City Form Lab

Better Design through Research | cityform.mit.edu

Urban Network Analysis Toolbox for Rhinoceros 3D

HELP

version 5.10.10.3 R5RS10

Andres Sevtsuk; Raul Kalvo
Contact:
cityformlab[at]mit.edu

Singapore University of Technology & Design in collaboration with MIT
8 Somapah Rd
Singapore, 487372
T: +65-64994557

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

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.

Download the UNA Toolbox folder from the following Dropbox link:

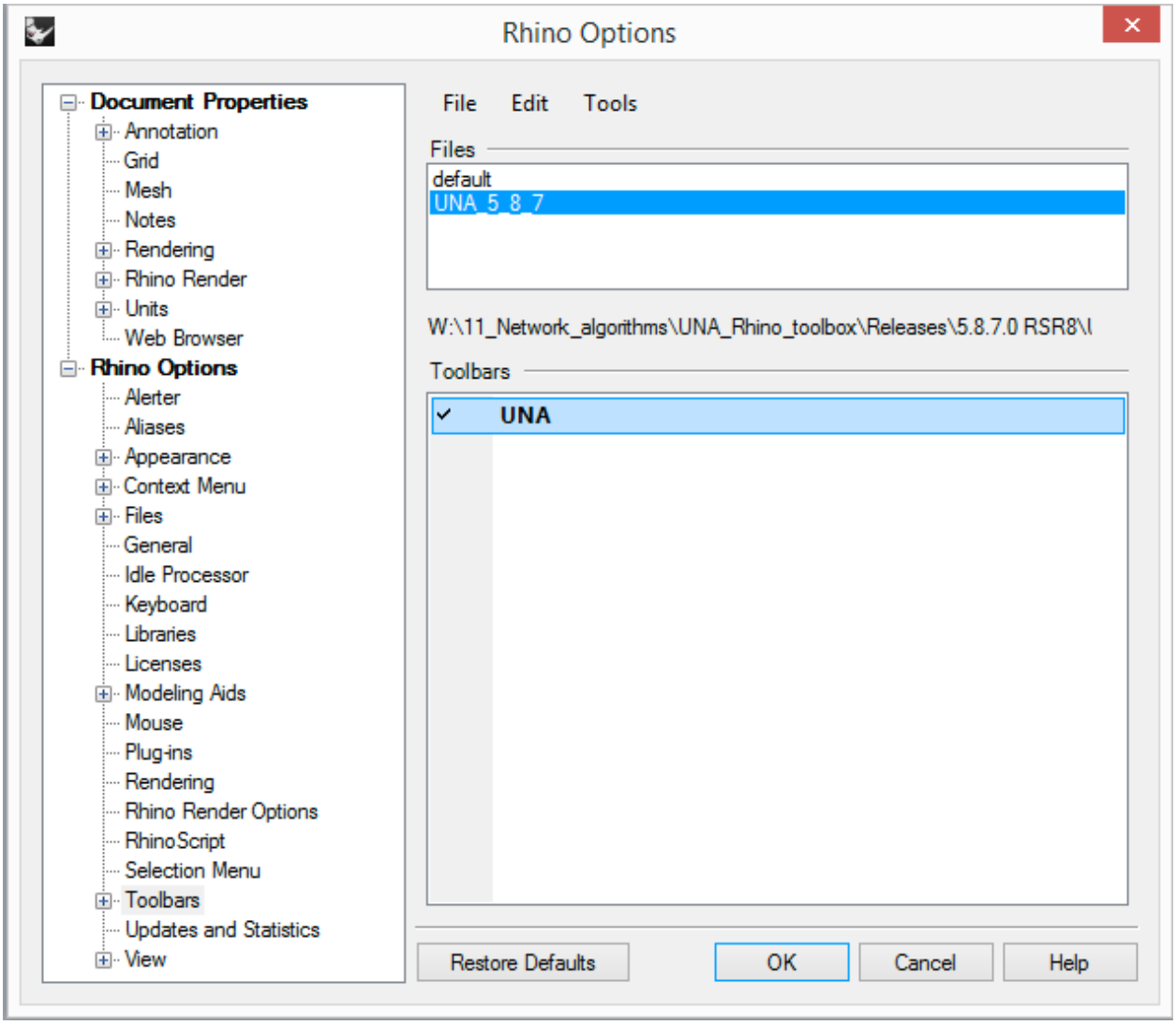
http://cityform.mit.edu/projects/una-rhino-toolbox

