Entropy-Based Knowledge Spreading and Application to Mobility Prediction

Jean-Marc François
francois@run.montefiore.ulg.ac.be

Guy Leduc
guy.leduc@ulg.ac.be

Reseach Unit in Networking (RUN)
Department of Electrical Engineering and Computer Science
Institut Montefiore, B28 — Sart-Tilman
University of Liège
4000 Liège, Belgium

ABSTRACT

The low quality of service provided by wireless networks does not facilitate the setup of long-awaited services, such as video conversations. In a cellular network, handoffs are an important cause of packet losses and delay jitter. These problems can be mitigated if proactive measures are taken. This requires each cell to guess the next handoff of each mobile terminal, a problem known as *mobility prediction*. This prediction can occur thanks to some clues (such as signal strength measurements) giving information about the terminals motion. For example, a clue that locates on which road a mobile is moving is likely to be interesting for all the prediction-enabled cells along that road —and should therefore be sent to them. This paper proposes a new method aimed at selecting the most relevant clues and finding where to propagate those clues so as to optimize mobility predictions. The pertinence of a clue is measured using information theory and by means of decision trees. This pertinence estimation is exchanged between the cells and allows to build a "relevance map" that helps determine where clues should be sent. It is adapted to the characteristics of wireless terminals such as low bandwidth and processing power.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication

This work has been partially supported by the Belgian Science Policy in the framework of the IAP program (Motion P5/11 project) and by the European E-Next NoE.

CoNEXT'05, October 24–27, 2005, Toulouse, France.

General Terms

Algorithms, Design

Keywords

Mobility prediction, Information theory, Decision trees

1. INTRODUCTION

Next generations of wireless networks are expected to allow new multimedia services requiring a high QoS level and characterized by a high throughput.

Mobile terminals experience frequent handoffs which are likely to cause delay jitter and packet losses. One way to mitigate this problem is to act proactively against next handoffs (using packets bicasting, anticipated address resolution, resource reservations,...).

Mobility prediction allows those proactive actions to take place. Its purpose is to guess where (and possibly when) terminal's next handoff(s) will occur. This involves monitoring the mobiles to infer a model of their motion. The kind of information used to perform this monitoring is varied: GPS coordinates, motion speed, access router identification, signal strength,...

It has been shown that the prediction ratio (*i.e.* the ratio between the number of correct next-cell predictions and the total number of predictions made) can strongly impact the network QoS ([2]). It is thus important to design mobility prediction schemes that are as accurate as possible and fitted to mobile networks mainly characterized by bandwidth, energy, and processing constraints.

Mobility prediction techniques can be divided into two schemes:

- either mobiles store their most frequent paths locally (at the expense of memory consumed) or they get them from a "home repository" (increasing delays and consuming bandwidth [7, 5]);
- or prediction is done locally, usually in each access router, using the typical behaviour of mobiles encountered in the past ([9]).

Both methods have pros and cons. Keeping pieces of information —such as the frequent paths of mobiles— for long periods of time can be profitable if they help guess mobile movements later on.

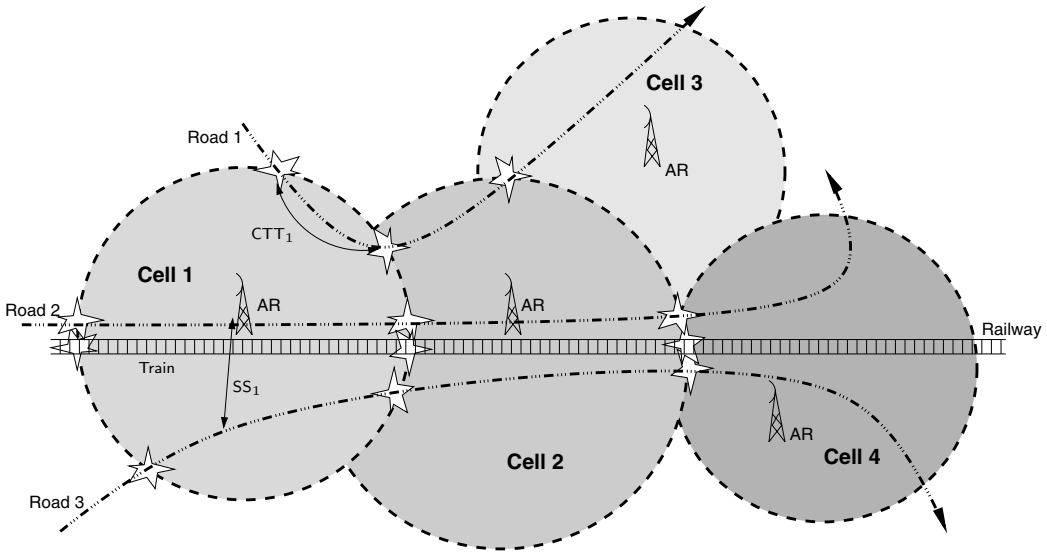

Figure 1: Overview.

Ideally, each mobile would know where each piece of information could be exploited and only keep those having a high probability of being used. This leads to a hybrid method which combines the advantages of both schemes. This hybrid method can be implemented thanks to information theory.

Some concepts directly linked to information theory — such as text compression algorithms— have already been used to tackle similar problems ([1, 6]). This is a sensible approach since good text compressors usually are good predictors. The method proposed in this paper might at first seem very different from those works since their aim is to reduce the update/paging overhead while this article shows a way to efficiently spread the data used to perform prediction. To decrease that overhead, [1, 6] build Lempel-Ziv tries that are used to predict each mobile's next cells; those tries could thus be compared to the decision trees used in this paper.

Section 2 gives an overview of the proposed method. Section 3 is a short introduction to information theory and decision trees. Sections 4 and 5 detail the approach and give some results of simulations. Section 6 shows results related to long-term predictions that would be difficult to achieve using standard schemes. The last two sections gives some future works and concludes.

2. PRINCIPLE

First, let's give an overview of the framework studied here. We suppose that *mobile nodes* (MNs) cross a cellular network and stay connected through *access routers* (ARs, one per cell).

Some clues gathered during the motion of a MN can help guess its future movements. They can be collected by the MN itself or by ARs. Those routers can use them to predict MNs' next cell. The precise way those clues are generated and exchanged between MNs and ARs will be described in section 4.1.

Figure 1 shows a situation where each cell *(a)* monitors each MN's mean signal strength (SS) experienced during its journey and *(b)* compute each MN's cell travelling time (CTT). Those clues have a subscript describing to which cell they are related (*e.g.* CTT_1 denotes the time needed to cross cell 1). Mobiles moving along the railway have received a special '*train*' clue (when they enter a station, for example). The clues can be considered as random variables.

Cell 2 is crossed by 3 roads and a railway; it should figure out if a MN's next cell will be cell 3 or 4. This prediction can be done using CTT_1 alone: mobiles located on road 1 are characterised by a short travelling time. On the other hand, the signal strengths SS_1 associated with roads 1 and 3 are about the same: this variable does not help much cell 2's prediction process.

