Node Position Forecast in MANET with PheroCast

Paulo R. Coelho Enrique Fynn Luis F. Faina Rafael Pasquini Lasaro Camargos
Distributed Computing Group
Computer Science Faculty
Universidade Federal de Uberlândia, Brazil
Email: {paulo,faina,pasquini,lasaro}@facom.ufu.br, enriquefynn@gmail.com

***Abstract*—In mobile ad hoc networks (MANET) nodes are free to move on the environment and interact with each other and with the infra-structure, enabling a multitude of applications and services. However, the same mobility that is key to MANET is also the greater limiting factor in the quality of the services provided in this environment, since the infrastructure must keep adapting to the mobility. This paper describes algorithms for predicting the future position of mobile nodes in MANET, allowing the infrastructure to proactively adapt. Our algorithms maintain the recent movement history of nodes in a compact representation, a graph, based on stigmergy of ant colonies. Predictions are generated through a limited depth first search. We have experimented this method against real world data and the results show that the prediction is accurate for short time horizons (30 seconds) in a metropolitan area (Seattle) divided in a grid of two or three-block square cells, in 77.8% of the cases. This method may be used in optimizing MANET and enabling novel applications, for example, proactive hand-off, tailored content generation, and proactive routing.**

I. Introduction

In a mobile network environment, nodes move about interacting with each other and with the infrastructure that surrounds them, if available. This interaction enables a multitude of services. For example, nodes may rely on each other to reach remotely located services, to propagate alerts about road conditions, to form opportunistic networks that allows communication between nodes that are never directly connected.

To enable such services, as nodes move, the network must adapt to their new physical location, a task that imposes many challenges. For example, in a client-server scenario, if a request is issued, the reply must reach the requester even if it has moved to a different location. While there are several networking protocols that allow nodes to move and still remain connected, such protocols are merely reactive and may not provide the best quality of service possible. If the future position of a node can be accurately estimated, handoffs could be carefully planned so to happen seamlessly and ensure a better quality of service. Moreover, even if the node may always be reached, for example through a 4G network, if data may be forwarded to future positions where more connection options are available, it might be delivered for a lesser cost.

Our main motivation in developing the ability to forecast nodes' positions has to do with optimizing communication in mobile *ad hoc* networks (MANET) via proactive management of software defined networks[1]. More specifically, we are interested in developing novel proactive routing algorithms for vehicular *ad hoc* networks (VANET). However, the very same capability may be used in many different applications, for various ends. For example, it could be used in predicting and optimizing traffic of vehicles in smart cities, in generating tailor made content for soon to arrive clients, or perceiving potential ride sharers for greener transportation.

In this paper we present an approach to predict the future position of a node on a mobile environment. That is, given the node's current position and a time horizon, our approach returns a set of probable positions for the node within the time horizon, along with the probabilities of each position. Our approach assumes that a node's movement history contains good indications of its future decisions or, in other words, we assume that nodes tend to use the same paths, over and over.

Human mobility is amenable to forecasting since it is predictable at several scales, as shown by [2]. Some works have explored such a characteristic in mobile network scenarios. For example, [3] has explored how the history of movements of friends, collected through their smart phones, can be used to improve the forecasting of their movements. Their technique, however, is expensive in that it requires heavy time series analysis and the knowledge of the friends movements; we believe this approach would require a centralized service. In [4], an example more focused on vehicular networks, buses on a well known route communicate with infrastructure elements to inform their position so a central service may determine where to forward messages targeted to the buses. The authors propose to send replicated messages to the bus' last known position plus the following three access points in the route, that is, no real forecasting is made.

Differently from the other works, our forecasting algorithm was developed to run in the mobile node itself, not a centralized service, and to be lightweight in processing. Because the algorithm is inspired by the ant colony abstraction [5], in which the exchange of information happens via the deposition and sensing of pheromones in the environment, we name our approach *PheroCast*, for *phero*mone based fore*cast*.

We have evaluated PheroCast using traces of Seattle Metro Transit's buses, neither particularly collected for our purposes nor of fine time granularity. Even so, we found that our algorithms render good forecasts of buses' positions. For example, if dividing the Seattle area in a grid of 1000ft^2 cells, PheroCast correctly forecasts the cell the buses will be after 30 seconds in 77.8% of the cases.