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

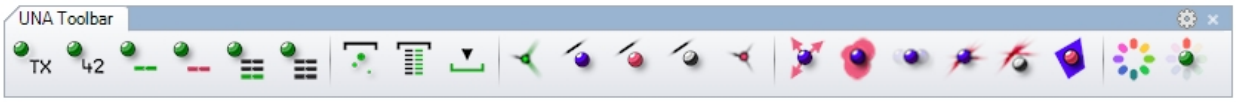
Note, that with some recent McNeel updates the .rhi installer for UNA may not be recognized as a program on windows. This is a known Rhino bug that McNeel is aware of. You can still install the UNA toolbox in one of two ways:

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:

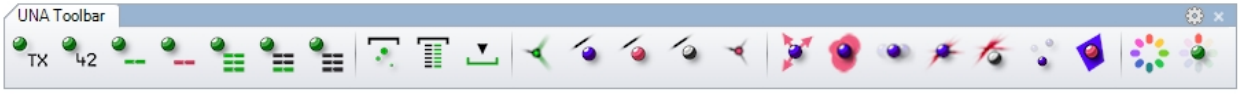


These are the tools that enable Urban Network Analysis in Rhino. You can also activate each of the tools from the Command Line. UNA tools typically start with the “una…” followed by the tool name for each function.

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.

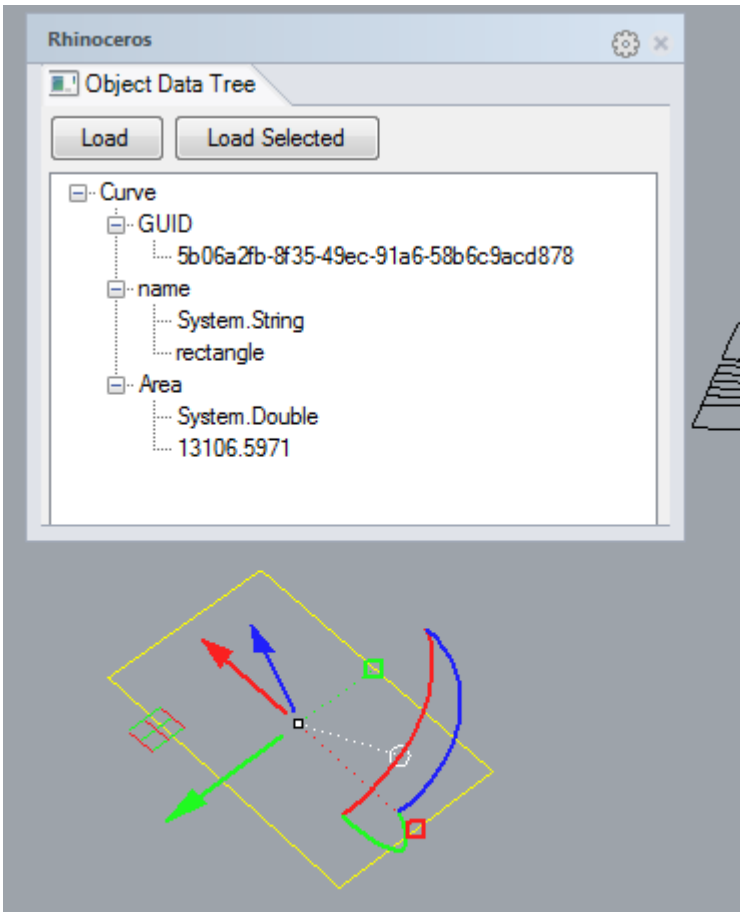
   After selecting the objects, the tool prompts you for the tag name by asking “Tag: “ on the command line. Type in the tag name that you want to remove. Note that you cannot remove GUID, which is a unique ID for each Rhino object.

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**

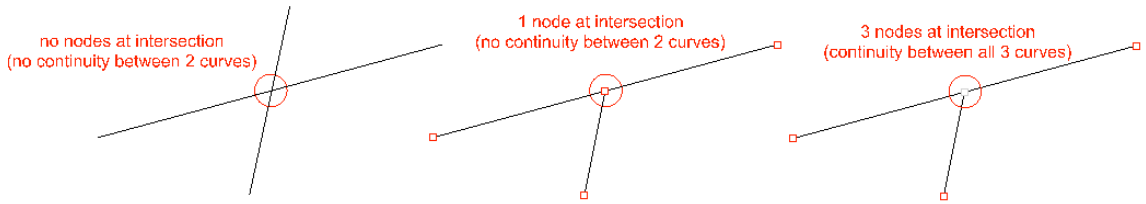
The way spatial networks are represented in the Rhino UNA toolbox is similar to the ArcGIS UNA toolbox (if you happen to have used it). Any network analysis requires at least two user inputs: first, an input analysis network and second, input locations. If you have been using our GIS UNA toolbox may 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.

