University of California Transportation Center
UCTC-FR-2010-31

The Personal Travel Assistant (PTA): Measuring the dynamics of human travel behavior

W. Recker, J. Marca, C. Rindt and
R. Dechter
University of California, Irvine
August 2010

The Personal Travel Assistant (PTA): Measuring the dynamics of human travel behavior

UNIVERSITY OF CALIFORNIA TRANSPORTATION CENTER

TECHNICAL REPORT

W. Recker, J. Marca and C. Rindt

Institute of Transportation Studies

R. Dechter

Department of Computer Science

Donald Bren School of Information and Computer Sciences

University of California, Irvine

Irvine, California 92697

September 2010

Contents

1 Introduction 1

2 Overview and Context 1

3 The Wireless Research Environment 1

3.1 Introduction 1

3.2 The target application 2

3.3 Enterprise Service Bus 4

3.4 Architectural Components 4

3.5 Problem Space 5

3.6 Future work 6

4 Creating a Behavioral Model 7

4.1 Introduction 7

4.2 Preliminaries and Definitions 8

4.3 Expectation propagation 10

4.4 Rao-Blackwellised Particle Filtering 12

4.5 The transportation model 15

4.6 Experimental Results 16

4.6.1 Finding destination or Goal of a person 17

4.6.2 Finding the Route taken by the person 20

4.7 Related Work 20

5 Simplifying the Behavioral Model 21

5.1 Introduction 21

5.2 Preliminaries 21

5.3 A simplified Dynamic Bayesian network model for recognizing human activities 22

5.4 Learning 23

5.4.1 Inference: Particle Filtering 23

5.5 Current Issues 25

6 Automatically Collecting Travel and Activity Data 26

6.1 Introduction 26

6.2 Data collection methods 27

6.3 Identifying activity locations 28

6.4 Map matching 30

6.4.1 A filtered, topological matching algorithm using A* 30

6.4.2 Implementation and Performance 34

6.5 Condensed time-space-activity trajectories 35

7 References 40

1 Introduction

The fundamental research question that was addressed with the project is whether a simple, continuously collected GPS sequence can be used to accurately measure human behavior. We applied Hybrid Dynamic Mixed Network (HDMN) modeling techniques to learn behaviors given an extended GPS data stream. This research project was designed to be an important component of a much larger effort. Unfortunately, the promised funding from a commercial sponsor for the larger project did not materialize, and so we did not have the resources to deploy a prototype personal travel assistant system. Work focused on developing the HDMN model. The learning and inference steps using the HDMN model were much slower than would be acceptable in an operational Personal Travel Assistant (PTA) system. We conducted research into alternate formulations that would improve convergence, handle noisy data more robustly and reduce the need for human intervention. This report describes how this project's results fit into the larger research context, details the work done for this UCTC grant, and outlines future directions of research based on the findings of this project.

2 Overview and Context

Wireless technologies allow people to communicate without wires. The first applications for wireless technologies are those that previously used wires such as making telephone calls and connecting computers to networks. But the really interesting applications are yet to come, and will build upon the earliest wireless devices to create applications that were previously both impossible and unimaginable.

This report documents the results of a project funded by UCTC entitled *The Personal Travel Assistant (PTA): Measuring the dynamics of human travel behavior*. The project formed one piece of a larger effort to develop the PTA system. The proposed plan of work was designed to leverage funding from other sources, including Calit2, a research unit jointly sponsored by UC Irvine and UC San Diego, as well as ZevNet, an electric vehicle shared-use vehicle demonstration project largely funded by Toyota.

Unfortunately, the externally funded projects did not materialize. Both the Calit2-funded project and the promised Toyota involvement failed to get beyond the planning stages. The direct impact on this project was that we did not deploy a prototype PTA system. That said, we were able to conduct useful research using GPS data collected from the ZevNet project, and the results will be useful going forward. The accomplishments and lessons learned from this project are detailed in this report.

3 The Wireless Research Environment

3.1 Introduction

In the early stages of preparing for this project, we considered several alternate approaches to what was meant by wireless communication in a transportation context. While we were tempted by the idea of local area networks allowing cars to talk to each other, after some discussion we decided that the most promising and practical set of problems came from considering a GPS-enabled cellphone with the ability to transmit data in real time as the platform of choice. In hindsight, this was a prescient decision, as these devices have since been widely deployed commercially, whereas vehicle to vehicle communications technologies are still experimental at best.

With a platform in mind, the next step was to detail the different kinds of applications and the communications architecture that would serve those applications. Important considerations included how individual drivers would communicate with the ATMS while still retaining their anonymity and preserving the security of their personal device (cellphone, in-dash unit, etc). Many ITS architecture documents discuss traffic probe implementations, but the implicit assumption is that the data is uploaded directly to a central service. Our discussions with Calit2 researchers from UCI and UCSD led us to the conclusion that this was unlikely to be the case for regular travelers. Instead travelers with GPS enabled phones would most likely prefer an application that did not share their travel patterns and plans with Caltrans or any other traffic agency, but rather operated entirely within the phone or on the traveler's desktop computer.

Thus it was decided to design around an Enterprise Service Bus (ESB) architecture. An ESB is a service-oriented architecture that allows services to be plugged into a network, and allows users to plug into the network and use those services. The communications bus itself is largely ignorant of the services being offered, which allows the services and use cases to be upgraded over time without needing to change the communications network. This also allowed us to modularize our research efforts. For example, this project could concentrate on developing a travel behavior modeling service, while other projects could concentrate on the ESB architecture, the hardware devices, and so on.

3.2 The target application

The general class of problems that we addressed with this research project are the multifaceted impacts that advances in information and communications technology (ICT) will have upon the market for traveler information services. There has been research supporting the idea that providing good information will improve each individual's travel choices, especially as related to improved route choice and improved departure time choice (Mahmassani and Jayakrishnan, 1991; Ben-Akiva et al., 1991; Jayakrishnan, 1994; Kim et al., 2005). But, simply providing information is not necessarily a good thing. Arnott et al. (1991) argue that in the absence of congestion tolls, the information has to be perfect to guarantee that the expected travel costs will be reduced. The same authors later argue that imperfect information can reduce social welfare in general (Arnott et al., 1999). Neither of these studies considered the real-time dynamics of travel (accidents, unplanned trips, and so on) which cause capacity, demand, and delays to vary in real time. One presumes that providing perfect information would imply perfect real-time information, and very good predictions of future downstream conditions most relevant to a traveler's trip. Indeed, Levinson (2005) built a model of congestion as a two-player game in which the players are presumed to have perfect information. There is also the compounding problem of induced demand (Noland, 2001) at off peak times resulting from the improved off-peak travel times (due to improved departure time choices). Being able to satisfy additional demand with the same supply is in general a good thing, but will result in a system with even more pervasive queuing and congestion.

In short, it is our expectation that ICT will be used more and more to increase the efficient use of the available transportation supply. At the same time, it will become more and more difficult to make trip decisions *without* leveraging available ICT information streams. By the time a traveler has entered his or her vehicle and has turned on the "in-dash" navigation system, it is likely to be too late to make any effective decisions about route choice, destination choice, or even higher level trip and tour goals. For this reason our PTA application is based on a cellphone platform, and is intended to learn total travel behavior patterns, so as to suggest optimal travel alternatives, including rescheduling or foregoing trips altogether, at any time such advice could be useful.

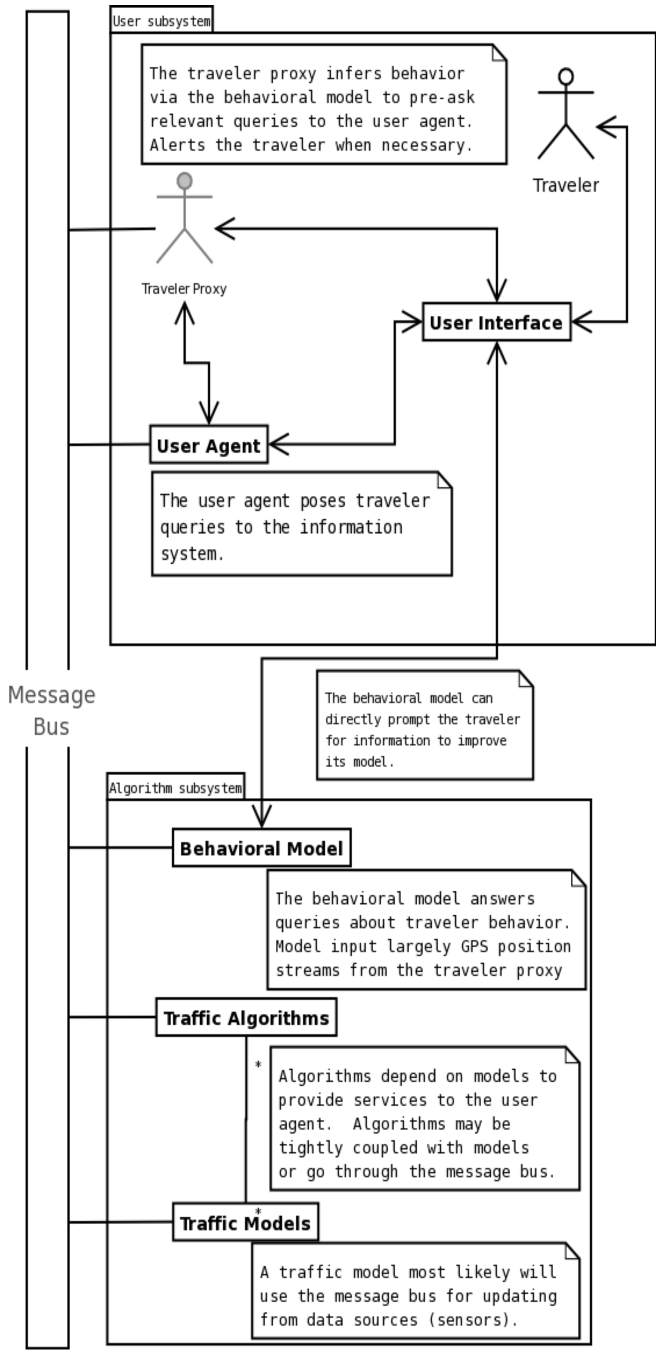
User subsystem
The traveler proxy infers behavior via the behavioral model to pre-ask relevant queries to the user agent. Alerts the traveler when necessary.
Traveler
Traveler Proxy
User Interface
User Agent
The user agent poses traveler queries to the information system.
Message Bus
The behavioral model can directly prompt the traveler for information to improve its model.
Algorithm subsystem
Behavioral Model
The behavioral model answers queries about traveler behavior. Model input largely GPS position streams from the traveler proxy
Traffic Algorithms
*
Algorithms depend on models to provide services to the user agent. Algorithms may be tightly coupled with models or go through the message bus.
*
Traffic Models
A traffic model most likely will use the message bus for updating from data sources (sensors).

Figure 1. The ESB system architecture

The system shown in Figure 1 was developed as the initial architecture for the PTA. The architecture consists of two subsystems: the user subsystem and the algorithm subsystem. These systems are mirrored in the real world implementation that was planned for the intial development: (1) client software that runs on a GPS-enabled cellphone, and (2) server software that supports the client, running on a personal computer. The client transmits messages (events and information requests) from the traveler to the server. The server processes the messages from the client, and if necessary produces an appropriate response to be displayed to the user. The nature of the application is highly distributed and adaptive, allowing for multiple clients (in-dash navigation units, hand held cellphones, *and* desktop widgets), and allowing the data processing algorithms and other services to change on the server side to reflect new innovations or new client capabilities.

3.3 Enterprise Service Bus

Given this initial conception, the key design decision behind the proposed architecture was to use an Enterprise Service Bus (ESB) to provide the platform for integration of the embedded devices with the back-end structure. The core idea behind an ESB architecture is a message bus that provides the communication infrastructure among the various components of the application. These components may include existing applications; data sources; new services; portals to services; and even workflows of different services. All communications amongst these components are triggered by message events. The bus is responsible for processing events, filtering them if necessary, and then delivering the events to the appropriate modules in the system. The modules receive messages, perform the required actions, and then generate a response event to be sent over the bus. The messages are often structured around various defined data types.

The proposed ESB-based architecture has distinct advantages to its alternatives, which include a structured CORBA architecture and various Web Services-based approaches. The primary advantage of an ESB is that it is loosely coupled. Approaches like CORBA require more tightly coupled components.

This tight coupling hinders the flexibility of the architecture and therefore stifles its evolution and its applicability to the heterogeneous and dynamic world of travel and transportation. The loose coupling of the ESB architecture makes a number of interesting applications possible. For example, a PTA can both consume services offered by a centralized ATMS, as well as offer services back to the ATMS and to other PTA clients to use. Other advantages of the ESB include mechanisms for ensuring quality of service—including reliability, fault tolerance, scalability, and security—as well as for encouraging the reuse of services.

3.4 Architectural Components

As shown in *Figure* 1, the architecture for this application can be broken into two subsystems, along the lines of the two different parts of this application: the user subsystem and the algorithm subsystem. The user subsystem contains the components that must be replicated for every user, on each user's device (cellphone, navigation system, etc). The components in the user subsystem include:

- a user interface,
- a user agent, and
- a traveler proxy

The intent behind these three systems is to limit the required interaction with the traveler. The initial configuration of the components of the user subsystem may contain commonalities within social groups such as households or fleets, but the user should also be able to tailor the configuration of the traveler proxy and the user interface.

The second subsystem, the algorithm subsystem, encapsulates the components that will be deployed on the server side, which may physically reside on home computers, public Web servers, or anywhere in the "cloud". These components will include various data sources, e.g. the Caltrans loop sensors, several different traffic models, and the various algorithms that can solve different travel problems, including shortest path, fastest path, best order of activities, etc.

There are four main causes of interaction among the various components in the two subsystems: (1) the traveler will make a direct query to the PTA system, (2) the traveler proxy will make a query based on the inferences of the behavioral model, (3) the traveler proxy will send position updates to the behavioral model, and (4) the traveler proxy will send queries to the behavioral model. These interactions go through two components of the user subsystem, the user agent, and the traveler proxy.

The user agent accepts queries from the traveler, and sends them along to the back-end structure. The traveler proxy exists to pre-fetch possibly relevant information from the back-end, so as to speed up actual queries from the traveler. The user agent and the traveler proxy can interact directly to streamline the user experience. All queries are transmitted to the Enterprise Service Bus, and are dispatched to the appropriate algorithm or data service that can generate a response.

The system is distinguished from competitors in a number of ways. First and foremost, it is based upon two sets of models: one set of models for the individual user's behavior, and one set of models for the urban system (and the transportation system in particular). This UCTC project has directly advanced the development of the user behavior models. The point of the individual behavioral models is to minimize the required interaction with the user by inferring the user's plans, and allowing system monitoring based on the inferred plans. The models of the transportation system will allow this decision support tool to make more informed recommendations based on predicted system behavior rather than currently measured behavior. Transportation system models are the subject of other, related research projects. Finally this overall system has been designed to leverage existing technology as much as possible, while still allowing new technology to be added when it becomes available. The basic functionality is possible today by using Android phones and iPhones.

3.5 Problem Space

From a traveler's perspective, the PTA is useful only to the extent that it helps the traveler perform his or her activities in an optimal manner in the face of an uncertain environment. Figure 2 describes some of the problems the traveler may face, and for which the PTA will provide additional information and decision support. The problems shown—route choice, destination choice, tour planning, and activity scheduling—are largely hierarchical. The higher problems tend to subsume the lower problems. For instance, the destination choice problem solution will include a solution for the shortest path to the destination. The hierarchy is not strict, however—planning a tour may involve sequencing trips among a set of given destinations.

