Mobile User Tracking Using A Hybrid Neural Network

KAUSIK MAJUMDAR
Electronics and Telecommunication Engineering Department, Jadavpur University, Calcutta-700032, India

NABANITA DAS
Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Calcutta-700108, India

Abstract. In this paper, a novel technique for location prediction of mobile users has been proposed, and a paging technique based on this predicted location is developed. As a mobile user always travels with a destination in mind, the movements of users, are, in general, preplanned, and are highly dependent on the individual characteristics. Hence, neural networks with its learning and generalization ability may act as a suitable tool to predict the location of a terminal provided it is trained appropriately by the personal mobility profile of individual user. For prediction, the performance of a multi-layer perceptron (MLP) network has been studied first. Next, to recognize the inherent clusters in the input data, and to process it accordingly, a hybrid network composed of a self-organizing feature map (SOFM) network followed by a number of MLP networks has been employed. Simulation studies show that the latter performs better for location management. This approach is free from all unrealistic assumptions about the movement of the users. It is applicable to any arbitrary cell architecture. It attempts to reduce the total location management cost and paging delay, in general.

Keywords: mobile users, location management, paging, location prediction, MLP network, SOFM network

1. Introduction

To cope with the increasing number of users and their demands for personal communication service (PCS), it is essential to build up an efficient and probably '*intelligent*' location management scheme. Since every mobile terminal is free to move within the whole service area, on call arrival, the network searches the terminal for call delivery by the process known as *terminal paging*. To facilitate paging, every mobile terminal informs the network periodically about its current location. This is known as *location update*. Though from operational point of view, paging is more fundamental than update, majority of the research on location management has actually focused on update schemes, assuming some simple paging strategies [1–7,10,18,24]. The naive approach for paging is to search the entire network, or in a part. Hence with the increase in number of mobile users, paging contributes to an appreciable amount of additional traffic. Another important issue is the paging delay, that is determined by the paging sequence. Researchers attempted to propose efficient paging techniques to reduce the cost and/or the delay [13,17,21,22]. Generally, the mobile's last updated position and its surroundings are considered to be the most probable current location, the probability decreasing uniformly along all possible directions with increasing distance [1,2,4,10]. However, in [6], authors considered the directional bias in user movement as well. In [20], user movement profile has been considered to design paging algorithm. There, it has been assumed that probabilistic information about movement profile is available either with user, or in the network.

However, most of the earlier works on location management are evaluated on the basis of many simplifying assumptions. They are mostly based on regular cell structures like linear, rectangular or hexagonal [1,2,4,18]. The cell residence time distribution is also assumed to be a geometric one that is independent and identically distributed for all the cells of the network. Moreover, the user mobility is often modeled as a random walk model [1,2,4,18]. Markov models are often constructed with states and arbitrary transition probabilities, assuming a regular geometric cellular architecture.

In practice, the users' movements are not at all random. A deeper insight would reveal that almost every individual has a mobility pattern which it follows, in general. This mobility pattern is highly dependent on the cell characteristics as well as the characteristics of the individual. This idea motivated researchers to study the recent movement patterns of the subscribers and with the knowledge thus gathered, to predict the location, instead of paging the whole location area for call delivery [5,16]. Tabbane [23], proposed to determine the terminal's location based on its quasi deterministic mobility behavior represented as a set of movement patterns stored in a user profile. In [11,12] the speed and trajectory of users are predicted but with the limitation that they consider rectilinear movement patterns only.

However, a mobile user, in general, travels towards a specific destination. Naturally, its future location is likely to be correlated with its personal mobility profile, the cells visited recently, and also the time spent in each. Over and above all, there may be different classes of individuals such that the movement of the users in the same class more or less follow similar patterns whereas those from different classes exhibit quite different mobility behavior in terms of cells visited along with the respective cell residence times. Hence, it is clear that knowledge representation and learning of user mobility profile

are two important key factors which may endow the paging mechanism a predictive power to reduce average paging cost, in general.

Considering these special characteristics of human mobility, an artificial neural network based approach is proposed in [8] for location prediction to reduce the cost of location management. There, for each individual host, one neural network is to be trained and maintained in the system, which is quite time consuming and almost infeasible when the number of users is large. Moreover, the mobile host is to determine the probable location areas where it may reside, and to keep track if it deviates, when it requires to switch to the conventional location management schemes. Obviously, this technique, though cost-efficient, will cause rapid drainage of battery power in the host. Moreover, the paging scheme proposed there, pages the predicted location areas in order of their probabilities, which will be efficient only for very punctual persons, and in a very smooth-running city.

In this paper, we propose a single neural network model for learning all the movement profiles for the whole set of users. The training part can be done off-line based on the movement history of a large number of subscribers. Following a standard update scheme, say distance-based update, the mobile sends its recent movement history along with the update message. When a call arrives for a particular user, based on the recent inputs obtained during the last update of the terminal, the present location is predicted by the neural network on line. Next paging is done in several steps in and around the predicted cell unless the terminal is found. Firstly, the performance of an MLP (multi layer perceptron) network [9] is investigated. It shows a success rate of 65–75%. Next, considering the complex characteristics of the training data from users with widely varied movement patterns, a hybrid neural network composed of an SOFM (self-organizing feature map) network [14] followed by a number of MLP networks [19] has been employed. The SOFM network attempts to partition the input data into clusters on the basis of their closeness in the p-dimensional hyperspace defined by the p features of an input pattern, and activates the appropriate MLP for a given input vector. The experimental results on hexagonal cell architecture show that the hybrid network performs better than the MLP network, indicating that the input patterns in the experimental data really consist of clusters.

The rest of the paper is organized as follows. The proposed model is introduced in Section 2. The artificial neural networks employed here, are described in Section 3. Simulations and results are presented in Section 4. Finally, Section 5 includes the conclusion.

2. Proposed scheme

Here, it is assumed that the network operates with a predetermined location update scheme. On call arrival, the current location of the terminal is predicted based on the information received during last update or call termination, and paging is done accordingly.

