A geographic information system for dynamic ridematching

Sasha Dos-Santos
University of South Florida

Follow this and additional works at: https://scholarcommons.usf.edu/etd

Part of the American Studies Commons

Scholar Commons Citation
Dos-Santos, Sasha, "A geographic information system for dynamic ridematching" (2005). Graduate Theses and Dissertations.
https://scholarcommons.usf.edu/etd/2859

This Thesis is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.
A Geographic Information System for Dynamic Ridematching

by

Sasha dos Santos

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Rafael Perez, Ph.D.
Miguel Labrador, Ph.D.
Dewey Rundus, Ph.D.

Date of Approval
March 31, 2005

Keywords: GIS, Geodatabase, Spatial Analysis, Carpool, Ridesharing

© Copyright 2005, Sasha dos Santos
# Table of Contents

List of Tables --------------------------------------------------------------------------------------------------------------- iii

List of Figures ------------------------------------------------------------------------------------------------------------- iv

Abstract --------------------------------------------------------------------------------------------------------------- vi

Chapter 1   Introduction ----------------------------------------------------------------------------------------------------- 1
  1.1   Background -------------------------------------------------------------------------------------------------- 1
  1.2   Geographic Information Systems ---------------------------------------- 1
  1.3   Dynamic Ridematching ------------------------------------------------ 6
  1.4   Motivation ------------------------------------------------------------- 7
  1.5   Objectives -------------------------------------------------------------- 9

Chapter 2   System Architecture ------------------------------------------------------------------------------------------- 10
  2.1   Introduction ---------------------------------------------------------- 10
  2.2   Geodatabase ----------------------------------------------------------- 11
  2.3   GIS Server ------------------------------------------------------------- 11
  2.4   Web Server ------------------------------------------------------------- 12
  2.5   User Interface---------------------------------------------------------- 12
  2.6   System Administration ------------------------------------------------ 13

Chapter 3   The Geodatabase ------------------------------------------------------------------------------------------------- 14
  3.1   Introduction----------------------------------------------------------- 14
  3.2   Locations Table ------------------------------------------------------ 15
  3.3   Trips Table ----------------------------------------------------------- 16
  3.4   Calendar Table -------------------------------------------------------- 17
  3.5   Trip Schedules Table-------------------------------------------------- 17
  3.6   Matches Table --------------------------------------------------------- 18
  3.7   Roads Feature Class---------------------------------------------------- 18
  3.8   Address Locator and Place Name Alias Table--------------------------- 20
  3.9   Trips Feature Class---------------------------------------------------- 21
  3.10  Geometric Network------------------------------------------------------- 21
  3.11  Shortest Path Feature Class -------------------------------------------- 22
List of Tables

Table 1      Attributes of the Locations Table ......................................................... 15
Table 2      Attributes of the Trips Table ................................................................. 16
Table 3      Attributes of the Calendar Table ............................................................ 17
Table 4      Attributes of the Trip Schedules Table ...................................................... 17
Table 5      Attributes of the Matches Table ............................................................... 18
Table 6      Attributes of the Roads Feature Class ....................................................... 19
Table 7      Attributes of the Trips Feature Class ......................................................... 21
Table 8      Attributes of the Shortest Path Feature Class ........................................... 22
Table 9      Possible Combinations of Round-trip/One-way Preference ......................... 24
Table 10     Possible Combinations of Driver/Passenger Preference ............................. 24
Table 11     Address Styles Permitted by the Address Locator ....................................... 34
Table 12     Set Operations ......................................................................................... 40
Table 13     Sorting the Matches ................................................................................... 41
Table 14     Overlapping Time Windows Test ................................................................. 52
Table 15     Recurrence Pattern Test ............................................................................. 53
List of Figures

Figure 1  Representing Objects  .................................................................  3
Figure 2  A Layer of Road Objects  ..............................................................  4
Figure 3  A Layer of House Objects  ............................................................  4
Figure 4  A Layer of Airport Objects  ...........................................................  5
Figure 5  Creating a Map from Layers  .........................................................  5
Figure 6  A Typical User Interface for Entering a Trip Schedule  .................  8
Figure 7  System Architecture  .................................................................... 10
Figure 8  The User Interface  ....................................................................... 13
Figure 9  Line Direction  ............................................................................. 19
Figure 10 Every Weekday ............................................................................ 25
Figure 11 Every X Days ................................................................................ 26
Figure 12 X Days On, Y Days Off ................................................................. 26
Figure 13 Every X Weeks on a Day of the Week ......................................... 27
Figure 14 1st, 2nd, 3rd, 4th, or Last Day of the Week of Every X Months .......... 28
Figure 15 Day Y of Every X Month(s) ......................................................... 29
Figure 16 A Recurring Trip Without a Pattern ............................................ 30
Figure 17 Overlapping Trip Dates ............................................................... 32
Figure 18 Overlapping Trip Times ............................................................... 33
Figure 19 Geocoding an Address ............................................................... 34
Figure 20  Calculating the Shortest Path .................................................. 35
Figure 21  Splitting the Path into Segments .............................................. 36
Figure 22  Creating Buffers ..................................................................... 37
Figure 23  Applying Buffers ..................................................................... 39
Figure 24  The Shortest Path of the Driver ............................................... 42
Figure 25  The Shortest Path of the Passenger .......................................... 42
Figure 26  Deviating from the Path ............................................................ 42
Figure 27  The Path from the Driver’s Source to the Passenger’s Source .... 43
Figure 28  The Path from the Passenger’s Source to the Passenger’s Destination........ 43
Figure 29  The Path from the Passenger’s Destination to the Driver’s Destination .... 43
Figure 30  Two Houses Lie on Opposite Sides of a River ......................... 45
Figure 31  One Must Travel Over a Bridge to Get from One House to the Other ...... 45
Figure 32  The Path Around a River .......................................................... 45
Figure 33  Two Houses Lie on Opposite Sides of an Interstate .................. 46
Figure 34  One Must Travel to an Underpass to Get from One House to the Other ...... 46
Figure 35  The Path Around an Interstate .................................................. 46
Figure 36  A User’s Shortest Path Includes Travel on an Interstate ............ 47
Figure 37  A Match Lies Next to the Interstate ............................................ 47
Figure 38  The User Must Get Back Onto the Interstate ............................ 47
Figure 39  Favoring the Direction of Travel .............................................. 48
Figure 40  Different Source, Same Destination Test ................................... 51
Figure 41  Similar Source, Different Destination Test ............................... 52
A Geographic Information System for Dynamic Ridematching

