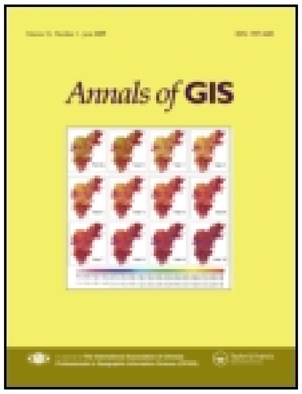


Annals of GIS
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tagi20

Estimating O-D travel time matrix by Google Maps API: implementation, advantages, and implications

Fahui Wang [a] & Yanqing Xu [a]
[a] Department of Geography & Anthropology, Louisiana State University, Baton Rouge, LA, USA

Available online: 23 Nov 2011

To cite this article: Fahui Wang & Yanqing Xu (2011): Estimating O-D travel time matrix by Google Maps API: implementation, advantages, and implications, Annals of GIS, 17:4, 199-209

To link to this article: http://dx.doi.org/10.1080/19475683.2011.625977

Annals of GIS
Vol. 17, No. 4, December 2011, 199–209

Taylor & Francis
Taylor & Francis Group

Estimating O–D travel time matrix by Google Maps API: implementation, advantages, and implications

Fahui Wang* and Yanqing Xu

Department of Geography & Anthropology, Louisiana State University, Baton Rouge, LA, USA

(*Received 19 August 2011; final version received 14 September 2011*)

Many spatial analysis tasks call for the use of travel time between multiple origins and destinations, that is, O–D travel time matrix. Commercial geographical information systems (GIS) software requires the input of a well-defined road network dataset and significant efforts in implementing the task. However, road network data are often outdated, miss critical road condition details, or are expensive to acquire; and skillful usage of related software is a major obstacle for researchers without advanced training in GIS. This research develops a desktop tool for implementing the task by calling the Google Maps Application Programming Interface (API). By doing so, we are able to tap into the dynamically updated transportation network data and the routing rules maintained by Google and obtain a reliable estimate of O–D travel time matrix. The results are compared with those computed by the ArcGIS Network Analyst module to demonstrate its advantages. A case study in accessibility analysis is presented to illustrate the implications.

Keywords: O–D travel time matrix; Google Maps API; network analysis; spatial analysis

1. Introduction

Estimation of travel time between a set of origins and a set of destinations (i.e., O–D travel time matrix) through a transportation network is a common task in spatial analysis. To list a few, spatial interaction modeling uses travel time between any pair of interacting places (Fotheringham and O'Kelly 1989); traffic demand forecasting relies on an accurate estimation of travel time among locations in various land uses (Black 2003); trade area analysis needs the travel time between each store and each residential area to define a store's customer base (Huff 2003); and accessibility measurement requires the travel time data between supply and demand locations (Luo and Wang 2003). In the absence of data of a transportation network or the computational power of geographical information systems (GIS), one has to resort to simple distance measures such as Euclidean distance or Manhattan distance (Wang 2006, pp. 19–20). These simple distance measures only need the input of geographic coordinates of origins and destinations and use simple mathematical formulas to calibrate, but are primitive indices of travel impedance.

Travel time estimation is based on a transportation network. A transportation network consists of a set of nodes (or vertices) and a set of arcs (or edges or links) that connect the nodes. An arc may also be directed (e.g., one-way street). Travel time estimation is usually performed to find the shortest time from a specific origin to a specific destination through a series of nodes and arcs connecting them (route) on the network, often referred to as the *shortest route problem*. Among various methods for solving the problem (Papadias *et al*. 2007), the label-setting algorithm (Dijkstra 1959) remains the most popular. Given the speed on each arc, turning restrictions, and time penalty of each turn, the algorithm seeks the route that minimizes the total travel time. Therefore, data requirements for defining a transportation network include many network elements such as link impedances, turn impedances, one-way streets, overpasses, and underpasses (Chang 2004, p. 351). Putting together such a network dataset requires extensive data collection and processing, which can be very expensive or infeasible for some applications. For example, a road layer extracted from the US Census Topologically Integrated Geographic Encoding and Referencing (TIGER)/Line files does not contain nodes on the roads, turning parameters, or speed information. When such information is not available, one has to make various assumptions to prepare the network dataset. For example, Luo and Wang (2003) assigned speeds to different roads according to the census feature class codes (CFCC) and whether in urban, suburban, or rural areas; and Wang (2003) used regression models to adjust travel speeds by land use intensity (business and residential densities) and other factors. All these attempts are by approximation and compromise the accuracy and

*Corresponding author. Email: fwang@lsu.edu

ISSN 1947-5683 print/ISSN 1947-5691 online
© 2011 Taylor & Francis
http://dx.doi.org/10.1080/19475683.2011.625977
http://www.tandfonline.com

reliability of travel time estimation. Furthermore, learning the related GIS software is no trivial task.

On the other hand, some online sources such as Google Maps (maps.google.com), Map Quest (www.mapquest.com), and Rand McNally (www.randmcnally.com) provide a convenient stop for solving the shortest route problem for the public. However, the task of common users is to find the travel time (and directions) of one route (perhaps several routes). The challenge for researchers, as outlined above, is to obtain a travel time matrix from many origins to many destinations, sometimes a sizeable matrix of thousands of routes or more. The purpose of this research is to develop a tool to automate the process. Specifically, a desktop tool is developed to implement the task by calling the Google Maps Application Programming Interface (API). By doing so, we are able to utilize the transportation network data as well the routing algorithm by Google behind the scene and obtain a reliable estimate of O–D travel time matrix with minimal data preparation and GIS software knowledge.

Section 2 explains how the Google Maps API tool is developed. Section 3 compares the results with those by ArcGIS Network Analyst module to demonstrate its advantages.Section 4 uses a case study on hospital accessibility analysis to illustrate the implications of various travel time estimates. A brief summary concludes this article and discusses some limitations of the developed tool.

2. Using the Google Maps API to calibrate O–D travel time matrix

Google Maps is a popular web-based mapping service launched by Google in early 2005 to provide a highly responsive visual interface using AJAX technologies. Shortly after that, Google launched the Google Maps API, a JavaScript API, to allow the customization of online maps (Taylor 2005). The Google Maps API enables one to embed the Google Maps site into an external website and overlay specific data on to the site (Mercurio 2008). In the following, we will discuss how to use the Google Maps API to estimate the travel time between origins and destinations without reloading the web page or displaying portions of the map.

