

Paper presented at the Third Annual World Congress on Intelligent Transport Systems,
October 14-18, 1996, Orlando, Florida

Seattle Smart Traveler

D. J. Dailey

Assistant Professor, Research
(206) 543-2493

D. Loseff

SST Project Manager
(206) 616-4078

D. Meyers

Research Associate
(206) 616-4073

fax (206) 616-1787
University of Washington
Dept. of Electrical Engineering, Box 352500
Seattle, WA 98195-2500

M. P. Haselkorn

Professor
(206) 543-2577
fax (206) 543-8858
University of Washington
Dept. of Technical Communication, Box 352195
Seattle, WA 98195-2195

ABSTRACT

Seattle Smart Traveler (SST) is a world wide web (WWW) application designed to test the concept of “dynamic” rideshare matching. SST collects spatial and temporal trip information using a series of WWW pages, performs a match using SQL specifications to a database engine, and supports both the standard phone-based contact methodology as well as two new, unique email-based contact methodologies.

INTRODUCTION

Seattle Smart Traveler (SST) is a Federal Highway Administration field operational test (FOT) designed to test the concept of “dynamic” rideshare matching. The SST application was developed at the University of Washington (UW) using a world wide web (WWW) interface. Members of the UW community are serving as the testbed for the FOT. The UW is an example of a large, closed environment where potential users have multiple, highly variable schedules and a financial incentive to arrive at the campus without a car. Additionally, the UW is an example of an environment with a high level of technological sophistication, where most potential users are computer literate and have access to multiple communications technologies (e.g. email, voice mail, and paging). This combination of variable schedules, computer literacy, and access to multiple communications media makes this population a reasonable representation of future work environments for employees of large agencies and an ideal testbed for “dynamic” rideshare matching.

We define dynamic rideshare matching as “the sharing of a single trip by two or more individuals, without regard to any previous history among the individuals involved.” Additionally, SST defines a “trip” as “a single instance of travel from one geographic location to another.” A “trip” is, therefore, by definition, a one-way trip, and the popular conception of a “round trip” is, for the purposes of SST, two “trips.” This is an important distinction between “dynamic” and “traditional” ridematching systems.

Dynamic rideshare matching differs from traditional rideshare matching in two important ways. The first major difference is the treatment of the traveler’s schedule. Traditional systems assume the traveler has a fixed schedule and a fixed set of origins and destinations. A dynamic system must consider each trip individually and be able to accommodate trips to arbitrary points at arbitrary times. The second major difference is that dynamic ridematch systems must provide the match information to the user quickly in order to accommodate near-term (same day) travel as well as long-term (future days or weeks) trips. Traditional systems frequently provide a matchlist through paper mail, a process which may take more than a day. For these two reasons, the requirements for dynamic rideshare matching are more demanding than those for the traditional rideshare application.

In designing SST, we postulated that the users of such a system view carpooling as a travel option in terms of three types of schedules: (1) the “traditional” fixed, commuting

schedule, (2) the regular but variable commuting schedule, and (3) the occasional trip or single trips to single destinations. The University environment inherently provides examples of all these schedules. The first situation occurs naturally for University staff with fixed hours of employment. The second situation arises because students and faculty often have varied class schedules. An example of the third situation occurs at the beginning and end of each term. Students travel either to home or to remote locations to spend the break periods or they return to campus after the break periods. This example is in addition to the truly dynamic situation where a University community member needs or can offer a ride on short notice. The existence of these three types of travel schedules makes the members of the UW community a good target audience for testing the viability of the dynamic ridesharing concept.

SST DESIGN

SST is designed to be a complete rideshare matching system, capable of providing matches for those with traditional needs as well as for those with variable schedules and occasional needs. This design recognizes that the traditional ridematching requirements are only a subset of the requirements for a dynamic rideshare system.

SST is designed to be accessible through the World Wide Web. This design decision was driven by two features of the WWW: (1) the popularity of the WWW along with the wide availability of free browser software and (2) the WWW is available 24 hours a day, 7 days a week. These two features ensure that SST can be delivered to a wide audience at the individual user's discretion and convenience. These features were also deemed necessary to entice a sufficient number of users to reach the critical mass where dynamic ridematching actually takes place.

In this paper we present the design of SST in two parts. We first review the matching process (the matching information that is collected) and the spatial and temporal matching scheme employed in SST, and we then review the technical design of the SST program.

