# A Spatio-Temporal Simulation Framework for Group Behaviour Analysis of Mobile Phone Users

*A thesis submitted in partial fulfillment of the requirements for the degree of*

**Master of Technology**

in

**Computer Science and Engineering**

by

**Aurosish Mishra**

**06CS3027**

Supervised by:

**Prof. Partha Pratim Chakrabarti**

**Prof. Sudeshna Sarkar**


**Department of Computer Science and Engineering**
**Indian Institute of Technology, Kharagpur**
**May 2011**

# CERTIFICATE

This is to certify that the thesis entitled "**A Spatio-Temporal Simulation Framework for Group Behaviour Analysis of Mobile Phone Users**" is a bonafide record of authentic work carried out by **Aurosish Mishra** under my guidance and supervision for the fulfillment of the requirement for the award of the Master of Technology degree in **Computer Science and Engineering** at the Indian Institute of Technology, Kharagpur. The thesis fulfills all requirements as per the regulations of the Institute and in my opinion has reached the requisite standards for submission.

**Prof. Partha Pratim Chakrabarti**
Department of Computer Science and Engineering
IIT Kharagpur

**Prof. Sudeshna Sarkar**
Department of Computer Science and Engineering
IIT Kharagpur

# ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Prof. Partha Pratim Chakrabarti, Department of Computer Science and Engineering, IIT Kharagpur, for extending all the necessary support throughout the duration of the project and for being a constant source of inspiration. Taking time out of his busy schedule, he ensured that the project work was carried out meticulously and methodically. I am greatly indebted to him for all his creative ideas and inspirations which gave me an impetus to come out with constructive creations. I would also like to express my gratitude to Prof. Sudeshna Sarkar for her intellectual guidance, constant attention and unceasing encouragement during the research and development of this project. Special thanks are due to Anshul Gupta for his help in several aspects of the implementation phase of the project.

I would like to avail this opportunity to express my deepest gratitude to my teachers in the Department of Computer Science and Engineering at IIT Kharagpur, who have taught me the value of hard work, perseverance, and thoughtful planning, and instilled in me the right attitude for conducting good research work.

I would also like to thank my parents and my brother who have been an endless source of encouragement for the entire duration of the project; this thesis is simply not conceivable without their blessings and support.

**Aurosish Mishra (06CS3027)**
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

May 9, 2011

# Contents

# *Abstract*

*In the past decade, the world has seen significant advancements in the field of telecommunications. Mobile network operators continuously seek to design new and innovative ways for providing highly personalized services to their users. One of the most powerful ways to personalize mobile services is based on location. With the advent of location-aware devices with positioning functionalities, the concept of location-based services (LBS), for providing information as well as entertainment, came into existence. As phones become 'location-aware', there is an opportunity to not only ascertain a user's physical location when they make a query, but to store and analyse a history of previous locations. Given user approval, analysis of the spatio-temporal characteristics of the user's current and recent movement can help provide accurate response to information requests.*

*The development of these sophisticated position-aware phones has created vast, dynamic spatio-temporal data sets for every individual who carries a phone. This has lead to the development of new data models and forms of analysis to find spatial and temporal trends. Spatio-temporal data mining techniques play a major role in knowledge discovery and delivery in location-based services. But this branch of data mining is still in its infancy. New algorithms are being developed everyday for mining user profiles, identifying frequent patterns, and even classifying maximal user groups. In order to test these algorithms, we need to have actual spatio-temporal data for mobile users. However, due to privacy issues, most of the datasets remain proprietary information of the service providers. In such a scenario, it becomes increasingly important to design simulation environments that will provide realistic spatio-temporal data to serve as testbeds for the algorithms.*

*In this work, we present the design principles behind a simulator that realistically models user behaviour - both at an individual as well as the group level. We analyse the existing mobility models and come up with an intuitive behaviour model for different user-classes (such as office-goers, college-students). We develop a formal representation for modelling different daily activity sequences of a user, which he performs either as part of a group or prefers to do them individually. We generate spatio-temporal movement patterns of a set of 1000 users on the IIT Kharagpur campus map and also present a thorough analysis of the obtained results.*

# Chapter 1

# Introduction

The dawn of the 21st century heralded a world of wireless communication, location-aware devices with geographic information system functionalities(GPS), and many more path-breaking innovations. With the advent of such sophisticated technology, location-based services [1] came into being. Users of a particular e-service involving location information disclose their positional information to the service, which in turn uses this information along with additional details to provide certain functionalities. Services accumulate data derived from the users' requests and integrate this with other user information in a database. Spatio-temporal data mining techniques are used for knowledge discovery and delivery in such location-based services.

Listed below are a few of the applications of location-based services [2].

- Recommending social events in a city.
- Requesting the nearest business or service, such as an ATM or restaurant
- Turn by turn navigation to any address
- Locating people on a map displayed on the mobile phone
- Receiving alerts, such as notification of a sale on gas or warning of a traffic jam
- Location-based mobile advertising
- Asset recovery combined with active RF to find, for example, stolen assets in containers where GPS wouldn't work

- Games where your location is part of the game play, for example your movements during your day make your avatar move in the game or your position unlocks content.
- Real-time Q &A revolving around restaurants, services, and other venues

The data mining field dates back almost 20 years. However, the field of spatiotemporal data mining [3], which is extremely challenging due to the exponential explosion of the search space for knowledge, is still in its infancy. Spatio-temporal data simply refers to data that accomodates both time and space coordinates. Several algorithms, such as those for mining user groups, identifying frequent trajectories, obtaining user events, etc. , have been developed in the field of spatio-temporal data mining. In order to test these algorithms, we need to have a good test-bed of spatio-temporal data. Often the task of collecting such data is costly in terms of money involved in setting up the necessary infrastructure, the time involved and issues of privacy among users. Therefore, a simple way to collect such data, is to design a realistic simulator that generates user behaviour - both individual and group, of a set of users in a well-defined simulation area for a fixed time period.

## 1.1 Motivation

In our day-to-day activities, we are affiliated to groups of many different sorts. Be it going to the college with our batchmates, going to the office with colleagues or even going to the movies with friends; groups form an essential part of our day-to-day life. Group dynamics and its influence on individual decision making have been well studied by sociologists, and it has been shown that peer pressure and group conformity can affect the behaviours of individuals. With a good knowledge of groups a customer belongs to, one can derive common interests among customers, develop group-specific behaviour models and provide personalized recommendation services.

There are many ways one can determine the groups a person belongs to, e.g., partitioning people into groups based on their purchase behaviour, such as past purchases of same product items; similar occupations, incomes; social behaviours such as partying, attending functions and so on. A very interesting group classification can be made based on the spatio-temporal information of users, namely

analysing their movement data. This brings into the fore-front two additional classification techniques:

- *Physical proximity between group members.* The group members are expected to be physically close to one another when they acts as a group. Such characteristics are common among many types of groups, e.g., shopping pals, game partners, etc.
- *Temporal proximity between group members.* The group members are expected to stay together for some meaningful duration when they act as a group. Such characteristic distinguishes an ad hoc cluster of people who are physically close from a group of people who come together for some planned activity.

Striking out a balance between the two sets of information can help us analyze group behaviour of a set of users. In this project, we consider the class of mobile phone users. A recent study has shown that there are around 5 billion phones in use across the world in 2010 which is more than 65% of the total world population. Typically, every mobile user carries his device for a major part of the day. This indicates that by analyzing the data of mobile users, we can get a very clear picture of their day-to-day activities. So, we focus our attention on obtaining movement patterns of mobile phone users.

As stated before, the cost of obtaining real-world data often outweighs its merits of being realistic. Even if we managed to collect such data, it would be specific to the region and might not represent a global behaviour. For this purpose, we aim to design a realistic simulator that provides us with spatio-temporal data as close to the actual user behaviour as possible. In addition to this, a simulator provides us with much better control over the data generated, in terms of being able to tune several parameters, such as user density, user activities, traffic patterns, and size of user groups, as well as vary several other constraints.

## 1.2 Background

Data mining and data analysis [4] have a long history in human-computer interaction, starting with early interests in tracking the users to infer models for adaptive systems, to more recent interests in attentional user interfaces, notifier systems, and recommenders. Data mining, also known as knowledge discovery, is the process of extracting valuable information from large amounts of data. Data mining technology provides us with the following capabilities:

- *Automated prediction of trends and behaviours.* Data mining automates the process of finding predictive information in large databases. Questions that traditionally required extensive hands-on analysis can now be answered directly from the data - quickly. A typical example of a predictive problem is targeted marketing. Data mining uses data on past promotional mailings to identify the targets most likely to maximize return on investment in future mailings.

- *Automated discovery of previously unknown patterns.* Data mining tools sweep through databases and identify previously hidden patterns. An example of pattern discovery is the analysis of retail sales data to identify seemingly unrelated products that are often purchased together.

In mobile and collaborative applications, data mining has been used to analyze users' behaviour, demonstrating how feedback produced by such analysis can radically change people's behaviours.

Mobile user data mining [3] is the process of extracting interesting information from the data collected from mobile users through various data mining methodologies. Today, most of the people in the world own a mobile phone, hence the data analysis we do on mobile users could be used for obtaining fast and inexpensive user-movement studies. Also, we obtain first-hand information about general patterns followed by a group of users such as knowledge about the set of locations they tend to visit, the set of people they tend to meet, and time periods during which they are typically active. Once we have understood the interests and needs of a user, vis-a-vis his group behaviour, we can design a good recommendation system providing suggestions varying from options for restaurants to choices for shopping malls, from recommendations for movies to automobiles to purchase.

Traditionally, temporal and spatial data mining were the two different approaches for mining data-sets. However, with the exploration of application domains in which the spatial and temporal contexts are intricately inter-woven, such as disaster management, tracking weather anomalies, analysing spread of epidemics,etc., techniques belonging to a confluence of the two, namely spatio-temporal data-mining came into existence.

Temporal data mining concerns the analysis of events ordered by one or more dimensions of time. There are two broad directions. One concerns the discovery of causal relationships among temporally-oriented events. The other concerns the

discovery of similar patterns within the same time sequence or among different time sequences. This latter area, commonly termed time series analysis (or trend analysis) focuses on the identification of similar pre-specified patterns, similar peaks for example, among time series. On the other hand, spatial data mining deals with finding patterns in geography from spatial data.

Research accommodating both spatial and temporal data mining is sparse at present; however, two directions can be identified. Firstly, the embedding of a temporal awareness in spatial systems, and secondly, the accommodation of space into temporal data mining systems. To date, the former approach has been the more popular, in part because of the relative maturity of geographic information systems and the availability of time-stamped snapshots of geographic/spatial test data.

## 1.3 Problem Overview

This project involves generating realistic spatio-temporal data using a simulator and developing mining algorithms to work on that data and obtain useful user information. The basic objectives of the project are as follows:

- Prepare a survey of the existing individual and group mobility models, and state-of-the-art simulators.
- Design a realisitc mobility simulator for generating readily-available and tunable spatio-temporal movement data of different user classes in the simulation region over a period of one month.
- Develop anytime algorithms for mining user groups, finding frequent trajectories, and characterizing events from the available spatio-temporal data.
- Using the developed mining algorithms and dynamic spatio-temporal data, build an e-service provider (such as a recommendation system or an event planner)

One of the long-term goals of this project is to design a recommendation system that will give relevant suggestions to a mobile user based on his profile details (self-reported and system-learned) such as his social networks, interests, hobbies etc. and his spatio-temporal data available. In order to achieve this goal, the project can be divided into three broad aspects, namely:

1. *Designing a Simulator.* The simulator would generate data for a set of mobile users in a particular simulation region. This would include data about his movement patterns corresponding to his regular and irregular behaviours, as well as his group oriented behaviour. It is preferable that the simulator generate event-driven behaviour (a higher layer of abstraction), rather than generating arbitrary random movements of users. In the absence of real traces, the data from the simulator would serve as the testing medium for the developed mining algorithms.

2. *Mining the User Profile.* This task involves mining the data relevant to a specific user in order to obtain patterns regarding his interests, his daily activities (with emphasis on spatio-temporal details), etc. Thus, we can know about the places the user often visits, the routes he regularly traverses, or the restaurant he frequents. Apart from this, we can also obtain information about the user's social group based on commonality of interests. For this purpose, we have to carefully define the specifications regarding trajectories, events and groups.

3. *Designing a Recommender Service.* After mining out the details about a person's interests and activities, we can proceed to design the recommendation service. Recommendations for a user can be broadly classified into two catagories:

    - *Recommendations that are specific to the user.* Example: Update about the theatre in which the new movie he has been waiting for is playing, information about his/her close friend passing nearby, etc.
    - *Recommendations that are specific to a group to which the user belongs.* Example: Updates about the restaurant he visits with his work colleagues, information about an art exhibition his family wants to visit, etc

Apart from this, we can also design services to track anomalies, assist in disaster management, and help in event planning.

## 1.4 Summary of Work Done

The focus of my work was to design a realisitc mobility simulator that would generate readily available and tunable spatio-temporal data. This task can be divided into three important problems - generating individual user behaviour from

the user-class specifications, generating group behaviour for individuals from the group specifications, and finally combining both of them such that both sets of specifications are satisfied.

For this purpose, I started off by surveying the existing mobility models and popular simulation frameworks. The existing models comprised a well-classified collection of individual and group mobility models. However, none of the relevant simulators captured the realisitc combination of both group and individual behaviour of any particular user class. Furthermore, any available spatio-temporal data was too campus-specific and could not be generalized to any random scenario. Thus, I came up with a realistic model for capturing the intuitive behaviour for different classes of users (such as office-goers, college-students, etc). A simple language was developed for the simulator model. [1]

Using the supplied individual user class specifications, and specifications for different group types, the individual user behaviour and group behaviour were simulated. Thorough and exhaustive analysis was done to verify whether independent or simultaneous generation of the two types of behaviour, would satisfy both sets of constraints. Activities corresponding to both sets of behaviours, for a single day, were combined together. If any activity clashes arose for a particular time interval, they were resolved by using overriding probabilities, which were fixed while keeping in mind the notion of planned group behaviour and priority user behaviour. Several consistency checks were also developed to validate the given sets of individual and group specifications. Thus, we can also verify whether the data generated for each individual user, when combined, gives back the provided user-class specifications, within a certain degree of error. Using this information, we can obtain combination of activities that is valid for a set of users in a particular user-class. Similarly, we can define other combinations that would lead to inconsistent behaviour states and hence should be avoided.

The actual simulator was tuned by parameters collected by a small survey made in the IIT Kharagpur campus itself. The simulator generated data on the IIT campus map obtained from the Google Static Maps API. Spatio-temporal data for 1000 mobile users comprising two user classes, namely office-goers and college-students, with about 5 different group types, was generated for a period of one month, with data being reported at 3 minute intervals. The validation

[1]Part of the work has been done in collaboration with Mr. Anshul Gupta, 4th Year B.Tech student, Department of Computer Science and Engineering, IIT Kharagpur

mechanism included checking for data consistency vis-a-vis the specifications for two types of data sets - data obtained for the realistic IIT Kharagpur campus map as well as for different sets of randomly generated maps.

## 1.5 Organization of Thesis

The remainder of the thesis is organized into several chapters as follows:

- *Chapter 2* presents a survey of the existing mobility models, which includes several individual and group mobility models, as well as an analysis of the recently developed RealMobGen simulator. A brief account of our specific requirements vis-a-vis the existing literature is also presented.

- *Chapter 3* presents the developed individual and group behaviour model for different user classes (office-goers and college students). An idea is given regarding the various input specifications for the simulator - for both individual and group behaviour. Algorithms used at various stages of the simulator for finding paths, grouping hotspots, performing consistency checks, resolving clashes between the two behaviour types, are also discussed. Apart from this, techniques for validating the simulator data (in accordance with the specifications) are also described.

- *Chapter 4* presents an account of all the work done. This includes details regarding the implementation of the simulator, as well as, spatio-temporal data (both individual and group) for 1000 mobile users comprising office-goers and college-students, at a frequency of 3 minutes, for a period of one month. We also present ideas regarding consistency verification of the supplied simulator specifications.

- *Chapter 5* outlines the conclusions of the project and prepares an action plan to be followed in the near future.

# Chapter 2

# Literature Survey: Simulation and Mobility Models

Simulation is a very useful tool for generating data that mimicks a user network. It is also one of the most important methods for evaluating the characteristics of a mobile network (or any of its related protocols). It provides researchers with a number of significant benefits, including repeatable scenarios, isolation of parameters, and exploration of a variety of metrics. Our task is to design a simulator to generate the topology and movement patterns of different mobile users in a region. Once the nodes have been initially distributed in the simulation area, mobility models dictate the movement of the nodes within the network. So, the accuracy of the simulation process depends on how realistic the generated movement patterns are. Thus, creating realistic mobility models for simulation environments emerges as a first-class design requirement.

## 2.1 Designing Mobility Models

Mobility models represent the movement of mobile users, and how their location, velocity and acceleration change over time. They form the backbone of the design of any simulation process. A good mobility model should attempt to mimic the movements of real mobile networks. Changes in speed and direction must occur and they must occur in reasonable time slots. For example, we would not want mobile users to travel in straight lines at constant speeds throughout the course of the entire simulation because real users would not travel in such a restricted manner.

In order to mimic any real-life network, it is very important to carefully simulate the movement patterns of the actors involved, i.e. design an accurate mobility model. There are two main approaches, namely:

- *Model-to-Trace approach.* It consists of developing mathematical mobility models(also called synthetic models) that attempt to capture the mobility characteristics of certain scenarios. Usually, these models consist of randomly generated movement patterns and hence, do not capture real-world scenarios very effectively.

- *Trace-to-Model approach.* It consists of measuring mobility traces from actual applications, characterizing them, and then designing mobility models derived from such characterizations. Traces are obtained by means of measurements of deployed systems and usually consist of logs of connectivity or location information. This is a fairly recent approach and the developed models focus on specific scenarios that are very hard to generalize.

Both approaches have their own sets of advantages and limitations. Historically, however, synthetic movement models have been mostly preferred. The reasons of this choice are many-fold:

- First of all, the publicly available traces are limited. Telecommunication companies usually collect and analyze large sets of data but these are kept secret since they may represent a source of competitive advantage, for example, for investments and marketing choices.

- Secondly, these traces are related to very specific scenarios (such as campus environments) and it is currently difficult to generalize their validity. However, it is important to note that these data show surprising common statistical characteristics, such as the same distribution of the duration of the contacts and inter-contacts intervals.

- Thirdly, the available traces do not allow for sensitivity analysis of the performance of algorithms, since the values of the parameters that characterize the simulation scenarios, such as the distribution of the speed or the density of the hosts, cannot be varied. So, the data obtained cannot be tuned properly.

- Finally, in some cases, it may be important to have a mathematical model underlying the movement of the hosts in simulations, in order to formally analyze its impact on the design of protocols and systems.

We now analyze the various types of synthetic and trace-based models currently being used. We describe several mobility models that represent mobile nodes whose movements are independent of each other (i.e. entity mobility models) and several mobility models that represent mobile nodes whose movements are dependent on each other (i.e. group mobility models). Apart from that we also look at constrained topology mobility models in which real-world scenarios such as freeway scenarios, city-block scenarios are handled. Also, we present a brief overview of the various state-of-the-art simulators and simulation frameworks being used to generate mobile-user networks.

## 2.2 Synthetic Mobility Models

Synthetic models are chiefly, mathematical models, such as sets of equations, which try to capture the movement of mobile devices. Based on whether they mimic the behaviour of an individual mobile user or the group behaviour of a set of users, synthetic models are classified into two types:

1. *Entity Movement Models* : capture the behaviour of individual users.
2. *Group Mobility Models* : capture the behaviour of a group of users.

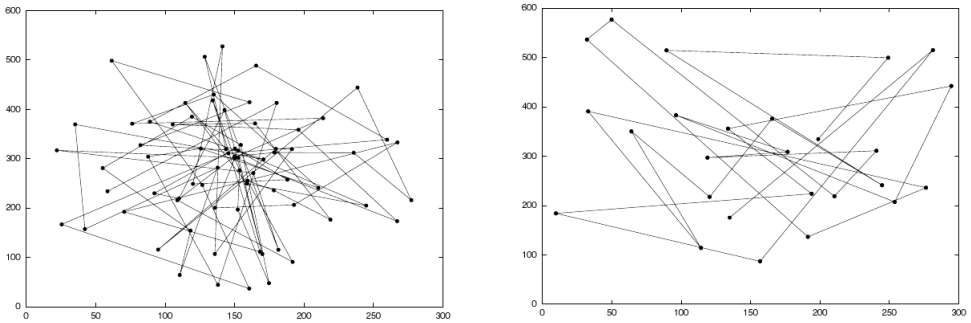
### 2.2.1 Entity Mobility Models

In this section, let us analyze the popular entity mobility models [5] used in the simulation of individual mobile users ($MN$). The first two models presented, the Random Walk Mobility Model and the Random Waypoint Mobility Model, are the two most common mobility models used by researchers.

#### Random Walk Model

The Random Walk Mobility Model [5] is a widely used mobility model which is similar to Brownian Motion. In this mobility model, a mobile node($MN$) moves from its current location to a new location by randomly choosing a direction and speed in which to travel. The new speed and direction are both chosen from predefined ranges, [speedmin, speedmax] and [0,2$\pi$] respectively. Each movement in the Random Walk Mobility Model occurs in either a constant time interval $t$ or a constant distance travelled $d$, at the end of which a new direction and speed are calculated. If an $MN$ which moves according to this model reaches a simulation boundary, it 'bounces' off the simulation border with an angle determined by the incoming direction. The $MN$ then continues along this new path.

Fig. 2.1(a) shows an example of the movement of a MN following the random walk model. At each point, the MN randomly chooses a direction between 0 and $2\pi$ and a speed between 0 and 10 m/s. The MN is allowed to travel for 60 seconds before changing direction and speed. The Random Walk Mobility Model is a memoryless mobility pattern because it retains no knowledge concerning its past locations and speed values. The current speed and direction of an MN is independent of its past speed and direction. This characteristic can generate unrealistic movements such as sudden stops and sharp turns (see Fig. 2.1(a)).




(a) Travelling Pattern of a MN using the Random Walk Mobility Model

(b) Travelling Pattern of a MN using the Random Waypoint Mobility Model

Figure 2.1: Two Popular Entity Models

**Random Waypoint Model**

The Random Waypoint Mobility Model [5] tries to introduce a certain amount of realism into the random walk model. It includes pause times between changes in direction and/or speed. An MN begins by staying in one location for a certain period of time (i.e. a pause time). Once this time expires, the MN chooses a random destination in the simulation area and a speed that is uniformly distributed between [minspeed, maxspeed]. The MN then travels toward the newly chosen destination at the selected speed. Upon arrival, the MN pauses for a specified time period before starting the process again. These points of pauses are called as *Waypoints.*

Fig. 2.1(b) shows an example traveling pattern of an MN using the Random Waypoint Mobility Model starting at a randomly chosen point or position, the speed of the MN in the figure is uniformly chosen between 0 and 10 m/s. We note that the movement pattern of an MN using the Random Waypoint Mobility

Model is similar to the Random Walk Mobility Model if pause time is zero and [minspeed, maxspeed] = [speedmin, speedmax]. The Random Waypoint Mobility Model is also a widely used mobility model. However, it is still quite unrealistic. The initial placement of the nodes in the network does not mirror any real-world situation. Also, the nodes tend to concentrate in the middle of the area if we consider a bounded region, thus, leading to large unused simulation regions.

