

Agent Mobility Model based on Street Centrality

Rupert Leigh
Department of Geomatics
University of Melbourne

ABSTRACT

Current agent mobility models are based on random movement. This means that the probability that a particular agent will be at a particular place is the same over the whole network. In contrast, we are trying to simulate an agent movement that reflects more realistically the distribution of vehicles in an urban street network. This can be achieved by moving away from the problem of an equal probability to a more unequal probability that is based on betweenness centrality, such that agents have a definite preference to traveling on a particular path.

Introduction

This research is aiming at simulating agent movement with the use of a mobility model that more realistically reflects the distribution of vehicles in an urban street network. The research is progressing from a model which uses a network with an equal probability distribution to a model which uses a network with an unequal probability distribution. This shows that agents can have a preference to traveling on a particular path as opposed to another path considered.

Currently, the mobility model is used in a shared ride trip simulation to supply way-finding strategies to host and client agents. The simulation evaluates the idea that a client can share rides with hosts to travel to its destination. Presently these agents move accordingly to a random walk mobility model that produces an equal distribution of agents occurring within the network. Consequently, there is an equal likelihood that an agent will be present at any location within the network. This is unrealistic, as it does not accurately reflect the actions in the real world, where a large volume of vehicles would be more likely to travel along main streets than they would along smaller streets.

To allow this shared ride trip simulation to use a more realistic network and therefore produce results with greater accuracy, a mobility model that considers this problem is in demand. The mobility model needs to distribute agents unequally within the given network. It needs to reflect suitably the movement of traffic, such that main streets will have a larger volume of vehicles and smaller streets will have lesser volumes. The unequal distribution will be calculated by centrality measures based on a named street network.

It is expected that it will be possible to implement a mobility model that is effective in achieving an unequal distribution of agents within the network, based on centrality measures, and will supply way-finding strategies to a shared ride trip simulation.

The model will be applied to two types of agents, the hosts and the clients. It will change their decision making ability to reflect knowledge of the street network. It is expected that by providing the hosts and the clients knowledge about the network, it should reduce the average length of the shared trips the client participates in.

In the next section, we will discuss the background information regarding the existing shared ride simulation, related centrality measures and mobility models and associated network theory being considered. Then we will discuss the concept of this research and the ideas behind putting this work into practice. Next we will discuss the implementation of the ideas discussed and the outcome of some preliminary work. We will then see the results produced by the methods implemented along with an interpretation of the results. Finally, we shall discuss reasoning behind the results along with further comments.

BACKGROUND

Existing Simulation

Current research is investigating the implementation of a shared ride trip planning simulation environment (Winter et al., 2005, Gaisbauer and Winter, 2006). As mentioned previously, the simulating evaluates the idea that a client can share rides with hosts to travel to its destination. Within this environment, there are two types of

agents, clients and hosts. Hosts are in control of a vehicle and have the capacity to carry clients whereas a client can ride with different hosts for different segments of the journey.

An overview of the current simulation program, which is written in Java, is as follows. It works by having a number of hosts travelling around the grid based on the random walk mobility model (discussed later) called hosts. These hosts are able to pick up a client from a particular location and carry them along edges for a certain distance. The distance is dependant on the hosts route and the clients intentions. There are three classes of hosts in the simulation. All the hosts have limited capacity, and therefore can only carry a certain number of passengers.

Taxis

Taxis move randomly, according to the host mobility model, searching for clients (fares) and when occupied, will then drive directly to the clients requested destination. Although taxis can take a client directly to wherever it wants to go, the result of this is a higher cost.

Private cars

A private car will have a route generated randomly using the host mobility model before the car starts to move on the network and will not deviate from this route to pick up a client. For hosts to give a client a lift, the client needs to be located on the cars route.

Buses

Busses are part of the public transport network, therefore they have a specified route and timetable to follow. This assumes that busses are the cheapest form of assisted transport (apart from walking). Clients are aware of the route information.

There are a number of parameters within the simulation. These parameters allow different functionality within the simulation accordingly. A list of the main parameters used in this research are as follows:

- Number of simulation executions
- Total number of hosts
- Number of taxis
- Number of cars
- Number of busses
- Communication range of the clients
- Grid dimensions
- Clients' ability to walk

Clients essentially have four modes of travelling, that is, lifts by private cars, lifts by taxis, lifts by busses or by walking. All forms of motorised transport move at the same speed of one edge per time interval, whereas the walking speed is set to one-quarter edge per time interval. In the situation where a client chooses to walk, the client will move partially along an edge at each time interval. When the client is partially along

an edge, it cannot accept lifts from hosts because they can only accept rides at nodes within the network.

The simulation is time dependant. At each time interval, a series of processes occur. A sequence of steps is as follows. Firstly, the simulation checks to see if the number of hosts is less than the specified parameter. If there are less, then the simulation will generate a new host with an accompanying predetermined route. This process of checking and creating hosts repeats until the simulation obtains the required number of hosts. Secondly, a client will broadcast a message to request the location and route information of hosts within a specified communication range. The available hosts respond and the client gathers this information. At this point, the client will decide which host, if any, could take it closest to the destination node of the client. Currently, the method used is a modified A* Shortest Path algorithm, known as the Lifelong Planning A* algorithm (Koenig et al., 2004) which uses the geodetic distance as a heuristic function. The algorithm will calculate the potential distance to the available node offered by the host, the g-value, and then the heuristic distance to the destination node from that node offered by the host, the h-value. The algorithm returns the node that will minimise the sum of both the g-value and the h-value.