In summary, in this paper we make the following contributions: i)introduce a method for predicting the future position of

The authors would like to thank PROPP-UFU for financially supporting the presentation and publication of this work.

nodes in mobile environments; ii)present experimental results showing the method's accuracy; iii)discuss several applications of PheroCast.

The remainder of this document is organized as follows. In Section II we describe our data structure and basic algorithm to fill it in. In Section III we describe how make forecasts using the data structure. We discuss some drawbacks in the algorithm and present an improved version in Section IV. Section V is dedicated to evaluating our approach. We present several uses of our algorithms in Section VI and discuss related work in Section VII. Finally, in Section VIII we present some conclusions and discuss future work.

II. PHEROCAST

We start the description of our pheromone-based position forecast algorithms, PheroCast, by introducing a few data structures, starting with *phero trails*.

A. Phero Trail

The ants' or, more generally, the nodes' wanderings are filled with sights of *landmarks*, distinguishable accidents in the environment. We say that a node is in a landmark site L iff L is the most outstanding landmark in sight from the node's current position. Consider a wireless access point (AP), for a real world example. The AP is a landmark and its site corresponds to the surrounding area in which no other AP has a stronger signal. Another example, which we explore later in this paper, might be coarse grained GPS coordinates; if a node has lattitude -18° 53.639' and longitude -48° 10.883', then we may say that it is within site (-18,-48).

The PheroCast algorithms track nodes based on the sites they visit. To do so, at regular time intervals, nodes log the time and the landmark in whose site they currently are; this is equivalent to dropping some pheromone in such a landmark's site, except that the environment is not changed but only the ledger the "ant" carries around. If, on the one hand, the same landmark is seen in consecutive intervals, then either the landmark is visible over a great stretch or the node is moving slowly, and more pheromone will be dropped in its site. On the other hand, if the landmark is just briefly seen, then either the site is small or the node is at high speed, and little pheromone should be dropped, since the node does not stay near the site for very long. Each entry in the log also has the general direction of the node at the moment the entry was created.

Fig. 1 and Fig. 2 depict, respectively, three trajectories of a single node travelling at constant speed through a field with several landmarks and the corresponding trip logs.

To be used in our algorithms, the logs are converted into *phero trails*, digraphs in which vertices represent the sites visited and the edges connect vertices corresponding to sites consecutively visited; the edge points to the vertex corresponding to the site visited latest. Each vertex v in a phero trail holds two values:

- $v.id$, of the form (i, d), is a unique identification of vertex v, where i stands for the landmark identification in the corresponding site and d is the direction the node was following when the respective entry was recorded in the log.
- $v.p$ corresponds to how much pheromone was dropped in the site, while going in the same direction d, i.e., the number of consecutive entries in the log with the same (i, d).

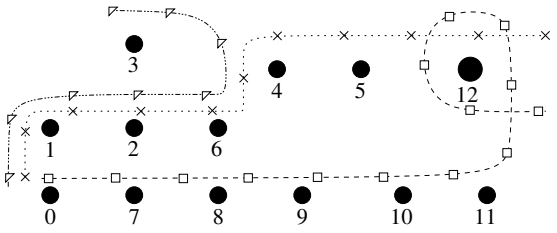


Fig. 1: Example of a node's trajectories on a field, marked as dotted and dashed lines. Large numbered dots identify landmarks and the marks over the lines mark the locations where entries were made in the corresponding trip log, depicted in Fig. 2.

	Trip ×		Trip □		Trip ▽	
Entry #	Site	Direction	Site	Direction	Site	Direction
0	0	N	0	E	0	N
1	1	N	7	E	1	N
2	1	E	8	E	1	E
3	2	E	8	E	2	E
4	6	E	9	E	6	E
5	4	N	10	E	4	N
6	4	E	11	E	3	W
7	5	E	11	N	3	W
8	12	E	12	N		
9	12	E	12	N		
10	12	E	12	W		
11			12	S		
12			12	E		
13			12	E		

Fig. 2: Example of a node's trip log. For each trip, the first column has the site the node was in and the second column has the direction the node was following when the entry was added to the log.

Fig. 3 depicts the phero trails from the example logs in Fig. 2. Phero trails are not necessarily cycle free, however, we do not expect them to frequently deviate from a single chain of vertices, as the one depicted here.

