City Form Lab

at the Harvard Graduate School of Design

Urban Network Analysis Toolbox for Rhinoceros 3D

HELP

version 5.10.10.20 R5RS10

Andres Sevtsuk; Raul Kalvo

Contact:
asevtsuk@gsd.harvard.edu

License

CONTENTS

1 INTRODUCTION

Urban Network Analysis (UNA) offers powerful methods for assessing distances, accessibilities and encounters between people or places along spatial networks. The new UNA Rhino toolbox makes urban network analysis available in Rhino 5, adding a number of new features and functionalities. The UNA Rhino toolbox was developed in order to make spatial network analysis tools available to architects, designers and planners who do not have access to GIS and typically work on designs in Rhino. Having UNA metrics in Rhino, not only allows one to analyze how a specific spatial network performs, but to also incorporate the analysis into a fast and iterative design process, where networks can be designed, evaluated and redesigned in seamless cycles to rapidly improve the outcome.

The UNA Rhino toolbox is significantly faster that its GIS counterpart, which has been available as a plugin for ArcGIS since 2012. Users also have an ability to rapidly create and edit networks from any Rhino curve objects, making network design and redesign simple and intuitive. The analytic options available to the user have expanded, giving you more precise control to analyze exactly what you need for every unique spatial network problem.

The toolbox is in beta version. It is still in development and we look forward to your feedback and suggestions in making it better for day-tod-day design and research practice.

The bibliography attached to the bottom of this tutorial includes a set of articles and books on the subject. A short introductory video about the Urban Network Analysis can be found online at: http://vimeo.com/44728530 . Tutorial videos about the different functions of the toolbox can be found online at: http://cityform.gsd.harvard.edu/projects/una-rhino-toolbox (scroll to the bottom of the page for videos).

2 INSTALLATION

2.1 DOWNLOAD

Before you install the UNA toolbar, make sure you have Rhino version 5 and the latest updates from McNeel installed. You can install the updates by opening Rhino and going to help > check for updates.

- You need to have Rhinoceros 5 on MS Windows, with Service Release 10 or later to use the UNA toolbox (free updates from McNeel).
- You also need to have an updated Dot Net Framework 4.5.1 or later on your windows operating system (free download from Mircrosoft).

Download the UNA Toolbox folder from the following Dropbox link:

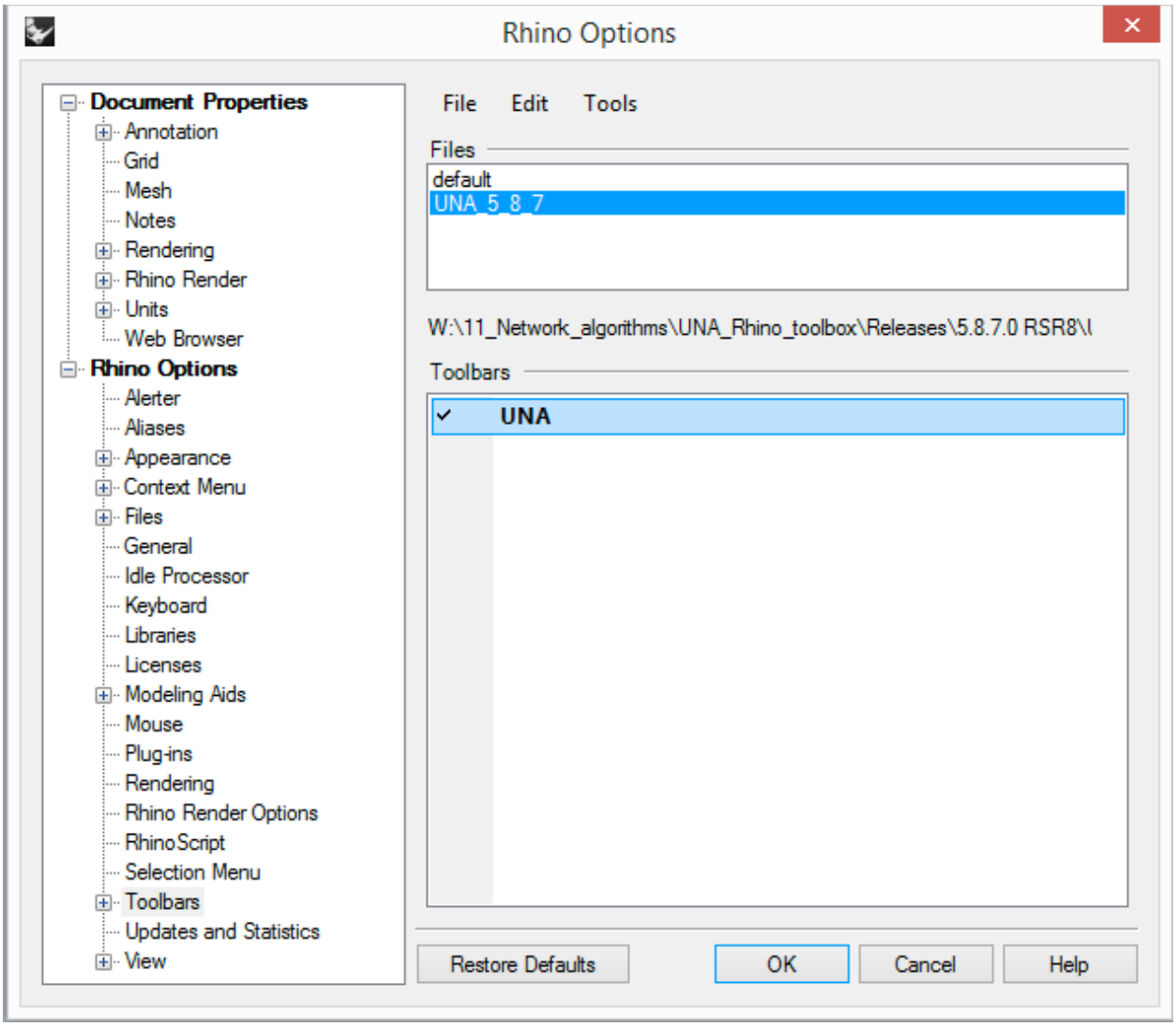
http://cityform.mit.edu/projects/una-rhino-toolbox

You should see the "UNAToolbox.rhi" file. Run it and navigate through the installation steps.

Note, that with some recent McNeel updates the .rhi installer for UNA may not be recognized as a program on windows. This is a known Rhino bug that McNeel is aware of. You can still install the UNA toolbox in one of two ways:

a) drag and drop the *.rhi installer file into an open Rhino window.
b) reassociate the *.rhi file in Windows again with C:\Program Files\Rhinoceros 5 (64-bit)\System\x64\rhiexec.exe

Once installed, you may see the UNA toolbar appear automatically in Rhino. If it does not appear automatically, then in Rhino, go to Tools > Toolbars tab and make sure you check the box next for UNA. Click "OK".



Now you should see a new toolbar in Rhino that looks like this:

These are the tools that enable Urban Network Analysis in Rhino. You can also activate each of the tools from the Command Line. UNA tools typically start with the “una…” followed by the tool name for each function.

3 GRAPHIC USER INTERFACE

The GUI is divided into five sections, providing attribute editing, data import/export, network setup, analysis, and graphic settings functionality respectively.

The first set of the tools (Tools 1-7 from the left) deal with assigning objects attributes, which are optional to the analysis.

The UNA Toolbox for Rhino utilizes “attributes” to give properties to origin and destination objects, such as names, numeric or weights. These attributes are analogous to Attribute Tables in ArcGIS shapefiles. An object can have any number of attributes. However, unlike ArcGIS, the Rhino attributes are not stored as a table, but rather as a dictionary, where the key indicates the attribute name and values indicate attributes values. Attributes can be either numeric or text based.

1. **Add Text Attribute to Objects** 

Attributes allow you to distinguish spatial objects from one another by giving them unique properties that correspond to their real world characteristics. Assigning a Text Attribute to geometric objects in Rhino could, for instance, be used to designate a “Business_name” or “Cleanliness_Rating” in the range of “A”, “B”, “C” etc. Text attributes you enter can also be used as an IDs or any other descriptive fields that may be helpful in your analysis. Text attributes can not be used as weights in the analysis.

   The tool allows you to select one or more objects at a time and then prompts for: Attribute Name > Attribute Value.
   Your chosen “Attribute Name” is a dictionary key, analogous to a column name in a table. The “Text Value” is the value that corresponds to the Attribute (e.g. dictionary entry or row value in a table).

2. **Add Numeric Attribute to Objects** 

This tools allows you to add a Numeric Attribute to geometric objects in Rhino. Numeric attributes are particularly useful for spatial network analysis as they allow you to “weigh” the analysis according to each object’s value. A “Numeric

Attribute" could, for instance, indicate the "Number of Floors" in a building, the "Area" of a space, the "Number of Employees" in a business etc. The attribute you provide can later be used as a "weight" in your analysis. The tool allows you to select one or more objects at a time for inputting these attributes and then prompts for:
Attribute Name > Attribute Value.
Your chosen "Attribute Name" is dictionary key, analogous to a column name in a table. The "Text Value" is the value that corresponds to this Attribute (e.g. dictionary entry or row value in a table). Note that the numeric input may be an integer or a decimal point number, negative or positive.

3. **Add Tag Attribute to Objects**
   This tool allows you to add a tag name to chosen objects. After selecting the objects, the tool prompts you for the attribute name by prompting "tag <>:" on the command line. Assign the tab name you want to use on the command line and press enter. The tag you enter is simply a name that doesn't contain any values, as opposed to the text or number values in the previous two tools.

4. **Remove Attribute from Objects**
   This tool allows you to get rid of some previously added attributes from chosen objects. If you choose all objects in the scene and apply this tool, then all objects will lose the tags you specify.