Cell 4 must also guess if MNs passing by are located on road 2, road 3, or on the railway. Fortunately, this can be inferred from SS_1 (mobiles on road 3 are characterized by a weaker signal since this road is on the cell periphery) and the *train* clue (which discriminates between road 2 and the railway). Consequently, some variables should be propagated from one cell to another. For example, the *train* variable is certainly pertinent for all the cells along the railway; the same could be said about any variable that pinpoints the road a mobile is driving on.

We assume that MNs are responsible for sending variables from cell to cell. Since, in general, those terminals have strong memory and bandwidth constraints, we consider that the maximum number of clues it can hold is low. Other strategies could be conceived, such as asking the cells to broadcast data to neighbouring cells, or to rely on a MN's home repository, but both methods are likely to be more complex, consume bandwidth, and add delays.

This short example emphasizes two kinds of problems.

Problem 1: information estimation. Each cell has to estimate the relevance of each variable regarding next cell prediction. It is proposed to use the mutual information between the next cells and each variable as a relevance estimation. Mutual information is a direct measure of how the next cell prediction uncertainty is reduced thanks to the

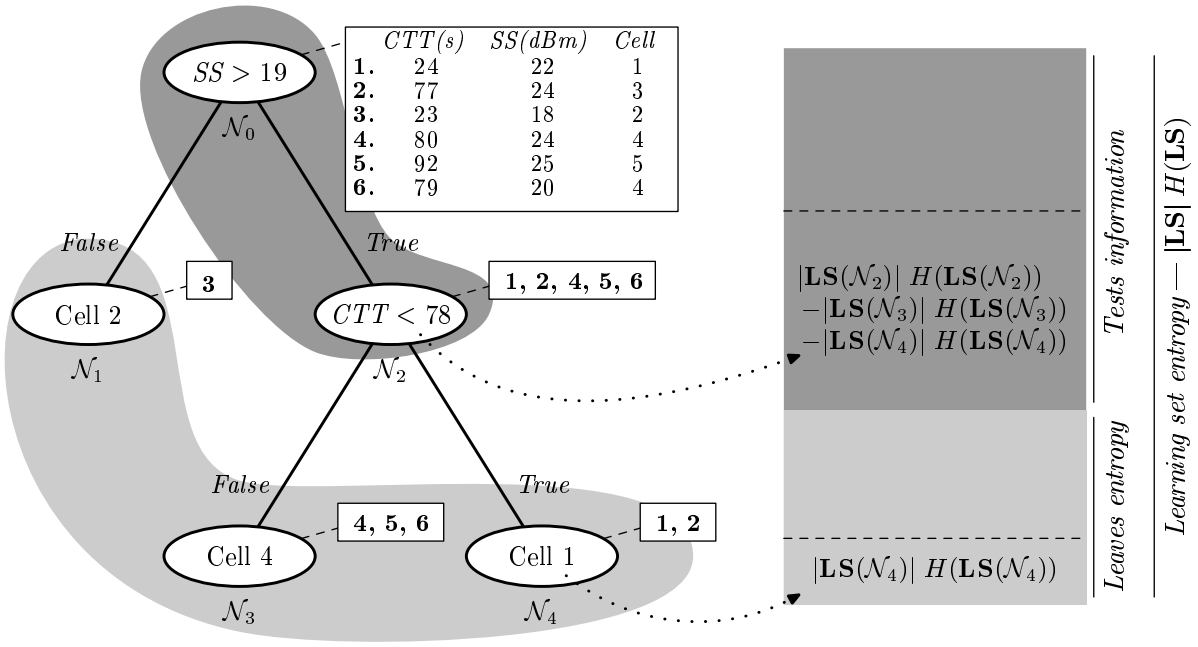

Figure 2: ***A simple decision tree.*** **This tree guesses a mobile's next cell using two measurements: a cell traveling time (CTT) and a signal strength (SS); those are the tree attributes $\mathcal{V}_1$ and $\mathcal{V}_2$. The $\mathcal{X}$ variable is the next cell and, consequently, the classes for this problem are cells 1 to 5.**

considered variable (section 3 gives a short reminder of information theory).

Problem 2: data spreading. Once —a reasonable approximation of— the information held by each variable for each cell is known, a sensible way of selecting the variables memorized by the MNs should be found. This selection scheme should be chosen so as to maximize the information gained by the cells along the mobiles paths.

One could argue that wireless terminals now have a comfortable amount of memory and could simply record all the collected clued, relevant or not. This calls for four comments. First, we will see that clues have to be exchanged regularly between the MN and ARs; thus, keeping unrelevant variables means wasting bandwidth. Second, this method should be fitted to mobiles' least common denominator: down-market terminals should not be omitted. Third, the amount of variables to consider could quickly grow since they could be frequently generated (*e.g.* GPS updates) and stay relevant for a long period of time. Finally, the technique presented here is expected to be applicable to other contexts and could therefore be considered a general "data routing" strategy aimed at maximizing data relevance.

3. INFORMATION THEORY AND DECISION TREES

Information theory ([10]) and decision trees ([8]) are building blocks of the work presented in the following sections. Those topics are thus briefly introduced here.

3.1 Information theory

Consider a discrete random variable $\mathcal{X}$, its *entropy* $H(\mathcal{X})$ and its *conditional entropy* knowing $\mathcal{Y}$ $H(\mathcal{X}|\mathcal{Y})$ defined as:

$$H(\mathcal{X}) = -\sum_{x\in X} P(x)\ \log P(x) \quad (1)$$

$$H(\mathcal{X}|\mathcal{Y}) = -\sum_{x\in X}\sum_{y\in Y} P(x,y)\ \log P(x|y) \quad (2)$$

where $P(x)$ is the probability of the event x, $P(x,y)$ the joint probability of x and y, $P(x|y)$ the probability of x given y, and X (resp. Y) the sample space of $\mathcal{X}$ (resp. $\mathcal{Y}$). In the following, the logarithm function is always supposed defined in base 2. The entropy quantifies the predictability of a random variable; an entropy equal to zero corresponds to a variable whose realizations values are known in advance.

The *mutual information* between variables $\mathcal{X}$ and $\mathcal{Y}$ is equal to:

$$I(\mathcal{X};\mathcal{Y}) = H(\mathcal{Y}) - H(\mathcal{Y}|\mathcal{X}) = I(\mathcal{Y};\mathcal{X}) \quad (3)$$

and measures the dependence between $\mathcal{X}$ and $\mathcal{Y}$ (*i.e.* the information gained on $\mathcal{Y}$ knowing $\mathcal{X}$; $I(\mathcal{X};\mathcal{Y}) = 0$ if $\mathcal{X}$ and $\mathcal{Y}$ are independent).

3.2 Decision trees

A *decision tree* encodes a set of tests related to random variables (say $\mathcal{V}_1, \mathcal{V}_2, \dots$) as a tree. This tree aims at encoding a conditional probability law $P(\mathcal{X}|\mathcal{V}_1, \mathcal{V}_2, \dots)$ where $\mathcal{X}$ is a discrete random variable. A tree $\mathcal{T}$ defines a leaf distribution and each leaf T_i matches a probability distribution $P(\mathcal{X}|T_i)$.