Sasha dos Santos

ABSTRACT

The Online Transportation Option System (OTOS) is a Geographic Information System (GIS) that addresses many of the limitations associated with traditional dynamic ridematching applications. The main improvements are in the areas of trip scheduling and match searching. OTOS is unique in its ability to accept trips with schedules that cannot be expressed in terms of a regular weekly trip. OTOS also distinguishes itself in its use of spatial analysis techniques to locate matches. Specifically, the use of a shortest path solver enables the ridematching algorithm to perform a search along the path of a user’s trip, in addition to the customary radial search around the endpoints. The shortest path solver is also used to calculate the driving distance between the user and a match. This provides a more accurate measurement than the straight-line distance used by other algorithms, especially in the presence of barriers.
1.1 Background

The Online Travel Option System (OTOS) is a project sponsored by the Center for Urban Transportation Research (CUTR) at the University of South Florida. The objective of the project is to help serve the transportation needs of the university community by promoting the concept of ridesharing. Ridesharing refers to two or more people traveling together in the same vehicle. The most common forms of ridesharing are carpooling and vanpooling. Carpools are arrangements of 2 - 4 people who travel in a personal vehicle. Vanpools are arrangements of up to 15 people who travel in a van. Ridesharing benefits the community by reducing traffic congestion and increasing the number of available parking spaces. OTOS acts as a ridematching system, matching individuals whose schedules and itineraries make them good candidates to form a carpool.

1.2 Geographic Information Systems

The United States Geological Survey (USGS) defines a Geographic Information System (GIS) as "a computer system capable of assembling, storing, manipulating, and displaying geographically referenced information, i.e. data identified according to their
locations. Practitioners also regard the total GIS as including operating personnel and the data that go into the system.”

In a GIS, an object can be represented by a point, line, or polygon. For example, a house can be represented by a point, a road by a line, and an airport by a polygon (see Figure 1). An object may be associated with several descriptive attributes. For example, a house can be associated with its address, a road by its speed limit, and an airport with its name.

The data in a GIS can be visualized on a map. A map consists of one or more layers. A layer is a collection of similar objects. Each layer in a map represents a different view of the same geographic area. In this respect, a layer is similar to a legend item on a paper map [1]. In Figures 2-5, a road layer, house layer, and airport layer are combined to form a map of Hillsborough County, FL.

A GIS includes tools that can be used to perform spatial analysis, the study of the relationships between objects on a map. Using the map in Figure 5 as an example, the following tasks can be easily performed:

- Find the quickest path to get from House # 1 to House # 2
- List the names of all roads that enter Tampa International Airport
- Count the number of houses that are within 1 mile from Platt Street
Figure 1 Representing Objects. Objects can be represented as points, lines, or polygons.
Figure 2  A Layer of Road Objects

Figure 3  A Layer of House Objects
Figure 4 A Layer of Airport Objects

Figure 5 Creating a Map from Layers. The result of overlaying the house, road, and airport layers.
1.3 Dynamic Ridematching

The Washington State Department of Transportation defines dynamic ridematching as an “instant ridematching service, usually provided via the internet.” A survey of dynamic ridematching applications will reveal that most follow the same basic algorithm:

- Ask the user for the day(s) of the week in which the trip will take place.
- Ask the user for the desired commute times. This is often done in terms of time to arrive at work and time to depart from work.
- (Optional) Ask the user if he would prefer to restrict the search to people who match a set of criteria. Common choices include: driver/passenger, male/female, smoking/non-smoking
- Ask the user for the address of his source and destination. The user either types the address or selects one from a list.
- Convert the addresses into x and y coordinates (source and destination points). The coordinates can represent the actual address or the centroid (center point) of the postal code or city.
- Draw a circle around the source and destination points. The radius of the circle is either hard-coded or set by the user. The radius is either a fixed (ex. 3 miles) or relative to the distance between the source and destination points (ex. 5 % of the total distance).
- Search for trips in the database whose source and destination points also reside within the circles.
• (Optional) Filter the results to include only those people who match the user’s trip schedule (day and time).

• Display the list of matches to the user. The match list often includes: 1) a description of the match’s trip schedule, 2) the distance between the source points of the two trips, and 3) the distance between the destination points of the two trips. If the results were not filtered in the previous step, the user must manually scan the list to see if anyone matches the user’s trip schedule.

1.4 Motivation

In traditional ridematching applications, the user is presented with a very limited way of expressing his trip schedule. Figure 6 illustrates the typical user interface for entering a trip schedule. A trip entered in this fashion is assumed to take place on a weekly basis for as long as the trip remains in the system (most applications automatically purge trips after some period of time). Two people do not need to have identical trip schedules in order to be matched together. For example, all other things being equal, if one person travels Monday – Friday and another travels Monday – Wednesday, these people can be matched together. The problem with this approach is that it may be too restrictive to describe some user’s schedules. Some examples include:

• a trip that occurs on the first Monday of every month

• a trip that occurs every 2 days (regardless of the day of the week)

• a trip that only occurs once, on March 25, 2005
In traditional ridematching applications, distances (ex. the distance between the source points of the user and match), are calculated by measuring the length of the straight line which connects two points. The problem with this approach is that it assumes that the straight-line distance is a good approximation for the actual driving distance between two points. Unless two points lie on the same street, the driving distance will always be greater than the straight-line distance. The difference between these distances can be quite large if two points lie on opposite sides of a barrier, such as a river or interstate. By definition, the presence of a barrier forces drivers to travel around it, possibly for miles, before reaching the other side.

In traditional ridematching applications, the focus is placed solely on the source and destination points of the user’s trip. However, it is also possible to find matches that can be made along the path of the user’s trip.
1.5 Objectives

The system must be flexible enough to handle a wide variety of trips, including:

Frequent vs. Infrequent

- a full-time student with classes every week
- a part-time student with classes on the first Saturday of every month

Recurring vs. One-time

- a student who goes to school every Wednesday
- a student who needs a ride home for the holidays

Local vs. Long-Distance

- a student who travels from Tampa to St. Petersburg (a distance of 15 miles)
- a student who travels from Tampa to Orlando (a distance of 80 miles)

One-way vs. Round-trip

- a student who would like to carpool with the same person to and from school
- a student who takes the shuttle to campus, but needs to carpool home

The system should utilize GIS concepts to:

- calculate distances based on the actual path a driver would take, instead of relying on a straight-line distance
- allow for matches to be found along a user’s path
- provide researchers with the ability to visualize and analyze the data collected from users
Chapter 2  System Architecture

2.1 Introduction

The system architecture is illustrated in Figure 7. The architecture is based on a client/server model where a user communicates with the GIS via the Internet. A series of ASP.NET Web pages are used to collect information regarding the trip he would like to find matches for. A ridematching algorithm, implemented in Visual Basic .NET, searches for trips in the database which meet the user’s requirements. The results of the search are then instantly presented to the user.