Design: Collection of User Information

SST is set up as a series of Hyper Text Markup Language (HTML) "forms" to request required information from the user. The opening page of SST requests either (1) a new user's student or staff number or (2) an established user's userid and password. Both these actions force authentication of the user. SST is presently an experimental program which limits the user group to the University of Washington community, and the authentication process enforces this restriction. The SST application also requires contact information and a password. The requested contact information is a phone number and an email address. The email address is important because SST is able to automatically generate and send email messages to users with matching schedules. It is noteworthy that the user's home address is not requested as part of the SST registration. A home address is not used for matching purposes, and there is no paper copy of a matchlist to mail out, so a home address is not needed. The items just mentioned (email,

phone number, and a password) are the total extent of the personal information collected as part of the SST registration process.

To perform ridematching, mechanisms to collect and manage individual trip data are necessary, and the methodologies used by SST to accomplish this are described in the following section.

Design: Matching Temporal/Spatial Domains and Contact Methodology

To describe the design of the matching process in SST, we first outline the information needed and the methodology for collecting it from the user. Information collected by SST is always specific to the individual trips. From a user perspective, trips are divided into three categories in SST: (1) “regular commute” trips, (2) “additional recurring” trips (a trip made on a regular basis, in addition to a commute trip), and (3) an “occasional” trip. However, all three types of trips require essentially the same temporal and spatial information: (1) trip spatial origin, (2) trip spatial destination, (3) day of the week and trip-departure time range, and (4) trip-arrival time range. In addition, all trips are assigned an expiration date. This expiration date is assigned automatically for the first two types of trips, and the user specifically selects the expiration date for an “occasional” trip. The expiration date is used to remove trips whose temporal relevance has expired, an important feature sometimes overlooked in ridematch systems that do not age the trip information.

In the SST system, two important design decisions were made regarding the collection of the temporal and spatial information required to perform ridematching. The first decision involves the collection of temporal data. SST requests that the user enter a range for both the departure and arrival times. This makes the level of flexibility in a trip schedule explicit to the user and under the control of the user entering the travel information.

The second design decision involves the spatial information for origin and destination. The spatial information is requested with a series of pull down menus that implement an efficient search tree. At the top of the tree, the user can select from eight landmark types. The tree structure for selecting spatial information is shown in Figure 1. This tree structure is used to sequentially reduce the area of the search until a single landmark can be identified. The depth to which the user must descend in each tree depends on the type of spatial landmark chosen as the starting point. For example, to get to a specific road intersection in Seattle, the user must descend four levels (the greatest depth in the entire tree). The user starts at “Puget Sound Intersections” in Figure 1, proceeds to select the city of Seattle, decides that the street is either a freeway, a named street, or an intersection identified by numbered streets, and finally selects the specific intersection from a list. In contrast, to select many other spatial landmarks, only two