Mobility Models

Currently, the simulation uses a random walk mobility model to generate routes for hosts (Boleng et al., 2002). The network used is a regular grid, where the edges represent roads for the agents to travel along and the nodes represent the intersections of the roads. The mobility model uses a random function to select one out of five possibilities for the next node of the hosts' path. These five possibilities are north, south, east, west or to be stationary. There are fewer possible outcomes if the current

node is situated along a boundary as an agents movement cannot extend beyond the limits of the network. Therefore, if this occurs, the process repeats until a node is generated that does not extend beyond the boundary. The nodes produced by the mobility model are not dependant on the location of the agent within the network.

A client is subject to the same random movement a host, resulting from the use of same mobility model. A number of different algorithms can be used for travel planning such as Dijkstra's shortest path and Lifelong Planning A* (Koenig et al., 2004). Ensuring that advancement towards the client's destination is achieved.

Centrality Measures

Centrality has many measures, three main measures include Betweenness Centrality, Closeness Centrality and Degree Centrality (Crucitti et al., 2005b, Crucitti et al., 2005a). Centrality is used to determine the relative importance of a node with respect to the other nodes within the network. In other words, it ranks each node by giving it a structural significance with respect to the rest of the network. Therefore, it is assumed that streets with a higher centrality value will be of more importance hence the higher likelihood that it will attract a higher volume of traffic.

Degree Centrality

Degree centrality is based on the number of edges a node has adjacent to it.

Hence the degree of node i is dependant on the adjacency matrix

$k_i = \sum_{j \in N} a_{ij}$ that is calculated in the following manner (Crucitti et al., 2005b):

$$C_i^D = \frac{k_i}{N-1} = \frac{\sum_{j \in n} a_{ij}}{N-1} \quad (1)$$

Closeness Centrality

“The simplest notion of closeness is based on the concept of minimum distance or geodesic d_{ij} ” (Crucitti et al., 2005b). It is calculated the following at node i :

$$C_i^C = \frac{N-1}{\sum_{\substack{j \in N \\ j \neq i}} d_{ij}} \quad (2)$$

Betweenness Centrality

This considers nodes that occur on many paths between other nodes of higher importance and hence a higher betweenness value (Crucitti et al., 2005b).

$$C_i^B = \frac{1}{(N-1)(N-2)} \cdot \sum_{\substack{j, k \in N \\ j \neq k; j, k \neq i}} \frac{n_{jk}(i)}{n_{jk}} \quad (3)$$

Network Theory

Two prominent methods of interpreting urban street networks are the Primal and Dual approach (Crucitti et al., 2004, Crucitti et al., 2005b). These methods are used to interpret the network topology. The Primal approach is able to “preserve a metric, geographic concept of distance, without abandoning the topology of the system” (Crucitti et al., 2005b). In a street network, it essentially turns the streets into edges

and intersections into nodes. As a consequence it is one of the simplest ways to encapsulate the distance factor from the geographic layout (Crucitti et al., 2004).

In contrast, the Dual approach is more abstract which “leads to a loss of any reference to geographic distance, but unlike other models it is purely spatial” (Crucitti et al., 2004). Used in a street network, though not limited to, it essentially turns each intersection into an edge and the streets into nodes. This approach can be modified slightly to encapsulate the concept of a named street (Crucitti et al., 2005a). The named street approach is to represent named streets as nodes and street intersections as edges. This technique allows known network analysis methods to work on these different graphs.

CONCEPT

The goal of this research is to create a mobility model that is able to influence the distribution of agents within a network in an unequal and more realistic manner. As mentioned previously, it is aimed at simulating a more realistic situation where main streets have a larger volume of traffic than those of smaller streets. This is true for the assumption that hosts prefer central streets. Here, we will investigate the idea that network centrality can be used to model streets within a network. These values will become part of the influence providing the unequal distribution. The centrality measures used namely will be betweenness centrality, closeness centrality and degree centrality.

First, a modified test grid needs to be created. It needs to differ from the regular grid so that when different centrality measures are performed, the results are unique and

distributed unequally. To base this on a more realistic scenario, grid segments will be combined to form larger units of named streets such that on top of a regular grid network an irregular network of named streets is generated. The streets will vary in lengths and in position and direction related to the grid: In effect, producing an asymmetric network. Later we will discuss the implementation of the named street approach to achieve this effect.

Giving a host knowledge of the network is crucial for it to move accordingly to the main streets within the network. This will assert the assumption that hosts will be more likely to move on central streets than those which are not. Knowledge of these central streets will be integrated into the mobility model which will have been derived from values calculated by each of the observed centrality measures. It will show that hosts have a noticeable preference to travelling on streets which are more central.

Clients too need knowledge of the network for route planning purposes. This allows them to calculate the most beneficial route from the offers currently on hand at any particular time. In this case, likely location of hosts would be known, where it would be more probable for the client to obtain a lift. In other words, clients should use the mobility model used by the hosts to determine routes so that they have the ability to anticipate the behaviour of the hosts.

The mobility model will be divided into three different sections. First the centrality values calculated from of the network. Second, to create routines that will be used to influence a hosts behaviour. This will be achieved by creating a weighted network based on the centrality values of the network. The weighting will be a based on the

surrounding edge weights at hand. It will consider the possible directions to the next node and from here, generate a normalised number for each direction. Each of these numbers can then be chosen at random, thus the higher the number, the higher the probability of going in that direction. This process does not eliminate the possibility of going to any of the available nodes. This will ideally represent a more realistic idea, that vehicles use all streets, that it is just the number of agents that use main streets compared to those that use smaller streets is offset accordingly.

Third, the client will use the values of the network to show likely locations of hosts within the network. This means that the two operations are independent, clients do not know where the hosts will be, likely locations of where they would be. In relation to the case at hand, clients are waiting for hosts to offer a suitable trip, where they will take a lift with the host to a better location within the network. A better location will be defined as a location that is likely to have a greater possibility that another suitable host will pass by. In the previous model, the geodetic distance to the destination node is used to determine if a particular route offered by a host is of any value to the client. In this situation, we will not use distance to determine the ranking of nodes, but to use, what is called, a probable waiting time for hosts. The probable waiting time is calculated based on the knowledge that a host would use to randomly determine their route.