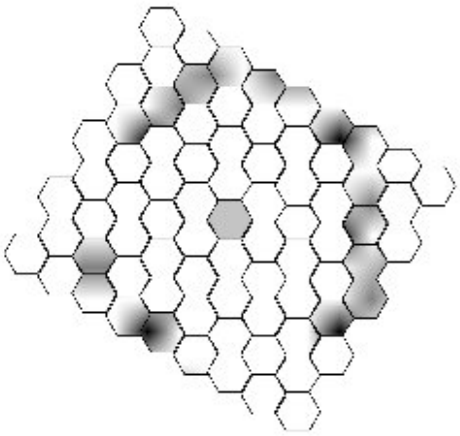

Figure 1. Boundary of distance-based update for distance threshold 3.

2.1. Location update

Among the three dynamic location update strategies, namely time based, movement based and distance based algorithms [4], we choose the third one, since it is the best in performance among the three, and implement it in the same way as has been mentioned in [24]. Let d be the threshold distance for update. Whenever a mobile terminal updates its location, the network transmits to it the list L of all the cell-ids on the ring of radius d around the present cell as shown in figure 1. The terminal stores this list L in its local memory. Whenever the terminal crosses a cell boundary, it checks whether the *cell-id* belongs to L, and it updates only when a match is found.

The update algorithm can be implemented on any kind of cellular architecture even with irregular shapes as is the case in real situations. Also the scheme involves very small overhead.

2.2. Location prediction

In distance based location update scheme, as if the terminal is logically free within a circle of radius d. Now, our basic assumption is that in general, every terminal has its own mobility pattern depending on the characteristics of itself and the cell it is traversing. By cell characteristics we mean the location of the particular cell, how roads and highways are placed within it and also location of offices, housing complexes, shopping centers, etc. within the cell. On the other hand individual characteristics depend on the particular individual.

Keeping these parameters in view, a prediction network is to be designed which learns the rules that determine the mobility patterns from past events. Then it uses this knowledge to predict the location of a user when a call arrives. A multi layer perceptron (MLP) network has exactly this capability of learning rules from past events. Therefore, an MLP network is first used here, for prediction. Later, a hybrid neural network consisting of a Self Organizing Feature Map (SOFM) network followed by multiple MLP networks is employed.

The input and output features are carefully chosen to incorporate the best possible information that can be supplied to the prediction network. To keep the movement history, the

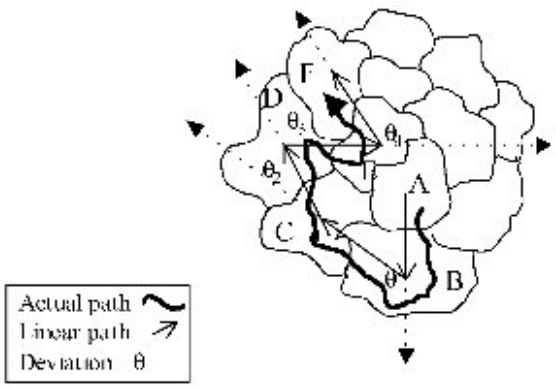

Figure 2. Intercellular path approximation with corresponding angular deviations.

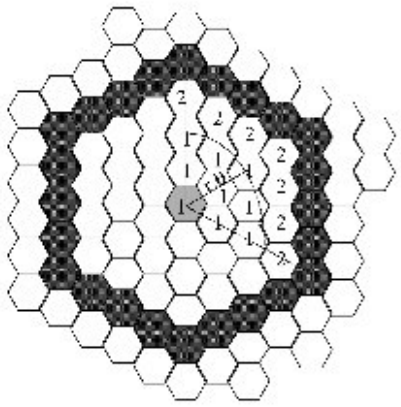

Figure 3. Location prediction and angular paging scheme.

terminal may store last $(m+1)$ *cell-id*'s it visited. To generalize this information, from the *cell-id*'s, the *intercellular path* of the user is determined which is obtained by joining the centers of the corresponding cells. Next, from the *intercellular path* the angular deviations θ_i, $1 \leq i \leq m$ are computed, as shown in figure 2.

This information gives the prediction network an idea about the direction of motion of the terminal in respect of cells. The cell-residence times corresponding to the $(m+1)$ cells visited recently are also kept stored in the terminal. This can be done easily by having time stamps when the terminal enters a new cell. Another input is the time elapsed after the last update, or call termination. This information combined with the previous ones can give the network an idea about the current location expressed as (r, θ), the origin being taken as the center of the cell of last update or call termination, and the line of reference as the direction of last hop.

Based on these features, the prediction network attempts to predict the current location.

2.3. Terminal paging

Theoretically, the current cell of the user should be predicted exactly by the network. However, due to possible fluctuations in movement patterns of the user, there is a finite probability that the user will not be found in that particular cell. Hence paging is essential. An angular range $\pm\phi$ is defined about the predicted deviation line as shown in figure 3. Let d be the threshold distance for update. Now, given the predicted distance r, for $0 \leq r \leq \lceil d/2 \rceil$ all the cells within the angular region $\pm\phi$, at distance x for $0 \leq x \leq \lceil d/2 \rceil$ are paged first. This paging region is termed here as the *lower shell*. If the terminal is not found, in the next paging cycle, all the cells in the *upper shell*, at distance x, $\lceil d/2 \rceil < x \leq d$ within the same angular region $\pm\phi$ are paged. In case of failure, the next angular region selected for paging is ϕ to 2ϕ and $-\phi$ to -2ϕ. The third and fourth paging cycles occur in these angular regions in the same order depending on the predicted value of r. This angular spreading and corresponding paging cycles continue until the terminal is found by the network. If $r > \lceil d/2 \rceil$, the *upper shell* in the angular region is paged first, followed by paging in the *lower shell*. An example is shown in figure 3, where depending on the value of r, the lower shell is paged in the first cycle with the cells labeled as 1, followed by the upper shell with the cells labeled as 2.

3. Neural network models

In this section, a brief overview of the neural network models used here for location prediction is presented. First, the MLP network is discussed briefly, then the hybrid network is introduced.

3.1. MLP network

In general, the multi layered perceptron (MLP) network is made up of sets of neurones arranged in several layers. The first layer is the input layer through which inputs are fed to the network. The last one is the output layer, that produces the predicted outputs. The remaining layers are called hidden layers. Figure 4 shows a typical MLP network with a single hidden layer. The outputs from each node of one layer are transmitted to all the nodes in the next layer through links with weights. Every node, except the input nodes, computes a weighted sum of its inputs and apply a sigmoid function to find its output, which then is transmitted to the next layer [9]. Each node is activated in accordance with the input to the node, the activation function of the node, and the bias of the node.