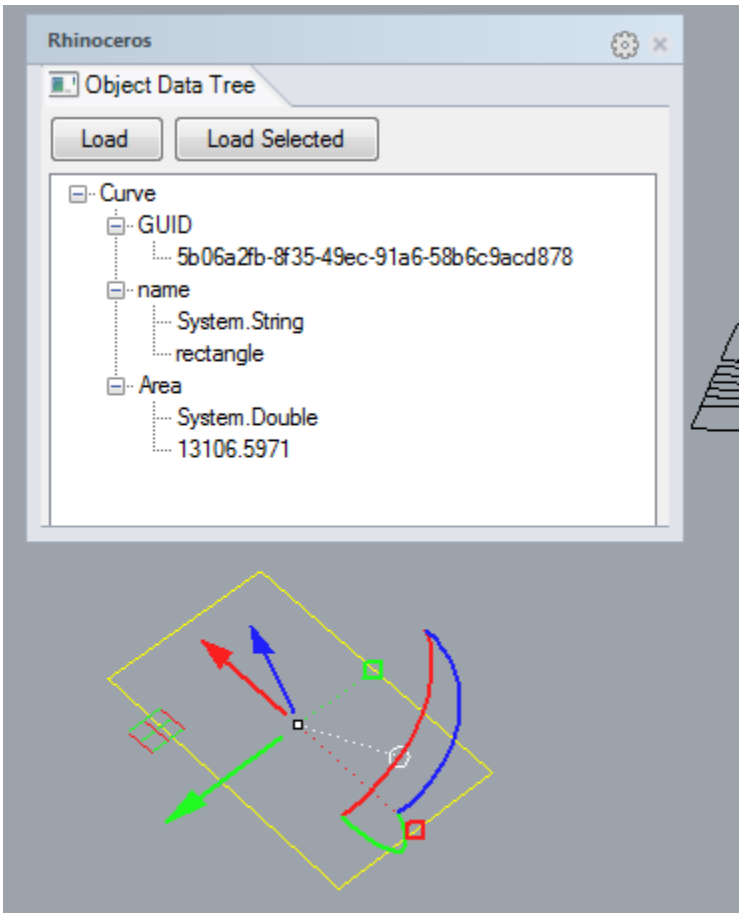
   After selecting the objects, the tool prompts you for the tag name by asking "Tag: " on the command line. Type in the tag name that you want to remove. Note that you cannot remove GUID, which is a unique ID for each Rhino object.

5. **Save Result as Weight**
   This tool allows you to one result at a time to the object attributes, optionally assigning it a custom column name in the table.

6. **Show Attribute Tree**
   This tool allows you to see what attributes objects carry. The attributes are structured as a tree, showing the name of the attribute, value type (e.g. string,

double, integer etc.) and value. Note that every geometric object in Rhino automatically carries a unique GUID. This is the first attribute you see in any object's Attribute Tree. In the image below, the rectangle that is selected has a GUID, a Text string Attribute called "Name" with a value of "rectangle" and a Numeric Attribute called "Area" with a value of "13106.5972".

The "Load Selected" button allows you to pull up the attributes of only selected objects. The "Load" button lists the attributes of all objects in your Rhino scene. Note that if you have a lot of objects, this list can be very long and hard to follow, loading selected attributes is typically more practical.



The next section of tools Next we turn to tools that allow you to import or export data from/to UNA network objects.

7. Import points

The import points tool can be used to bring in origin or destination point data with attributes from GIS or other table files to Rhino. The tables you import may include X,Y and optionally Z coordinates, which will be drawn as points in Rhino. All other columns from the original table are brought along as attributes to the points in Rhino.

Note that there are some restriction in how your table columns can be named so that Rhino can recognize them as attribute names:

- Spaces in column names are automatically ignored (e.g. “My Name” is converted to “MyName”).
- Underscores that form the first character in a column name are automatically removed.
- column names can contain numbers but the entire name may not be a number.
- A couple of names are reserved for internal UNA processes – “none” and “count” may not be used as column names.

8. Import Table 

This tool allows you to bring in a table of attributes that may pre-exist or that may be available as part of some other dataset you have in GIS, Excel, etc. In order to import any of such attribute tables, you will need to first convert the pre-existing tables into either “tab separated values” (“tsv”) or comma separated values (“csv”). The tool can only accept these formats. It is very common to have data in an Excel sheet, which allows you to save to either a “Text (Tab Delimited) (*.txt)” or “CSV (Comma Delimited) (*.csv)” formats. The default format is “tsv”. Click on the “Format = *tsv*” option in the command line if you want to switch to “csv”.

Click on the “File” option in the command to browse to the file you want to import.

The “Geometry = *all*” default option will attempt to match the imported attributes to all objects in your Rhino file. You can also choose to only try to join attributes to selected objects, by changing this to “Geometry = *selected*” and clicking on this value in the command line. This will prompt you for an object selection.

The "UpdateWeights = *On*" default option requests that your newly imported attribute values will automatically be available as weights for UNA analyses. If weights with the same name already exist on Rhino objects, they will be re-written as part of the ImportTable process.

9. **Export** 

This tool allows you to export the attributes you have added manually or obtained when Rhino UNA analysis tools, into a table. The command line offers a few options. You can change formats between tab separated values "tsv" and comma separated values "csv", and you can choose to either output the entire table to clipboard (memory) or to a file. If you output to the default memory, then you can simply "paste" in Excel and the exported values are dropped into a table. This can be useful if you would like to manipulate object weights or join the attributes with additional indicators in Excel.

Next we turn to tools that allow you to construct spatial networks and add/remove origins and destinations on the networks.

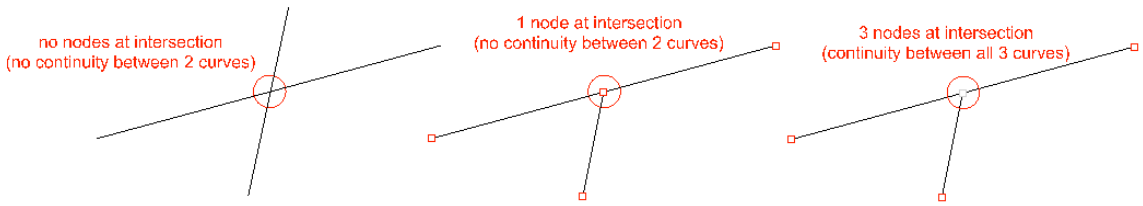
10. **Add Curves to Network** 

The way spatial networks are represented in the Rhino UNA toolbox is similar to the ArcGIS UNA toolbox (if you happen to have used it). Any network analysis requires at least two user inputs: first, an input analysis network and second, input locations. If you have been using our GIS UNA toolbox may recall that creating new network datasets (ND) in ArcGIS requires multiple steps and is difficult to automate. Furthermore, at the start of calculating each centrality analysis in ArcGIS, an "adjacency matrix" is calculated, among the input points on the network. Depending on the size of the dataset, this can take substantial time in GIS. These two procedures – creating a network and creating an adjacency matrix – are substantially more efficient in Rhino, as explained below.

The Add Curves to Network tool asks you to select curves that you want to build your network out of. Select all the curves you want to involve in the network and press enter. This will automatically turn the selected curves into a network and build an adjacency matrix that is used for analysis later.

Note that through a left click, this tool also enables you to add more curves to the network you have already built. If some of the curves you might be adding to a pre-existing network are already part of the network, their GUIDs will be recognized and they will not be double represented. Right clicking the tool, on the other hand, allows you to remove curves from an existing network.

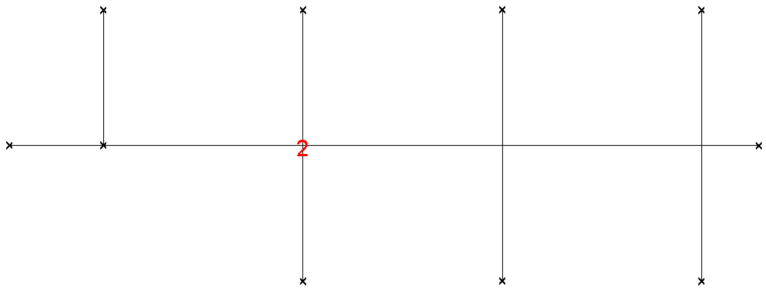
In order for network analyses to work, network curves need to provide continuity between the "input locations" you analyze. If your network curves do not share a common end node, then the "input locations" found on different network segments will not be topologically connected to each other. If one curve ends on top of another, but the latter does not have a node at the intersection point, then topologically there will not be a continuity between the two curves. Curves that intersect without sharing a common node can be used to model 3d overpasses or underpasses.



The tool can accept any kinds of curves to participate in networks – lines, polylines, curves, arcs, etc. Networks of these curves can either be planar (2D) or three-dimensional, as long as adjacent curves share a common end node with each other. While 2D networks may be adequate for representing street and building networks in urban settings, 3D networks offer additional opportunities for analyzing circulation systems and layouts within buildings or urban infrastructure systems.

For visual clarity, the default settings in the UNA graphics options will visualize dead-end or naked ends of the network with black crosses. You can turn the black crosses off in the graphicOptions by turning off the Nodes option. The black crosses can be useful to visualize where your network might contain topological errors. In the figure below, for instance, the first and second intersections from the left along the horizontal curve have topology issues. On the first intersection, a black cross is drawn, indicating that one or more curves around that node has a dead end and doesn't connect to any other curve. On the second node, a red warning with a numeric value "2" is displayed. This warning, which can be toggled on/off in the GraphicsOptions using the *NodeD2* option. The red number

tells us that the node does not have any dead ends, but that there are two overlapping curves that do not connect with each other (e.g. as in an overpass).

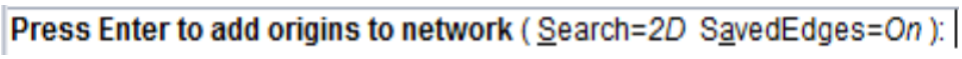


11. Add Origins

Once you have added curves to a network, you may proceed to locating points on the network. Most of the analytic functions in the UNA toolbox use discrete locations (points, buildings, entrances etc.) as units of analysis. Accessibility results are typically computed to desired point locations separately (though in the case of Redundancy and Betweenness tools also enable results to be shown for network links rather than points next to the network). There are three types of inputs points in the Rhino UNA Toolbox: Origin Points, Destination Points and Observer Points.