As shown in Figure 1, the process begins with data preparation. As explained later in this section, the location information as geographic coordinates is fed into the Google Maps for geocoding. Therefore, both the origin and destination layers need to be point features in a projection of geographic coordinates. The data are fed into a tool in Python that automates the following process. In each round, travel time is estimated between one origin and one destination by calling the Google Directions API. The iterations stop when the program reaches the last combination of origins and destinations. The result is saved in an ASCII file listing each origin, each destination, and travel time between them. The core of the process is the Python program as discussed below.

The Python programming language is chosen here since many GIS datasets are in ArcGIS format and ArcGIS 10 has adopted Python as its native scripting language (ESRI 2010). Python is a free, open-source, and cross-platform programming language with efficient high-level data structures and a simple yet effective approach to object-oriented programming. The Environment Systems Research Institute (ESRI) has created the *ArcPy* module for Python to provide access to geoprocessing environment settings. The Python program *traveltime.py*, shown in the Appendix, is written to implement the tool.

The program begins by calling the *ArcPy* module. It then calls the *urllib* module that provides a high-level interface for fetching data across the World Wide Web. The *urllib* module enables us to access the Google server and use its Directions API. The program also calls for the *time* module in order to define *time.sleep*. In our case, we set it to 3 seconds as the time to pause the computation

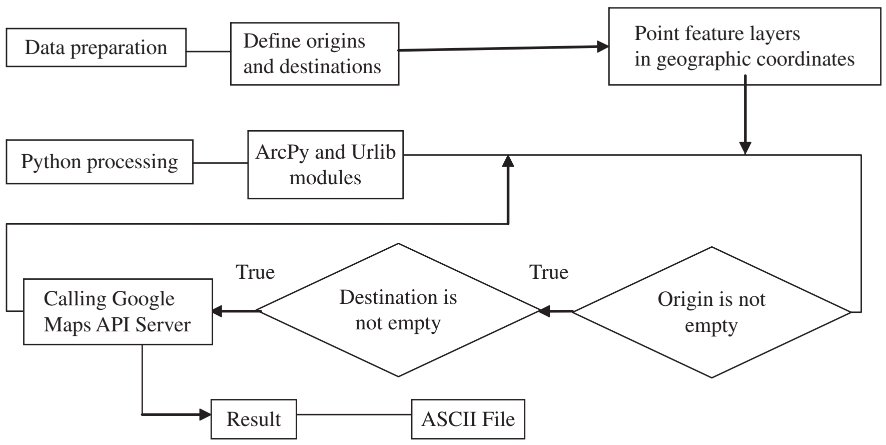


Figure 1. Travel time estimation process by Google Maps API.

process after each request (i.e., completing one route computation). Google also has its internal limit for 'sleep time' (unknown to users) so that the server would not be accessed too frequently in a short time. The 3-second setting is a rule of thumb that helps pace the requests and reduces the chances of receiving the error message 'OVER_QUERY_LIMIT'. On rare occasions, if the user does get the error message, the program skips the request, flags the record with 'NA' in the result table, and moves on to complete the rest of the requests. The user can revisit the problem record(s) later, complete the missed request(s), and update the final travel time table. This is likely caused by the limitation of Google, particularly when the O–D matrix gets large, for users without a paid license to Google Maps API Premier. 'Use of the Google Geocoding API is subject to a query limit of 2,500 geolocation requests per day' to prevent abuse (http://code.google.com/apis/maps/documentation/geocoding/). A licensed Premier user may perform up to 100,000 requests per day (http://code.google.com/apis/maps/documentation/distancematrix/).

The program then defines two input files (*fromFile* and *toFile*) and one output file (*ResultFile*). The *From* parameter means the origins and the *To* parameter means the destinations, both of which are point feature layers. Both input features need to be in a projection of geographic coordinates. The *Result* parameter means the output file of travel time matrix as a text file. The program uses the function *SearchCursor* to move from one request of route computation to next and control the iteration. The function *SearchCursor* establishes a read-only cursor on a feature class or table and extracts the information such as id and coordinates from the input files. The function continues to read the information of origins and destinations through row objects and completes the task until it reaches the end of the files.

The major component of the program is to use the Google Directions API to calculate the best route between two locations by a hypertext transfer protocol (HTTP) request. Directions specify origins and destinations as latitude/longitude coordinates or text strings. Our program uses geographic coordinates that are simple, accurate, and fast for geocoding. As long as the input features are in geographic coordinates, there is no need to have the coordinates physically residing in their attribute tables. The *ArcPy* module automatically extracts the location information from the input features. In the uniform resource locators (URLs) of the request, three parameters are required: *origin*, *destination*, and *sensor*. The parameters *origin* and *destination* are, respectively, the trip origins and destinations defined by latitude/longitude values. The parameter *sensor* indicates whether the direction request comes from a device with a location sensor and takes a value 'true' or 'false'. In our case and for the purpose of most research, its value is set to 'false'.

The tool is added to ArcToolbox in ArcGIS. A user interface is shown in Figure 2. A user only needs to define two point features in geographic coordinates ('From' and 'To') as inputs and name the resulting text file under 'Result' and click 'OK' to execute it. The total computation time depends on the size of O–D travel time matrix, the preset pause time between requests, and the Internet connection speed. In our test with 2314 records, the total computation time was less than 15 minutes.

3. Advantages of travel time estimation by Google Maps API

The ArcGIS Network Analyst, in particular its OD Cost Matrix function, is often used by researchers to calibrate the O–D travel time matrix (http://www.esri.com/software/arcgis/extensions/networkanalyst/). Figure 3

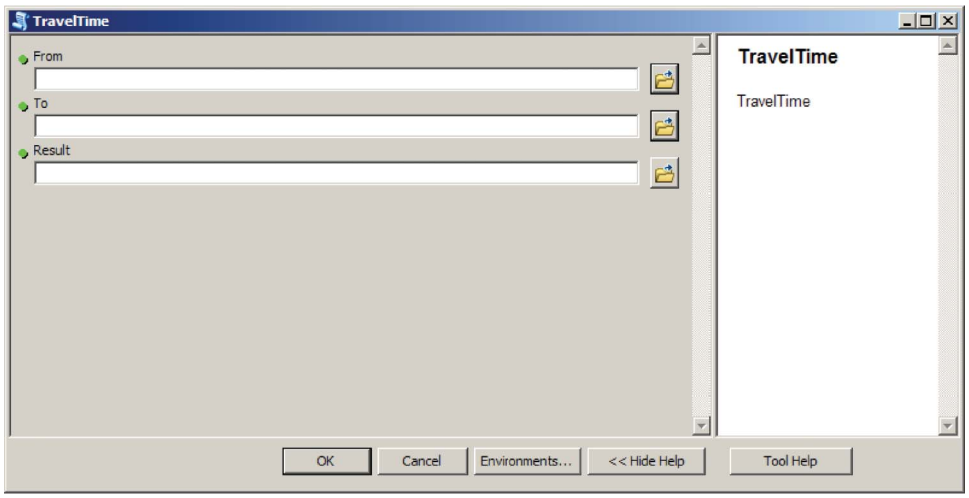


Figure 2. The Google Maps API tool user interface.