**Random Direction Model**

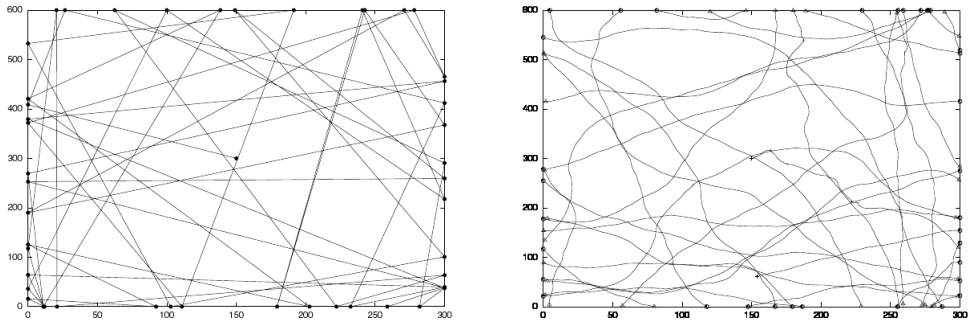
The Random Direction Mobility Model [5] was created to overcome density waves in the average number of neighbours produced by the Random Waypoint Mobility Model. A density wave is the clustering of nodes in one part of the simulation area. In the case of the Random Waypoint Mobility Model, this clustering occurs near the center of the simulation area. In the RWP Mobility Model, the probability of an MN choosing a new destination that is located in the center of the simulation area, or a destination which requires travelling through the middle of the simulation area, is high (See Fig. 2.1(b)). Thus, the MNs appear to converge, disperse, and converge again.

In this model, MNs choose a random direction in which to travel, similar to the Random Walk Mobility Model. An MN then travels to the border of the simulation area in that direction. Once the simulation boundary is reached, the MN pauses for a specified time, chooses another angular direction (between 0 and 180 degrees) and continues the process. Basically, the way-points occur along the boundary of the simulation area. Fig. 2.2(a) shows an example path of an MN using the Random Direction Mobility Model. The dots in the figure illustrate when the MN has reached a border, paused, and then chosen a new direction.

**Boundless Simulation Area Model**

In the Boundless Simulation Area Mobility Model [5], a relationship between the previous direction of travel and velocity of an MN with its current direction of travel and velocity exists. A velocity vector $\overrightarrow{v} = (v, \theta)$ is used to describe an MN's velocity $v$ as well as its direction $\theta$; the MN's position is represented as $(x, y)$. Both the velocity vector and the position are updated at every $\triangle\ t$ time steps according to the following formulaes:

$v(t+\triangle\ t) = min[max(v(t)+\triangle\ v; \theta); V_{max}];$
$\theta(t+\triangle\ t) = \theta(t)+\triangle\ \theta;$



(a) Travelling Pattern of a MN using the Random Direction Model

(b) Travelling Pattern of a MN using the Boundless Simulation Area Mobility Model

Figure 2.2: Variants of the Random Model

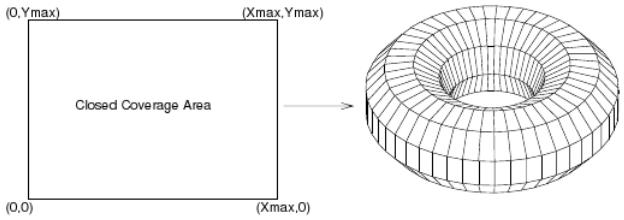



Figure 2.3: Torus-shaped Simulation Area

$x(t+ \triangle\ t) = x(t) + v(t) * \cos\theta(t);$
$y(t+ \triangle\ t) = y(t) + v(t) * \sin\theta(t);$

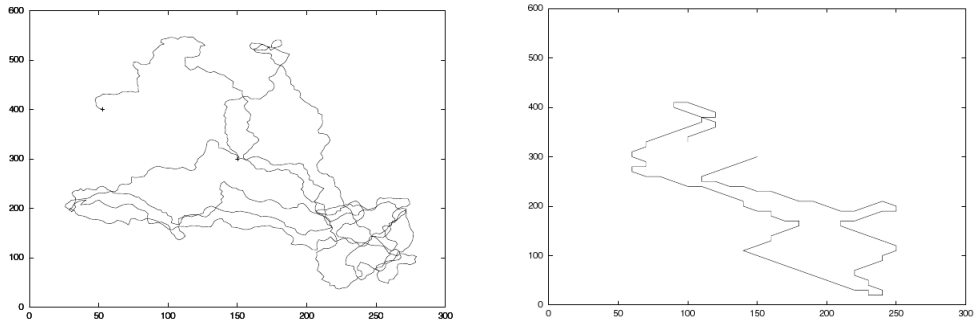
where:
$V_{max}$ is the maximum velocity defined in the simulation, $\triangle\ v$ is the change in velocity, $\triangle\ \theta$ is the change in direction of the MN

In all the mobility models previously mentioned, MNs reflect off or stop moving once they reach a simulation boundary. In the Boundless Simulation Area Mobility Model, MNs that reach one side of the simulation area continue traveling and reappear on the opposite side of the simulation area. This technique creates a torus-shaped simulation area allowing MNs to travel unobstructed (see Fig. 2.3).

### Gauss Markov Model

The Gauss-Markov Mobility Model [5] was designed to adapt to different levels of randomness via one tuning parameter. Initially each MN is assigned a current speed and direction. At fixed intervals of time, $n$, movement occurs by updating the speed and direction of each MN. Specifically, the value of speed and direction at the $n^{th}$ instance is calculated based upon the value of speed and direction at the $(n-1)^{th}$ instance and a random variable. At each time interval the next location is calculated based on the current location, speed, and direction of movement. To ensure that an MN does not remain near an edge of the grid for a long period of time, the MNs are forced away from an edge when they move within a certain distance of the edge.

Fig. 2.4(a) illustrates an example traveling pattern of an MN using the Gauss-Markov Mobility Model; the MN begins its movement in the center of the simulation area or position (150, 300) and moves for 1000 seconds.




(a) Travelling Pattern of a MN using the Gauss Markov Mobility Model

(b) Travelling Pattern of a MN using the Probabilistic Random Walk Model

Figure 2.4: Other Important Models

### A Probabilistic Version of Random Walk Model

Chiang's mobility model [5] utilizes a probability matrix to determine the position of a particular MN in the next time step, which is represented by three different states for position $x$ and three different states for position $y$. State 0 represents the current ($x$ or $y$) position of a given MN, State 1 represents the MN's previous ($x$ or $y$) position, and State 2 represents the MN's next position if the MN continues to move in the same direction. Each entry in $P(a, b)$ in the

$3 \times 3$ probability matrix represents the probability that an MN will go from State $a$ to State $b$. The values within this matrix are used for updates to both the MN's $x$ and $y$ positions.

This implementation produces probabilistic rather than purely random movements, which may yield more realistic behaviours. For example, as people complete their daily tasks they tend to continue moving in a semi-constant forward direction. Rarely do we suddenly turn around to retrace our steps, and we almost never take random steps hoping that we may eventually wind up somewhere relevant to our tasks. However, choosing appropriate values of $P(a, b)$ may prove difficult, if not impossible, for individual simulations unless traces are available for a given movement scenario. Fig. 2.4(b) illustrates an example traveling pattern of an MN using the Probabilistic Version of the Random Walk Mobility Model.

### 2.2.2 Group Mobility Models

In Section 2.2.1, we discussed mobility models that represent multiple MNs whose actions are completely independent of each other. In real life, however, there are many situations where it is necessary to model the behaviour of mobile users as they move together. For example, a group of soldiers in a military scenario may be assigned the task of searching a particular plot of land in order to destroy land mines, capture enemy attackers, or simply work together in a cooperative manner to accomplish a common goal. Or a group of people go to a restaurant for dinner. In order to model such situations, a group mobility model is needed to simulate that cooperative characteristic. In this section, we present six group mobility models. We note that four of the these group mobility models are closely related. The most general of these models is the Reference Point Group Mobility (RPGM) model. Specifically, three of the group mobility models (Column, Nomadic, and Pursue) can be implemented as special cases of the RPGM model.

#### Exponential Correlated Random Mobility Model

One of the first group mobility models to be proposed is the Exponential Correlated Random Mobility Model [5]. In this model, a motion function is used to create MN movements. Given a position (MN or group) at time $t$, $b(t)$ is used to define the next position (MN or group) at time $t+1$, given by the following formula:

$$b(t+1) = b(t)e^{-1/\tau} + (\sigma\sqrt{1-(e^{-1/\tau})^2})r$$

where, $\tau$ adjusts the rate of change from the MN's previous location to its new location (i.e. a small $\tau$ equates to large change) and $r$ is a random Gaussian variable with variance $\sigma$.

Unfortunately, it is not easy to create a given motion pattern by selecting appropriate values for the parameters in the Exponential Correlated Random Mobility Model. The next four group mobility models improve upon this drawback.

### Column Mobility Model

The Column Mobility Model [5] proves useful for scanning or searching purposes. This model represents a set of MNs that move around a given line (or column), which is moving in a forward direction (e.g., a row of soldiers marching together towards their enemy). A slight modification of the Column Mobility Model allows the individual MNs to follow one another (e.g., a group of young children walking in a single-file line to their classroom).

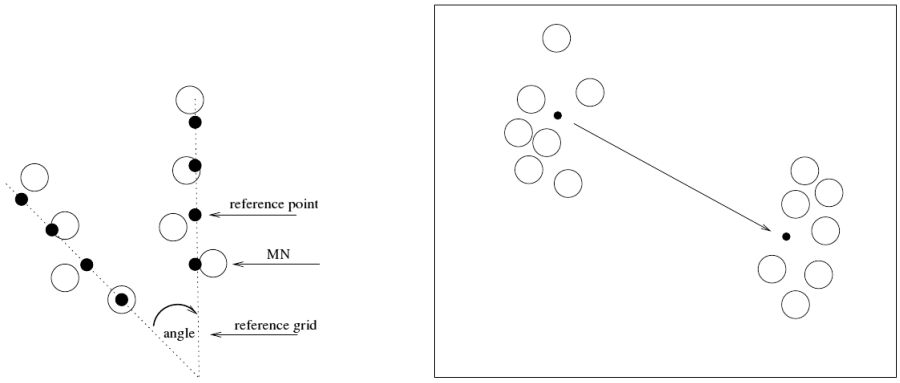
For the implementation of this model, an initial reference grid (forming a column of MNs) is defined. Each MN is then placed in relation to its reference point in the reference grid; the MN is then allowed to move randomly around its reference point via an entity mobility model. (The authors propose using the RWP Mobility Model, which is described in Section 2.2.1, as the entity mobility model.) The new reference point for a given MN is defined as:

$$new\_reference\_point = old\_reference\_point + advance\_vector$$

where, $old\_reference\_point$ is the MN's previous reference point and $advance\_vector$ is a predefined offset that moves the reference grid. The predefined offset that moves the reference grid is calculated via a random distance and a random angle (between 0 and $\pi$ since movement is in a forward direction only). Since the same predefined offset is used for all MNs, the reference grid is a 1-D line. Fig. 2.5(a) gives an illustration of four MNs moving in the Column Mobility Model.

### Nomadic Community Mobility Model

The Nomadic Community Mobility Model [5] represents groups of MNs that collectively move from one point to another. Within each community or group of MNs, individuals maintain their own personal 'spaces' where they move in random ways. Numerous applications exist for this type of scenario.




(a) Movements of three MNs using the Column Mobility Model

(b) Movements of seven MNs using the Nomadic Mobility Model

Figure 2.5: Group Mobility Models

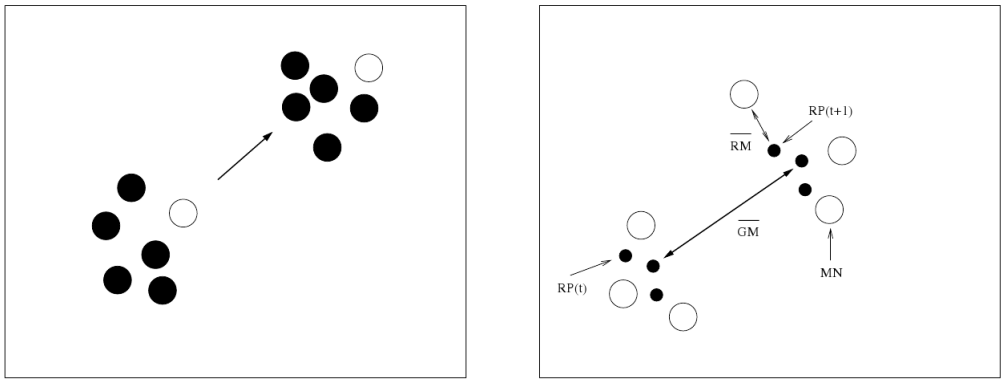
In the Nomadic Community Mobility Model, each MN uses an entity mobility model (e.g., the Random Walk Mobility Model) to roam around a given reference point. When the reference point changes, all MNs in the group travel to the new area defined by the reference point and then begin roaming around the new reference point. The parameters for the entity mobility model define how far an MN may roam from the reference point. Compared to the Column Mobility Model, the MNs in the Nomadic Community Mobility Model share a common reference point versus an individual reference point in a column. Thus, we would expect the MNs to be less constrained in their movement around the defined reference point. Fig. 2.5(b) illustrates the movement of seven MNs using this mobility model.

### Pursue Mobility Model

The Pursue Mobility Model [5] attempts to represent MNs tracking a particular target. For example, this model could represent police officers attempting to catch an escaped criminal. The Pursue Mobility Model consists of a single update equation for the new position of each MN:

$$new\_position = old\_position + acceleration * (target - old\_position) + random\_vector$$

where, $acceleration * (target - old\_position)$ is information on the movement of the MN being pursued and $random\_vector$ is a random offset for each MN. Fig. 2.6(a) illustrates this model.




(a) Movements of six MNs using the Pursue Mobility Model

(b) Movements of three MNs using the Reference Point Group Mobility Model

Figure 2.6: Group Mobility Models

### Reference Point Group Mobility Model

The Reference Point Group Mobility (RPGM) Model [6] mimics the behaviour of a group of hosts that move as a single entity to perform some task. It is a very effective framework and may be adapted to model various scenarios, such as battlefield situation, disaster recovery and convention scenario. In this model, each group has a logical 'center'. The center's motion defines the entire group's motion behaviour, including location, speed, direction, acceleration, etc. Thus, the group trajectory is determined by providing a path for the center.(see Fig. 2.6(b))Usually, nodes are uniformly distributed within the geographic scope of a group. To node, each is assigned a reference point which follows the group movement. A node is randomly placed in the neighborhood of its reference point at each step. The reference point scheme allows independent random motion behaviour for each node, in addition to the group motion.

The RPGM model defines the motion of groups explicitly by giving a motion path for each group. A path which a group will follow is given by defining a sequence of checkpoints along the path corresponding to given time intervals. As time goes by, a group moves from one check point to the next on a continuing basis. Each time the group center reaches a new check point, it computes the new motion vector from current and next check point locations and from the time interval.

By proper selection of check points, one can easily model many realistic situations, where a group must reach predefined destinations within given time intervals to accomplish its task. For example, this can be used to model In-Place Mobility Models in which different groups have been assigned specific zones for their activities. At the same time, it can also be used to model Overlap Mobility Models in which different groups carry out their jobs in the same area (like various rescue teams in a disaster area).The check point scenario file has the advantage of decoupling the motion pattern from the model itself. Many methods can be used to generate a scenario file, such as, typing in manually, digitizing a route from a map, using outputs from a program or a profile from real world. The model has the advantages of providing a general and flexible framework for describing mobility patterns, which are task oriented and time restricted as well as easy to implement and verify.

**Mobility Vector Model**

The Mobility Vector Model [7] provides a more general framework. It can be used to describe a very large set of scenarios. It is especially useful in a heterogeneous environment where different types of hosts have different mobility patterns. In this model, the velocity of any host, the Mobility Vector, is composed of a Base Vector and a Deviation Vector: $\overrightarrow{M} = \overrightarrow{B} + \alpha \overrightarrow{V}$ ,where $\alpha$ is the acceleration factor. The Base Vector defines the primary velocity component and the Deviation Vector defines the deviation from the Base Vector. By changing $\overrightarrow{M}, \overrightarrow{V}, \alpha$, we can produce many different mobility patterns. For example, if we use Base Vector to represent the movement of the whole group and use Deviation Vector to represent the movement of individual hosts within the group, we get the RPGM model. One of the motivations behind the Mobility Vector Model is to make movement smooth. Some statistical models like Random Waypoint Model will produce unrealistic movements such as sudden stops and sharp turns. We can use the Mobility Vector Model to make it more realistic. For example, if we choose a negative $\alpha$, we can mimic the deceleration of a host approaching its destination. The host slows down and stops at the destination, while in Random-Waypoint Model it may suddenly stop at the destination. Average speed, distance travelled, contact range are some of the input parameters for a simulator implementing this model.

The Mobility Vector Model provides the framework for several other models as follows:

1. *Gravity Model.* In some wireless communication systems, receivers may tend to move towards the signal source, looking for a better signal. For example,

in a cellular system, if a user experiences a low quality of communication and can move around, he may try to move towards a Base Station. The Gravity Model reproduces the above mobility patterns.

2. *Location Dependent Model.* This model represents a collective mobility pattern in a specific area. For example, if a node is on a freeway, its mobility vector has a common component which represent the direction and the allowed speed of the freeway. If we have a digitized map and traffic pattern based on the map,we can use the base vector to implement the collective mobility. When a node moves around the area, it acquires the location dependent base vector specified at the current position.

3. *Targeting Model.* Targeting is a common pattern of mobility, where nodes move towards a target. Given the target coordinate, it is simple to calculate a proper base vector. When a node approaches a target, it reduces its velocity using negative acceleration factor and then pause when the mobility vector is adjusted to zero. This is an improved implementation of Random Waypoint model which avoids sudden stops.

4. *Group Motion Model.* In ad hoc networks, communications are often among teams which tend to coordinate their movements (e.g., a rescue team in a disaster recovery situation). To support this kind of communications and movements, the Mobility Vector model can provide efficient and realistic group mobility models. Different group patterns can be represented using base vectors while deviation vectors show the individual behaviours of members in a group. Thus the model can provide exible group motion patterns for heterogeneous networks, such as those including UAVs (Unmanned Aerial Vehicles).

### 2.2.3 Other Useful Models

#### Individual Mobility: The Truncated Power Law

Despite their importance for urban planning, traffic forecasting, and the spread of biological and mobile viruses, our understanding of the basic laws governing human motion remains limited thanks to the lack of tools to monitor the time resolved location of individuals. Barabasi et. al [8] studied the trajectory of 100,000 anonymized mobile phone users whose position was tracked for a six month period. They found that in contrast with the random trajectories predicted by the prevailing Levy flight and random walk models, human trajectories show a

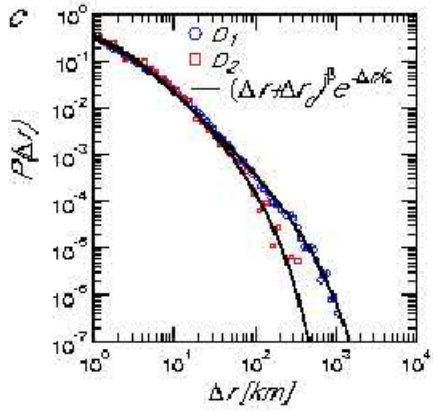



Figure 2.7: The Truncated Power Law

high degree of temporal and spatial regularity, each individual being characterized by a time independent characteristic length scale and a significant probability to return to a few highly frequented locations. After correcting for differences in travel distances and the inherent anisotropy of each trajectory, the individual travel patterns collapse into a single spatial probability distribution, indicating that despite the diversity of their travel history, humans follow simple reproducible patterns. This inherent similarity in travel patterns could impact all phenomena driven by human mobility, from epidemic prevention to emergency response, urban planning and agent based modelling.

They used two data sets to explore the mobility pattern of individuals. The first set $D_1$ consists of the mobility patterns recorded over a six month period for 100,000 individuals selected randomly from a sample of over 6 million anonymized mobile phone users. Each time a user initiates or receives a call or SMS, the location of the tower routing the communication is recorded, allowing the reconstruction of the user's time resolved trajectory. The time between consecutive calls follows a bursty pattern, indicating that while most consecutive calls are placed soon after a previous call, occasionally there are long periods without any call activity. To make sure that the obtained results are not affected by the irregular call pattern, they also study a data set $D_2$ that captures the location of 206 mobile phone users, recorded every two hours for an entire week. In both datasets the spatial resolution is determined by the local density of the more than 104 mobile towers, registering movement only when the user moves between areas serviced by different towers. It was observed that the distribution of displacements over all users could be well approximated by a truncated power-law:

$$P(\Delta r) = (\Delta r + \Delta r_0)^{-\beta} exp(\frac{-\Delta r}{\kappa})$$

with $\beta = 1.75, \Delta r_0 = 1.5km$ and cutoff values $\kappa|D_1 = 400km$, and $\kappa|D_2 = 80km$.

The observed shape of $P(\Delta r)$ could be explained by three distinct hypotheses:

- A: Each individual follows a Levy trajectory with jump size distribution given by eqn. 1.
- B: The observed distribution captures a population based heterogeneity, corresponding to the inherent differences between individuals.
- C: A population based heterogeneity coexists with individual Levy trajectories, hence eqn. 1 represents a convolution of hypothesis A and B.

A thorough analysis of the movement patterns revealed that hypothesis C is in fact the most accurate explanation.

**Obstacle Mobility Model**

This is based on the following real-life observations [6]:

- First, people move towards specific destinations rather than randomly choosing some destinations.
- Second, there are obstacles in the real world. These obstacles, most commonly the buildings, block people's movements as well hinder signal-propagation.
- Third, people do not walk along random trajectories; they usually move along pathways and select shortest paths.

Take a campus scenario as an example. Buildings are modelled by placing rectangles of random size at random locations; pathways are constructed by Voronoi diagram of the vertices of these rectangles; doorways are the intersections of these pathways with the buildings. A Voronoi diagram is a partitioning of a plane with $n$ points into $n$ convex polygons such that each polygon contains exactly one point and every point in a given polygon is closer to its central point than to any other. A host randomly chooses a building as its destination, moves towards it, pauses sometime, and then moves on to another building. To reach a destination, the host can only move along pathways, although it may cross buildings through doorways. Among all these pathways, the host selects the shortest path. The model also considers the signal blocking problem. The communication of a host with other hosts will be totally blocked by buildings if the transmission is out of its *Line-Of-Sight*.

Object locations and connecting pathways are computed once at the beginning of the simulation and do not change during the course of the simulation. The initial placement of the mobile nodes is obtained by distributing the nodes at random locations along the pathways. Each node selects a destination location, such as a building entrance, and then moves to that location using the shortest route from its current location. Thus, after the selection of a destination, each node runs a shortest path computation on the graph created by the pathways to determine the path it will traverse, and then moves towards that destination using its computed pathway.