Figure 7  System Architecture
2.2 Geodatabase

The geodatabase stores the inputs and outputs of the ridematching algorithm. The geodatabase is a model for storing and managing geographic data using a relational database. It is based on a multitier architecture where an underlying database management system (DBMS) is responsible for the storage and retrieval of data, and an application tier, ArcSDE, is responsible for enforcing behavior and integrity rules. ArcSDE works with the DBMS to manage the storage, indexing, access, and editing of data. ArcSDE provides an interface by which applications can access the underlying database. This allows a developer to work with the high-level GIS data structures without concern for how they are implemented in a particular database platform [1]. OTOS uses Microsoft SQL Server 2000. The contents of the geodatabase are discussed in Chapter 3.

2.3 GIS Server

ArcGIS Server is a platform for building GIS applications that are accessible via the Internet. It includes a Web Application Developer Framework (ADF) for building Web pages that make use of ArcObjects. ArcObjects is a library of software components (COM classes) that allow developers to incorporate GIS functionality into their Web pages or desktop applications [2]. The OTOS Web site consists of ASP.NET pages written in Visual Basic.NET.
2.4 Web Server

The Web server hosts the Web pages that act as the user interface for the system. The ArcGIS Server ADF includes a Web application runtime that is installed on the Web server. The runtime allows the Web server and GIS Server to communicate with each other. When a user requests a Web page that makes use of ArcObjects, the Web server creates and uses objects that run in the GIS Server [2]. OTOS uses the Internet Information Services (IIS) Web server.

2.5 User Interface

The user interacts with the GIS through a series of Web pages (see Figure 8). The pages perform these basic functions:

- Collect information from the user and enter it into the geodatabase
- Use ArcObjects to perform the ridematching algorithm
- Display the results of the ridematching algorithm
- Allow a user to e-mail a match for the purpose of setting up a carpool
2.6 System Administration

The system administrator is responsible for maintaining the system. The database and Web server are administered using their built-in tools. For SQL Server 2000 and IIS, these tools are SQL Server Enterprise Manager and IIS Internet Services Manager, respectively. ArcGIS Desktop is a suite of three applications, ArcCatalog, ArcMap, and ArcToolbox, which are used to create, manage, view, and edit geographic data [1].

Figure 8  The User Interface
Chapter 3  The Geodatabase

3.1 Introduction

The geodatabase is a repository for all of the data collected by the user and output by the ridematching algorithm. The geodatabase is populated with tables that either explicitly or implicitly refer to geographic objects. These tables are related to each other through the use of common attributes. Only the most relevant tables are discussed here.

Tables with an explicit geographic reference use GIS data structures to represent objects. A feature represents an object as a row in a table. Each feature has an associated shape attribute that describes a point, polyline, or polygon. Common attributes associated with features include length, area, elevation, and name. A feature class is a collection of features and is stored as one table in the geodatabase. All of the features in a feature class have the same shape and represent the same type of object. The feature class is the implementation of the layer concept described in Section 1.2. A feature dataset is a collection of conceptually related feature classes. All of the feature classes used by the ridematching algorithm are stored in one feature dataset [1].

Tables with an implicit geographic reference describe, rather than represent, geographic objects [1]. Several of the tables in the geodatabase store descriptions of the users and the trips they would like to find matches for. The majority of this data is collected through the user interface. These tables implicitly refer to feature classes in the
geodatabase. For example, the geodatabase contains a table, Locations, which stores addresses. An address can be represented as a point in a feature class named Trips. Therefore, a record in the Locations table describes a feature in the Trips feature class.

Relationships among tables are created using keys. A primary key uniquely identifies a record in a table. A foreign key is an attribute in one table that refers to the primary key of another table. For example, the Trips feature class contains an attribute, Trip ID, which is the primary key for the Shortest Path feature class. The use of the Trip ID attribute creates a relationship between the Trips and Shortest Path feature classes. Most of the tables and feature classes are related through the use of a Trip ID attribute.

3.2 Locations Table

The Locations table stores the addresses of source and destination points (see Table 1). Each location is associated with a particular user and can be associated with several of the user’s trips. The table stores a location as a street address and as its x and y coordinates.

Table 1  Attributes of the Locations Table

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location ID</td>
<td>Primary key</td>
</tr>
<tr>
<td>User ID</td>
<td>The user associated with this location.</td>
</tr>
<tr>
<td>Location Description</td>
<td>A name given to this location by the user.</td>
</tr>
<tr>
<td>Address</td>
<td>The street address of this location.</td>
</tr>
<tr>
<td>City</td>
<td>The city where this location resides.</td>
</tr>
<tr>
<td>State</td>
<td>The state where this location resides.</td>
</tr>
<tr>
<td>Zip Code</td>
<td>The zip code or postal code of this location.</td>
</tr>
<tr>
<td>X-Coordinate</td>
<td>The x-coordinate.</td>
</tr>
<tr>
<td>Y-Coordinate</td>
<td>The y-coordinate.</td>
</tr>
</tbody>
</table>
3.3 Trips Table

The Trips table describes the trips that are taken by the users (see Table 2). Each trip is associated with two locations, source and destination. Once the ridematching algorithm determines the shortest path from source to destination, it stores the length of this path in the table. Each trip has a time window, a time interval in which the user would like to arrive at his destination.