IMPLEMENTATION

Test Network

Based on preliminary findings that the network responded with symmetric values when acted on by symmetric or regular changes, it was decided to create an

asymmetric which as a result would be a distorted graph. To escape the regularity of previously implemented test network, a network of 10×10 edges was created that vaguely represented an arbitrary street network. The network was drawn using multiple line segments at a number of locations. Some edges were joined together to create a unique asymmetrical network of a number of named streets, which form the main streets that are intended to have more traffic on them since they are likely to be more central.

Shown below in Figure 1, we see the Primal network as it exists before being converted to a Dual representation. The graph has been based on the named street approach proposed by Jiang and Claramunt (2004). The reason for this is so that the graph can be interpreted by using the centrality measures that can only operate on with the topology of Primal networks. Each edge in the network either has been labelled with a number or lettered pair. The numbered edges, from 1 to 171, represent the normal single edge streets, where the lettered paired edges, “AA” to “AI”, represent the named streets.

	1	2	3	4	5	6	7	8	9	10
11	12	13	14 AA	15	16	17		18	19	20
							21	22	23	24
25	26	27	28 AE	29	30	31		32	33	34
									35	36
39	40	41	42		43	44 B A	45		46	47
56	57	58	59		60	61		62 C A	63	64
74	75	76	77	78 L A	79			80	81	82
		AG								
87	88	89	90 D A		91	92	93	94	95	96
105	106	107	108		109	110	111	112	113	114
			AH							
121	122	123	124		125	126	127 AF	128	129	130
135	136	137 AI	138	139 L A	140	141	142	143	144	145
153	154	155 I A	156	157	158	159	160	161	162	163

Figure 1 – Test network containing named streets

Network Analysis

To calculate the centrality values for the test network, a program called Pajek (version 1.14) is used. For each graph, a file containing the nodes (referred to as vertices within the Pajek environment) and the edges of the network was created in the following form using the syntax and structure shown:

```
*Vertices NumberOfVertices
1 "1" x y z
↓
n "1" x y z
```

```
where:
  n is the next consecutive number,
  l is the label of the vertex,
  x,y,z are the screen coordinates of the node with
    range [0,1].

*Edges
i j
↓
...

where:
  i and j are verticis that have common edge.
```

Figure 2 – Pajek file format

Pajek uses an alternative algorithm for calculating betweenness centrality. It is based on the faster algorithm for betweenness values proposed by Brandes (2001).

From the graphs produced of each centrality measure, betweenness was shown to have the greatest relative change in values for each node. This is seen in a visual representation in the following Figure 3 and Figure 4. A reason for this could be that the values, as per the formula, are normalised as part of the calculation, whereas the closeness values are not. From this point forward, only the betweenness centrality is considered.