This tool adds origin locations to the network. For Centrality analysis, Origins are the points that analysis results are calculated for. For Betweenness, Redundancy and Closest Facility analysis, origins represent points where trips start.

When clicked, the tool first asks for a selection of points. After selecting points with a mouse and pressing enter, the following options appear on the command line:



First, the"Search" option allows you to choose whether you want your points located in 2D or 3D space. If either your input points or network contain variations in the Z-dimension, then your analysis should become 3D. You can toggle between 3D and 2D by clicking on "Search".

Use_saved_edge=*On SavedEdges=On* option allows you to check if a point has a saved, designated edge that it is tied to. It is possible in Rhino UNA toolbox to tie a point to a particular edge of the network, not necessarily the closest edge. If the edge reference that a point carries is not found in the drawing file, then it is ignored and regular closest edge joining is used instead. You can typically ignore this input.

Once you have added your origin points, each origin point that is located will be tied to its closest network element with a blue connection line. The point at which this blue connection line snaps to the original network, designates the assumed network location of the point. You can toggle the blue connection lines on/off in the GraphicOptions by controlling the *DotConnections* option.

As with network lines, left-clicking adds origin points, right-clicking removes origin points.

12. Add Destinations 

This tool adds destination locations to the network. For Centrality analysis, Destinations are the points that the analysis tries to reach; each destination affects results for origins, but destinations themselves are not given results. For Betweenness, Redundancy and Closest Facility analysis, destinations represent points where trips end. The user command line options for destinations points are the same as for Origin Points. Destinations points that have been successfully added to the network are indicated with red arrows that point from the network towards the point. You can toggle the red connection lines on/off in the GraphicOptions by controlling the *DotConnections* option.

Left-clicking adds destination points, right-clicking removes destination points.

13. Add Observer points

Observer points are only relevant for Betweenness analysis. Observer points enable you to return Betweenness analysis results for points that are neither origins nor destinations of trips themselves, but are potentially passed by routes in between origins and destinations. If trips originate from bus stops and go to shops, for instance, then buildings in between bus stops and shops can be used as ObserverPoints to illustrate how many trips pass by each building. The Betweenness analysis can keep track how many times each observer point is passed in the analysis. The weights of observer points are not used as part of the analysis. Observer points are tied to the network with gray dot connections (see below), which can be toggled on or off in the Graphics options. Left click adds observers, right click removes observers.

Note that Observer points can be important if the origin and destination points are located on the same edge segment. The betweenness results for observer points are always exact, while the betweenness values for edges are approximations, found by the taking the average of betweenness values of the segment's end nodes.

14. Delete Network

This function allows you to delete the entire network representation you have previously added in the Rhino scene, including all origin, destinations and observer points, and start over with a clean network and O-D point selections.

The next section of tools in the UNA toolbar includes the key analytic functions that produce network analysis results – centrality tools, betweenness, redundancy etc.

15. Centrality Indices

The BuildCentralityIndices tool launches the centrality computations for the network and origin-destinations points you have included. Centrality tools analyze how central each origin location is with respect to given destination

locations; analysis results are returned to origin points. If the analysis is weighted, then the weights are applied to destination objects. If you analyze accessibility from buildings to stops, for instance, then buildings should be taken as origins and bus stops as destinations. You may optionally want to weigh each bus stops by a numeric attribute indicating how many bus lines each given stop serves.

First, the command line message will ask you to select the Origins for the analysis or to accept the pre-selection. The pre-selection automatically detects all origin points you have previously added to the network. If you wish to use all of them for the analysis, then go with the pre-selection, if you wish to only select a subset of points for the present analysis, then select those origins you want to include. All centrality results will be computed for the points you decide to select here. Selected Origins will be temporarily indicated with blue crosses.

Next the tool prompts you to select Destination Points in the same way. Selected destinations are marked with red crosses.

Once you have selected the origin and destination points, you will be prompted for the following inputs:

Search Radius <600> (Reach=*On* Gravity=*On* Closeness=*On* Straitness=*On* Weight=*Count* Beta=*0.004* Alpha=*1*):

The Search Radius input defines the network radius used for computing the accessibility measures you choose. For each Origin point, only destination points whose shortest network distance from the origin is less than Search Radius, are considered in the analysis. The Search Radius units follow the drawing units – if your drawing is in meters, Search Radius is also in meters. The active Search Radius is shown at the beginning of the command line prompt; you can change the Search Radius by typing a number on the command line.

Next you can see a list of metrics that are offered in the Centrality tool, each turned "on" by default. There are four network centrality metrics available: *Reach, Gravity Index, Closeness and Straightness*. Turn them on or off by clicking each one on the command line.

The Weight option allows you to weigh the analysis with destination attributes. If you use the Reach analysis, for instance, to measure how many jobs you can walk to in from each building in a 600m walking range on the network, then you might weight the destination buildings with a "Jobs" weight, indicating the number of jobs at each building. The results would then illustrate the numbe of jobs reached from every origin building in a 600m range.

The *Beta* and Alpha values at the end only affect the Gravity Index. Each of the indices is explained in detail in the next section.

Reach

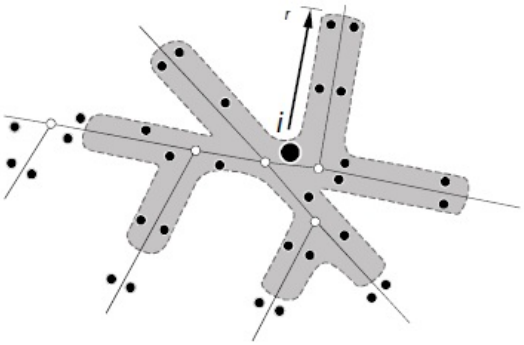
The "Reach" measure (Sevtsuk, 2010) captures how many surrounding points (e.g buildings, doors, bust stops etc.) each building reaches within a given *Search Radius* on the network.[1] The reach centrality, $R^r[i]$, of an Origin i in a graph G describes the number of Destinations in G that are reachable from i at a shortest path distance of at most r. It is defined as follows:

$$Reach[i]^r = \sum_{j \in G - \{i\}, d[i,j] \leq r} W[j]$$

where $d[i,j]$ is the shortest path distance between origin i and destination j in G, and $W[j]$ is the weight of a destination j.[2] Figure 10 illustrates how the Reach index is calculated visually. A buffer is traced from each building i in every direction on the network until the limiting radius r is reached. The Reach index corresponds to the number of Destinations j that are found from the Origin within the Search Radius on the network.

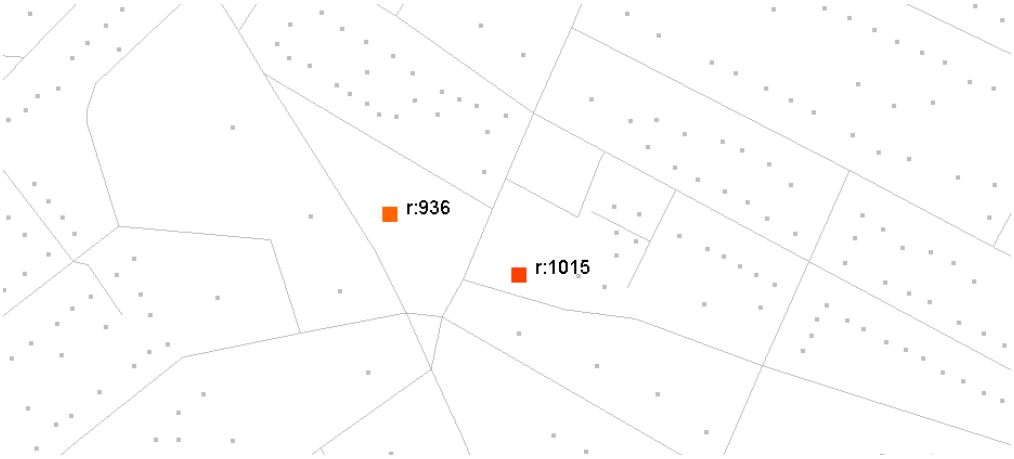
[1]The toolbox does not model buildings with multiple entrances, which can play an important role for building accessibilities in a

[2]The Reach measure we use is identical to a cumulative opportunities type accessibility index, described by Bhat et al. (2007), but applied on a network rather than Euclidian space.



Reach can be calibrated to measure access to any type of destination. In order to simply compute how the number of Destination points are reached within the given *Search Radius*, you can set Weights=count, so that no weighting will be applied and only the count of destination buildings will be returned. To weight the measure by building size, for instance, you can give a "building GFA" attribute to destination points and use them as Weights. The Reach measure will then compute how much built volume is reached within the Search Radius around each Origin on the network. To capture Reach to activities or land use destinations, you can use the number of jobs, the number of residents, the number of business establishments and so on at Destination points as *Weights*.

The figure below illustrate Reach from two subway entrance points in Cambridge, MA to surrounding building destinations in a 600m SearchRadius.



Gravity Index

Whereas the Reach metric simply counts the number of destinations around each Origin within a given Search Radius (optionally weighted by Destination attributes), the Gravity measure additionally factors in the travel cost required to arrive at each of the destinations. First introduced by Hansen (1959), the Gravity

index remains one of the most popular spatial accessibility measures in transportation research.