To support the position forecasting, phero trails are combined into *phero maps*.

B. Phero Map

A phero map is a digraph that represents the past trips of a single node or, depending on the application and as discussed in Section VI, multiple nodes. Each vertex v in the phero map holds three values:

- $v.id$, similarly to vertices in a phero trail, is a unique identification of vertex v, in the form (i, d);

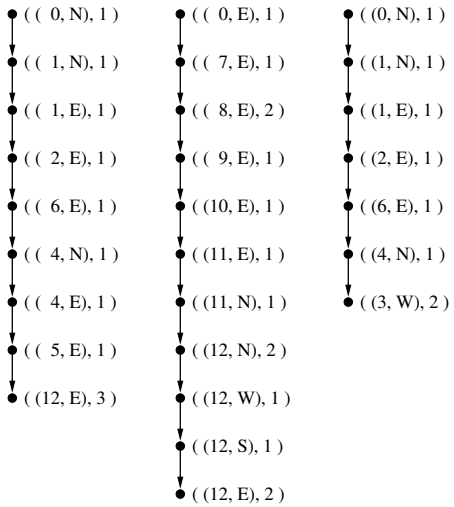


Fig. 3: Phero trails of logs in Fig. 2.

- $v.a$ is the average amount of hormone dropped in each trip in the site i, while going in direction d; and,
- $v.t$ is the number of times site i has been visited, while going in direction d, in all recorded trips.

Given a set of phero trails, the corresponding phero map has a vertex corresponding to each vertex of every phero trail in the set, with the same id. For each such vertex w, the value of $w.a$ is the average of the p value of all vertices with id $w.id$ in all trails in the set, and $w.t$ is the number of such vertices. Formally, given the set $T = \{(V_1, E_1), \ldots, (V_n, E_n)\}$, where each pair (V_i, E_i) is a phero trail with vertices V_i and edges E_i, let $U_T = \bigcup_i^n V_i$; the corresponding phero map $M_T = (V, E)$ is as follows:

- $\exists w \in V \iff \exists v \in U_T, w.id = v.id$
- $\forall w \in V, w.a = Avg(\{v.p : v \in U_T, v.id = w.id\})$
- $\forall w \in V, w.t = |\{v : v \in U_T, v.id = w.id\}|$
- $\exists (e, e') \in E \iff \exists E_i : \exists (f, f') \in E_i : e.id = f.id \wedge e'.id = f'.id$

The pseudo code in Algorithm 1 builds a phero map according to this definition. Fig. 4 depicts the phero map resulting from the trails in Fig. 3.

Using a phero map, one may draw some conclusions about the movement of a node. Based on the map in Fig. 4, for example, we may conclude that the node uses two different routes to travel from site 0 to site 12 and that, in average, it stays within site 8 for twice as much as within site 9. More interestingly, one may predict where the node is headed. For example, from the same map, if at site 6, travelling east, the node may be headed towards sites 3 or 12, but once at site 4, it is likely to be going towards 12 only.

The drawing of accurate conclusions hinges on how accurate the information in the map is, and the map construction, as presented in Algorithm 1, does not deal with one important source of inaccuracy: information ageing. We show how to fix this problem in Section IV, along with discussing some important implementation details. For now, we proceed showing how to make forecasts based on the map.

Algorithm 1 Building a Phero Map.

```
V ⇐ ∅
E ⇐ ∅
for all phero trail (V_t, E_t) ∈ T do
    for all vertex v ∈ V_t do
        if ¬∃w ∈ V : w.id = v.id then
            V ⇐ V ∪ {(id = v.id, a = v.p, t = 1)}
        else
            w.a ⇐ (w.a * w.t + v.p) / (w.t + 1)
            w.t ⇐ w.t + 1
        end if
    end for

    for all edge (f, f') ∈ E_t do
        if ¬∃(e, e') ∈ E : e.id = f.id ∧ e'.id = f'.id then
            E ⇐ E ∪ {(f, f')}
        end if
    end for
end for
```