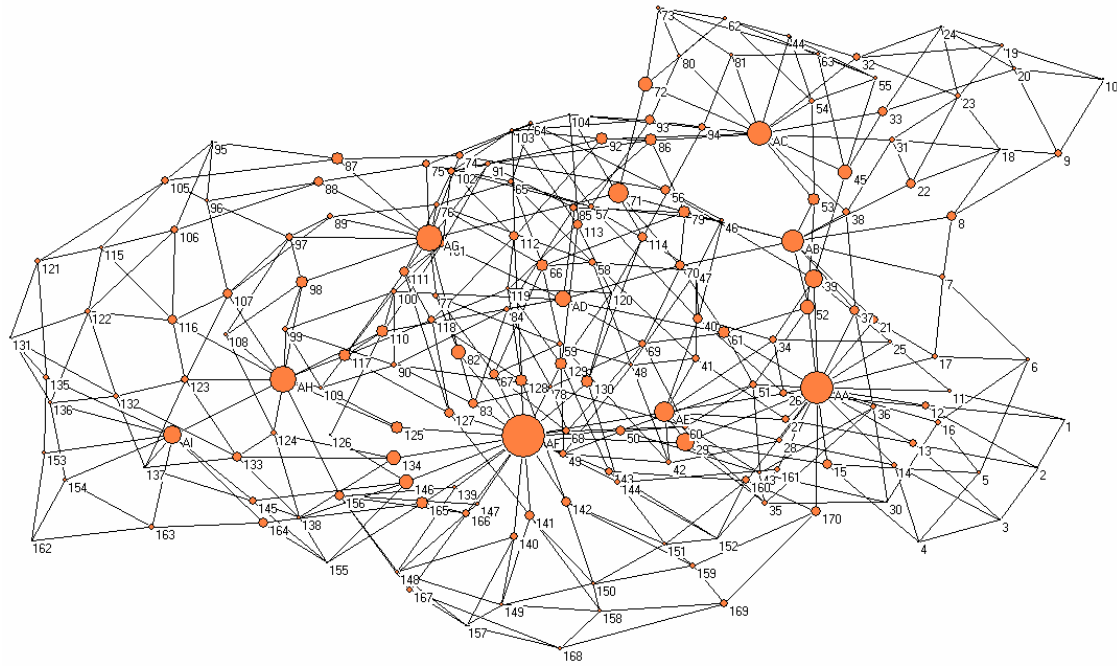


Figure 3 – Pajek drawing of betweenness values of network

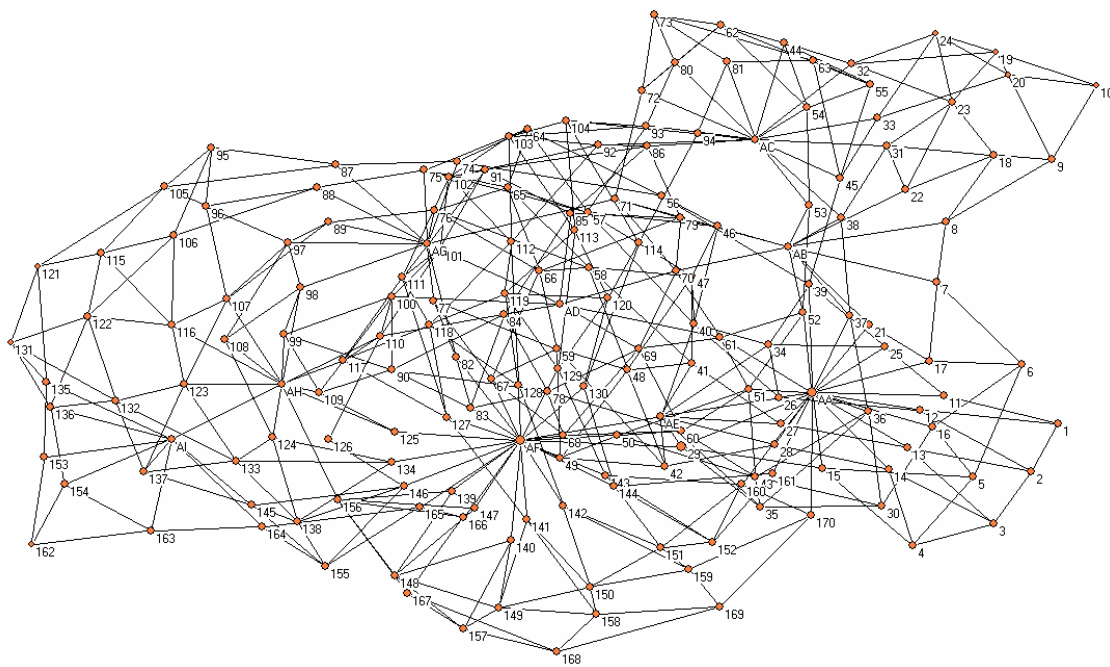


Figure 4 – Pajek drawing of closeness values of network

At this point, while looking at Figure 3, we can see that the named streets have are significantly larger in general than that of the numbered streets when comparing to Figure 1. We can also see that some numbered street segments such as 71, though not

a named street has a relatively large value compared to other numbered streets. This could be because there are two named streets connected to either end of it.

Mobility Model

The mobility model has been implemented in Java so that it can be easily integrated with the existing simulation. It has been developed independently from the simulation so that it can be run in isolation for testing purposes or for any future integration into other environments.

The mobility model implements the following rules for the agents.

- agents cannot move beyond the boundaries of the network
- agents are always moving, no stationary position
- agents cannot move to their last location
- agents only have local knowledge of the network
- hosts randomly choose their next position, but with an offset probability
- clients have local knowledge of where hosts will likely be

First, a multi-dimensional array is initialised so that it can represent the Primal graph created in Section 4.1. The betweenness centrality values calculated with Pajek in section 4.2 need to populate the designated array. In other words, the edges which became nodes have values associated with them now become the edges again but with the values of the nodes. Through this process, the network values are transformed from the Dual representation back to the Primal representation, thus giving the streets a range of values which we will call weights.

This process also allows us to reference the array based on arbitrary grid coordinates of the graph. The grid was referenced with the bottom left node being (0,0) and the top right node being (10,10). The graph is now a non-directional Primal representation containing betweenness centrality values for the streets within the network. All 220 edge values were statically coded by entering the information manually. This was a result of not having an automated system in doing so.

There are two possibilities to implement the mobility model, first the agent could decide its action given information about the network, or the mobility model could give the agent a position to move to based on its current location. The later was decided, as it offered more central control based within the model itself rather than the agent. Furthermore, the existing environment used similar functions to generate new random positions for the agent thus it would be less complicated approach.

Second, host movement is considered. A function to control host movement is created. Two methods for the function are specified, one that takes a single argument and one that takes two arguments and both return a new position. If a single argument is provided, which is the current position of the host, then the function will consider all adjacent nodes as possible next nodes. Otherwise, if the function is given two arguments, the current position and the previous position, then the function will consider all adjacent nodes, except the previous one, as possible next nodes. Thus providing a new position of a node relative to the current node if given one argument, or relative to the current and previous node if given two arguments.

The function uses the information stored in the multi-dimensional array previously mentioned. It first determines the direction of movement the agent is heading given the arguments. If there was only one argument then the direction is determined as stationary, otherwise the direction is derived from the current and previous nodes. The betweenness centrality value for each direction is obtained. When obtaining the weight for each direction, the following conditions are tested and if true, the weight is automatically set to zero:

1. If the direction of movement will be beyond the bounds of the graph.
2. If the direction of movement returns to the previous node.

Thus, effectively eliminating these erroneous directions from further calculations.

The process of the selection can be seen in the pseudo code provided in Figure 5. From the values obtained (nWeight, eWeight, wWeight, sWeight), including those that may have been set to zero, a sum is calculated. The values are then normalised, so that they sum to one. From here, a random number is generated producing a value r , $0.0 < r \leq 1.0$. This then will match one of the specified criteria and move accordingly. If one of the weights has been set to zero then it will play no role in the calculation. For example, if the north weight had been set to zero, then it would have produced a zero probability of choosing the north direction. Hence it will not influence following west weight calculation.

```

sumWeight = (sWeight + wWeight + nWeight + eWeight)

nProb = (nWeight / sumWeight)
eProb = (eWeight / sumWeight)
wProb = (wWeight / sumWeight)
sProb = (sWeight / sumWeight)

if (r <= nProb) { move north }
else if (r <= nProb + wProb) { move west }
else if (r <= nProb + wProb + eProb) { move east }
else if (r <= nProb + wProb + eProb + sProb) { move south }
else { error }

```

Figure 5 – Host movement pseudo code

Based on the various centrality values of each edge, the probability will be unequal, thus not providing simple random movement, but movement that has particular influence in one or more directions over others available.