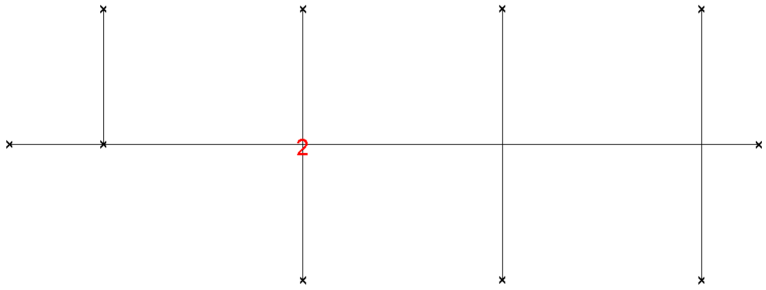
In order for network analyses to work, network curves need to provide continuity between the "input locations" you analyze. If your network curves do not share a common end node, then the "input locations" found on different network segments will not be topologically connected to each other. If one curve 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).

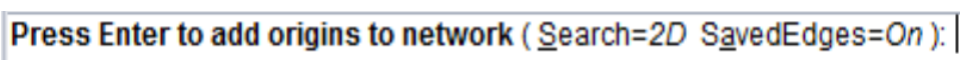


11. Add Origins

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.

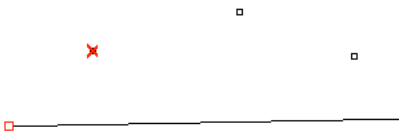
15. Centrality Indices

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

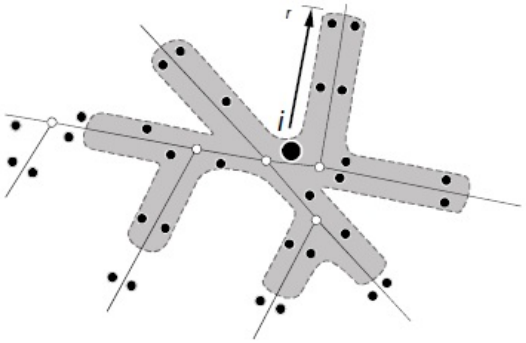
The "Reach" measure (Sevtsuk, 2010) captures how many surrounding points (e.g buildings, doors, bust stops etc.) each building reaches within a given *Search Radius* on the network.[1] The reach centrality, $R^r[i]$, of an Origin i in a graph G describes the number of Destinations in G that are reachable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

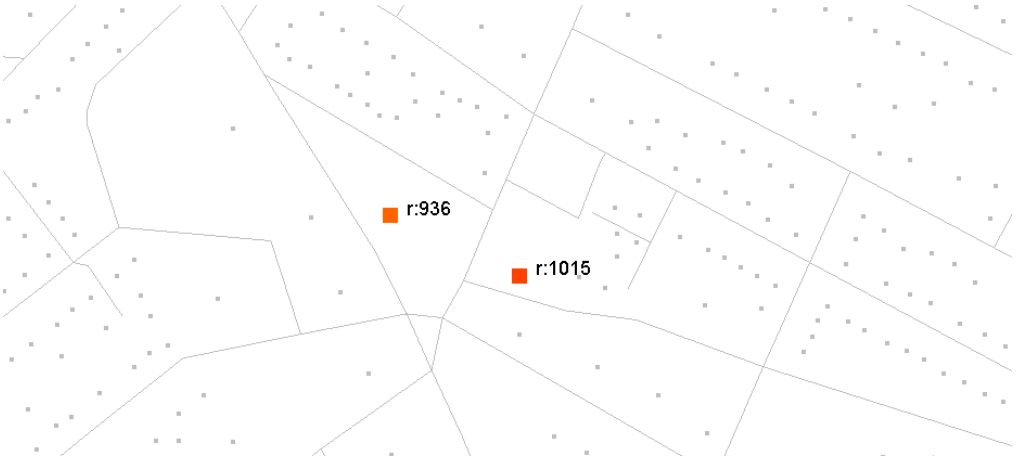
[1]The toolbox does not model buildings with multiple entrances, which can play an important role for building accessibilities in a real-life urban settings.