The Gravity measure assumes that accessibility at Origin i is proportional to the attractiveness (weight) of destinations j, and inversely proportional to the distances between i and j:

$$Gravity[i]^r = \sum_{j \in G - \{i\}, d[i,j] \leq r} \frac{W[j]^\alpha}{e^{\beta \cdot d[i,j]}}$$

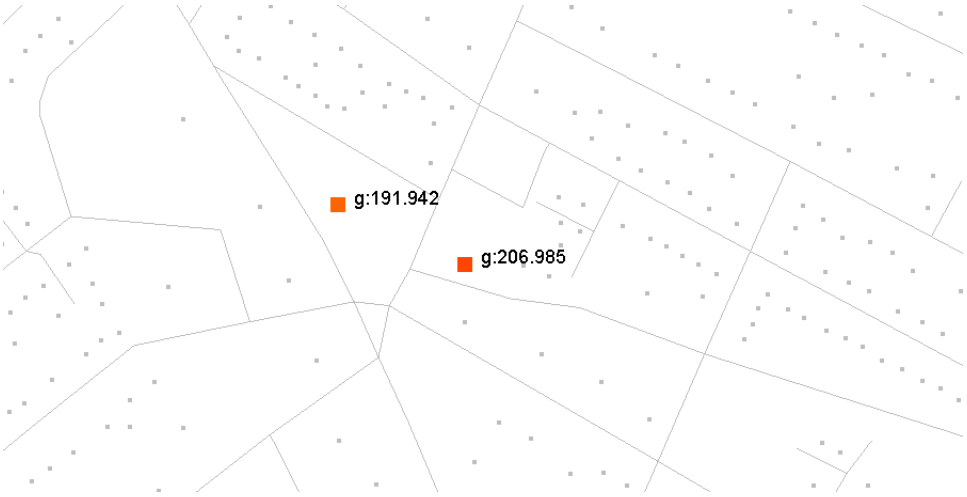
where $Gravity[i]^r$ is the Gravity index at Origin i within graph G at Search Radius r, $W[j]$ is the weight of Destination j, $d[i,j]$ is the geodesic distance between i and j, α is the exponent that can control the destination Weight or attractiveness effect, and β is the exponent for adjusting the effect of distance decay. The gravity index thus captures both the attraction of the destinations W[j]^α) as well as the spatial impedance of travel required to reach those destinations ($d[i,j]$) in a combined measure of accessibility.[3] If no *Weight Attributes* are given, then the weight of each destination is considered to be "1". The default value for alpha is also set at "1", so that the destination weight has a linear effect.

The inverse effect of distance specified in the Gravity index decreases exponentially. The exact shape of the distance decay can be controlled with the exponent β, specified in the command line input when running the Centrality tool. β and the corresponding shape of distance decay should be derived from the assumed mode of travel - for walking measured in "minutes", for instance, researchers have found β to fall around 0.1813 (Handy and Niemeier 1997)[4]. This corresponds to *0.00217* in meters. In the context of Singapore, for instance, we have determined that beta for walking varies between 0.005-0.005. Beta values that are higher (closer to 1) indicate stronger aversion towards walking distance.

[3] Consider two homes nearby a retail store. The first home is located a mile away, but the second home only half a mile away from the store. Even though both homes have the same number of store destinations available (one) and the destinations are identical in weight, the gravity index would consider the closer home to be more retail accessible than the further home.

[4] The equivalent value of Beta for impedance units in "feet" 0.000663; in "kilometers" 2.175, and in "miles" 3.501.

The figure below illustrates Gravity accessibility results from two subway entrance points in Cambridge, MA to surrounding building destinations in a 600m SearchRadius. Note how the resulting values are lower than Reach values due to the distance decay effect in the denominator of the index.



Closeness

The *Closeness* of an *Origin* is defined as the average distance required to reach from that origin to all the specified destinations that fall within the S*earch Radius* along the shortest paths (Sabidussi 1966). Similar to Gravity, the *Closeness* measure indicates how close an Origin point is to Destinations within a given distance threshold, but the distance that is used is a simple linear distance and the result is given as an average closeness between all destination points. The *Closeness* measure is defined as follows:

$$Closeness[i]^r = \frac{\sum_{j \in G-\{i\}, d[i,j] \leq r} \frac{W[j]}{d[i,j]}}{n}$$

where $Closeness[i]^r$ is the *Closeness* of Origin i within the *Search Radius* r, $d[i,j]$ is the shortest path distance between nodes i and j, and $W[j]$ is the weight of the destination j, and n is the number of destinations found.

Straightness

The *Straightness* metric (Vragovic, Louis, et al. 2005) illustrates the extent to which the shortest paths from Origins to Destinations resemble straight

Euclidian paths. Put alternatively, the *Straightness* metric captures the positive deviations in travel distances that result from the geometric constraints of the network in comparison to straight-line distances in a featureless plane. The *Straightness* measure is formally defined by (Porta, Crucitti et al. 2005) as:

$$Straightness[i]^r = \sum_{j \in G - \{i\}, d[i,j] \leq r} \frac{\delta[i,j]}{d[i,j]} \cdot W[j]$$

where $Straightness[i]^r$ is the Straightness of node i within the *Search Radius* r, $\delta[i,j]$ is the straight-line Euclidian distance between i and j, and $d[i,j]$ is the shortest network distance between the same nodes. The *Straightness* index illustrates how long the shortest path connections from each Origin to the surrounding Desinations j are in comparison to the as-a-crow-flies distance. Naturally, as the distances between nodes get longer, the differences between the network distance and as-a-crow-flies distance start diminishing – a walk from Boston to Los Angeles is much closer to a straight line than a walk from MIT to downtown Boston.

The figure below illustrates Straightness results from all buildings to all other buildings in Cambridge, MA in a 600m SearchRadius. Warmer colors indicate origins that have higher straightness values too all destinations that were found around them.

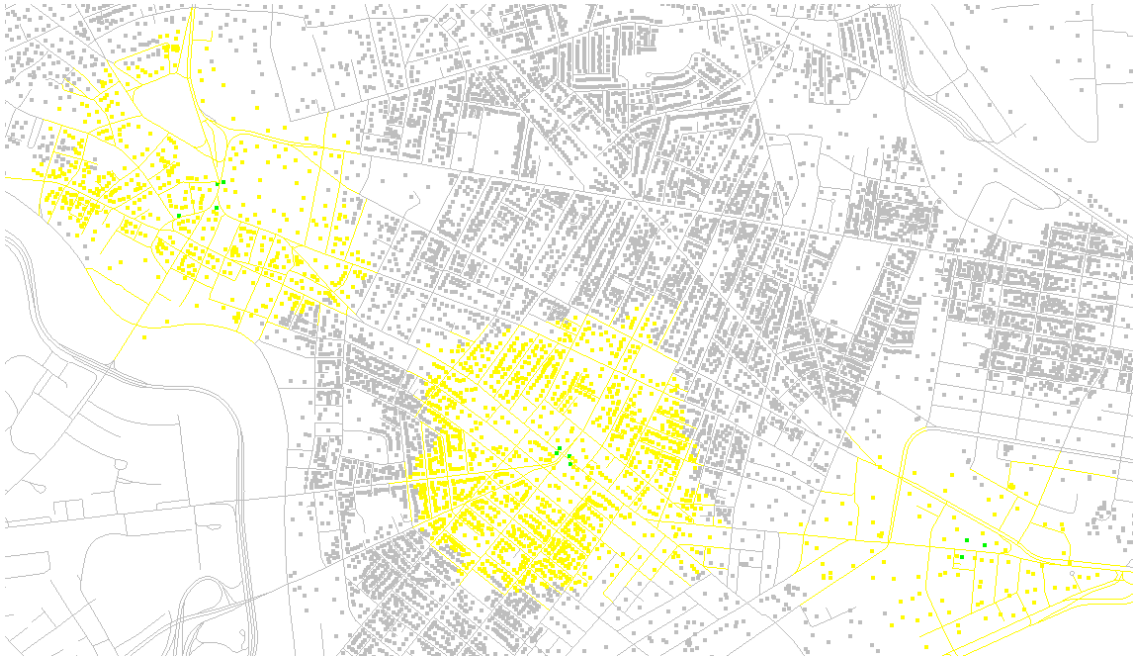
16.Service Area

The Service Area tool selects or copies destination points and path segments that fall within a give network radius from origins. The tool can be used, for instance, to select all restaurants (destinations) that fall within a 200m radius from a set of subway stops (origins). The Search Radius is <200> can be typed into the command line. The tool offers four options :

Service area <600> (SelectPoints=*On* SelecCurves=*On* Copy=*Off* Tight=*Off*):

SelectPoints and *SelectCurves* enables the destination points or curves that fall within the specified network radius, to be selected. Note that curves are selected with their full original lengths and not cut into shorter curves that exactly correspond to the Service Area radius. The additional option Tight allows the curve selection to only pick curves that are fully enclosed within the radius buffer. The Copy option allows you to create a copy of the selected features on the active Rhino layer. The tool is analogous to the ArcGIS Network *Analysis Service* Area tool, but it selects both destination points as well as networks curves that are reached within the given network radius.

The figure below illustrates ServiceArea results from selected surbay entrance points (green) to all surrounding buildings destinations in Cambridge, MA in a 600m SearchRadius. The yellow selected buildings and street segments indicate destinations that can be reached in a 600m network radius from the origins.



17. Redundancy Index

The Redundancy Index computes the increase in linear streets that becomes available when the shortest walk between an origin and destination is extended by a given percentage, called the Detour Ratio. First introduced in the ArcGIS UNA Toolbox in 2014, the index finds alternative redundant routes between an origin and destinations. When all available routes are found, the index is computed as the combined length of all routes between O and D divided by the length of the shortest path between O and D (Sevtsuk, Mekonnen et al. 2014). The result can be interpreted as a factor that sizes how much more linear street length becomes available on a walk when we assume that the walk may follow any of the paths that are up to x% longer that the shortest path. This percentage is input by a user variable called *Detour Ratio*, and shown as greater or equal to one. A detour ratio 1.2 means that all routes that are up to 20% longer than shortest will be analyzed.

When the index is weighted with observer points, then the results illustrate how many more point weights become accessible on a walk from an origin to a destination along redundant paths compared to the shortest path alone.