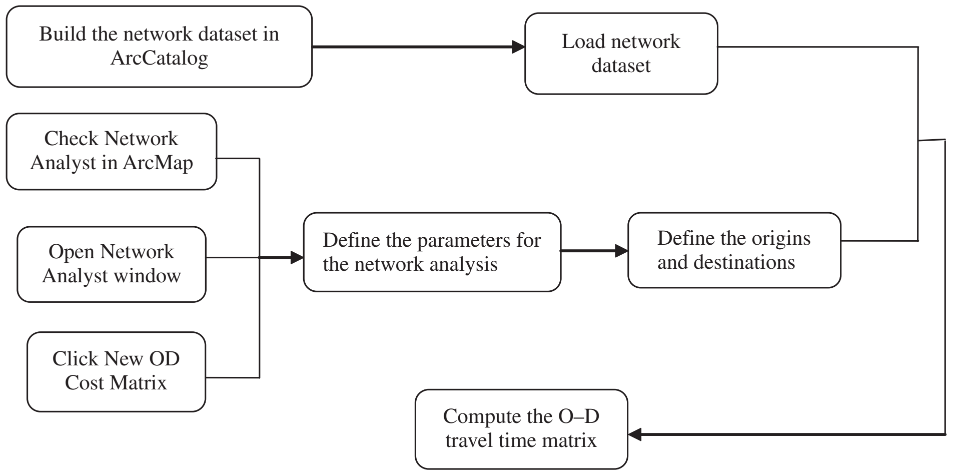

Build the network dataset in ArcCatalog
Load network dataset
Check Network Analyst in ArcMap
Open Network Analyst window
Click New OD Cost Matrix
Define the parameters for the network analysis
Define the origins and destinations
Compute the O–D travel time matrix

Figure 3. Travel time estimation process by ArcGIS Network Analyst.

illustrates the process for implementing the task in ArcGIS. The process begins with building the network dataset in ArcCatalog, where a user chooses the settings for connectivity, modeling turns, and others and specifies the network attributes such as length, travel time, and others. Then, in ArcMap, the user activates the Network Analyst extension and its OD Cost Matrix function. After loading the predefined network dataset to the active project, various network parameters (impedance, distance unit, etc.) need to be defined. In the Network Analyst window, one can use the origins feature to define 'Facilities' and the destinations feature to define 'Incidents' and then choose the OD Cost Matrix tool to solve the problem and save the result.

As in most applications, 'modeling is as good as the data get'. The key to quality travel time estimation is the network data. The following uses a study area in East Baton Rouge Parish of Louisiana (Figure 4) to examine how the results by ArcGIS and Google Maps API differ. Parish is the county unit in Louisiana. The study area is simply referred to as Baton Rouge hereafter. The network data used by ArcGIS are extracted from the data DVDs that came with the ArcGIS 10.0 release, more specifically, StreetMap North America. The road network data are based on the TomTom (TeleAtlas) 2005 version 7.2 data (according to personal communication with James Shimota of ESRI on 28 June 2011). On the other side, data used in the Google Maps are fairly updated. Most of this study was conducted in the summer of 2011. In the case study reported here, we used the data generated by the Google Maps API tool on 8 June 2011.

In comparison to the ArcGIS Network Analyst approach, at least four advantages are identified in using the Google Maps API. Note that our discussion below is limited to our experiment of using the aforementioned dataset in ArcGIS 10.0. It does not apply to one with access to more recent and extensive datasets such as those from TeleAtlas (www.teleatlas.com) that contain data of 'speed profiles' to capture congestion effects.

(1) *The Google Maps API approach does not need the preparation of a network dataset*.
An important step in modeling the OD cost matrix in ArcGIS is to prepare the network dataset including the extraction of data for the study area and defining network settings and attributes. In addition to the time investment, this requires the users to be knowledgeable about the transportation network analysis and familiar with the road network data structure. The API approach taps into the network data residing in a Google server.

(2) *The Google Maps API approach uses more updated road data*.
As explained above, the road network dataset that came with ArcGIS 10 was based on the data in 2005. Google updates its data more frequently, usually twice a month (www.gearthblog.com/blog/archives/2010/10/how_often_does_google_update_the_im.html). Figure 5(a) and (b) show an example where the Ben Hur Road near the Louisiana State University (LSU) Fireman Training Center in the study area changed recently. The road was straight on the ArcGIS StreetMap (Figure 5(a)), but a recent development project led to the curved road, captured by the Google Maps (Figure 5(b)).

(3) *The Google Maps API approach accounts for road congestion*.
The road network data used in ArcGIS 10 contain the speed limit for each road segment, which is assumed to be the travel speed by the Network Analyst module. For illustration, we choose the route from the GSRI Avenue (at geographic