Table 2 Attributes of the Trips Table

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trip ID</td>
<td>Primary key</td>
</tr>
<tr>
<td>User ID</td>
<td>The user associated with this trip.</td>
</tr>
<tr>
<td>Trip Description</td>
<td>A name given to this trip by the user.</td>
</tr>
<tr>
<td>Carpool Type</td>
<td>Indicates if the user prefers to be a driver or passenger.</td>
</tr>
<tr>
<td>Source Location</td>
<td>The Location ID for the source of the trip.</td>
</tr>
<tr>
<td>Destination Location</td>
<td>The Location ID for the destination of the trip.</td>
</tr>
<tr>
<td>Roundtrip</td>
<td>Indicates if the user would like to travel from the source to the destination and then back to the source.</td>
</tr>
<tr>
<td>Recurring</td>
<td>Indicates whether this trip occurs once or on a regular basis.</td>
</tr>
<tr>
<td>Active</td>
<td>Indicates if the user is still looking for matches for this trip.</td>
</tr>
<tr>
<td>Earliest Arrival Time</td>
<td>The earliest time that the user can arrive at the destination.</td>
</tr>
<tr>
<td>Latest Arrival Time</td>
<td>The latest time that the user can arrive at the destination.</td>
</tr>
<tr>
<td>Earliest Departure Time</td>
<td>The earliest time that the user can depart from the destination.</td>
</tr>
<tr>
<td>Latest Departure Time</td>
<td>The latest time that the user can depart from the destination.</td>
</tr>
<tr>
<td>New Trip</td>
<td>Indicates whether the ridematching algorithm has been run on this trip.</td>
</tr>
<tr>
<td>Length</td>
<td>The length of the shortest path from source to destination, as determined by the ridematching algorithm.</td>
</tr>
</tbody>
</table>
3.4 Calendar Table

The Calendar table is used, much like its namesake, to list and describe the days of the year (see Table 3).

Table 3 Attributes of the Calendar Table

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>The calendar date associated with this record.</td>
</tr>
<tr>
<td>Weekday</td>
<td>The value of this attribute is 1 if this date is a weekday.</td>
</tr>
<tr>
<td>Year</td>
<td>The year part of the date.</td>
</tr>
<tr>
<td>Month</td>
<td>The month part of the date.</td>
</tr>
<tr>
<td>Day of the Week</td>
<td>The day of the week: M, T, W, Th, F, S, or Sun</td>
</tr>
<tr>
<td>Week of the Year</td>
<td>A number designating the week of the year that this date takes place.</td>
</tr>
</tbody>
</table>

3.5 Trip Schedules Table

The Trip Schedules table is used to store each occurrence of a trip (see Table 4).

For example, a trip that is scheduled for every Friday in March will be associated with 4 records in the table. One of these records will correspond to the trip that occurs on Friday March 25.

Table 4 Attributes of the Trip Schedules Table

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trip ID</td>
<td>The trip associated with this record.</td>
</tr>
<tr>
<td>Earliest Arrival</td>
<td>The earliest date and time that the user can arrive at the destination.</td>
</tr>
<tr>
<td>Latest Arrival</td>
<td>The latest date and time that the user can arrive at the destination.</td>
</tr>
<tr>
<td>Earliest Departure</td>
<td>The earliest date and time that the user can depart from the destination. This attribute is empty for one-way trips.</td>
</tr>
<tr>
<td>Latest Departure</td>
<td>The latest date and time that the user can depart from the destination. This attribute is empty for one-way trips.</td>
</tr>
</tbody>
</table>
3.6 Matches Table

The Matches table (see Table 5) stores the results of the ridematching algorithm. Each record in this table represents two users whose travel patterns make them ideal candidates to form a carpool.

Table 5  Attributes of the Matches Table

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Match ID</td>
<td>Primary key</td>
</tr>
<tr>
<td>Trip ID 1</td>
<td>A trip that is similar in schedule and path to the trip represented by Trip ID 2.</td>
</tr>
<tr>
<td>Trip ID 2</td>
<td>A trip that is similar in schedule and path to the trip represented by Trip ID 1.</td>
</tr>
<tr>
<td>Source Difference</td>
<td>The distance between the source points of the two trips.</td>
</tr>
<tr>
<td>Destination Difference</td>
<td>The distance between the destination points of the two trips.</td>
</tr>
<tr>
<td>Path Length</td>
<td>The length of the shortest path that incorporates the source and destination points of the two trips.</td>
</tr>
</tbody>
</table>

3.7 Roads Feature Class

The ridematching algorithm requires a feature class that represents roads (interstates, highways, city streets) as lines. This polyline feature class is the foundation for all the spatial analysis done by the algorithm.

Each feature has a shape attribute that stores the coordinates of the line. One endpoint of the line is regarded as the starting point and the other as the endpoint. This is to give the lines a sense of direction. The right and left portions of the road are relative to this sense of direction (see Figure 9). The direction is not related to the flow of traffic on the road [3].
Figure 9  Line Direction

Table 6 lists the attributes that the feature class is required to have. This is not a literal representation - the actual storage is dependent on the company or agency that compiles the data. OTOS uses the Dynamap/Block Boundary v4.2 County Tile compiled by Geographic Data Technology [3].

Table 6  Attributes of the Roads Feature Class

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape</td>
<td>The coordinates of the line.</td>
</tr>
<tr>
<td>Shape Length</td>
<td>The length of the line.</td>
</tr>
<tr>
<td>Length</td>
<td>The length of the line in units consistent with that of the Speed Limit attribute.</td>
</tr>
<tr>
<td>Speed Limit</td>
<td>The maximum legal speed for traveling on the road.</td>
</tr>
<tr>
<td>Cost of Travel</td>
<td>Calculated by dividing the Length by the Speed Limit. This is set to -1 if travel in a direction is prohibited.</td>
</tr>
<tr>
<td>Name</td>
<td>One or more names associated with the road.</td>
</tr>
<tr>
<td>House Addresses</td>
<td>The range of house addresses on each side of the road.</td>
</tr>
<tr>
<td>Zip Codes</td>
<td>The zip code on each side of the road.</td>
</tr>
</tbody>
</table>

Most commercially available data, including Dynamap, uses a geographic coordinate system. This system represents the earth as a spheroid. A point is represented by a longitude and latitude, which are the angles the point makes with the center of the earth. Degrees of longitude and latitude do not have a standard length across the earth’s surface and, therefore, should not be used to calculate distances, such as the length of a
path. To overcome this difficulty, the data must be projected onto a two-dimensional surface using tools available in ArcGIS Desktop. A map projection uses a formula to transform spherical coordinates to planar coordinates. There are several projections available and each will cause distortions in shape, area, distance, or direction. It is important that a projection be chosen that minimizes the distortion in the particular geographic area of study. The projection chosen for the Roads feature class is Universal Transverse Mercator North American Datum 1983 Zone 17. This projection minimizes the distortion for the state of Florida (the panhandle region is slightly more distorted than the rest of the state) [4]. Because all of the feature classes in the geodatabase participate in the same feature dataset, they must all use the same projection.