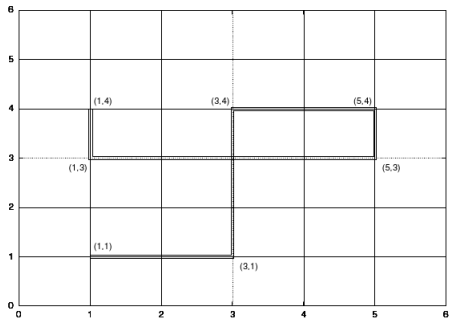
Upon reaching its destination, the node pauses for some rest period. It then selects a new destination point, cal- culates the path it will take to reach the new destination, and resumes movement. We point out that nodes can move through the buildings to reach their destinations; a shortest path between two locations can require going through a building, as is often the case in the real world.

**City Section Mobility Model**

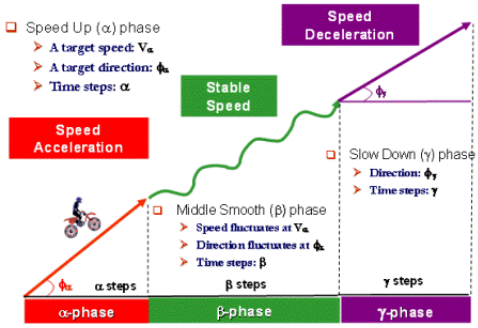
This model is very similar to the obstacle mobility model. In the City Section Mobility Model [5], the simulation area is a street network that represents a section of a city where the ad hoc network exists. The streets and speed limits on the streets are based on the type of city being simulated. For example, the streets may form a grid in the downtown area of the city with a high-speed highway near the border of the simulation area to represent a loop around the city. Each MN begins the simulation at a defined point on some street. An MN then randomly chooses a destination, also represented by a point on some street. The movement algorithm from the current destination to the new destination locates a path corresponding to the shortest travel time between the two points; in addition, safe driving characteristics such as a speed limit and a minimum distance allowed between any two MNs exists. Upon reaching the destination, the MN pauses for a specified time and then randomly chooses another destination (i.e., a point on some street) and repeats the process. Fig. 2.7(a) shows the movements of an MN using an example city section in the City Section Mobility Model.

The City Section Mobility Model provides realistic movements for a section of a city since it severely restricts the traveling behaviour of MNs. In other words, all MNs must follow predefined paths and behaviour guidelines (e.g. traffic laws). In the real world, MNs do not have the ability to roam freely without regard to

obstacles and traffic regulations. In addition, people typically tend to travel in similar patterns when driving across town or walking across campus.




(a) Travelling Pattern of a MN using the City Section Mobility Model




(b) SMS Mobility Model

Figure 2.8: Other Useful Mobility Models

### SMS Mobility Model

The Semi-Markov Smooth (SMS) mobility model [6] is a novel mobility model in which each SMS movement contains a random number of equal-length time steps. According to the physical law of a smooth motion, a moving object would experience speed acceleration, stable speed and speed deceleration in one movement. And a temporal correlations exists during the velocity transition. Based on this observation, in each SMS movement, a node will randomly select a target direction $\phi_\alpha$ and a target speed $v_\alpha$ as the expected direction and speed of the movement. Each SMS movement contains three consecutive moving phases: Speed Up phase for even speed acceleration from 0 m/s to the target speed $v_\alpha$; Middle Smooth phase for maintaining stable velocities which respectively fluctuate around $v_\alpha$ and $\phi_\alpha$ in each time step; and Slow Down phase for even speed deceleration to 0 m/s. The node experiences a random pause time after each SMS movement.

The Semi-Markov Smooth (SMS) mobility model is designed to specify the movements of individual users. Because it complies with the physical law of smooth motion, SMS model always generates smooth movements. According to its stationary moving behaviours, the SMS model generates stable node speed without speed decay problem; and it maintains uniform node distribution all the time. By adjusting model parameters, the SMS model can be easily and flexibly

controlled to support various network scenarios. It also has the has the desired steady state properties which are essential for simulation of mobile-user networks.

**In-Office Behaviour Mobility Model**

Unlike other models like Random Way-Point model etc., this model [9] captures the real movements occuring in real life scenarios, like in an office. The base part of this model is Simulation Time Controller, on which the structure of offices and their behaviour pattern is defined. Studies show that expressions in this model always have less than 10% error. This model defines three typical patterns of heterogeneous behaviour namely entity movements (talk behaviour), group movements (meeting behaviour), and regular movements (captures periodical appearance in the same place).

Moving on to the description of the framework, this model uses a tree based structure to model the various departments of the office. Department is an important factor influencing the movements of the people in the office. It is defined recursively. A level $(i+1)$ department is constructed by level $i$ departments and an atomic department acting as the leader of this dept, however the level of the leader is promoted to $i$. A notion of level $i$ coworkers is also used that distinguishes various people from each other. If $A$ and $B$ are not leaders, and both work in the same level $i$ department, or if $A$ is a leader of a level $i$ department and $B$ is one of the leaders of the level $(i-1)$ department in it, then in both cases $A$ and $B$ are called level $i$ co-workers. It is practically observed that $A$ interacts more with $B$ in the former case than in the latter case. Now, a tree is build with the above information.

The model us divided into three modules, namely: structure, behaviour and statistical modules, each of which are controlled by the Simulation Time Controller. The Behaviours model considers the three types of movements mentioned above and deals with them separately. The Statistics Module is used to gather metrics we are interested in, such as hitting time, ratio of staying and duration of a talk or a meeting. Using this as the base, the model derives various theoretical expressions using concepts of probabilities and other techniques to evaluate statistical metrics.

The main advantage of this model is that its results matches with close proximity to the real MIT traces that are available. This framework can also be extended

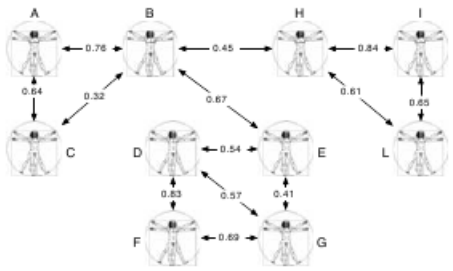



Figure 2.9: A Simple Social Network

to match the users needs and to incorporate more kinds of movements apart from the three mentioned in this model [10].

### 2.2.4 Social Network Based Mobility Models

Social networks are perhaps, one of the most defining aspects our life. We form social networks with our colleagues in the work place, friends from book clubs, our family members and with many others. A key characteristic of such social networks is the presence of clusters that are usually dependent on the relationships among the members of the social group. This notion of clusters, called *communities*, distinguishes social networks from any other types of random networks [12].

Social network mobility models are based on a simple observation. In mobile networks, devices are usually carried by humans, so the movement of such devices is necessarily based on human decisions and social behaviour. In order to capture this type of behaviour, mobility models dependent on the structure of the relationships among people carrying the devices have been defined.

#### Community Based Mobility Model

This mobility model [11] is based on the concept of social networks. A key input is the social network that links the individuals carrying the mobile devices in order to generate realistic synthetic network structures. The model allows collections of hosts to be grouped together in a way that is based on social relationships among the individuals. This grouping is then mapped to a topographical space, with topography biased by the strength of social ties. The movements of the hosts are also driven by the social relationships among them. The model also allows for the

definition of different types of relationships during a certain period of time (i.e. a day or a week). For instance, it might be important to be able to describe that in the morning and in the afternoon of weekdays, relationships at the workplace are more important than friendships and family ones, whereas the opposite is true during the evenings and weekends.

The following is a set of key features of this model:

- Model social networks as weighted graphs and provide them as input to the mobility model
- Identify communities and groups in the network and associate them to a geographical space. It is interesting to observe that people with strong social links are likely to be geographically colocated often or from time to time.
- Develop an algorithm that is at the basis of the dynamics of the nodes, that, again, is based on the strength of social relationships. Again, it can be argued that individuals with strong social relationships move towards (or within) the same geographical area.

The simulation area is in the form of a grid. A host is placed in a certain square to start off. Each square (i.e., place) exerts a certain *social attractivity* to a certain host. The social attractivity of a square is a measure of its importance in terms of the social relationships for the host taken into consideration. The social importance is calculated by evaluating the strength of the relationships with the hosts that are moving towards that particular square (i.e., with the hosts that have a current goal inside that particular square). Based on this value, there may be a deterministic or probabilistic selection of the next goal state of the host.

## 2.3 Trace-Based Mobility Models

In recent years, many researchers have tried to refine existing models in order to make them more realistic by exploiting the available mobility traces [14]. The key underlying idea of these models is the exploitation of available measurements such as connectivity logs to generate synthetic traces that are characterized by the same statistical properties of the real ones.

### 2.3.1 Some Important Studies

Various pioneering measurement studies have been conducted both in infrastructure-based and infrastructure-less environments since the first wireless networks have been deployed. Here are a few notable examples involving real-world traces.

- *Intel Research.* In collaboration with the Univeristy of Cambridge, Intel Research distributed Bluetooth devices in the university campus to collect data about human movements and study the characteristics of the co-location patterns among people. This was done with students at Cambridge and then participants of INFOCOMM, 2005.

- *Wireless Topology Discovery Project.* At the University of California, San Diego, 300 wireless PDAs, running Windows Pocket PC, were distributed among a select group of UCSD freshmen, to collect data on the dynamic characteristics of a real world wireless network.

- *CRAWDAD.* This was a workshop conducted by Dartmouth college [13] to obtain useful data from their Wirless LAN users. At Dartmouth, they have a campuswide wireless infrastructure, with comprehensive data-collection mechanisms to gather traces of wireless users and their behaviour. Several popular regions in the campus were identified, and the transitions of the users from one hotspot to another was characterized using Markov models. It was observed that the pause time and user speed followed log-normal distributions. They have also developed an extensive toolset for collecting, anonymizing, and analyzing the trace data. They have also started working on creating a repository of publicly available traces for the mobile networking community.

- *ETH, Zurich.* The researchers use a simulation area divided into squares and derive the probability of transitions between adjacent squares from the data of the access points. It was observed that the session duration data follow a power-law distribution. This is very similar to the Weighted Way-Point Model developed at USC. In this model, the probability of user movements between different areas of the USC campus are represented by means of a Markov model. The model is extracted from data collected from user surveys (i.e, the users were asked to keep a diary of their movements for one month).

- *Urban Pedestrian Flow.* This is a model derived from observations on the streets of downtown, Osaka in Japan. The authors reproduce the movements

of pedestrians by analyzing the characteristics of the crowd in subsequent instants of time and maps of the city using an empirical methodology, without relying on any wireless measurements. Based on the density of the streets in various time windows, pedestrian flow is characterized.

### 2.3.2 RealMobGen: A Campus Simulator

RealMobGen is a realistic university campus-based pedestrian mobility model [15]. It is based upon Dartmouth's model of mobile network traces and USC's Weighted WayPoint (WWP) model. The characteristics adapted from Dartmouth's model include direction of movement, speed distribution, initial location of nodes, pause time distributions, ratio of mobile nodes to stationary nodes, and start and stop time distributions for a node's active period. Other attributes of RealMobGen stem from USC's WWP model, i.e., hotspots and the time dependent probabilities of hotspot-to-hotspot transition decisions.

In RealMobGen, nodes come in two flavours: stationary and mobile. Stationary nodes are placed-and are restricted to the popular locations in the scenario called *hotspots*. Mobile nodes are allowed to start in any location within the simulation area. These mobile nodes pick a destination based upon a transition matrix. This matrix, which was adopted from USC's WWP model, is a time-variant set of probabilities that defines the chance of movement from $hotspotX$ to $hotspotY$. After a destination is chosen, nodes move to that destination via waypoints. Dartmouth's study was used for the implementation of waypoints.

Table 2.1: Parameters in RealMobGen

| Variable Name | Distribution |
|---|---|
| StationaryStartTime | Exponential Distribution |
| StationaryEndTime | Exponential Distribution |
| MobileStartTime | Exponential Distribution |
| MobileEndTime | Exponential Distribution |
| TransMatrix | Probability Weighted Matrix |
| Speed | Lognormal Distribution |
| PauseTime | Lognormal Distribution |

**Model Parameters** In RealMobGen, both the pause time and pedestrian speed distributions are log normal, following Dartmouth's findings. The start (stop)

**Algorithm 1** *RealMobGen*

**Input:** Scenario

```
1: for each Stationary Node do
2:     Select a hotspot location.
3:     Turn the node on at StationaryStartTime.
4:     Stay at this location until StationaryEndTime
       is reached.
5:     Turn stationary node off.
6: end for
7: for each Mobile Node do
8:     Select a start location anywhere within the
       simulation area.
9:     Turn the node on at MobileStartTime.
10:    repeat
11:        Select next destination based on the
           TransMatrix.
12:        Move to selected location via way-
           points.
13:        Pause at the destination for the allotted
           PauseTime.
14:    until MobileEndTime is reached
15:    Turn mobile node off.
16: end for
```

Figure 2.10: High Level Pseudo-Code for RealMobGen

time defines when a node will enter (leave) the simulation area and follows an exponential distribution in RealMobGen; these two distributions are also adopted from Dartmouth's study. See Table 1 for a list of variables in RealMobGen and their distributions. See Fig. 2.10 for high-level pseudocode of RealMobGen.

The input to RealMobGen is the scenario. The user can select from two pre-defined scenarios called the small campus and the large campus. The user can also construct a customized scenario. If the user chooses to customize a scenario, then he or she needs to provide the number of hotspots, area and flavour thereof, transitional probabilities, size of simulation area, and the ratio of mobile nodes to stationary nodes. Fig. 2.11 represents a sample visualization of the hotspots of a custom scenario generated using RealMobGen. Although this simulator is for actual wireless networks, the basic idea provides us with an useful insight into the process of designing a realistic simulator for modelling the behaviour of a mobile-user.

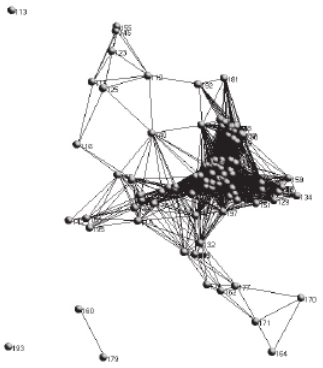


Figure 2.11: A custom scenario in RealMobGen (visualization using iNSpect)

### 2.3.3 Important Metrics

All the studies done by different research groups based on campus-wide wireless networks focus on two important metrics for analysis, namely:

- *Contact Duration.* It is defined as the time interval during which two devices are in radio range.
- *Inter-Contacts Time.* It is defined as the time interval between two contacts.

These indicators are particularly important in ad hoc networking and, in particular, in delay tolerant mobile ad hoc networks, since inter-contacts times define the frequency and the probability of being in contact with the recipient of a message or a potential message carrier in a given time period. Typically, the contacts duration follows a log-normal distribution while the inter-contacts time follows a power-law distribution. It has also been observed that the inter-contacts time distributions generated by means of classic random mobility models such as the Random Way-Point model show properties that can also be observed in real traces such as power-law behaviour in a certain range of values and an exponential tail after a characteristic time.

## 2.4 Analysis

The degree of realism in any simulator depends heavily on the mobility model used, i.e. the behaviour pattern of each user. Thus, it is crucial to define the mobility model efficiently and elegantly for any simulator design. In this chapter,

we have discussed several of the existing group and individual mobility models as well as a few state-of-the-art simulators. The trace-based models, which are more realistic, are constructed from actual collected data. But as mentioned before, these data are quite campus-specific and hence not easily generalizable. Further, it is hard to tune the parameters in such models, and thus repeated generation under different scenarios is difficult using these models. So, we focus on designing realistic synthetic models. Indeed, among the group mobility models, the Reference Point Group Mobility Model, when used with the right parameters can simulate several group activities. Similarly, the Random Waypoint Model with proper velocity and pause-time distributions can model simple individual behaviour. But often times, our behaviour, and correspondingly the movement patterns, are much more complicated. Our individual movements may follow a combination of several models during different periods of the day. Add to that the notion of social groups, it becomes increasingly more complex to capture human behaviour perfectly by a single model.

Almost all of the existing models focus either on generating individual behaviour or group behaviour emerging from a set of individual behaviours. None of the models capture the realistic combination of the activities of a person, in an individual as well as in a group sense. A single person can belong to several groups during different portions of the day. It is important to understand that out of the members of a group, only some people may be doing certain activity with the group, while the other members may be doing something alone, or as part of some other group. This spatio-temporal context is missing in the existing models. In addition, the notion of a group in many of the models is quite different from our social networks.

Consider the Reference Point Group Mobility Model. It captures the behaviour of users in a group by keeping track of a group-oriented motion and individual motion within the group. But this notion of group based on physical proximity is not really an example of a social group. For example, people travelling in a train fall into the category of groups as per the RPGM model. However, none of the people on the train may know each other. The only common thing between them is the fact that there destination and mode of transport are the same. This brings us to the concepts of planned behaviour and emergent behaviour. People on the train are an example of emergent behaviour. On the other hand, a social group, will follow planned behaviour: sites visited by the members, activities and events taken part in, are likely to be the same and these are pre-decided.

The social network based mobility models seem to capture this inherent notion of group. However, defining regions of the simulation grid based on social attractiveness, and deciding movement patterns based on them, seems a bit far-fetched. This model assumes that, for any individual, a social network exists at different grid locations. The strength of his ties decide his movement, either towards or away from them. But in real life, it often happens that the entire social network changes location along with the individual, say a family is going out for dinner. When moving with the group, the movement pattern will be decided based on what sites the group frequents, say a restaurant. This dynamism in group behaviour is not captured by this model. Also, defining the social networks in terms of weighted graphs requires actual data to specify the weights.

Given all these considerations, it becomes necessary to define user behaviour by an intuitive model rather than a model driven by any fixed mathematical equations. Keeping this in mind, we propose the framework of our simulator.

# Chapter 3

# Proposed Framework

The goal of this project was to design a realistic simulator that could generate readily avaliable and tunable spatio-temporal data corresponding to user movement patterns. As we discussed in the previous chapter, no single model can account for individual user behaviour in its entirety. Also, none of the existing models capture the intuitive notion of groups. So, we constructed an intuitive model for capturing user behaviour of different user classes. A user-class is a broad categorization of users in terms of commonality in behaviour. For example, almost all users in the class of office-goers will typically follow a common behaviour regime - go from home to office, from office to cafeteria, and back to home. In addition to this, each individual user will have his own set of deviations. But the overall picture remains the same for users in a single user class. In addition to the behaviour model for inter-location transition, we can also use some simple models like Random Way-Point Model to simulate individual user behaviour at a specific location, say in the office.

## 3.1 Terminology

The following terms come up quite frequently in the rest of the chapter and are described below.

- *Simulation Area.*The simulation depicts the behaviour of mobile users in a certain region known as the simulation area. Typically, this area consists of several locations, visited by the mobile users, with inter-connecting paths. The simulation area could be a rectangular grid with randomly generated hotspots or it could be a realistic portion of a city, obtained from Google Maps.

- *User/Actor.* This refers to the individual whose movement patterns we are trying to simulate as spatio-temporal data. Every actor belongs to some user-class.

- *User-class.* This refers to the broad categorization of users into various classes based on commonality in behaviour patterns. For example, all college students follow a similar behaviour regimen, i.e. the activities performed, locations visited in a logical representation are more or less the same, and hence fall into one class. Similarly, we have classes for office-goers, sports players, etc.

- *Hotspots.*Certain locations in the simualtion grid have high node densities at various points during the day. Such locations are obviously popular locations that are visited frequently by the mobile users. In the simulation jargon, these are called as *hotspots*. A hotspot may be an office building, a shopping mall or even a popular theatre. The actors, i.e. the mobile users, tend to move from one hotspot to another throughout the duration of the day. Usually, this behaviour is goal-driven. If a mobile user $X$ moves from Hotspot $A$ to Hotspot $B$, it is because he has some interest in the features of the hotspot $B$. It is this movement that we try to capture and represent in our simulation efforts.

- *Group.*A set of users who perform some activities together several times during a week. The concept of co-location is a necessary but not sufficient condition for defining groups. Thus, we have the notion of planned groups and emergent groups. A set of colleagues going to watch a movie together constitute a group, whereas the other people in the theatre are not a part of their group.

- *Event.* Any activity at a location that causes several users to turn up and stay for a certain thresholded time period is said to be an event. That location is termed as the venue of the event. For example, a cricket match is an event with the cricket stadium being the venue.

## 3.2 Setting up the Problem

The task of generating individual and group behaviour can be divided into five interesting problems as follows:

1. *Modelling Individual User Behaviour from User class Specifications.* In this problem, the broad set of specifications for a user-class, in terms of inter-location transition probabilities as a function of time, departure time distributions, and transition velocity distributions are provided. We then construct a simple timeline representation of all the possible sequence of activities for the user-class. Each timeline will have an associated probability of being chosen as computed from the input distributions and probabilities. We pick a day, and for every user in the user-class, simulate his behaviour by probabilistically picking all or some of the activities from the available sequences of activities in his user-class. Repeat this for the entire simulation period making sure that any specified constraints are not violated.

2. *Modelling Group Behaviour of an Individual from Group Specifications.* In this version, the specifications are provided for different types of generic group behaviour, in terms of sets of locations visited, and combinations of activities performed by the members of the group, along with their corresponding probabilities. For example, we may have a set of specifications corresponding to restaurant-going behaviour for the class of office goers. As before, we can construct a timeline representation of the activity sequences. Our next task is to assign users to all the different groups. Then, we instantiate each of the activity locations with an actual physical hotspot to obtain an instantiated group. For the simulation, we pick a day, and probabilistically choose a set of members of the group who are going to perform a particular sequence of group activities. This choice of activities is also chosen with a probability, from the set of various possible activity sequences. Repeat this for the entire simulation period making sure that none of the group constraints are violated.

3. *Combined User Behaviour from User-class and Group Specifications.* In this problem, we want to generate both individual behaviour as well as group behaviour of a particular user. The idea here is to simulate the individual behaviour and group behaviour simulataneously, and then combine both the activity sequences, i.e. in between his usual individual activities, the user also performs some of his group activities, with different groups throughout the duration of the day. It is important to understand that even if the individual and group behaviour generated, satisfy the specification constraints independently, we may still have cases where we have clashes between individual and group activities. In such a case, we decide to override one activity over the other probabilistically. These overriding probability values are fixed based on the priority of individual and group behaviour at different

periods of the day.

4. *Validating the Specifications.* Given the individual user-class and group specifications, we saw how to generate a combination of individual and group behaviour systematically. But a lot of probabilistic choices are involved in terms of chosing activity sequences as well as resolving clashes by overriding. Thus, it is essential to verify if the data generated satisifes the input specification constraints as a whole. So, after generating the user behaviour for the entire gamut of users in a particular user-class, we combine the data generated and try to generate back the original user-class and group specifications. This will tell us, if any activity sequence has been over-chosen or under-chosen, i.e. if any of the original specifications have been drastically violated.

