Probabilistic Travel Modeling using GPS

[1]Morris, S, [2]R. Gimblett and [1]K. Barnard

[1]University of Arizona, Department of Computer Science, E-Mail: smorris@cs.arizona.edu [2]University of Arizona, Department of Natural Resources

Keywords: *Recreation modeling; Agent-based modeling; GPS; GIS*

EXTENDED ABSTRACT

Recreation simulation modeling, when combined with intelligent monitoring, is becoming a valuable tool for natural resource managers. The goal of recreation simulation is to accurately model recreational use, both current and future. Models are applied to gain a thorough understanding of the characteristics of recreation. Indicator variables such as visitor experience, carrying capacity and impact on resources can be computed. If the model is valid it can be used to predict future use as well as to investigate the effect of new scenarios and management decisions.

Recent research has focused on agent-based modeling techniques. Recreators are represented by autonomous, intelligent agents that travel across the landscape. A central issue is the model used for agent travel decisions. Current techniques range from replicating trips exactly to making local, intersection level decisions based on probability. But little attention has been paid to justifying these models.

In this work we examine a range of probabilistic models. The models differ in the length of the Markov chain used to compute agent decisions. The length of the chain ranges from zero (local decisions only) to infinity (exact trip replication). We test the length of the chain on held out data for validation. We show that the choice of model strongly influences the validity and results of the simulation.

To test these models we present a framework for automatically constructing agent-based models from an input set of GPS tracklogs. The GPS tracklogs are collected by volunteers as they recreate in natural areas. Traditionally, data on where recreators travel is collected in the form of trip diaries, filled out on paper by visitors or by interview. Other demographic and attitudinal data is also collected along with the actual route traveled. Although the additional information is valuable, the data must be collected and entered by hand. Paper diaries also place a significant time burden on visitors, reducing the compliance rate as well as skewing the results (ensuring only visitors with excess time participate).

Using GPS devices to record visitor trips helps alleviate these problems. The framework for processing GPS trips and automatically building a model presented in this work significantly reduces the time required to build a model, lowers the cost and widens the applicability of recreation simulation modeling to new areas. GPS devices automatically record their data, requiring only that visitors turn the unit on and carry it with a marginal view of the sky. GPS use is also becoming more widespread among recreators. As more recreators use GPS to record their trips, data useful to modeling is becoming increasingly available.

The steps in GPS driven model generation are as follows. First, the set of GPS tracklogs is combined to form the underlying travel network along which agents will travel. Each GPS tracklog is then traced along the network in order to determine what choices were made as the recreator traveled across the network. This produces a list of trip itineraries. Model parameters (probability tables) can then be computed from the trips.

The length of the Markov chain used in the probability tables is a parameter to the model. The optimal value is found by testing the likelihood of heldout data for different chain lengths. This step is done automatically. Once the optimal length of the chain is chosen the model is complete and agent-based simulation can proceed.

The entire framework for automatically producing GPS driven agent-based models is implemented in our TopoFusion GPS mapping software.

We present results from two collections of GPS tracklogs from different trail systems. The first is from Tucson Mountain Park and is the result of a volunteer collection effort by the authors. A trails master plan is underway at the park, with input from our model. The second is a collection of tracks from mountain bike rides in the Finger Rock Wash area, collected by the author.

Testing by held-out data on both GPS datasets indicates that current modeling methods are insufficient to model recreator travel decisions. The middle ground (neither exact replication nor local decisions) consistently performs better.

1. INTRODUCTION

Typically, recreational use is modeled using a combination of statistical and GIS techniques (Forer 2002; Ploner 2002). Other approaches (Gimblett et al. 2001; Itami 2002) have employed the use of autonomous agents (Franklin 1996) to simulate human recreators.

To model effectively, data must be collected on current use. Past methods have focused on paper trip diaries and oral interviews. (Lynch 2002).

There have been several approaches to modeling recreator behavior as described by the data collected. RBSim's (Gimblett et al. 2001),(Itami 2002) probabilistic models use exact trip replication, where actors in the simulation are assigned a specific itinerary that is chosen from the database of actual trips collected. Manning et al (Manning 2001),(Wang B. 1999) present a different model using the Extend modeling package, where local decisions are made at each intersection. These two approaches represent the extremes of the "K-history" model presented in this paper. Here, *K* is a model parameter that represents the length of the Markov chain (Rabiner and Juang 1986) considered for a trail intersection decision. The K-history model is probabilistic, relying on counts of decisions made based on the data collected.