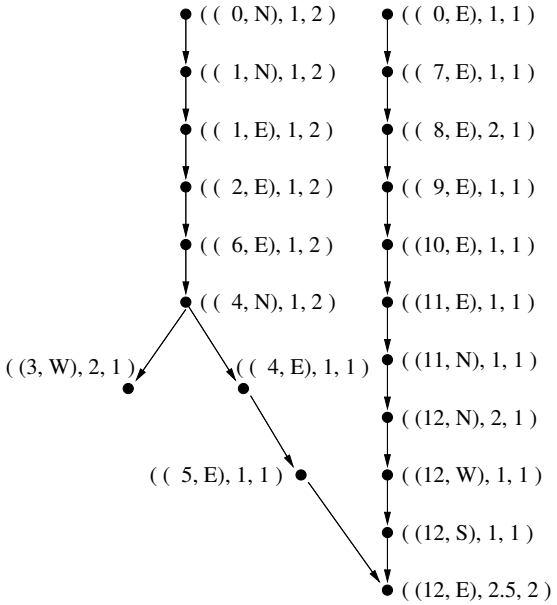


Fig. 4: Phero map of phero trails in Fig. 3.

III. Position Forecast

Given a node's phero map M, a site i, a direction d, the amount of time t the node has been within site i with direction d, and the amount of time h in the future for when the forecast is needed, we need to determine where will the node be after time h has elapsed. We translate this question into the following: what vertices w are reachable from vertex $v, v.id = (i, d)$, such that the sum of the average amount of pheromone dropped in the path from v to w is smaller or equal than h?

Let the sequence of vertices that form a path from vertex v_1 to v_n in M be represented as $\langle v_1, \ldots, v_n \rangle$ and let:

- $A(\langle v_1, \ldots, v_n \rangle) = \sum_{i=1}^{n} v_i.a$;
- P_h be the set of all $s = \langle v_1, \ldots, v_n \rangle$ such that $v_1.id = (i, d)$ and $A(s) \leq h + t$.

The set of vertices reachable from v within time h, which answers the question stated in the previous paragraph, is the set $R = \{v_n : \langle v_1, \ldots, v_n \rangle \in P_h\}$. The pseudo code in Algorithm 2 describes a breadth first search that calculates R.

Algorithm 2 Position Forecast.

Require: i, d, t, h

$R \leftarrow$BST$(i, d, t + h)$

function BST$((i, d), th)$
  Chose $v \in V, v.id = (i, d)$
  if $v.a < th$ **then**
    return $\cup_{w \in V, \exists (v,w) \in E}$ BST$(w.id, 0, th - v.a)$
  else
    return $\{v\}$
  end if
end function

As important as determining R is determining the probability of each vertex v in R be the node corresponding to the next site to be visited, $\mathbf{P}[v]$, which is given by the equation:

$$\forall v, v \in R, \mathbf{P}[v] = \frac{v.t}{\sum_{w \in R} w.t}$$

Once the forecast is generated, it may be post-processed according to the application needs. For example, we envision the forecast being used in a Delay Tolerant Network[6] to decide where to send messages so addressed node will likely collect them. In this scenario, a network router may select the k sites forecast with the highest probabilities to forward messages; forwarding to all forecast sites may give a better coverage, but the cost may be too prohibitive.

IV. Improvements

The phero map construction, as presented in Algorithm 1, is simple and of easy understanding. As a matter of fact, it is too simple, and has some drawbacks that we now discuss.

First and foremost, the resulting phero map does not capture any notion of information ageing. Hence, old and, possibly, stale information affect the predictions just as much as new and up-to-date information. If the moving pattern changes over time, it is desirable that the older a trip, the lesser its impact in the forecasting.

Second, the algorithm requires an *a priori* knowledge of all trails to be processed. This requirement is not realistic for two reasons: mobile nodes may not have the storage required to keep all the phero trails plus the phero map; moreover, to be useful the mechanism should generate predictions from day one, which will be more or less accurate depending on the information already summarized by the map.

Third, the map does not preserve the notion that different origins may normally lead to different destinations, even if the routes share some sites. For example, suppose that a node has two route patterns, one from site 1 to 3 passing by 2 and one from 4 to 5 also passing by 2 *en route*. When forecasting the node's position while at site 2, the proposed map will not provide any means to differentiate trips started at 1 from those started at 3.

We have addressed the first and second problems in Algorithm 3, which we discuss next, along with some implementation details. As for the third problem, it may be tackled by means of other techniques, which we discuss in the Section VII. Such techniques, however more accurate, are also more expensive to execute.