5. *Checking Consistency of Specifications.* This problem is quite similar to that of the validation problem. But in this, we do not actually generate the behaviour. Given a set of individual user-class and group specifications, we have to perform consistency checks and verify whether the given set of constraints is consistent. If not, we try to adjust the probabilities so as to make the specifications consistent, thereby making the behaviour generation procedure smoother. Along this same line of thought, if we certain parts of the specifications missing, we can fill them by performing consistency checks to obtain a set of valid choices for the empty probabilities.

## 3.3 Modelling User Behaviour

The typical behaviour of any two mobile users may be quite independent and varied. One user may keep moving between hotspots, spending very little time at each location, while another may have limited motion and stay at certain hotspots for long periods of time. Clearly, it is a challenging task to encompass the behaviours of each individual in the simulation region. That having been said, not all users are completely independent. More often than not, there is some common uniting factor. Usually, it is their employment class, common hobbies and interests, or even any general social network for that matter. So, office-goers, school students, salesmen, and doctors form different classes of mobile users.

Classifying users into several classes, we can analyze certain behaviours common to all the actors in a class. For each class, we divide their activities into three broad category of behaviours, namely:

- ***Regular Behaviour***

  This is the most likely behaviour of any mobile user, which he follows for most parts of the week. Going from home to office on weekdays for the class of office-goers, housewives staying at home, students going from hostel to college, constitute some typical examples. It may be noted here that for any individual, even weekends may have certain regular behaviour like going to a specific restaurant every Saturday. Any regular behaviour will be associated with specific locations, characterized by certain features, such as pathways leading up to those locations, behaviour of an actor at those locations, duration for which he stays at such locations. It is these markers that we look to exploit in our simulation models.

- ***Irregular Behaviour***

  Any behaviour that is a deviation from the regular behaviour falls into this category. Typically, this irregular behaviour is what that distinguishes one user from another in a class of users. For example, consider the prototype office-goer with a 9AM to 5PM job. In order to break the montony of his work, he chooses to take leave on some afternoon and go see a movie. Or he chooses to go to a fancy restaurant for lunch rather than the usual office food court. Or he may choose to skip work all together for a day. All these are examples of irregular behaviour - behaviour that is not routine but is likely happen. Similarly, activities performed with his friends, like going to a theatre to watch a movie, or hiking, are also part of irregular behaviour. Thus, we see that the social group behaviour of a person can significantly influence the irregular behaviour.

- ***Event-driven Occasional Behaviour***

  On certain special occasions, the behaviour of an individual may differ from his regular behaviour. This is different from irregular behaviour in the sense that there is a certain event/happening that actuates the specific behaviour of the individual, rather than random choice. Again, this behaviour may be closely linked to his social groups. For example, consider the case when a cricket match between India and Australia is taking place in the local stadium on a certain weekday. A cricket lover may choose to watch the match. So, he departs from his regular behaviour by not going to the

office. This departure from the norm is marked by a choice of a different route, maybe even a different means of transportation, or choosing a different group of people to hang out with (social groups come into the picture). It is evident that the change is behaviour has been brought about by the cricket match, and we say that this behaviour is event-driven and occasional.

## 3.4 Case Study: The Class of Office-goers

Our simulation process consists of modelling the behaviour for different classes of users. Let us start off with the different types of behaviour of our quintessential office-goer. This is an important class of users with over 20% of the world population falling into this category. As discussed before, we may classify his behaviours into Regular, Irregular or Event-Driven behaviours. We may also choose to distinguish between weekday and weekend behaviours.

### 3.4.1 Regular Behaviour

Although the regular behaviour may vary from person to person in the class of office-goers, there is a certain degree of commonality in their activities. Listed below are a few of those possible common behaviours.

1. *Going to Office.* The most common behaviour among almost all office-goers on a weekday is going from home to office in the morning. This activity has certain specific features. The time to report at the office is usually between 9 AM - 11 AM. Accordingly, the start time from the home is determined. He usually chooses the shortest, least-crowded roads to reach his office. So there is a pre-determined daily route for reaching the office. A more detailed analysis of the route may reveal certain traffic signals where he spends a fixed amount of time daily. Maybe, there is a temple on the way, at which he stops for a certain period of time. Or he may even stop at a certain hotspot, say a posh apartment to pick up a colleague. Taking into account all the minute details, one may compute a more accurate office reaching time, which should lie within a small time window.

2. *At Office.* After a person reaches his office, he proceeds to his cubicle or room. Again, he takes a fixed path, maybe the elevator, to reach his room, say, on the fifth floor. Once in his room, he may have to visit the office of his boss on a different floor for a daily meeting. During break time, he may go out to the roof for a few minutes. During lunch, he may go to the

office cafe for food. In the afternoon, he may join his colleagues ina light chat session. All these behaviours may be habitual, and may have inherent movement patterns associated with them. We can use the Random Way-Point mobility model with appropriate pause time distributions to simulate in-office behaviour.

3. *Going back Home.* At the end of the office hours, which are usually fixed, the office-goer again chooses a particular means of transport to go back home. He may choose a different route with different characteristics. Again this may be different from the coming to office behaviour. However, all the temporal parameters that were considered for the coming to office behaviour need to be considered for the going back home behaviour.

4. *At Home.* Once he reaches home, he may go for his usual jog in the nearby park at 6 PM. Or he may go to the local tea-stall to hang out with a few neighbours. Or he may simply choose to stay at home and watch TV. These relatively common behaviours can be mapped as regular behaviour.

5. *Weekends.* It is very difficult to classify behaviour on weekends as regular behaviour. Some users may choose to go to the office while others may choose to simply stay at home or go for an outing. Then again, among those who go to the office, there may not be any fixed arrival and departure times from the office. Someone may choose to visit a barber for a haircut. Clearly, it is better to capture this in terms of irregular behaviour.

### 3.4.2 Irregular Behaviour

The irregular behaviours of an office-goer vary widely throughout the week. Some behaviours may keep him in the simualtion area while others may even take him outside (going on a fishing trip). Note that this behaviour can vary widely from one office-goer to another. Let us examine a few common irregular behaviours among office-goers.

1. *Going to Office.* The office-goer may choose to start early and take a different route to office. This route will have different markers such as less/more number of traffic signals, different levels of traffic congestion, different distance to the office and hence, will also affect the arrival time at the office. He may also choose to stop in-between to pick up a few stationary items. Or he may have to drop off his kids in their school because they missed their bus. Then again, he may choose not to go to the office altogether if he is sick. All these are common-place irregular behaviour.

2. *At Office.* He may have a seminar to attend in the office on a floor different from where he usually works. The seminar may have arrangements for lunch, which would preclude him from visiting the food-court. He may have to go to the work-site to supervise the ongoing work. Or he may leave early to keep an appointment with a doctor. Again, all these activities would have their own movement patterns as well as time distributions.

3. *Going back Home.* On his way from office to home, he may visit the supermarket to buy certain urgently needed commodities. Or he may go to the local fish market, to get fish to cook for dinner. Or he may again have to go to pick up his children from the school. These will definitely lead to a deviation in movement patterns.

4. *At Home.* Once he reaches home, he may choose not to go out for his jog. Rather, he may rent a movie to watch at night. Or he may go to the movie theatre to enjoy a new movie. He may have to go out to attend a wedding reception. Or he may choose to take his family out for dinner at a nearby restaurant. Clearly, these will have low likelihoods and are difficult to generalize for the entire class.

5. *Weekends.* He may wish to take his family out on a vacation to a hill-station. Or he may simply visit a local hangout spot. He may be interested in organizing a social awareness campaign in the nearby village. He may also have to attend the monthly colony meeting. Or clean the garden with his children as a family project. He may have to go to the office and work late to meet an upcoming project deadline. All these occur without any fixed pattern and comprise the irregular behaviours of a typical office-goer.

### 3.4.3 Event-Driven Behaviour

The typical office-goer will have certain occasional behaviour in response to a certain event happening in his locality. The event maybe a dance competition, a cricket match, a political rally or even a survey organized by his office. Opening of a new restaurant or screening of a blockbuster movie also fall into this category. In all these cases, the user will deviate from his daily routine, and we would obtain completely distinct movement patterns and spatio-temporal distributions. It is important to note that this behaviour is occasional and highly specific from person to person. For example, if a person is interested in painting, only then he may wish to see an art exhibition in the museum. Clearly, this sort of behaviour cannot be categorized as regular or irregular behaviour. Another point worth mentioning

here is that in order to capture the event-driven behaviour, we must be able to simulate these events with as much detail as possible. Thus, algorithms for event generation are also an important aspect of the simulation process.

**Group Behaviour.** A further driving factor for this sort of behaviour is the individual's social network based on commonality of interests. What this means is that event-driven behaviours are typically manifested by closely knit social groups of people. In the class of office-goers, there may be 10 people belonging to one office, who are very interested in reading novels. A majority of these 10 people would preferrably go together to the launch event of a new book, depicitng their group behaviour. But then again, there may be individuals who wish to ignore the event altogether or may want to go on their own. A similar example might be a 2-day business workshop in the city. Typically, people working in a office tend to attend these workshops as a group. So, their individual behaviour is governed by the group dynamics. But, the group behaviour, by no means guarantees a fixed type of behaviour in response to a event, rather it represents a more likely set of behaviours. The underlying thread responsible for the group behaviour may be similarity in social interests, hobbies, specific skill-sets, or even simple co-location (people of a certain neighbourhood form a group).

All things said and done, the behaviours of different individuals will vary widely if we go into the intrinsic details of movement patterns and spatio-temporal information. Nevertheless, similarity in behaviour exists, which is visible at a higher layer of abstraction, and provides us with the starting point in our quest for designing an exhaustive but elegant simulator. Also, the above analysis is only for the class of office-goers. The behaviour of other classes have to be carefully analyzed and incorporated into the simulator to capture the behaviour of a wide range of individuals in a simulation region. Keeping these few basic ideas in mind, we design the formal specifications of the simulator for mimicking the network of mobile-users.

## 3.5 Modelling Individual Behaviour

In order to ensure smooth implementation, it is better to have a formal structure for the specifications of the simulator. This involves defining the various parameters that characterize the movement patterns of an individual mobile user, viz., his speed, acceleration, direction of motion, time distribution of departure and arrival at certain locations, inter-location transition probabilities and so on.

It is possible to define a finite-state automaton that captures the broad behaviour of a class of users.

### 3.5.1 The Finite State Automaton

The Finite State automaton is a very effective tool for depicting the generic behaviour of a class of mobile users. Formally, it can be defined as:

$M = <S, T_D, T_P, V_D, \delta>$

where:

- $S$ - the set of states defining the user's location. $S = RB_S \cup IB_S \cup OB_S$ with $RB_S$ depicting the set of regular behaviour states, $IB_S$ depicting the set of irregular behaviour states, and $OB_S$ depicting the set of event-driven occasional behaviour states.
- $T_D$ - the set of departure time distributions at various states.
- $T_P$ - the set of state transition probabilities
- $V_D$ - the average speed distribution along the state transitions
- $\delta$ - the state transition function : $\delta : S \times T_D \times T_P \times D_V \rightarrow S$

The arrival times can be easily computed from the departure time of the previous state and the velocity of transition. We may also choose to maintain velocity constraints in different segments of the map so as to model traffic on the roads. The most common start and end states are the home/residence hotspots. All the detailed distributions mentioned above are obtained from the specifications provided as an input to the simulator.

The states of the FSM are as follows:

$S_1$ = Home
$S_2$ = Office
$S_3$ = Food Court
$S_4$ = Shopping Mall
$S_5$ = Worksite
$S_6$ = Cricket Stadium

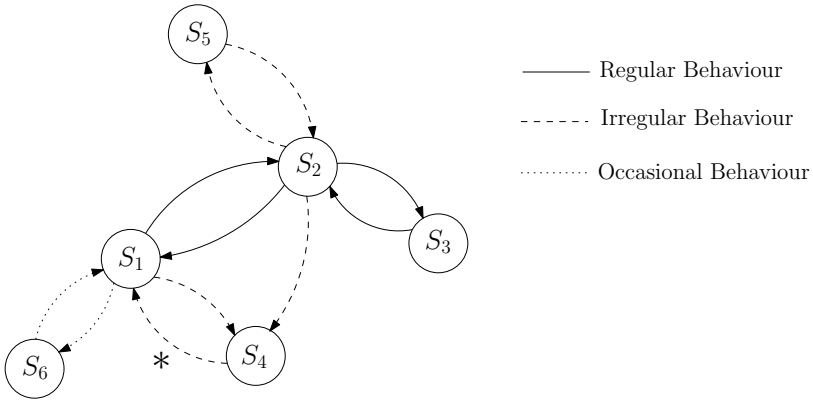



Figure 3.1: A Sample FSM: Office Goer

### 3.5.2 Simulation Area

The simulation area is the region on which the data is generated. It could be a randomly generated rectangular region or an actual city section obtained using Google Maps. In this project, we performed the simulation on an IIT Kharagpur campus map. For convenience it is preferable to overlay a grid on the region. The granularity of the grid may be specified depending on the extent of accuracy needed in the simulation process. The simulation area consists of many hotspots, marked by certain specific $<X, Y>$ coordinates, which represent the places of high node densities during the day. Typical hotspots in a map are home apartments, office buildings, markets, restaurants, playgrounds, cafeterias, theatres and so on. Transitions can happen between a normal grid location and a hotspot, between two hotspots or even between two ordinary grid locations, by following roads on the map. Apart from hotspots and roads, there are several regions on the map that are inaccessible during vehicular transportation, which we call as *blocked* regions. Thus, each grid on the map is of three types:

- Road grid
- Hotspot grid
- Blocked grid

### 3.5.3 Input Specifications

To model the individual user behaviour, we use the intuitive behaviour model described in the beginning of the chapter. The basic inputs to the simulator are


Figure 3.2: IIT Kharagpur Campus Map

as follows:

1. Simulation Area (preferably a map)
    - Types of hotspots associated with the simulation region (eg.: restaurants, offices, theatres, etc.)
    - Road networks
    - Accessible and inaccessible areas
2. Actor Information
    - The actor-class (e.g. : office-goers, students etc.)
    - Total number of actors in the class to be simulated
    - Probabilities for modes of transportation
3. Useful Distributions
    - Departure time distributions from each state
    - State transition probabilities
    - Average speed distributions for different modes of transportation
    - Velocity distributions associated with different sections of the roads (traffic modelling)

**Input Details**

**States.** Each of the states of the automaton represents a location for the mobile user. Locations can be classified based on the type of movements corresponding to the three broad categories of behaviours - regular, irregular, and occasional. The $<X, Y>$ coordinates define the state on the simulation grid. Apart from this the state is associated with several properties, such as departure time from that state, probability of staying in that state or undergoing a transition, and so on. It is necessary to understand that the automaton depicts a general category of behaviour for the entire class. Obviously, the movement pattern of each individual user will be quite different. The state *home* for office-goer $X$ will be different from that of office-goer $Y$, as will be their modes of transportation and states corresponding to their offices. These choices are obtained by scrupulously following the distributions constructed from the user-defined specifications.

**Transitions.** The transition from one state to another is the basic movement pattern that we want to generate. Each transition arc is associated with three properties, namely:

- $p_{ij}$ - the transition probability from state $S_i$ to state $S_j$
- $t_d$ - the departure time from the state at which the transition will take place
- $v_d$ - the average speed distribution along the arc for a particular mode of transportation

These parameters can be easily obtained from the appropriate distributions. Other details such as arrival time at a state can be obtained from the actual departure time from the previous state and the velocity chosen for that particular transition.

### 3.5.4 A Simple Representation

Given the specifications and the inputs to the simulator, it is quite easy to generate spatio-temporal data. We designed a simple logical representation to capture the user-class behaviour from the input details. In this representation, we plot the various behaviours of the user-class on a timeline. Each sequence of behaviours has an associated probability range, based on which an individual user behaviour is chosen. Fig. 3.3 is a simple version of the representation. Note that we can infinitely many such timelines for any user-class. To prevent this, we

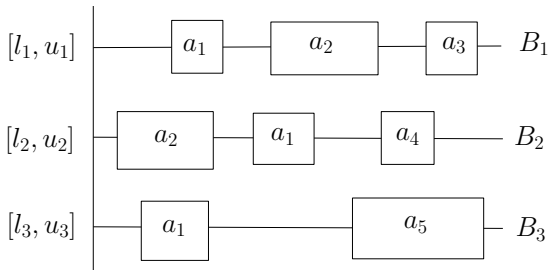



Figure 3.3: Timeline Representation: A - activity set, B - behaviour set

only choose those activity combinations whose probabilities lie above a certain threshold.

The three timelines in Fig. 3.3 represent three behaviour sequences. The activity set is actually a set of hotspot locations such as home, office, restaurant, playground, etc.The $[l_i, u_i]$ values define a range which tunes the probability of a timeline being chosen. These values are computed from the input specifications taking into account any necessary consistency checks. For every user, we first fix a value $x_i$ for each timeline, and then choose the behaviour sequence probabilistically. For any single user-class, several such timelines corresponding to different behaviour sequences exist. While deciding the timelines, we have to consider the velocity of transportation as well so as to provide the user with sufficient transit time. So, the departure time and velocity probability distributions of the input specifications come to our rescue. The following algorithm is followed in order to generate individual user behaviour.

**Algorithm Idea**

The simulation period and the frequency of data generation are provided as input specifications. Let $n$ be the number of actors in the user class, and let $b$ be the number of behaviour lines (with probabilities higher than a threshold).

1. Pick a day $d_i$.

2. For each $u \in U_i$, generate $(x_1, x_2...x_b)$ from the probability ranges of the timelines. Of course, the sum of these values: $x_1 + x_2 + ... + x_b \leq 1$. We have obtained the set of hotspots to be visited by the user during the day.

3. Simulate the chosen behaviour on the map. Use any path-finding algorithm to find paths between activity locations (hotspots) to generate the data. Utilize the velocity distributions to transit between states.

4. Repeat for the remainder of the simulation period.

After choosing the behaviour sequences for each of the $n$ users, we then compile the chosen behaviour, to obtain the user-class behaviour generated. Next, we compare this against the original input specifications to verify whether the generated behaviour actually satisfies the specified constraints. This forms the validation technique for our simulation.

## 3.6 Modelling Group Behaviour

The concept of groups is central to human behaviour. Any single individual belongs to several groups. We perform activities with different groups during different portions of the day. Most of the time the group activities are a result of some planning among the members. Thus, it becomes essential to distinguish between planned group behaviour and random emergent group behaviour. The existing group models capture the movement of the groups as a whole. But often times, it is possible that only some of the group members are doing a certain activity, while other members are acting individually or as part of some other group at that point in time. In this project, we try to capture not only the movement of an individual along with the group but also the movement of the individual in response to a planned group activity. So, when an individual goes to eat at the restaurant alone, it will be a part of his individual behaviour. But when the same individual goes to eat at the restaurant where his friends would join him, it will be a part of his group behaviour.

### 3.6.1 Input Specifications

The input specifications for the group behaviour need to be given separate from the user-class specifications. While specifying the group behaviour, we do not specify the complete behaviour of each group in the network. Rather, we specify generic group behaviour for certain activity types. For example, instead of specifying the exact details of a group going to a movie theatre followed by a restaurant visit, we specify a restaurant-going behaviour. This generic behaviour will have combinations of different behaviour types based on when restaurants are visited during the day, whether a movie follows a dinner or it is the other way

round. Probabilities will be assigned to each behaviour type which will allow us to choose a specific behaviour for a particular group. Each group will have a list of generic behaviours it performs along with associated probabilities. Further, we can classify groups based on associations with different people in our life. Thus, we have groups corresponding to family members, colleagues, hostel-mates and so on.

The group input details are as follows:

- Types of groups (hostel-mates, colleagues, etc.)
- Number of such groups
- Number of people in each group type
- Generic Group Behaviour types (consists of possible activity sequences)
- Probabilities of each group type picking a generic group behaviour
- Probability of choosing each activity sequence
- Overriding probabilities for different days for different pairs of groups

In addition, we may also have number of groups each person belongs to. The first three specifications are needed for assigning the people in a particular user-class to different groups. Apart from this, we also have group velocity distributions for transitions from one hotspot to another. Instead of specifying the various timelines, we can also specify the inter-state departure and transition probabilities from which we can compute the activity sequences.

### 3.6.2 A Simple Representation

Given these input details, our first task is to assign people to groups. This can be modelled as a simple satisfiability (SAT) problem and we can obtain a valid assignment. Or we may use a simple greedy algorithm to assign people to groups. Next, we can construct a simple timeline representation for each generic group behaviour. Note that the activity locations will be only the hotspots where the group meets together, i.e. home or hostel of an individual does not feature in this representation as these are low-probability group activity locations. We are simply trying to capture behaviour of users from one group-favoured hotspot to another. The movement from home/office to these hotspots and back will be incorporated when we combine both the individual and group behaviours.

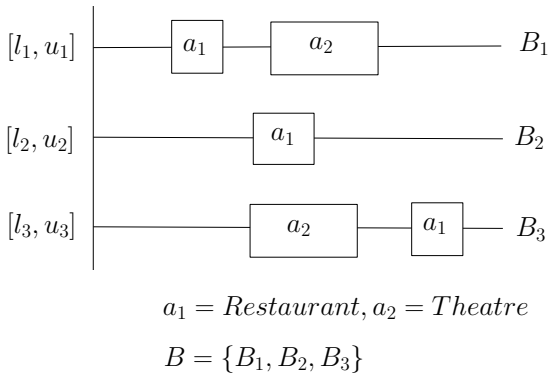



Figure 3.4: Timeline Representation: Generic group behaviour

Once we obtain the generic group behaviour sequence, we have to instantiate it for specific groups. This involves assigning actual physical hotspots to the logical ones. The movement from one hotspot to another can be governed by taking group departure and transition probabilities as explicit inputs. Or we can simulate the data for one member of the group taking part in the activity, and force others to follow the same transition values. In the second approach, we have to take care so that we don't violate any individual constraints.

Consider Fig. 3.4. It represents a generic restaurant-going weekend behaviour with colleagues. The three timelines correspond to three possible behaviour sequences, namely:

- Eating at a restaurant followed by watching a movie in the theatre
- Eating at a restaurant
- Watching a movie in the theatre followed by eating at a restaurant

The probability ranges are fixed as per the group behaviour specifications. As it can been seen, the time of visiting the restaurant or the theatre varies with each behaviour chosen. At the same time, even the time spent at each hotspot may vary with each possible instantiation. Note that the choice of each of these generic behaviour types will depend on the group type that performs the activities. For example, the probability of choosing a playground going behaviour will be much higher for hostel-mates than say colleagues. The following simple algorithm is used to generate group behaviour.

**Algorithm Idea**

The simulation period and the frequency of data generation are provided as input specifications. Also, the percentage of the time, for which the user decides on group activities may also be provided. Let $g$ be the number of actors in the group $G$, and let $b$ be the number of behaviour lines for a particular generic behaviour.