3.8 Address Locator and Place Name Alias Table

The term geocoding refers to associating an address with x and y coordinates. It is accomplished using ESRI’s built-in geocoding engine. The engine requires an address locator, a table in the geodatabase that specifies the types of addresses that can be matched and the reference data they are to be matched against. This table is created using tools available in ArcGIS Desktop [5].

Landmarks are geocoded with the assistance of a place name alias table. The table consists of two attributes, Name and Address, which associates the name of a landmark with an address. This address can then be geocoded in the usual manner. The alias table contains airports, hospitals, malls, stadiums, theatres, universities, and local
attractions. Because the alias table is manually created, only landmarks that are commonly referenced by the user group are included.

3.9 Trips Feature Class

The Trips feature class (see Table 7) stores the geocoded source and destination points of all the trips in the Trips table. This feature class is related to most of the user profile tables by the Trip ID attribute.

Table 7 Attributes of the Trips Feature Class

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object ID</td>
<td>Primary key</td>
</tr>
<tr>
<td>Shape</td>
<td>The coordinates of this point.</td>
</tr>
<tr>
<td>Trip ID</td>
<td>The unique Trip ID associated with this trip. This is a foreign key linking this class with the Trips table.</td>
</tr>
<tr>
<td>Class</td>
<td>‘S’ if this point represents the source point. ‘D’ if this point represents the destination point.</td>
</tr>
</tbody>
</table>

3.10 Geometric Network

The Geometric Network is a data structure that stores the connectivity of features in the Roads feature class. In the terminology of networks, each edge represents a road. The intersection of two or more roads is a junction [6].

A weight is a number that represents the cost of traversing an edge. Each edge is associated with two weights corresponding to the two ways of traversing the edge. In the majority of cases, these weights are identical and equal to the length of the edge divided by its speed limit. Edges representing one-way roads are given a weight of -1 in the direction in which the road is impassable [7].
3.11 Shortest Path Feature Class

The Shortest Path feature class (see Table 8) stores the shortest path between two points (source and destination) in the Trips feature class. This path is calculated by the ridematching algorithm. The algorithm splits the path into three line segments, so one trip is associated with three features.

Table 8  Attributes of the Shortest Path Feature Class

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object ID</td>
<td>Primary key</td>
</tr>
<tr>
<td>Shape</td>
<td>The coordinates of this line segment.</td>
</tr>
<tr>
<td>Shape Length</td>
<td>The length of the line segment.</td>
</tr>
<tr>
<td>Trip ID</td>
<td>The unique Trip ID associated with this trip. This is a foreign key linking this class with the Trips table.</td>
</tr>
</tbody>
</table>
Chapter 4  Ridematching Algorithm

4.1 Introduction

The ridematching algorithm searches the geodatabase for individuals with similar schedules and itineraries. The algorithm is based on the basic algorithm described in Section 1.3, but with the following enhancements:

- Flexibility to accept any type of trip schedule
- Ability to search for matches along the path of a trip
- Ability to calculate driving distances

A user is asked to input a trip profile consisting of the source and destination points, trip schedule, and other preferences. The algorithm will search for matches by comparing the user’s profile with those of other trips. The term 'match' is used to refer both to a trip in the geodatabase and the person associated with the trip. Initially, the search space consists of all the trips in the geodatabase. The algorithm reduces the search space by applying a set of rules to reject potential matches. The rules are presented here in the order in which they are applied.

4.2 Using Round-trip/One-way Preference to Reduce the Search Space

In a typical scenario, the user travels from the source to the destination and then returns back to the source at some later date or time. A round-trip requires that a match be
available for both portions of the trip. In a one-way trip, the user is a) not returning to the source, b) not interested in carpooling for the return trip, or c) not concerned with carpooling with different people on each portion of the trip. The search space is reduced based on the rules in Table 9.

Table 9 Possible Combinations of Round-trip/One-way Preference

<table>
<thead>
<tr>
<th>User</th>
<th>Potential Match</th>
<th>Accepted?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round-trip</td>
<td>One-way</td>
<td>No</td>
</tr>
<tr>
<td>Round-trip</td>
<td>Round-trip</td>
<td>Yes</td>
</tr>
<tr>
<td>One-way</td>
<td>One-way</td>
<td>Yes</td>
</tr>
<tr>
<td>One-way</td>
<td>Round-trip</td>
<td>No</td>
</tr>
</tbody>
</table>

4.3 Using Driver/Passenger Preference to Reduce the Search Space

Users are given the option of limiting the search to only finding drivers or only finding passengers. If a preference is given, the search space is reduced based on the rules in Table 10.

Table 10 Possible Combinations of Driver/Passenger Preference

<table>
<thead>
<tr>
<th>User</th>
<th>Potential Match</th>
<th>Accepted?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver</td>
<td>Driver</td>
<td>No</td>
</tr>
<tr>
<td>Driver</td>
<td>Passenger</td>
<td>Yes</td>
</tr>
<tr>
<td>Passenger</td>
<td>Driver</td>
<td>Yes</td>
</tr>
<tr>
<td>Passenger</td>
<td>Passenger</td>
<td>No</td>
</tr>
</tbody>
</table>
4.4 Using Trip Schedules to Reduce the Search Space

A one-time trip is a trip that the user intends to make only once. The user may select a date (ex. April 4) or range of dates (ex. April 3 through April 5) when the trip can take place.

A recurring trip is a trip that occurs more than once. In the majority of cases, a recurring trip occurs at regular intervals that can be modeled by a pattern. The length of an interval can be expressed in terms of days, weeks, or months. A user can describe these intervals by selecting from a list of recurrence patterns. Figures 10 – 15 lists each pattern along with an illustrative example. A trip that does not follow a pattern, but occurs more than once, can be entered on a day by day basis (see Figure 16).