Algorithm 3 Building a Phero Map – Improved version.

$V \Leftarrow \emptyset$
$E \Leftarrow \emptyset$
for all phero trail $(V_t, E_t) \in T$, with timestamp τ **do**
  for all $vertex v \in V_t$ **do**
    if $\neg \exists w \in V : w.id = v.id$ **then**
      $V \Leftarrow V \cup \{(id = v.id, a = v.p, s = \tau, t = \sigma)\}$
    else
      $w.t \Leftarrow \sigma + (1 - \sigma)^{\tau - w.s} * w.t$
      $w.a \Leftarrow w.a + \sigma(v.p - w.a)$
      $w.s \Leftarrow \tau$
    end if
  end for

  for all edge $(f, f') \in E_t$ **do**
    if $\neg \exists (e, e') \in E : e.id = f.id \wedge e'.id = f'.id$ **then**
      $E \Leftarrow E \cup \{(f, f')\}$
    end if
  end for
end for

A. Information Ageing

As aforementioned, the a field of a vertex in the phero map stores the average amount of time spent in the corresponding site; this value is used in Algorithm 2 to limit the depth of the search over the map. In order to account for information ageing, we have replaced the standard average used to calculate a in Algorithm 1 by an exponentially weighted moving average, EWMA[7], also known as exponential smoothing.

In short, the EWMA is an iteratively built average that gives older values exponentially smaller weights. Given a set of $t - 1$ values and its EWMA S_{t-1}, the EWMA of t values $S_t = \sigma v_t + (1 - \sigma) S_{t-1} = S_{t-1} + \sigma(v_t - S_{t-1})$, where v_t is the t-esim value added. The bigger the σ factor, $0 \leq \sigma \leq 1$, the smaller the weight of older values and the faster the ageing. The estimation of round trip times in TCP is one example of other uses of EWMA in networks [8].

Once the set of possible paths is determined, the t field is used to weight them against each other to calculate the probability of using each one. Although this field is not an average, we have made it so without change in the forecasting algorithm, and also used EWMA to account for ageing. Replacing the calculation of field a for EWMA is straightforward

since the field was an average in the first algorithm. Applying EWMA to field t required other changes.

First, to keep track of how old the information is, we tag each trail with a *timestamp*. The timestamp is an all time monotonically increasing counter of the number of trails a node has processed; the first trail has timestamp 1, the second 2, and so forth. Second, we extended each vertex in the map with field s, which stores the timestamp of the trail that last caused the vertex to be updated. Whenever processing a new trail, causing a vertex w to be updated, the following procedure is used: the old value of $w.t$ is multiplied by $1-\sigma$ as many times as there were trails in between $w.s$ and the one being processed, and summed to σ. This way, the value of $w.t$ corresponds to the EWMA of values w_t, where w_t is 1 if the vertex was visited in trail t and 0 otherwise. The resulting value is used in the same way as in the search, that is, to weight possible paths against each other.

B. Execution Optimization

To allow for an online phero map construction, the outer **for all** in Algorithm 3 may be executed interactively, as new trails are added to T. Moreover, the first inner **for all** in the algorithm may also be executed interactively. This way, the algorithm does not have to wait until the new trail is complete and, instead, may process new vertices as they are added to the trail.

To keep a small footprint, a garbage collection procedure runs in parallel with the construction, removing nodes with timestamps that are too old from the map. The process is optimized by locating garbage collection candidates during map (forecasting) searches. The garbage collection is not described in the pseudo code to keep the presentation simple.

As a final optimization, the different powers of $1-\sigma$, needed in the EWMA calculations, are pre-computed and stored in a table for quick reference; values smaller than a predefined threshold are considered zero, effectively discarding information that is too old.

V. Evaluation

In order to test the effectiveness of PheroCast, we have evaluated it using traces of Seattle buses collected in November 2001[9]. The traces indicate the position of uniquely identified buses, the route they were following, and the time the record was made; Fig. 5 shows an excerpt of the logs.