1. Pick a day $d_i$.
2. For the group $G$, pick a generic behaviour using the specified probabilities. Let, the probability of choosing a behaviour $B$ is $p$. The probability ranges of picking the various behaviour lines for this particular generic behaviour type are normalized w.r.t this value of $p$.
3. For each $u \in G_i$, generate $(x_1, x_2...x_b)$ from the probability ranges of the timelines. Of course, the sum of these values: $x_1 + x_2 + ... + x_b \leq 1$.
4. Instantiate each activity location $a_i \in$ hspot-type($i$) to some specific hotspot of that type. These steps generate an instantiated group from a generic group.
5. Simulate the chosen group behaviour on the map. Either simulate for one user and force the rest of the group members or use the group specified distributions. Use any path-finding algorithm to find paths between activity locations (hotspots) to generate the data.
6. Repeat for the remainder of the simulation period.

After choosing the behaviour sequences for each of the $g$ users, we then compile the chosen behaviour, to obtain the generic group behaviour generated. Next, we compare this against the original group specifications to verify whether the generated behaviour actually satisfies the specified constraints. Again after generating all the group behaviours perform a similar check to verify the validity of the overall group simulation.

### 3.6.3 Resolving Clashes

As described before, each individual can belong to several groups. Also, on a single day, he may do activities with several of his groups. While the group generation algorithm takes care of consistencies within the group, we have to also ensure that the generated behaviour timelines are actually consistent on a higher

level of abstraction, i.e. for all the group activities done during the day. It is quite possible that clashes arise between activities of two different groups for a single user. This means that the user can do only one of the clashed activities. To do this, we use the concept of overriding probabilities. These enable us to choose between the clashes and pin-point the user to a single activity at a time. The values of these probabilities are fixed based on group priorities during different days of a week. Thus, the validation procedure becomes all the more important since it is more likely now that we might lose our desired group behaviour.

## 3.7 Combining Individual and Group Behaviour

Now that we know how to generate individual behaviour from user-class specifications and group behaviour from generic group specifications, we turn our attention to the important issue of combining the two sets of behaviour sequences. The behaviour of a particular user will be a sophisticated mix of individual activities and group activities. For this purpose, we would prefer to know the percentage of time for which a user chooses individual behaviour and for which he chooses group behaviour. Note that these values could vary from day to day. For example, a user may follow individual regular behaviour during weekdays in majority while group behaviour may be of higher priority during weekends. Given the distribution of these two types of behaviours, we need to generate both behaviours independently and then combine them in a single timeline. Since the group behaviour specifications do not comprise the regular behaviour states such as home or hostel, we generate both kinds of behaviour in the following sequence:

1. Movement from an individual regular behaviour state to the first group behaviour hotspot by following the individual distributions. (keep in mind the arrival time requirements at the group hotspot)

2. Movement along with group members from one hotspot to another by following the group velocity distributions and the selected activity sequence, till the last hotspot

3. Movement from the last group hotspot to an individual behaviour state by again following the individual transiton/velocity distributions.

**Algorithm Idea**

Let there be $n$ users: $\{u_1, u_2, ...u_n\}$, and $m$ groups: $\{G_1, G_2, ...G_m\}$. Let there be $q$ user classes: $\{U_1, U_2, ...U_q\}$ and $t_i$ group types, $1 \leq i \leq q$, for each of

the user-classes. Each of the group type of a class $c$ has $n_j$ such groups, $1 \leq j \leq t_c$, and each such group is of size $g_{n_j}$. Let there be $k$ generic group behaviour types: $\{B_1, B_2...B_k\}$, and each generic behaviour type $B_i$ has $l_i$ activity sequences: $b_{i1}, b_{i2}, ...b_{il_i}$. In addition the user class specification for $U_c$ has $b_c$ behaviour lines. We have the split between individual and group behaviour durations, as well as the probability distributions for choosing generic group activities. The first task is to assign users to groups while satisfying the requirements of maximum group size for different types of groups. We may sometimes impose an additional constraint that the group membership of each user is restricted to the range $[min, max]$ groups. The following is a simple idea of the algorithm used.

1. Pick a day $d_i$.
2. For each $u \in U_c$, generate $(x_1, x_2...x_{b_c})$ from the probability ranges of the timelines. Of course, the sum of these values: $x_1 + x_2 + ... + x_{b_c} \leq 1$.
3. Repeat for each user class.
4. For the group $G_i$, pick a generic behaviour $B_j$ using the specified probabilities.
5. For each $u \in G_i$, generate $x_{b_{i1}}, x_{b_{i2}}, ...x_{b_{il_i}}$ from the probability ranges of the timelines. Of course, the sum of these values: $x_{b_{i1}} + x_{b_{i2}} + ...x_{b_{il_i}} \leq 1$ (normalised w.r.t the probability of choosing the generic behaviour in the first place.
6. Instantiate each activity location $a_i \in$ hspot-type$(i)$ to some specific hotspot of that type. These steps generate an instantiated group from a generic group.
7. For each user $u$, compare his generated individual behaviour timeline as well as the generated group behaviour timelines.
8. Resolve any clashes in activity locations by using specified overriding probabilities for group $G_i$ vs group $G_j$ clashes, as well as group $G$ vs individual behaviour $b$ clashes. Construct the overall timeline comprising activity locations corresponding to valid individual as well as group behaviour.
9. Simulate the chosen behaviour on the map. Use any path-finding algorithm to find paths between activity locations (hotspots) to generate the data. Again, as before enforce the group movements by using either the provided group transition ditributions or by making members of a group follow a

single user of that group. (Note that the consistency checks get more complicated if we combine several user classes in a single group)

10. Repeat for the remainder of the simulation period.

### Validation

The final task after generating the combined data is to validate it. For this purpose, we pool the generated timelines of all users and extract back the user-class specifications and the various generic group specifications. Next, we compare the obtained sets of specifications with the original specifications and decide whether the data generated was indeed aligned with the input specifications. For better quality validation, we need to perform experiments, tuning the probabilities and activity durations, to see what combination of individual and group user specifications, lead to the most consistent data generation. Using a similar approach, we can also verify whether a given set of specifications is consistent. If not, then we can provide a set of possible modifications that would make the specifications consistent. The validation mechanism for the combining individual and group behaviour is under progress.

### Interesting Statistics

We also perform an analysis of the generated data and come up with statistics such as:

- Average time spent by each user type at different hotspots during a day.
- Busiest road of the region and its traffic distribution for different periods of the day.
- Busiest hotspot in the region and its user distribution throughout the day.

If we have some real-world data, then we can compare these statistics to verify the extent to which our model is applicable to any generic scenario. These interesting data give us a clear-cut idea about the simulation region, and can be used in a lot of applications such as controlling traffic, managing pedestrians, etc.

# Chapter 4

# Implementation and Results

In the previous chapter, we established the proposed framework for the simulator. We discussed algorithms for generating data corresponding to individual as well as group behaviour, given the user-class specifications and generic group specifications. In this chapter we take a look at the implementation of the simulator. To start off, we focussed on generating the individual behaviour data and represent the movement patterns on the IIT Kharagpur campus map. The group behaviour was also generated for a few generic groups. However, the process is still under evaluation and detailed results on this topic will be furnished during the final presentation. Also, the validation process is currently under progress.

## 4.1 Simulator: Basic Components

The task of generating spatio-temporal data from the input specifications goes through the following steps:

1. Parse the input specifications file to obtain the necessary probabilities and transition distributions for both individual and group behaviours.
2. Use image processing to overlay a grid on the campus map obtained from the Google Static Maps API.
3. Use the connected components algorithm to identify the hotspots on the map. Also classify the hotspots into different types from the map details. This gives us the simulation area.
4. Generate the set of activity locations corresponding to a behaviour sequence for every user corresponding to individual and group behaviours. (If required, combine both sets of behaviours using overriding probabilities.)

5. Use a path-finding algorithm like $A^*$ to find movement paths between hotspots.

6. Obtain the spatio-temporal data in the form $\langle x, y, t \rangle$ at a frequency specified as input.

7. Perform validation of the data obtained to verify its consistency with input specifications.

Fig. 4.1 represents a high-level flow diagram for the individual behaviour generation by the simulator.

## 4.2 Simulation Area

The simulation area is the region in which the simulation is carried out. We can use either a randomly generated synthetic map or an actual city section obtained from any standard map-making API. For our implementation, we used the IIT Kharagpur campus map obtained using the Google Static Maps API. The map has clear classification of hotspots, roads and blocked regions. Given the map, we want to convert it into a grid, with each grid square denoting one of the three types of regions - road, hotspot, blocked.

### 4.2.1 Google Static Maps API

For overlaying a grid on the map, we need to able to distinguish various features on the 2D map. The Google Static Map API helps us color various points of interest, roads, blocked regions using different colors. So, we can do some simple image processing to identify road grids, hotspot grids and other regions. Here are a few important details on the Static Map API.

#### Using the API

To obtain a map from the Google Static Map API, we provide a URL specifying the region and necessary parameters. A Google Static Maps API URL must be of the following form:
`http://maps.google.com/maps/api/staticmap?parameters`
As is standard in URLs, all parameters are separated using the ampersand character. The Static Maps API defines map images using the following URL parameters:

- **Location Parameters**

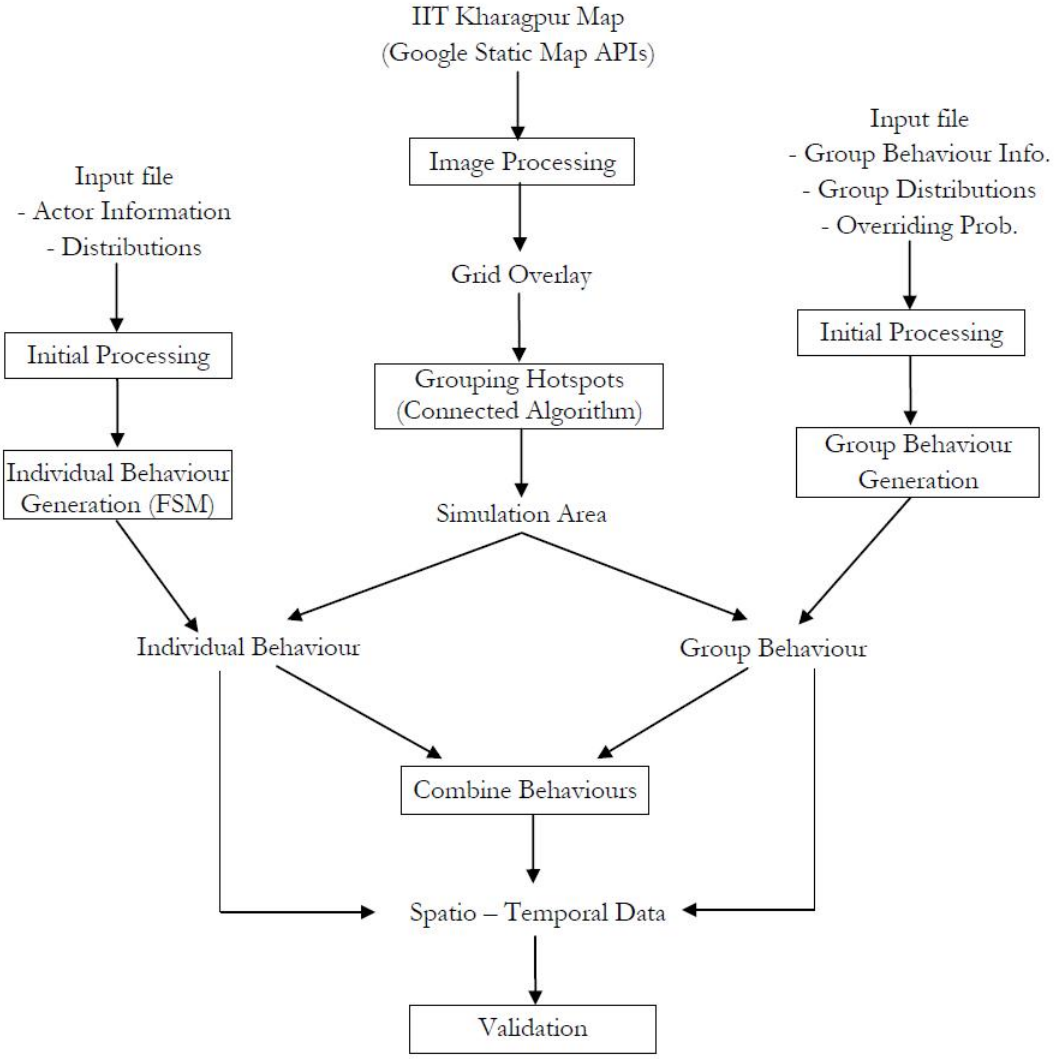



Figure 4.1: Simulator: High-Level Flow Diagram

- *center* defines the center of the map, equidistant from all edges of the map. This parameter takes a location as either a comma-separated latitude,longitude pair or a string address (e.g. "city hall, new york, ny") identifying a unique location on the face of the earth. The centre of our IIT map is: $(22.314468, 87.306376)$.
- *zoom* defines the zoom level of the map, which determines the magnification level of the map. This parameter takes a numerical value corresponding to the zoom level of the region desired. Zoom levels between 0 (the lowest zoom level, in which the entire world can be seen on one map) to 21 (down to individual buildings) are possible within the default roadmap maps view. Our IIT map is at a zoom level of 15.

- **Map Parameters**
  - *size* (required) defines the rectangular dimensions of the map image. This parameter takes a string of the form $value \times value$ where horizontal pixels are denoted first while vertical pixels are denoted second. For example, 500x400 defines a map 500 pixels wide by 400 pixels high. Our IIT map is of size $560 \times 560$.
  - *format* (optional) defines the format of the resulting image. By default, the Static Maps API creates PNG images. There are several possible formats including GIF, JPEG and PNG types. Which format you use depends on how you intend to present the image. JPEG typically provides greater compression, while GIF and PNG provide greater detail. Our kgp map is in .jpeg format.
  - *maptype* (optional) defines the type of map to construct. There are several possible maptype values, including roadmap, satellite, hybrid, and terrain. We have used the roadmap type for the IIT campus map.
  - *language* (optional) defines the language to use for display of labels on map tiles. (not used)

- **Feature Parameters**
  - *markers* (optional) define one or more markers to attach to the image at specified locations. This parameter takes a single marker definition with parameters separated by the pipe character (|). Multiple markers may be placed within the same markers parameter as long as they exhibit the same style; you may add additional markers of differing styles by adding additional markers parameters. Note that if you supply

markers for a map, you do not need to specify the (normally required) center and zoom parameters. We do not use markers on our IIT map.

- *path* (optional) defines a single path of two or more connected points to overlay on the image at specified locations. This parameter takes a string of point definitions separated by the pipe character (|). You may supply additional paths by adding additional path parameters. Note that if you supply a path for a map, you do not need to specify the (normally required) center and zoom parameters.
- *visible* (optional) specifies one or more locations that should remain visible on the map, though no markers or other indicators will be displayed. Use this parameter to ensure that certain features or map locations are shown on the static map.
- *style* (optional) defines a custom style to alter the presentation of a specific feature (road, park, etc.) of the map. This parameter takes feature and element arguments identifying the features to select and a set of style operations to apply to that selection. You may supply multiple styles by adding additional style parameters.

- **Reporting Parameters**
  - *sensor* (required) specifies whether the application requesting the static map is using a sensor to determine the user's location. This parameter is required for all static map requests. The sensor usage is set to false for our map generation.

### Styled Maps

Styled maps allow us to customize the presentation of the standard Google map styles, changing the visual display of such elements as roads, parks, and built-up areas to reflect a different style than that used in the default map type. These components are known as features and a styled map allows us to select these features and apply visual styles to their display (including hiding them entirely). With these changes, the map can be made to emphasize particular components or complement content within the surrounding page. Thus, the Visible and Style features are used extensively in our map generation, for obtaining only roads, only hotspots, man made landscapes, etc.

A customized "styled" map consists of one or more specified styles, each indicated through a style parameter within the Static Map request URL. Additional

styles are specified by passing additional style parameters. A style consists of a selection(s) and a set of rules to apply to that selection. The rules indicate what visual modification to make to the selection. Each style declaration consists of the following arguments, separated using a pipe (|) character within the style declaration:

- *feature* (optional) indicates what features to select for this style modification. If no feature argument is passed, all features will be selected.
- *element* (optional) indicates what sub-set of the selected features to select. If no element argument is passed, all elements of the given feature will be selected.

Note that: the style declaration must specify the above arguments in the order stated. For example, a feature selection with two rules would appear as show below:

```
style=feature:featureArgument|element:elementArgument|rule1:rule1Argument|
rule2:rule2Argument
```

**Map Features** A map consists of a set of features, such as roads or parks. The feature types form a category tree, with *feature:all* as the root. Some common features are listed below:

- *feature:all* (default) selects all features of the map.
- *feature:road* selects all roads on the map.
- *feature:landscape* selects all background landscapes on a map, which is often the area between roads, for example. In cities, landscape usually consists of built-up areas.

Some feature type categories contain sub-categories which are specified using a dotted notation (landscape.natural or road.local, for example). If the parent feature (road, for example) is specified, then styles applied to this selection will be applied to all roads, including sub-categories.

**Map Feature Elements** Additionally, some features on a map typically consist of different elements. A road, for example, consists of not only the graphical line (geometry) on the map, but the text denoting its name (labels) attached the map. Elements within features are selected by declaring an element argument. The following element argument values are supported:

- *element:all* (default) selects all elements of that feature.
- *element:geometry* selects only geometric elements of that feature.
- *element:labels* selects only textual labels associated with that feature.

If no element argument is passed, styles will be applied to all elements of the feature regardless of element type.

**Style Rules** Style rules are formatting options which are applied to the features and elements specified within each style declaration. Each style declaration must contain one or more operations separated using the pipe (|) character. Each operation specifies its argument value using the colon (":") character, and all operations are applied to the selection in the order in which they are specified. The following operation arguments, and the values that take, are currently supported:

- *hue* (an RGB hex string of format 0xRRGGBB) indicates the basic color to apply to the selection.
- *lightness* (a floating point value between -100 and 100) indicates the percentage change in brightness of the element. Negative values increase darkness (where -100 specifies black) while positive values increase brightness (where +100 specifies white).
- *saturation* (a floating point value between -100 and 100) indicates the percentage change in intensity of the basic color to apply to the element.
- *gamma* (a floating point value between 0.01 and 10.0, where 1.0 applies no correction) indicates the amount of gamma correction to apply to the element. Gammas modify the lightness of hues in a non-linear fashion, while unaffecting white or black values. Gammas are typically used to modify the contrast of multiple elements. For example, you could modify the gamma to increase or decrease the contrast between the edges and interiors of elements. Low gamma values (¡ 1) increase contrast, while high values (¿ 1) decrease contrast.
- *inverse-lightness:true* simply inverts the existing lightness.
- *visibility (on, off, or simplified)* indicates whether and how the element appears on the map. *visibility:simplified* indicates that the map should simplify the presentation of those elements as it sees fit. (A simplified road structure may show fewer roads, for example.)

Style rules must be applied as separate, distinct operations, and are applied in the order they appear within the style declaration. Order is important, as some operations are not commutative. Features and/or elements that are modified through style operations (usually) already have existing styles; the operations act on those existing styles, if present.

**URLs Used.** The following URLs are used for obtaining different types of IIT Kharagpur campus maps.

- Color map (without labels) (see Fig. 4.2)

```
http://maps.google.com/maps/api/staticmap?center=22.314468,87.
306376&zoom=15&size=560x560&sensor=false&style=feature:all|element:
labels|visibility:off&format=jpg
```

- Monochromatic map (only roads, no labels) (see Fig. 4.3)

```
http://maps.google.com/maps/api/staticmap?center=22.314468,87.
306376&zoom=15&size=560x560&sensor=false&style=feature:all|hue:
0xffffff|saturation:100|lightness:100&style=feature:road|hue:0x000000|
saturation:-100|lightness:-100&style=feature:all|element:labels|
visibility:off&format=jpg
```

- Monochromatic map (only hotspots, no labels) (see Fig. 4.4)

```
http://maps.google.com/maps/api/staticmap?center=22.314468,87.
306376&zoom=15&size=560x560&sensor=false&style=feature:all|element:
all|hue:0xffffff|saturation:100|lightness:100&style=feature:poi.
sports_complex|element:all|hue:0x000000|lightness:-100|saturation:
-100&style=feature:landscape.man_made|element:all|hue:0x000000|
lightness:-100|saturation:-100&style=feature:all|element:labels|
visibility:off&format=jpg
```

- Color map (roads and popular hotspots) (see Fig. 4.5)


Figure 4.2: IIT Kharagpur Color Map: No Labels

```
http://maps.google.com/maps/api/staticmap?center=22.314468,87.
306376&zoom=15&size=560x560&sensor=false&style=feature:all|element:
all|hue:0xffffff|lightness:100&style=feature:poi.sports_complex|
element:all|hue:0xff0000|lightness:-50|saturation:100&style=feature:
landscape.man_made|element:all|hue:0xff0000|lightness:-50|saturation:
100&style=feature:road|element:all|hue:0x0000ff|lightness:-50|saturation:
100&style=feature:all|element:labels|visibility:off&format=jpg
```

#### Main Locations on the Map

The IIT Kharagpur campus map has one major road - the 2.2 km road, and several other small interconnecting roads. The major hotspots are the hostels, the institute building, the departments, the tech. market, the two stadiums of Jnan Ghosh and Tata Sports Complex, the restaurants, the Netaji auditorium, the residential houses, etc.

### 4.2.2 Overlaying the Grid

Once we have obtained the IIT Kharagpur roads and hotspots map, we have to overlay a grid on it. Taking each pixel as one small grid square, we construct a grid of $560 \times 560$. Each small grid square is one of three types:


Figure 4.3: IIT Kharagpur Monochromatic Map: Only roads


Figure 4.4: IIT Kharagpur Monochromatic Map: Only hotspots


Figure 4.5: IIT Kharagpur Color Map

- 0 - roads
- 1 - hotspots
- 2 - blocked

For obtaining this classification, we consider the only hotspots and only roads maps. The images obtained from the Google Static Maps API are RGB images. We have to convert them to intensity images by using the Matlab function: *rgb2gray*. Since, the roads are black in color, they will have intensity pixel values close to 0, while other regions that were white in the RGB map will have values close to 255. However, the road boundary pixels are not completely white or completely black, rather they are a mixture of the two and hence, their intensity values can vary between 0 - 100. But, this is acceptable and we consider roads to have intensity values: $I \leq 50$. The figures 4.6 to 4.8 depict the generated grid images.