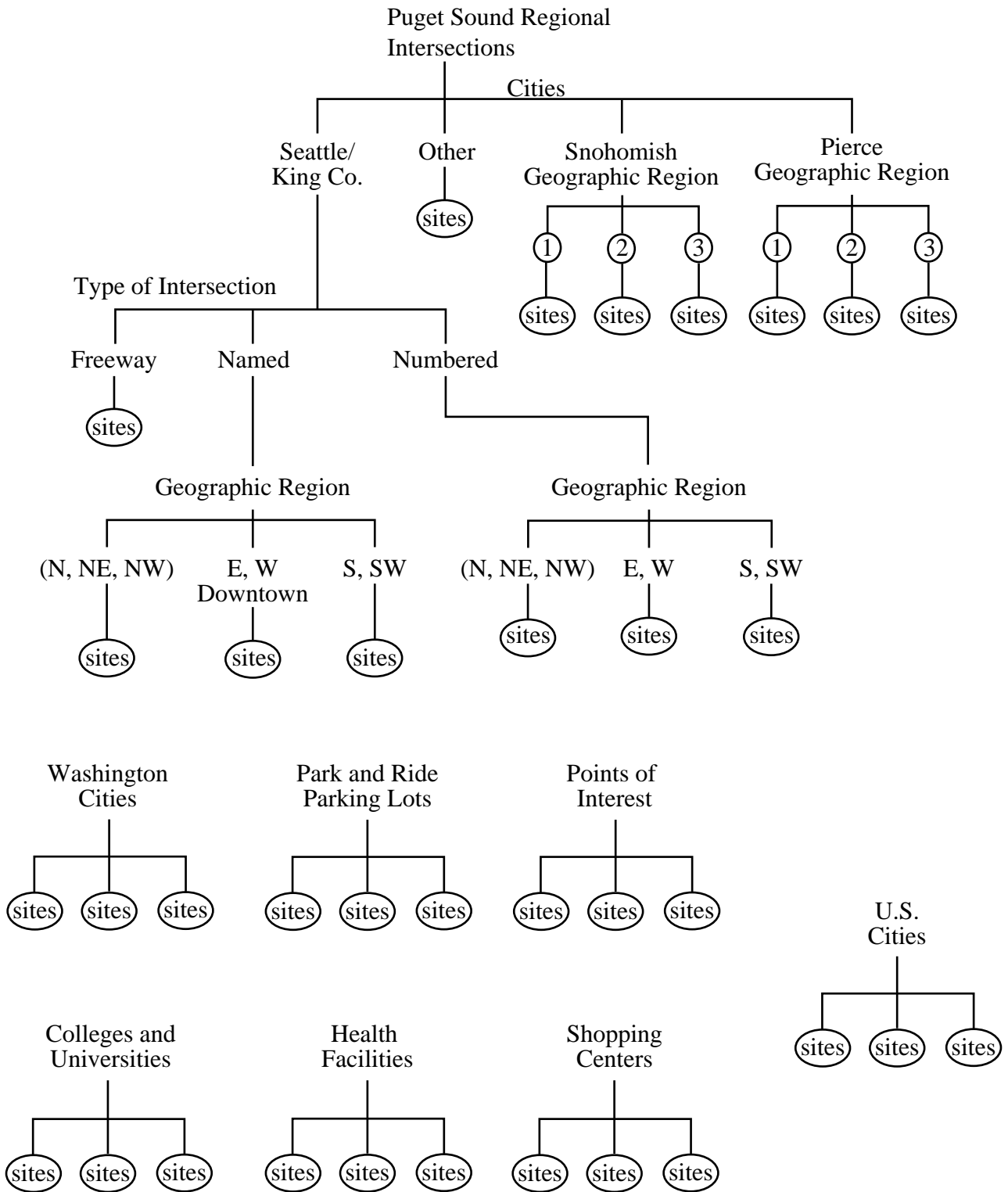


Figure 1. Geographic location search trees

choices need be made (see “Shopping Centers” in Figure 1). This reliance on landmarks to describe the trip endpoints removes the need for origin and destination addresses as well as the need for GIS software, but it does limit the total number of origins and destinations that are available to the user. (The implementation of SST uses latitude and longitude geo-coordinates of over 3,500 locations in the Puget Sound region, 200 cities in Washington State outside of the Puget Sound region, and 100 cities across the nation. The coordinates are taken from the TIGER spatial data database.)

Once temporal and geographic information is collected, the matching is done using a database engine and SQL commands. The database is queried for entries that match temporally and then geographically. The temporal match finds pairs of participants whose start and end point temporal ranges overlap (recall that the user explicitly enters ranges for the departure and arrival times). The default geographical matching range is 15% of the length of the trip in all directions from each endpoint to match against other user endpoints. This geographical coverage is user configurable and can be varied by broadening (25%) or narrowing (5%) the scope of the search. Once a set of matching trips has been produced, the names, phone numbers, and email addresses of users with matching trips are displayed on the screen. Riders who have a matching trip schedule must contact each other, and SST takes a unique approach to supporting this contact.

SST provides a matchlist with phone numbers and email addresses, a standard practice in ridesharing programs; however, in addition to a simple matchlist, an automated email option is integrated into SST. This option allows a user who has found matches to automatically email one or more of the other users with matching trips. In addition to basic email, SST provides a functionality that permits the user to email a message to a remote user’s Seiko message watch. The message contains either a phone number or a notification of the arrival of email. These two email contact methodologies, used to augment the more traditional match list, are unique to the SST program.

The SST program collects spatial and temporal trip information using a series of WWW pages as described, performs a match using SQL specifications to a database engine, and supports both the standard phone-based contact methodology as well as two new, unique email-based contact methodologies. The next section presents the actual mechanics of creating a ridesharing application that requires state information about the user in a stateless WWW environment.