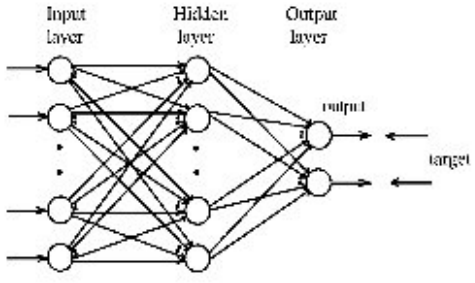

Figure 4. A typical MLP network with a single hidden layer.

Mathematically, the output y of any node can be represented as:

$$y = F(\mathbf{W}^T X + b),$$

where, $\mathbf{X}$ is the input vector, $\mathbf{W}$ the weight vector, F the activation function and b the bias.

The objective of the MLP learning is to set link weights such that for any given input pattern, the error between the MLP output and the desired output (target) is the minimum. In fact there exists several learning techniques for updating the network weights, of which the back propagation technique is the most popular one. Moreover, it has been established that a single hidden layer is sufficient for an MLP network to learn all non-linearities present in input data provided there is sufficient number of hidden nodes. Hence in this study, only single hidden layer architecture is considered. Let us assume that the net consists of p, n_h and n_o number of neurones in the input layer, hidden layer and the output layer respectively.

The weight matrices $\mathbf{W}^{ih}_{p\times n_h}$, for links from input layer i to hidden layer h, and $\mathbf{W}^{ho}_{n_h\times n_o}$, from hidden layer h to output layer o, respectively, are updated by the following rule:

$$\mathrm{W}^{ih} = \mathbf{W}^{ih} + \Delta\mathbf{W}^{ih}, \quad \text{where } \Delta\mathbf{W}^{ih}_j = \alpha\mathbf{ie} + \beta\Delta\mathbf{W}^{ih}_{j-1}.$$

$$\mathrm{W}^{ho} = \mathbf{W}^{ho} + \Delta\mathbf{W}^{ho}, \quad \text{where } \Delta\mathbf{W}^{\mathrm{ho}}_j = \alpha\mathbf{hd} + \beta\Delta\mathbf{W}^{ho}_{j-1}.$$

Here, α is the learning rate, and β the momentum factor for weight changes,

$\mathbf{i} : [i_1 i_2 \ldots i_p]^T$, the input vector
$\mathbf{o} : [o_1 o_2 \ldots o_{n_o}]^T$, the final output vector
$\mathbf{h} : [h_1 h_2 \ldots h_{n_h}]^T$, the output vector from the hidden layer
$\mathbf{t} : [t_1 t_2 \ldots t_{n_o}]^T$, the target output, where t_k is the target output for k-th neurone,
$\mathbf{d} : [d_1, \ldots, d_{n_o}]$, where $d_k = o_k(1 - o_k)(o_k - t_k)$, for $k = 1, 2, \ldots, n_o$.

The error vector $\mathbf{e} : [e_1, \ldots, e_{n_h}]$, where:

$$e_k = h_k(1 - h_k)\sum_{l=1}^{n_o} W^{h_o}_{lk} d_l, \quad k = 1, 2, \ldots, n_h.$$

The activation function is assumed to be $F(x) = \frac{1}{1+e^{-x}}$.

For training, the weights on the links $\mathbf{W}^{ih}$, and $\mathbf{W}^{ho}$ are initialized randomly. Next, applying an input pattern, the error is computed, and the weights are adjusted accordingly by the above rules. The procedure is repeated for large number of input samples, until the weights are stabilized. After the training is over, the network can be used for prediction.

3.2. SOFM network

In real situation, the input data set corresponding to the mobility profiles of all the subscribers, is in general highly complex and of widely varying characteristics. So it may be quite difficult for a single MLP network to learn the whole data set. To refine the action of MLP network, we employ a hybrid neural network SOFM-MLP proposed in [19]. In this network, the input patterns are first clustered into several subgroups on the basis of closeness in the p-dimensional input space (each input pattern is a p-element vector), by a Self Organizing Feature Map (SOFM) network. Then for each cluster, a unique MLP network is trained. For prediction, given an input vector, first the appropriate MLP is chosen by the SOFM network and then that MLP is used for the actual prediction. Experimental results show this hybrid network performs better than the single MLP network which proves that the experimental data really contains clusters of input patterns.

Kohonen's Self Organizing Feature Map networks [14] are designed primarily for *unsupervised learning*. Here, the training data set contains only input variables but no outputs for supervision, or comparison. The network simply attempts to learn the structure, and hence to recognize clusters in the input data, and to respond accordingly. Once the network has been trained to identify the clusters in data, it can be used as a visualization tool to examine the data. In this case, since we have no idea about the topological distribution of the mobility profiles in the input space, it would be helpful to study the performance of an SOFM network in order to recognize the clusters in the training set, if it exists, and to train several MLP networks separately for different clusters.

The network essentially induces an algorithmic transformation $A^D_{SOFM} : R^p \to \mathbf{V}(R^q)$ that helps to visualize the topology, i.e., the local and neighborhood properties of input vectors $\mathbf{x} \in R^p$, and attempts to map input patterns to output nodes such that *nearness* property is preserved. The architecture of the SOFM network used here is shown in figure 5. An input vector $\mathbf{x} \in R^p$ is distributed by a fan out (input) layer with p nodes to each of the $(l \times n)$ output nodes in the competitive (output) layer. Each node (i, j) of this layer is associated with a $(1 \times p)$ weight vector $\mathbf{w}_{ij}$, the weights on its links from the input nodes. Now, for any input vector, the SOFM net activates just a single output node (the winning node), say, with a weight vector $\mathbf{w}_{ab}$. This winning node can be mapped to the cell (a, b) in a *2-D* grid, as shown in figure 5. Hence, all the input vectors in R^p are essentially mapped on a *2-D* grid O_2.