### 4.2.3 Grouping Hotspots

Now that we have successfully classified each grid as a road or a hotspot or a blocked region, we focus on grouping hotspots. The grid overlay simply tells us which grids are hotspots. But it does not tell us which grids belong to which

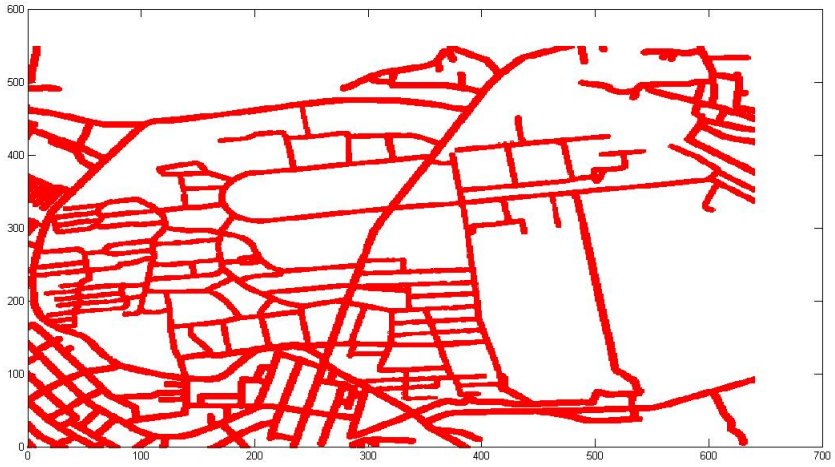



Figure 4.6: Generated Grid: Roads only

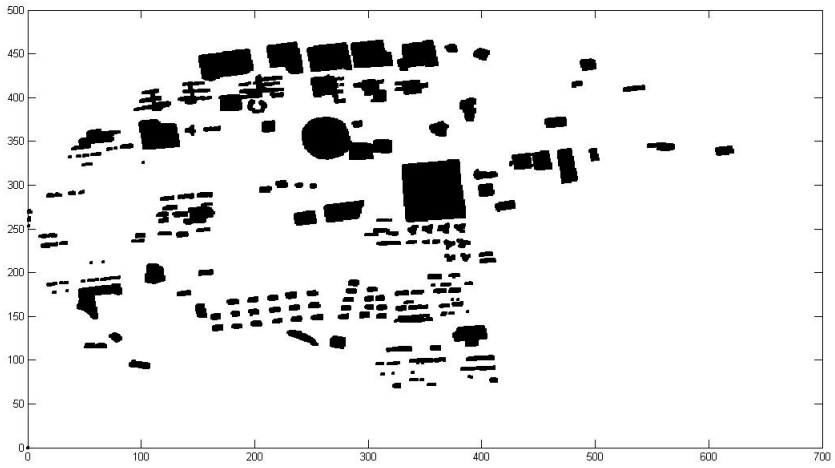



Figure 4.7: Generated Grid: Hotspots only


Figure 4.8: Generated Grid: Entire map

hotspot. This is essential to capture because often a hotspot spans over several grids. Knowing this information helps us to generate movements within the hotspots. Also, defining hotspot types becomes much easier.

### Connected Components Algorithm

For the purpose of identifying all grids of a single hotspot, we use the connected components algorithm. The grid representation of a map can be easily converted into a simple graph with hotspot grids as the vertices. All the grids belonging to a single hotspot will be in one connected component of the graph. So, we simply have to find all the connected components in the graph and number the hotspot grids accordingly.

**Algorithm.** The algorithm involves a repeated invocation of the simple Breadth first search procedure(see Fig. 4.9). A connected component consists of the BFS tree obtained from a source vertex. Maintain a component number to mark all the vertices in the current component. When the BFS is exhausted, we pick another vertex and invoke the function again. For this new component, we increase the component number and continue the same procedure till all components are found. A priority queue is used to implement the BFS procedure. The following is a simple idea of the algorithm used.

```
BFS(G, s)
 1  for each vertex u ∈ V[G] − {s}
 2       do color[u] ← WHITE
 3          d[u] ← ∞
 4          π[u] ← NIL
 5  color[s] ← GRAY
 6  d[s] ← 0
 7  π[s] ← NIL
 8  Q ← {s}
 9  while Q ≠ ∅
10       do u ← head[Q]
11          for each v ∈ Adj[u]
12               do if color[v] = WHITE
13                     then color[v] ← GRAY
14                          d[v] ← d[u] + 1
15                          π[v] ← u
16                          ENQUEUE(Q, v)
17          DEQUEUE(Q)
18          color[u] ← BLACK
```

Figure 4.9: BFS Algorithm: Pseudo-Code

1. Order the hotspot grids to obtain the total number of nodes in the graph $G(V, E)$.
2. Construct the adjacency matrix of the graph $G$.
3. Pick a source vertex $v \in V$.
4. Invoke the breadth first algorithm: $BFS(G, v)$.
5. Modify the BFS algorithm to mark each of the explored vertex with the current component number.
6. Remove all the vertices already covered in the BFS tree $T$: $V = V/T$.
7. Increase the component number.
8. Repeat from step 2 till all vertices are exhausted.

The final component number gives us the number of hotspots in the map. In our implementation, the IIT Kharagpur map has 210 hotspots of different types.

**Challenges.** The IIT Kharagpur campus map's grid representation had 25,339 hotspot grids. For the graph we plan to construct involving hotspot grids as vertices, if we store it as an adjacency matrix of $25339 \times 25339$ of type integer, we would be consuming about 3 GB of memory space, which is often too high for most personal computers. Adjacency list representations involve pointers but are an efficient way to go about it. However, we can use a simple modified representation of the adjacency matrix. The trick here lies in the maximum possible degree of a node in the hotspot graph. Each hotspot grid can be connected to atmost 8 of its neighbouring grids. So, each node can have a maximum degree of only 8. Thus, we can easily maintain an adjacency matrix of $25339 \times 8$ with $A[i][j]$ storing the node number of the $jth$ $(0 \leq j \leq 8)$ possible neighbour instead of just 1 or 0. This reduces the space requirements to only about 1 MB.

## 4.3 Individual Input Specifications

In our implementation of the simulator we focussed on two user classes, namely office-goers and college students. The following is a sample input format that we used in the design of our simulator.

1. Frequency of data generation
2. Velocity distribution of different modes of transportation
3. College students (User-Class 1)
    - No. of actors
    - Velocity Probability Distribution (for using different modes of transportation)
    - Type of state (for the hotspots)
    - Maximum duration of stay in that state
    - Departure and State Transition Probabilities' distribution as a function of time
4. Office Goers (User-Class 2)
    - No. of actors
    - Velocity Probability Distribution (for using different modes of transportation)

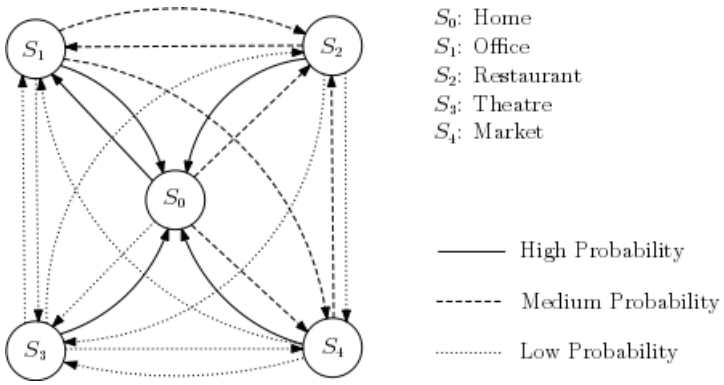



Figure 4.10: Office Goers: Sample Transitions

- Type of state (for the hotspots)
- Maximum duration of stay in that state
- Departure and State Transition Probabilities' distribution as a function of time

### 4.3.1 Specifying Behaviour

The simulator covers the user-classes of office-goers and college students. We implemented the regular as well as irregular behaviour of the users from an individual standpoint. In the previous chapter we discuessed in detail the various behaviour types of a sample office-goer. Typical transitions include home to office, office to cafeteria, office to home, home to restaurant, office to restaurant, home to playground, home to market and so on. Similarly for college students the typical transition include hostel to department, department to nescafe, department to tech. market, department to hostel, hostel to movie theatre and so on. The state machine in Fig. 4.10 gives us an idea of the low and high probability transitions for office goers.

### 4.3.2 Probability Distributions

The departure time distributions together with the state transition probabilities form a very important element of the simulation process. In addition to these, we also have probability distributions for different modes of transportation being used by different user classes. The traffic modelling aspect by modifying velocity in different sections is under implementation. The following tables show the state transition probabilities and departure time distributions for the class of office

goers. These parameters were obtained from a small survey made among the students in the IIT Kharagpur campus.

Table 4.1: Velocity Distributions for various modes of transportation

| Transportation Mode | Min-speed (grids/hr) | Max-speed (grids/hr) | Probability Office Goers | Probability College Students |
|---|---|---|---|---|
| Car | 3000 | 4000 | 0.6 | 0 |
| Cycle | 800 | 1500 | 0.3 | 0.5 |
| Walk | 50 | 400 | 0.1 | 0.5 |

Table 4.2: State Transition Probabilities and Departure Time Distributions: State = Home, Max duration of stay = infinity

| Departure Time | Departure Probability | To Home | To Office | To Restaurant | To Theatre | To Market |
|---|---|---|---|---|---|---|
| 0-8 | 0.1 | 1 | 0 | 0 | 0 | 0 |
| 8-10 | 0.35 | 0 | 0.9 | 0.1 | 0 | 0 |
| 10-12 | 0.05 | 0 | 0.8 | 0.1 | 0 | 0.1 |
| 12-14 | 0.05 | 0 | 0.2 | 0.8 | 0 | 0 |
| 14-16 | 0.1 | 0 | 0.8 | 0.2 | 0 | 0 |
| 16-18 | 0.05 | 0 | 0.5 | 0.15 | 0.05 | 0.3 |
| 18-20 | 0.05 | 0 | 0 | 0.4 | 0.3 | 0.3 |
| 20-22 | 0.05 | 0 | 0.1 | 0.5 | 0.3 | 0.1 |
| 22-24 | 0.2 | 1 | 0 | 0 | 0 | 0 |

**Analysis of the Tables.** The first table represents the probabilities of various classes choosing different modes of transportation as well as the [min-speed,max-speed] ranges for the modes. The subsequent tables can be interpreted as follows. Consider table 4.2 which depicts transitions out of home. The maximum time of stay indicates the duration for which a user can stay at that location without any transitions. The first column indicates the departure time ranges from 0:00 hrs to 24:00 hrs. The second column represents what is the probability of a transiton out of home in that particular time interval. The next columns denote that if a transition happens in that time interval (depending on the departure probability), then what are the probabilities by which the user will choose a destination state. For example, consider the row corresponding to 10:00 - 12:00 hrs in the table 4.2.

Table 4.3: State Transition Probabilities and Departure Time Distributions:
State = Office, Max duration of stay = infinity

| Departure Time | Departure Probability | To Home | To Office | To Restaurant | To Theatre | To Market |
|---|---|---|---|---|---|---|
| 0-8 | 0.05 | 1 | 0 | 0 | 0 | 0 |
| 8-10 | 0.05 | 0.2 | 0 | 0.6 | 0 | 0.2 |
| 10-12 | 0.1 | 0.6 | 0 | 0.2 | 0 | 0.2 |
| 12-14 | 0.3 | 0.6 | 0 | 0.4 | 0 | 0 |
| 14-16 | 0.2 | 0.4 | 0 | 0.4 | 0.1 | 0.1 |
| 16-18 | 0.1 | 0.4 | 0 | 0.1 | 0.4 | 0.1 |
| 18-20 | 0.1 | 0.6 | 0 | 0 | 0.1 | 0.3 |
| 20-22 | 0.05 | 0.5 | 0 | 0.4 | 0.1 | 0 |
| 22-24 | 0.05 | 1 | 0 | 0 | 0 | 0 |

Table 4.4: State Transition Probabilities and Departure Time Distributions:
State = Theatre, Max duration of stay = 180 min

| Departure Time | Departure Probability | To Home | To Office | To Restaurant | To Theatre | To Market |
|---|---|---|---|---|---|---|
| 0-24 | 1 | 0.5 | 0.05 | 0.4 | 0 | 0.05 |

The departure probability from home is quite low at 0.05. This is because it is more likely that he has already left home in an earlier time interval, say for going to the office (which explains the 0.35 departure probability from 8-10 AM). Given that the user leaves home between 10 and 12, he is more likely to go to the office than to say a theatre or a market. This is accounted for by having a 'Transition to' probability of 0.8 for Office, 0.1 for Restaurant, 0 for Theatre and 0.1 for Market. Similarly other tables can be explained. Note that the less-visited behaviour states of restaurant, market, and theatre have a finite maximum duration value. Also, the departure time distribution is kept uniform throughout the 24 hours, which means that once the max time expires, the user has to depart from that state to some other state according to the 'transition to' probabilities. The actual duration of stay is generated randomly between [0,max-duration]. Thus, we see that these tables help us generate a sequence of daily activities (hotspots visited) for every user.

Table 4.5: State Transition Probabilities and Departure Time Distributions: State = Market, Max duration of stay = 40 min

| Departure Time | Departure Probability | To Home | To Office | To Restaurant | To Theatre | To Market |
|---|---|---|---|---|---|---|
| 0-24 | 1 | 0.8 | 0.1 | 0.1 | 0 | 0 |

## 4.4 Generating Individual Behaviour

In the previous section, we saw how to generate a sequence of activity locations starting from the start state of Home, using the departure time and transition probability distribution. Now the only task left to do is to generate the movement between two hotspots, i.e. plot the path taken by the user. The velocity distributions associated with different modes of transportation come into the picture. But most important of all, we have to use a path-finding algorithm to find possible intermediate locations on the path leading from one hotspot to another, at a frequency specified as input. We use the $A^*$ algorithm for this job. For motion inside a hotspot, we use one of the popular entity models - the random waypoint model.

### 4.4.1 Path-Finding Algorithm: $A^*$

The $A^*$ algorithm is a very popular graph traversal and path finding algorithm used commonly in the domain of Artificial Intelligence. It uses a best-first search and finds the least-cost path from a given initial node to one goal node (out of one or more possible goals). It uses a distance-plus-cost heuristic function (usually denoted by $f(x)$) to determine the order in which the search visits nodes in the tree. The distance-plus-cost heuristic is a sum of two functions:

- The path-cost function, which is the cost from the starting node to the current node (usually denoted by $g(x)$)
- An admissible "heuristic estimate" of the distance to the goal (usually denoted by $h(x)$).

The $h(x)$ part of the $f(x)$ function must be an admissible heuristic; that is, it must not overestimate the distance to the goal. Thus, for an application like routing, $h(x)$ might represent the straight-line distance to the goal, since that is physically the smallest possible distance between any two points or nodes.

**Algorithm Idea**

The following description gives a simple idea about the algorithm.

1. Create a search graph $G$, consisting solely of the start node, $s$. Put $s$ in a list called OPEN.

2. Create a list called CLOSED that is initially empty.

3. If OPEN is empty, exit with failure.

4. Select the first node on OPEN, remove it from OPEN, and put it on CLOSED. Called this node $n$.

5. If $n$ is a goal node, exit successfully with the solution obtained by tracing a path along the pointers from $n$ to $s$ in $G$. (The pointers define a search tree and are established in Step 7.)

6. Expand node $n$, generating the set $M$, of its successors that are not already ancestors of $n$ in $G$. Install these members of $M$ as successors of $n$ in $G$.

7. Establish a pointer to $n$ from each of those members of $M$ that were not already in $G$ (i.e., not already on either OPEN or CLOSED). Add these members of $M$ to OPEN. For each member, $m$, of $M$ that was already on OPEN or CLOSED, redirect its pointer to $n$ if the best path to $m$ found so far is through $n$. For each member of $M$ already on CLOSED, redirect the pointers of each of its descendants in $G$ so that they point backward along the best paths found so far to these descendants.

8. Reorder the list OPEN in order of increasing $f$ values. (Ties among minimal f values are resolved in favor of the deepest node in the search tree.)

9. Go to Step 3.

**Our Implementation**

In our version of the implementation, the search graph $G$ comprises of road grids as nodes. So, we search roads for a path from one hotspot to another, reporting intermediate nodes (road points) at every interval defined by the specified frequency.Our start node $s$ is one of the grids of Hotspot $X$, and the goal node $g$ is one of the grids of Hotspot $Y$. The various cost/heuristic functions used in the implementation are mentioned below.

- The heuristic function is taken to be the Euclidean distance from each intermediate road grid to one of the grids in Hotspot $Y$.
- The cost function depends on the weights of the edges between neighbouring node grids. We have chosen the weight $w(i, j) = 2$ if nodes $i$ and $j$ are adjacent neighbours, and $w(i, j) = 3$ if they are diagonal neighbours.

**Challenges**

**Problem 1.** The number of road grids on the map is close to 60,000, which gives us a graph with as many vertices. We can use a modified adjacency matrix representation (see Section 4.2.3) or a linked list to store the graph effectively. But still the $A^*$ algorithm takes higher order polynomial time ($O(n^3)$ or higher) for its execution. This leads to a very large execution time, which is not preferrable if we want to compute paths for every movement of all users.

**Solution.** There are two solutions to this problem. The main issue with the previous approach was that considering each road grid as a node did not lead to an elegant representation. So, we fix certain important road grids (such as intersections between roads) and of course some road grids near the hotspots, as our nodes of the graph. This leads to a graph of quite small size and $A^*$ can run easily. The second solution is based on the fact that the number of hotspots on the map and thus, the possible paths between hotspots are finite and small in number. So, we can run $A^*$ repeatedly for finding all pairs shortest path between all hotspots. These paths, i.e. the road grids on each path can be stored in a table. Whenever we want to generate a path between hotspots, we simply perform a table lookup and obtain the sequence of road grids. Thus, we can easily report $(x, y, t)$ coordinates for a user movement.

**Problem 2.** The connected component algorithm mentioned in the previous section grouped hotspot grids effectively. But most of the hotspots, are not directly connected to the major roads in the map, i.e. the local hotspot roads are not depicted at the current level of detail. So, it becomes hard to actually move between hotspots if neither are connected to a nearby road.

**Solution.** An easy solution to this problem is to find a representative point of the hotspot on a nearby road, and then compute the shortest path among these points. In addition, we may choose to extend the roads by joining these representative road grids to their corresponding hotspots. These points can be


Figure 4.11: Extending roads to join hotspots

easily found by iterating in the vicinity of the hotspots till we find a road grid. (see Fig. 4.11)

### 4.4.2 Spatio-Temporal Data Generation

The implementation of the entire simulator was done on the IIT Kharagpur campus map, obtained from the Google Static Maps API. A $560 \times 560$ grid was overlayed on the map with each grid being classified as a road/hotspot/blocked grid. Fig. 4.12 shows the campus map with the hotspots and the roads. The following are the types of hotspots on the map:

- Hostel
- Home
- Department/Office
- Restaurant/Nescafe
- Playground

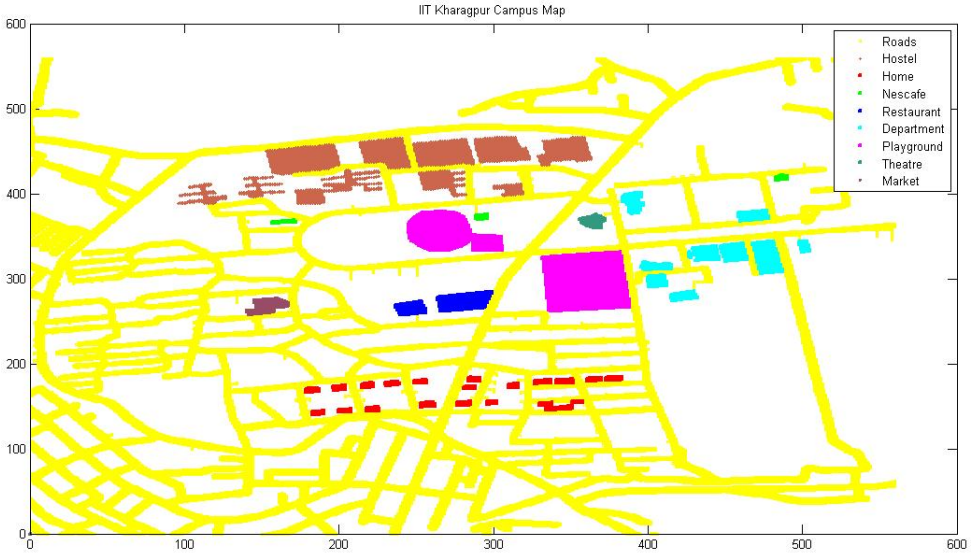



Figure 4.12: IIT Kharagpur Campus Map

- Theatre
- Market

The simulation comprised a 1000 users belonging to two user-classes, namely college students and office goers (500 each). The frequency of data generation was kept at 3 minute intervals. We obtained spatio-temporal data for these users for a period of 1 month. The subsequent figures represent the generated user paths on the map.

### Analysis of Paths

Fig. 4.13 represents the generated path corresponding to the behaviour of one college student for a single day. The college student's day begins at 8:30 AM when he goes from his hostel to his Department on a bicycle. He reaches his department in 9 minutes. He stays in his department till 12:30 PM at which point he comes back to the hostel for lunch. He stays in the hostel till 4:30 PM at which point he goes to the market on his bicycle. It takes him 15 minutes to reach the market. He stays at the market for 35 minutes. Then, he returns to his hostel at 5:40 PM.

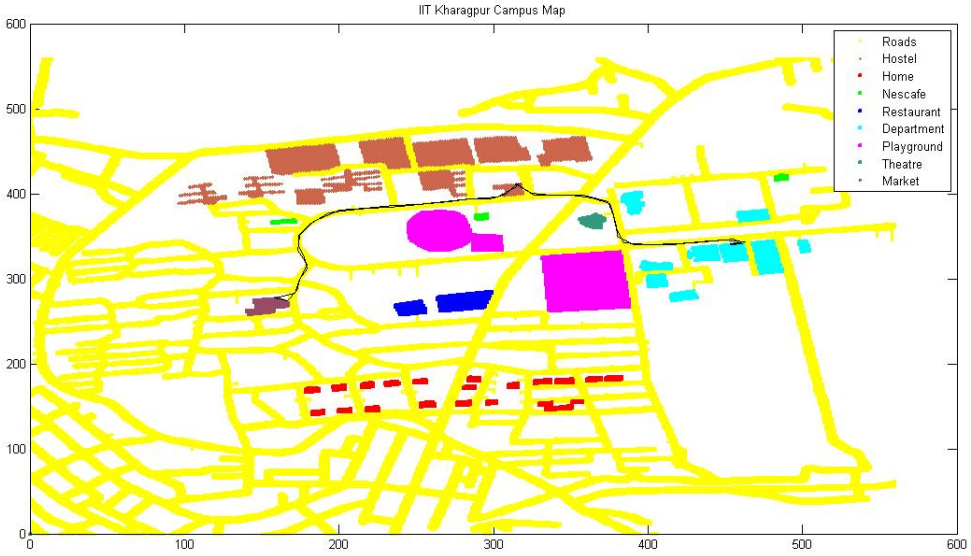



Figure 4.13: Generated Path of 1 College Student

For the remainder of the day, he stays in the hostel itself. RWP model is used to simulate his in-hotspot behaviour while the individual specifications are used for generating the inter-hotspot motion.

Similarly, Fig. 4.14 represents the generated path corresponding to the behaviour of one office goer for a single day. His day starts at 9:15 AM when he goes to the department/office from his home in a car. It takes him 9 minutes to reach the department. He stays in the department till 4:30 PM in the afternoon. Possibly he has brought his lunch from home. He leaves for the market from his office, in his car, and reaches the market in 12 minutes. He spends 25 minutes at the market. Next, he leaves for a restaurant to have his evening snacks reaching there by 5:20 PM. He spends 40 minutes at the restaurant and returns home after having his food. It takes him all of 6 minutes to reach his home from the restaurant. For the remainder of the day, he stays at home. Similar analysis can be performed for the Fig. 4.15 comprising ten office goers and ten college students. Note that the behaviour generated could be completely different for the next day.