Using the approach presented in this paper, two GPS datasets were processed and K-history models were built using TopoFusion software (Morris and Morris 2002-2005). The K-history model allows trips to be generated based on a variable length history of decisions. Depending on the length of the history considered, this model will simulate the range of possible models: from exact trips to local decisions. By computing the likelihood of held out data we present experimental results comparing the choices of K for the two datasets. We conclude that the choice of *K* can be a difficult matter and that it is highly dependent on characteristics of the data.

2. DATA COLLECTION

For this work we focus on two GPS datasets: Tucson Mountain Park and Finger Rock Wash. Tucson Mountain Park is a 20,000 acre natural area that borders the city of Tucson. The park offers a multi-use trail system open to non-motorized recreation. Due to its proximity to an urban area, it is subject to increasing use as well as pressure from development. A park master plan is under way to evaluate the necessity of the complex network of trails and access points available. There is a proliferation of social trails as a result of a lack of management in the area.

The authors have begun a GPS data collection effort in Tucson Mountain Park. Volunteers are asked to carry a GPS device as they recreate in the park. Current GPS users were also solicited to submit GPS tracks of their trips. The collection effort is still underway, but to date 100 trips have been collected.

The second data set, Finger Rock Wash, is a collection of 75 GPS tracks of bicycle rides collected by the author over a period of three years. All tracks originated at the author's residence, following a myriad of routes in order to access a local trail system.

3. AUTOMATED MODEL GENERATION FROM GPS DATA

The process of generating a model given a collection of GPS tracks proceeds in three steps: first a travel network is produced from the combination of all tracks, second, each track is matched to edges of the network to determine the routes taken and third the model parameters are computed from the routes. The following sections briefly outline the algorithms used to achieve these three steps. All of the algorithms presented in this paper are implemented in the TopoFusion GPS Mapping Software.

3.1. Travel Network Production

A travel network is required for the actual simulation (for the agents to travel across) to run. A common, baseline network is also required to relate the trips recorded by GPS tracks to each other. Since most tracks will overlap each other, traveling on the same roads or trails as other tracks, it is necessary to determine what the underlying structure of the travel network is in order to determine what routes were taken by each track.

One solution is to trace out the network, by hand, using topographic maps, aerial photographs as well as the GPS tracks themselves. This is not a trivial task since nodes and edges in the network must align exactly. Tracing complex networks can be very time consuming. We focus on automated methods for these reasons, as well as to decrease the level of expertise required to build a recreational model.

Determining the travel network automatically is a non-trivial task. Due to GPS errors, tracks do not overlap exactly. Instead, they intersect and parallel each other in unpredictable ways, as seen in Figure 1. The key issue is how to determine when portions of tracks correspond to the same trail or road and when they do not.

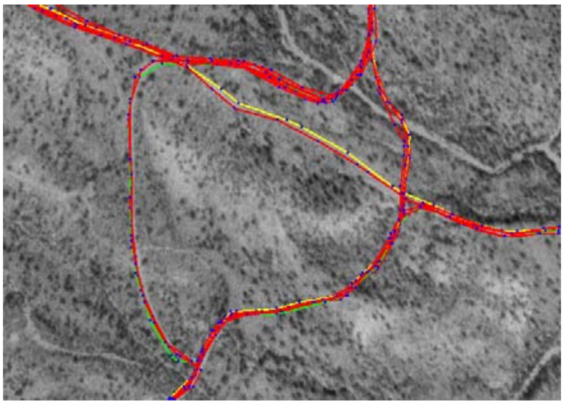

Figure 1. Overlapping GPS tracks from Tucson Mountain Park Dataset

A procedure for producing GPS networks is presented by the authors in (Morris et al. 2004). Given a set of overlapping GPS tracks in an area, it produces the complete travel network where all duplicates have been eliminated. It proceeds by first finding all intersections among the GPS track logs. The result is a network, but one that is far too complex and contains mostly irrelevant nodes and edges. The main task of the procedure is to traverse the network, reducing the areas that are determined to actually be representations of the same trail or road.