The SOFM network is trained by the following iterative algorithm:

Step 1: The weight vectors $\mathbf{w}_{ij}$ from p inputs to $(l \times n)$ outputs are initialized with small random values $(0.4 - 0.6)$.
Step 2: A new input vector $\mathbf{x}$ is presented.
Step 3: Compute the Euclidean distance d_{ij} between the input and each output node (i, j) using:

$$d_{ij} = \sum_{k=0}^{p-1}(x_k(t) - w_{ijk}(t))^2,$$

where $x_k(t)$ is the input to node k in the input layer, and $w_{ijk}(t)$ is the weight from the input node k to output node (i, j), at t-th iteration.
Step 4: Select the output node (a, b) with the minimum distance d_{ab} as the *winner node*.
Step 5: The weights of the node (a, b) and its neighborhood nodes $N_{ab}(t)$ with the center at (a, b) are updated as:

$$w_{ijk}(t + 1) = w_{ijk}(t) + A_{abij}(t + 1)(x_k(t) - w_{ijk}(t)),$$

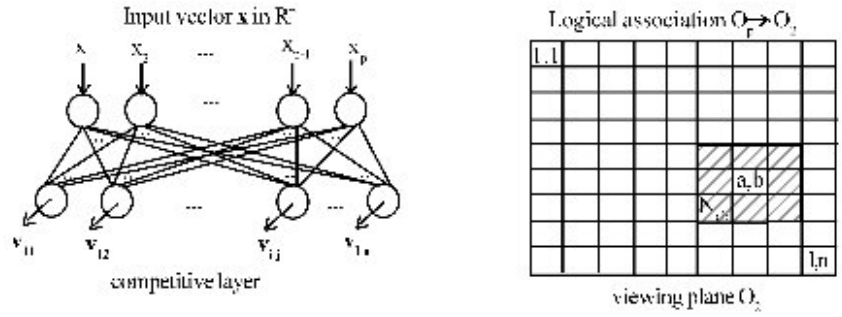

Figure 5. SOFM net and its 2-D display with winning node(a,b) and its 3 × 3 neighborhood.

where $0 \le k \le p-1$, and $(i, j) \in N_{ab}(t)$.

The function $A_{abij}(t)$ represents the strength of interaction between cells (a, b) and (i, j) in its neighborhood. Usually,

$$A_{abij}(t) = \alpha(t)e^{-d(abij)^2/\sigma(t)^2},$$

both $\alpha(t)$, and $\sigma(t)$ monotonically decrease with t, and $d(abij)$ is the distance between cells (a, b) and (i, j).

Step 6: Go to *Step 2*.

This iterative training procedure simply runs through a number of epochs, on each epoch with a new input pattern, it selects the winning node, adjusts the weights of the winning node and its neighborhood to make it closer to the input. It uses a time-decaying learning rate $\alpha(t)$, and also the neighborhood shrinks with time, until the neighborhood consists solely of the winning node itself. Once the training is complete, the node with the weight vector closest to a given input pattern will win for that pattern, and also for any other pattern that it is closest to. Hence the input patterns which cause the same node to win are then deemed to be in the same group.

3.3. SOFM-MLP hybrid network

Once the SOFM network successfully partitions the input data set into several clusters, unique MLP's can be trained for processing each group. Now, any input pattern is presented firstly at the fan out layer of the SOFM network, the corresponding winning node then activates a single MLP network to generate the predicted output. The architecture of this hybrid network is shown in figure 6. The scaling layer with p nodes maps each feature of the input within the range [0, 1] which is then presented at the input of the SOFM network. Let k be the number of nodes in the output layer of SOFM network which feeds data to the input layer of k MLP networks, as shown in figure 6. Without loss of generality, we assume that each of the k MLP networks has only one hidden layer, although it could be more than one and different for different MLP nets. Each node in the input layer of any MLP computes the product of the two inputs it receives, one the scaled feature, and the other from a unique output node of SOFM net. Since, given any input pattern, there will be a single winning node of the SOFM net,

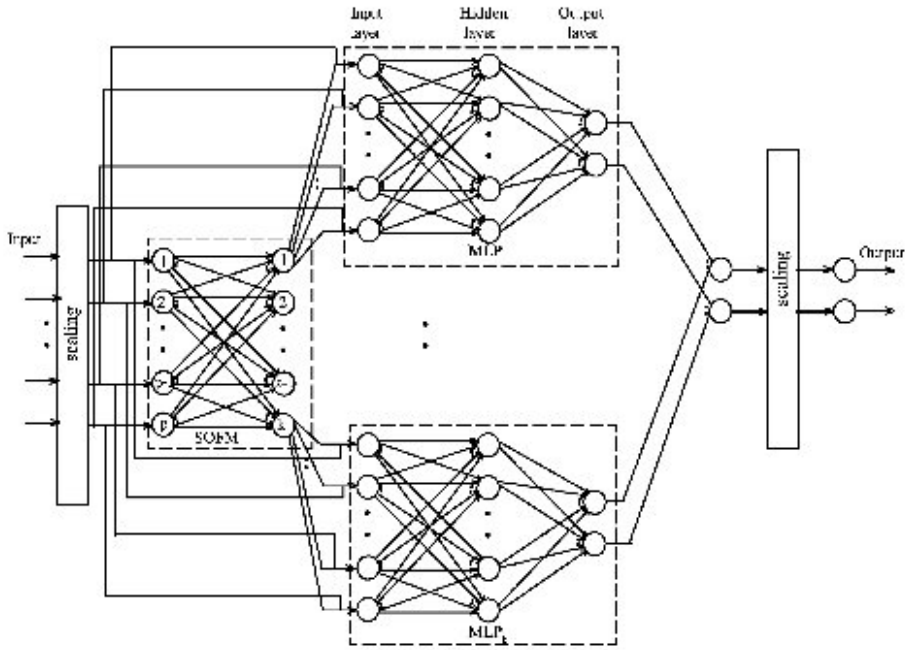

Figure 6. SOFM-MLP hybrid network.

only one of the MLP nets, connected to the winner node, will get the normalized input un-attenuated, while all inputs to the remaining $(k-1)$ MLP will be zero. Outputs from the MLP nets are finally aggregated to the scale-back layer to map them within proper range. It is to be noted that this architecture ensures that aggregated output that will be fed to the scale-back layer is nothing but the output of the MLP corresponding to the winning node of the SOFM net.