The tree is built in such a way that the entropy $H(\mathcal{X}|\mathcal{T})$ is reduced as much as possible; this means that once the trees tests have been applied, the remaining uncertainty about $\mathcal{X}$

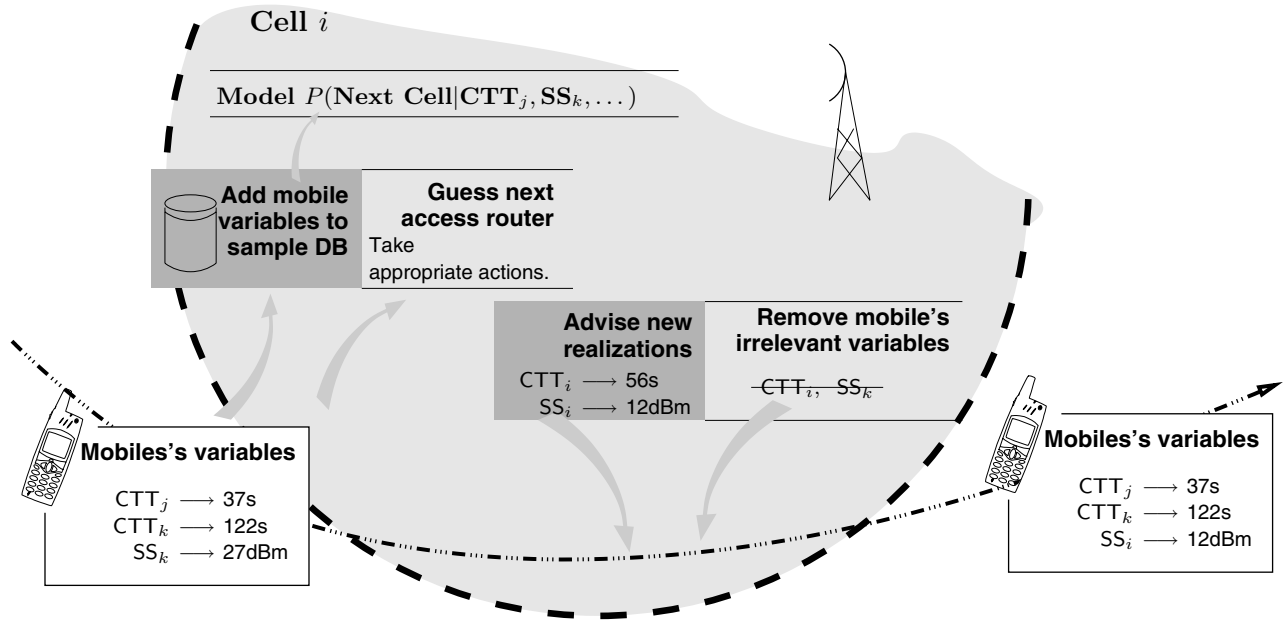

Figure 3: ***Mobility prediction overview.*** **Each cell performs a variable selection. Variables names have a subscript describing where they have been generated; thus, $\mathbf{CTT}_j$ and $\mathbf{SS}_i$ represent the time taken to cross cell j and a signal strength measurement done in cell i.**

is small. The left part of figure 2 shows a simple decision tree.

In this context, variables' realizations are often denoted *samples*, $\mathcal{V}_i$ variables *attributes* and $\mathcal{X}$ the *goal attribute*.

We will see that the tree learning algorithm plays an important role in what follows. It is thus briefly described below[1].

The problem is to deduce the distribution $P(\mathcal{X}|\mathcal{V}_1, \mathcal{V}_2, \dots)$ given a set of realizations (or *learning set*, **LS**). A test is first chosen for the root node; this node has two sons, $\mathcal{N}_{\text{true}}$ and $\mathcal{N}_{\text{false}}$ corresponding the test's possible outcomes. The node $\mathcal{N}_{\text{true}}$ (resp. $\mathcal{N}_{\text{false}}$) matches a subset of the learning set —denoted $\mathbf{LS}(\mathcal{N}_{\text{true}})$ (resp. $\mathbf{LS}(\mathcal{N}_{\text{false}})$)— so that the test applied to the proper attribute of its elements is true (resp. false). The same procedure can be repeated for both root's son nodes.

Tree tests are chosen so as to maximize the information gained on the goal attribute; this quantity of information is estimated by subtracting the learning set entropy once the test outcome is known to the entropy before the test is applied. Formally, the information brought by the test matching node $\mathcal{N}_{\text{f}}$ is estimated by:

$$I(\mathcal{N}_{\text{f}}) = H(\mathbf{LS}(\mathcal{N}_{\text{f}})) - \frac{|\mathbf{LS}(\mathcal{N}_{\text{ls}})|}{|\mathbf{LS}(\mathcal{N}_{\text{f}})|} H(\mathbf{LS}(\mathcal{N}_{\text{ls}})) - \frac{|\mathbf{LS}(\mathcal{N}_{\text{rs}})|}{|\mathbf{LS}(\mathcal{N}_{\text{f}})|} H(\mathbf{LS}(\mathcal{N}_{\text{rs}})) \quad (4)$$

where $\mathcal{N}_{\text{ls}}$ and $\mathcal{N}_{\text{rs}}$ are the left and right sons of $\mathcal{N}_{\text{f}}$ and where $|\mathbf{S}|$ denotes the cardinality of set $\mathbf{S}$. A set of samples' entropy is estimated using (1) applied to the estimated distribution of $\mathcal{X}$ for this set.

[1]This work could easily be adapted to other learning methods (*e.g.* [3, 4]).

The tree is recursively expanded until a pruning criterion is met (refer to [8] for in-depth explanations).

An interesting property links the tests information, the learning set entropy and the leaves entropy: the sum of the weighted tests information (as estimated by equation 4) is equal to the learning set entropy ($|\mathbf{LS}|H(\mathbf{LS})$) minus the weighted entropy of the samples sets associated with leaves (as given by equation 1). The weights are given by the number of samples matching the corresponding node.

This section is summarized in figure 2. It shows a small tree that could be used in a cellular network to guess in which cell a mobile might be going given the time it took to cross the cell (its cell traveling time, CTT) and a measured signal strength (SS). This tree has been pruned ($\mathcal{N}_3$ and $\mathcal{N}_4$ might have been split).

The right-hand side of the figure depicts that the learning set entropy can be divided into two parts: the tests information (dark grey) and the leaves entropy (light grey). The formulas needed to compute the information gained in $\mathcal{N}_2$ and the entropy of $\mathcal{N}_4$ —which are particular cases of (4) and (1)— are explicitly given. This example tree is quite inefficient: the information collected by its tests is only two thirds of the learning set entropy.

Each leaf $\mathcal{N}_l$ can be labelled with a *class*, *i.e.* the most frequent value of the $\mathcal{X}$ attribute for the set $\mathbf{LS}(\mathcal{N}_l)$.