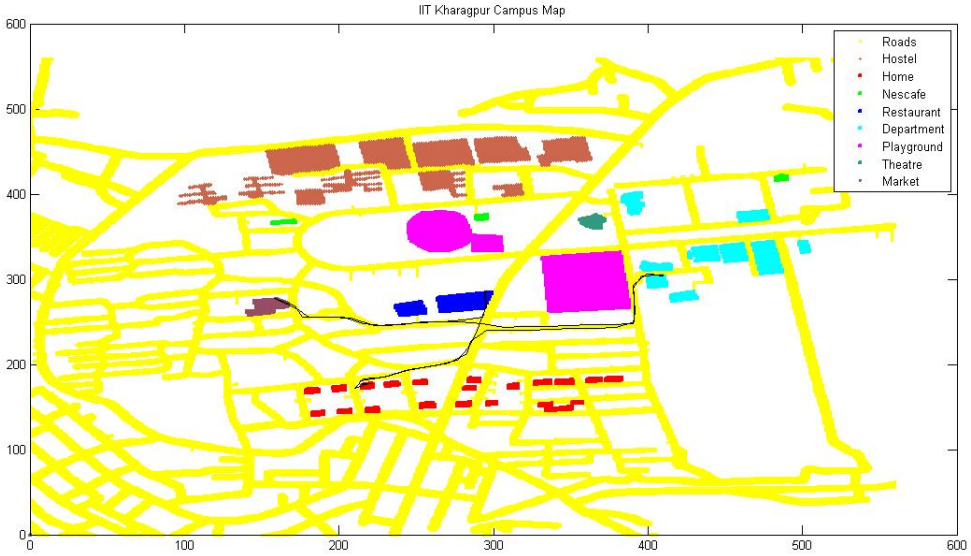



Figure 4.14: Generated Path of 1 Office Goer

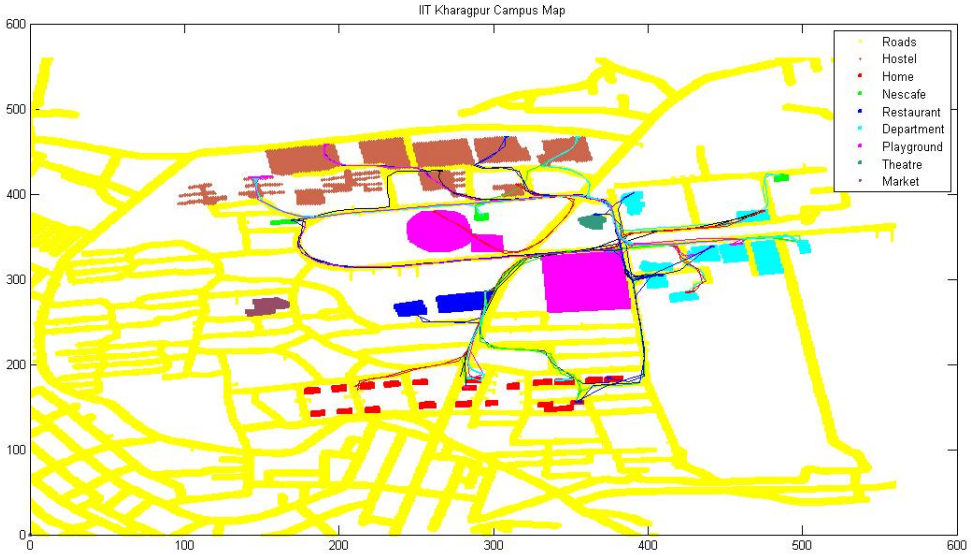



Figure 4.15: Generated Path of 10 College Students and 10 Office Goers

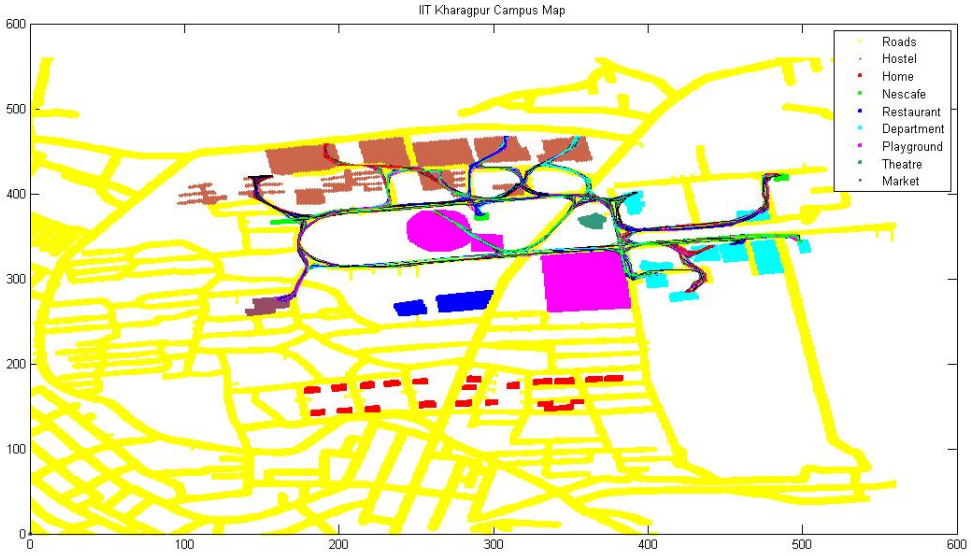



Figure 4.16: Generated Path of 500 College Students

Fig. 4.16 and 4.17 depict the movement patterns of the two user-classes separately. We can see that the 500 college students are more or less concentrated in the top half of the map and the 500 office goers on the bottom half of the map because of the orientations of their home locations. However, both transit into the institute very freuquently during a day. Fig. 4.18 represents the simulated data for a day for all the 1000 users. Since, we have used an intuitive model to generate data on an actual campus map, the movement patterns obtained are quite realistic.

### 4.4.3 Validation

The validation procedure involves pooling back the generated activity sequences for all the users in a particular user-class to obtain the user-class specifications, and then comparing it with the original input specifications. If the difference between the two is within an acceptable level of error, then our data generated is consistent with the specifications. Since, the input specifications are given in the form of transition distributions, it is quite easy to validate the data by computing the various probabilities of transitions and comparing them with the

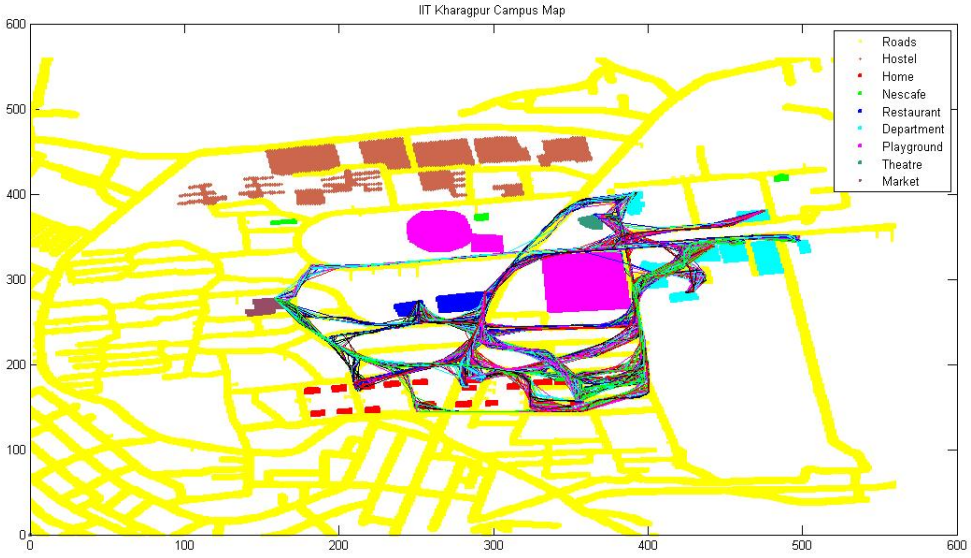



Figure 4.17: Generated Path of 500 Office Goers

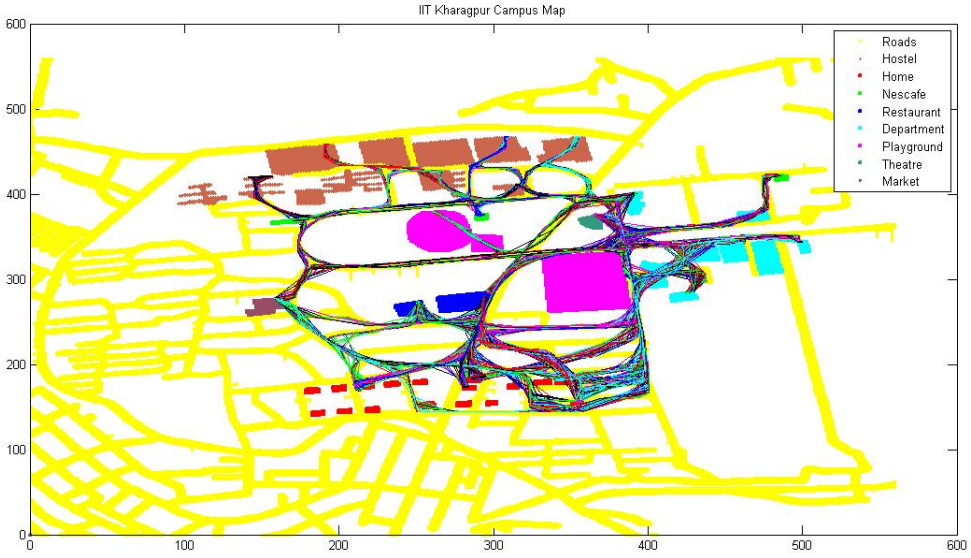



Figure 4.18: Generated Path of 500 College Students and 500 Office Goers

input probabilities.

For this purpose, we considered all the possible transitions (with probabilities above a threshold) between a pair of hotspots made by the 500 users of a user-class in a single day on an average. Then, we counted the actual number of transitions made in the generated data between hotspots. The following formula gives us the average probability of transition from one hotspot to another:

$$p_{i \to j} = \frac{count(i \to j)}{\sum count(i \to k)}$$

Note that we could also compute these values corresponding to different periods during the day. In our version, we have considered an average probability value for a day for each transition.

The tables 4.6 to 4.9 denote the various probability comparisons for college students and office goers for transitions from home/hostel and the department/office. We also report the rms error between the input and generated specifications, which is of the order of 0.01, implying that the data generated is consistent. Note that the sum of all transitions from a state for a particular user-class is less than 500. This is because not all users of a particular user class do each activity transition on a given day.

Table 4.6: College Students: From Hostel Transitions
Total No. of transitions = 486, R.M.S Error = 0.012

| Transition Type | Specified Probability | Observed No. of Transitions | Observed Probability | Error |
|---|---|---|---|---|
| Hostel-Dept. | 0.6285 | 296 | 0.609 | 0.0195 |
| Hostel-Nescafe | 0.1857 | 96 | 0.1975 | 0.0118 |
| Hostel-Playground | 0.107 | 52 | 0.107 | 0.0 |
| Hostel-Market | 0.0785 | 42 | 0.0864 | 0.0079 |

### Validation on Random Maps

**Generating Paths.** In the beginning of this section, we saw how to validate the generated data on the IIT Kharagpur campus map. We also generated data on different random maps using the same behaviour models and validated these data as well. The random maps give us more control over the number and distribution

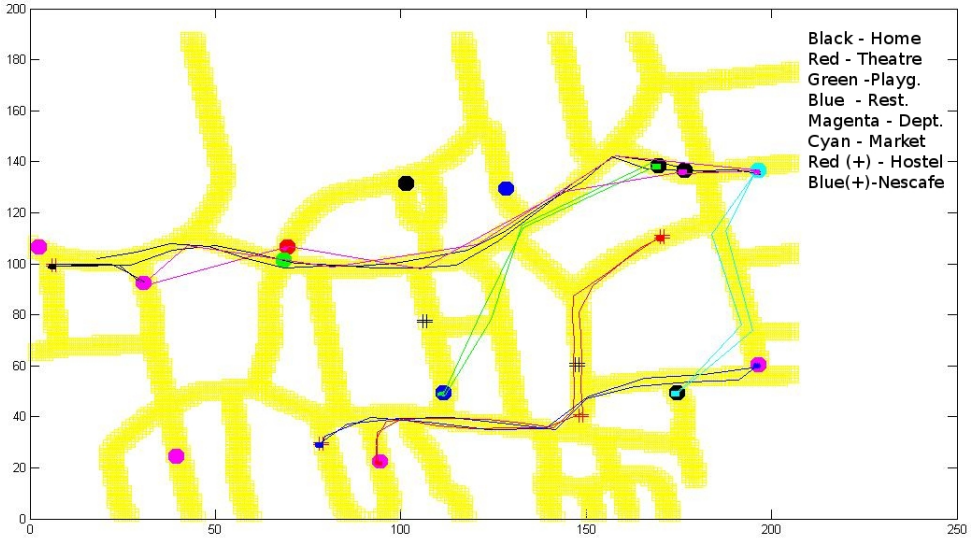



Figure 4.19: Map with Random Hotspots

of hotspots, blocked regions as well as road grids. Here, we present a small random map of size $200 \times 200$ with randomly generated hotspots of types - Hostel, Home, Office/Department, Nescafe, Restaurant, Market, Theatre, Playground. For convenience sake, we have assumed that these hotspots lie on the road itself. Fig. 4.19 provides a snapshot of the generated paths for 3 office goers and 3 college students.

**Validating the Data.** The behaviour model chosen for different user-classes is an intuitive user model. This means that the probabilities of transitions, different hotspots visited every day, their sequence, should remain more or less the same irrespective of the simulation area chosen. This would mean that the data generated for any particular class of users should be similar on any simulation grid. To verify this, we tried to generate data on a random section of the IIT Kharagpur campus map with hotspots randomly placed on the roads. For validating this data, we used the same procedure as before, counting the relative number of transitions from one state to different states. Tables 4.10 to 4.13 give us a picture of the validation procedure on the random map. Note that we have chosen the same transitions as in the original map validation so as to facilitate comparision. Table

Table 4.7: College Students: From Dept. Transitions
Total No. of transitions from Dept. = 286, R.M.S Error = 0.051

| Transition Type | Specified Probability | Observed No. of Transitions | Observed Probability | Error |
|---|---|---|---|---|
| Dept.-Hostel | 0.55 | 168 | 0.587 | 0.037 |
| Dept.-Nescafe | 0.22 | 74 | 0.258 | 0.038 |
| Dept.-Playground | 0.195 | 36 | 0.125 | 0.07 |
| Dept.-Market | 0.035 | 8 | 0.028 | 0.007 |

Table 4.8: Office Goers: From Home Transitions
Total No. of transitions = 471, R.M.S Error = 0.0193

| Transition Type | Specified Probability | Observed No. of Transitions | Observed Probability | Error |
|---|---|---|---|---|
| Home-Dept. | 0.6785 | 298 | 0.6467 | 0.0318 |
| Home-Rest. | 0.2178 | 101 | 0.219 | 0.0012 |
| Home-Theatre | 0.0464 | 28 | 0.0607 | 0.0143 |
| Home-Market | 0.0571 | 34 | 0.0737 | 0.0166 |

4.14 shows that the average RMS error for the chosen transitions is quite low for both the actual map and the random map, thereby validating the accuracy of our model. An interesting observation is that the number of transitions to irregular behaviour states actually increases in a smaller map. This is because we are using the actual map's velocity distributions for this small map. So, the transition time is quite low, and hence more no. of transitions for irregular behaviour are possible.

## 4.5 Interesting Statistics

Now that we have generated the individual data for two classes of users, viz. office-goers and college students, we can perform a comparative analysis of the two data-sets. We also analyze some of the properties associated with different hotspots and different road segments, such as busiest road segment and its traffic distribution for different periods of a day, busiest hotspot and its user distribution for a day. We can also compute statistics such as average time spent per hotspot, frequently visited hotspots for the two user-classes and so on.

Table 4.9: Office Goers: From Office/Dept. Transitions
Total No. of Transitions = 327, R.M.S Error = 0.0166

| Transition Type | Specified Probability | Observed No. of Transitions | Observed Probability | Error |
|---|---|---|---|---|
| Dept-Home | 0.555 | 190 | 0.581 | 0.026 |
| Dept-Rest. | 0.28 | 85 | 0.26 | 0.02 |
| Dept-Theatre | 0.075 | 23 | 0.0703 | 0.0047 |
| Dept-Market | 0.09 | 29 | 0.0886 | 0.0014 |

Table 4.10: College Students: From Hostel Transitions (On Random Map)
Total No. of transitions = 477, R.M.S Error = 0.046

| Transition Type | Specified Probability | Observed No. of Transitions | Observed Probability | Error |
|---|---|---|---|---|
| Hostel-Dept. | 0.6285 | 267 | 0.559 | 0.0695 |
| Hostel-Nescafe | 0.1857 | 85 | 0.1782 | 0.0075 |
| Hostel-Playground | 0.107 | 79 | 0.1656 | 0.0586 |
| Hostel-Market | 0.0785 | 46 | 0.0964 | 0.0179 |

### Time Spent at Each Hotspot

The behaviour of the two classes is governed by their respective state machines. The college students' FSM comprises the states of hostel, department, nescafe, playground and market, while that of the office-goers consists of states like home, office/department, restaurant, theatre and market. Table 4.15 gives a comparative view of the time spent by a member of each user class on an average at each hotspot during a day. As expected, most of the time is spent in the regular behaviour states of home/hostel and office/department, with a few hours devoted to the irregular behaviour states.

### Busiest Road Segment

The busiest road segment can be easily found by computing the number of users travelling through that section of the road on a given day. We also obtain a distribution of the traffic on that road at different hours of the day. It is observed that the road segment between Vidhan Chowk and the Gate No. 5 is the busiest. This is obvious since most of the hotspots corresponding to regular behaviour of the users, chiefly departments/offices are located in the vicinty of this road. Thus,

Table 4.11: College Students: From Dept. Transitions (On Random Map)
Total No. of transitions from Dept. = 336, R.M.S Error = 0.0213

| Transition Type | Specified Probability | Observed No. of Transitions | Observed Probability | Error |
|---|---|---|---|---|
| Dept.-Hostel | 0.55 | 177 | 0.5267 | 0.0233 |
| Dept.-Nescafe | 0.22 | 85 | 0.253 | 0.033 |
| Dept.-Playground | 0.195 | 61 | 0.182 | 0.013 |
| Dept.-Market | 0.035 | 13 | 0.039 | 0.004 |

Table 4.12: Office Goers: From Home Transitions (On Random Map)
Total No. of transitions = 466, R.M.S Error = 0.0498

| Transition Type | Specified Probability | Observed No. of Transitions | Observed Probability | Error |
|---|---|---|---|---|
| Home-Dept. | 0.6785 | 277 | 0.5944 | 0.0841 |
| Home-Rest. | 0.2178 | 109 | 0.2339 | 0.0161 |
| Home-Theatre | 0.0464 | 43 | 0.0923 | 0.0459 |
| Home-Market | 0.0571 | 37 | 0.0794 | 0.0223 |

any regular behaviour transition of users, i.e. hostel to department or home to office and back, is highly likely to take them through this road segment. Fig. 4.20 gives us an idea of the busiest road segment.

**Traffic Distribution.** In Fig. 4.21, we present the traffic distribution for the busiest road segment for both classes of users. The traffic gradually increases as the day progresses, with more and more students and office-goers coming inside the academic campus. The traffic hits a maximum during the period of 12 - 1 PM, when most of the users return to their homes/hostels for lunch. It also hits a local maximum in the 2 - 3 PM window when a lot of the users come back to work from their lunch breaks. Another local maximum is obtained in the 4 - 5 PM window when several of the users return home after their day's work. The traffic gradually falls for the remainder of the day.

### Busiest Hotspot

The busiest hotspot can be computed in a manner similar to that of the busiest road segment. For this purpose, we count the number of people visiting a hotspot

Table 4.13: Office Goers: From Office/Dept. Transitions (On Random Map)
Total No. of Transitions = 358, R.M.S Error = 0.0211

| Transition Type | Specified Probability | Observed No. of Transitions | Observed Probability | Error |
|---|---|---|---|---|
| Dept-Home | 0.555 | 202 | 0.5642 | 0.0092 |
| Dept-Rest. | 0.28 | 88 | 0.2458 | 0.0342 |
| Dept-Theatre | 0.075 | 35 | 0.0978 | 0.0228 |
| Dept-Market | 0.09 | 33 | 0.0922 | 0.0022 |

Table 4.14: Comparision of Validated Data: Complete IIT Map vs Random Map

| Transition Type | Average R.M.S Error (Actual Map) | Average R.M.S Error (Random Map) |
|---|---|---|
| From Hostel (College Students) | 0.012 | 0.046 |
| From Dept. (College Students) | 0.056 | 0.0213 |
| From Home (Office Goers) | 0.0193 | 0.0498 |
| From Office/Dept. (Office Goers) | 0.0166 | 0.0211 |

at different time intervals. The busiest hotspot is the one with the peak number of users at any point during the day. In the IIT Kharagpur campus map, the busiest hotspot was the Institute main building with the Tech Market coming in a close second. Again this is obvious, since the main building houses several departments which comprise the regular behaviour of both office-goers and college students. Also, the tech market is the most likely irregular behaviour state visited by either class. Fig. 4.20 gives us an idea of the busiest hotspot.

**User Distribution.** In Fig. 4.22, we present the user distribution of the Institute main building for both classes of users. The number of users gradually increases as the day progresses with a peak around 10 - 11 AM. This is because most of the offices start in that duration and several classes also go on in that period in the main academic building. The distribution reaches a local minimum in the 12 - 2 PM interval as most people go for lunch. It again peaks around the 2 - 3 PM mark, as people return after their lunch breaks. As the day ends, the number of users gradually decreases till nightfall. Fig. 4.23 gives comparative bar graphs of the user distributions at the busiest road and busiest hotspot.