<table>
<thead>
<tr>
<th>Sunday</th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
<th>Saturday</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 27</td>
<td>28</td>
<td>March 1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
</tr>
<tr>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>April 1</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 10 Every Weekday. Every weekday starting with March 1st
Figure 11 Every X Days. Every 4 days starting with March 1st

Figure 12 X Days On, Y Days Off. 3 days on, 2 days off starting with March 1st
Figure 13  Every X Weeks on a Day of the Week. Every 2 weeks on a Monday or Wednesday
Figure 14  1st, 2nd, 3rd, 4th, or Last Day of the Week of Every X Months. 1st Wednesday of every 1 month
Figure 15  Day Y of Every X Month(s). The 1st day of every 1 month
<table>
<thead>
<tr>
<th>Sunday</th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
<th>Saturday</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 27</td>
<td>28</td>
<td>March 1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
</tr>
<tr>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>April 1</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 16  A Recurring Trip Without a Pattern. The user enters the dates one by one.

Each trip is associated with a time window, (Earliest Arrival Time, Latest Departure Time), in which the user would like to arrive at his destination. The size of this window can be expressed in terms of minutes or hours. The largest possible window is 23 hours, 59 minutes. The following are examples of possible time windows: (8:45 AM, 9:00 AM), (9:00 AM, 11:00 AM), and (10:00 PM, 2:00 AM). A round-trip is also associated with a second window, (Earliest Departure Time, Latest Departure Time), in which the user would like to depart for the return trip.

A trip instance is defined as a specific occurrence of a trip. One-time trips are associated with one instance, while recurring trips are associated with several. For example, if a user travels to work every weekday, each week is associated with 5
instances, one for Monday, Tuesday, etc. A trip instance is stored as one record in the Trip Schedules table.

To reduce the search space, the Trip Schedules table is queried to locate overlapping trip instances. The trips that will remain in the search space are those trips where at least one instance overlaps with an instance of the user’s trip. It is important to emphasize that two trips do not have to have similar recurrence pattern in order to match. Figure 17 illustrates a scenario where two trips have different recurrence patterns, but can still carpool on selected dates (instances). Trip 1, shown in red, takes place every Monday, Wednesday, and Friday. Trip 2, shown in blue, has a three days on, two days off pattern. Even though the recurrence patterns are different, the two trips have several dates in common when they could carpool (provided that both trips meet all the other criteria to match). Figure 18 illustrates a scenario where several trip instances take place on March 25, 2005. Since the date is the same for each trip instance, only the time is considered in this example. A match can be made between 2 & 3, 3 & 4, or 3 & 5.
Figure 17  Overlapping Trip Dates. One recurrence pattern is illustrated in red, the other in blue.
### 4.5 Geocoding the Source and Destination

The addresses of the user’s source and destination can take the form of a house address, a street intersection, or a landmark (see Table 11). An address locator (as described in Section 3.8) is used to geocode the addresses into their x and y coordinates (see Figure 19). Addresses are geocoded by referencing the attributes of the Roads feature class. The attributes used are House Addresses, Zip Codes, and Name. The
address locator is able to associate an address with a location relative to the start and end points of the line (ex. left side of the road, at a distance of 1 mile from the start point).

Table 11  Address Styles Permitted by the Address Locator

<table>
<thead>
<tr>
<th>Address Style</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>House Address</td>
<td>14535 Fletcher Ave. Tampa, FL 33613</td>
</tr>
<tr>
<td>Street Intersection</td>
<td>Fowler Ave. &amp; Skipper Road Tampa, FL 33613</td>
</tr>
<tr>
<td>Landmark</td>
<td>University of South Florida</td>
</tr>
</tbody>
</table>

(a) A house is associated with an address.

(b) The address locator associates an address with a point on the earth’s surface.

Figure 19  Geocoding an Address
4.6 Finding the Shortest Path

Dijkstra’s shortest path algorithm is used on the Geometric Network to calculate a path from source to destination (see Figure 20). This path represents the quickest way for the user to get from the source to the destination based on the weight of each edge. Recall from Section 3.10 that the weight is equal to the time it takes to traverse an edge in a specified direction. The shortest path solver is biased towards main roadways and highways because of their relatively high speed limit. A weight of -1 signifies that travel is not permitted in the specified direction. The shortest path solver regards a negative weight as a barrier and does not attempt to traverse the road in the specified direction.

![Initial conditions](image1) ![The shortest path is shown in red.](image2)

Figure 20 Calculating the Shortest Path

The shortest path is split into three segments (see Figure 21). The first segment represents the first 15% of the path. The second segment represents the middle 70% of the path. The third segment represents the final 15% of the path. The segments are stored in the Shortest Path feature class.
(a) Segment 1 represents the first 15% of the path.

(b) Segment 2 represents the middle 70% of the path.

(c) Segment 3 represents the last 15% of the path.

Figures 21  Splitting the Path into Segments. The segments are highlighted in blue.
4.7 Creating Buffers

The next step is determining those trips in the search space whose source and destination are ‘close’ to that of the user. Close is a relative term that, by default, is based on a percentage of the total length of the path. This allows the algorithm to scale to both local and long-distance trips.

A buffer is a zone of a specified distance around a feature (see Figure 22). (The distances discussed here are the default distances.) Buffer 1 is applied around the source point with a radius of 15% of the path length. For example, if the total length of the path is 10 miles, the first buffer will represent a radial search of 1.5 miles. Buffer 2 is applied around Segment 2, the middle 70% of the path, with a radius of 5% of the path length. For a path of 10 miles, this buffer will represent half a mile on either side of the path. Buffer 3 is applied around the destination point with a radius of 15% of the path length.

Figure 22  Creating Buffers
4.8 Using Buffers to Reduce the Search Space

The Trips feature class is queried to find the Trip IDs of potential matches. Each query consists of a spatial element and an attribute element. The spatial element corresponds to one of the buffers. The attribute element will be either class = 'S' for source or class = 'D' for destination. The current user’s source and destination point are excluded from the search. The result of each query is stored in a selection set. Set 1 contains those features that are in Buffer 1 and which represent a source. Set 2 contains those features that are in Buffer 2 and which represent a source. Set 3 contains those features that are in Buffer 3 and which represent either a source or destination.