There is an adjustable tuning parameter for how close (and similar) areas of the network must be in order to be considered the same road or trail. This value is set in meters, and is dependent on the accuracy of the input data. It is not a hard threshold, but a threshold based on the Hausdorf distance between edges. A typical value used is 40 meters. It can be set higher than needed; the only problem occurs when the travel network contains actual trails or roads that parallel each other closely (e.g. less than 40 meters from each other). Then the procedure may reduce that section to one edge where there should be two. Generally the default value is sufficient, but the procedure may need to be run again, testing different values. This is but a brief overview, for other issues and details see(Morris et al. 2004). The resulting network from the GPS tracks shown in Figure 1 is given in Figure 2.

3.1. Determining Trip Itineraries

Given a travel network and a set of GPS tracks traveling the network, the next step is to determine the route each track traveled. Due to GPS errors, each track will vary in distance to the edges in the network, usually intersecting at various points. Tracks can also turn around at any point, or may be traveling close to two edges in the network. Formally, this problem is known as the Map Matching problem. One map is the underlying network, and the other is the GPS track (which is a map of where the GPS receiver traveled). The problem is to find out how best to match these two maps together in some optimal way.

Efrat et al (Efrat 2003) present a dynamic programming approach using the Frechet distance as a metric. The details of the algorithm are complex and to date is has not been tested on actual GPS data (only fabricated data). Using the Frechet distance is problematic, since the algorithm is optimizing for the minimal Frechet distance across each edge. The problem is that recreators can travel partial edges of the network, which will result in inflated Frechet distances.

TopoFusion uses a simple greedy algorithm to determine the routes traveled. It proceeds as follows. First, the beginning point of a GPS track is compared with all nodes in the network to determine the closest starting point. Then, each outgoing edge from the node is compared with the first portion of the GPS track using the Hausdorf distance (which is invariant to the "back-tracking" problem described above). The outgoing edge with minimal Hausdorff distance is selected as the first edge traveled by the track. The track is then traversed until it reaches the next node (measured by simple distance). The process is then repeated (comparing Hausdorf distances with the next section of the track) until the track reaches its endpoint.

The list of edges in the network traversed is output to a text file. When the procedure is run on all of the input GPS tracks, the result is a single text file listing the itineraries of each track, as shown in Figure 3. The first number preceding the colon is the node number where the trip begins and the trailing -1 indicates an end of line.

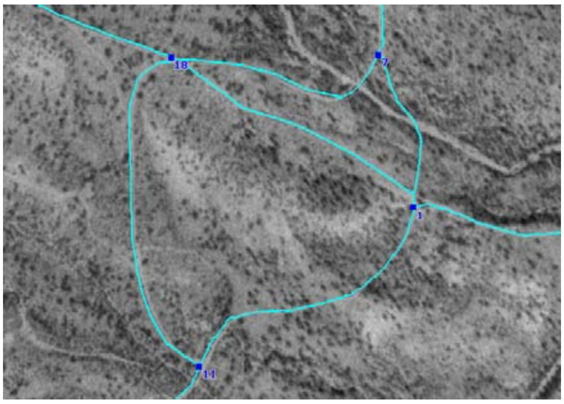

Figure 2. Resulting network from GPS tracks shown in Figure 1. Duplicate representations of trails have been eliminated.

```
1:,1,4,2,7,10,9,10,7,8,11,-1
1:,4,2,7,10,9,10,7,8,12,-1
1:,4,2,7,10,9,10,10,9,9,10,7,8,12,-1
1:,1,4,2,7,10,9,10,7,8,11,12,-1
1:,4,2,8,11,-1
```

Figure 3. Example trip itineraries

3.3. Probabilistic Decision Modeling

Given the outputs of the two proceeding steps, the travel network and trip itineraries, we can now proceed to compute the parameters of our model. We first give details on the K-history model used to evaluate different travel models.