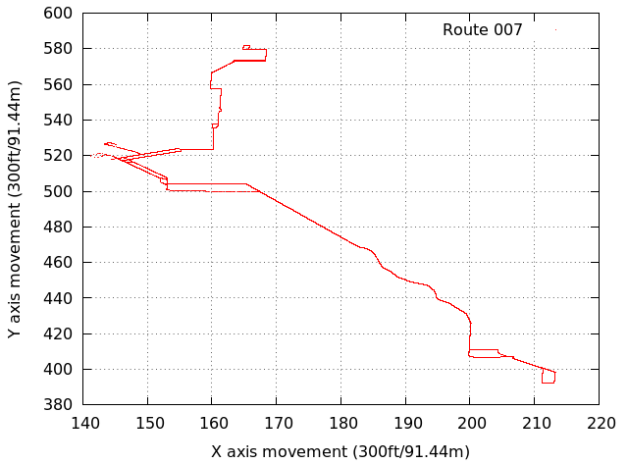
```
20-11:20:50:06 1008 007 00700572 53313.750000 143677.765625
20-11:20:50:37 1008 007 00700572 52934.000000 144449.687500
20-11:20:51:06 1008 007 00700572 52518.625000 145292.718750
20-11:20:51:37 1008 007 00700572 52136.875000 146063.234375
20-11:20:52:05 1008 007 00700572 51806.125000 146747.406250
20-11:20:52:37 1008 007 00700572 51289.625000 147785.031250
```

Fig. 5: Trace excerpt of bus 1008 following route 007. Columns are date:time, bus id, route id, unknown, and distances in X and Y from a reference point, in feet.

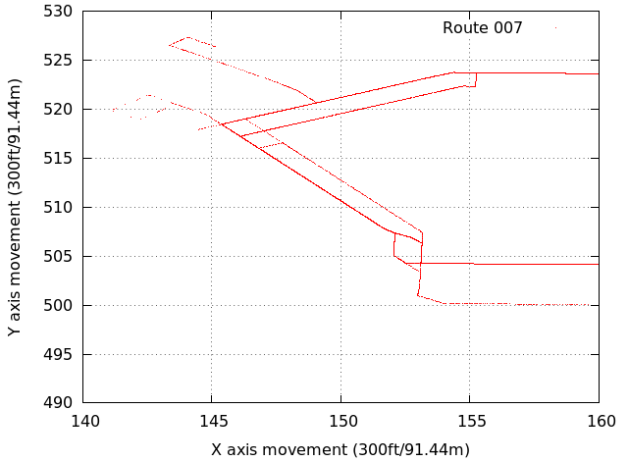
A. Parameters

During the experiment, the log is processed as if each entry had just been produced. Although there is no data about the trace informing how often the entries should have been recorded, we identified that most entries are around thirty seconds apart. To handle variations and small gaps in the log, we considered any interval of less than $m \times 30$ seconds as *one time unit*, where m is a test parameter, and disregarded larger intervals. That is, if two log entries were more than one time unit apart, then they do not result in an edge in the phero map. For each entry processed, we used PheroCast to predict the position in the next entry, if not disregarded. E.g., when processing the second entry in Fig. 5, we estimate where the bus would be after one time unit, around 20:51:07 and check weather it matches the coordinates in the third entry in the log.

Because the buses' coordinates in the log are too fine grain for our purposes (1 foot), for the tests we considered coarser units of g feet. That is, we have the equivalent of a grid of $g\text{ft}^2$ square cells overlaid on Seattle. In Fig. 6, for example, we considered $g = 300\text{ft}$.



(a)



(b)

Fig. 6: (a)All entries for buses on route 007 on the fifth trace and a (b) zoom. The entries show a clear movement pattern, as expected from a regular bus line.

In summary, the results presented are dependent on four

parameters: k, the maximum number of entries from the forecast which are considered; g, the dimensions of a cell over the map; m, the threshold for considering consecutive log entries; and σ, the smoothing parameter for EWMA.

B. Validity of Assumptions

The results we present here are for the 186 trips over route 007, which ammount to 310,206 entries in the log. We focus on this route because it is the largest in the trace set; results for other routes and traces are similar. The prototype used is available at https://gitorious.org/ants-colony-grid/aging-fixes, along with an explanation of how the traces were pre-processed.

From the logs and Fig. 6(a) it is clear that the assumption we made that the mobile node follows clear patterns is satisfied, as expected by buses following well defined routes. A zoom of Fig. 6(a) is provided in Fig. 6(b).