Using a series of set operations, the search space is reduced and a match list is created. The first step is to deal with the overlap of Buffers 2 and 3. Any source that lies within Buffer 3 should not be considered as a potential match, even if it resides in the intersection of Buffers 2 and 3. A set difference, S2 – S3, is used to remove from Set 2 those elements which are also in Set 3. The union of Set 1 and Set 2, S1 ∪ S2, results in a new set whose elements are features which represent a source. Set 3 is queried to find features whose Trip ID is equal to one of the Trip IDs found in the union. These trips form the match list. This process is illustrated in Figure 23 and Table 12.
Figure 23  Applying Buffers

Shown here is the Roads feature class (in light blue), Shortest Path feature class (in red), Trips feature class (in green), and the buffers (in light gray). Each feature in Trips is represented as an ordered triple -- (Object ID, Trip ID, Class). The TripID of the user’s trip is 162. All the trips are assumed to match the user’s schedule.
Table 12  Set Operations

<table>
<thead>
<tr>
<th>Set</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set 1</td>
<td>(471, 305, S)</td>
</tr>
<tr>
<td>Set 2</td>
<td>(447, 301, S)</td>
</tr>
<tr>
<td></td>
<td>(449, 302, S)</td>
</tr>
<tr>
<td></td>
<td>(451, 303, S)</td>
</tr>
<tr>
<td></td>
<td>(454, 304, S)</td>
</tr>
<tr>
<td>Set 3</td>
<td>(449, 302, S)</td>
</tr>
<tr>
<td></td>
<td>(452, 303, D)</td>
</tr>
<tr>
<td></td>
<td>(453, 304, D)</td>
</tr>
<tr>
<td>S2 – S3</td>
<td>(447, 301, S)</td>
</tr>
<tr>
<td></td>
<td>(451, 303, S)</td>
</tr>
<tr>
<td></td>
<td>(454, 304, S)</td>
</tr>
<tr>
<td>S1 ∪ S2</td>
<td>(447, 301, S)</td>
</tr>
<tr>
<td></td>
<td>(451, 303, S)</td>
</tr>
<tr>
<td></td>
<td>(454, 304, S)</td>
</tr>
<tr>
<td></td>
<td>(471, 305, S)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trips in S3 whose TripID matches a TripID in S1 ∪ S2</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(452, 303, D)</td>
</tr>
<tr>
<td></td>
<td>(453, 304, D)</td>
</tr>
</tbody>
</table>

4.9 Sorting the Matches

The matches are stored in the Matches table and are presented to the user as a sorted list. By default, the matches are sorted in the order that they are presented in Table 13. The matches are sorted based on the first criterion and then subsequent criteria are used as tiebreakers. For example, if two matches are equal in the Number of Trip Instances, then they will be sorted by their Path Deviation, and if this is also equal, by the
Preferred Arrival Time, etc. At the user’s request, another criterion may serve as the primary criterion. In this case, the Number of Trip Instances will serve as the first tiebreaker, and so on.

Table 13 Sorting the Matches

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Trip Instances</td>
<td>The number of trips instances that the user and match have in common</td>
</tr>
<tr>
<td>Path Deviation</td>
<td>An estimate of the distance the driver will cover to travel to the passenger’s source and destination.</td>
</tr>
<tr>
<td>Preferred Arrival Time</td>
<td>The time difference between the preferred arrival times of the user and match.</td>
</tr>
<tr>
<td>Preferred Departure Time</td>
<td>The time difference between the preferred departure times of the user and match. This criterion is only used for round-trips.</td>
</tr>
<tr>
<td>Age of Record</td>
<td>The length of time since the match’s trip was entered into the geodatabase.</td>
</tr>
</tbody>
</table>

4.10 Calculating the Path Deviation

The path deviation is an estimate of the distance the driver will cover to travel to the passenger’s source and destination (see Figures 24 - 26). The term ‘driver’ is based on the user’s driver/passenger preference. The path deviation is the difference between the lengths of the driver’s shortest path and the new path that is required to pick up the passenger and drop him off. The new path is calculated in three steps (see Figures 27 - 29). Length 1 is the length of the shortest path from the driver’s source to the passenger’s source. Length 2 is the length of the shortest path between the passenger’s source and the passenger’s destination. Length 3 is the length of the shortest path from the passenger’s destination to the driver’s destination.
Figure 24  The Shortest Path of the Driver

Figure 25  The Shortest Path of the Passenger

Figure 26  Deviating from the Path
Figure 27  The Path from the Driver’s Source to the Passenger’s Source

Figure 28  The Path from the Passenger’s Source to the Passenger’s Destination

Figure 29  The Path from the Passenger’s Destination to the Driver’s Destination
Calculating Length 1 and Length 3 using shortest paths has several advantages over relying on the straight-line distance between features. A straight-line distance can be deceiving because it can not take into account barriers. A barrier can be either natural or man-made and impedes travel. Consider the following scenarios:

- Two houses lie on opposite sides of a river. To travel from one house to the other requires traveling to a bridge and crossing the river (see Figures 30 - 32).
- Two houses lie on opposite sides of an interstate. To travel from one house to the other requires traveling to an underpass (see Figures 33 - 35).
- A user’s path requires travel on an interstate. A match lies next to the interstate. The straight-line distance from the match to the interstate is negligible. However, the user must actually travel to the nearest exit, pick up the match, and travel back to the interstate before continuing his trip (see Figures 36 - 38).
Figure 30  Two Houses Lie on Opposite Sides of a River

Figure 31  One Must Travel Over a Bridge to Get from One House to the Other

Figure 32  The Path Around a River
Figure 33  Two Houses Lie on Opposite Sides of an Interstate

Figure 34  One must Travel to an Underpass to Get from One House to the Other

Figure 35  The Path Around an Interstate
Figure 36  A User’s Shortest Path Includes Travel on an Interstate

Figure 37  A Match Lies Next to the Interstate

Figure 38  The User Must Get Back Onto the Interstate
The inclusion of Length 2 in the path deviation is to favor those matches that lie in the direction of the driver’s travel. Consider a scenario where a driver and two passengers, Passenger A and Passenger B, have the same destination. The source points of the passengers lie at an equal distance on either side of the driver. This scenario is illustrated in Figure 39. Lengths 1 and 3 will be identical for both passengers. However, if Passenger B lies in the direction that the driver would normally travel and Passenger A lies in the opposite direction, these passengers will not have the same path deviation because Length 2 for Passenger B will be smaller than Length 2 for Passenger A. The path deviation for Passenger A is \( X + (2X+Y) + 0 \). The path deviation for Passenger B is \( X + Y + 0 \). All other things being equal, if the driver must choose one passenger, Passenger B is the clear choice because the path deviation is smaller.

![Figure 39 Favoring the Direction of Travel. Passengers A and B are equidistant from the driver, but B is a better match because he is in the direction the driver normally travels.](image-url)
4.11 Implementation

The algorithm can be logically broken into the following tasks:

- **Geocode the source and destination points**: Geocoding is accomplished using the ArcObjects Location COM class.