4. Experimental study

In this section we discuss the simulation of the model proposed above. It is to be noted that in a real situation, the prediction network is to be trained with the past movement history of different users. However in absence of such real data in hand, here a statistical mobility model is developed for generating the user profiles.

4.1. User mobility model

User mobility pattern incorporates two major views: the directional mobility and the average velocity. The average velocity of a user is reflected by the cell residence times. Here, we assume a very general distribution for the cell residence time, the gamma distribution given by

$$\gamma(t) = K_1 t^{a-1} e^{-t/\lambda},$$

where $t > 0, a > 0$, and, $K_1 = \frac{1}{\Gamma(a)\lambda^a}$. The mean and variance of this distribution are $\mu = a\lambda$ and $\sigma^2 = a\lambda^2$.

Since, in reality, depending on the exact time during a day, the cell residence time of a particular user in a cell may vary widely, the whole day is divided into six time zones, and the values of μ and σ are varied randomly within a range, 5 min $\leq \mu \leq$ 4 h, and 10 min $\leq \sigma \leq$ 60 min within each time zone.

For directional mobility, most of the related works assumed random walk model where after a user leaves a cell, it moves to any neighboring cell with equal probability. Although in practical situation it seems to be hardly true. In this paper, we assumed that the probability of deviation of a user from the current direction of motion decreases with increase in the amount of deviation. This idea follows from the fact that in general, a user moves with some inertia, i.e., it always tries to maintain its state of motion or rest. It is also assumed that the probability of deviation increases with the increase in the cell residence time. It is evident that when the cell residence time is high, the user generally tries to return, i.e., the deviation would be 180°. Based on these two assumptions, the probability density function of deviation from current direction of motion is presented as:

$$f(\theta) = K t^{n_1} e^{-n_2|\theta|}, \quad \theta \in [-\pi, \pi], \quad n_1, n_2 \geq 0,$$

where t is the cell residence time of the user in the cell, θ is the corresponding angular deviation from current direction, and K is a constant. Here, it has been assumed that $n_1 = n_2 = K = 1$.

Based on these two models, we generate 1500 movement patterns, out of which 1200 input and target patterns are selected randomly for training the system, and the rest has been used for testing.

4.2. Simulation

Though the scheme presented here does not assume any regular cellular structure, for simulation, a hexagonal cellular architecture is considered. It is also assumed that the terminal keeps in local memory the most recent $(m+1)$ cells visited from which the network derives m angles of deviation. The last m cell residence times are also kept in memory. In our simulation, results are studied for $2 \leq m \leq 4$. Thus the input patterns of the prediction network contain $(2m+1)$ features including m angles of deviation, m cell residence times and Δt, the time elapsed from the last update/call termination. Here, Δt is assumed to be exponentially distributed, with a mean value of 30 min. Based on these inputs, the network predicts the current cell of the user in terms of distance r and deviation θ with respect to the last reported cell. Therefore, two output nodes are required in the prediction network. For the single MLP network, by trial and error, the number of hidden nodes is determined as 7. It produced best results for the given test data set.

For the hybrid network, number of clusters (i.e. the number of nodes in the competitive sub-layer of the SOFM layer) is a very important parameter. It determines the number of clusters into which the network divides the whole set of movement patterns according to the proximity of the input data points in the $(2m+1)$-dimensional space. Training of the SOFM network with the experimental input data shows that 5 output nodes would be sufficient for clustering the inputs. Now according to the clustering, the input data is partitioned into five disjoint clusters. With each partition, a unique MLP network is trained, and attached at the appropriate output node of the SOFM network. The number of hidden nodes in each MLP network is taken as 5. In fact, the number of nodes in different layers are decided by trial and error method to produce best results.

The whole set of input patterns, say X (comprising of 1500 patterns), is partitioned into two mutually exclusive but exhaustive subgroups, one is used to train the system, and the other to test the system. Typically, 80% of the whole generated data is selected randomly to train the system and the rest 20% is used for testing. The learning rate for the MLP networks was taken to be 0.1. Training of the network is terminated when the difference between errors in successive iterations is below the limit (10^{-5}). For SOFM networks, the training is over after 6×10^5 iterations.

Since MLP output results may land into local minima, so while training, every MLP is run 5 times with different sets of weight initializations for statistical consistency. The whole process of testing is repeated five times with different partitioning of the input data to make the testing more rigorous.

4.3. Results

In this section, the results are presented which are obtained from the simulation described in the preceding section.

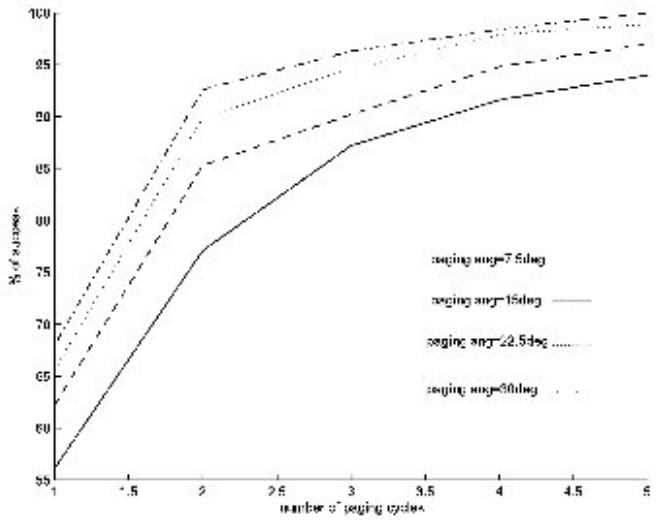

Figure 7. Success rate with number of paging cycles for MLP net with $m = 2$.

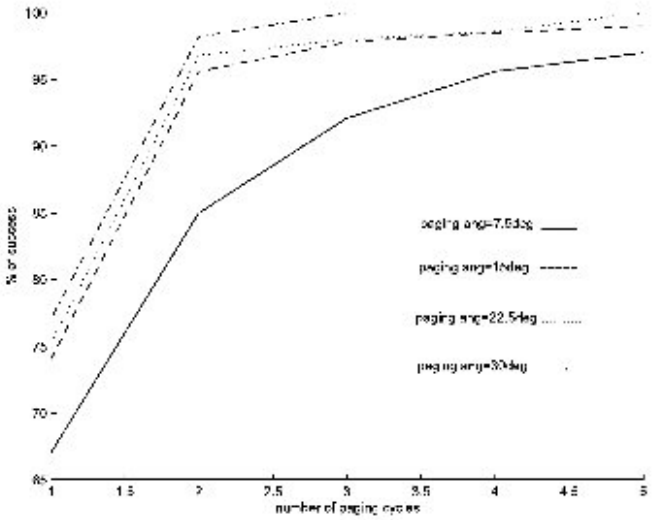

Figure 9. Success rate with number of paging cycles for MLP net with $m = 4$.