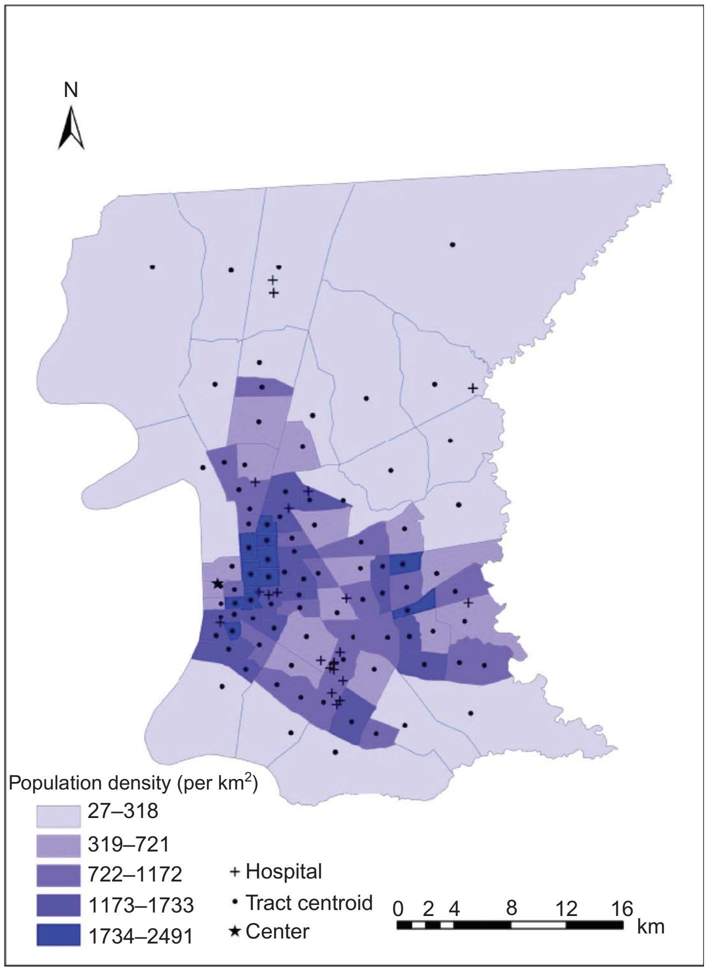


Figure 4. Population density in census tracts and hospitals in Baton Rouge.

coordinates 30.363512, −91.150997) to the Earl K. Long Medical Center (at geographic coordinates 30.5783, −90.9942) as an example. Figure 6(a) shows the path by ArcGIS on the StreetMap, and Figure 6(b) shows the path on the Google Maps. The two routes are similar. However, the travel time is 30 minutes by ArcGIS, but 40 minutes by Google. The difference is significant. A close examination of travel speed on each road segment shows that the travel speeds on roads around the LSU and the downtown area were much slower than the posted speed limits.

(4) *The Google Maps API approach considers the difference between peak hours and off-peak hours*. It is known that Google now enables one to estimate travel time in rush-hour traffic in a limited set of metropolitan areas (http://google-latlong.blogspot.com/2007/08/how-long-will-it-take-at-rush-hour.html). Google also attempts to predict traffic conditions on a certain day and time based on the live traffic data collected on a daily basis (Schwartz 2010). Our experiments in the study area indicated that travel time reported by Google Maps differed according to the time of the day when the computation requests were issued. Figure 7 shows the traffic condition along a route in Baton Rouge.

To further highlight the differences of travel time estimated by the two methods, we have computed the travel time from each census tract centroid to the city center (commonly recognized as the State Capitol Building) (see Figure 4). Figure 8 shows that travel time by the Google Maps API approach is consistently longer than that by the ArcGIS Network Analyst approach. The estimated travel time by either method correlates well with the (Euclidean) distance from the city center (with a $R^2 = 0.91$ for both methods). However, the regression model of travel time against corresponding distances by Google has a significant intercept of 4.68 minutes (vs. a negligible 0.79-minute intercept in the model of travel time by ArcGIS). The 4.68-minute intercept by Google probably reflects the elements of starting

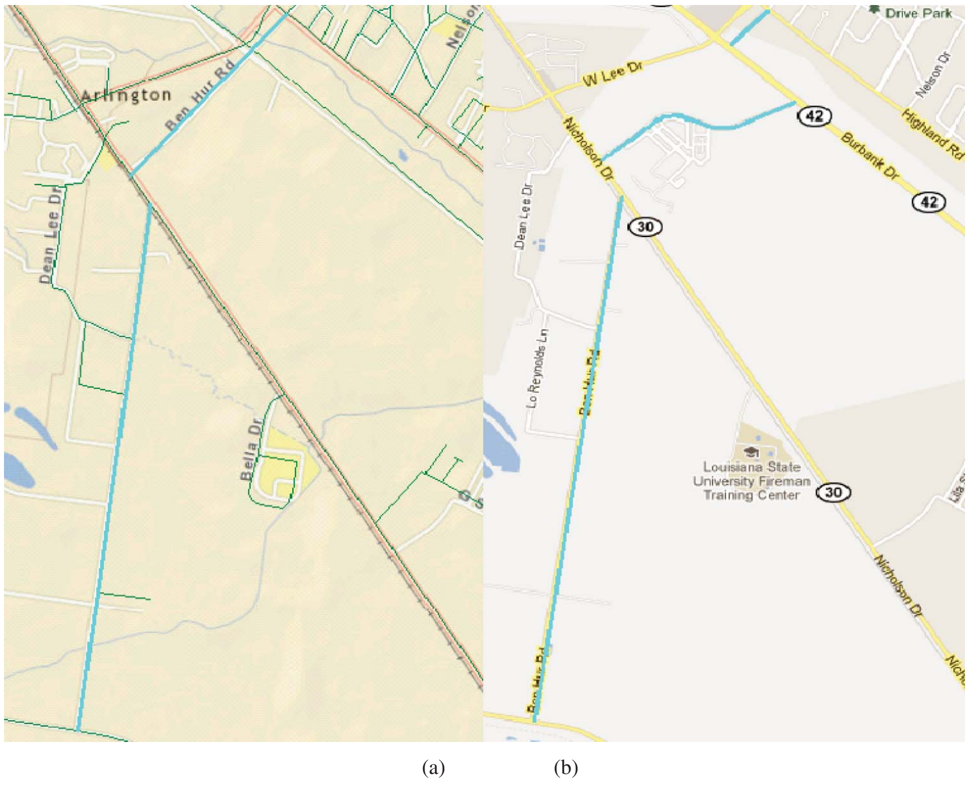


Figure 5. Road network near LSU: (a) ArcGIS StreetMap and (b) Google Maps.