In the context of these problems, the human social system and the various systems of the built environment generate and constrain human activity. The user agent needs to have access to models of these encompassing systems to perform its function. These models are contained in the algorithm subsystem of the PTA architecture shown in Figure 1.

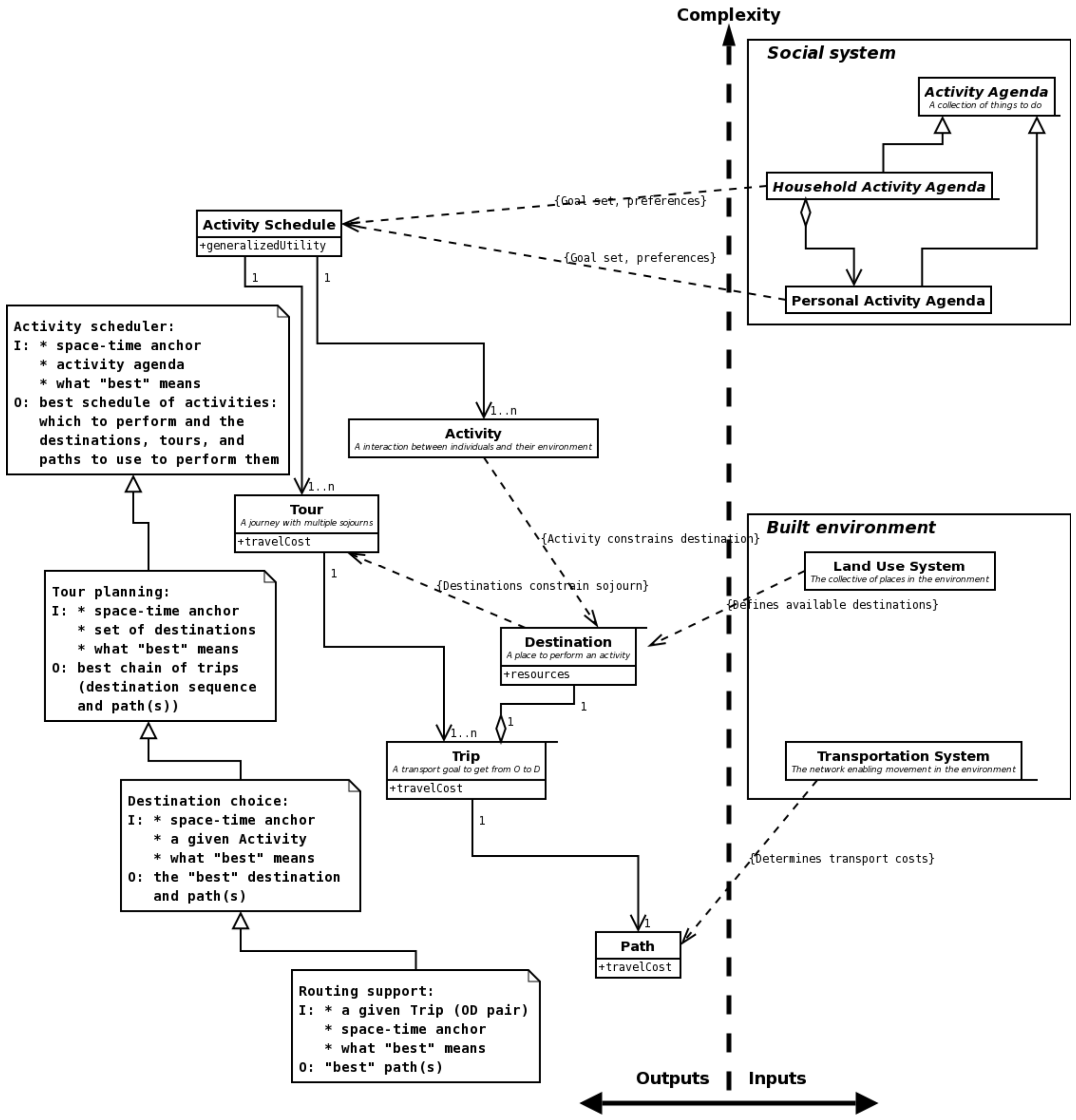
Complexity
Social system
Activity Agenda
A collection of things to do
Household Activity Agenda
Personal Activity Agenda
{Goal set, preferences}
{Goal set, preferences}
Activity Schedule
+generalizedUtility
1
1
Activity scheduler:
I: * space-time anchor
* activity agenda
* what "best" means
O: best schedule of activities: which to perform and the destinations, tours, and paths to use to perform them
1..n
Activity
A interaction between individuals and their environment
1..n
Tour
A journey with multiple sojourns
+travelCost
{Activity constrains destination}
Built environment
Land Use System
The collective of places in the environment
{Destinations constrain sojourn}
{Defines available destinations}
Tour planning:
I: * space-time anchor
* set of destinations
* what "best" means
O: best chain of trips (destination sequence and path(s))
Destination
A place to perform an activity
+resources
1
1
1..n
Trip
A transport goal to get from O to D
+travelCost
Transportation System
The network enabling movement in the environment
Destination choice:
I: * space-time anchor
* a given Activity
* what "best" means
O: the "best" destination and path(s)
1
{Determines transport costs}
1
Path
+travelCost
Routing support:
I: * a given Trip (OD pair)
* space-time anchor
* what "best" means
O: "best" path(s)
Outputs
Inputs

Figure 2. Some of the problems and the related domains that a typical traveler will face that can be served by a PTA system

This UCTC grant directly supported work to develop individualized travel behavior models from observed travel data. These models are also useful for these server-side algorithms. Note that each of the four problem areas—route choice, destination choice, tour planning, and activity scheduling—all require personalized notions of what "best" means. For example, when considering routing support, one traveler might prefer streets over freeways. This information can be gleaned from the observed paths that the traveler takes, and from the responses of the traveler to suggested routes. Without a behavior model that can learn high level behavior from low level inputs, these algorithm services will not be as useful to the traveler.

3.6 Future work

The intended goal of developing the ESB architecture (Figure 1), building a block diagram of a traveler's problem space (Figure 2), and developing a variety of use cases was to design, build, and deploy a real Personal Travel Assistant implementation. We do not currently have funding for such an endeavor, but the cost of developing such a system has been steadily declining. When we started this research, GPS-enabled cellphones were rare, and the standard way to get applications onto phones was to program them in Java and hope that a cellular carrier was willing to deploy your application. Since then, Apple has released the iPhone and its app store, Google has responded with Android and its app store, and almost overnight GPS-enabled smart phones are commonplace.

By focusing our efforts to integrate travelers with traditional infrastructure-based information streams such as Caltrans CMSs and loop detector feeds, we were forced to step outside of the usual vertically-integrated design approach, and instead adopt a more decentralized, federated approach. Our ESB design allows independent vendors and devices to offer and consume software services via a common communications infrastructure. It doesn't just apply to travelers seeking better travel patterns. It also applies to researchers and consultants who wish to integrate their services with the next generation of navigational aids, to data aggregators who want to build a better model of real-time traffic conditions, and even to state agencies who might want to conduct low-cost origin-destination surveys.

4 Creating a Behavioral Model

4.1 Introduction

In the preceding section, the need for a traveler information system and its component subsystems was motivated by the needs of the individual traveler. The transportation-related problem domains that an individual faces are depicted in Figure 2, and a conceptual system that can help solve those problems is diagrammed in Figure 1. The key benefit of the PTA's design is that the Behavioral Model can predict a person's future travel and activity behavior by processing the travel patterns it has observed. The ability to infer future actions then allows the "Traveler Proxy" shown in Figure 1 to formulate and "pre-ask" queries that the traveler may want to ask explicitly but is busy doing something else. For example, the likelihood of a major accident happening on a worker's home-bound commute path is very small—so small that only a tiny fraction of commuters will spare the time and effort required to go to the internet and check traffic before getting in their cars. In contrast, it is no trouble at all for an automated script to anticipate the need to check traffic conditions starting 30 minutes before quitting time, and only interrupt the traveler if something major comes up.

Thus the core of the PTA wireless device's functionality depends on the Behavioral Model, which is the primary subject of this project's research. Modeling sequential real-life domains often requires the ability to represent both probabilistic and deterministic information. Hybrid Dynamic Bayesian Networks (HDBNs) were recently proposed for modeling such phenomena Lerner (2002). In essence, these are factored representations of Markov processes that allow discrete and continuous variables. Since they are designed to express uncertain information they represent constraints as probabilistic entities which may have negative computational consequences. To address this problem Dechter and Mateescu (2004) and Larkin and Dechter (2003) introduced the framework of Mixed Networks. The work documented here extended the Mixed Networks framework to dynamic environments, allowing continuous Gaussian variables, yielding Hybrid Dynamic Mixed Networks (HDMN). We address the algorithmic issues that emerge from this extension and demonstrate the potential of our approach on a complex dynamic domain of a person's transportation routines using real travel data.

Focusing on algorithmic issues, the most popular approximate query processing algorithms for dynamic networks are Expectation propagation(EP) Heskes and Zoeter (2002) and Rao-Blackwellised Particle Filtering (RBPF) Doucet et al. (2000). We therefore extend these algorithms to accommodate and exploit discrete constraints in the presence of continuous probabilistic functions. Extending Expectation Propagation to handle constraints is easy, extension to continuous variables is a little more intricate but still straightforward. The presence of constraints introduces a principles challenge for Sequential Importance Sampling algorithms, however. Indeed the main algorithmic contribution of this research was developing a class of Rao-Blackwellised Particle Filtering algorithm, IJGP-RBPF for HDMNs which integrated a Generalized Belief Propagation component with the Rao-Blackwellised Particle Filtering scheme.

Our motivation for developing HDMNs as a modeling framework is more general than simply satisfying the needs of the PTA application sketched in Figure 2.1. A whole range of problems in the transportation literature depend upon reliable estimates of the prevailing demand for travel over various time scales. At one end of this range, there is a pressing need for accurate and complete estimation of the global origins and destinations (O-D) matrix at any given time for an entire urban area. Such estimates are used in both urban planning applications Sherali et al. (2003) and integrated traffic control systems based upon dynamic traffic assignment techniques Peeta and Zilaskopoulos (2001). Even the most advanced techniques, however, are hamstrung by their reliance upon out-dated, pencil-and-paper travel surveys and sparsely distributed detectors in the transportation system.

We view the increasing proliferation of powerful mobile computing devices as an opportunity to remedy this situation. If even a small sample of the traveling public agreed to collect their travel data and make that data publicly available, transportation management systems could significantly improve their operational efficiency. At the other end of the spectrum, personal traffic assistants running on the mobile devices could help travelers replan their travel when the routes they typically use are impacted by failures in the system arising from accidents or natural disasters. A common starting point for these problems is to develop an efficient formulation for learning and inferring individual traveler routines like traveler's destination and his route to destination from raw data points.

The following sections are organized as follows. In the next section, we discuss preliminaries and introduce our modeling framework. We then describe two approximate inference algorithms for processing HDMN queries: an Expectation Propagation type and a Particle Filtering type. Subsequently, we describe the transportation modeling approach and present preliminary empirical results on how effectively a model is learnt and how accurately its predictions are given several models and a few variants of the relevant algorithms.

The contribution of this research is addressing a complex and highly relevant real life domain using a general framework and domain independent algorithms, thus allowing systematic study of modeling, learning and inference in a non-trivial setting.

In addition to the support provided by the ATMS Testbed, part of the research documented here was supported by the National Science Foundation.

4.2 Preliminaries and Definitions

Graphical models help both to visualize and encode the relationships between variables. While they are easily described for systems with three or four variables, their strengths become apparent when many variables are involved, such as is the case with travel routing and planning. One of the nice features of graphical models is that they can be partitioned into largely independent subgraphs. For example, a network system representing the relationships between second by second GPS readings, transportation links, destinations, and activity goals might appear to be overwhelming, except that each time slice can be partitioned from its neighbors and analyzed with just key information about its predecessor subgraphs.

Bayesian Networks are graphical models that specify the probabilistic dependencies between variables. Typically, Bayesian networks contain just discrete variables. The term *Hybrid* is usually used for graphical models that have both discrete and continuous variables. To model travel behavior, in which the variables are both continuous (time and space) and discrete (destination chosen, link used, etc.), Hybrid Bayesian Networks are required.

Formally, ***Hybrid Bayesian Networks (HBN)*** Lauritzen (1992) are graphical models defined by a tuple $\mathcal{B} = (X,G,P)$, where X is the set of variables, G is a directed acyclic graph whose nodes corresponds to the variables, and $P = \{P_1,...,P_n\}$ is a set of conditional probability distributions (CPD). Given variable x_i and its parents in the graph $pa(x_i)$, the conditional probability P_i is defined as $P_i = P(x_i|pa(x_i))$. In a hybrid network, X is partitioned into discrete and continuous variables $X = \Gamma \cup \Delta$, respectively. Further, the graph structure G is restricted such that continuous variables cannot have discrete variables as their child nodes. The conditional distribution of continuous variables are given by a linear Gaussian model: $P(x_i|I = i, Z = z) = N(\alpha(i)+\beta(i)*z, \gamma(i))$ $x_i \in \Gamma$ where Z and I are the set of continuous and discrete parents of x_i, respectively and $N(\mu,\sigma)$ is a multivariate normal distribution. The network represents a joint distribution over all its variables given by a product of all its CPDs.

A ***Constraint Network*** Dechter (2003) is a graphical model $\mathcal{R} = (X,D,C)$, where $X = \{x_1,...,x_n\}$ is the set of variables, $D = \{D_1,...,D_n\}$ is their respective discrete domains and $C = \{C_1,C_2,...,C_m\}$ is the set of constraints. Each constraint C_i is a relation R_i defined over a subset of the variables $S_i \subseteq X$ and denotes the combination of values that can be assigned simultaneously. A *Solution* is an assignment of values to all the variables such that no constraint is violated. The primary query is to decide if the constraint network is consistent and if so find one or all solutions.

The recently proposed ***Mixed Network*** framework Dechter and Mateescu (2004) for augmenting Bayesian Networks with constraints, can immediately be applied to HBNs yielding the ***Hybrid Mixed Networks (HMNs)***. Formally, given a HBN $\mathcal{B} = (X,G,P)$ that expresses the joint probability $P_{\mathcal{B}}$ and given a constraint network $\mathcal{R} = (X,D,C)$ that expresses a set of solutions ρ, an HMN is a pair $\mathcal{M} = (\mathcal{B},\mathcal{R})$. The discrete variables and their domains are shared by $\mathcal{B}$ and $\mathcal{R}$ and the relationships are those expressed in P and C. We assume that $\mathcal{R}$ is consistent. The mixed network $\mathcal{M} = (\mathcal{B},\mathcal{R})$ represents the conditional probability $P_{\mathcal{M}}(x) = P_{\mathcal{B}}(x|x \in \rho)$ if $x \in \rho$ and 0 otherwise.

Dynamic Bayesian Networks are Markov models whose state-space and transition functions are expressed in a factored form using Bayesian Networks. They are defined by a prior $P(X_0)$ and a state transition function $P(X_{t+1}|X_t)$. Hybrid Dynamic Bayesian Networks (HDBNs) allow continuous variables while Hybrid Dynamic Mixed Networks (HDMNs) also permit explicit discrete constraints.

