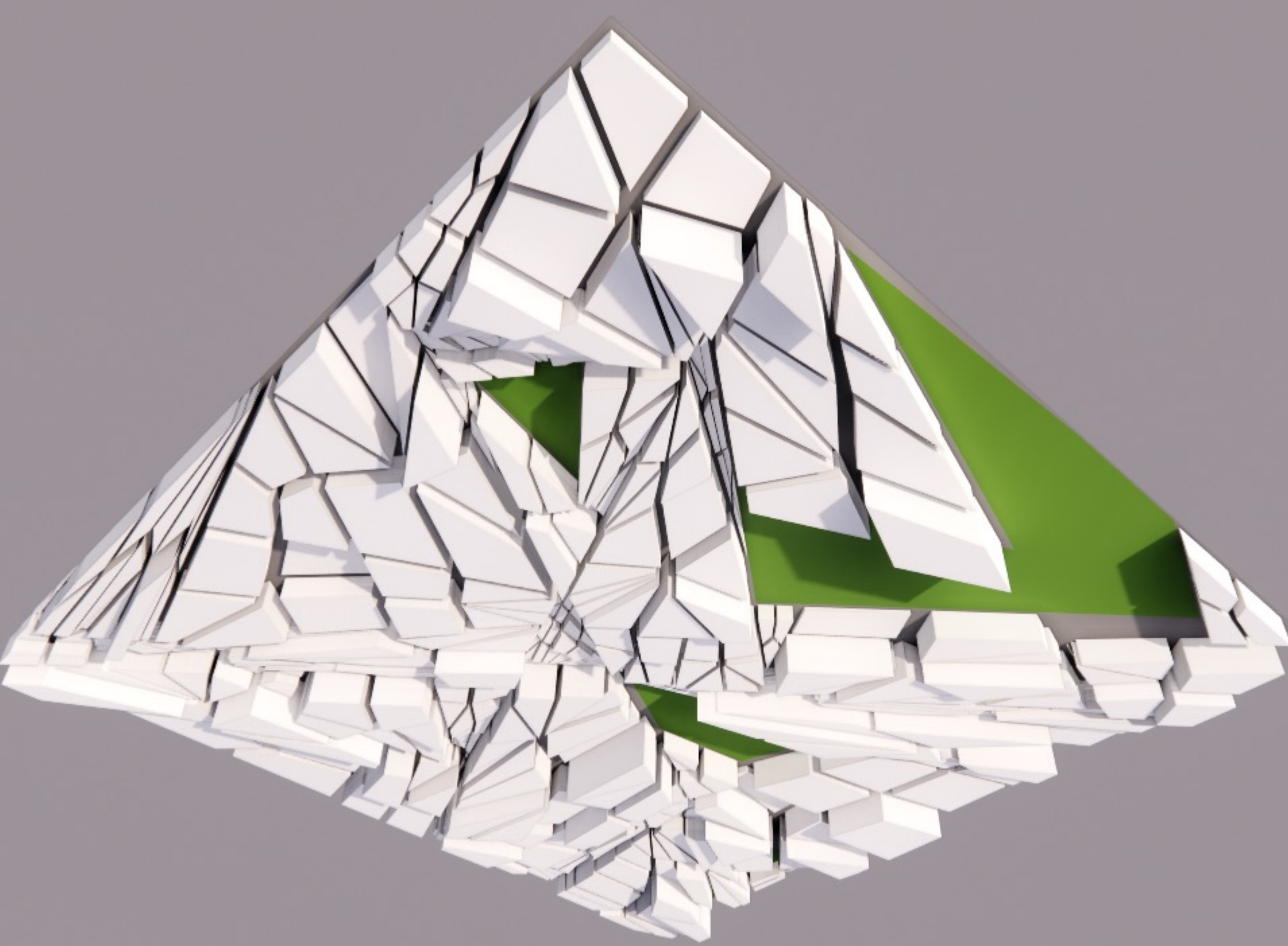
FV. 4 : 739.755001

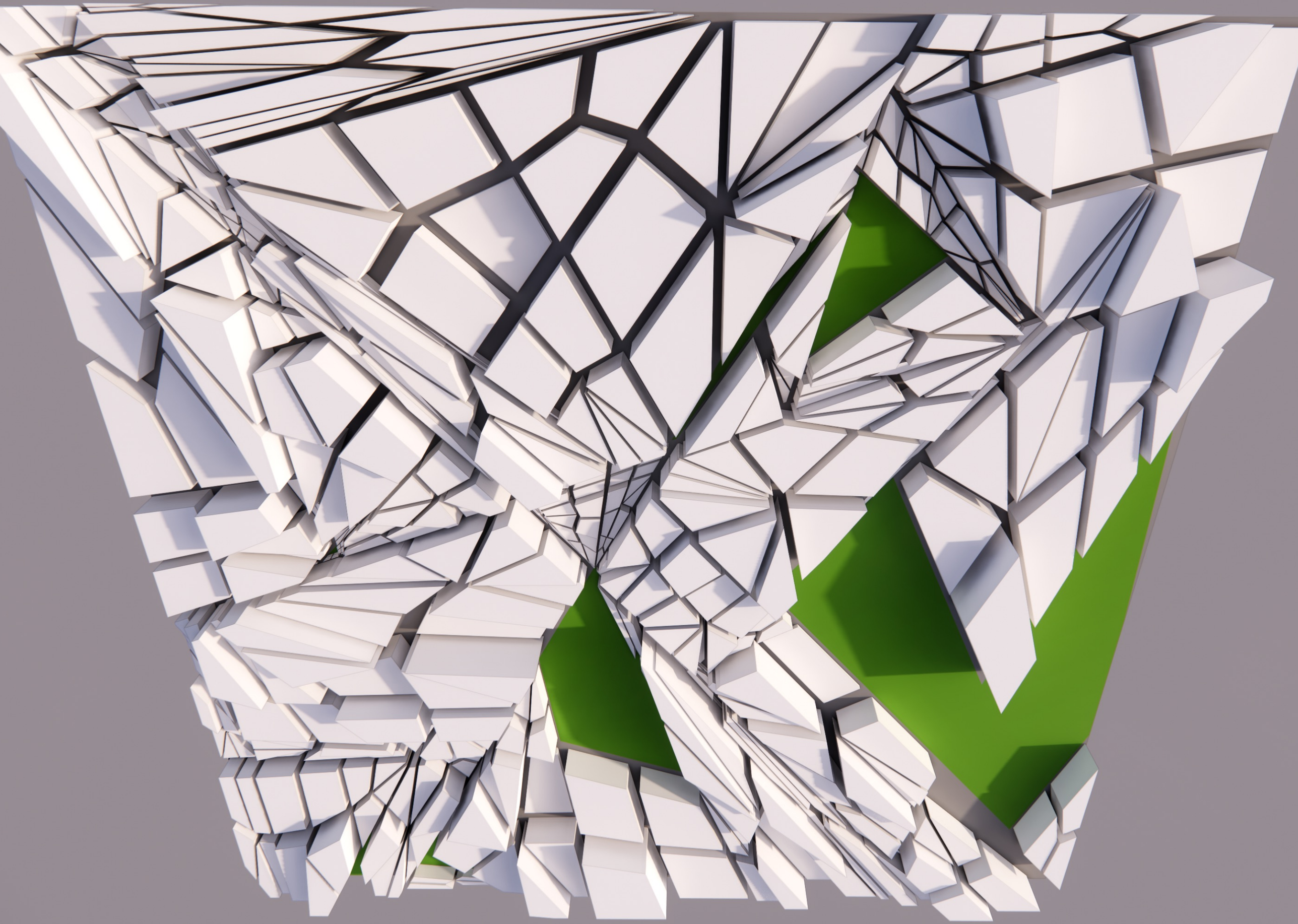
FV. 5 : 242.87914

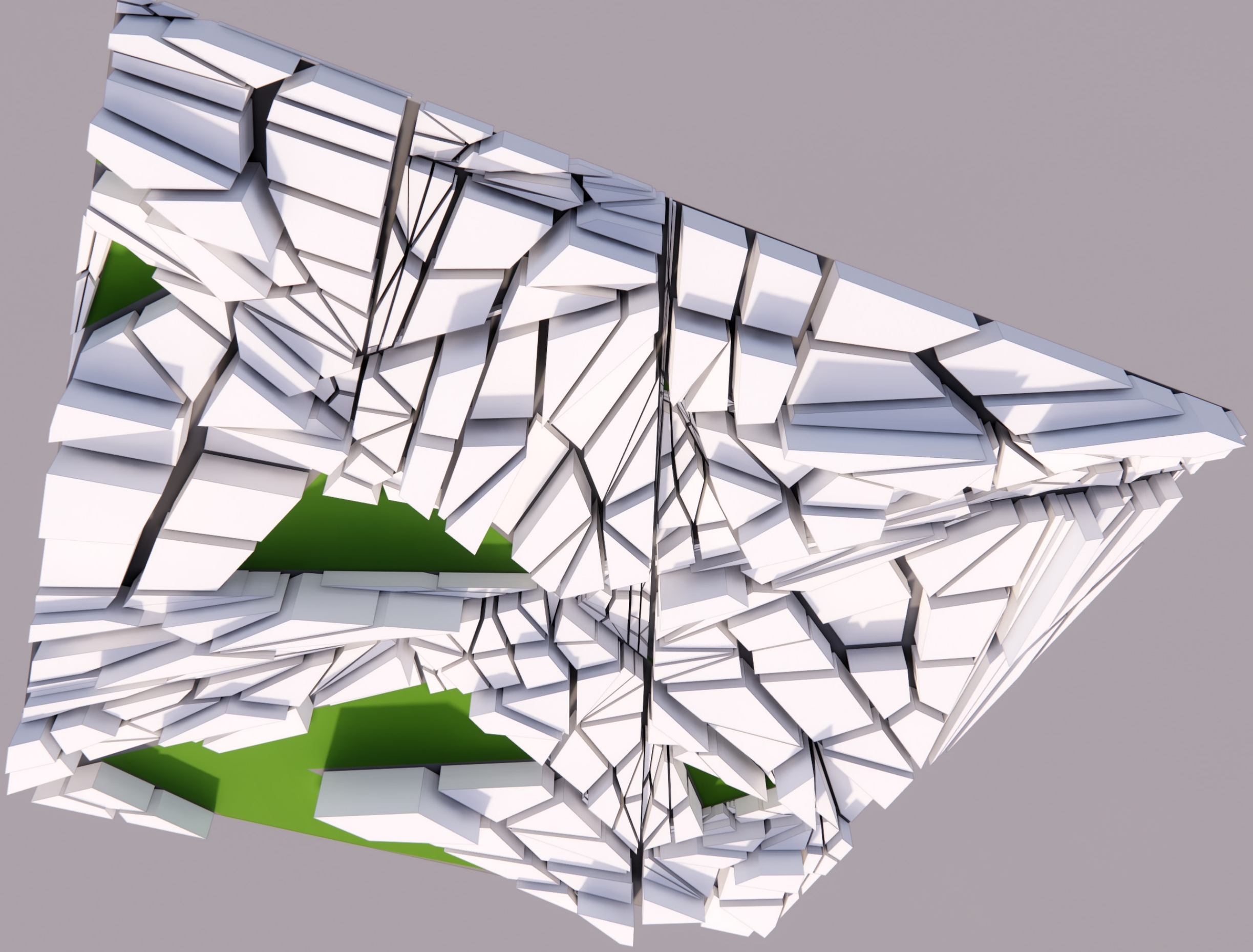
FV. 6 : 1174.330815

Final Solution : Transport Nodes and Green Spaces









6. Discussion

Limitations, Control, and Improvements

As previously discussed, when tested the model tends to one of 3 forms of interest; minimal, large cell and dispersed. Minimal and large cell can both be put down to defects is the model, minimal model occurs due to the lines occurring directly on top of each other giving the effect of less lines and cells. Large cell phenomenon occurs when the model promotes one cell over taking a proportion of the grid, effectively isolating a large section of the area - this occurs due to the testing criteria using the centroid of cells for walkability measurements, so the expansion of area in the isolated section aren't considered in calculations.

Due to these defects, the model was programmed to reduce the standard deviation and maximum cells sizes, in an attempt to create an evenly dispersed division of area, one that closer represents a city grid. This resulted in the dispersed cell model, of which the final design took its form. On one hand this felt logical for application of design and represented a true built environment, avoiding