C. Error

In order to measure the error of our algorithms, we considered, among the k selected forecast, the one closest to the actual node position. The error is the difference, in number of cells, between the real and the forecast coordinate.

For an example, consider the scenario with with parameters $k = 3$, $g = 600$, $m = 2$ and $\sigma = 0.4$. For such scenario, Fig. 7(a) shows the x coordinate of few thousand entries in the traces, as well as the best forecast. The error is presented in the bottom part of Fig. 7(a). A similar graph, for the y coordinates and the combination of X and Y coordinates is presented in Fig. 7(b) and (c), respectively.

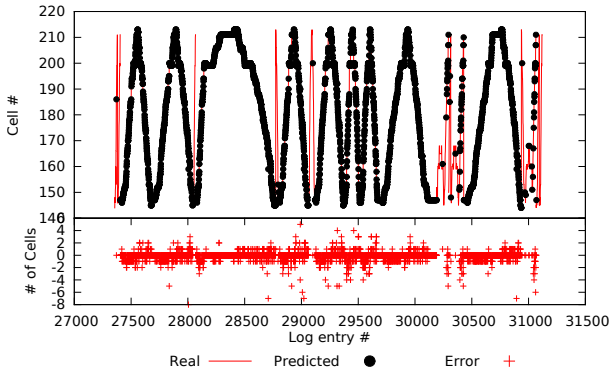
D. Results

We considered several combinations of values for k, g and σ, which are compiled in Table I. At the first (top) part of the table we present the results for varying g, and fixed k, m and σ. Although the value of $k = 3$ seems arbitrary, it was chosen to reflect the possible directions a bus may follow from a block corner. Since g value of 300,600, or $1000ft$ roughly correspond to one, two, and three blocks, $k = 3$ should be well suited in these cases. We use the $g = 1000ft$ configuration as baseline in the middle and bottom parts of the table.

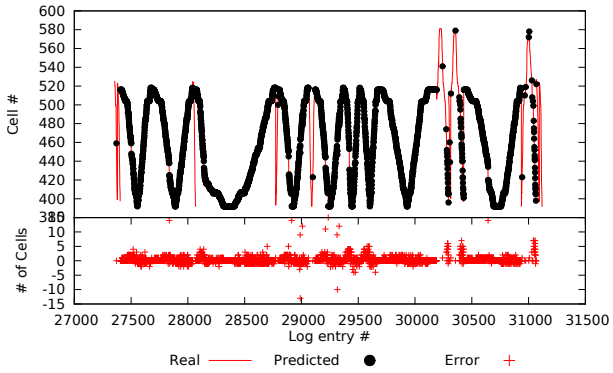
The data shows that the larger the parameter g the more forecasts are correct. However, a number of forecasts are still off, on average, by around 0.320892 cell, even when each cell is $1000ft$ wide. We believe this happens due to the temporal granularity of the log; thirty seconds may represent a wide variety of displacements, depending on the time of the day, traffic, and events such as load and unload of wheel chairs, which will lead to each node in the phero map having a large degree. This suspicion is somewhat confirmed by the data on the second part of Table I where we varied the parameter k. The larger k is, the more alternatives in the forecast are taken into account and, therefore, the smaller is the number of errors.

Parameter m plays an important role in the algorithm, allowing bad data to be ignored and causing the algorithm not to risk a forecast. This fact is seen on the third part of the table; with $m = 1$, the algorithm correctly forecasts half of what it would with $m = 2$, but is makes only one fourth of the mistakes, reaching an accuracy of 77.8%. Depending on the application, PheroCast may be expected to risk and, therefore, m should be made large. We expect, however, that if it is combined with large g, the rate of success will be very low, as the phero map should become filled with nodes for the same route but not connected.

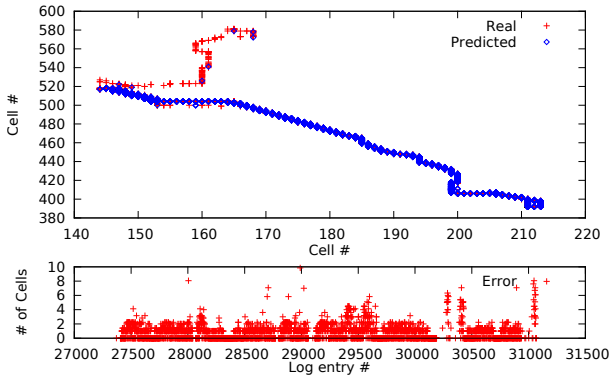
Finally, the bottom part of the table shows the effect of varying σ. Because the buses routes are very stable, the value of σ had little influence in the algorithm's outcome. Further testing is required to better highlight the effects of varying σ.