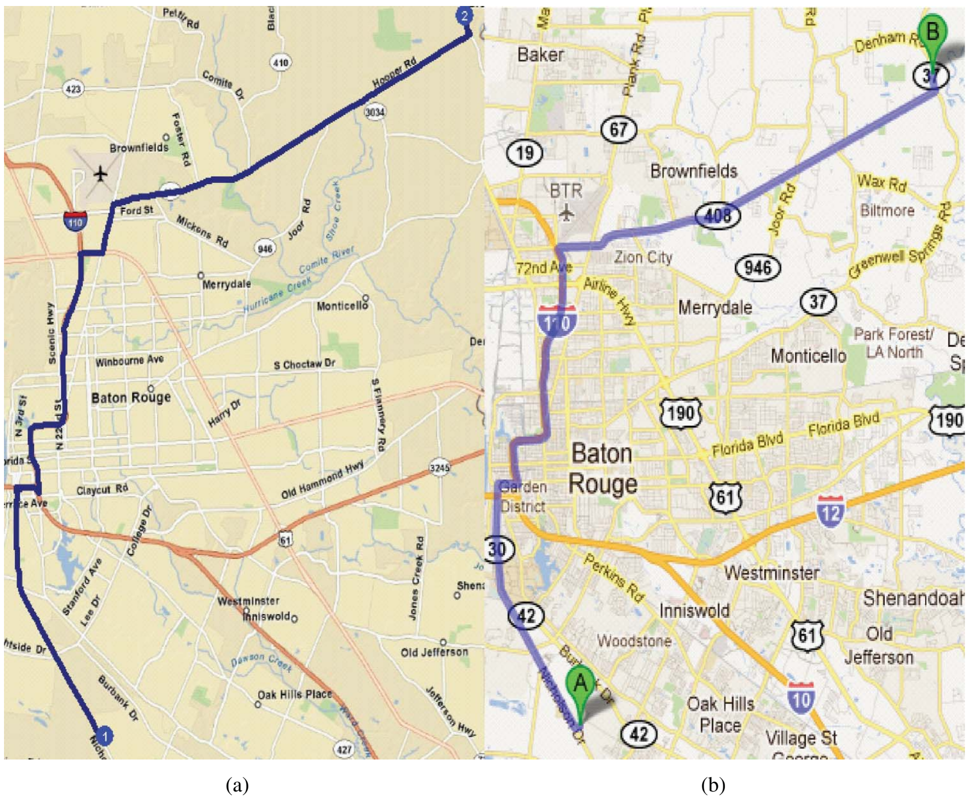


Figure 6. A O–D route on (a) ArcGIS StreetMap and (b) Google Maps.

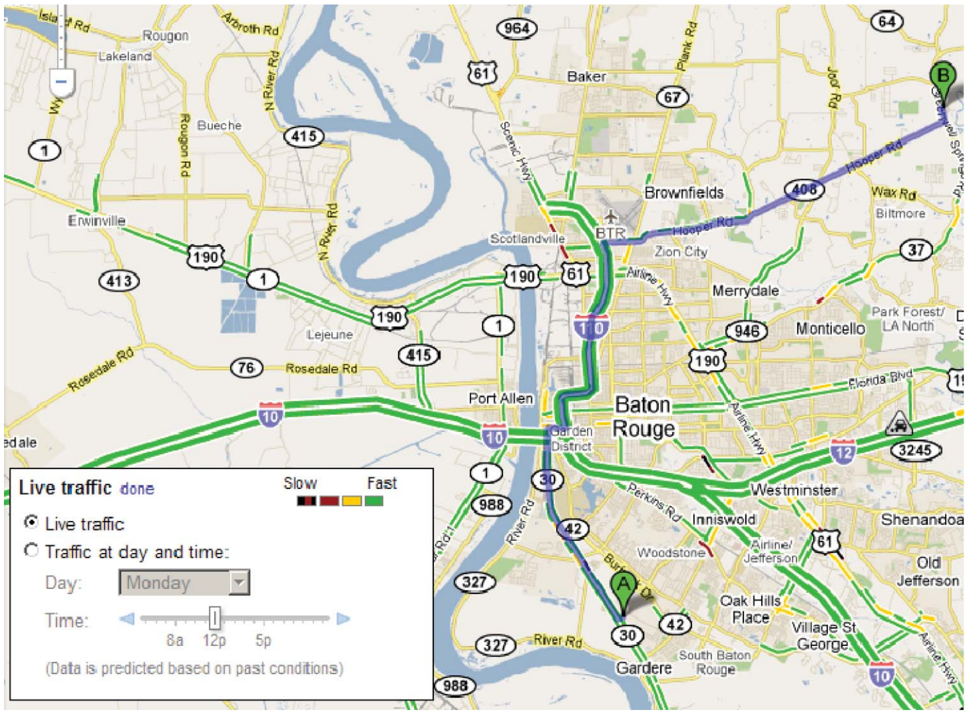


Figure 7. Traffic condition in Baton Rouge.

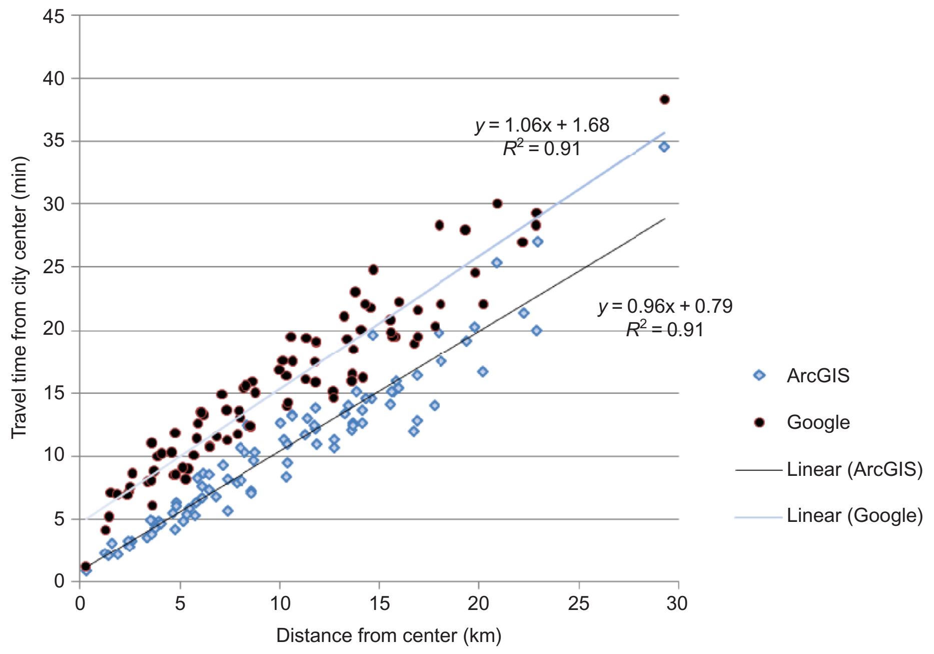


Figure 8. Estimated travel time from the city center by ArcGIS and Google.

and ending time on a trip (getting on and off a street on the road network). This is consistent with our daily travel experience and also empirical data. According to Wang (2003, p. 258), this 'inertia' time (also considered as 'intrazonal travel time') was reported as high as 11 minutes based on the Census Transportation Planning Package (CTPP) data in urban areas. The regression model for the travel time by Google also has a slightly steeper slope (1.06) than the slope in the model for the travel time by ArcGIS (0.96), but this difference is minor. Figure 9 displays how the difference, measured as [(travel time by Google – travel time by ArcGIS)/travel time by ArcGIS] in percentage, varies with distance from the city center. Clearly, the difference declines exponentially with distance from the city center.