DEFINITION **3.1** *A* ***Hybrid Dynamic Mixed Network*** **(HDMN)** *is a pair $(M_0, M_{\rightarrow})$, defined over a set of variables $X = \{x_1,...,x_n\}$, where M_0 is an HMN defined over X representing $P(X_0)$. $M_{\rightarrow}$ is a 2-slice network defining the stochastic process $P(X_{t+1}|X_t)$. The* 2-time-slice Hybrid Mixed network (2-THMN) *is an HMN defined over $X' \cup X''$ such that X' and X'' are identical to X. The acyclic graph of the probabilistic portion is restricted so that nodes in X' are root nodes and have no CPDs associated with* *them. The constraints are defined the usual way. The 2-THMN represents a conditional distribution $P(X'|X)$.*

The semantics of any dynamic network can be understood by unrolling the network to T time-slices. Namely, $P(X_{0:t}) = P(X_0)*\mathbf{Q}_{t=1}^{T} P(X_t|X_{t-1})$ where each probabilistic component can be factored in the usual way, yielding a regular HMN over T copies of the state variables.

The most common task over Dynamic Probabilistic Networks is filtering and prediction. Filtering is the task of determining the belief state $P(X_t|e_{0:t})$ where X_t is the set of variables at time t and $e_{0:t}$ are the observations accumulated at time-slices 0 to t. Performing filtering in dynamic networks is often called *online inference* or *sequential inference*.

Filtering can be accomplished in principle by unrolling the dynamic model and using any state-of-the art exact or approximate reasoning algorithm. For example, one could use the join-tree clustering algorithm. When all variables in a dynamic network are discrete, the treewidth of the dynamic model is at most $O(n)$ where n is the number of variables in a time-slice (see [Murphy, 2002] for details). However, when continuous Gaussian variables are mixed with discrete ones, the treewidth of HDMN is at least $O(T)$ when T is the number of time slices. This is because to maintain correctness of join-tree propagation, we have to eliminate all continuous variables before the discrete ones which creates a clique between discrete variables from time slices $1, \ldots, T$. Thus exact inference is clearly infeasible in dynamic graphical models having both discrete and continuous variables [Lauritzen, 1992, Lerner, 2002]. Therefore the applicable approximate inference algorithms for Hybrid Dynamic Networks are either sampling-based such as Particle Filtering or propagation-based such as Expectation Propagation. In the next two sections, we will extend these algorithms to HDMNs.

4.3 Expectation propagation

In this section we extend an approximate inference algorithm called Expectation Propagation (EP) Heskes and Zoeter (2002) from HDBNs to HDMNs. The idea in EP (forward pass) is to perform Belief Propagation by passing messages between slices t and $t+1$ along the ordering $t = 0$ to T. EP can be thought of as an extension of Generalized Belief Propagation (GBP) to HDBNs Heskes and Zoeter (2002). For simplicity of exposition, we will extend a GBP algorithm called Iterative Join graph propagation Dechter et al. (2002) to HDMNs and call our technique IJGP(i)-S where "S" denotes that the process is sequential. The extension is rather straight-forward and can be easily derived by integrating the results in Murphy (2002); Dechter et al. (2002); Lauritzen (1992); Larkin and Dechter (2003).

IJGP (Dechter et al., 2002) is a Generalized Belief Propagation algorithm which performs message passing on a join-graph. A join-graph is collection of cliques or clusters such that the interaction between the clusters is captured by a graph. Each clique in a join-graph contains a subset of variables from the graphical model. IJGP(i) is a parameterized algorithm which operates on a join-graph which has less than $i+1$ discrete variables in each clique. The complexity of IJGP(i) is bounded exponentially by i , also called the i-bound. In the message-passing step of IJGP(i), a message is sent between any two nodes that are neighbors of each other in the join-graph. A message sent by node N_i to N_j is constructed by multiplying all the functions and messages in a node (except the message received from N_j) and marginalizing on the common variables between N_j and N_i (see Dechter et al., 2002).

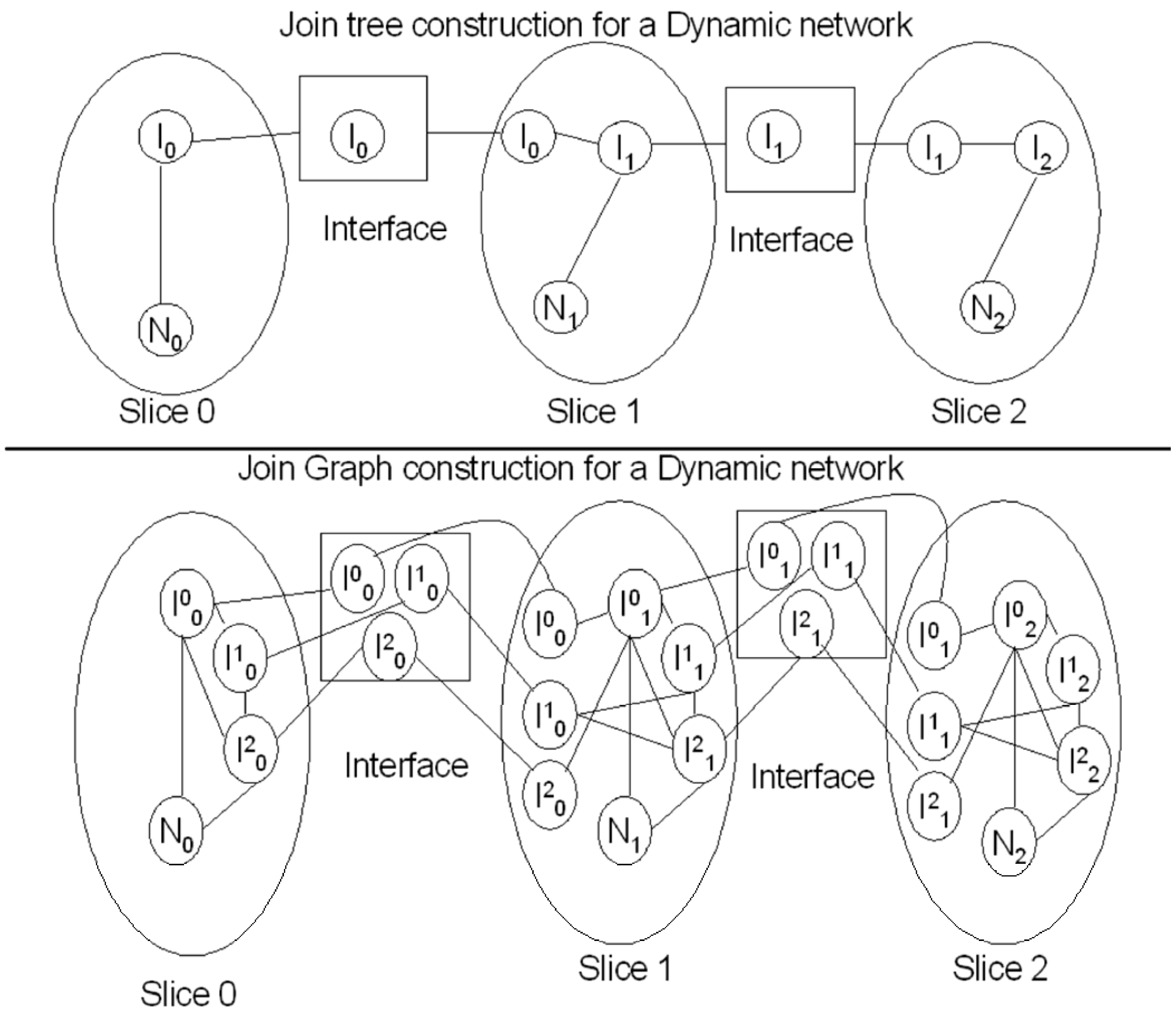

Figure 3 Schematic illustration of the Procedure used for creating join-graphs and join-trees of HDMNs

IJGP(i) can be easily adapted to HDMNs (which we call IJGP(i)-S) and we describe some technical details here rather than a complete derivation due to lack of space. Note that because we are performing online inference, we need to construct the join-graph used by IJGP(i)-S in an online manner rather than recomputing the join-graph every time new evidence arrives. Murphy Murphy (2002) describes a method to compute a join-tree in an online manner by pasting together join-trees of individual time-slices using special cliques called the interface. Dechter et al. (2002) describe a method to compute join-graphs from join-trees. The two methods can be combined in a straightforward way to come up with an online procedure for constructing a join-graph. In this procedure, we split the interface into smaller cliques such that the new cliques have less than *i+1* variables. This construction procedure is shown in Figure 3.

Message-passing is then performed in a sequential way as follows. At each time-slice t, we perform message-passing over nodes in t and the interface of t with $t -1$ and $t +1$ (shown by the ovals in Figure 3). The new functions computed in the interface of t with $t +1$ are then used by $t +1$, when we perform message passing in $t +1$.

Three important technical issues remain to be discussed. First, message-passing requires the operations of multiplication and marginalization to be performed on functions in each node. These operators can be constructed for HDMNs in a straightforward way by combining the operators by Lauritzen (1992) and Larkin and Dechter (2003) that work on HBNs and discrete mixed networks respectively. We will now briefly comment on how the multiplication operator can be derived. Let us assume we want to multiply a collection of probabilistic functions P' and a set of constraint relations C' (which consist of only discrete tuples allowed by the constraint) to form a single function PC. Here, multiplication can be performed on the functions in P' and C' separately using the operators in Lauritzen (1992) and Dechter (2003) respectively to compute a single probabilistic function P and a single constraint relation C. These two functions P and C can be multiplied by deleting all tuples in P that are not present in C to form the required function PC.

Second, because IJGP(i)-S constructs join-graphs sequentially, the maximum-i-bound for IJGP(i)-S is bounded by the treewidth of the time slice and its interfaces and not the treewidth of the entire HDMN model (see Figure 3).

Third, IJGP(i) guarantees that the computations will be exact if i is equal to the treewidth. This is not true for IJGP(i)-S in general as shown in Lerner (2002). It can be proved that:

THEOREM **4.1** *The complexity of IJGP(i)-S is $O(((|\Delta_t| + n) * d^i * \Gamma_t^3) * T)$ where $|\Delta_t|$ is the number of discrete variables in time-slice t, d is the maximum-domain size of the discrete variables, i is the i-bound used, n is the number of nodes in a join-graph of the time-slice, Γ_t is the maximum number of continuous variables in the clique of the join-graph used and T is the number of time-slices.*

4.4 Rao-Blackwellised Particle Filtering

In this section, we will extend the Rao-Blackwellised Particle Filtering algorithm (Doucet et al., 2000) from HDBNs to HDMNs. Before, we present this extension, we will briefly review Particle Filtering and Rao-Blackwellised Particle Filtering (RBPF) for HDBNs.

Particle filtering uses a weighted set of samples or particles to approximate the filtering distribution. Thus, given a set of particles $X_t^1,\dots,X_t^N$ approximately distributed according to the target-filtering distribution $P(X_t = M|e_{0:t})$, the filtering distribution is given by $P(X_t = M|e_{0:t}) = 1/N\mathbf{P}_{i=1}^{N}\delta(X_t^i = M)$ where $\delta$is the Dirac-delta function. Since we cannot sample from $P(X_t = M|e_{0:t})$ directly, particle filtering samples from an appropriate (importance) proposal distribution $Q(X)$. The particle filter starts by generating N particles according to an initial proposal distribution $Q(X_0|e_0)$. At each step, it generates the next state $X_{t+1}{}^i$ for each particle X_t^i by sampling from $Q(X_{t+1}|X_t^i,e_{0:t})$. It then computes the weight of each particle based given by $w_t = P(X)/Q(X)$ to compute a weighted distribution and then *re-samples* from the weighted distribution to obtain a set of un-biased or un-weighted particles.

Particle filtering often shows poor performance in high-dimensional spaces and its performance can be improved by sampling from a sub-space by using the *Rao-Blackwellisation (RB) theorem* (and the particle filtering is called Rao-Blackwellised Particle Filtering (RBPF)). Specifically, the state X_t is divided into two-sets: R_t and Z_t such that only variables in set R_t are sampled (from a proposal distribution $Q(R_t)$) while the distribution on Z_t is computed analytically given each sample on R_t (assuming that $P(Z_t|R_t,e_{0:t},R_{t-1})$ is tractable). The complexity of RBPF is proportional to the complexity of exact inference step i.e. computing $P(Z_t|R_t,e_{0:t},R_{t-1})$ for each sample R_t^k. w-cutset (Bidyuk and Dechter, 2004) is a parameterized way to select R_t such that the complexity of computing $P(Z_t|R_t,e_{0:t},R_{t-1})$ is bounded exponentially by w. Below, we use the w-cutset idea to perform RBPF in HDMNs.

Since exact inference can be done in polynomial time if a HDBN contains only continuous variables, a straightforward application of RBPF to HDBNs involves sampling only the discrete variables in each time slice and exactly inferring the continuous variables (Lerner, 2002).

Extending this idea to HDMNs, suggests that in each time slice t we sample the discrete variables and discard all particles that violate the constraints in the time slice. Let us assume that we select a proposal distribution Q that is a good approximation of the probabilistic filtering distribution but ignores the constraint portion. The extension described above can be inefficient because if the proposal distribution Q is such that it makes non-solutions to the constraint portion highly probable, most samples from Q will be rejected (because these samples R_t^i will have $P(R_t^i) = 0$ and so the weight will be zero). Thus, on one extreme sampling only from the Bayesian Network portion of each time-slice may lead to potentially high rejection-rate.

On the other extreme, if we want to make the sample rejection rate zero we would have to use a proposal distribution Q' such that all samples from this distribution are solutions. One way to find this proposal distribution is to make the constraint network backtrack-free (using adaptive-consistency (Dechter, 2003) or exact constraint propagation) along an ordering of variables and then sample along the reverse ordering. Another approach is to use join-tree-clustering which combines probabilistic and deterministic information and then sample from the join-tree. However, both join-tree-clustering and adaptive-consistency are time and space exponential in treewidth and so they are costly when the treewidth is large. Thus on one hand, zero-rejection rate implies using a potentially costly inference procedure. On the other hand, sampling from a proposal distribution that ignores constraints may result in a high rejection rate.

We propose to exploit the middle ground between the two extremes by combining the constraint network and the Bayesian Network into a single approximate distribution Ω_{app} using IJGP(i), which is a bounded inference procedure. Note that because IJGP(i) has polynomial time complexity for constant i, we would not eliminate the sample-rejection rate completely. However, by using IJGP(i) we are more likely to reduce the rejection-rate because IJGP(i) also achieves Constraint Propagation and it is well known that Constraint Propagation removes many inconsistent tuples thereby reducing the chance of sampling a non-solution (Dechter, 2003).

```
Algorithm IJGP-RBPF
Input: A Hybrid Dynamic Mixed Network $(X,D,G,P,C)_{0:T}$ and an observation sequence $e_{0:T}$ Integers N,
w and i.
Output: $P(X_T|e_{0:T})$
    For $t = 0$ to $T$ do
        Sequential Importance Sampling step:
        Generalized Belief Propagation step
        Use IJGP(i) to compute the proposal distribution $\Omega_{app}$
        Rao-Blackwellisation step
```

```
Partition the Variables $X_t$ into $R_t$ and $Z_t$ such that the treewidth of a join-tree of $Z_t$ is $w$.
Sampling step
For $i = 1$ to $N$ do
    Generate a $R_t^i$ from $\Omega_{app}$.
    reject sample if $r_t^i$ is not a solution. i=i-1;
    Compute the importance weights $w_t^i$ of $R_t^i$.
    Normalize the importance weights to form $\tilde{w}_t^i$.
Selection step:
Resample N samples from $\hat{R}_t^i$ according to the normalized importance weights $\tilde{w}_t^i$ to obtain
new N random samples.
Exact step:
For i = 1 to N do
    Use join-tree-clustering to compute the distribution on $Z_t^i$ given $Z_{t-1}^i$, $e_t$, $\hat{R}_t^i$ and $\hat{R}_{t-1}^i$.
```

Figure 4: IJGP-RBPF for HDMNs

Another important advantage of using IJGP(i) is that it yields very good approximations to the true posterior (see a companion paper Gogate and Dechter, 2005, for empirical results) thereby proving to be an ideal candidate for proposal distribution. Note that IJGP(i) can be used as a proposal distribution because it can be proved using results from Dechter and Mateescu (2003) that IJGP(i) includes all supports of $P(X_t|e_{1:t},X_{t-1})$ (i.e $P(X_t|e_{1:t},X_{t-1}) > 0$ implies that the output of IJGP(i) viz. $Q > 0$).