3.3.1. The K-History model

The goal is to model the decision process recreators go through at a trail intersection. A straightforward approach is to compute a probability distribution based on the data collected on what decisions actual recreators made. At a given intersection, if 70 percent of people turned left and 20 percent right, as indicated in Figure 4(a), the model would send agents left 70 percent of the time and right the other 20 percent. This is the simplest model and corresponds to a K-history model with $K = 0$.

But suppose that an agent returns to the intersection after traversing the loop shown in Figure 4(b). The probability distribution remains the same, so he will likely traverse the loop again, with 70 percent probability. This will result in continuous loops with very little chance of exit. Although this pattern may actually be represented by a data set, in actual recreation situations, loops are rarely traversed in succession; this model is not sufficient for modeling this situation.

To solve this problem, we can condition the probability distribution based on the incoming edge that the agent is traveling on to reach the intersection. In this case, for every incoming edge (there are 3 in Figure 4) there exists a different probability distribution. Thus, agents traveling on the loop could have a much higher probability of exiting the loop than continuing on it. This model of conditioning the probability distribution on the incoming edge is the K-history model with $K = 1$. That is, the decision process is based on the previous intersection decision.

Analogous situations can be constructed where the $K = 1$ model also fails to accurately model the data. A natural question arises, what value of K is optimal? This is the intent of this work: to develop a framework for evaluating the choice of K.

If the goal is to only model the data collected as accurately as possible, then the choice of K is obvious: as large as possible. This ensures that paths not seen in the training data are not allowed to occur in the simulation. This is exactly the model used in RBSim, where agents are assigned a specific trip from the training data. The problem with this approach is that the trips collected are only a small sample of the use actually occurring. Since there is no room for variation from the collected data, over-fitting results (K is too large).

3.3.2. Computing Model Parameters

Given a choice of K, we are able to compute the model parameters (i.e. the probability distributions) simply by counting the choices shown in the training data. For a given node, there exists a probability distribution for each possible path (sequences of choices) of length K that ends at that node. Thus, as K increases, the number of possible paths ending at a node increases exponentially. Many of the tables will simply be empty (full of zeros) since they represent impossible paths given the training data. To save space and to decrease the complexity of the implementation, we do not pre-compute the probability distributions for all possible nodes. Instead, we simply store all of the paths that lead to a given node, at the node. Since the number of total trips is generally low (only 100 in the Tucson Mountain Park dataset), the number of paths leading to a node is generally small.

As an agent reaches a node during simulation, the probability distribution is calculated by counting the number of trips (only looking K steps back) that match the agent's path history. A uniform random number generator is used to choose an outgoing edge based on the newly computed distribution.

Another model parameter is the entrance and exit nodes of the network. These attributes are simply assigned automatically to the nodes where trips enter and exit the simulation.

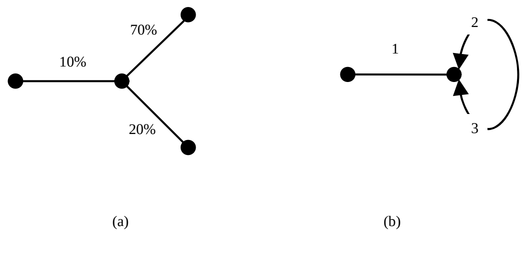

Figure 4. Example trail intersections. (a) Simple $K = 0$ probability distribution. (b) Looping example where K=0 model breaks down

4. COMPARING K-HISTORY MODELS

To compare different values of K in the K-History model, we compute the likelihood of held out data, given the model. This gives us a measure of how well our model (for different K values) predicts new trips. Since our data sets are always a sample of the actual use occurring, the performance on new trips is the most important measure of the validity of a model. Held out data is chosen by selecting a random 10% of the training set. The model is then trained on the remaining 90%.