Third, the clients are considered. Here, the clients are slightly different to the hosts as they need to consider route planning to minimise their travel time by maximising their chance of finding a suitable host. They need to know probable locations where hosts will likely to be. However, due to the type of calculations used, we need some form of time-based value. For this, a value called the probable waiting time for a client is used. We assume that at each node, it is independent from the surrounding nodes.

The probable waiting time calculates the likely time, in simulation time intervals, of how long the client is likely to wait to get a lift with a host along a specific edge. The first stage is similar to the calculation seen above in Figure 5. It calculates the sum of the surrounding weights followed by normalising each value. At this point, we calculate the inverse of the edge value to produce the probable weighting time (PWT) for that particular direction.

```
sumWeight = (sWeight + wWeight + nWeight + eWeight)

dProb = (dWeight / sumWeight)

where:
    dWeight and dProb represent the respective value in a
    particular direction in question

PWT = 1/dProb
```

Figure 6 - Host weighting

Now the client has a time based value, it will be put to use for planning purposes. It is integrated into two different parts of the simulation. First it is implemented into Dijkstra's shortest path algorithm. This was a relatively small change to the existing code. The change occurred where the algorithm searches from one node to the successor nodes. Here it calculates the PWT based on these two nodes and sets this as the weight for that particular edge.

The implementation of the Lifelong Planning A* algorithm is slightly more involved than the last, due to the number of places values for the edges can be updated. As mentioned before, the algorithm maintains two values, the potential distance from the start node to the current node known as the g-value and the estimated heuristic value from the current node to the finish node. Here we change these two values.

Initially these values were based on the worst time to get from one node to the other. These values were calculated based on the geodetic distance. The g-value became a sum of the previous g-value with the potential PWT calculated from node p to node n. The h-value estimate became the shortest path using the previously modified Dijkstra's algorithm. The changes were made to the following functions of the Lifelong Planning A* algorithm outlined in Figure 7.

```
insertNewNode
```

```

OLD
g = p.gValue + wW;
h = geodeticDistance(n, g) * wW;

NEW
g = p.gValue + PWT(p, n);
h = sPath.getTime(n, g);

getgValue
OLD
return nodeLst.get(sid).gValue;

NEW
return getNode(sid).gValue + PWT(getNode(sid),getNode(gid));

setrhsValue
OLD
rhs = minimum(rhs, geodeticDistance(s, n) * wW + wW);

NEW
rhs = minimum(rhs, getgValue(s.ID) + PWT(s, n));

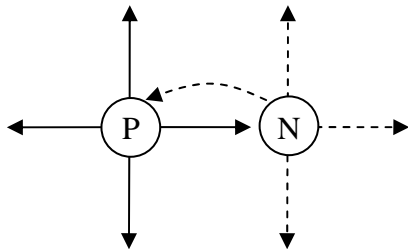
sethValue
OLD
p.hValue = minimum(p.hValue, geodeticDistance(g ,n ) * wW);

NEW
p.hValue = minimum(p.hValue, sPath.getTime(n, g) + PWT(p, n));

where:
    p.gValue = g-value of previous node
    sid = id of current node
    gid = id of goal node
    wW = worstWeight
geodeticDistance() = calculate geodetic distance between given
nodes
minimum() = calculate minimum between given values
sPath.getTime() = calculate shortest path between given nodes
PWT() = get probable waiting time for between given
nodes

```

Figure 7 – Changes to the LPA* Algorithm



5. RESULTS

Following are two visualisations of the host movement within the simulation. The following Figures 5, 6 and 7 are represented by showing the grid on a 45° tilt, so the left corner is node (0,0) and the right corner is node (10,10). The parameters used for the simulation to generate these were as follows:

- A random start position in the 10×10 grid
- A Predetermined route by using the random mobility model
- A route length of 10 segments
- 1000 simulations
- Displaying the average host frequency of each edge

The first, shown in Figure 8, is the average frequency of hosts at each edge in the network using the random walk mobility model. This shows the relatively equal distribution of hosts around the network. We see that there are slightly more hosts around the edge of the network than in the centre, and slightly more again in the corners of the network. We can explain this by the number of choices a host has when it is travelling within the network. In the centre of the network, all edges except the ones located along the boundary, have a choice of four connected edges to travel along whereas the boundary edges have a choice of three and the corners have a choice of two. Therefore, this increases the likelihood of travelling along these particular boundary edges and this is what we see.