Note that IJGP(i) we use here is different from the algorithm IJGP(i)-S that we described in the previous section. This is because in our RBPF procedure, we need to compute an approximation to the distribution $P(R_t|R_{t-1}^k,e_{0:t})$ given the sample R_{t-1}^k on variables R_{t-1} and evidence $e_{0:t}$. IJGP(i) as used in our RBPF procedure works on HMNs and can be derived using the results in Dechter et al. (2002), Lauritzen (1992), and Larkin and Dechter (2003). For lack of space we do not describe the details of this algorithm (see a companion paper Gogate and Dechter, 2005, for details).

The integration of the ideas described above into a formal algorithm called IJGP-RBPF is given in Figure 4. It uses the same template as in Doucet et al. (2000) and the only step different in IJGP-RBPF from the original template is the implementation of the Sequential Importance Sampling step (SIS).

SIS is divided into three-steps: (1) In the Generalized Belief Propagation step of SIS, we first perform Belief Propagation using IJGP(i) to form an approximation of the posterior (say Ω_{app}) as described above. (2) In the Rao-Blackwellisation step, we first partition the variables in a 2THMN into two sets R_t and Z_t using a method due to Bidyuk and Dechter (2004). This method Bidyuk and Dechter (2004) removes minimal variables R_t from X_t such that the treewidth of the remaining network Z_t is bounded by w. (3) In the sampling step, the variables R_t are sampled from Ω_{app}. To generate a sample from Ω_{app}, we use a special data-structure of ordered buckets which is described in a companion paper Gogate and Dechter (2005). Importance weights are computed as usual Doucet et al. (2000).

Finally, the exact-step computes a distribution on Z_t using join-tree-clustering for HMNs (see a companion paper Gogate and Dechter (2005) for details on join-tree-clustering for HMNs). It can be proved that:

THEOREM **5.1** *The complexity of IJGP-RBPF(i,w) is $O([N_R * d^{w+1} + ((|\Delta| + n) * (d^i * |\Gamma|^3))] * T)$ where $|\Delta|$ is the number of discrete variables, d is the maximum-domain size of the discrete variables, i is the adjusted-i-bound, w is defined by w-cutset, n is the number of nodes in a join-graph, Γ is the number of continuous variables in a 2-THMN, N_R is the number of samples actually drawn and T is the number of* *time-slices.*

4.5 The transportation model

In this section, we describe the application of HDMNs to a real-world problem of inferring car travel activity of individuals. The major query in our HDMN model is to predict where a traveler is likely to go and what his/her route to the destination is likely to be, given the current location of the traveler's car. This application was described in Liao et al. (2004) in a different context for detecting abnormal behavior in Alzheimer's patients and they use a Abstract Hierarchical Markov Models (AHMM) for reasoning about this problem. The novelty in our approach is not only a more general modeling framework and approximate inference algorithms but also a domain independent implementation which allows an expert to add and test variants of the model.

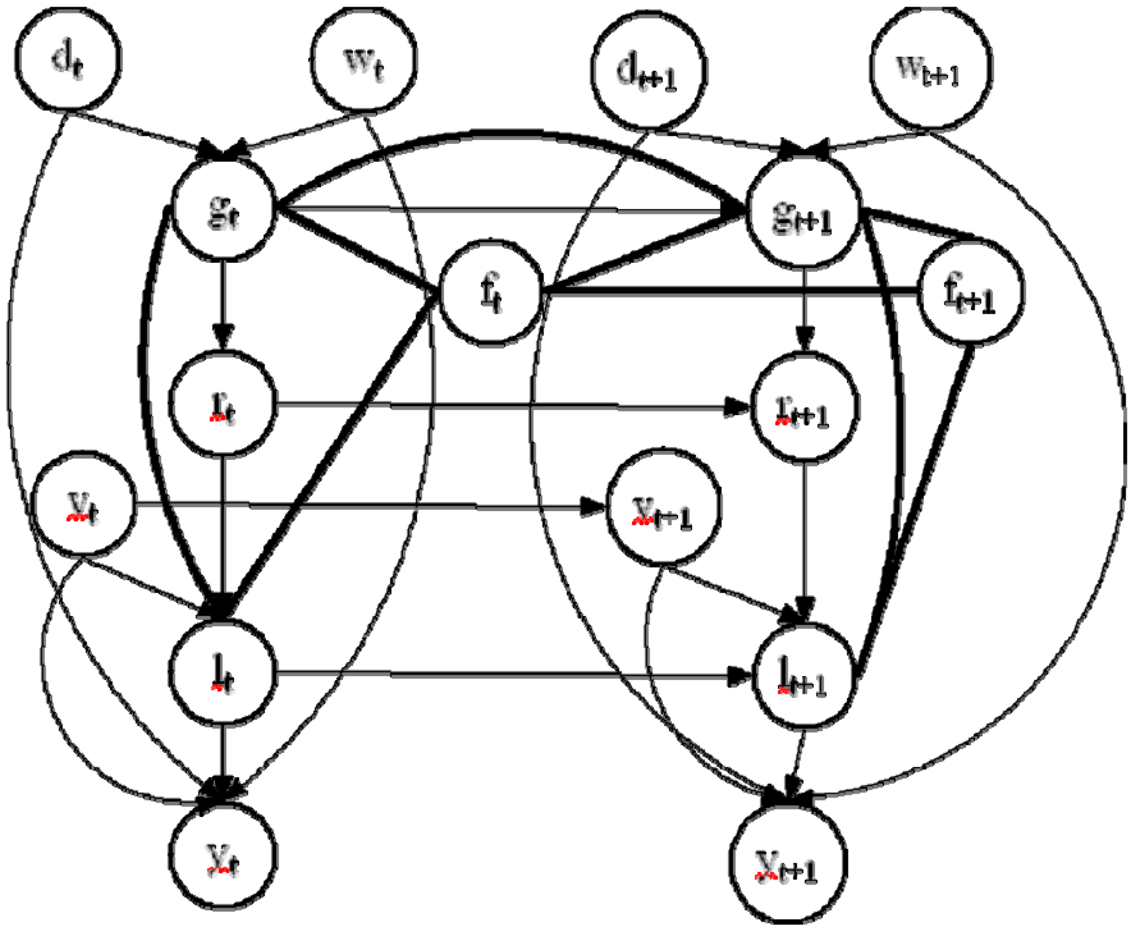

Figure 5: Car Travel Activity model of an individual

Figure 5 shows a HDMN model for modeling the car travel activity of individuals. Note that the directed links express the probabilistic relationships while the undirected (bold) edges express the constraints.

We consider the roads as a Graph $G(V,E)$ where the vertices V correspond to intersections while the edges E correspond to segments of roads between intersections. The variables in the model are as follows. The variables d_t and w_t represent the information about time-of-day and day-of-week respectively. d_t is a discrete variable and has four values *(morning,afternoon,evening,night)* while the variable w_t has two values *(weekend,weekday)*. Variable g_t represents the persons next goal (e.g. his office, home etc). We consider a location where the person spends significant amount of time as a proxy for a goal (Liao et al., 2004). These locations are determined through a preprocessing step by noting the locations in which the dwell-time is greater than a threshold (15 minutes). Once such locations are determined, we cluster those that are in close proximity to simplify the goal set. A goal can be thought of as a set of edges $E_1 \subset E$ in our graph representation. The route level r_t represents the route taken by the person to move from one goal to other. We arbitrarily set the number of values it can take to $|g_t|^2$. The person's location l_t and velocity v_t are estimated from GPS reading y_t. f_t is a counter (essentially goal duration) that governs goal switching. The Location l_t is represented in the form of a two-tuple (a,w) where $a = (s_1,s_2),a \in E$ and $s_1,s_2 \in V$ is an edge of the map $G(V,E)$ and w is a Gaussian whose mean is equal to the distance between the person's current position on a and one of the intersections in a.

The probabilistic dependencies in the model are straightforward and can be found by tracing the arrows (see Figure 3). The constraints in the model are as follows. We assume that a person switches his goal from one time slice to another when he is near a goal or moving away from a goal but not when he is on a goal location. We also allow a forced switch of goals when a specified maximum time that he is supposed to spend at a goal is reached. This is modeled by using a constant D. These assumptions of switching between goals are modeled using the following constraints between the current location, the current goal, the next goal and the switching counters:

If $l_{t-1} = g_{t-1}$ and $F_{t-1} = 0$ Then $F_t = D$,

If $l_{t\,1} = g_{t-1}$ and $F_{t-1} > 0$ Then $F_t = F_{t-1} -1$,

If $l_{t-1} \neq g_{t-1}$ and $F_{t-1} = 0$ Then $F_t = 0$ and

If $l_{t-1} \neq g_{t-1}$ and $F_{t-1} > 0$ Then $F_t = 0$,

If $F_{t-1} > 0$ and $F_t = 0$ Then g_t is given by $P(g_t|g_{t-1})$,

If $F_{t-1} = 0$and $F_t = 0$ Then g_t is same as g_{t-1},

If $F_{t-1} > 0$ and $F_t > 0$ g_t is same as g_{t-1} and

If $F_{t-1} = 0$ and $F_t > 0$ g_t is given by $P(g_t|g_{t-1})$.

4.6 Experimental Results

The test data consists of a log of GPS readings collected by one of the authors. The test data was collected over a six month period at intervals of 1-5 seconds each. The data consist of the current time, date, location and velocity of the person's travel. The location is given as latitude and longitude pairs. The data was first divided into individual routes taken by the person and the HDMN model was learned using the Monte Carlo version of the EM algorithm (Liao et al., 2004; Levine and Casella, 2001).

We used the first three months' data as our training set while the remaining data was used as a test set. TIGER/Line files available from the US Census Bureau formed the graph on which the data was snapped. As specified earlier our aim is two-fold: (a) Finding the destination or goal of a person given the current location and (b) Finding the route taken by the person towards the destination or goal.

To compare our inference and learning algorithms, we use three HDMN models. Model-1 is the model shown in Figure 3. Model-2 is the model given in Figure 3 with the variables w_t and d_t removed from each time-slice. Model-3 is the base-model which tracks the person without any high-level information and is constructed from Figure 3 by removing the variables w_t, d_t, f_t, g_t and r_t from each time-slice.

We used 4 inference algorithms. Since EM-learning uses inference as a sub-step, we have 4 different learning algorithms. We call these algorithms as IJGP-S(1), IJGP-S(2) and IJGP-RBPF(1,1,N) and IJGP-RBPF(1,2,N) respectively. Note that the algorithm IJGP-S(i) (described in Section 3) uses i as the i-bound. IJGP-RBPF(i,w,N) (described in Section 4) uses i as the i-bound for IJGP(i), w as the w-cutset bound and N is the number of particles at each time slice. Three values of N were used: 100, 200 and 500. For EM-learning, N was 500. Experiments were run on a Pentium-4 2.4 GHz machine with 2G of RAM. Note that for Model-1, we only use IJGP-RBPF(1,1) and IJGP(1)-S because the maximum i-bound in this model is bounded by 1 (see section 3).

4.6.1 Finding destination or Goal of a person

The results for goal prediction with various combinations of models, learning and inference algorithms are shown in Tables 1, 2 and 3. We define prediction accuracy as the number of goals predicted correctly. Learning was performed offline. Our slowest learning algorithm based on GBP-RBPF(1,2) used almost *5* days of CPU time for Model-1, and almost *4* days for Model-2—significantly less than the period over which the data was collected. The column 'Time' in Tables 1, 2 and 3 shows the time for inference algorithms in seconds while the other entries indicate the accuracy for each combination of inference and learning algorithms.

In terms of which model yields the best accuracy, we can see that Model-1 achieves the highest prediction accuracy of 84% while Model-2 and Model-3 achieve prediction accuracies of 77% and 68% respectively or lower.

For Model-1, to verify which algorithm yields the best learned model we see that IJGP-RBPF(1,2) and IJGP(2)-S yield an accuracy of 83% and 81% respectively while for Model-2, we see that the average accuracy of IJGP-RBPF(1,2) and IJGP(2)-S was 76% and 75% respectively. From these two results, we can see that IJGP-RBPF(1,2) and IJGP(2)-S are the best performing learning algorithms.

For Model-1 and Model-2, to verify which algorithm yields the best accuracy given a learned model, we see that IJGP(2)-S is the most cost-effective alternative in terms time versus accuracy while IJGP-RBPF yields the best accuracy.

Table 1Goal-prediction: Model-1

			LEARNING			
			IJGP-RBPF		IJGP-S	
N	Inference	Time	(1,1)	(1,2)	(1)	(2)
100	IJGP-RBPF(1,1)	12.3	78	80	79	80
100	IJGP-RBPF(1,2)	15.8	81	84	78	81
200	IJGP-RBPF(1,1)	33.2	80	84	77	82
200	IJGP-RBPF(1,2)	60.3	80	84	76	82
500	IJGP-RBPF(1,1)	123.4	81	84	80	82
500	IJGP-RBPF(1,2)	200.12	84	84	81	82
	IJGP(1)-S	9	79	79	77	79
	IJGP(2)-S	34.3	74	84	78	82
	Average		79.625	82.875	78.25	81.25

Table 2Goal Prediction: Model-2

			LEARNING			
			IJGP-RBPF		IJGP-S	
N	Inference	Time	(1,1)	(1,2)	(1)	(2)
100	IJGP-RBPF(1,1)	8.3	73	73	71	73
100	IJGP-RBPF(1,2)	14.5	76	76	71	75
200	IJGP-RBPF(1,1)	23.4	76	77	71	75
200	IJGP-RBPF(1,2)	31.4	76	77	71	76
500	IJGP-RBPF(1,1)	40.08	76	77	71	76
500	IJGP-RBPF(1,2)	51.87	76	77	71	76
	IJGP(1)-S	6.34	71	73	71	74
	IJGP(2)-S	10.78	76	76	72	76
	Average		75	75.75	71.125	75.125

Table 3Goal Prediction: Model-3

			LEARNING	
N	Inference	Time	IJGP-RBPF(1,1)	IJGP(1)-S
100	IJGP-RBPF(1,1)	2.2	68	61
200	IJGP-RBPF(1,1)	4.7	67	64
500	IJGP-RBPF(1,1)	12.45	68	63
	IJGP(1)-S	1.23	66	62
	Average		67.25	62.5

Table 4 False positives (FP) and False negatives for routes taken by a person (FN)

		Model-1	Model-2	Model-3
N	INFERENCE	FP/FN	FP/FN	FP/FN
	IJGP(1)	33/23	39/34	60/55
	IJGP(2)	31/17	39/33	
100	IJGP-RBPF(1,1)	33/21	39/33	60/54
200	IJGP-RBPF(1,1)	33/21	39/33	58/43
100	IJGP-RBPF(1,2)	32/22	42/33	
200	IJGP-RBPF(1,2)	31/22	38/33	

4.6.2 Finding the Route taken by the person

To see how our models predict a person's route, we use the following method. We first run our inference algorithm on the learned model and predict the route that the person is likely to take. We then superimpose this route on the actual route taken by the person. We then count the number of roads that were not taken by the person but were in the predicted route i.e. the false positives, and also compute the number of roads that were taken by the person but were not in the actual route i.e. the false negatives. The two measures are reported in Table 4 for the best performing learning models in each category: viz GBP-RBPF(1,2) for Model-1 and Model-2 and GBP-RBPF(1,1) for Model-3. As we can see Model-1 and Model-2 have the best route prediction accuracy (given by low false positives and false negatives).