THE SST APPLICATION AND THE WWW

In this section we describe the paradigms used inside the SST application. SST as a WWW application must address many of the problems that all program developers face in this environment. The main WWW problem encountered in developing SST is that browser interactions with a WWW server are inherently stateless. SST needs to maintain the state of an ongoing transaction (e.g. the user information about trips and contacts), and maintenance of that state information is difficult. In addition to the difficulty with state, most web browser

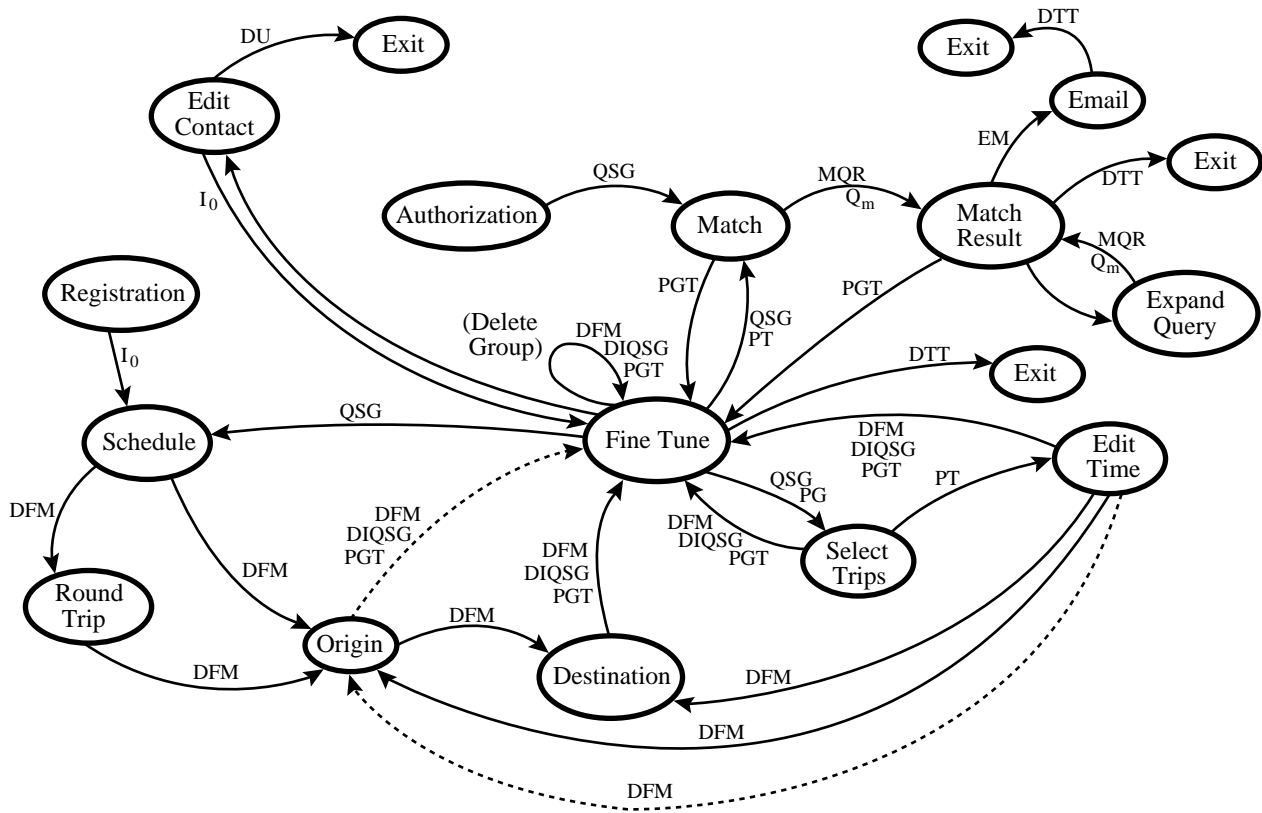
applications have a facility that enables the user to return to a previously displayed page without notifying the server of that page in any way (the “back button”). The existence of a back button makes it virtually impossible to guarantee that the state information is accurate. SST overcomes these difficulties using a database paradigm.