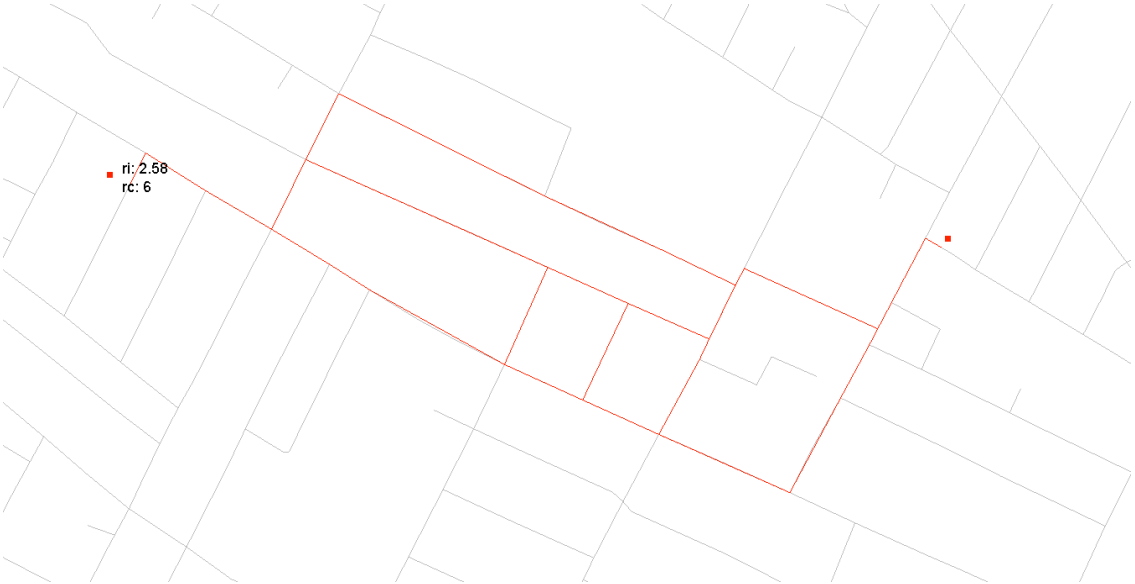
When the user inputs more than one destinations, then the *Search* option on the command line tells the algorithm whether to search for only the *Nearest* destination for each origin, *All* destinations or all destinations that fall within a given *Radius* from the origin. The Draw option controls whether the routes that are found are drawn.

Note that the routes that the Redundancy Index finds are not necessarily simple routes – they may contain loops and retrace repeat nodes as described in the related paper (Sevtsuk, Mekonnen et al. 2014). The Redundancy Index and the Redundant Route count are also output by the Betweenness tool if a detour ratio is applied there.

In the example below, Redundancy Index from the origin point on the left to the destination point on the right is 3.30 using a Detour Ratio of 1.2. That is, the length of routes from O to D expands 3.30 times compared to the shortest walk when routes that are up to 20% longer than the shortest are considered. When the Draw function on the command line is kept on, then the set of paths that participate in the solution is drawn with polylines and saved as a group.

18. Redundant paths

This tool finds all individual alternative paths between a set of origins and destinations that are up to x% longer than the shortest path. The tool can be used to study pedestrian route choice, for instance; it allows one to identify all plausible paths that a person might be expected to walk between and O and a D. Unlike the Redundancy Index, the paths found here are simple paths – they contains no loops or repeating nodes. The paths that are output are not grouped, each individual routes is drawn separately and precisely from an origin to a destination.
The figure below illustrates redundant paths (red curves) from an origin point (left) to a destination (right) that were found within a +10% detour ratio above the shortest path.



19. Betweenness

This tool calculates and visualizes the Betweenness index, which approximates by-passing traffic or footfall at particular locations in a spatial network.

The *Betweenness* of a building is defined as the fraction of shortest paths between pairs of other origins and destinations in the network that pass by a particular location i (Freeman 1977). If more than one shortesst path is found between two nodes, as is frequently the case in a rectangular grid of streets, then each of the equidistant paths is given equal weight such that the weights sum to unity. The *Betweenness* measure is defined as follows:

$$Betweenness[i]^r = \sum_{j,k \in G - \{i\}, d[j,k] \leq r} \frac{n_{jk}[i]}{n_{jk}} \cdot W[j]$$

where $Betweenness[i]^r$ is the betweenness of location i within the Search Radius r (specified in the *Search Radius* box); $n_{jk}[i]$ is the number of shortest paths from origin j to destination k that pass by i; and n_{jk} is the total number of shortest paths from j to k. *Betweenness* for location i is computed by considering all pairs of buildings j, k that are within a distance r from each other. It is not computed by considering all pairs of buildings j, k that are within a distance r from i. This is because we do not consider any trips between origins and destinations that are more than r apart. If we know that j, k are within r from each other, and that a shortest path from j to k (or k to j) passes by location i, then both j and k are also certainly within a distance r from i.

Before you run the tool, make sure you have built a network and added locations to it. First click on Origins, this prompts you to select which points you want to use as origins of your journeys. Second, select the destinations.

Betweenness (Search=*Nearest* DetourRatio=*1.2* Weight=*Count*):

The SearchOption input offers three options: “All”, “Nearest” or “Radius”. This determines which destinations will be used by each origin. By setting SearchOption to “All” asks the tool to route one trip from each origin to all provided destinations. If your origins are houses and destinations are bus stops, for instance, then a path is determined from each house to all bus stops in the graph, no matter how far they may be. “Nearest” asks the algorithm to only route trips from each origin to the nearest available destination. Using the same example, a path is found from each house to its nearest available bus stop. “Radius” allows you to input a distance and asks the algorithm to find paths from each origin to all destinations that lie within the given network distance from it. It allows you, for instance, to model walks from each house to all bus stops that are up to 600m away from the house.

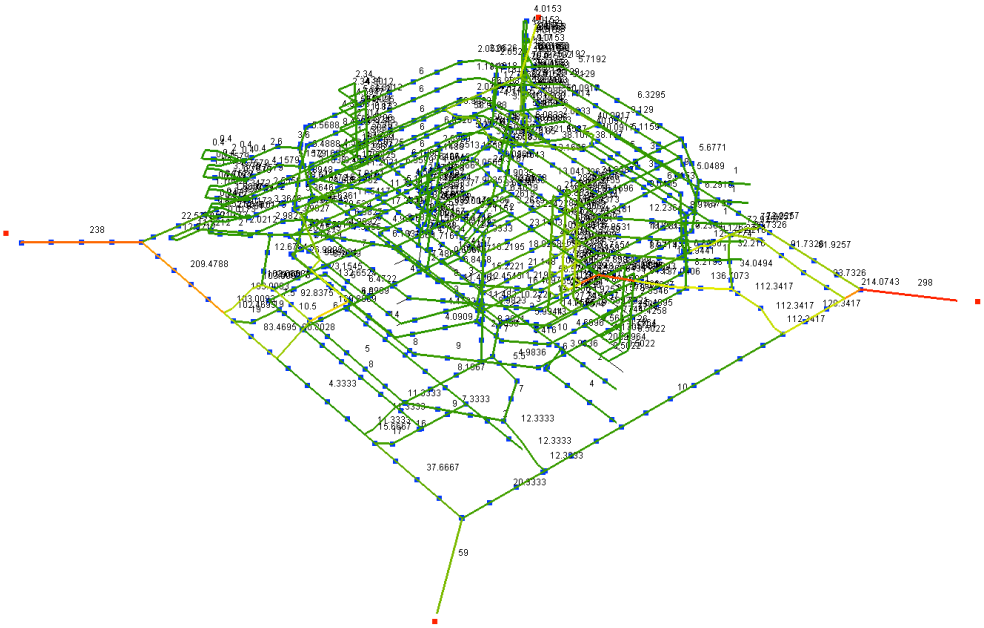
The Betweenness algorithm in the Rhino UNA toolbox has been specifically customized to make it useful and practical for estimating realistic pedestrian movement in spatial networks. Whereas a traditional Betweenness index would estimate trips from a set of origins to a set of destinations along shortest paths, keeping track of how many trips follow each route, the Betweenness algorithm in the Rhino UNA toolbox allows you to relax the shortest path assumption. A “DetourRatio” variable allows walks between origins and destinations to deviate up to the specified % above the shortest paths (the maximum deviation is fixed at 200%). Using a DetourRatio of “1.1”, for instance, allows walks to use paths that are up to 10% longer than the shortest path to reach the destination. Each alternative path that is found is given an equal likelihood, dividing the weights of the Origin point equally between all alternative paths. This is useful since people don’t necessarily take shortest paths in the city. Prior research has found that it is common for pedestrians to deviate around 10-20% above the shortest route in order to use a more useful, pleasant, simpler, or comfortable path.

Unlike other centrality indices, the Betweenness tool can also use “observer points”. Observers are points that are not origins or destinations themselves, but for which betwenness results are calculated. They can be points that represent buildings on city streets, for instance, or rooms inside builds where routes pass by. Observers need to first be added to the network, just like origins and destinations. If no observers are used, then betwenness results are returned to network edges instead.

Weights allow you to weight the analysis according to the properties of origin points. If an origin point has a weight of “100” for instance, describing its number of residents, then weighted Betwenness will route 100 trips from the origin location to the destination, instead of just one trip. If Weights is set on “Count”

then each origin is just counted as “1”, routing one trip from each origin to its destination(s).

The figure below illustrates Betweenness values from 712 office doors (blue points) in a 3d building network to the nearest exit point (red points in four corners), using a +20% detour ratio on all walks. The results essentially indicate which building corridors could be most congested in case of an evacuation scenario.



20. Closest Facility

This tool finds a closest destination facility to each origin point along the network and summarizes how many unique origins are within Reach or Gravity access in the market area of the destination facility. Unlike the normal Reach and Gravity metrics, here each origin point is tied to only one facility that it lies closest to. The command line choices offer the following options:

Search Radius <1500> (Weight=*Count* Gravity=*Off* Beta=*0.004* Lines=*On* SaveAs):

The SearchRadius determines the maximum network radius for linking origin points to destination facilities. Weights allow the analysis results to be weighted by numeric attributes at the origins. The default output indicates the Reach results of unique origins at each facility, but turning Gravity on also adds the gravity results. The Beta value controls the distance decay effect in the Gravity measure (see above about the Gravity metric). The Lines options allows the tools

to draw a straight line from each destination facility to each associated origin point – note that while these lines are straight just for visual simplicity, the points that relate to closest facilities are still determined on the network. SaveAs allows you to assign a custom name for the output results.