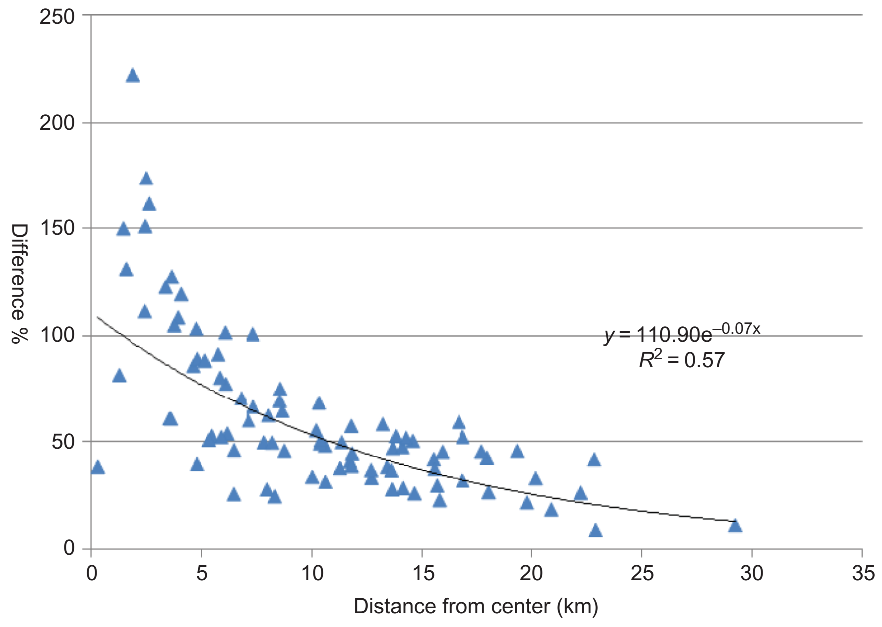


Figure 9. Spatial pattern of differences in estimated travel time by ArcGIS and Google.

It is possibly attributable to a combination of two effects: the extra 'inertia' time discussed above and slower speeds in the downtown area, both captured by the Google Maps API approach.

4. Case study in applying estimated travel time in assessment of hospital accessibility

This section presents a case study illustrating the application of estimating O–D travel time matrix in accessibility analysis. Accessibility refers to the relative ease by which the locations of activities, such as work, school, shopping, recreation, and health care, can be reached from a given location. It is a classic issue in geography. For the purpose of emphasizing the impact of travel time estimation, this case study does not consider issues such as the match ratio between supply and demand or the distance decay effect in spatial interaction. Readers who are interested in more advanced models of accessibility may refer to some recent work such as McGrail and Humphreys (2009) and Dai and Wang (2011). When the capacity of supply (i.e., hospital sizes in terms of numbers of beds or physicians) is unknown or of less concern, the main concern of accessibility is the travel time from residents to hospitals (Brabyn and Gower 2003). Here, accessibility is measured as the average travel time by automobile between a residential location and all hospitals in the study area, that is, Baton Rouge.

The study area has 89 census tracts with total population close to a half million and 26 hospitals. Locations of the census tracts are represented by their population-weighted centroids (based on the block-level population), and the hospitals are geocoded according to their addresses (see Figure 4). The 89 census tract centroids serve as origins, and the 26 hospitals are destinations. Therefore, the number of O–D trips is $89 \times 26 = 2314$.

We used both the Google Maps API and the ArcGIS Network Analyst methods to estimate the O–D matrix, as discussed in the previous two sections. One technical issue merits some discussion. In executing the Google Maps API tool, we noticed that the Google server skipped a couple of requests due to the system error discussed in Section 2. The problem persisted in numerous experiments, and the skipped requests varied each time. Even though our number of requests (i.e., 2314) is less than the daily cap of 2500 requests set by Google, we suspect that this is a common problem for users without a paid premier license to Google Maps API. Our strategy was to divide the requests into several blocks, implement the tool piece by piece, and eventually integrate the results together.

Similar to the results discussed in Section 3, the results by the two methods are overall consistent with each other, and most (2269 out of 2314) travel time results estimated by the Google Maps API are larger than those by the ArcGIS Network Analyst. The average travel time between census tract centroids and hospitals is 17.9 minutes by Google and 13.3 minutes by ArcGIS. However, such a discrepancy is not uniform or proportional across the study area. The following examines the impact on the spatial pattern of accessibility.