disproportionality large or small buildings. Although, considering the scope of this project and the desire for truly organic design, these controls could be considered restrictive, especial since the logic is based of the current grid system.

It's important to remember were trying to reconceptualise the design of a street network, abstract form, and lack of resemblance to other city grids isn't a failure of the model it's in fact proof of concept for organic form, as well as a statement for the ineffective nature of or ridged grid systems. The result of this project shouldn't be a fully functional city design but a vision that provokes and challenged existing urban design.

Future Expansion

This study evidence the application for walkability considerations in urban, generative design. It has been shows how consideration for access to transport and green space can be used in shaping the city, but these are just two of the many aspects that make a city walkable and accessible. If we wish to greater develop this study and our vision of Masdar, further consideration for access to other amenities would be considered – applying a purpose driven cells we could quantify accessibility of shops, school, health care, recreational spaces and more.

Further, for greater application in real world development, qualitative ranking of desired proximity could be applied to different assets. Thus, allowing a tailored approach to the organic design and providing a application for a diverse range of cities and functions.

The renders show a vision for a walkable city, but the design is by no means complete. Growth, land use and scale must be decided by other parameters and testing outwith the scope of our investigation. But instead, what we show case here is the capabilities for something organic, Masdar presents an opportunity to design from scratch and we believe an ideal model for user centric design can be birthed from the organically generated modelling.

References

Brown, 2016 Brown N.C. Multi-
Objective Optimization for the Conceptual Design of Structures.Massachusetts Institute of Technology; 2016. Available at https://dspace.mit.edu/handle/1721.1/106367.

Brown, 2019 Brown N.C. Early Building Design Using MultiObjective Data Approaches. Cambridge: Massachusetts Intitute of Technology; 2019. Thesis. Available at https://dspace.mit.edu/handle/1721.1/123573.

Nourian et al., 2015 Nourian P., et al. Configurbanist: urban configuration analysis for walking and cycling via easiest paths. In: Towards Smarter Cities. Towards Smarter Cities, eCAADe 2015 33rd Annual Conference 16–18 Sept. 2015; 2015

Masdar Reimagined

By Chukwuemeka Nwosu & Felix Mallinder