Sample classification (*i.e.* finding a sample's most likely $\mathcal{X}$ attribute value) can be done efficiently. It only requires to cross the tree from its root down to a leaf according to the outcomes of the nodes' tests applied to the sample. The complexity of tree building is not considered a critical issue since this operation is not supposed to occur frequently[2].

[2]The complexity of the tree building algorithm is $KN \log N$ where K is the number of attributes and N the cardinality of the training set.

4. MOBILITY PREDICTION

4.1 Principle

We suppose that MNs travel a cellular network and collect clues related to their motion. Those clues can be generated by the network (*e.g.* a cell identifier) or by the mobile terminal itself (*e.g.* a received signal strength measurement). Since, in general, MNs are small terminals, we consider that the maximum number of clues it can hold is low. The problem is thus to find which clues are worth keeping.

Figure 3 shows how the clues can be handled. We consider that MNs send these pieces of information to an AR which is responsible for analyzing them so as to guess the terminal's next cell (*i.e.* collecting them in a *samples database* as depicted figure 2 and building a decision tree). Clues can be considered by the ARs as realizations of random variables likely to give information on the random variable matching the *a priori* distribution of possible next cells.

Notice that the memory and processing consuming tasks (*i.e.* collecting a samples database and using it to predict a mobile's destination) are left to the ARs: the mobile acts as a dumb terminal.

The ARs use the mobile's variables to guess its probable next cell, and then take appropriate actions (such as resources reservation). This requires the samples database to be populated with the actual next cells reached by the mobiles; this can be solved by asking all the cells to warn the previous cell each time a handoff occurs.

During the mobile's cell crossing, the AR generates new variables that can be added to those the mobile already holds. Since the MN's amount of memory is limited, the AR asks to remove variables considered not relevant enough.

The next section is dedicated to next cell prediction. The following two deal with problems we have already mentioned: determining the information received from a variable realization and choosing which variable should a MN keep or remove.

4.2 Next cell prediction

The samples database collected by an AR can be represented as a table where each row is a sample and each column matches a variable; we denote $\mathcal{A}_k$ the set of those variables for cell k. This set holds all the variables that have already been observed.

All the mobiles entering a given cell will not hold the same variables (since it is likely that they did not cross the same cells). The variables that belong to $\mathcal{A}_k$ but not to the set of variables held by an incoming mobile are assigned the special *notApplicable* value. This value should be easy to discriminated from the other realizations by the tree (*e.g.* a negative number if the variable domain is positive numbers).

An AR can build a decision tree as soon as the database is considered populated enough. Cell k's tree is denoted $\mathcal{T}_k$; notice that it models the probability distribution of next cells given the mobile clues. Next cell prediction just amount to classify the sample made of the MN's variables using the tree.

4.3 Information estimation

4.3.1 Principle

Once a mobile leaves a cell, the AR should filter its variables and remove those expected to be less useful to the

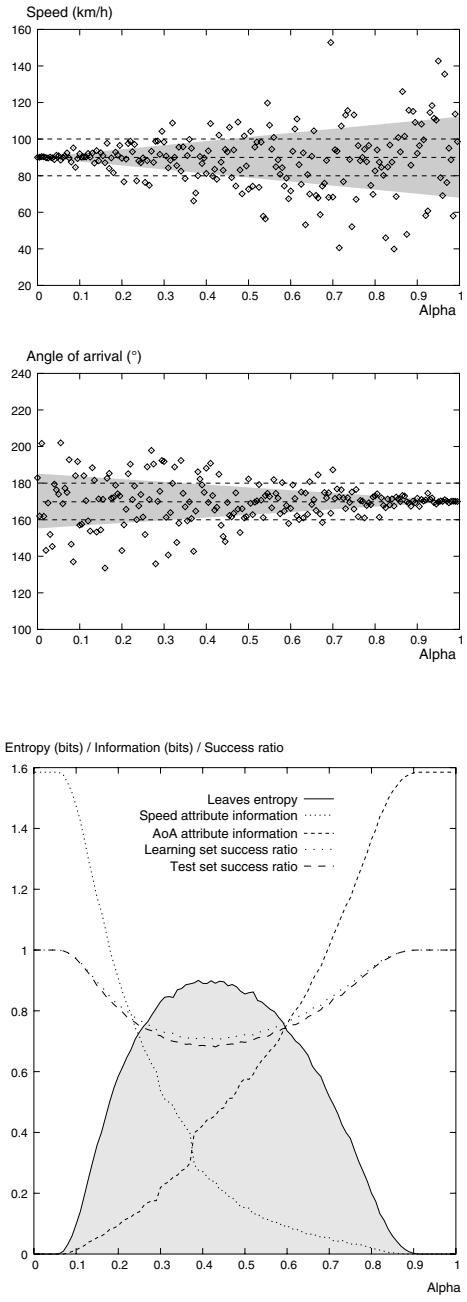

Figure 4: ***Attributes information. Top:*** **distributions of the speed and angle of arrival variables. The dashed lines depict the mean values for each road and the grey zone corresponds to the distributions' standard deviations.** ***Bottom:*** **various values extracted from the decision trees for different values of the α parameter.**

MN's next cells. Section 4.4 will show that this requires that each cell estimates how useful a variable is to its own prediction (*i.e.* the information it gives regarding next cell guessing).

This can be estimated thanks to the decision tree introduced in the previous section. We have seen that the information $I(\mathcal{N})$ of each test node $\mathcal{N}$ composing the tree can be estimated by (4); a way to evaluate the information related to an attribute A is thus to sum the information of all the tree's tests involving A. This leads directly to the following formula:

$$I(A) = \sum_{\text{Test node } \mathcal{N}} \frac{|\mathbf{LS}(\mathcal{N})| \cdot I(\mathcal{N})}{|\mathbf{LS}|} \quad (5)$$

where the sum operates over the nodes testing A.

4.3.2 Example

Figure 4 gives a simple application of (5). A mobile speed and angle of arrival are used to figure out on which road it is located. There are three roads, each with a specific mean speed (80, 90 or 100 km/h) and angle of arrival (160, 170 or 180°). The speed and angle values are distributed according to:

$$\begin{aligned} V_{\text{speed}} &= \overline{V_{\text{speed}}} + \alpha \cdot N(0;25) \\ V_{\text{AoA}} &= \overline{V_{\text{AoA}}} + (1-\alpha) \cdot N(0;15) \end{aligned}$$

where $N(\mu, \sigma)$ denotes a normal distribution with mean μ and standard deviation σ. The α factor is a constant between 0 and 1 describing if the Gaussian noise is added to one variable or the other. The first two plots of figure 4 depicts those distributions. The roads are equally populated, yielding an *a priori* system entropy of ~ 1.6 bits (log 3).

The information theory related values we have discussed so far have been estimated using decision trees built for various values of α (bottom of figure 4). When α is nearly equal to zero (resp. one), the system entropy is entirely compensated by the information given by the speed (resp. angle) variable. Notice that the attribute's information curves cross each other when $\alpha \approx .37$ (not .5) since the variables' standard deviations are different. At this moment, the curves have a small gap because the tree's root node's attribute changes; the algorithm presented here over-estimates the information brought by attributes near the tree's root. Thus, in practice, it cannot be assumed that equally meaningful attributes will receive the same information measurement.