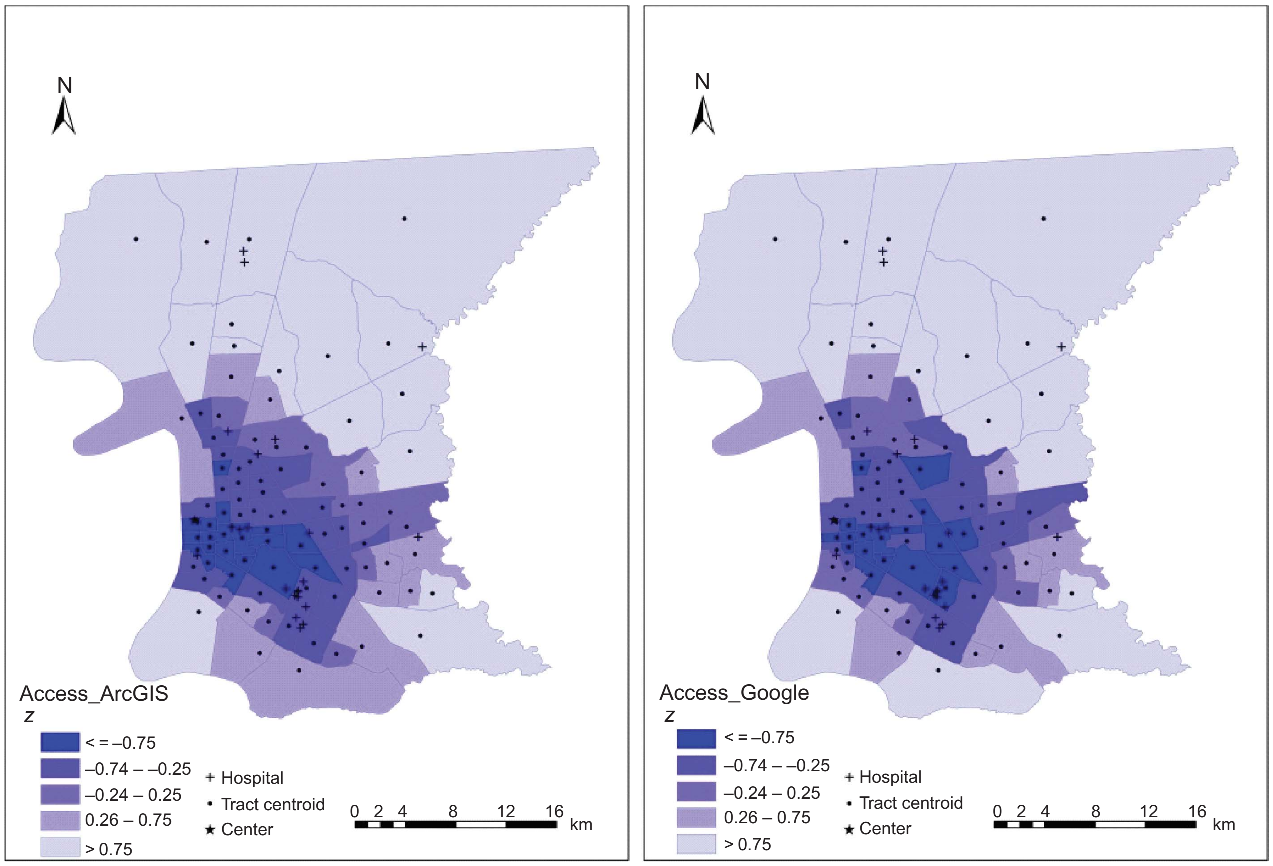
As stated earlier, accessibility in this case is simply the average time between a census tract centroid and all 26 hospitals. Since the time by Google is systematically longer than the time by ArcGIS, the comparison based on the time itself is less meaningful. Accessibility is to measure *relative* ease of reaching an activity or opportunity. Therefore, each series of average travel time to hospitals needs to be standardized (with a zero mean and standard deviation of 1) to be comparable. The resulting (standardized) *Z* value reflects the relative accessibility. A *higher Z* value corresponds to more travel time to hospitals and thus a *poorer accessibility*.

Figure 10(a) and (b) shows the accessibility *Z* scores by the ArcGIS and Google methods, respectively. Darker colors correspond to lower and negative *Z* scores, that is, average travel time to hospitals below the areawide mean, and better accessibility. The accessibility is the highest around the Essen medical campus with several hospitals (southeast of the city center) and declines outwards. The patterns are generally consistent on the two maps. However, there are differences, particularly in the middle-range-distance areas from the city center. Figure 11 maps the difference (i.e., *Z* score by Google – *Z* score by ArcGIS). Cold colors correspond to negative values (lower *Z* scores by Google than by ArcGIS) and indicate that the accessibility in the area based on the Google time is better than that suggested by the ArcGIS time. Warm colors indicate otherwise. From Figure 10, most areas toward northeast are in cold colors and thus indicate that Google time tends to suggest better accessibility than ArcGIS time does. The opposite can be said on the southwest corner.

In summary, the travel time estimated by different methods may lead to somehow different assessments of accessibility patterns.

5. Concluding remarks

Estimation of travel time between a set of origins and a set of destinations through a transportation network is a common task in spatial analysis. Calibrating the O–D travel time matrix in a commercial GIS package requires extensive data collection and processing to prepare the road network and also adequate knowledge of the software to implement related tools. Both are no trivial efforts. This research has developed a desktop tool to complete the task by calling the Google Maps API. The Python program automates the process by reading the layers of origins and destinations in geographic coordinates, executing a HTTP request to access the Google Maps and calibrate the



(a) (b)

Figure 10. Accessibility standardized *Z* score (a) by ArcGIS and (b) by Google Maps API.

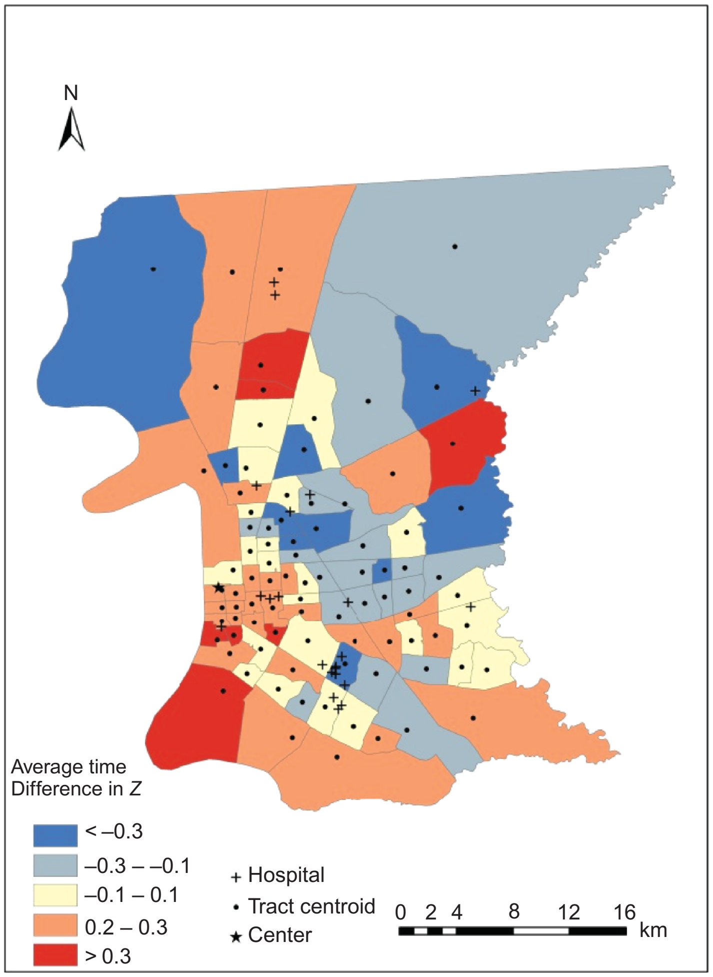


Figure 11. Difference in accessibility scores.

travel time between each O–D pair, and saving the results in an external ASCII file. By comparing the approach with the commonly used ArcGIS Network Analyst module, several advantages are identified: no need of preparing a road network, using a more updated road network, and accounting for congestion in high-traffic areas and peak hours. Our case study in accessibility analysis indicates that an accurate estimate of travel time is essential in spatial analysis.