- **Translate the recurrence pattern into a list of calendar dates**: ASP.NET pages (written in Visual Basic.NET) are responsible for translating the input from the user into the search criteria for a SQL SELECT statement. The SQL SELECT statement is used to query the Calendar table for the actual dates which satisfy the recurrence pattern. Some of these queries make use of User Defined Functions (UDFs) that execute inside the SQL Server and return a value to the SELECT statement. UDFs are used to find the last day of a month and to find the Nth occurrence of a day of the week (ex. last Monday in March, 2005).

- **Find overlapping trip instances**: Overlapping trip instances are determined using a UDF that compares each trip instance of the user’s trip with all of the other instances in the Trip Schedule table.

- **Find the shortest path between two points**: Shortest paths are calculated using the ArcObjects Network Analysis COM class.

- **Create buffers and work with selection sets**: Buffers and selection sets are parts of the ArcObjects Geodatabase COM class.

- **Sort the match list**: The match list is sorted using the SQL ORDER BY clause. The order in which attributes are to be sorted is specified by the user and the appropriate version of the ORDER BY clause is used.
Chapter 5  Evaluation

5.1 Introduction

OTOS was tested under several different scenarios to assess its ability to perform the tasks described in the last chapter. Each scenario tested one aspect of the ridematching algorithm. For example, one scenario tested the ability of the algorithm to find matches with overlapping time windows. All other variables (source, destination, recurrence pattern, etc.) were kept constant within the set so that only the variable in question would determine the match list. For each scenario, an expected match list was created by manually reducing the search space. For all of the scenarios, the algorithm had a 100% success rate when compared to the expected match list. The following sections describe each scenario.

5.2 Different Source, Same Destination

The variable of interest is the source point. Figure 40 shows the source points scattered throughout the map and labeled with their Trip ID numbers. All of the trips share the same destination point. Though all of the trips will match based on Buffer 3, only those that are also present in Buffer 1 should be listed in the final match list.
5.3 Similar Source, Different Destination

The variable of interest is the destination point. Figure 41 shows that the source point of every trip resides in the same general area, but the destination points are scattered throughout the map. Though all of the trips will match based on Buffer 1, only those that are also present in Buffer 3 should be listed in the match list.
5.4 Overlapping Time Windows

The variable of interest is the time window. Table 14 shows a small subset of the test data. The only trips that can match with each other are those with overlapping time windows. In Table 14, only trips 1 and 4 can produce a match.

Table 14 Overlapping Time Windows Test

<table>
<thead>
<tr>
<th>#</th>
<th>Earliest Arrival</th>
<th>Latest Arrival</th>
<th>Earliest Departure</th>
<th>Latest Departure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7:30 a.m.</td>
<td>8:00 a.m.</td>
<td>4:45 p.m.</td>
<td>5:15 p.m.</td>
</tr>
<tr>
<td>2</td>
<td>4:45 a.m.</td>
<td>5:00 a.m.</td>
<td>7:45 p.m.</td>
<td>8:15 p.m.</td>
</tr>
<tr>
<td>3</td>
<td>7:45 p.m.</td>
<td>8:15 p.m.</td>
<td>4:45 a.m.</td>
<td>5:15 a.m.</td>
</tr>
<tr>
<td>4</td>
<td>8:00 a.m.</td>
<td>9:00 a.m.</td>
<td>5:00 p.m.</td>
<td>5:30 p.m.</td>
</tr>
</tbody>
</table>
5.5 Recurrence Pattern

The variable of interest is the recurrence pattern. Table 15 shows a small subset of the test data. The only trips that can match with each other are those that can take place within the same day and time. The start and end dates of the recurrence are set to March 1, 2005 and April 1, 2005, respectively. For this period of time, trips 1, 2, 3, and 4 can match. Trip 5 does not have a match.

Table 15 Recurrence Pattern Test

<table>
<thead>
<tr>
<th>#</th>
<th>Recurrence Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MWF every week</td>
</tr>
<tr>
<td>2</td>
<td>MTWRF every week</td>
</tr>
<tr>
<td>3</td>
<td>MW every 2 weeks</td>
</tr>
<tr>
<td>4</td>
<td>3 days on 3 days off</td>
</tr>
<tr>
<td>5</td>
<td>5th day of every 1 month</td>
</tr>
</tbody>
</table>
Chapter 6  Conclusions and Future Work

6.1 Conclusions

The Online Transportation Option System (OTOS) is a Geographic Information System (GIS) that addresses many of the limitations associated with traditional dynamic ridematching applications. The main improvements are in the areas of trip scheduling and match searching.

OTOS is unique in its ability to accept trips with schedules that can not be expressed in terms of a regular weekly trip. The ridematching algorithm can accept recurrence patterns expressed in terms of days, weeks, or months. The algorithm’s flexibility is due to the fact that a trip’s recurrence pattern is translated into a series of trip instances. While traditional algorithms are designed to compare recurrence patterns, the OTOS ridematching algorithm compares individual trip instances. This allows users with different recurrence patterns to be matched together.

OTOS distinguishes itself in its use of spatial analysis techniques to locate matches. The use of a shortest path solver enables the ridematching algorithm to calculate the shortest path from a user’s source to destination and then perform a search along that path. This is in addition to the customary radial search around the endpoints. In the worst case, the algorithm will find the same matches that a traditional algorithm would. In the best case, the algorithm is able to find additional matches by looking along the path.
OTOS introduces the idea of path deviation as a factor when ranking or sorting match lists. The path deviation is an accurate measurement of the distance between the user and the match. OTOS bases its distance measurements on the driving distance, calculated by the shortest path solver, instead of the straight-line approximation used by other algorithms. The advantage of such an approach is that it takes into account barriers and can be used to favor matches that lie in the direction of travel.

OTOS also provides transportation researchers with the ability to analyze and visualize the data collected by the user. Researchers can access the contents of the geodatabase using the ArcGIS desktop suite.

6.2 Future Work

While the focus so far has been on carpooling, the application can be extended to support vanpooling, as well. One possible strategy is to periodically use the ArcGIS software to visualize the spatial distribution of the trips in the geodatabase and have a person look for a cluster of individuals who are all traveling in the same direction at the same time. Another strategy is to build a vanpooling algorithm that will locate these clusters programatically.
References


Bibliography