[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:

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

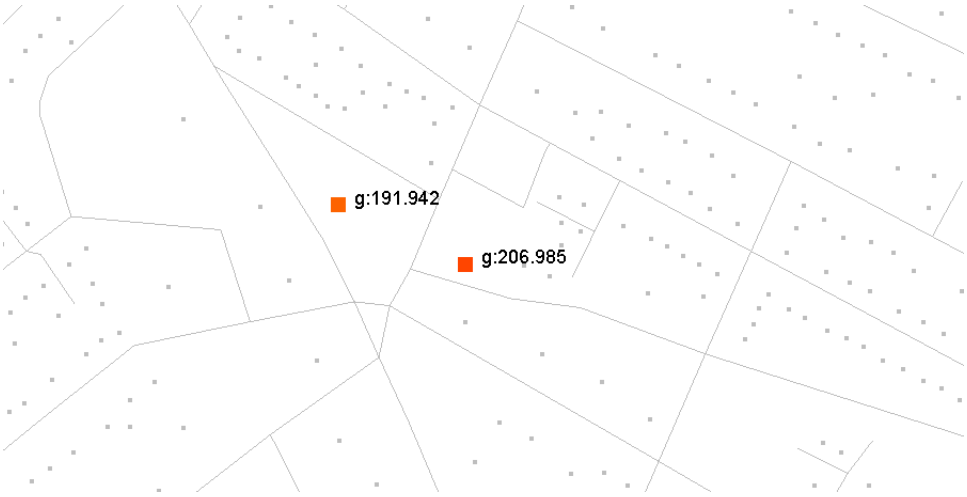
where $Gravity[i]^r$ is the Gravity index at Origin i within graph G at Search Radius r, $W[j]$ is the weight of Destination j, $d[i,j]$ is the geodesic distance between i and j, α is the exponent that can control the destination Weight or attractiveness effect, and β is the exponent for adjusting the 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 *Search 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.

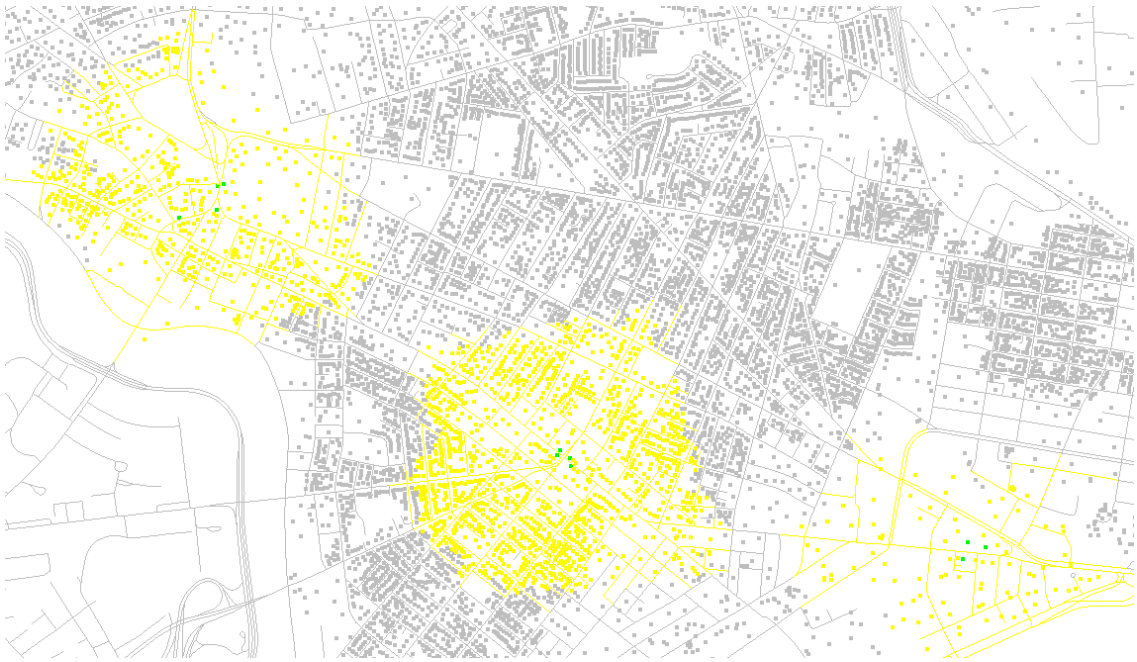
16.Service Area

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.