Prediction success ratios have been estimated with the learning set (*i.e.* classifying the samples used to build the tree) and with an independent test set. The resulting curves are nearly merged, showing that no over-learning (leading to over-estimated variable information) has occurred. The sum of the entropy of each trees' leaves has been plotted. This sum shows the remaining uncertainty about the samples' class once all the tree's tests have been applied. As expected, the success ratio is low when the leaves entropy is high.

4.3.3 Biasing the tree building

When two variables $\mathcal{V}_1, \mathcal{V}_2$ hold the same information regarding $\mathcal{X}$ (*i.e.* $I(\mathcal{X};\mathcal{V}_1) \approx I(\mathcal{X};\mathcal{V}_2) \approx I(\mathcal{X};\mathcal{V}_1,\mathcal{V}_2)$), (5) tends to divide the information $I(\mathcal{X};\mathcal{V}_1,\mathcal{V}_2)$ evenly among $\mathcal{V}_1$ and $\mathcal{V}_2$. Unfortunately, as we will see later, it would be preferable to clearly differentiate those variables and to associate a high information with one of them —and a low information with the other.

Procedure `spread`($\mathcal{V}_i^j$) Information spreading of $\mathcal{V}_i^j$ for cell k

```
1  var Ī_{V_i^j} : N_{V_i^j} →]0;1];
2  if N_{V_i^j} = ∅ and j ≠ k then
3  |  ∀p ∈ P_{V_i^j} : send treeInfo(T_k, V_i^j) to p;
4  |  return;
5  end
6  ∀n ∈ N_{V_i^j}: receive Ī_{V_i^j}(n);
7  if N_{V_i^j} ≠ ∅ and j ≠ k then
8  |  ∀p ∈ P_{V_i^j} : send treeInfo(T_k, V_i^j) +
   |      Σ_{n∈N_{V_i^j}} Ī_{V_i^j}(n) · f_{V_i^j}(n) to p;
9  end
10 return Ī_{V_i^j}(·)
```

To get this behaviour, we slightly change the way node tests are chosen while building a tree. We have said that tests are selected so as to maximize the test information given by (4); we propose to slightly change this behaviour. A test is chosen in the set of tests T such that:

- it contains the test t that maximize (4), let $I_{\max}$ be t's information;
- the difference between $I_{\max}$ and the information of the other tests in T is smaller than a constant chosen equal to 0.01 bit[3].

The "first" test among the elements of T is finally chosen using an arbitrary order relation defined on the tests' attributes *a priori*. This way, the problem mentioned above is mitigated since, when it is reasonable, the same variable is chosen for most tree nodes.

4.4 Data spreading

The previous section shows how to estimate the information carried by a variable *for a specific cell*. Our aim is to maximize the information collected by the cells during the mobile journey, so we need *(a)* to let a cell figure out the expected information brought by an attribute along a mobile's future path and *(b)* to deduce from that information which attributes should be carried by a mobile.

In the following, $\mathcal{V}_i^j$ will denote cell j's i-th variable and V_i^j its realization. For example, the variables generated by cell i in figure 3, CTT_i and SS_i, might be denoted $\mathcal{V}_0^i$ and $\mathcal{V}_1^i$.

4.4.1 Attribute's expected information

We try to guess how much information can the $\mathcal{V}$ variable give to the cells on a MN's path.

We first consider an AR that has never seen any MN leaving its cell with $\mathcal{V}$ in its attributes set. Obviously, $\mathcal{V}$ will not give any information to the MN's next cells and the expected information for $\mathcal{V}$ amounts to the information given to the current cell (see procedure `spread`, lines 2–5 —the `treeInfo` procedure is given by (5)).

[3]Simulations results have shown to be insensible to the actual value of this parameter as long as it is chosen small enough.

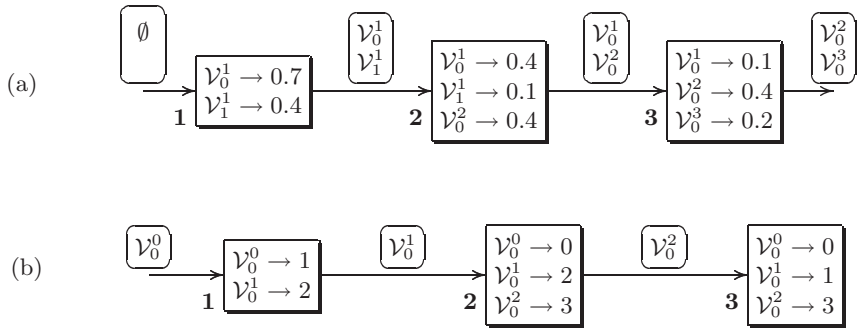

Figure 6: Information spreading. ***(a)*** **Normal behaviour** ***(b)*** **A catch. The bold rectangles give the variables expected information for the next cells.**

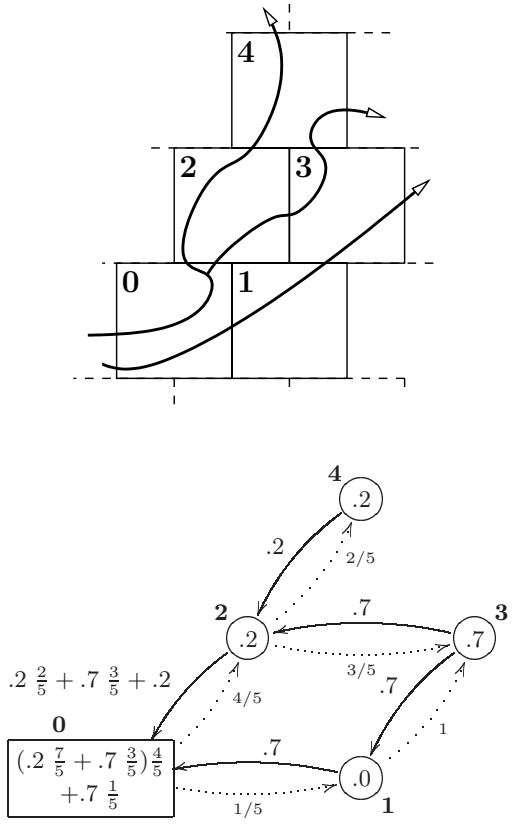

Figure 5: ***Information spreading:*** **principle. Dotted arrows depict the mobiles movement; their labels give the value of the $f_\mathcal{V}$ functions. Plain arrows show the expected information propagation back to cell 0.**

This expected information can be sent to the *previous cells*, *i.e.* the cells that have sent at least one mobile holding $\mathcal{V}$; the set of previous cells is denoted $\mathbf{P}_\mathcal{V}$. Similarly, the *next cells* set, $\mathbf{N}_\mathcal{V}$, contains the cells where at least one mobile holding $\mathcal{V}$ has gone.