The held out data is a set of m trips, $H = \{t_1, t_2, \ldots t_m\}$. To compute the likelihood of H, it is sufficient to multiply the likelihood of each trip:

$$P(H) = P(t_1) \cdot P(t_2) \ldots \cdot P(t_m) \qquad (1)$$

Since each trip is an independent event, multiplying the probabilities in (1) is valid. The probability of each trip is computed as:

$$P(t) = \frac{\prod_{i=1}^{n} P(c_i \mid c_{i-1}, \ldots, c_{i-K})}{n} \qquad (2)$$

Where a trip is represented by a sequence of n choices, $c_1, c_2, \ldots c_n$, one choice for each node reached in the network. The probability values $P(c_i \mid c_{i-1}, \ldots, c_{i-K})$ are determined as in the simulation model for agents. For a given node where a choice of outgoing edge, c, occurs, we compute a probability distribution for the outgoing edges based on the 90% training data. The distribution is computed by counting the outgoing edge choice made by each trip in the training set that matches previous choices $c_{i-1}, \ldots, c_{i-K}$. When the sequence of previous choices is less than K in length, the probability is simply conditioned on all of the previous choices.

Equation (2) is normalized by the number of choices in the trip, n, so that each trip receives equal weight in equation (1). There is also an underlying assumption that each choice made is independent of all other choices. Unfortunately there is no simple fix for this assumption.

5. RESULTS

First we present a plot of the likelihood of the Tucson Mountain Park data set, when trained with 100% of the data, for varying values of K. In this case, the held out set H is the training set itself. This measures how well different values of K model the training set. As figure 5 shows, as K increases, the likelihood increases. A K value of zero performs poorly and performance levels out past $K = 10$. This is the expected behavior; as more parameters are added, we are fitting the training set more accurately. This figure also confirms the notion that trail decisions are made based on prior information. $K=0$ corresponds to no prior, which we see performs poorly. We also note that if the model's goal is to accurately simulate current use, not predict future use, a high value of K should be used (assuming a statistically sufficient sample size has been collected).

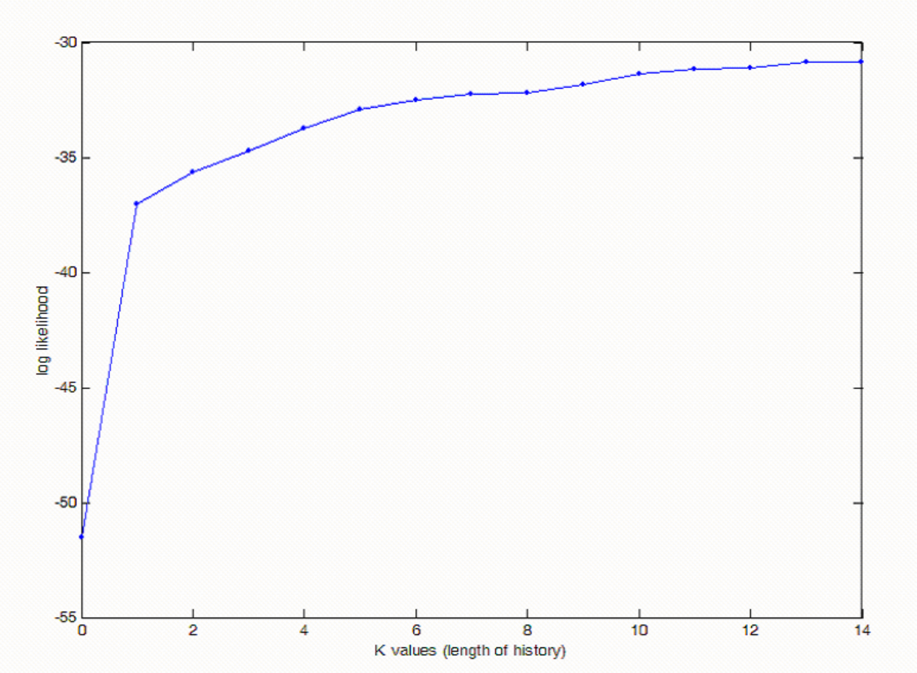

Figure 5. Log Likelihood plot of training data from Tucson Mountain Park. Here higher values of likelihood indicate a better fit of the model. We see that as model parameters are increased (the value of K is increased) the model better fits the data.