The SST application uses a database to store the information about users and the trips they are registering. (We use the mSQL database written by David Hughes available as shareware at: <ftp://bond.edu.au/pub/Minerva/mssql>). This cumulative information (trips and contact details) makes up the “state” information about a user. Figure 2 is a complete state diagram for the SST program. The states are represented by ovals and the state transitions by arrows. Users always transit between states in the direction of the arrows. Each of the transitions is marked with the abbreviation for one or more of the 14 specific routines that must be invoked to accomplish individual state transition. If more than one routine is needed, several program abbreviations appear. SST was designed modularly so that a fixed set of routines is applied in appropriate order to implement the overall state transition process.

We use the permanent and temporary database relations to maintain state information during “transactions.” A “transaction” in the SST program is the action of moving from one state to another. To preserve the state information, the SST application uses two sets of database tables: (1) the permanent tables that exist as a file and (2) temporary tables created when a user registers or logs in. The user information is updated in the temporary tables from the WWW pages, and when specific state transitions take place (specifically, those labeled DEQSG in Figure 2), the temporary tables are written to the permanent tables. In this way the user information is saved across state transitions.

SST is implemented as HTML with embedded forms. The usual method for handling a form embedded in an HTML document is to create a custom, common gateway interface (CGI) program that performs any actions required by the form. State transitions are then implemented as the invocation of the CGI FormServer by an HTML form.

Because the state transition takes place only on the submission of a form to the CGI FormServer, several WWW pages may exist within one state. For example, when selecting trip origins, the user is in the state labeled “Origin” in Figure 2, and the entire location tree is traversed while in the Origin state. The CGI FormServer is invoked only after a site at the base of the tree is chosen, and the trip origin coordinates are then committed to the database. The act of committing the data to the database causes the transition to either the “Destination” state (where trip destination coordinates are collected) or to the “Fine Tune” state.



Legend

- Q Query
- D Delete
- S Sort
- G Group
- Qm Match Query
- Io Initial/uses info
- I Insert

Data Management

		IN	OUT
PT	Print Trips	(List)	(Trips in List)
PGT	Print Grouped Trips	(List)	(Trips in List)
EM	Email		
PG	Print Group	(Number)	
PPS	Package Per Screen		
DTT	Delete Temp Tables		
DU	Delete User		
MQR	Match Query Report		

Data Base

- QSG (Filename)
- DIQSG (Filename)
- Qm (List, Parameter)
- Io (Name, Password....)
- DFM Data File Manager (List, Variables)
 - {Purge Date, Inception Date,
 - Round Trip (flag), Return Day
 - (flag), Day, Type, t_{d1}, t_{d2}, t_{a1},
 - t_{a2}, P_d, P_a, Group#, Trip#}
- DF Data File

Figure 2. SST state diagram

PRELIMINARY RESULTS

The SST program went “on-line” mid March 1996. In this section we present the preliminary results as of May 29, 1996, and Figure 3 shows the cumulative statistics. The solid line is the number of cumulative participants (248) and is also the total number of registered users in the database. Of these users, 42 are students and 206 are faculty/staff. The higher number may result from two factors: (1) faculty/staff have more regular and longer trips and/or (2) we have not done any significant marketing to the student body. This question will be resolved in