It is likely that MNs will not visit cells in $\mathbf{N}_\mathcal{V}$ homogeneously. We denote $f_\mathcal{V}$ the function $\mathbf{N}_\mathcal{V} \to]0;1]$ such that $f_\mathcal{V}(n)$ is the ratio of mobiles holding $\mathcal{V}$ leaving the current cell for cell n ($\sum_{n\in\mathbf{N}_\mathcal{V}} f_\mathcal{V}(n) = 1$).

Once an AR knows, for all its next cells, the expected information received along a mobile's path starting with cell n —denoted $\overline{I}_\mathcal{V}(n)$—, it can compute its own information expectation using *(a)* the information given by $\mathcal{V}$ *(b)* the values $\overline{I}_\mathcal{V}(n)$ weighted by $f_\mathcal{V}(n)$; this is summarized in the `spread` procedure, line 8.

Figure 5 gives an overview of the information spreading process for a variable emitted by cell 0. The top part shows 5 cells crossed by a few roads. The bottom part depicts the spreading algorithm applied to this simple topology. It shows the local information as computed by equation 5 (circled numbers), $\mathbf{N}_\mathcal{V}$ sets (dotted arrows), the corresponding $f_\mathcal{V}$ functions (dotted arrows' labels) and the information estimation propagation from cells 3 and 4 back to cell 0 (plain arrows).

Notice that the algorithm presented here does not deal with cycles: the motion graph is supposed acyclic. Even if this should not cause any real impact in practice, it would be desirable to modify it to get rid of that limitation.

4.4.2 Attributes selection

In the following, we assume that every MN can stock a fixed, finite number of attributes. Once all the cells have learned their $\overline{I}_{\mathcal{V}_i^j}$ functions (one function for each variable that might leave the cell), the network's ARs can efficiently select a MN's most pertinent variables.

Ideally, this selection should be such that the expected sum of the MNs' variables information for all the visited cells is maximized; we call it the *optimum selection scheme*.

Since the $\overline{I}_{\mathcal{V}_i^j}$ functions are known, the straightforward wait to select m variables out of a variables set V is to sort them by decreasing information expectation and pick up the first m. A small improvement of this algorithm is to first guess the MN's next cell n so as to use $\overline{I}_\mathcal{V}(n)$ to measure the usefulness of $\mathcal{V}$.

Figure 6 *(a)* shows the normal behaviour of this algorithm.

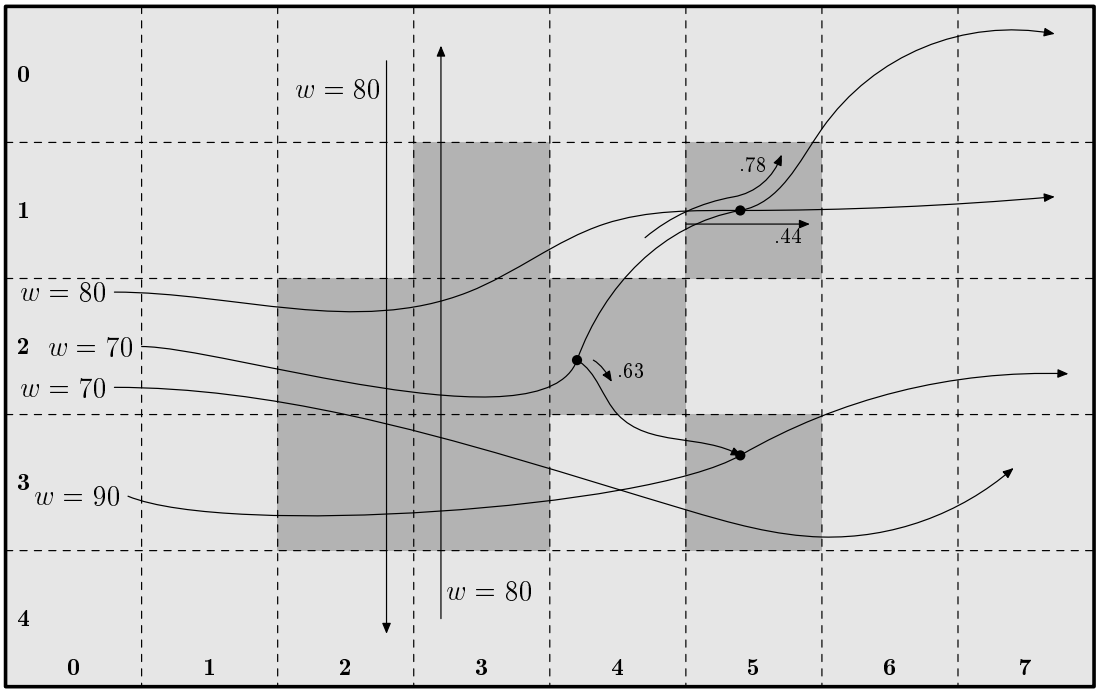

Figure 7: The simulation setup.

In this example, a MN handles 2 attributes (rounded rectangles) and crosses 3 cells laid out linearly. The bold rectangles show the variables' expected information (*e.g.* in cell 1, $\overline{I}_{\mathcal{V}_0^1}(2) = 0.7$); one can verify that variables are selected as explained above. Notice that on that simple topology, the variables information have to decrease from a cell to the next, the difference being the information gained locally (*e.g.* since $\mathcal{V}_0^1$ drops from 0.7 in cell 1 to 0.4 in cell 2, 0.3 bits are local to cell 2).

This is where the observation formulated section 4.3.3 is relevant: it helps get very contrasted variables lists with very relevant and very irrelevant variables and few variables in between.

Figure 6 *(b)* demonstrates that this simple solution is not optimal. Cell 1 chooses to send $\mathcal{V}_0^1$, but this variable is immediately replaced by $\mathcal{V}_0^2$ in cell 2. The attribute $\mathcal{V}_0^1$ does not give any information to cell 2, but $\mathcal{V}_0^0$ does: it was the best variable to send between cell 1 and 2. The algorithm is thus a victim of its greediness.

5. SIMULATION

5.1 Simulation setup

The work presented above has been tested on a simulation. A grid made of 8x5 cells composes the simulation setup (figure 7).

Two roads let the MNs cross the map vertically. Five roads allow the mobiles to cross the terrain from left to right. If the roads had been bidirectional, inferring a MN's direction would have required to know one of its previous cells. Since every variable is implicitly related to the cell that generated it, knowing the mobile's direction would have been an easy task. It has thus been decided to use a simpler configuration with unidirectional roads since bidirectional ones would not have changed the simulations results significantly.

Each cell generates two kinds of variables: the MN's *cell travelling time* (CTT) and a *noise* variable with randomly generated realizations. The latter demonstrates the ability of the system to differentiate between relevant and less relevant variables. One should note that this *noise* variable can nevertheless bring some information since a noise variable indicates at least that the mobile has crossed the cell that generated it; this might give a piece of information about its future path.

One third of the MNs are created with a special *emergency* variable. This variable flags mobiles that are certainly going to cell (7,1). This shows that variables could be associated with destinations when the mobile path is known in advance; this emergency variable could be associated with emergency vehicles that are very likely to go to the nearest hospital or police station. Another possibility would be to flag mobiles entering a train.