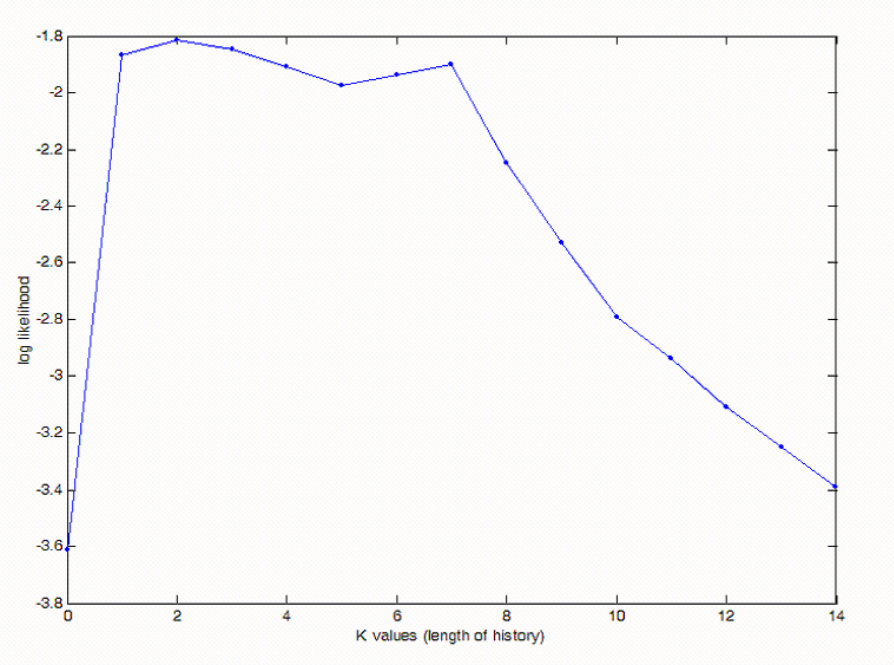

Figure 6. Log Likelihood plot of P(h) for Tucson Mountain Park dataset. K=2 is the optimal value, meaning that considering the last two decisions produces the best performance.

Figure 6 shows the plot of *P(H)* where *H* is a randomly selected 10% of the Tucson Mountain Park dataset. The likelihood of the held out set is maximal at *K* = 2. This means that to best model recreator use at Tucson Mountain Park, trail decisions should be based on the last two decisions, but no more. However, we see that the model performs relatively well for values in the range 1 to 7. After 7, the performance begins to suffer due to over-fitting. As *K* grows larger, the model loses the ability to predict trips that were not part of the training data.

A plot of *P(H)* for the Finger Rock Wash dataset is given in Figure 7. *K* = 5 is maximal here, however, a larger range of *K* values produces near optimal performance. It is interesting to note that *K=0* (no prior information) performs well. This is due to the fact that the dataset is highly directional. That is, the trails are almost always traveled in a certain direction. Thus, it is very likely that regardless of the previous decisions made, the most likely choice will be a good one. However, higher values of *K* still perform slightly better, showing that prior information does improve performance. We again see the characteristic decline in performance as *K* increases.

6. DISCUSSION

The results presented in this paper indicate that the choice of *K* is a delicate matter. The two extremes represented by current approaches have their drawbacks. The trip modeling of RBSim (*K*=∞) suffers from a poor ability to predict trips not present in the training dataset. The Extend (*K=1*) models employed by Manning et al do not sufficiently model the training set and also have poor predictive power. Optimal *K* values on both of the datasets presented in this paper were between the two extremes.

It is important to note that the comparisons with previous techniques are for baseline travel models only. Actual models include agent logic and other rules that influence travel decisions and improve accuracy.

It is clear that the choice of *K* is highly dependent on both the travel network and the structure of the trips. There is no magic value of *K* that should be applied in all cases. We have presented an automated framework for producing a K-history model using an optimal value of *K*. Given input trips in the form of GPS tracks, the procedure automatically generates the travel network, the trip itineraries, the optimal value of *K* as well as the model given that choice of *K*. Little human supervision is required.

There is much work to be done. This is a preliminary study to evaluate the current approaches taken by other researchers. We plan to investigate more complex models, continuing to compare performance with current models using held out data. Some examples of more complex models include variable *K* values, where each node is assigned a *K* value, perhaps differing from other nodes. Another possibility is to cluster the trips beforehand, selecting different values of *K* for each cluster.