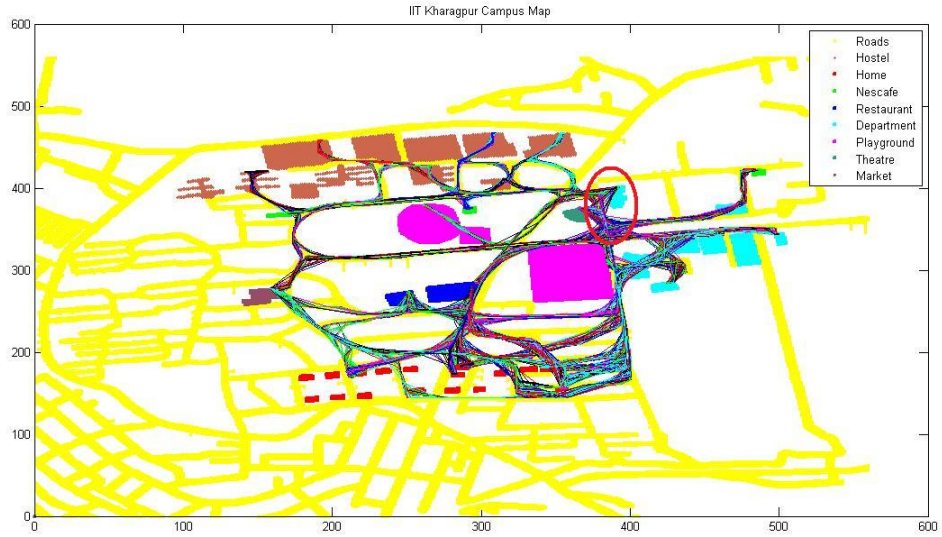



Figure 4.20: Busiest Road Segment and Busiest Hotspot

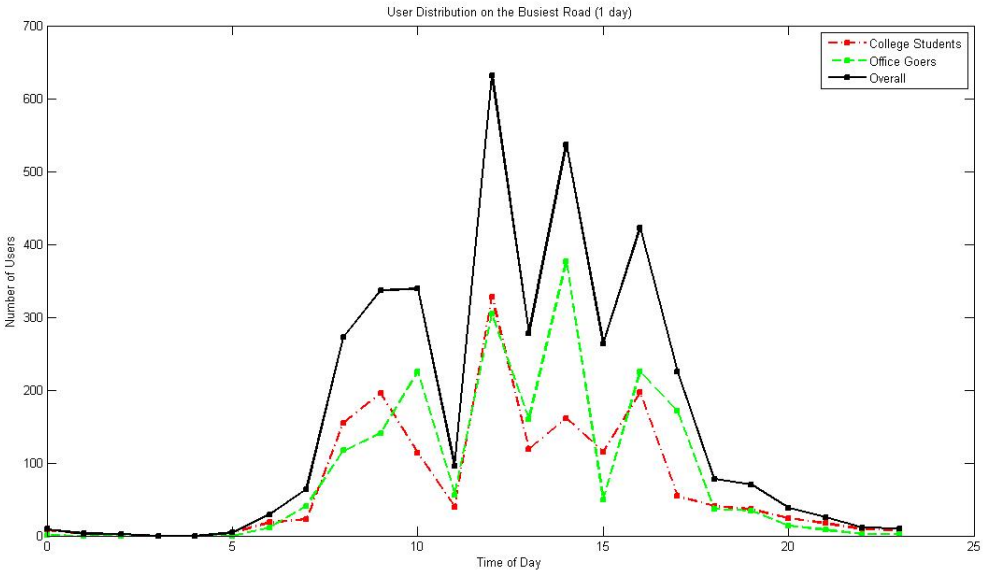



Figure 4.21: Traffic Distribution of the Busiest Road

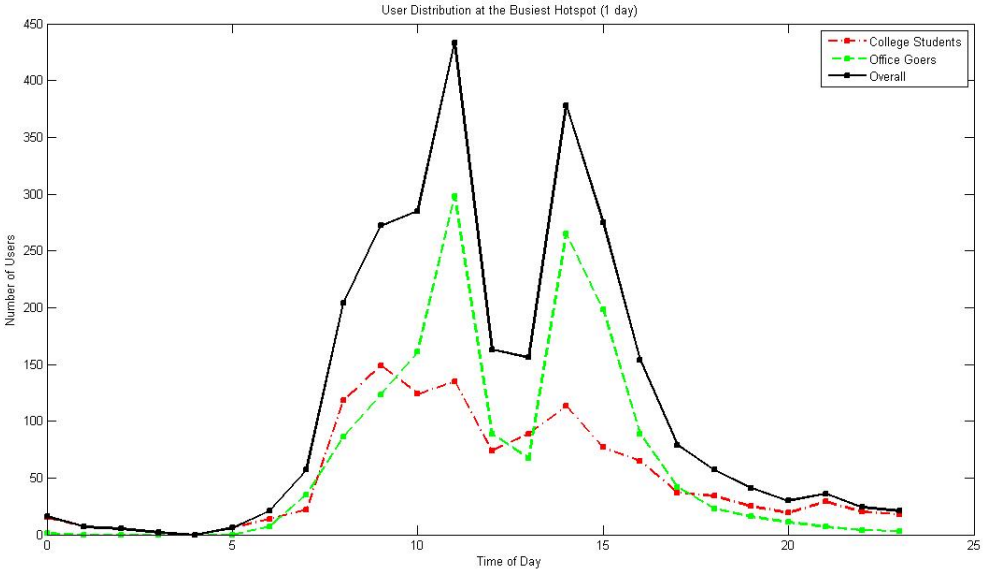



Figure 4.22: User Distribution at the Busiest Hotspot

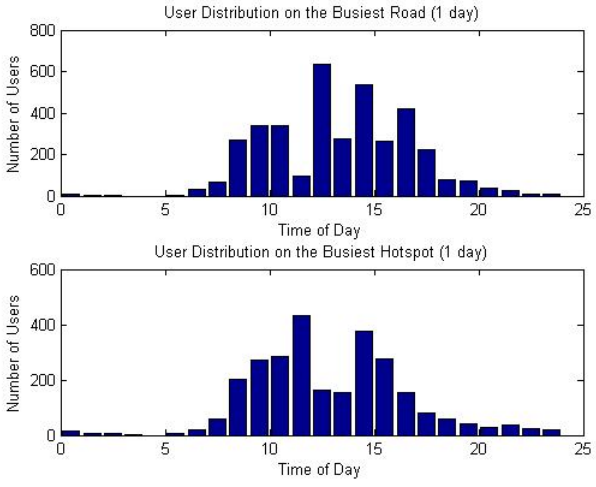



Figure 4.23: Comparative Bar Graphs: Busiest Road and Busiest Hotspot

Table 4.15: Average time spent at different Hotspots for 1 day

| Type of Hotspot | College Student (no. of hrs) | Office Goer (no. of hrs) |
|---|---|---|
| Home/Hostel | 12.66 | 11.91 |
| Dept. | 7.03 | 9.24 |
| Rest./Nescafe | 1.12 | 0.65 |
| Theatre | - | 0.47 |
| Market | 0.29 | 0.48 |
| Playground | 0.88 | - |

## 4.6 Generating Group Behaviour

In this section, we will see how to generate planned group behaviour for different group types. The input specifications provide us the different types of groups for each user class, and also the number of such groups. Also, we have a list of generic behaviour types along with the listed behaviour sequences. In addition, the input also specifies the various probabilities for different groups picking different behaviour sequences. Our first task, given these specifications is to greedily allot users to different groups. The various types of groups are:

- College Students
  - Hostel-mates
  - Department-mates
  - Research group members
- Office Goers
  - Family Members
  - Colleagues

### 4.6.1 Generic Behaviour Types

The generic behaviour types are defined based on the most prominent activity location among all the possible behaviour sequences. The following is a list of some of those types:

1. Nescafe Going Behaviour

- Go to Nescafe, followed by Theatre (prob. 0.1)
- Go to Nescafe, followed by Market (prob. 0.1)
- Go to Nescafe (prob. 0.8)

2. Playground Going Behaviour
   - Go to Nescafe, followed by Playground (prob. 0.3)
   - Go to Playground, followed by Nescafe (prob. 0.5)
   - Go to Playground (prob. 0.2)
3. Department Going Behaviour
   - Go to Department (prob. 0.75)
   - Go to Department, followed by Nescafe (prob. 0.2)
   - Go to Department, followed by Playground (prob. 0.05)

**Choosing Generic Behaviour.** Consider the group type Hostel-mates. They can choose from among two types of generic behaviours, namely nescafe going behaviour with probability 0.6, and playground-going behaviour with probability 0.4. Similarly, for the group type Dept.-mates , we have two possible generic behaviour choices in Dept. going behaviour and Playground going behaviour with probabilities of 0.7 and 0.3 respectively. Such details can be provided as part of the input specifications itself. Note that these probabilities will be controlled by the number of days of the month the user chooses to perform group behaviour.

### 4.6.2 Spatio-Temporal Data Generation

The group behaviour of different groups is modelled in the following way. First, each user follows his individual behaviour model to arrive at a certain common point where all the group members meet. Then, the group distributions take over and members perform activities together as a group. Once the activities are over, each of them again follows their own individual distributions. Fig. 4.24 and 4.25 show the group behaviour for the group types of Department-mates and Colleagues.

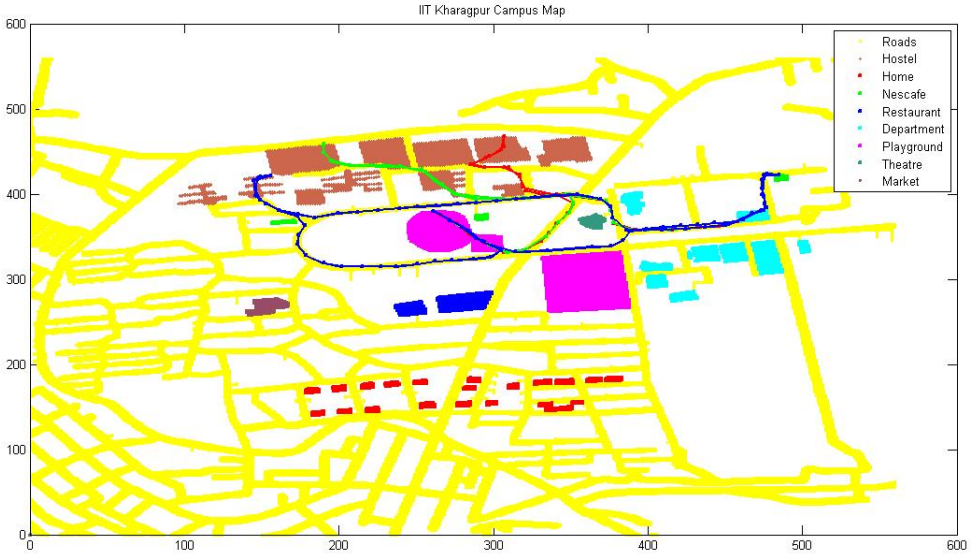



Figure 4.24: Group Behaviour: Dept. Mates

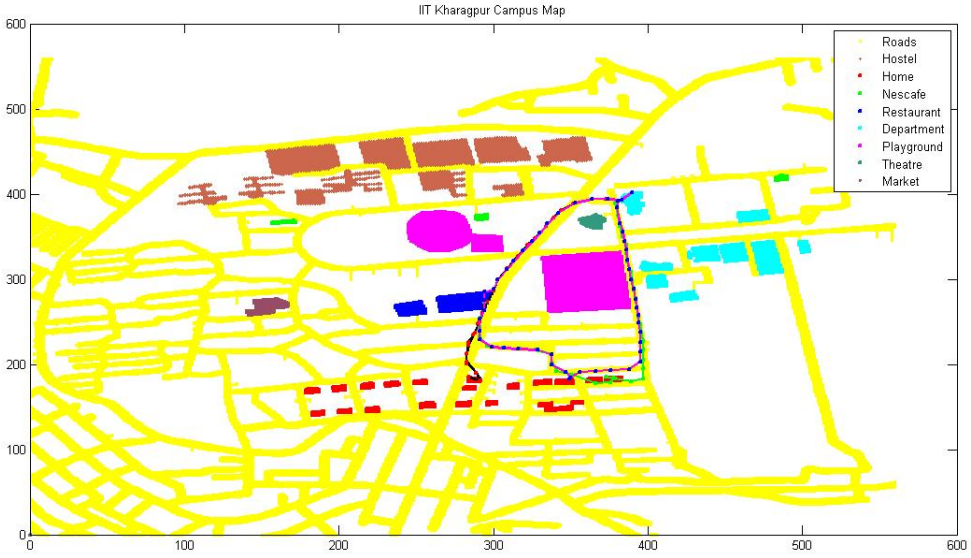



Figure 4.25: Group Behaviour: Colleagues

Table 4.16: Hostel Mates: Generic Behaviour
Total No. of Days = 30, R.M.S Error = 0.033

| Generic Behaviour | Specified Probability | Observed No. of Days Chosen | Observed Probability | Error |
|---|---|---|---|---|
| Nescafe-Going | 0.6 | 17 | 0.567 | 0.033 |
| Playground-Going | 0.4 | 13 | 0.433 | 0.033 |

Table 4.17: Hostel Mates: Nescafe-Going Behaviour
Total No. of Days = 30, R.M.S Error = 0.0713

| Behaviour Sequence | Specified Probability | Observed No. of Days Chosen | Observed Probability | Error |
|---|---|---|---|---|
| Nescafe → Theatre | 0.1 | 2 | 0.118 | 0.018 |
| Nescafe → Market | 0.1 | 3 | 0.176 | 0.076 |
| Nescafe | 0.8 | 12 | 0.706 | 0.094 |

**Analysis of Paths**

Fig. 4.24 represents the generated path corresponding to the group behaviour of three department mates living in different hostels. All three of them have a class in the department at 9:30 AM. They all start from their homes independently so as to meet near Vidhan Chowk at 9:15 AM. Student 1 stays nearest and takes only 6 minutes to reach the meeting point by cycle. Student 2 takes 12 minutes, and Student 3 takes 15 minutes by cycle. So, they start from their hostels at 9:09 AM, 9:03 AM and 9:00 AM respectively. At 9:15 AM, they meet and start for the department together. It takes them another 9 minutes to reach the department. They stay in the department till 11:30 AM at which point they go to the nearby nescafe for some food. The entire trip lasts for 50 minutes. They stay in the department till 4:30 PM in the evening. Then, they go to the playground to play some games. They take 18 minutes to reach the playground from the department. They stay there for 70 minutes and finally return to their hostels. Users 1 and 2, return home by the same route, while user 3 takes a different shortest route. Similar analysis can be done for the behaviour of a group of colleagues (see Fig. 4.25) who go to the department followed by a restaurant visit in the evening and then return to their homes.

Table 4.18: Hostel Mates: Playground-Going Behaviour
Total No. of Days = 30, R.M.S Error = 0.0287

| Behaviour Sequence | Specified Probability | Observed No. of Days Chosen | Observed Probability | Error |
|---|---|---|---|---|
| Nescafe → Playg. | 0.3 | 4 | 0.308 | 0.008 |
| Playg. → Nescafe | 0.5 | 6 | 0.462 | 0.038 |
| Playg. | 0.2 | 3 | 0.231 | 0.031 |

Table 4.19: Department Mates: Generic Behaviour
Total No. of Days = 30, R.M.S Error = 0.033

| Generic Behaviour | Specified Probability | Observed No. of Days Chosen | Observed Probability | Error |
|---|---|---|---|---|
| Department-Going | 0.7 | 22 | 0.733 | 0.033 |
| Playground-Going | 0.3 | 8 | 0.267 | 0.033 |

### 4.6.3 Validation

For the purpose of validating groups, we consider a period of one month. We count the number of days a particular generic behaviour and a corresponding activity sequence from that behaviour is chosen, and compare it with the specified choice probabilities.Tables 4.16 - 4.18 and 4.19 - 4.21 provide the validation for the group types of hostel mates and dept. mates respectively. A technique similar to the individual validation is used. We divide the no. of days a particular behaviour sequence is done by the total no. of days that generic behaviour is done to obtain the observed probabilities. Once again, the RMS error values are quite low for each behaviour sequence implying that the generated group behaviour is consistent with the input specifications.

Table 4.20: Department Mates: Department-Going Behaviour
Total No. of Days = 30, R.M.S Error = 0.0589

| Behaviour Sequence | Specified Probability | Observed No. of Days Chosen | Observed Probability | Error |
|---|---|---|---|---|
| Dept. | 0.75 | 16 | 0.727 | 0.023 |
| Dept. → Nescafe | 0.20 | 3 | 0.136 | 0.064 |
| Dept. → Playg. | 0.05 | 3 | 0.136 | 0.076 |

Table 4.21: Department Mates: Playground-Going Behaviour
Total No. of Days = 30, R.M.S Error = 0.089

| Behaviour Sequence | Specified Probability | Observed No. of Days Chosen | Observed Probability | Error |
|---|---|---|---|---|
| Nescafe → Playg. | 0.3 | 2 | 0.25 | 0.05 |
| Playg. → Nescafe | 0.5 | 5 | 0.625 | 0.125 |
| Playg. | 0.2 | 1 | 0.125 | 0.075 |

# Chapter 5

# Conclusions and Future Work

## 5.1 Conclusions

The proposed framework for modelling individual and group behaviour of a mobile user has established the basis for designing a realistic simulator producing readily-available spatio-temporal data. Currently, our simulator is generating individual user behaviour for 1000 users belonging to two user-classes, namely office-goers and college-students, on the IIT Kharagpur campus map. The reported data is of the form $\langle x, y, t \rangle$, at intervals of 3 minutes each for a period of one month. The validation process of both individual data and group data suggest that they are consistent with the input specifications. Listed below are a few tasks that are essential for the completion of a comprehensive spatio-temporal simulation framework.

## 5.2 Future Work

In this work, we have performed the data generation and validation of individual and group behaviour. Several experiments have been done on combining the two behaviours but a systematic validation procedure is yet to be developed.

From the point of view of making the simulator robust and extensive, the following tasks need to be carried out.

- *Combining Group and Individual Behaviours.* In this work, we have performed the data generation and validation of individual and group behaviour. Several experiments have been done on combining the two behaviours but a systematic validation procedure is yet to be developed. The

task of resolving clashes among individual vs group and group vs group has to be handled quite subtly. More experimentation needs to be done with regards to fixing the overriding probabilities. After thorough examination of different cases, we will be able to combine the two types of behaviour elegantly.

- *Integrating Different Classes.* The implementation of the simulator currently has only two classes of users. We should start focussing on a wide range of user classes and the behaviour of each class needs to be seamlessly integrated into the simulator design. The specifications for each class should be exhaustive and capture a wide variety of user behaviours. Consistency checks should be defined more thoroughly so as to effectively validate the complex combination of individuals and groups comprising users of different classes.

- *Identifying Events.* We have not yet looked at the event-driven behaviour of an individual. To model this type of behaviour, we need to define events properly. Also, we need algorithms to identify when an event is taking place, so that we can generate user behaviour in response to this. To achieve this, we need to know what are the different types of events, what are the various parameters that distinguish between events, how many people need to be present for an event and several such questions.

- *Extending the Simulation Area.* Our current version of the simulator is implementing the user behaviour in only a particular city region. We would like to extend this idea to capture inter-city behaviour. Indeed it would be nice to have the group behaviour of a set of friends, mapping their entire holiday plan on the India road map.

- *Obtaining Actual Data.*One of the important tasks is to obtain some existing real-world data, if available. This would help us validate our developed user model comprising regular, irregular and event-driven behaviour, and provide us with a clear picture regarding any modifications that need to be made.

To achieve the ultimate objective of the project, i.e. to design a location-based service like a recommendation system, several tasks need to be carried out.

- *Analysis of Spatio-temporal Data.* Once the simulator is generating data, we can provide results regarding no. of people visiting a hotspot - whether they visited alone or as part of a group, the duration spent at each hotspot, average number of hotspots visited by an individual during different days of

the week, etc. This will help us rank hotspots based on customer popularity, dependance on the day of the week, and any other features that can be explored by a recommendations service.

- *Mining Information.* The task of designing a recommendation service is intricately intertwined with the tasks of mining the profile of a user. The mining process consists of mining both the existing information on the user as well as his dynamic spatio-temporal information. While the first part can be done offline, the second part needs a great deal of online processing. Data needed for offline processing should be carefully chosen, and properly mined to obtain the necessary information. Also, the real-time spatio-temporal data should be processed carefully and efficiently whilst keeping in mind the level of detail needed. Existing mining algorithms may have to be tweaked to fit into our problem specifications.

We certainly hope that an effective location-based recommendation scheme can be designed in the near future, by a collective effort on both the mining and simulation fronts.

# Bibliography

[1] G. Yava, D. Katsaros. O.Ulusoy, and Y. Manolopoulus. A Data Mining Approach for Location Prediction in Mobile Environments. *Data and Knowledge Engineering, vol. 54, no. 2*, Aug. 2005, pp. 121-146.

[2] David Mountain and Jonathan Raper. Modelling human spatio-temporal behaviour: A challenge for location-based services. *Geographic Information Science Group, Dept Information Science, City University, London*

[3] Thi Hong Nhan Vu, Jun Wook Lee, and Keun Ho Ryu. Spatiotemporal Pattern Mining Technique for Location-Based Service System. *ETRI Journal, Volume 30, Number 3*, June 2008

[4] Sherry Y. Chen, Robert D. Macredie and Xiaouhi Liu, Brunel University and Alistair Sutcliffe, Univeristy of Manchester, UK. Editorial: Data Mining for Understanding User Needs. *ACM Transactions on Computer-Human Interaction*, Vol. 17, No. 1, Article 1, March 2010.

[5] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad-hoc network research. *Wireless Communications and Mobile Computing (WCMC) Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*,2(5) 483502, 2002.

[6] Qunwei Zheng, Xiaoyan Hong, and Sibabrata Ray. Recent Advances in Mobility Modeling for Mobile Ad Hoc Network Research. *In ACMSE, Alabama, USA*, 2004, pp. 70-75.

[7] X. Hong, M. Gerla, G. Pei, and C. Chiang. A group mobility model for ad hoc wireless networks. *In Proceedings of the ACM International Workshop on Modeling and Simulation of Wireless and Mobile Systems (MSWiM)*, August 1999.

[8] Marta C. Gonzalez, Cesar A. Hidalgo and Albert Barabasi. Understanding individual human mobility patterns. *Physical Review Letters*, 2002

[9] F. Ekman, A. Keranen, J. Karvo, and J. Ott. Working day movement model. *In Proceedings of First ACM SIGMOBILE International Workshop onMobility Models for Networking Research*, May 2008.

[10] Shiqing Shen, Ningning Cheng, Guihai Chen. A behavior pattern based mobility simulation framework for office environments. *Proceedings of the 2009 IEEE conference on Wireless Communications & Networking Conference*, p.2379-2384, April 05-08, 2009, Budapest, Hungary

[11] M. Musolesi, and C. Mascolo. A community based mobility model for ad hoc network research. *In Proceeding of the Second International Workshop on Multi-hop Ad Hoc Networks*, May 2006.

[12] M. E. J. Newman and Juyong Park. Why social networks are different from other types of networks. *PHYSICAL REVIEW E 68*,September 2003.

[13] D. Kotz and T. Henderson. CRAWDAD: A community resource for archiving wireless data at Dartmouth. *IEEE Pervasive Computing*, 4(4):1214, 2005.

[14] M. Kim, D. Kotz, and S. Kim. Extracting a mobility model from real user traces. *In Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFO-III. Conclusions COM)*, Pages 113, April 2006.

[15] Chris Walsh, Arta Doci, Tracy Camp. A Call to Arms: It's time for REAL Mobility Models. *ACM SIGMOBILE Mobile Computing and Communications Review*, Vol. 12, No. 1. (January 2008), pp. 34-36.

[16] R. Baumann, F. Legendre, and P. Sommer. Generic Mobility Simulation Framework(GMSF). *In Proceedings of First ACM SIG-MOBILE International Workshop onMobility Models for Networking Research*, May 2008.

[17] Alberto Medina, Gonca Gursun, Prithwish Basu, Ibrahim Matta. On the Universal Generation of Mobility Models. *18th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp.444-446, 2010