21. Find Patronage

The Find patronage tool offers users powerful methods to estimate the patronage of facilities located on a network based on a given distribution of demand points and competition from other facilities. It can be used, for instance, to predict the patronage of stores, markets, bike-sharing facilities, community centers or bus stops in the presence of competing facilities and demand points. The tool was inspired by a retail patronage model presented by Eppli and Shilling (1996).

There are two related tools in the button: a left click opens the *Find Patronage* tool, a right-click opens the *Find Patronage Window* tool. Both use a discrete-choice model to estimate patronage of facilities, but the *Find Patronage Window* tool works with a graphic user interface and contains additional functionality that allows the user to differentiate destination facilities into up to three different categories, each of which can have a different distance decay coefficient (beta) and a different cutoff radius variable.

FindPatronage tool requires users to generate a network (using *Add Curves to Network* tool) and to add both origin and destinations points to the network (using *Add Origins* and *Add Destinations* tools), before using the tool. Origin points can contain weights that indicate demand for facilities. For instance, if buildings are used as origins, then buildings' weights could indicate the number or residents at each building. The destination facilities can also contain a numeric weight that can be used to model their attractiveness. An "area" weight, for instance, can indicate the estimated sizes of retail facilities in square meters or feet, which can be accounted for in allocating the demand to facilities.

The user options on the command line include the following:

Search Radius <800> (OriginsWeights=*Count* DesintationWeights=*Count* Beta=*0.004* Alpha=*1* ApplyImpedance=*On* ApplyOriginWeight=*On* CopyRestultsToMemory=*Off*):

SearchRadius defines the maximum distance between demand points and facilities, beyond which no demand weight is allocated to a facility. In case of retail facilities, for instance, if a household is located further than the cutoff radius from a particular store, then none of the household demand is allocated to that store.

OriginWeights defines the attribute weight of the origin that is used to estaimte demand (e.g. residents at a building).

DestinationWeights defines the attribute weight of the destination facilities, used to estimate their attractiveness (e.g. area, capacity, brand recognition).

Beta defines the effect of distance decay used in allocating demand to facilities. Just as in the Gravity Index (see above), beta ranges from 0 to 1. If beta is less than 1, then the amount of demand allocated to a facility decreases exponentially as the distance to the facility grows. Beta should be derived from the assumed mode of travel - for walking measured in "minutes", for instance, researchers have found β to fall around 0.1813 (Handy and Niemeier 1997)[5]. This corresponds to *0.00217* in meters. In the context of Singapore, for instance, we have determined that beta for walking averages around 0.004, in Boston US around 0.002. Beta values that are higher (closer to 1) indicate stronger aversion towards distance. If no value is provided by the user, then a default value of "1" is assumed. See below for equations that use beta.

[5] The equivalent value of Beta for impedance units in "feet" 0.000663; in "kilometers" 2.175, and in "miles" 3.501.

Alpha defines the effect of destination attractiveness to patronage. If the destination weight indicates store areas in square meters (e.g. 1,500 m^2), then alpha defines how the effect of store area affects attractiveness as the area changes. Alpha is modeled as an exponent to area (e.g. $weight^{\alpha}$). See below for equations that use alpha. If no value is provided by the user, then a default value of "1" is assumed.

ApplyImpedance defines whether or not a distance decay function is applied to demand points. If applied, then not all demand reaches the reaches the destination facilities, resulting in lower total patronage across all destination than the sum of original demand weights would suggest. If a household is estimated to allocate 2 customers to a particular café located a mile away, for instance, then turning ApplyImpedance to "on" and using a beta coefficient for distance decay, would only allocate a fraction of the original "2" patrons to the café, since travel costs reduce patrons probabilities of visiting cafés at a distance. Similar households located nearer to the café, would allocate a larger proportion of their original demand to the café. This is similar to the Gravity function described above. See below for equations where *ApplyImpedance* is turned *on* or *off*.

ApplyOriginWeight defines if the results are calculated based on the demand weights at the origin (e.g. residents in each building point, when "on") or based on simply the number of origin points used (when "off"). See below for equations.

CopyResultsToMemory creates a clipboard copy of each facility's estimated patronage results, which can be pasted to Excel for further analysis.

The following explain the mathematical definitions of the different options of the tool:

Let "DP" = Demand Point (origin "i") and let "C" = Center (destination "j").

The gravity accessibility from a demand point "i" to a particular destination center "j" is given as:

$$\text{DP[i]Gravities[j]}=\frac{C[j]weight^{\alpha}}{e^{\beta * dist[i,j]}}$$

, where dist[i,j] is the network distance from origin "i" to a particular destination facility "j".

The sum of accessibilities from the demand point "i" to all available destination centers "j" that are available within the specified cutoff radius around it, is given as:

$$\text{DP[i]Gravity}=\sum_{j}^{\#c}\text{DP[i]Gravities[j]}$$

The probability that a person at origin point "i" will patronize a particular facilite at destination point "j" is given as a ratio between i's accessibility to that destination divided by i's accessibility to all possible destinations, including "j":

$$\text{DP[i]Probability[j]}=\frac{\text{DP[i]Gravities[j]}}{\text{DP[i]Gravity}}$$

Given these definitions, the tool can output four different results, depending on if the user turns on the *ApplyImpedance* and *ApplyOriginWeight* options.

1) if *ApplyImpedance*=**on** and *ApplyOriginWeight*=**on**, then patronage at center "j" is calculated as follows:

$$C[j]Patronage = \sum_{i}^{\#DP}\left(\text{DP[i]Weight} * \text{DP[i]Probability[j]} * \frac{1}{e^{\beta * dist[i,j]}}\right)$$

This solution reflects the "gravity discounted patronage" at each center "j". If empirically calibrated alpha and beta values are used, then the result can be used to estimate the actual number of patrons visiting each facility. Due to the last distance decay term in the equation (1/e^beta*dist[i,j]), not all demand weights reach destination facilities, resulting in lower total patronage across all destination than the sum of original demand weights would suggest

2) if *ApplyImpedance*= **off** and *ApplyOriginWeight*=**on**, then last terms in the above equation drops out and patronage at center "j" is calculated as follows:

$$C[j]Patronage = \sum_{i}^{\#DP}\left(\text{DP[i]Weight} * \text{DP[i]Probability[j]}\right)$$

This solution reflects the proportion of total demand weights allocated to a center "j". A result of "500", for instance, would indicate that facility "j" is estimated to get 500 patrons. The sum of all destinations total patronage equals the sum of original weights at demand points that are located within the specified search radius from destinations.

3) if *ApplyImpedance*= **on** and *ApplyOriginWeight*=**off**, then patronage at center "j" is calculated as follows:

$$C[j]Patronage = \sum_{i}^{\#DP}\left(\text{DP[i]Probability[j]} * \frac{1}{e^{\beta * dist[i,j]}}\right)$$

This solution reflects the "gravity discounted patronage" of demand point count (not their weights) allocated to a center "j". Each origin point is counted as "1", their weight is ignored. If alpha and beta values used are empirically calibrated and reliable, then the result can be used to estimate the number of demand points (e.g. households, not their weight "residents") estimated to patronize each facility. Due to the last distance decay term in the equation (1/e^beta*dist[i,j]), not all of the "1" demand unit at origins points reaches destination facilities, resulting in lower total number of points patronizing stores across all destinations than the count of original demand points would suggest.

4) if *ApplyImpedance*= **off** and *ApplyOriginWeight*=**off**, then patronage at center "j" is calculated as follows:

$$C[j]Patronage = \frac{\sum_{i}^{\#DP}\text{DP[i]Probability[j]}}{\#DP}$$

This solution reflects the proportion of total demand point count (not their weights) allocated to a center "j". Each demand point is counted as "1". A result of "0.5" here, would indicate that facility "j" serves 50% of all demand points in the system, which is different from serving 50% of the demand weights in the system as shown in (2). The sum of all destinations total patronage equals the count of original demand points that are located within the specified search radius from destinations.

Find Patronage Window tool functions with a graphic user interface and contains additional functions that allow the user to differentiate destination facilities into up to three different categories, each of which can have a different distance decay coefficient (beta) and a different cutoff radius under the Settings variables. The three center types are marked as C1, C2 and C3. In order for the tool to recognize, which center type each destination facility belongs to, the user

needs to attach a “C1”, “C2” or “C3” tag (without quotation marks) to desired destination centers. The tags should be added using the *Add Tag Attribute* tool from the UNA toolbar  .

The tool only requires a built network as a starting point. The origin and destination points should be added directly from the *Find Patronage Window*, not the regular UNA toolbar functions.

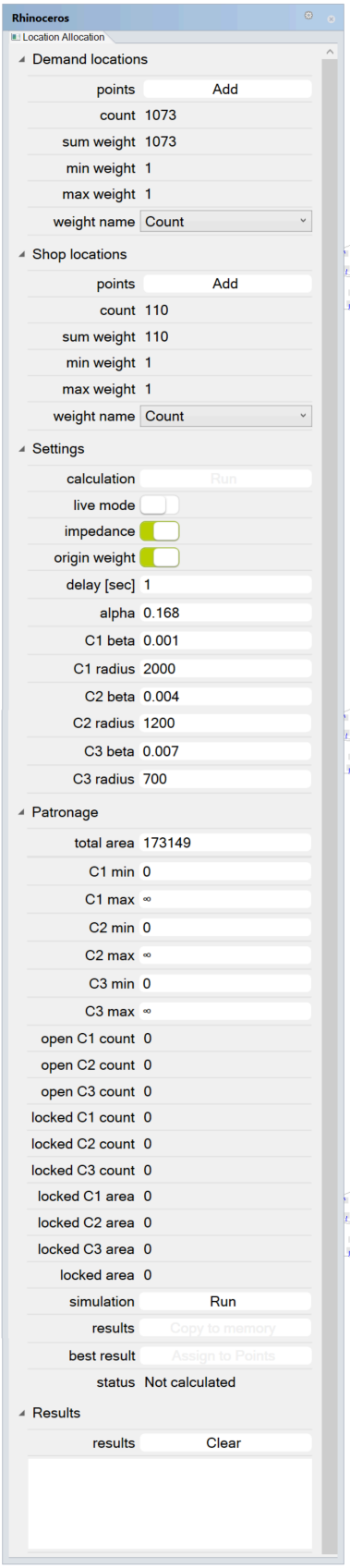
The tool contains a simulation engine that allows users to optimize the destination weight allocation to individual facilities, given a total weight deemed suitable for the whole system (town), so that the total patronage of all centers is collectively maximized. This can be useful for planning a number of grocery stores in a given market, for instance. Given a total combined floor area that is deemed absorbable for the given market, and given the suggested locations of stores, the tool can help identify optimal sizes for each individual grocery store such that the overall patronage to all stores combined is maximized. The size simulation keeps the total area constant at the user input total size.