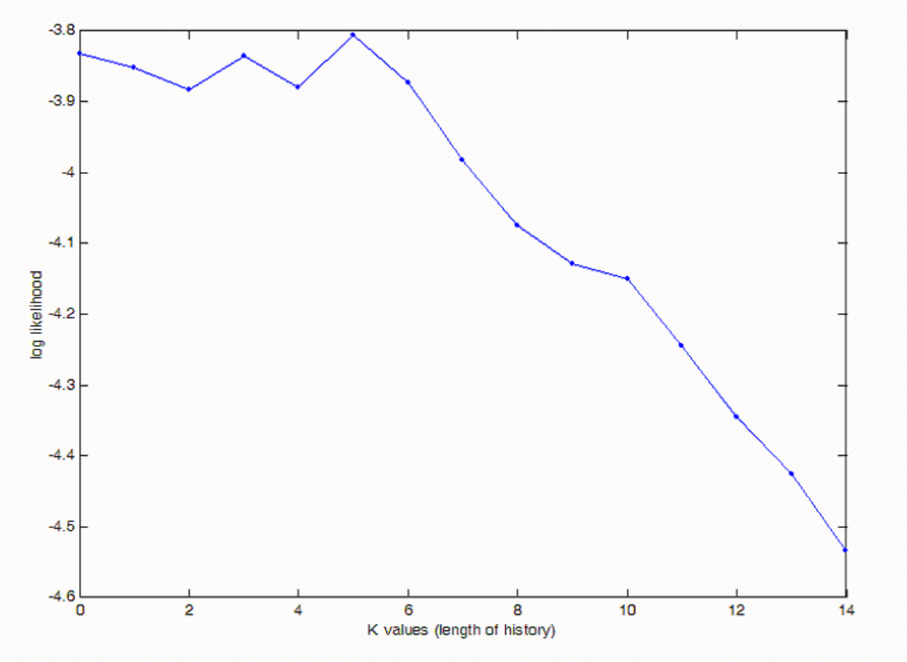

Figure 7. Log likelihood plot of *P(H)* for Finger Rock Wash dataset. *K*=5 is maximal, with a sharp decline in performance due to over-fitting.

7. REFERENCES

Efrat, A., Helmut Alt, Gunter Rote, Carola Wenk (2003). Matching Planar Maps. ACM-SIAM Symposium on Discrete Algorithms, 2003.

Forer, P. (2002). Serial Experiences: Monitoring, Modeling and Visualizing the Free Independent Traveller in New Zealand at Multiple Scales with GIS. Monitoring and Management of Visitor Flows in Recreational and Protected Areas, Vienna, Austria.

Franklin, S., Art Graesser (1996). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. Third International Workshop on Agent Theories, Springer Verlag.

Gimblett, H. R., M. T. Richards and R. M. Itami (2001). "RBSim: Geographic Simulation of Wilderness Recreation Behavior." Journal of Forestry **99**(4): 36-42.

Itami, R., Rob Raulings, Glen MacLaren, et al (2002). RBSim 2: Simulating the Complex Interactions between Human Movement and the Outdoor Recreation Environment. Monitoring and Management of Visitor Flows in Recreational and Protected Areas, Vienna, Austria.

Lynch, J. (2002). A Spatial Model of Overnight Visitor Behavior in a Wilderness Area in Eastern Sierra Nevada. Monitoring and Management of Visitor Flows in Recreational and Protected Areas, Vienna, Austria.

Manning, R. (2001). "Visitor Experience and Resource Protection: A Framework for Managing the Carrying Capacity of National Parks." Journal of Park and Recreation Administration **19**: 93-108.

Morris, S. and A. Morris (2002-2005). TopoFusion**:** GPS data management and mapping software. http://www.topofusion.com

Morris, S., A. Morris and K. Barnard (2004). Digital Trail Libraries. Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries.

Ploner, A., Christine Brandenburg (2002). Modelling the Dependency between Visitor Numbers and Meteorological Variables via Regression Trees. Monitoring and Management of Visitor Flows in Recreational and Protected Areas, Vienna, Austria.

Rabiner, L. R. and B. H. Juang (1986). "An introduction to hidden Markov models." IEEE ASSP Magazine: 4-15.

Wang B., M. R. E. (1999). "Computer Simulation Modeling for Recreation Management: A Study on Carriage Road se in Acadia National Park." Environmental Management **23**: 193-203.