Figures 7–9 show the average percentage of success in locating a terminal against the number of paging cycles for direct MLP network, with angle of paging as a parameter for various values of m, $2 \leq m \leq 4$. It is evident from the figures that the success rate improves with the increase in the value of m. However, since the improvement is insignificant as the value of m is increased from 3 to 4, for later studies, m is kept fixed at value 3. From the graph it is found that about 75% of the users can be located just in the first attempt with $m = 3$. This success rate also increases, as expected, with the increase in the angular range (ϕ). Four different values of ϕ have been studied, 7.5°, 15°, 22.5° and 30° respectively.

Figure 10 shows the same variation for $m = 3$, when hybrid SOFM-MLP network is used. One should note the increase in percentage of success from the previous case at any given angular range, except at the first attempt with $\phi = 7.5°$. It may be due to the cause that without SOFM layer, the whole data set was utilized to train a single MLP network.

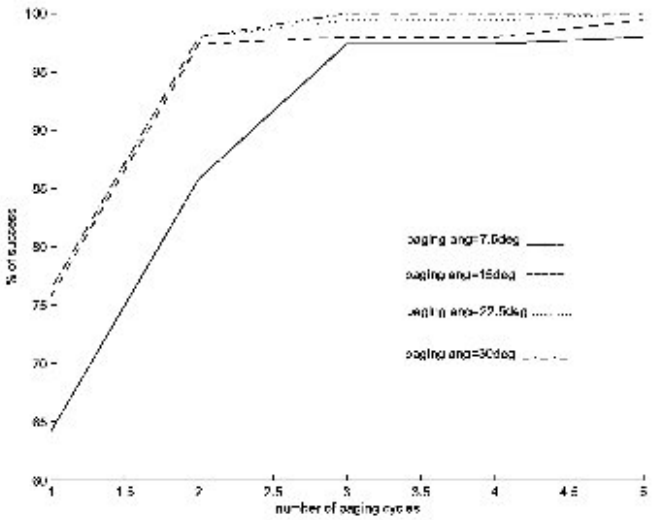

Figure 10. Success rate with number of paging cycles for SOFM-MLP net with $m = 3$.

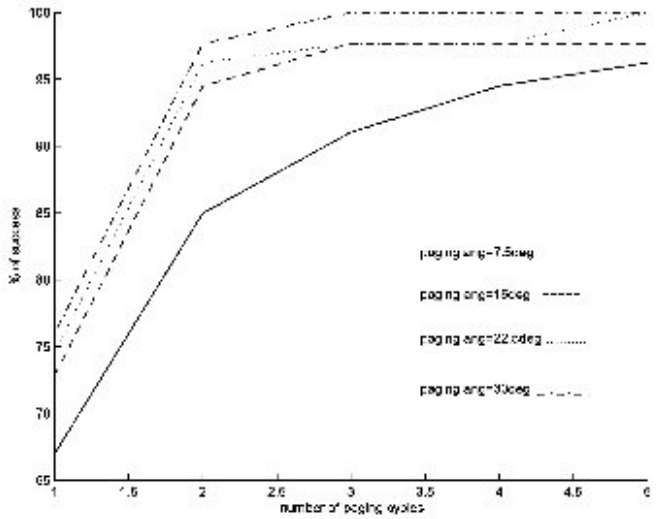

Figure 8. Success rate with number of paging cycles for MLP net with $m = 3$.

But with SOFM, the same data set has been partitioned into five groups, and a particular group of data may be too small to train the corresponding MLP network. Also, at low angular range, even small discrepancy in prediction may fail to locate the terminal in first attempt. However, with larger angular ranges, SOFM-MLP performs better in all cases, and the trend shows that training with enough input data may improve the performance further. Next, the performance of the proposed paging scheme is compared with that of the conventional paging strategy, assuming that all the cells within the distance threshold are to be paged simultaneously. Figures 11 and 12 show the average paging cost in terms of the average number of cells to be paged per call arrival, in case of MLP and SOFM-MLP networks, respectively. Simulation results are shown varying the threshold distance d, $3 \leq d \leq 6$. Results show that with increase in d, the paging cost increases rapidly in case of conventional paging, which in practice, limits the value of d. But in the case of paging with prediction, the

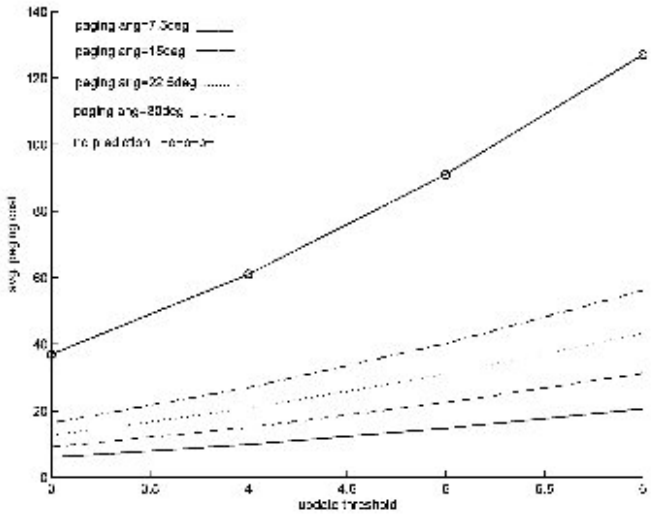

Figure 11. Comparison of average paging cost for MLP and conventional paging.