4.7 Related Work

Liao et al. (2004) and Patterson et al. (2003) describe a model based on AHMEM and Hierarchical Markov Models (HMMs) respectively for inferring high-level behavior from GPS-data. Our model goes beyond their model by representing two new variables day-of-week and time-of-day which improves the accuracy in our model by about 6%.

A mixed network framework for representing deterministic and uncertain information was presented in Dechter and Larkin (2001), Larkin and Dechter (2003), and Dechter and Mateescu (2004). These previous works also describe exact inference algorithms for mixed networks with the restriction that all variables should be discrete. Our work goes beyond these previous works in that we describe approximate inference algorithms for the mixed network framework, allow continuous Gaussian nodes with certain restrictions in the mixed network framework and model discrete-time stochastic processes. The approximate inference algorithms called IJGP(i) described in Dechter et al. (2002) handled only discrete variables. In our work, we extend this algorithm to include Gaussian variables and discrete constraints. We also develop a sequential version of this algorithm for dynamic models.

Particle Filtering is a very attractive research area (Doucet et al., 2000). Particle Filtering in HDMNs can be inefficient if non-solutions of constraint portion have high probability of being sampled. We show how to alleviate this difficulty by performing IJGP(i) before sampling. This algorithm IJGP-RBPF yields the best performance in our settings and might prove to be useful in applications in which particle filtering is preferred.

5 Simplifying the Behavioral Model

5.1 Introduction

While the HDMN model presented in the preceding section worked well, it required significant computing power—a dedicated workstation rather than a handheld device. We next attempted to extend this work by simplifying it. This report describes our efforts in this area. The earlier HDMN for modeling travel activities used both discrete and continuous random variables. It is known that both exact and approximate inference are harder on hybrid networks than on networks that have only discrete or only continuous variables [Lerner, 2002]. Therefore, for simplicity and efficiency, we focused on a *discrete* Dynamic Bayesian network model for modeling travel activities. Our aim is to compare this discrete model with the previous hybrid model in terms of accuracy of prediction and time required to answer a query. In the end, we were unsuccessful in our attempt to develop a simpler model. This report describes our progress and outlines work that remains to be done.

The rest of the report is organized as follows. In the next section, we present preliminaries on dynamic probabilistic networks. The probabilistic network used for modeling activities is described in Section 3. We then describe how to learn and answer queries using this model in Section 4. Finally, in Section 5 we describe the issues that need to be resolved before our system could be deployed.

5.2 Preliminaries

Rather than restate what has already been written in this report, the reader is encouraged to review Section 4.2 for formal definitions of Hybrid Bayesian Networks, Constraint Networks, Hybrid Mixed Networks, and Hybrid Dynamic Mixed Networks.

As was stated in Section 4.2, the most common task over Dynamic Probabilistic Networks is filtering and prediction. Filtering is the task of determining the belief state $P(X_t|e_{0:t})$ where X_t is the set of variables at time t and $e_{0:t}$ are the observations accumulated at time-slices 0 to t. Performing filtering in dynamic networks is often called *online inference* or *sequential inference*. We noted that for HDMN models, exact filtering is infeasible, and we must use approximation algorithms, for example a sampling-based algorithm like Particle Filtering. However this inference scheme for HDMNs requires a substantial amount of computing power (or time) for both learning and predicting. In this work, we attempted to develop a simplified version of the proposed HDMN model to speed up both inference and learning steps.

5.3 A simplified Dynamic Bayesian network model for recognizing human activities

The graphical model used is described in Figure 7. Similar to [Liao et al., 2007, Gogate et al., 2005], we assume that the person travels over a Graph $G(\mathbf{V}, \mathbf{E})$ where the vertices $\mathbf{V}$ correspond to intersections or road segments while the edges $\mathbf{E}$ correspond to segments of roads between intersections. The variables in the model are indexed by time t. Variable g_t represents the persons next goal at time t (e.g. his office, home etc). We consider a location where the person spends significant amount of time as a proxy for a goal as in [Liao et al., 2007]. These locations are determined through a preprocessing step by noting the locations in which the dwell-time is greater than a threshold (15 minutes). The route level r_t represents the route taken by the person moving from one goal to the next. We arbitrarily set the number of values it can take to k^2 where k is the number of goals. l_t is the person's current location. The domain of l_t is the set of all road segments within the person's usual travel area (for example, the usual travel area of a person residing in Hollywood is all road segments in Los Angeles and its neighboring counties). gps_t is the current (possibly noisy) evidence about the person's current location. The domain of gps_t equals that of l_t.

As compared to the HDMN model presented earlier, the current model does not employ GPS information about time-of-day and day-of-week. It also makes another simplification: we assume that all random variables have discrete domains. This simplification is useful because it makes the underlying inference task much easier (not only computationally but also in terms of ease of software development). Also inference over discrete BNs does not suffer from floating-point numerical precision issues [Lauritzen and Jensen, 2001] and is therefore more robust.

The input to our model is a sequence of GPS data points available as latitude and longitude pairs. Because the domain of the (evidence) variable gps_t in our model is the set of road segments, we simply replace each continuous GPS point by its closest road segment. This can be achieved as follows. The graph of the world is available in the Tiger/Line ESRI shape file format (published by US Census Bureau in 2002). We first convert TIGER/Line data to the Postgres/Postgis database format. Then, we can take advantage of the specialized GIS tools in Postgres/Postgis to efficiently find the closest location to the input GPS point.

Also, because there are inherent geography constraints in the GPS data, a constraint processing step is necessary to reduce the size of the conditional probability tables and avoid the rejection problem

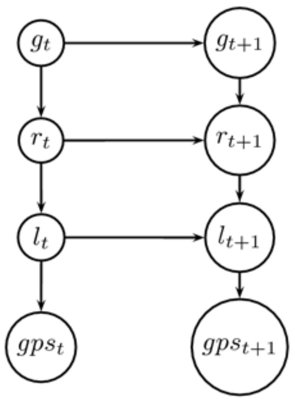

Figure 6 Car travel activity model.

[Gogate et al., 2005] that might occur when performing sampling based inference (described in next section) on the DBN. In this constraint processing step, pair of locations/GPS points that are too far away will be marked as invalid and will receive transition probability of 0.

5.4 Learning

The proposed model was learned using the well known Expectation Maximization (EM) algorithm [Dempster et al., 1977]. As an iterative learning algorithm, EM requires an initial (possibly informed) guess of all the parameters (or CPTs) of our DBN model. In section 3 we documented our proposed heuristic scheme to make a good initial guess. However, for simplicity, in this work we initialize each conditional distribution $P(G_{t-1}|G_t)$, $P(R_t|G_t,R_{t-1})$, $P(L_t|R_t,L_{t-1})$, $P(GPS_t|L_t)$ to the uniform distribution.

5.4.1 Inference: Particle Filtering

Similar to what was reported earlier for the HDMN model, we use Particle Filtering for performing inference within the EM framework. Particle filtering, which is an approximate inference algorithm, is used instead of an exact algorithm due to the constraints we have on time and computer resources. Particle filtering consists of two steps: forward and backward as outlined in Algorithms 1 and 2 respectively.

In the algorithm for forward pass, we need to choose a proposal distribution from which samples are drawn. The choice of the proposal distribution can have a big impact on the accuracy of the algorithm [Rubinstein, 1981, Doucet et al., 2000]. In this work we just use the prior distribution as the proposal distribution. Therefore, one immediate improvement that can be made is to replace the prior distribution with a more accurate proposal distribution. Two different schemes for choosing a good proposal distribution were discussed in Section 3.

```
**Input:** A Dynamic Bayesian network
$p(\mathbf{X}_{0:t}) = p(\mathbf{X}_0) * \prod_{t=1}^{T} p(X_t|X_{t-1})$, a proposal distribution
$\pi(\mathbf{X}_{0:t}) = \pi(\mathbf{X}_0) * \prod_{t=1}^{T} \pi(X_t|X_{t-1})$ and an integer $N$.
**Output:** A set of $N$ particles for each time slice and their weights.
For $k = 1, \ldots T$; perform the following steps.
  1) For $L = 1, \ldots, N$ draw samples from the "proposal distribution"
$x_k^{(L)} \sim \pi(x_k|x_{0:k-1}^{(L)}, y_{0:k})$
  2) For $L = 1, \ldots, N$ update the importance weights up to a
normalizing constant:
$\hat{w}_k^{(L)} = w_{k-1}^{(L)} \frac{p(y_k|x_k^{(L)})p(x_k^{(L)}|x_{k-1}^{(L)})}{\pi(x_k^{(L)}|x_{0:k-1}^{(L)}, y_{0:k})}$. Note that this simplifies to the following
: $\hat{w}_k^{(L)} = w_{k-1}^{(L)} p(y_k|x_k^{(L)})$, when we use :
$\pi(x_k^{(L)}|x_{0:k-1}^{(L)}, y_{0:k}) = p(x_k^{(L)}|x_{k-1}^{(L)})$.
  3) For $L = 1, \ldots, N$ compute the normalized importance weights:
$w_k^{(L)} = \frac{\hat{w}_k^{(L)}}{\sum_{J=1}^{N} \hat{w}_k^{(J)}}$
  4) Compute an estimate of the effective number of particles as
$\hat{N}_{eff} = \frac{1}{\sum_{L=1}^{N} \left(w_k^{(L)}\right)^2}$
  5) If the effective number of particles is less than a given threshold
$\hat{N}_{eff} < N_{thr}$ perform resampling:

    1. Draw $N$ particles from the current particle set with probabilities
       proportional to their weights. Replace the current particle set with this
       new one.

    2.   For $L = 1, \ldots, N$ set $w_k^{(L)} = 1/N$.
```

Algorithm 1 Particle Filtering: Forward Pass

```
**Input:** A Dynamic Bayesian network
$p(\mathbf{X}_{0:t}) = p(\mathbf{X}_0) * \prod_{t=1}^{T} p(X_t|X_{t-1})$, a proposal distribution
$\pi(\mathbf{X}_{0:t}) = \pi(\mathbf{X}_0) * \prod_{t=1}^{T} \pi(X_t|X_{t-1})$ and an integer $N$.
**Output:** A set of $N$ particles for each time slice and their weights.
1) **for** $t$=0,1,...,$T$ **do**
     perform forward pass to obtain weighted measure $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^{N}$;
**end**
2) **for** $i$=0,1,...,$T$ **do**
     set $w_{T|T}^{(i)} = w_T^{(i)}$;
**end**
3) **for** $t$=$T$-1,...,1 **do**
     **for** $i$=1,...,$N$ **do**
         $w_{t|T}^{(i)} = w_t^{(i)} \sum_{j=1}^{N} w_{t+1|T}^{(i)} \frac{p(x_{t+1}^{(j)}|x_t^{(i)})}{\sum_{k=1}^{N} w_t^{(k)} p(x_{t+1}^{(j)}|x_t^{(k)})}$
     **end**
**end**
```

Algorithm 2 Particle Filtering: Backward Pass

The literature describes two alternatives for the backward pass. One involves keeping all samples from the forward pass in memory while the second involves generating new samples in backward direction.

Each alternative has its own advantages and disadvantages. The scheme that utilizes the samples generated during the forward pass was found to have larger error in previous studies but it is much simpler to implement. The second scheme [Klaas et al., 2006] is more accurate but for it to work well, we need to choose another proposal distribution to sample backward. Picking a good proposal distribution in the backward pass is quite tricky and will be addressed in the future. In our implementation, we use the first option.

5.5 Current Issues

At this point, there are still some issues that need be resolved before our software could be deployed. Most importantly, the estimation algorithm does not converge. The standard convergence criterion of the EM algorithm expects that the likelihood of evidence given the model will generally increase after each iteration. While we were unable to solve this problem before the project ended, we believe that this convergence criterion is not being met because of the approximate nature of our computations. Namely, the likelihood computed using our approximate scheme may not increase monotonically as required by EM. It may be that the Monte Carlo version of the EM algorithm [Levine and Casella, 2001] might be useful in addressing this issue.

Second, our set up of EM learning has the following flaw. We consider each continuous segment of observed locations as a distinct route and the two end points of the route as origin and destination. This simplifies the preprocessing step but the model learns a lot of spurious goal locations and routes between them. For example:

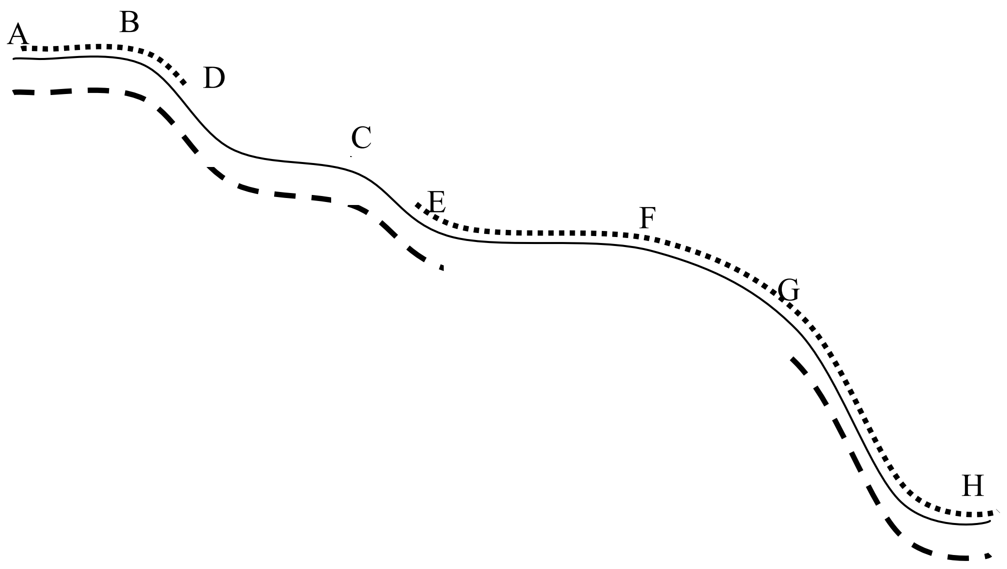

Figure 7**:** Two trip segments between goals A and H, one dotted, one dashed. The "actual" route is the solid line

Considering Figure , suppose there are two trips between A and H. Assume that in both cases the GPS signal was lost in between as shown. Based on our criteria for designating goal locations as a location where a person spends more than ten minutes, we would learn multiple goal locations (A, D, E, G and H instead of two. With more trips, and more erratic GPS readings, one can imagine that the number of false destinations can increase dramatically.

One possible solution to the problem described above is to employ the following data cleaning step. We train the model only on “valid” training examples that obey the following constraint: Let us assume that the GPS signal was lost at time *t* and at location *l*. If the GPS signal is resumed at time “> (*t* + *d*)” at location “*l* ± *k*”, then it is a valid training example, where we decide the parameters *d* and *k* in advance. With this data cleaning step, the probability of learning routes over spurious goals will be reduced substantially.

6 Automatically Collecting Travel and Activity Data

6.1 Introduction

The Behavioral Model and other implicit query techniques require the ability to automatically collect and process travel data. This chapter documents our research into data collection protocols for gathering activities of individuals and groups. In contrast with our modeling work, this chapter focuses primarily on data collection methods, specifically with automatically extracting travel routes and activity locations from group travel data, and identifying common characteristics in travel and activity patterns. We describe a data reduction process that takes raw GPS data and converts it into sequences of temporal behavior projected onto a graph derived from commonly available map data.