Each road is labelled with a weight w that indicates its significance; the higher this number, the faster the cell is crossed and the higher the number of mobiles using it. The cell travelling time is related to the road weight w by the relation $\text{CTT} = 120 - 30\ w/100 + N(0;5)$.

Crossroads are depicted with a black dot. When applicable, an arrow gives the ratio of mobiles going in each direction.

Dark grey cells are those where mobility prediction is needed: the next cell of MNs leaving those cells is not known in advance.

Three variable selection methods have been compared. The first is the most obvious scheme where the oldest variable is removed first; this is equivalent to putting variables in a bounded FIFO queue. The second method is based on

decision trees; each cell learns the parameters of a tree and use it to measure variables relevance. The less relevant variables are removed first. The cells do not exchange variable relevance estimations with each other. The rational behind this scheme is that if a cell finds a variable interesting, it is likely that it will also be relevant for its neighbours. Those schemes have been compared to the entropy-based scheme explained above.

Notice that those three methods differ only in the way the relevant variables are chosen. A direct consequence is that the amount of information exchanged between the mobiles and the ARs is the same; the only difference resides in the processing required by the ARs.

All the mobiles can carry the same fixed number of variables; this number is a simulation parameter. Its influence on the next cell prediction ratio has been quantified.

5.2 Results

Figure 8 shows, for 4 different cells, the ratio between the number of correct next cell predictions and the total number of predictions

The first plot is related to cell (2,2); it shows that all the methods perform equally. This is not surprising since mobiles are crossing the map from left to right: variable selection can only happen after a few cells have been crossed, so the interesting cells are on the right-hand side of the map.

Thus, the other plots are related to cells (4,2), (5,1) and (5,3); they allow to draw several conclusions. On this simple cell topology, the information-based method gives an optimum result with as few as 3 variables[4]. When the MNs' memory is low, the other methods always gives lower prediction ratios. The number of variables required to catch up varies from cell to cell but it can be as high as 11, which is more than one would expect for such a small map.

The improvement of the *locally optimized* method over the simple, *older-first* scheme does not seem to be worth the increased implementation complexity. This shows that the upstream propagation of variable relevance measurements is needed: variable significance is not a local property.

Even if they should be confirmed on a large scale experiment, those results are encouraging since the method's usefulness is expected to increase with the number of cells — and, thus, the number of potential variables— encountered. For instance, an attribute created when a mobile user enters a train will be useful for tens of cells along the railroad; the data spreading paradigm should allow to bring this observation to the fore.

6. LONG TERM PREDICTIONS

In some circumstances, it is possible to guess not only one, but several cells the mobile will eventually cross. Such long-term predictions can be achieved accurately when, for example, mobiles are moving in a train or on a highway and can be used, for example, to lower the number of location updates needed to perform paging.

A terminal destination can also be determined thanks to the value of one (or several) random variables realization(s) — such as the *emergency* variable introduced in section 5. The nature of those variables lets them give information to

[4]Experiments using an arbitrary large number of variables have shown that the prediction ratio cannot be increased further.

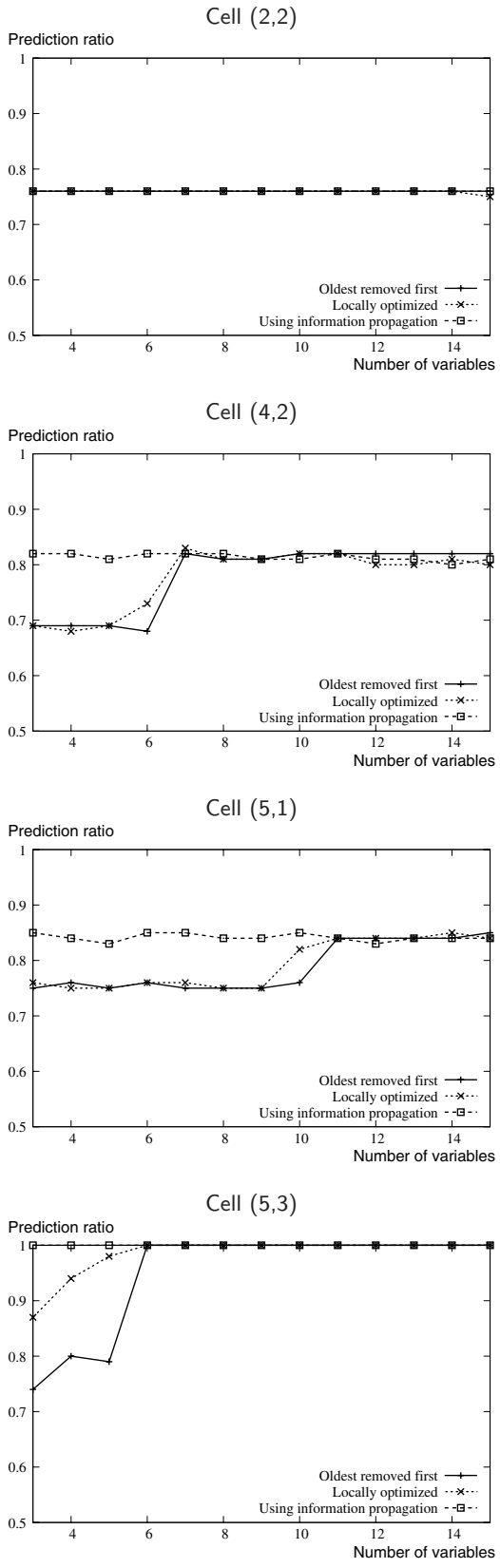

Figure 8: Simulation results for 4 cells depicted in figure 7.

a large number of cells, increasing the probability of being picked up by the cells on the mobile path thanks to the information spreading method. Such variables should allow to determine which cells the mobile will go through to reach its destination.

The following explains how long-term predictions can be obtained and, using simulations, shows how it performs.

The learning and information spreading techniques presented in section 3 and 4 do not have to be modified, thus we assume that the expected information brought by the variables has been computed and is known to every cell.

When a mobile enters a cell, the set of variable realizations V it brings can be used to predict its set of potential next cells $C = \{c_i\}$ and the associated probabilities $p_C : C \to [0, 1]$. Those probabilities are directly derived from $\mathbf{LS}(\mathcal{N}_l)$ where $\mathcal{N}_l$ is the decision tree's leaf corresponding to V.

The current cell can apply the variables selection algorithm as before (section 4.4.2), but does not know what the realizations of the variables added by the cell will be when the mobile performs its handoff. Those realizations are thus replaced by *unknown values* to produce a new V_C vector. To classify a sample with unknown values, a tree is allowed to explore both possible outcomes of a test and to average the results; the interested reader can refer to [8] for a detailed explanation.

This vector and the probability $p_C(c_i)$ can be sent to the potential next cell c_i. c_i repeats the same procedure to compute a vector $V_{C'}$ and to get the mobile's potential next cells $C' = \{c'_i\}$ and the probabilities function $p_{C'} : C' \to [0, 1]$. The product $p_C(c_i) \cdot p_{C'}(c'_j)$ gives the probability that the mobile crosses cells c_i and c'_j. This process can be repeated until the product $P_{c_i,c'_j,c''_k\ldots} = p_C(c_i) \cdot p_{C'}(c'_j) \cdot p_{C''}(c''_k) \ldots$ is small enough.