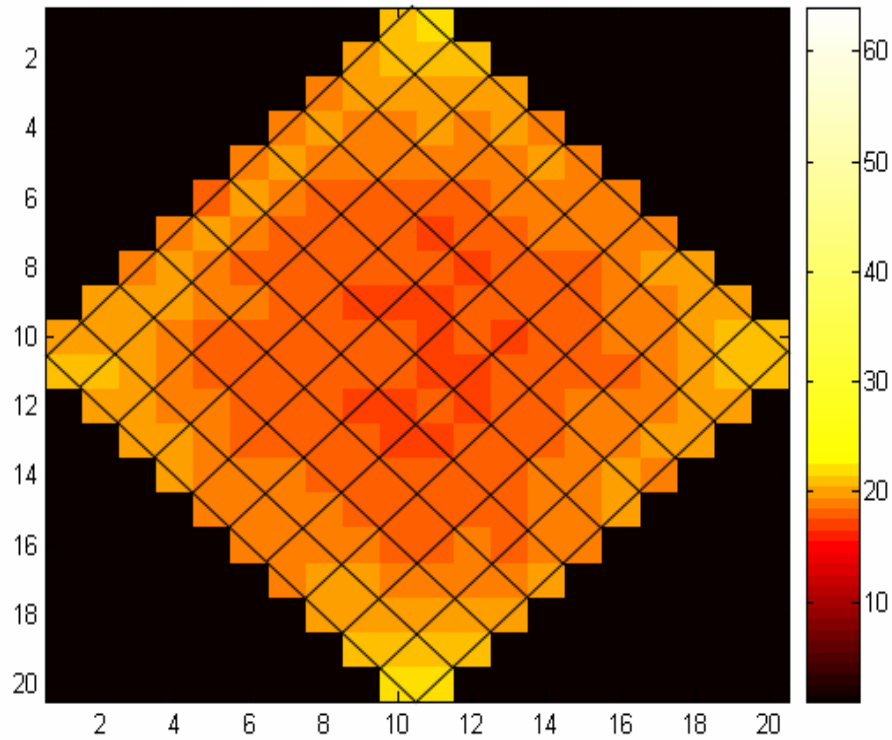


Figure 8 – Host frequency based on random mobility model

The second, shown in Figure 9, is the average host frequency at each edge in the network using the developed mobility model based on betweenness centrality values. We see that it shows an unequal distribution of hosts. It quite clearly represents the pattern of the inverted betweenness values shown in Figure 10 and the test network shown in Figure 1.

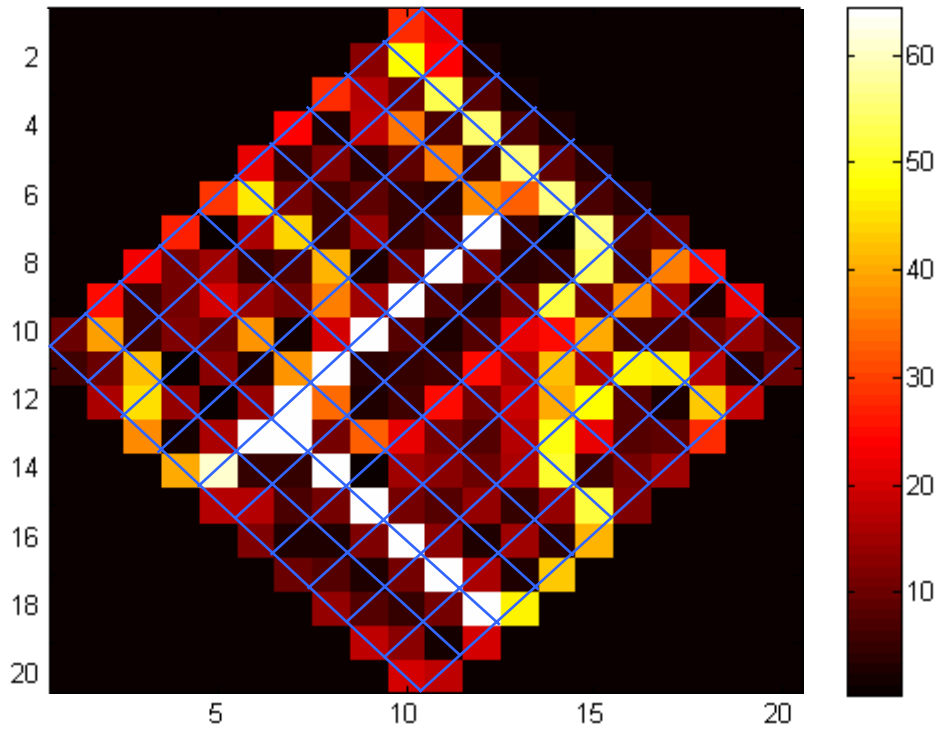


Figure 9 – Host frequency based on centrality mobility model

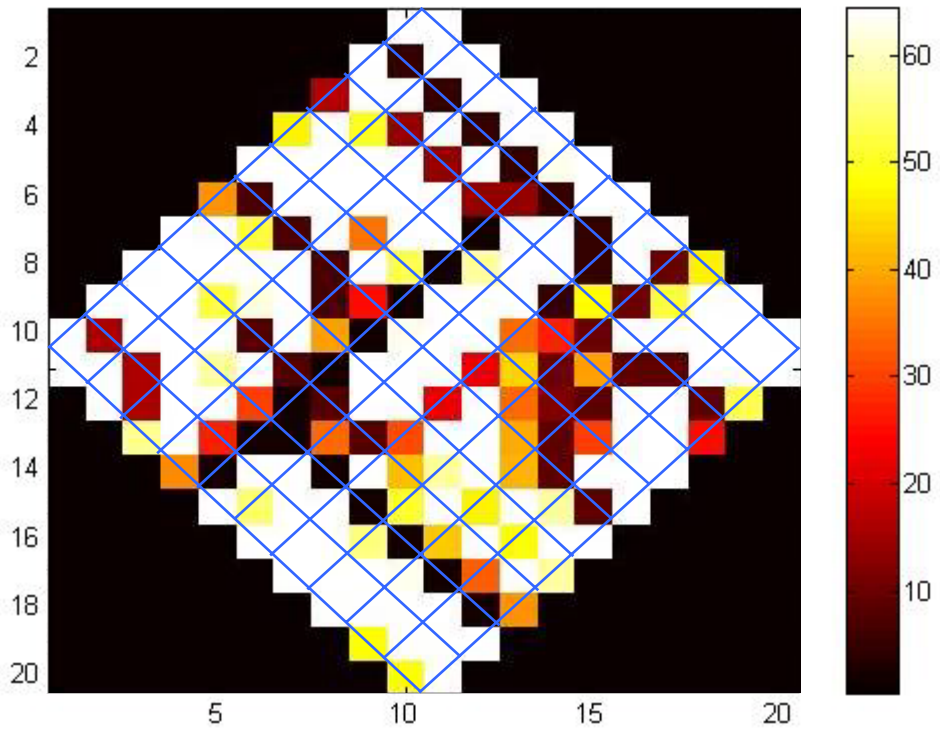


Figure 10 – The inverted weight of each edge

Following in Figure 11 and Figure 12 show the statistics for a simulation running with three different combinations of mobility models. Firstly with the original simulation,

secondly host movement from the betweenness centrality mobility model and client movement from original simulation and thirdly, both host and client using the betweenness centrality mobility model. The simulation used to following parameters each time for each mobility model combination:

- 1000 simulations
- Clients cannot walk
- Only private cars as hosts
- A 10×10 grid network
- Communication range of 4 segments
- Hosts start randomly with a predefined route
- 120 hosts
- Client travelling 10 segments from (0,5) to (10,5)

Decisions that agents make which the street network influences are said to have knowledge of this network. Therefore, hosts that have knowledge of the network will generally move around according to the streets and the weight each of these has, as shown previously in Figure 9. In addition, clients that have knowledge of the network will plan their trip based on likely locations of hosts that follow the network. We can see that knowledge of the street network has had a significant effect in the average travel time of a client, shown in Figure 11. It shows the results as expected. Following in Figure 12, we can see more clearly the distribution of travel times in comparison to different levels of knowledge.

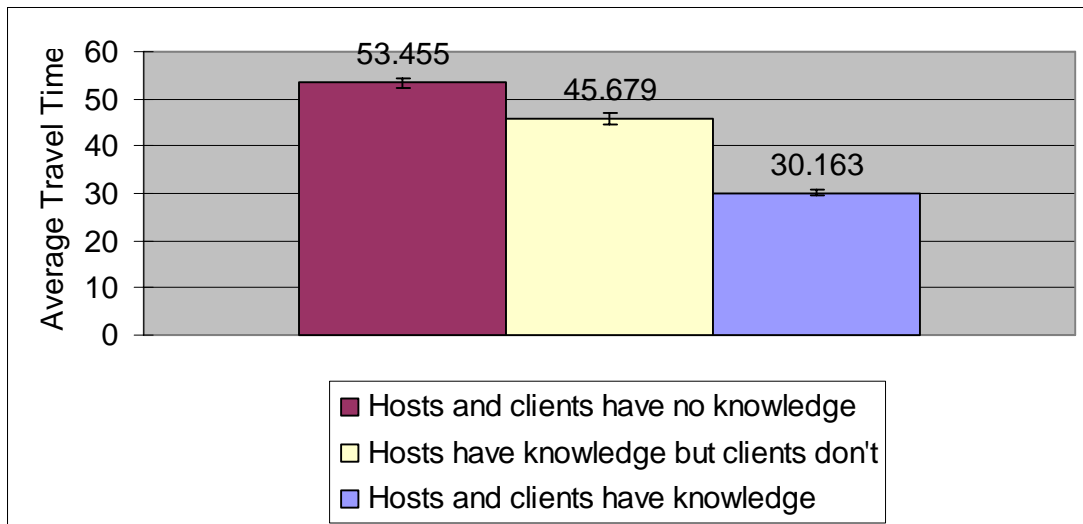


Figure 11 – Average travel time comparison

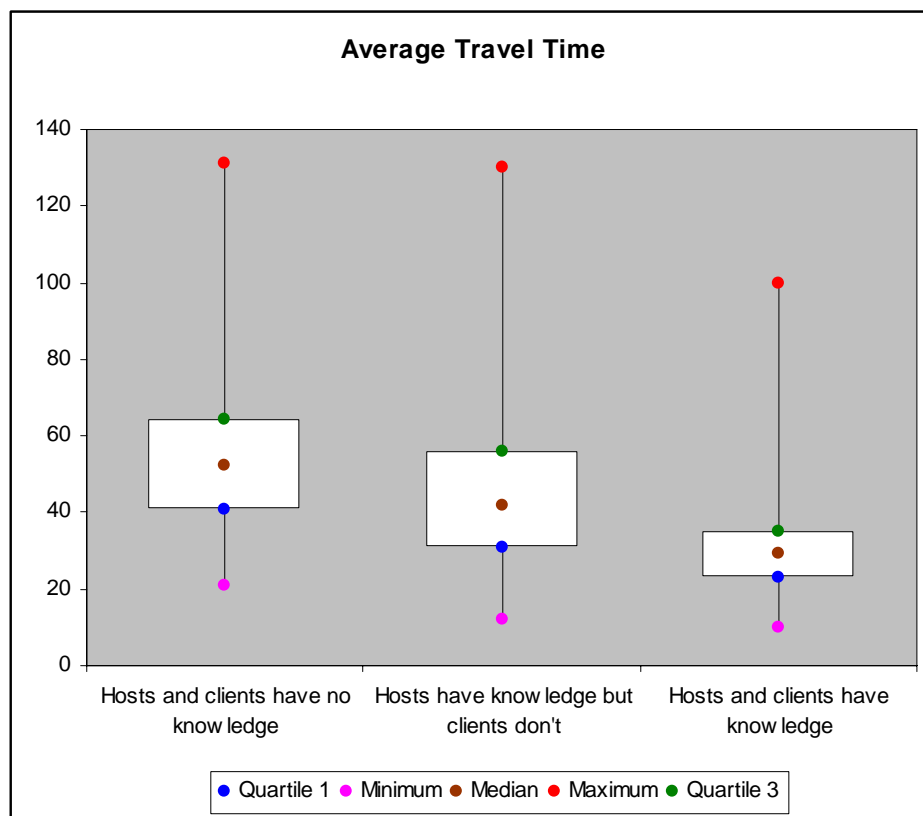


Figure 12 – Average travel time box plots

6. DISCUSSION

It is interesting to see the effects of introducing knowledge into the movement decisions of the agents. For hosts, it is simply distributing them unevenly across the network, which in this case was in relation to the named streets. Whereas for clients, it

has been slightly more complex since they use local knowledge to determine the probable waiting time for each edge. The client uses this information during the booking stage to determine the best route at that particular time interval along with the available knowledge of the location of the hosts. For example, if a client knows that two available hosts will intersect each other, while the first host can take the client to the second host. It will book both of these hosts, and thus travel with each host for the required journey.