We hypothesize that individual and group behavior is sufficiently routinized such that common activity locations and travel routes will form a small countable set that can be enumerated for modeling purposes. We evaluate this hypothesis by analyzing five months of GPS data collected for a single household and find some evidence to support the data reducibility hypothesis. Activity-based analysis has traditionally been held up as the future of urban planning and travel analysis (Kitamura, 1988; Pas, 1996). As the four-step planning model was increasingly shown to be ineffective, activity modeling held out the promise of providing a more reasoned approach to modeling the need for travel. While theoretical and practical advances have been made, a difficult barrier to overcome has been the inability to collect enough data. People do not want to fill out travel diaries, and even the most well-intentioned respondent is going to give up on even the most advanced computerized survey after a handful of days.

One approach to collecting activity and travel data is to rely almost entirely on electronic data collection with no user intervention necessary. This capability is easily programmed into a modern cellphone such as Apple’s iPhone or a phone based on Google’s Android, both of which allow programmatic access to GPS reading, compass direction, data storage, and wireless communications. Such a device would constantly be able to track a traveler, store a record of his or her travel patterns, and determine what activity resources and opportunities existed along the route of travel and at any possible destination.

Given this, the problem we address in this chapter is that of reducing time-space data into a condensed, but still information-rich format for activity-based analysis. The process is shown in Figure 8 as three main tasks: sequence partitioning, map matching, and clustering to reduce the matched paths into locations and unique travel routes. Next we discuss the data collection problem and the system we used in this research. This is followed by an discussion of the data reduction techniques we employ, an analysis of an longitudinal data set to illustrate the our methods, and finish with some thoughts regarding potential applications of this type of data reduction.

6.2 Data collection methods

Regardless of the application, we assume a data collection process that collects individual time-space trajectories using some form of GPS. Ideally, the GPS device is hand held, but mode-specific devices can also be used (such as vehicle based devices) if individual users can be associated with the vehicles. The GPS data can be manually downloaded from the device (by reading from a removable card) or it can be received via some form of telemetry.

The data collection system we employed in this research originated with the ATMS Testbed Tracer system (Marca et al., 2003). As such, the basic pieces of the tool chain have been in operation for several years and have undergone at least two major rewrites. Tracer has been fully documented elsewhere, and so this section merely provides an overview of the components of the system, with an emphasis on recent additions. The interested reader is directed to the project website that is a part of the ATMS Testbed website.

The idea behind Tracer is to track travel and activity behavior as unobtrusively as possible. The original system was made up of an in-vehicle data collection unit that wirelessly communicated with a server to save GPS points while the vehicle was in motion. This data was used to generate an activity survey (Marca, 2002). The wireless component of the original Tracer system stopped working when CDPD technology was abandoned by all wireless carriers as obsolete. As a result, data must be read from the Tracer data collection units using a standard CompactFlash card reader. Because the Tracer tool chain is generally neutral with regard to the source of the GPS data it can therefore accept data that is manually downloaded from any GPS device. For instance, we have augmented our original vehicle-

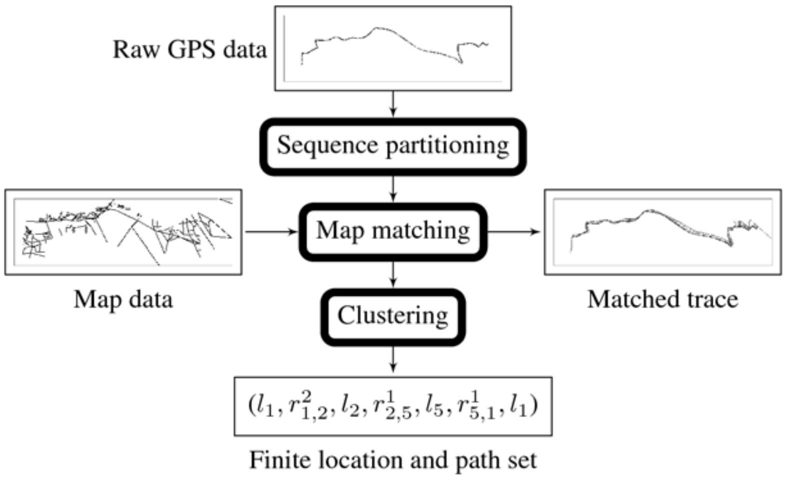
Raw GPS data
Sequence partitioning
Map data
Map matching
Matched trace
Clustering
$(l_1, r^2_{1,2}, l_2, r^1_{2,5}, l_5, r^1_{5,1}, l_1)$
Finite location and path set

Figure 8 The map matching problem

based devices with a set of Garmin eTrex units. GPS data is read from these units as GPX documents (Foster, 2004) via the GPSMan program (Filgueiras, 2007), and speed (which is not saved by the eTrex units) is interpolated using GPS Babel (Lipe, 2007).

Tracer stores the GPS data into a PostgreSQL database that contains the PostGIS (Ramsey et al., 2007) spatial modeling library routines. The PostGIS library is primarily used to enable fast spatial searches, but it also has a number of useful routines for manipulating points, lines, and areas, interpolating points, and so on. The database has been designed to link travelers with their devices and data, and to allow researchers to create accounts for travelers and hand out devices. Raw GPS points are not in themselves very useful for modeling the travel behavior as it relates to the built environment. This fact has motivated the development of a data reduction system that produces data consistent with activity-based analysis problems.

6.3 Identifying activity locations

Minimally, the raw GPS points must be combined into temporally continuous and spatially contiguous sequences that approximate actual activity locations and the trips that connect them. This task might appear straightforward, but the devil is in the details. For example, using simple rules such as defining an activity as any time the GPS sequence is stationary for more than 5 minutes will miss short activities such as a stop at a gasoline station, or picking up or dropping off a passenger. With the original Tracer system, the GPS units were continuously powered when in use because they were vehicle-based. Furthermore, they had high quality antennas that very quickly acquired an accurate position fix. The additional hand-help units used in the current research are of lower quality and are necessarily battery-powered. This exacerbates the problems associated with identifying origins, destinations, and routes, but more accurately reflects the characteristics of a GPS chip embedded in a cellphone buried in a pocket or a handbag.

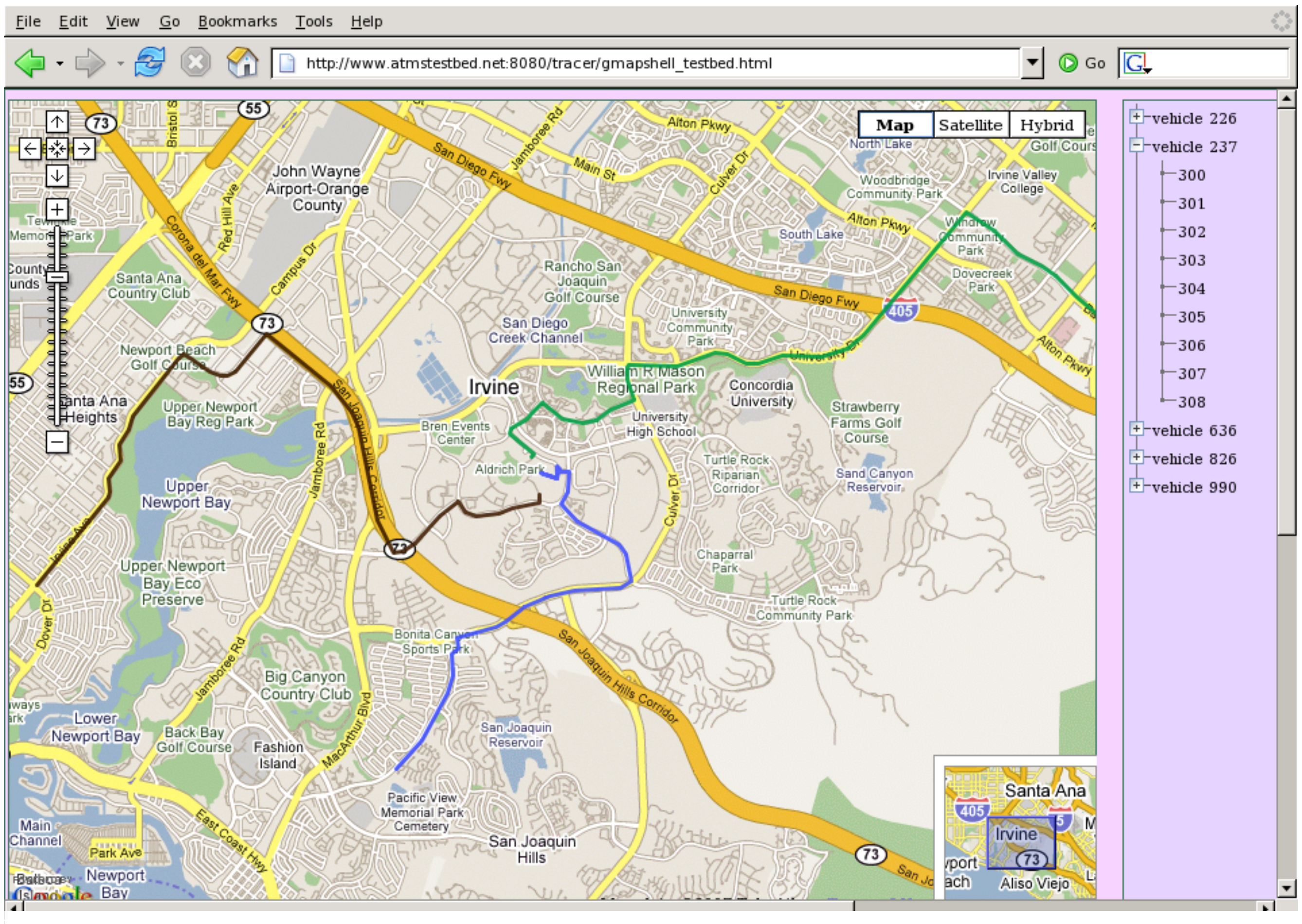

Figure 9 Actual trips displayed on the Tracer site via Google Maps API. Note that the GPS points have not been matched to streets, but still accurately indicate visually the route a traveler drove.

While in the past we have relied on the accuracy of the GPS antenna to perform tricky calculations such as detecting possible activity points when the vehicle backtracked or performed some sort of U-turn over a short distance, the hand-held GPS units generate more spurious apparent movement due to their weaker antennas. Therefore the basic rule to detect a continuing trip was to look in the database for a current sequence of points (called a trace) with a nearby end point (using the PostGIS routines), given that the earlier recorded point is not separated by the current point by more than 6 minutes—arbitrarily chosen to weed out spurious stops at the expense of missing some short activities. If such an end point to an existing sequence or trace is not found, then a new trace is started. Collecting points into sequences that approximate trips is useful for generating maps. For example, one can generate a map of a GPS points using the Google Maps public API, as shown in Figure 9. The polylines are generated using a smoothing technique on the raw GPS points, but otherwise the points have not been adjusted in any way to fit the streets on the map. Rather, the underlying map data is so good that the GPS points show up in exactly the right place for an entire route. Similar accuracy can be obtained in some parts of the world using the free OpenStreetMap mapping site (Coast, 2009).

6.4 Map matching

A map such as that drawn in Figure 9 is a picture, and no matter how informative it is to a reader, it is not very helpful to a program. Our goal is not to ask the traveler about origins, destinations, routes, and activities, but rather to deduce as much information as possible with no assistance from the traveler. Therefore the next step in the data processing chain is to associate the GPS data with the underlying street network.

The availability of quality street network data varies depending on location and budget. Commercial map providers offer the high-quality map data that drives the best Internet mapping services, but the licensing and cost of this data makes it's use problematic in some instances. In the United States, the Census has made available for free the TIGER/Line data set, a reasonably accurate digital map of nearly every street in every US state and territory which we are using as our base network. (This is not the case in other countries, although the OpenStreetMap project is progressing very rapidly in many European cities thanks to hundreds of citizen-mappers.) For this reason, we have used TIGER/Line data in this research, but any available street network data that can be converted into a mathematical graph can be used. Since work on this project ended, the OpenStreetMap project has completed the import of TIGER data, and through the power of many eyeballs, is actively hand editing *every* street in the United States. Future work on related projects will use the OpenStreetMap data.

The map matching problem has seen recent interest in the literature with the increasing prevalence of vehicle navigation systems. Solutions include a variety geometric, topological, probabilistic, and advanced hybrid methods (see Quddusa et al., 2007, for a detailed review). In this research, our primary interest is the development of a consistent map matching system that will produce identical mappings for identical inputs and will typically generate the same route for data from trips traversing the same portions of the network. Consistency is more important than the integrity of the algorithm if we are to compare routes used during repeated trips because it is more important in our application to identify the fact that the same route is used multiple times than it is to know exactly what that route is with a high level of confidence.

We also require an algorithm that can handle relatively large errors in map data because of our current reliance on public map data sources that tend to have larger errors. For this reason we developed a custom, deterministic algorithm instead of applying probabilistic methods such as particle filtering because consistent, repeatable results are required for the purposes of identifying regular routes.

6.4.1 A filtered, topological matching algorithm using A*

Map matching in our system is a two step process. First, to reduce the size of the search space, a spatial GIS is used to compute candidate mappings based upon a given error bound that must be chosen to account for both GPS positioning errors and errors in the reference map. This step provides a first cut at matching, but is far from acceptable due to the rather poor accuracy of the TIGER/Line data. This data is combined into a candidate network graph (consisting of only relevant portions of the map) and the GPS trace data with each GPS trace point associated with a set of feasible map edges. This input data is then passed to a map matching routine that uses the A* algorithm (Russell and Norvig, 2003) to find a topological match between a GPS trace and candidate map. The output of the algorithm is mapping between each point in the trace and a single edge in the directed graph representing the map network. This is an offline method that searches the set of possible point-to-edge mappings that produces the best total trace to route mapping. To keep the algorithm simple and the data requirements low, only position data is used. Heading, speed, and fix information available from the GPS is ignored, none of

```
Input: trace: t
Output: map graph: G
var closed := { }
var q := new priority_queue()
foreach y in successors( {} ) do
    var x' := append_point_edge_mapping( { }, y )
    add_to_queue_with_score( q, x', f( x' ) )
end
while q is not empty do
    var x := remove_first( q )
    var pem := last_point_edge_mapping( x )
    if pem in closed then
        continue
    end
    if get_point( pem ) = last_point( t ) then
        return x
    end
    add( closed, pem )
    foreach y in successors( x ) do
        var x' := append_point_edge_mapping( x, y )
        add_to_queue_with_score( q, x', f( x' ) )
    end
end
return no_feasible_match
```

Figure 10 The core A* algorithm

which are directly recorded by low-end handheld GPS units like the Garmin eTrex.

The core algorithm used is shown in Figure 11. The algorithm starts by creating a set to store closed point-edge mappings and priority queue for storing open partial solutions. It then adds a scored partial solution for each feasible point edge mapping for the first trace point, as determined by a user-defined successors function. In our implementation, the successors function returns all point-edge mappings for the next trace point. At this point the algorithm loops by continuing to remove the highest scored partial solution from the priority queue until no open partial solutions remain. The last point edge mapping of each partial solution is checked to see if it is in the closed set. If so, then the algorithm has already reached this point-edge mapping with a lower score meaning that this partial solution can be skipped. If this is not the case, the last point edge mapping is checked to see if it is a mapping for the last trace point. If it is, then this partial solution is the best complete map matching possible for the given scoring function and the algorithm terminates returning the solution. Otherwise, the last point edge mapping is added to the closed set and a set of new partial solutions is added to the priority queue by appending each successor point-edge mapping for the next trace point to the current partial solution. The algorithm continues this process until a solution is found or it is determined that no solution exists.