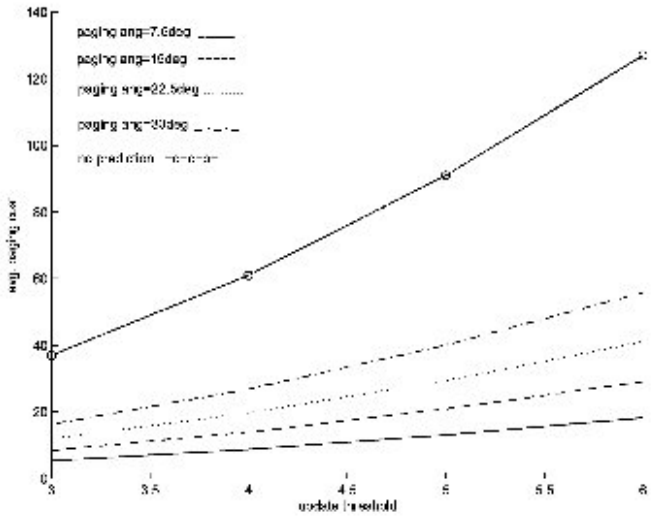

Figure 12. Comparison of average paging cost for SOFM-MLP and conventional paging.

increase in paging cost is insignificant, and hence, it reveals another important advantage of proposed technique. A higher value of d can be allowed in this scheme without deteriorating the paging cost. It would help to reduce the update cost as well.

Finally, figures 13 and 14 compare the average paging delay (in terms of number of paging cycles) and average paging cost (in terms of number of cells paged), for MLP and SOFM-MLP networks, with threshold distance 3. Obviously SOFM-MLP network outperforms the MLP network in both the cases. As expected, the delay reduces largely as the angular range is made larger, on the other hand average cost increases steadily with paging angle. So we are to compromise between these two parameters.

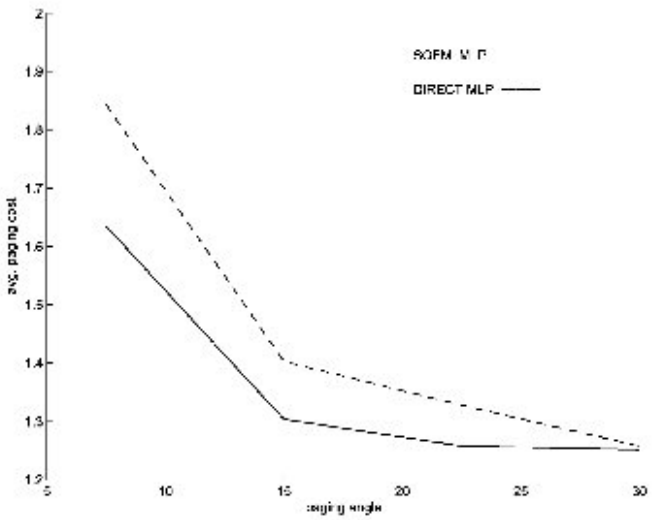

Figure 13. Comparison of paging delay in MLP and SOFM-MLP networks.

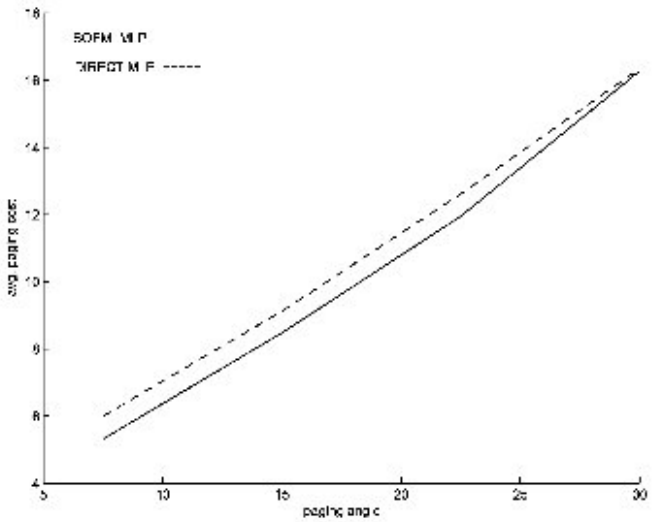

Figure 14. Comparison of paging cost in MLP and SOFM-MLP networks.

4.4. Implementation considerations

The implementation of the above scheme is quite simple. When a terminal updates its location, the network notes its location and gives it the *cell-id*'s of the cells lying on the threshold boundary centering the terminal which are stored in the local memory of the terminal. Whenever a cell boundary crossing occurs, the terminal checks and updates only when a match is found.

The terminal keeps the last $(m + 1)$ *cell-id*'s visited with corresponding time stamps in its local memory. When it informs the network about its current location, it also sends this information. Since the cell locations are stationary to the network, the network can calculate last three angular deviations of the terminal along with the cell residence times. With these information, the input vector is prepared.

A single prediction network is kept at the backbone network of the mobile system. When a call arrives for a user, the input vector is prepared for the particular user and is passed through

the hybrid network as described in Section 3.3. According to the output of the system, the network starts paging for the terminal.

The success of the scheme heavily depends on the training of the hybrid network. So the mobility patterns for training are to be selected very carefully and should cover the whole set of users. Since, training can be done off-line, enough time should be devoted for that. Once the network is trained, it can predict the location precisely, and hence can reduce the paging overhead appreciably.

5. Conclusion

In this paper, a new paging scheme has been developed using a single hybrid neural network to learn the movement patterns of mobile users. The principal assumption here, is that user movements do possess some mobility patterns, in general. A neural network is capable of capturing these patterns provided the network is trained appropriately. The training time of the network may be quite long, but once implemented, the system can instantaneously predict the location of a terminal. It requires very limited local memory and computation power of the terminal.

In all previous location management schemes based on the movement history of the user, the mobility profile of each user is to be maintained at the network controller, or the user. However by our proposed scheme, the neural network is trained off-line by the movement history of a large number of users, and during run time, only the recent movement history (just 3 to 5 cells traversed last) is to be stored in the network controller at each update or call termination, the rest is managed by the terminal itself. Therefore, it reduces the storage cost and the maintenance cost appreciably, making the implementation much more cost-effective. Again, since one hybrid network is sufficient for the whole network, the training overhead would remain within limit.

However, the success of the scheme is highly dependent on the choice of training data. The experimental study with simulated movement profiles shows high success rate of prediction, most interestingly, by the SOFM-MLP network. It reveals the fact that the movement patterns of the whole set of subscribers really form some clusters in the input space. Hence MLP networks can be trained for given clusters resulting more accuracy in prediction. Moreover, the results show that with this scheme, the value of the threshold distance may be increased much without any significant rise in paging cost. It indicates that this technique would also reduce the update cost appreciably. However, the models are yet to be studied extensively by real data.