A user can specify a desired minimum or maximum weight limit (e.g. floor area) for each type of center, using the “C1 min” and “C1 max” inputs under the Patronage variables.

The tool allows users to dynamically move around locations of facilities in the Rhino drawing canvas and to observe on the GUI how the patronage results change due to origin or destination location changes. This provides an interactive functionality for planners, urban designers and architects to try out different facility location and observe the implications of patronage in real time. This interactive functionality is only applicable if the apply “impedance” button is turned on. In this case, the calculation becomes analogous to equations (1) and (3) in the FindPatronage tool above, where due to the application of a distance decay function in the last term of the equation, patronage at destinations depends on the distance between demand points and destinations.

The tool prints out the patronage result next to each destination facility. The net difference in overall patronage in all stores’ collective patronage is printed into the bottom text box of the GUI.

The GUI is divided into 5 sections: *Demand Locations*, *Shop Locations*, *Settings*, *Patronage*, and *Results*.



Demand Locations. Click the *Add Points* button to add demand points and use the *Weight Name* drop-down to choose a weight for demand points. The min, max and sum indicators automatically read the assigned weights and summarize their min, max and sum values.

Shop Locations. Click the *Add Points* button to add facility points and use the *Weight Name* drop-down to choose a weight for destination points. The min, max and sum indicators automatically read the assigned weights and summarize their min, max and sum values.

Settings:

"Run" button is grayed out until both demand and shop locations have been added.

If the *live mode* is toggled to "on", then the patronage result at each destination center is automatically updated at one-second intervals each time the user graphically alters the destination or demand point locations in the drawing.

Impedance and *Origin Weights* buttons set the same calculation differences as shown in equations 1-4 above in the *Find Patronage* tool.

Delay determines the delay for graphic results updating in the drawing document when either origin or destination point locations are graphically altered by the user.

Alpha determines the attractiveness (e.g. size) coefficient for all center levels (C1-C3). The tool does not offer differential alphas for different destination types.

The following beta and radius inputs for C1, C2 and C3, define a unique beta and search radius cutoff for each center type.

Patronage:

Total area determines the desired total area across all destination facilities in the system for optimal size allocation simulation purposes.

The min and max variables for each center level that follow, determine the desired upper and lower limits for sizes in center type.

If some of the input destination shops do not contain tags, indicating "C1", "C2" or "C3", but have an area weight, then such destination shops are treated as "locked" in the simulation, which means that their areas are not altered by the simulation engine. This functionality allows the user to include existing or already built stores as part of the patronage analysis.

Press the "run" button to initiate the simulation, which will iterate through numerous possible size options for each center type and produce a resulting size allocation that maximizes overall patronage access all centers.

Results:

The results text box at the bottom of the GUI is used to print out overall patronage results across all destination centers in the system. If the "live mode" is toggled to "on" under *Settings* above, then the this overall patronage result is automatically updated at one-second intervals each time the user graphically alters the destination or demand point locations in the drawing.

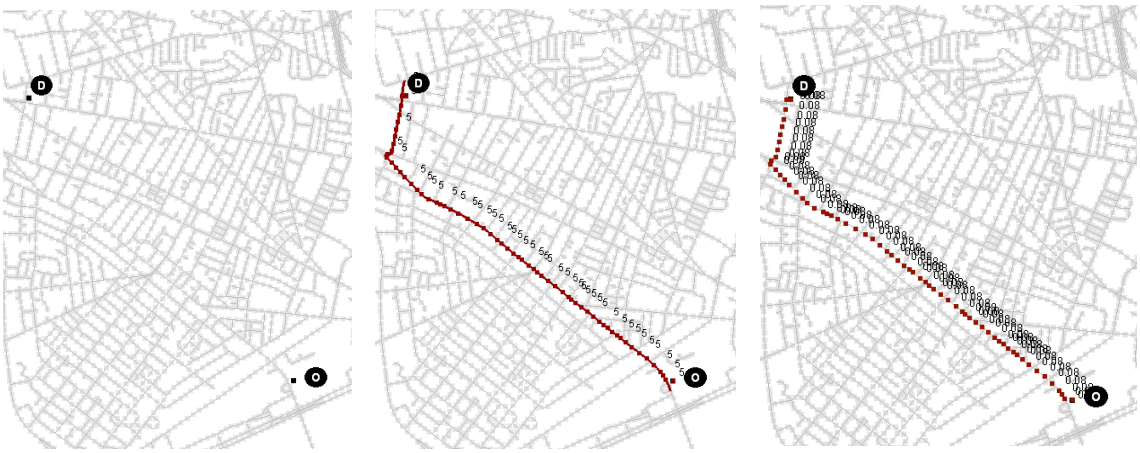
22. Distribute Weights 

This tool allows you to re-distribute point weights from stationary origin locations to network routes that lead to chosen destinations. For instance, if your original point weights indicate the number of households in residential buildings, then the tool can help your re-distribute these weights to walking routes to nearby transit stations, dropping demand points at chosen distance intervals (e.g. 50m), such that the total sum of demand weights stays the same as the original sum of demand weights at residential locations. This can be useful for modeling *en-route* demand for facilities (e.g. demand from residents is modeled as walking along streets who might decide to go to shops) instead of stationary demand from fixed locations.

Consider an origin point "O" and a destination point "D" in the following example. Let us assume that there are 5 students at the Origin, which happens to

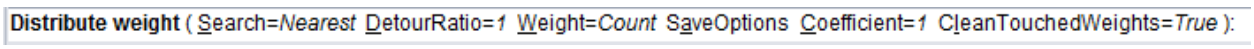
be the main MIT campus entrance, who will walk to the destination, which happens to be the Harvard GSD (left image). If we used the *Betweenness* tool to estimate these walks along the shortest route, then each of the observer points along the shortest path would get a value of "5", indicating that it is passed by 5 students (middle image). This is useful for predicting the number of passersby at each observer point. But if we wanted to use these values to estimate demand for ice cream on the way for instance, then the betweenness values might not be suitable. Given that there are 62 observers between "O" and "D", the Betweenness result would suggest that the demand for ice cream is made up of 62 x 5 = 310 students, which is not the case.

The *Distribute Weights* tool, instead, allows us to distribute the origin weight of "5" equally to all observer points on the way, such that the sum of the observer values still equals five. Each of the 62 observer points obtains a value of 5/62 = 0.08 (right image). If this sequence of spatially distributed demand points were used to illustrate demand for ice cream from students walking through Cambridge, then the total demand present would still be 5 students.



The tool requires origin points, destinations points as well as observer points, the latter of which receive the redistributed origin weights. If you don't have a pre-defined set of observer points available, the tool offers an option to create observer points for you at a chosen distance interval, placed on all of your network segments. Note, that if you only want to distribute weights on a local part of the network (e.g. to walking routes from a few origin locations to selected destinations), then you might want to first reduce the extent of the network to only the local segments around your question area (using the add/remove curves to network tool), otherwise the entire network will obtain observer points at your chosen intervals.

Once origins, destinations and observers are given, the tool offers the following options on the Rhino command line:



The **Search** option defines which destinations are used for each origin point – only a single nearest destination, all destinations, or all destinations that are within a given network radius from the origin.

The **Detour Ratio** option defines which travel routes are used to go from origins to destinations. The ratio describes the ratio between the allowable path length to the shortest available path length and it varies between "1" and "2". If "1" then the allowable path lengths are the same as shortest path lengths, and only shortest paths are used. If "1.1", then all paths that are up to 1.1 times the shortest path length to the destination (up to 10% longer than shortest path) are found and so on.

The **Weight** option defines which weight at the origin points is being redistributed to observer points.

The **Save Option** defines the name with which the new, re-distributed weights will be saved. To use a custom name, click on the option and set *Use* to *True* and type in a custom name (e.g. "d_weight"): 

Using a custom name allows you to keep track of multiple weight distributions. It also allows you to iteratively append multiple distributions to the same weight name if you set choose CleanTouchedWeights = False (see below).

The **Coefficient** variable determines what portion (%) of the origin weight is redistributed to observers. If you keep the default "1" coefficient, then 100% of the origins weights are redistributed. If you set it to "0.5", then only half the weights are redistributed, in which case the total sum of redistributed weights will equal half the sum of original weights you started with.

The **CleanTouchedWeights** option determines whether or not (True, False) the previous distributed weights at the observer points carrying the same weight name are over-written or not. If set to True, then previous results are over-written. If set to False, then new results are appended to previous results, allowing you to cumulatively add additional weight distributions to observers.

The example images below illustrate the tool in three steps, where demand weights are redistributed from original home locations to walking routes that lead to nearby transit stops.

The first image shows the original demand points, which in this case illustrate the number of households at multistory residential buildings. Warmer colors indicate more households.

The second image shows the transit stop destinations. There are three types of destinations – MRT, light rail and bus stops – which each can receive a different proportion of the original demand weights (e.g. 70% of HH go to MRT, 20% to light rail, 10% to bus stops).