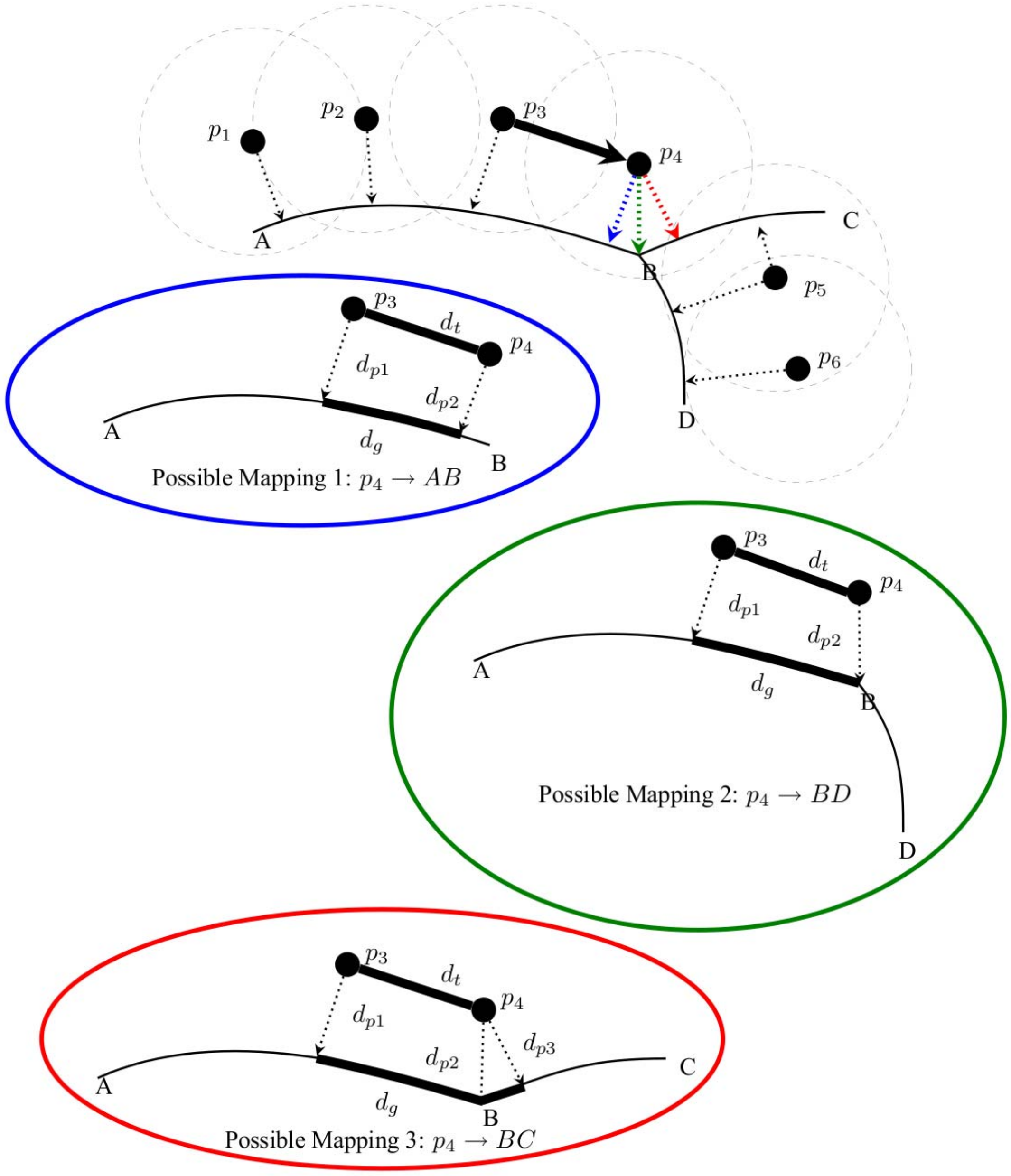
p_1
p_2
p_3
p_4
p_5
p_6
A
B
C
D
d_t
d_{p1}
d_{p2}
d_{p3}
d_g
Possible Mapping 1: $p_4 \rightarrow AB$
Possible Mapping 2: $p_4 \rightarrow BD$
Possible Mapping 3: $p_4 \rightarrow BC$

Figure 11 An illustration of the map matching error function. A six point GPS trace p_1 to p_6) is to be matched to the map graph G=(V={A,B,C,D},E={AB,BC,BD}). The dashed circles around each GPS point illustrate the maximum point error bound. The dotted arrowed lines show feasible point to edge mappings based upon this error bound. The three larger ellipses show the relevant values for computing the scoring function of three possible mappings for p_4, given a partial solution that has mapped p_3 to edge AB

To illustrate the algorithm and explain the scoring function, consider the map matching problem shown in Figure 11 in which a six point GPS trace is being matched to a three edge map graph *G =(V = {A, B, C, D}, E = {AB, BC, BD})* with a given error bound. As noted, every candidate solution x (a list of GPS point to edge mappings) is evaluated using a scoring function, which is defined as: *f (x) = g(x) + h(x)* where *g(x)* is the known error of the partial solution and *h(x)* is an estimate of the lower error bound for reaching a final solution from *x*. We define the components *g(x)* and *x(x)* of the scoring function with two objectives in mind for the overall algorithm. The first is to minimize the area between the GPS trace and the matched route. This is a form of curve-to-curve matching that is known to have some problems—particularly in instances when a trace point is closer to an orthogonal map edge than the true (parallel) map edge. Consequently, the second objective seeks to minimize the difference between the total length of the GPS trace and the length of the matched route, effectively penalizing any side tracks that don't increase the area between the curves but do significantly increase the length of the matched route in a manner inconsistent with the observations. These two objectives can be represented in a single error function that is based upon computing the error of two adjacent point edge mappings: the prior point edge mapping and the candidate point edge mapping. Note that the local error is independent of any other mappings in a candidate solution. For a given point edge mapping pair, we define the match error as:

$$\epsilon(i) = \left(1 + |d_t(i) - d_g(i)|\right)\left(1 + \frac{\sum_i^N d_p}{N}\right)$$

where:

$d_t(i)$ as the distance from the last trace point $i - 1$ to the current trace point i

$d_g(i)$ as the projected distance along edges in the graph induced by the mapping

N as the number of projection distance samples taken

d_{pi} is the length of the orthogonal projection ith sample point

In this function, the first term represents the difference in the GPS trace length and the second term approximates the area between the trace and the matched route using the average distance between them. To compute this approximation, samples are taken at the two point edge mappings and at any map nodes that are traversed in the graph. Thus, in possible mapping 3 in figure 5.4, a sample is taken between p_4 and node *B* in addition to the end points of the mapping.

Assuming that the partial match x includes points *1* through *j* from the trace, then *g(x)* is the sum of all match errors up to the *j* th point in the GPS trace.

$$g(x) = \sum_{i=1}^{j} \epsilon(i)$$

The heuristic *h(x)* must compute a lower bound on the remaining error for the best complete match that includes match *x*.

$$h(x) = \sum_{i=j}^{T} \min(\epsilon(i)).$$

We compute the minimum possible mapping error for each point i as the minimum possible error of all point edge mapping pairs in which point i is the latter point in the mapping. Referring again to figure 5.4, *min*$((p_5))$ would be the minimum value of computed for possible match pairs: $\{(p_4\ AB, p_5\ BD),(p_4\ AB, p_5\ BC),(p_4\ BC, p_5\ BC),(p_4\ BC, p_5\ BD),(p_4\ BD, p_5\ BC),(p_4\ BD, p_5\ BD)\}$. This heuristic requires that all point edge mapping pair errors be computed prior to performing the search. This makes the algorithm's complexity sensitive to the error bound, which determines the number of candidate matches per point. That said, the number of computations required to determine the bound for all possible trace to map mappings is $O(T M^2)$ where T is the length of the GPS trace and M is the maximum number of edges associated with any given point (recall that the determination of point to edge mappings is handled by the indexed spatial database). Since edge density is physically limited in real world street networks, reasonable error bounds should result in relatively a small M .

6.4.2 Implementation and Performance

We built a Java implementation of the A* algorithm described above using mostly custom algorithms but also leveraging some generally available libraries for graph theoretical (Naveh, 2006) and geographic computational problems (Dautelle, 2007; BBN Technologies, 2006). Java was chosen for convenience despite the likely performance loss compared to a language that compiles to machine code directly. The implementation has proven to be a robust map matcher that can correct for severe error between the GPS data and the network graph regardless of the source of the error. Figure 12 shows a typical map matching solution where a GPS trace (blue) is mapped to the network graph (black) to produce a solution (red). In this case, there are notable distortions in the TIGER/Line data that the algorithm overcomes.

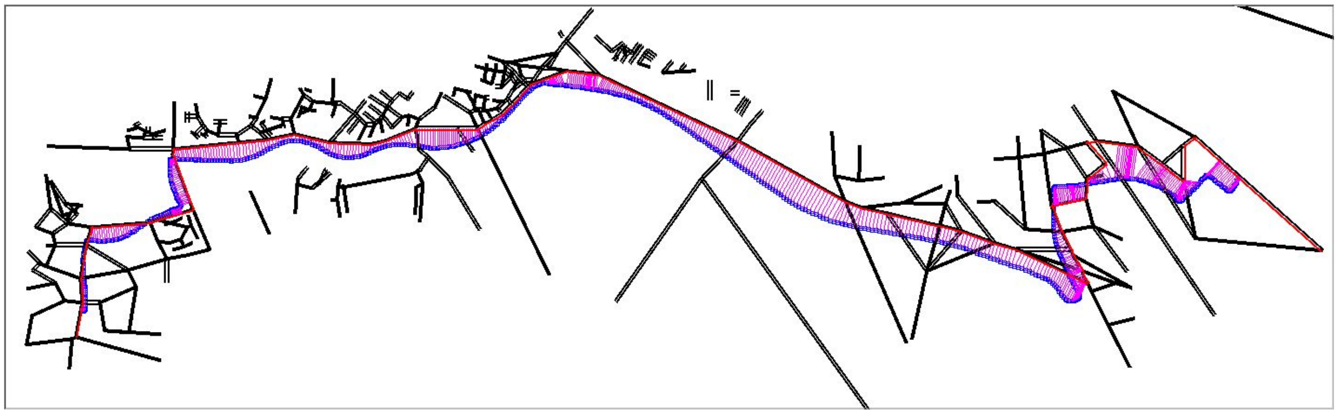

Figure 12 A typical map matching solution using the A* algorithm. GPS data is shown in blue. The part of the map graph selected by the spatial database is in black. The map matching solution is drawn in red.

The algorithm is fairly robust in the face of gaps in the GPS data. If the possible edge mappings for two adjacent GPS trace points are not adjacent in the graph, the algorithm uses the shortest path between the mapped edges to connect the mappings and compute the error function. This approach works as long as the graph provided by the spatial GIS connects the edges in question. The longest traces matched in this research contained on the order of 1000 GPS points representing trips of approximately 60 minutes through dense urban areas. The algorithm typically took 5-10 minutes to solve these problems on an Intel Core2 Duo 6600 running at 2.4GHz with 2GB of RAM, though specific performance also depends on the characteristics of the underlying map graph. The mean running time for all jobs was 50s. This is clearly slower than the best available algorithms, but is sufficient for the task at hand. A secondary problem with the algorithm is its relatively high memory requirement—a known problem with A* algorithms. The memory issue, however, did not prevent any of the map matching jobs from completing.

6.5 Condensed time-space-activity trajectories

After the trip has been matched to a sequence of streets, the original GPS data have been physically reduced often by more than an order of magnitude, while still retaining much of the sense of the original point data. The points can paint an accurate picture, while the street names paint a logical picture of a person's route. That logical picture is much easier to store and process, and provides a way to start comparing origins, destinations, and routes across different individuals.

If individuals use routines in their daily behavior it follows that a sufficiently sized longitudinal sample of that individual's behavior should capture the majority of activity-location pairs that that individual is likely to visit. Note that we can treat travel as a distinct activity that is carried out at any number activity "locations," each of which represents a unique travel route. The degree to which travel choices are disaggregated can vary depending on modeling needs. For instance, we could model a multi-modal trip by treating it as a single "location" for a single travel activity, or we could treat it as a series of travel activities, each with their own "location" representing the mode-specific routing choices (e.g, the path taken through the road network or the specific set of trains taken to connect particular transit hubs). We believe this is a novel approach to reducing individual trips into a manageable set for analysis.

If these assumptions are true, an individual's routine behavior—including all travel choices—could potentially be represented by a finite, countable set of activity-location pairs that is conducive to modeling in a variety of application contexts. To evaluate this a possibility, we looked at the longitudinal behavior of a single vehicle in one household collected using the Tracer system for a period of approximately five months. The subset of all trips that were made in Orange County, California are plotted in Figure 13. The plot shows diverse, but apparently routine travel patterns. The map matching algorithm was able to find acceptable matches for 242 of the 248 traces recorded by the system (97.5%). Because we have no ground truthing information regarding the actual trips taken, we can make no definitive statement regarding the integrity of the matches. A manual sampling of the results, however, indicate they are generally accurate with some incorrect route diversions caused by significant errors in the TIGER/Line data. In these cases, however, data from repeated trips produces the same matched routes, thus satisfying the consistency requirement.

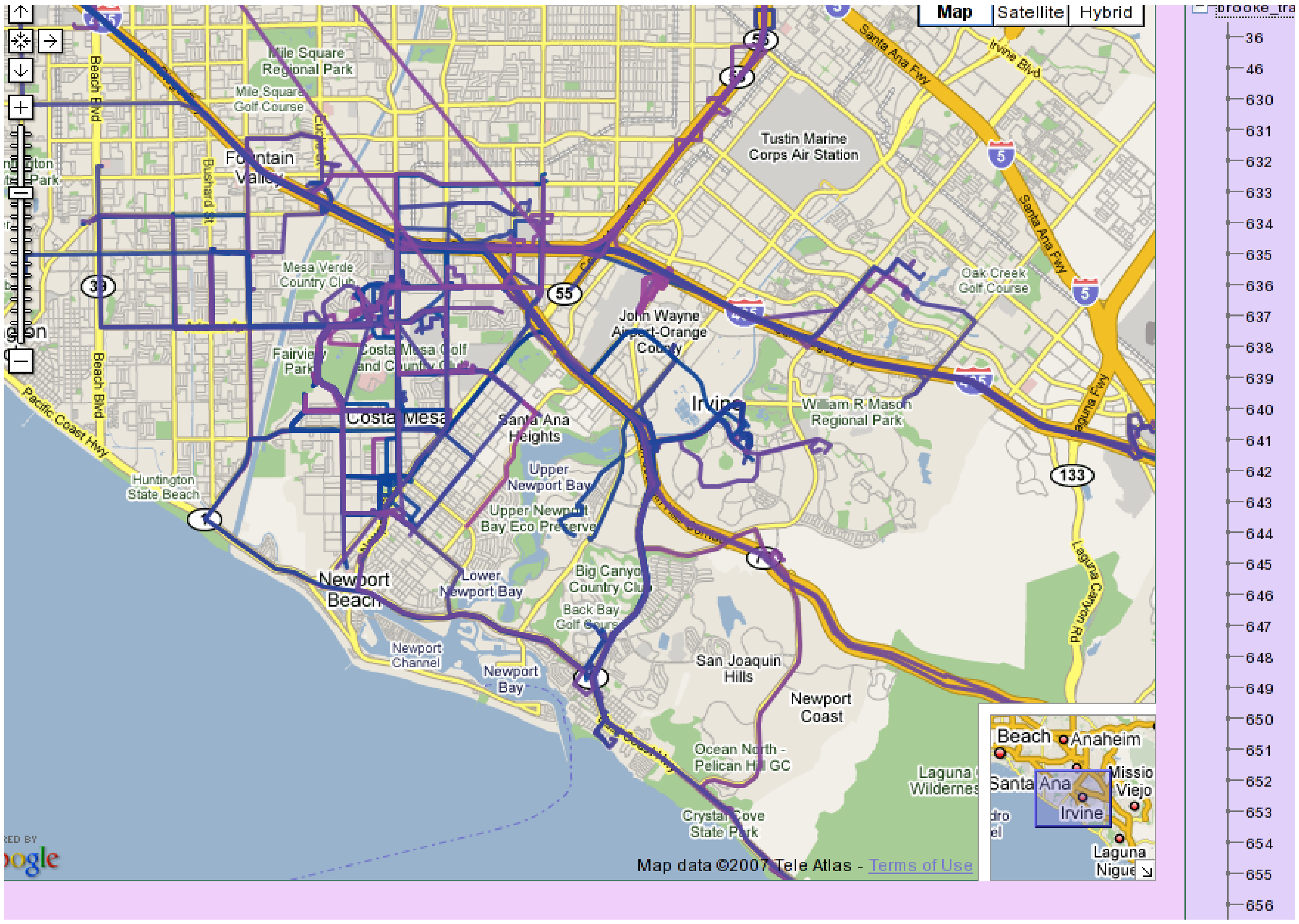

Figure 13 Five months of trip data for one vehicle in a household shown on the Tracer website using Google maps