References

[1] I.F. Akyildiz and J.S.M. Ho, Movement-based location update and selective paging for PCS networks, IEEE/ACM Transactions on Networking 4(4) (1996) 629–638.

[2] I.F. Akyildiz and J.S.M. Ho, Dynamic mobile user location update for wireless PCS networks, Wireless Networks 1(2) (1995) 187–196.

[3] B.R. Badrinath and T. Imielinski, Location management for networks with mobile users, in: *Mobile Computing*, T. Imielinski and H.F. Korth eds., (Kluwer Academic Publishers, 1996) 129–152.

[4] A. Bamoy, I. Kessler and M. Sidi, Mobile users: To update or not to update? Wireless Networks 1(2) (1995) 175–185.

[5] A. Bhattacharya and S.K. Das, Lezi-Update: An information- Theoretic approach to track mobile users in PCS networks, Proc. ACM/IEEE Mobicom '99, Seattle, WA 5(5) (1999) 1–12.

[6] Y. Birk and Y. Nachman, Using direction and elapsed-time information to reduce the wireless cost of locating mobile units in cellular networks, Wireless Networks 1(4) (1995) 403–412.

[7] T.X. Brown and S. Mohan, Mobility management for personal communication systems, IEEE Transactions on Vehicular Technology 46(2) (1997) 269–278.

[8] G. Chakrabarty, Efficient location management by movement prediction of mobile host, in: *Proc. Int. Workshop on Distributed Computing IWDC 2002, Lecture Notes in CS*, Vol. 2571 (Springer 2002) pp. 142–153.

[9] S. Haykin, *Neural Networks: A Comprehensive Foundation* (McMillan College Publishing Co., New York, 1994).

[10] J.S.M. Ho and I.F. Akyildiz, Mobile user location update and paging under delay constraints, ACM-Baltzer J. Wireless Networks 1(4) (1995) 413–425.

[11] D. Hong and S.S. Rappaport, Traffic model and performance analysis for cellular radio telephone systems with prioritized and nonprioritized hand-off procedures, IEEE Transactions on Vehicular Tech. 42 (1993).

[12] K. Ivanov and G. Spring, Mobile speed sensitive handover in mixed cell environment, in: *Proc. IEEE Vehicular Tech. Conference* (1995) pp. 892–896.

[13] S.J. Kim and C.Y. Lee, Modeling and analysis of the dynamic location registration and paging in cellular systems, IEEE Transactions on Vehicular Technology 45(1) (1996) 82–90.

[14] T. Kohonen, The self-organization map, Proc. IEEE 78(9) (1990) 1464–1480.

[15] J. Li, H. Kameda and K. Li, Optimal dynamic mobility management for PCS networks, IEEE/ACM Trans. on Networking 8 (2000) 319–327.

[16] T. Liu, P. Bahl and I. Chlamtac, Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks, IEEE Transactions on Selected Areas of Commn. 16 (1998) 922–936.

[17] G.L. Lyberopoulos, J.G. Markoulidakis, D.V. Polymeros, D.F. Tsirkas and E.D. Sykas, Intelligent paging strategies for third generation mobile telecommunication systems, IEEE Transactions on Vehicular Technology 44(3) (1995) 543–553.

[18] U. Madhow, M. Honig and K. Steiglitz, Optimization of wireless resources for personal communications mobility tracking, IEEE/ACM Trans. on Networking 3 (1996) 629–638.

[19] S. Pal, J. Das and K. Majumdar, A hybrid neural architecture and its application to temperature prediction, in: *Proc. Joint Int. Conf. ICANN/ICONIP, 2003, Lecture Notes in CS* Vol. 2714 (Springer 2003), pp. 581–588.

[20] G. Pollini and C.-L. I, A profile-based location strategy and its performance, IEEE Transactions on Selected Areas of Commn. 15(8) (1997).

[21] C. Rose, Minimizing the average cost of paging and registration: A timer-based method, Wireless Networks 2(2) (1996) 109–116.

[22] C. Rose and R. Yates, Minimizing the average cost of paging under delay constraints, Wireless Networks 1(2) (1995) 211–219.

[23] S. Tabbane, An alternative strategy for location tracking, IEEE Transactions on Selected Areas of Commn. 13 (1995) 880–892.

[24] V.W.S. Wong and V.C.M. Leung, An adptive distance-based location update algorithm for next-generation PCS networks, IEEE Transactions on Vehicular Technology 19(10) (2001) 1942–1952.

[25] M. Zonoozi and P. Dassanayake, User mobility modeling and characterization of mobility patterns, IEEE Transactions on Selected Areas of Commn. 15 (1997) 1239–1252.

Kausik Majumdar Received the B.E. degree in Electronics & Telecommunication from Jadavpur University, Kolkata in 2003. He is presently studying for M.Tech. degree in Optoelectronics & Optical Communication from IIT Delhi. Research interests include optical communication, computer networks, semiconductor devices and neural networks.
E-mail: k_majumdar1@rediffmail.com

Nabanita Das received the B.Sc. (Hons.) degree in Physics in 1976, B.Tech. in Radio Physics and Electronics in 1979, from the University of Calcutta, the M.E. degree in Electronics and Telecommunication Engineering in 1981, and Ph.D in Computer Science in 1992, from Jadavpur University, Kolkata. Since 1986, she has been on the faculty of the Advanced Computing and Microelectronics unit, Indian Statistical Institute, Calcutta. She visited the department of Mathematik and Informatik, University of Paderborn, Germany, under INSA scientists' exchange programme. She has co-authored many papers published in International journals of repute. She has acted as the co-guest Editor of the special issue on 'Resource Management in mobile, ad hoc and sensor networks' of Microprocessors and Microsystems, by Elsevier. She has acted as program chair of International workshop on distributed computing, IWDC 2004, and also as co-editor of the proceedings to be published as LNCS by Springer. Her research interests include parallel processing, interconnection networks, wireless communication and mobile computing. She is a senior member of IEEE.
E-mail: ndas@isical.ac.in