As mentioned previously, all of the decisions made by both the hosts and clients are based on local knowledge at a node of the surrounding nodes. This consequently produces a new network structure that is bi-directional. This is because the weights are likely to be different depending on the direction of one node to the other.

The use of the probable waiting time is interesting when compared to the geodetic distance as a method for determining the best route to take. Both methods will use the same booking procedures, but as expected, they will work differently. The geodetic distance method simply calculates the distance from one node to another. When applied with the shortest path algorithm, it focuses on finding the most direct route to the destination. This method does not have knowledge of the street network or does not directly correlate with the random host movement described by the model. Whereas when the probable waiting time is calculated, it is based on the most probable location hosts are likely to be at, since it can be linked more closely with the hosts' actual movement. When integrated with the shortest path algorithm, the algorithm tries to minimise the total probable waiting time. In practice, the probable waiting time solution seems to be able to work around possible problems. For

example, the client wants to travel to a location that is three segments away from its current position. The first method will most likely produce a solution that involves travelling directly across the three segments. The second method may find a more probable path that is much longer in terms of travelling but may end up being quicker than waiting for a host to travel along the improbable route that the first method chose. This example could explain the reason behind high travel times shown in Figure 12. One reason why the average travel times may be higher when using the first method, is that the client may continue to stay at one particular node while waiting for the less likely host. This reasoning when applied to the second method, will state that the client could travel larger distances by taking detours, which in effect could produce the larger travel times recorded. However, as shown, the average time is reduced greatly when the probable waiting time use used by the client along with the appropriate host movement. From this observation, we can speculate that one of two things happen. Firstly, that the client takes fewer detours than the time taken if waiting for a direct ride to the destination. Secondly, that the client will more likely avoid these situations to begin with if using the probable waiting time.

A problem with the current implementation is that the estimates used for the heuristic part of the LPA* calculation can be considered an optimistic estimate considering it essentially calculates the shortest path of probable waiting times. An alternative to this estimate, maybe to take an average of the top five shortest paths to create a more realistic, but not pessimistic, estimate.

In hindsight, the model could be changed to better represent certain situations. For example, allowing the host to stop occasionally such as in the real world. Though, it

would not have a large possibility such as the previously implemented random walk model, which had a probability of 0.20. This could consider how many times a host has stopped on its journey and therefore be less likely to stop again.

Another situation occurs at the corner of a network, where if the host enters from one direction, the only possible movement once it reaches the corner node will be for it to continue to the single possible node. Exemptions to the mobility rule set could allow the agent to travel back to way it came. However, this is a small problem considering it happens a maximum of 8 out of 121 locations in the network.

One limiting factor of statically coding the values into the mobility model is that it is not easy to change the values if required to or for easy changes to the test grid. Therefore, it is desirable to develop an automatic approach to gathering and recording this data into the multi-dimensional array. This data can be entered through a three-step process:

- First would be to generate a graph which can be read by Pajek.
- Second would be to read this information into Pajek and generate centrality values determined by the centrality method
- Third would be to import the values generated by Pajek into Java (or whatever the programming language may be) so it can be used within the model.

7. CONCLUSION

We have seen a successful implementation of a mobility model that can distribute agents unequally within a network. These agents can also be distributed according to the betweenness centrality measure of a network containing named streets. When integrated into the shared ride simulation, we can see a significant improvement of the average travel time for a client to travel a 10-segment route, based on a series of parameters. As expected, when the hosts have knowledge of the street network and the client does not, we see a slight improvement in this average travel time. However, when both the hosts and the client have knowledge of the street network, we see that this average travel time is a significant amount more. This results in a more realistic scenario, that was aimed for.

We have also developed a new way of generating a waiting time that is the probable waiting time of clients. Although this method is in its early stages, and therefore may be looked upon as a primitive solution, it is a critical step forward. Overall, this research has successfully met its aims.

Bibliography

- Boleng, J., Camp, T. and Davies, V. (2002) *Wireless Communications and Mobile Computing*, **2**, 483-502.
- Brandes, U. (2001), 163-177.
- Crucitti, P., Latora, V. and Porta, S., (2004), *The network analysis of urban streets: a dual approach* [online]. Available from: arXiv:cond-mat/0411241 [22 April].
- Crucitti, P., Latora, V. and Porta, S., (2005a), *Centrality measures in spatial networks of urban streets* [online]. Available from: arXiv:physics/0504163
- Crucitti, P., Latora, V. and Porta, S., (2005b), *The network analysis of urban streets: a primal approach* [online]. Available from: arXiv:physics/0506009 [8 April].
- Gaisbauer, C. and Winter, S. (2006) Melbourne.
- Jiang, B. and Claramunt, C. (2004) *Geoinformatica*, **8**, 157-171.
- Koenig, S., Likhachev, M. and Furcy, D. (2004) *Artif. Intell.*, **155**, 93-146.
- Winter, S., Nittel, S., Nural, A. and Cao, T. (2005) In *Proceedings Symposium on Societies and Cities in the Age of Instant Access*(Ed, Miller, H.) Salt Lake City, Utah.