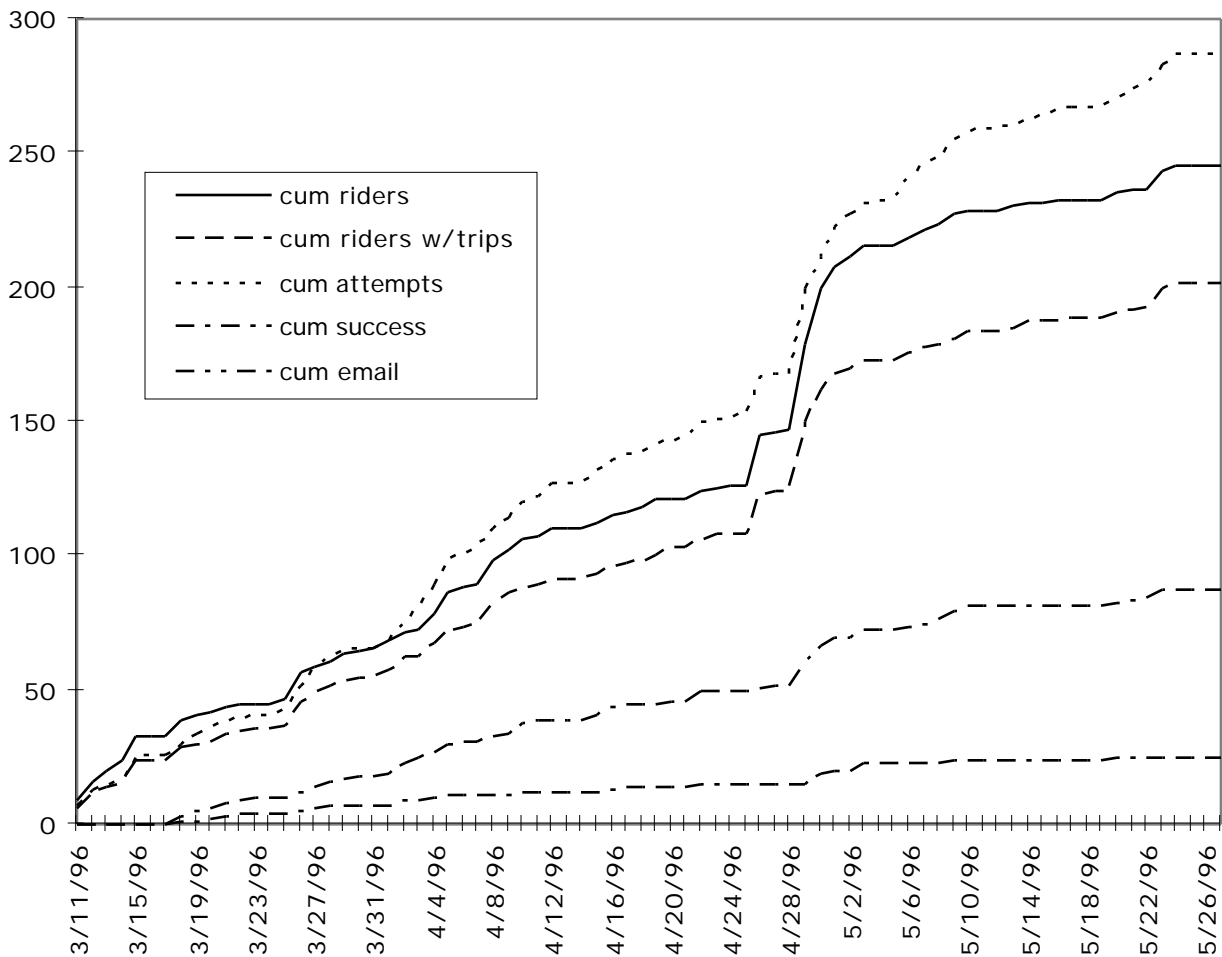


Figure 3. Usage as a function of time

fall 1996 when classes start and we begin significant marketing efforts to the students. The other lines in Figure 3 describe the temporal data for other statistics. The cumulative number of attempted matches was 293, the cumulative number of successful matches was 88, and the cumulative number of email messages sent to attempt matches was 25. Given the small number of users, these statistics suggest a surprising 35% match rate for attempted matches. At present,

funding is not available for a participant survey; therefore, reports on actual rides taken are anecdotal. As of May 29, 1996, six users have reported that they have taken rides.

OPEN QUESTIONS

SST is a system designed to enable dynamic ridesharing. A ridematching system, that matches schedules and trips, is only one component of the dynamic ridesharing equation. The other vital component is human behavior. Behavioral questions that need to be dealt with are: (1) What steps are required before someone actually gets into a car with another person? (2) How much time is needed before that decision is made? (3) Are potential ridesharers able to go beyond the traditional view of carpools as a static arrangement and willing to ride with different people on different days? A survey evaluation of users and non-users is a potential mechanism for answering these questions.

Also uncertain at this time is the extent to which SST can encourage mode shift among single occupancy vehicle (SOV) drivers. One common reason given for not carpooling is “lack of flexibility.” Can SST, which accommodates flexible and variable schedules, entice SOV drivers to a shared-ride mode? Can SST capture transit riders who might otherwise drive alone on those days when they have special travel needs? Again, time and a detailed evaluation may provide some answers.

The SST program provides a unique, real time, WWW accessible, ridematching system that uses a variety of communication media to facilitate communication between users with overlapping travel needs. Is “dynamic ridesharing” a viable concept? Perhaps an evaluation of SST will tell...