The Google Maps API approach is not free of concerns. Our experiments have revealed some limitations. First of all, an ordinary user without a paid license to Google Maps API Premier is subject to a daily query limit of 2500 geolocation requests and is likely to experience some 'hiccups' in executing the tool. We welcome feedbacks from users with a Google Maps API Premier license. Second, all data used in the computation are maintained by Google, and thus a user has neither control over its quality nor any editing rights. While the Google's road network has a reputation of good quality, no data are completely free of errors. Most users may not care about any modification rights of network data, but the lack of data transparency is nevertheless a drawback for many advanced researchers in spatial analysis. Furthermore, the tool can only generate the contemporary travel time by accessing the most updated road network data in Google. In some cases, researchers need the travel time in the past. Such a task is only feasible by using a historical road network. Finally, the tool is currently implemented in a desktop GIS environment. It is our plan to develop an online version in the near future.

Acknowledgements

The support by the National Natural Science Foundation of China to Wang (No. 40928001) and a research grant to Xu by the South Central Arc User Group (SCAUG) are gratefully acknowledged. Comments by two anonymous reviewers and the editor helped us prepare the final version of this article.

References

Black, W.R., 2003. *Transportation: a geographical analysis*. New York: Guilford.

Brabyn, L. and Gower, P., 2003. Mapping accessibility to general practitioners. *In*: K., Omar and S, Riceds. *Geographic information systems and health applications*. Hershey, PA: Idea Group Publishing, 289–307.

Chang, K.T., 2004. *Introduction to geographic information systems*. 2nd ed. New York: McGraw-Hill.

Dai, D. and Wang, F., 2011. Geographic disparities in spatial accessibility to food stores in southwest Mississippi. *Environment and Planning*, B38, 659–677.

Dijkstra, E.W., 1959. A note on two problems in connection with graphs. *Numerische Mathematik*, 1, 269–271.

Environment Systems Research Institute, Inc. (ESRI), 2010. *Getting started with python in ArcGIS 10* [online]. Available from: http://www.esri.com/library/fliers/pdfs/python-in-arcgis10.pdf [accessed 7 October 2011].

Fotheringham, A.S. and O'Kelly, M.E., 1989. *Spatial interaction models: formulations and applications*. London: Kluwer Academic.

Huff, D.L., 2003. Parameter estimation in the Huff model. *ArcUser*, October–December, 34–36.

Luo, W. and Wang, F., 2003. Measures of spatial accessibility to health care in a GIS environment synthesis and a case study in the Chicago region. *Environment and Planning*, B30, 865–884.

McGrail, M.R. and Humphreys, J.S., 2009. A new index of access to primary care services in rural areas. *Australian and New Zealand Journal of Public Health*, 33, 418–423.

Mercurio, R., 2008. Improving operation, marketing and customer service with Google maps. *Malaya business insight* [online]. Available from: http://www.malaya.com.ph/june08/info1.html [accessed 7 October 2011].

Papadias, D., Zhang, D., and Kollios, G., (eds.), 2007. Advances in Spatial and Temporal Databases, *Proceedings of 10th International Symposium, SSTD*, 16–18 July, Boston, MA, Lecture Notes in Computer Science 4605. Berlin: Springer-Verlag, 460–477.

Schwartz, B., 2010. *How does Google's predictive traffic maps work?* [online]. Available from: http://www.seroundtable.com/archives/023155.html [accessed 7 October 2011].

Taylor, B., 2005. The world is your JavaScript-enabled oyster. *The official Google blog* [online]. Available from: http://googleblog.blogspot.com/2005/06/world-is-your-javascript-enabled_29.html [accessed 7 October 2011].

Wang, F., 2003. Job proximity and accessibility for workers of various wage groups. *Urban Geography*, 24, 253–271.

Wang, F., 2006. *Quantitative methods and applications in GIS*. Boca Raton, FL: CRC Press.

Appendix. Program *traveltime.py* for calibrating O–D travel time matrix by Google Maps API

```
import arcpy
import urllib
import time
from xml.etree.ElementTree import XML,
fromstring, tostring
fromFile = arcpy.GetParameter(0)
toFile = arcpy.GetParameter(1)
Resultfile = arcpy.GetParameterAsText(2)
Result = open(Resultfile,”w”)
Result.write(“FromFID,toFID,TravelTime”)
Result.write(“\n”)
fromCursor = arcpy.SearchCursor(fromFile)
toCursor = arcpy.SearchCursor(toFile)
fromRow = fromCursor.reset()
fromRow = fromCursor.next()
while (fromRow!=None):
fromX = fromRow.shape.centroid.X
fromY = fromRow.shape.centroid.Y
fromFID = fromRow.FID
arcpy.AddMessage(str(fromFID))
toCursor = arcpy.SearchCursor(toFile)
toRow = toCursor.reset()
toRow = toCursor.next()
while (toRow!=None):
toX = toRow.shape.centroid.X
toY = toRow.shape.centroid.Y
toFID = toRow.FID
arcpy.AddMessage(str(toFID))
googletext = “http://maps.googleapis.com/
maps/api/directions/
xml?origin=(“ + str(fromY) + “,” +
str(fromX) + “) &destination=(“ + str(toY)
+ “,” + str(toX) + “) &sensor=false”
time.sleep(3)
xmlfile = urllib.urlopen(googletext)
xml = xmlfile.read()
value = “NA”
dom = fromstring(xml)
nodelist = dom.getchildren()
if (nodelist[0].text == “OK”):
arcpy.AddMessage(nodelist[0].text)
route=nodelist[1]
leg=route.getchildren()[1]
duration = leg.find(“duration”)
value = duration.getchildren()[0].text
else:
arcpy.AddError(nodelist[0].text)
Result.write(str(fromFID))
Result.write(“,”)
Result.write(str(toFID))
Result.write(“,”)
Result.write(value)
Result.write(“|n”)
toRow = toCursor.next()
fromRow = fromCursor.next()
Result.close()
del fromCursor
del toCursor
```