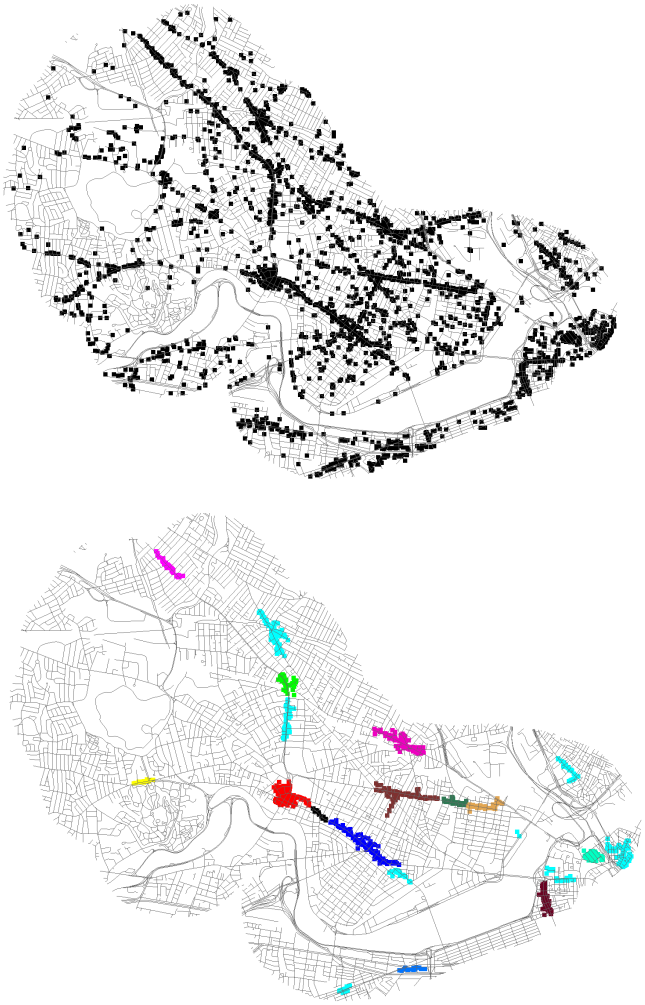
The third images shows the result, where the original demand weights are now spread at equal intervals along walking routes from homes to transit destinations. Note that the tool does not automatically erase any weights from your origin points, it only places newly distributed weights on observer points. The sum of redistributed weights on the observer points is identical to the sum of weights at the origin locations.

23. Clusters 

This tool finds spatial clusters of points along networks. It can be used to detect business clusters, street vendor clusters, residential foreclosure clusters, tree clusters, crime location clusters, or any other event aggregations along networks. Clusters are defined according to two user inputs: 1) the minimum number of points that constitute a cluster and 2) the maximum allowable network distance from each member of the cluster to at least one other member in the same cluster. For instance, a cluster can be defined as an aggregation that contains at least 25 members, where the maximum distance from each member is no more than 75 meters to at least on other in its cluster.

The example below illustrates the detection of well-known business clusters in Cambridge, MA using these inputs (*ClusterRadius* = 75m*, MinClusterSize*=25*)*, where each detected cluster has been grouped and differentiated by color. Points

that do not satisfy the above two cluster conditions are not included in the groupings.



In order to evaluate whether given points form clusters, a network is required (using *Add Curves to Network* tool), and points have to be added to a network as both origins and destinations (using *Add Origins* and *Add Destinations* tools). The user options on the command line include the following:

Cluster Radius <800> (MinClusterSize=*10* GroupAndCopy=*On* SaveWeight=*Off*):

Cluster Radius defines the maximum allowable network distance from each member of the cluster to at least one other member in the same cluster.

MinClusterSize defines the minimum number of points that constitute a cluster.

GroupAndCopy triggers whether the points that satisfy the above two cluster criteria are grouped by cluster and copied to the active layer.

SaveWeight defines whether cluster numbers are recorded as "Cluster" attributes in the weights of the original input points. This can be useful, for instance, for copying the weights out to Excel and counting or analyzing the clusters there. Note that only the points that are found to belong to a cluster obtain a "Cluster" attribute weight.

24. Graphic Options

This tool allows you to change the graphic appearances of UNA features: the color schemes of results, the display of labels, point connection lines etc. The following options are available on the command line:

```
Graphics ( Color=GreenToRed Results=Reach Weight=SumCount Mode=Weight Node=On NodeId=Off NodeD2=Off DotConnections=Off DotArrow=Off
DotLabel=On DotId=Off Dots=On Edges=On EdgeLebels=On Font=18 DotSize=14 ):
```

- Color allows you to choose different coloring schemes for visualizing UNA analysis results.
- Results option controls which analysis result is currently being displayed.
- Weight option controls which object weights (numeric attributes) are currently being displayed.
- Mode option toggles whether the graphics display analysis results or numeric object weights.
- Node option determines whether small black cross symbols are drawn at the dead-end nodes of the network.
- NodeId toggles Rhino node IDs on or off from the display mode. The NodeIds are network nodes with automatically assigned values that the user typically does not need to see.
- NodeD2 option turns on/off red warnings with a numeric value "2" , which tells you which nodes are degree two and that do not connect with each other (e.g. as in an overpass). (see Add Curves to Network tool for more).
- DotConnections option controls whether the blue, red and gray connection lines from all Origin, Destination and Observer points to the closest network edge are shown or not.
- DotArrow option allows you to further add blue/red/gray arrows to the network representation to show the direction of trips at each node.

- DotLabel controls whether numeric UNA analysis results are shown next to nodes or not. This is an important feature – turning DotLabels off will hide the analysis result numbers from display.
- DotId controls whether UNA ID numbers for each network node are displayed.
- Dots controls whether network nodes with resulting values are shown or hidden.
- Edges controls whether color-coded results are shown on network edges. This control only affects the Betweenness analysis, which can show betweenness results at the edge level.
- EdgeLabels option allows you turn numeric results on or off from edge. This control only affects the Betweenness analysis.
- Font controls the size of the font on result outputs on the screen.
- DotSize controls the size of the dots where UNA results are shown.

25. Paint weight color 

This tool allows you to assign the temporary weight or result colors that you visualize to the source objects as Rhino object colors. This can be useful if, for instance, you would like to export the resulting UNA color graphics out to other graphic software, such as Adobe Illustrator, AutoCAD, etc.

4 RELATED BIBLIOGRAPHY

Eppli, M., & Shilling, J. (1996). How Critical is a Good Location to a Regional Shopping Center? *Journal of Real Estate Research*, Vol. 12(3), 459–469.

Garrison, W. L., & Marble, D. F. (1962). *The Structure of Transportation Networks*. (U. S. A. T. Command, Ed.) (pp. 73–78). U.S. Army Transportation Command Technical Report.

Hensher, D. A. (2004). *Handbook of transport geography and spatial systems* (p. xxii, 672 p.). Amsterdam ; Boston: Elsevier.

Hillier, B. (1996). *Space is the machine : a configurational theory of architecture* (p. xii, 463 p., [8] p. of plates). Cambridge ; New York, NY, USA: Cambridge University Press. Retrieved from http://www.loc.gov/catdir/toc/cam023/95021500.html

Hillier, B., & Hanson, J. (1984). *The Social Logic of Space*. Cambridge: Cambridge University Press.

Hillier, B., Hanson, J., & Peponis, J. (1987). Syntactic Analysis of Settlements. *Architecture and Behaviour*, *3*(3), 217–231.

Jiang, B., Claramunt, C., & Batty, M. (1999). Geometric accessibility and geographic information: extending desktop GIS to space syntax. *Computers, Environment and Urban Systems*, *23*(2), 127–146. doi:10.1016/S0198-9715(99)00017-4

Kansky, K. J. (1963). *Structure of transportation networks: relationships between network geometry and regional characteristics* (p. x, 155 p.). Chicago, IL.

Mohsenin, M., & Sevtsuk, A. (2013). The impact of street properties on cognitive maps. *Journal of Architecture and Urbanism*, *Volume 37*(Issue 4), 301–309.

Okabe, A., & Shiode, S. (2001). SANET: A toolbox for spatial analysis on a network. *Journal of Geographical Analysis*, *Vol.38*(No. 1), pp.57–66.

Okabe, A., & Sugihara, K. (2012). *Spatial Analysis Along Networks: Statistical and Computational Methods (Statistics in Practice)* (p. 296). Wiley. Retrieved from http://www.amazon.com/Spatial-Analysis-Along-Networks-Computational/dp/0470770813

Porta, S., Crucitti, P., & Latora, V. (2005). The network analysis of urban streets: a primal approach. *Environment and Planning B*, *35*(5), 705–725.

Porta, S., Strano, E., Iacoviello, V., Messora, R., Latora, V., Cardillo, A., … Scellato, S. (2009). Street centrality and densities of retail and services in Bologna,Italy. *Environment and Planning B: Planning and Design*, *36*, 450–465.

Sevtsuk, A. (2010). *Path and Place: A Study of Urban Geometry and Retail Activity in Cambridge and Somerville, MA.* MIT, Cambridge.

Sevtsuk, A. (2012). Analysis and Planning of Urban Networks. In K. A. Zweig (Ed.), *Encyclopedia on Social Network Analysis and Mining*. Springer.

Sevtsuk, A. (2013). Networks of the built environment. In D. Ofenhuber & C. Ratti (Eds.), *[de]coding the city -- how "big data" can change urbanism*. Birkhäuser.

Sevtsuk, A., & Mekonnen, M. (2012). Urban Network Analysis Toolbox. *International Journal of Geomatics and Spatial Analysis*, *22*(2), pp. 287–305.

Sevtsuk, A., Mekonnen, M., Kalvo, R., & Amindarbari, R. (2014). Redundant Paths for Urban Network Analysis. *In Review*.

Stibbs, R., & Tabor, P. (1970). The Evaluation of Circulation in Buildings: A Mathematical Model. Cambridge: Cambridge University.

Tabor, P. (1976). Networks Distances and Routes. In L. March (Ed.), (pp. 366–367). Cambridge: MIT Press.

Xie, F., & Levinson, D. (2007). Measuring the structure of road networks. *Geographical Analysis*, *July 2007*.