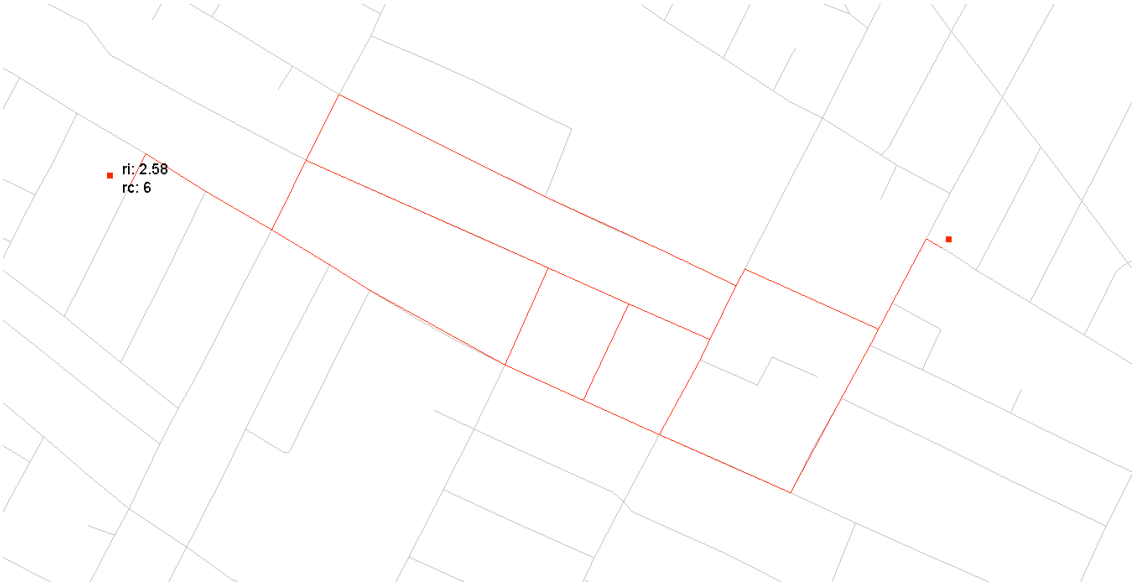
Note that the routes that the Redundancy Index finds are not necessarily simple routes – they may contain loops and retrace repeat nodes as described in the related paper (Sevtsuk, Mekonnen et al. 2014). The Redundancy Index and the Redundant Route count are also output by the Betweenness tool if a detour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

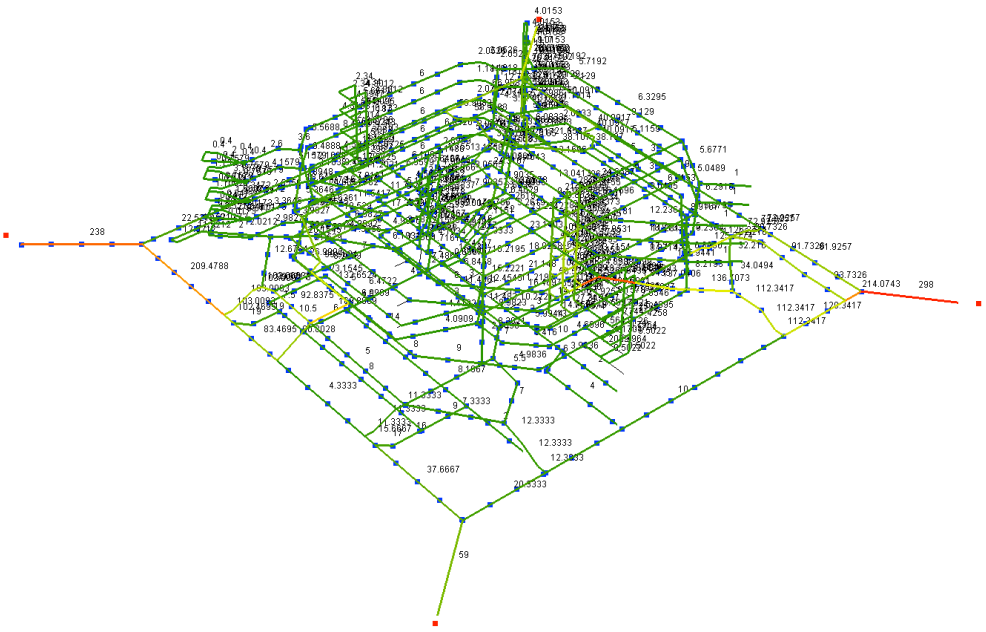
The Betweenness algorithm in the Rhino UNA toolbox has been specifically customized to make it useful and practical for estimating realistic pedestrian movement in spatial networks. Whereas a traditional Betweenness index would estimate trips from a set of origins to a set of destinations along shortest 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. 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.

1. 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

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*.