Because the data is limited to the single vehicle, the dataset only allows inferences to be drawn as to the frequency of locations accessed by the vehicle and the particular routes used. Using the results of the map matching, we defined origin and destination by street name. In the 289 observed trips there are:

- 69 unique origin streets,
- 65 unique destination streets,
- 192 unique OD pairs, and
- 226 unique routes

Note that the matrix of origins crossed with destinations is mostly empty—out of 4,485 possible cells, only 192 contain observed trips. However, the number of unique routes seems high, and contradicts our hypothesis that routine behavior will permit significant reduction of the individual's behavior. Two problems are at work here. First, a more careful analysis demonstrates that some of the variability can be explained away by noting the limitations of the GPS data collection unit—in particular, the nontrivial time required to obtain an initial fix to start recording positions. This makes determination of origins problematic as the first point in a trace will vary greatly depending on the time to fix and the exact route taken. Second, the unique origins and destinations here are defined in terms of matched street names. This will nearly always be true to some extent with a vehicle-based system such as Tracer. A cellphone or handheld GPS device will not be restricted to streets and parking lots, but may have other problems related to determining a trip destination. For many of the trips (and especially for residential origins and destinations), multiple streets can serve as the final stopping point for the vehicle. Parking on one street or another will change the final destination, when in fact the traveler will consider the true destination to be the same.

In general, looking at destinations is more informative since the GPS unit will typically have a quality fix by the end of a trip and will not shut down until the vehicle itself is stopped. The most frequent destinations are shown in table 5.1 (destinations visited fewer than 5 times are not shown). Here we see that 2 out of 5 trips (41%) are made to one of four destinations. Even so, the analysis appears to be misleading in that many destination streets are adjacent to each other, implying that these destinations should be combined.

Table 1: Destination frequency for individual longitudinal data

Destination	Frequency	Unique routes	Observations of most frequent route
1	36 (12%)	13	18 (50%)
2	34 (12%)	26	4 (12%)
3	27 (9%)	21	5 (18%)
4	23 (8%)	20	3 (13%)
5	10 (3%)	6	5 (50%)
6	8 (3%)	7	2 (25%)
7	7 (2%)	7	1 (14%)
8	7 (2%)	7	1 (14%)
9	6 (2%)	5	3 (43%)
10	6 (2%)	5	2 (33%)
11	6 (2%)	5	2 (33%)
12	6 (2%)	3	4 (67%)
13	5 (2%)	3	4 (67%)
14	5 (2%)	5	1 (20%)
15	5 (2%)	4	2 (40%)

To address this, we next analyzed the destinations spatially, clustering the final end points of the GPS trip data. The clustering rules were to create new clusters wherever two points were less than 50 meters apart, and to add to existing clusters all points that were within 10 meters of the cluster. The top ten clusters are shown in table 5.2, in which over one-third (36%) of all trip ends go to the largest cluster. Even with clustering there still appears to be more variability that we would have expected than expected under the routine behavior hypothesis. However, if one assumes that the most frequent destination is a home location, the variability of routes to that destination makes more sense.

Table 2: Destination cluster frequency for individual longitudinal data. Only the top 10 clusters are shown

Dest Cluster	Frequency	Unique routes	Observations of most frequent route
1	103 (36%)	85	5 (5%)
2	37 (13%)	14	18 (49%)
3	11 (4%)	9	2 (18%)
4	8 (3%)	7	2 (25%)
5	6 (2%)	6	1 (17%)
6	6 (2%)	5	2 (33%)
7	6 (2%)	4	3 (50%)
8	4 (1%)	4	1 (25%)
9	4 (1%)	4	1 (25%)
10	4 (1%)	3	2 (50%)

The clusters are not perfect, and tend to overstate the size of a destination. As an example, consider the three clusters shown in figure 14. The middle destination cluster stretches out into the nearby streets, while the bottom cluster covers only the parking lot. It is likely, given the street layout, that the street parking associated with the library destination should instead be matched to the church parking lot destination. Perhaps the clustering algorithm could be improved by including the time of day and day of week first, and then considering distance. We have not settled on a suitable algorithm for clustering origins, due to the variability of when the GPS recorder gets it first position fix.

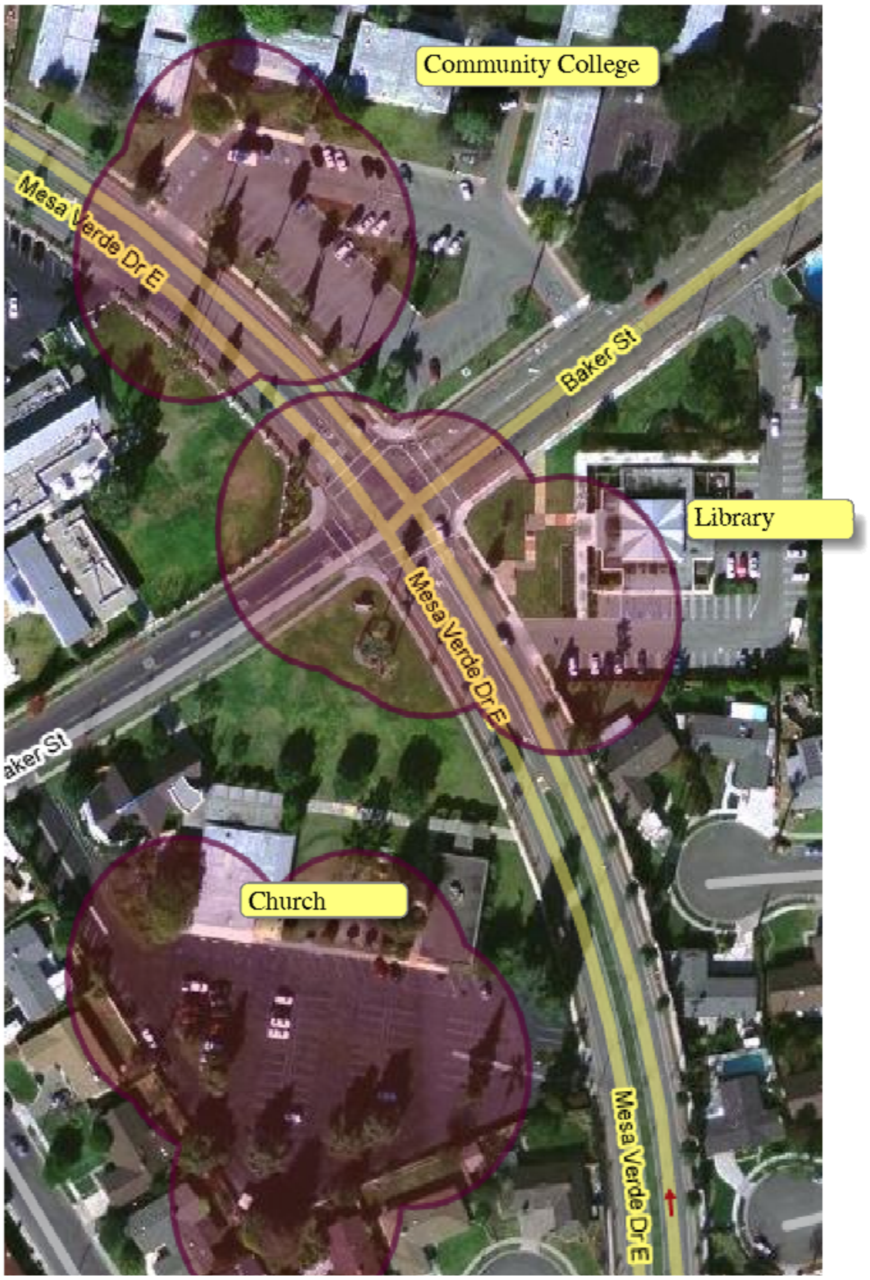

Figure 14 Three clusters for three nearby destinations. The middle cluster probably includes street parking that should be associated with the lower cluster.

7 References

BBN Technologies (2006). Openmap 4.6.3. A programmer's toolkit for building applications and applets that access geospatial data from legacy databases and applications.

Bidyuk, B. and Dechter, R. (2004). On finding minimal w-cutset problem. In UAI-04.

Coast, S. (2009). OpenStreetMap. Available on the web at http://www.openstreetmap.org, accessed Fall 2009.

Damm, D. (1980). Interdependencies in activity behavior. *Transportation Research Record 750*.

Dautelle, J.-M. (2007). JScience 4.1. A comprehensive scientific library for java including a coordinates module for geographic applications.

Dechter, R. (2003). Constraint Processing. Morgan Kaufmann.

Dechter, R., Kask, K., and Mateescu, R. (2002). Iterative join graph propagation. In UAI '02, pages 128-136. Morgan Kaufmann.

Dechter, R. and Larkin, D. (2001). Hybrid processing of beliefs and constraints. In Proc. Uncertainty in Artificial Intelligence, pages 112-119.

Dechter, R. and Mateescu, R. (2003). A simple insight into iterative belief propagation's success. UAI-2003.

Dechter, R. and Mateescu, R. (2004). Mixtures of deterministic-probabilistic networks and their and/or search space. In Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI-04).

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38.

Doucet, A., de Freitas, N., Murphy, K. P., and Russell, S. J. (2000). Rao-Blackwellised particle filtering for dynamic Bayesian networks. In UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, pages 176-183. Morgan Kaufmann Publishers Inc.

Dumais, S. (1998). Beyond content-based retrieval: Modeling domains, users, and interaction. Presented at IEEE ADL'99 - May 21, 1998.

Filgueiras, M. (2007). GPS manager. GPSMan v. 6.3.2.

Foster, D. (2004). The GPX 1.1 schema.

Friedman, N. (2003). Dashboard. Presented at the Linux Symposium, 2003.

Gogate, V. and Dechter, R. (2005). Approximate inference algorithms for hybrid mixed networks. Technical report, Donald Bren School of Information and Computer Science, University of California, Irvine.

Gogate, V., Dechter, R., Bidyuk, B., Rindt, C., and Marca, J. (2005). Modeling transportation routines using hybrid dynamic mixed networks. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 217–224.

Gogate, V., Dechter, R., Bidyuk, B., Rindt, C., and Marca, J. (2006). Modeling travel and activity routines using hybrid dynamic mixed networks. Presented at the 85th annual meeting of the TRB.

Heskes, T. and Zoeter, O. (2002). Expectation propagation for approximate inference in dynamic bayesian networks.

Kitamura, R. (1988). An evaluation of activity-based travel analysis. *Transportation*, 15:9–34.

Klaas, M., Briers, M., de Freitas, N., Doucet, A., Maskell, S., and Lang, D. (2006). Fast particle smoothing: If I had a million particles. In *International Conference on Machine Learning (ICML*, pages 25–29.

Krumm, J. and Horvitz, E. (2006). Predestination: Inferring destinations from partial trajectories. In *UbiComp 2006: Eighth International Conference on Ubiquitous Computing*.

Larkin, D. and Dechter, R. (2003). Bayesian inference in the presence of determinism. In AI-STAT.

Lauritzen, S. (1992). Propagation of probabilities, means, and variances in mixed graphical association models. Journal of the American Statistical Association, 87(420):1098-1108.

Lauritzen, S. L. and Jensen, F. (2001). Stable local computation with conditional gaussian distributions. *Statistics and Computing*, 11:191–203.

Lerner, U. (2002). Hybrid Bayesian Networks for Reasoning about Complex Systems. PhD thesis, Stanford University.

Levine, R. and Casella, G. (2001). Implementations of the Monte Carlo EM algorithm. Journal of Computational and Graphical Statistics, 10:422-439.

Liao, L., Fox, D., and Kautz., H. (2004). Learning and inferring transportation routines. In AAAI-2004.

Liao, L., Patterson, D. J., Fox, D., and Kautz, H. (2007). Learning and inferring transportation routines. *Artificial Intelligence*, 171(5-6):311–331.

Lipe, R. (2007). GPS babel 1.3.4.

Marca, J. E. (2002). The design and implementation of an on-line activity survey. CASA Working paper AS-WP-02-1, Center for Activity Systems Analysis, University of California, Irvine. Presented at the 82nd annual meeting of the Transportation Research Board.

Marca, J. E., Rindt, C. R., and McNally, M. G. (2003). The tracer data collection system: implementation and operational experience. In *Proceedings of the 82nd annual meeting of the Transportation Research Board*, Washington, D. C.

Mateescu, R. and Dechter, R. (2009). Mixed deterministic and probabilistic networks. *Annals of Mathematics and Artificial Intelligence (AMAI); Special Issue: Probabilistic Relational Learning*.

McNally, M. G. and Lee, M. S. (2002). Putting behavior in household travel behavior data: An interactive gis-based survey via the internet. Technical report, Center for Activity Systems Analysis, UC Irvine.

Murphy, K. (2002). Dynamic Bayesian Networks: Representation, Inference and Learning. PhD thesis, University of California Berkeley.

Naveh, B. (2006). JGraphT 0.7. A java library that provides mathematical graph-theory objects and algorithms.

Pas, E. I. (1996). Recent advances in activity-based travel demand modeling. In *Activity-Based Travel Forecasting Conference Proceedings, June 2–5, 1996, Austin Texas*. TMIP.

Patterson, D. J., Liao, L., Fox, D., and Kautz, H. (2003). Inferring high-level behavior from low-level sensors. In Proceedings of UBICOMP 2003: The Fifth International Conference on Ubiquitous Computing, pages 73-89.

Peeta, S. and Zilaskopoulos, A. K. (2001). Foundations of dynamic traffic assignment: The past, the present and the future. Networks and Spatial Economics, 1:233-265.

Quddusa, M. A., Ochieng, W. Y., and Noland, R. B. (2007). Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research, Part C, Emerging Technologies*, 15(5).

Ramsey, P., Santilli, S., Hodgson, C., and Lounsbury, J. (2007). PostGIS v1.3.1. Technical report, Refractions Research.

Recker, W. W. (1995). The household activity pattern problem: General formulation and solution. *Transportation Research, Part B: Methodological*, 29B(1):61–77.

Rubinstein, R. Y. (1981). *Simulation and the Monte Carlo Method*. John Wiley & Sons Inc.

Sherali, H. D., Narayanan, A., and Sivanandan, R. (2003). Estimation of origin-destination trip-tables based on a partial set of traffic link volumes. *Transportation Research, Part B: Methodological*, pages 815–836.

UCI Institute of Transportation Studies (2009). ATMS Testbed. Available on the web at http://www.atmstestbed.net. Accessed Fall 2009.