The probability to cross a given cell c is given by the sum of the products relative to a list of cells terminated with c (such as $P_{c_i,c'_j,\ldots c}$). Figure 9 gives this probability computed for the cells of the topology presented in the previous section. The results shown have been obtained for a mobile using the *emergency* variable, which, as explained before, is a good candidate for long-term prediction.

As expected, both the *oldest-first* and *local* methods only predict the first four cells the mobile will cross. Using the *information-based* method, the *emergency* variable is kept throughout the MN's journey since it gives information to several cells, increasing its probability to be picked up by the `select` algorithm.

Notice that this approach applied to the problem of paging would yield a method where the most probable cells could be paged first; this result is similar to what could be achieved using the work presented in [1].

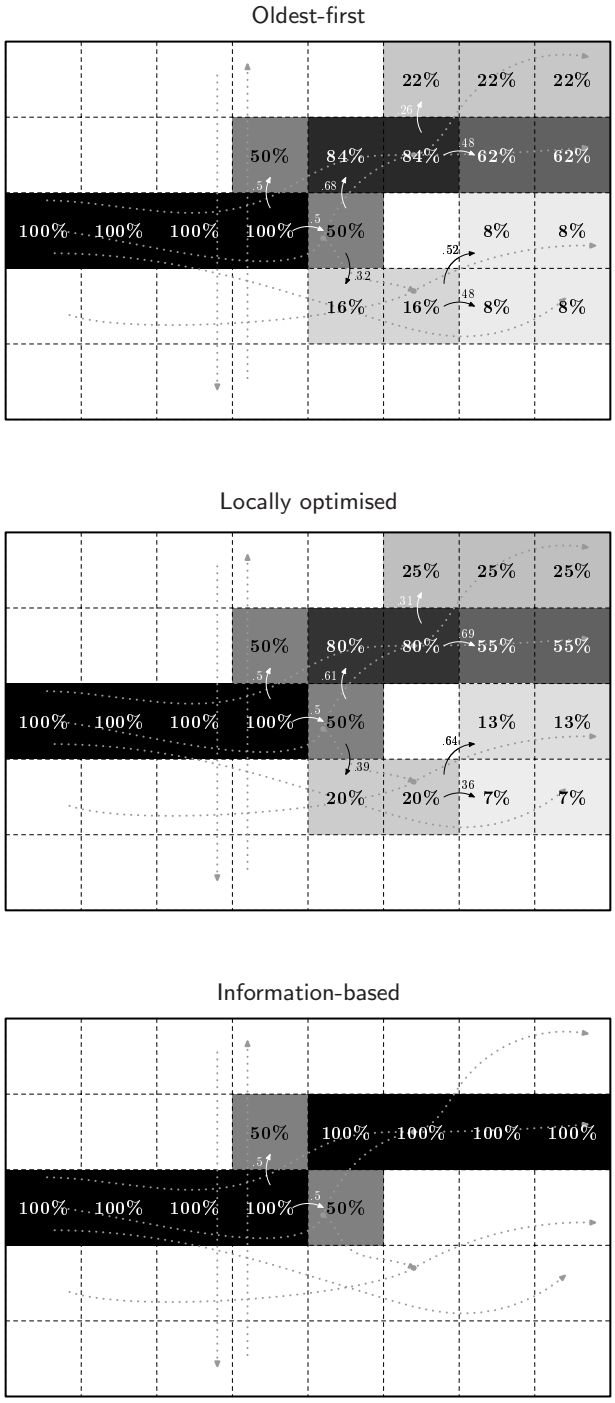

Figure 9: Simulation results of long-term predictions using different information dissemination methods.

7. FUTURE WORK

This work can be extended along several lines:

1. The network has been considered static; this work could be adapted to deal with changing probabilities, where new variables are added and the information associated with a variable can vary with time.

2. This work applies the method to mobility prediction, where information conveyed is used to guess the value of a discrete random variable. It would be interesting to use the method to maximize the information gained

about multiple and/or continuous variables (*e.g.* by means of regression trees).

3. The spreading algorithm could be improved; in particular, it is desirable to compute the *optimum selection scheme* (or a reasonable approximation if the problem is proven NP-complete, see section 4.4.2).

4. The simulations shown here involve a small map and a small number of variables. It would be desirable to show how the method performs with real-scale scenarios and real-world data.

5. The method presented in this paper is also expected to be applicable to other contexts than mobility prediction. Since is well suited to situations where sending information is costly, it could thus be applied to sensor networks where both low available energy and low bandwidth are reasons to transmit as few data as possible. The proposed approach could select the most relevant sensors among all those that have been scattered at a given place.

8. CONCLUSIONS

This paper proposes a method for sending pieces of mobility prediction information where it is most needed. Knowing which clues are useful is done by estimating their expected information on the upcoming path of each mobile.

Simulations have shown that the method allows discriminating relevant clues, sending them where they are needed in the network and, consequently, improving mobility prediction. It has been shown that when a clue is pertinent for a large number of cells, it increases its probability of staying in the network and allows to predict several next cells in advance.

This work is expected to be applicable to other contexts involving constraints such as scarce bandwidth and pieces of information of varied relevance; sensor networks seem good candidates.

Acknowledgements

Thanks are given to Pierre Geurts for his help, explanations and remarks.

9. REFERENCES

[1] A. Bhattacharya and S. Das. LeZi-update: an information-theoretic approach to track mobile users in PCS networks. In *Proc. of ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, Seattle, August 1999.

[2] J.-M. François and G. Leduc. Mobility prediction's influence on QoS in wireless networks: a study on a call admission algorithm. In *Proc. of 3rd International Symposium on Modeling and Optimization in Mobile, Ad-hoc and Wireless Networks (WiOpt)*, Trentino, Italy, April 2005. IEEE Press.

[3] Y. Freund. Boosting a weak learning algorithm by majority. *Inf. Comput.*, 121(2):256–285, 1995.

[4] P. Geurts. *Contributions to decision tree induction: bias/variance tradeoff and time series classification*. PhD thesis, Department of Electrical Engineering and Computer Science, University of Liège, May 2002.

[5] T. Liu, P. Bahl, and I. Chlamtac. Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks. *IEEE JSAC*, 16(6):922–936, August 1998.

[6] A. Misra, A. Roy, and S. Das. An information-theoretic framework for optimal location tracking in multi-system 4G wireless networks. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM)*, pages 286–297, 2004.

[7] V. Pandey, D. Ghosal, and B. Mokherjee. Exploiting user profile to support differentiated services in next-generation wireless networks. *IEEE Network*, pages 40–48, September/October 2004.

[8] R. Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[9] W.-S. Soh and H. S. Kim. Dynamic bandwidth reservation in cellular networks using road topology based mobility predictions. In *Proc. of IEEE Infocom'04*, Hong Kong, Mars 2004.

[10] J. van der Lubbe. *Information theory*. Cambridge University Press, 1997.