(a) X coordinate forecast and error.



(b) Y coordinate forecast and error.



(c) X and Y coordinate forecast and error.

Fig. 7: Estimated X/Y coordinate and error for bus following route 007 with parameters $k = 3$, $g = 600$, $m = 2$ and $\sigma = 0.4$. Error is measured as the difference, in number of cells, between real and forecast coordinates.

	$k=3, g=1000, m=2, \sigma=0.4$	$k=3, g=600, m=2, \sigma=0.4$	$k=3, g=300, m=2, \sigma=0.4$
Correct	164144	142134	113187
Incorrect	78733	100662	129160
Avg Error	0.380292	0.568888	1.04426
Std Deviation	0.628262	0.87885	1.59737

	$k=3, g=1000, m=2, \sigma=0.4$	$k=1, g=1000, m=2, \sigma=0.4$	$k=5, g=1000, m=2, \sigma=0.4$
Correct	164144	148272	170055
Incorrect	78733	94605	72822
Avg Error	0.380292	0.463901	0.352533
Std Deviation	0.628262	0.669402	0.612979

	$k=3, g=1000, m=2, \sigma=0.4$	$k=3, g=1000, m=1, \sigma=0.4$	$k=3, g=1000, m=3, \sigma=0.4$
Correct	164144	80131	178908
Incorrect	78733	22816	105657
Avg Error	0.380292	0.239235	0.445941
Std Deviation	0.628262	0.485933	0.6708

	$k=3, g=1000, m=2, \sigma=0.4$	$k=3, g=1000, m=2, \sigma=0.9$	$k=3, g=1000, m=2, \sigma=0.1$
Correct	164144	162990	164099
Incorrect	78733	80287	78772
Avg Error	0.380292	0.383664	0.380648
Std Deviation	0.628262	0.623157	0.631531

TABLE I: Correct/incorrect forecasts for different configurations of g, k, m and σ. The scenario $k=3, g=1000, m=2$ and $\sigma=0.4$ is used as baseline.

Even though all configurations lead to errors in the forecasting, their distribution, presented in Fig 8, shows that most are around one or two cells, and very few are above five cells. In a delay tolerant network, for example, if the error is for a coordinate that will still be reached, even if not within the expected time horizon, the network's ability to delivery of messages may not be affected.

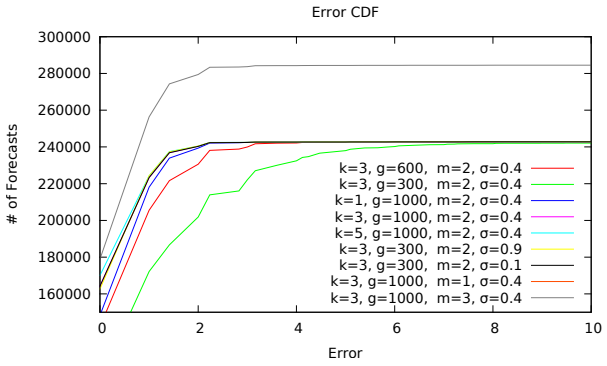


Fig. 8: Cumulative Distribution Functions for errors in forecasting with diverse configurations of g, k, m and σ.

VI. Variations and Applications

There are many different scenarios in which PheroCast or some variation thereof may prove useful. Here we discuss some examples.

a) GPS Pre-Calculated Routes: If PheroCast is embedded in a vehicular on-board system equipped with GPS, then whenever the GPS is used to calculate a route, it may be used by PheroCast to improve the forecasts. Based on our own experience, which we conjecture is valid to other drivers, whenever driving on a well known area, the GPS suggestions matters very little. On lesser known areas, however, the GPS suggestion will likely be followed. Hence, PheroCast may consider the GPS suggestion as a regular phero-trail, which will bear little weight on the well known areas but be very important in complementing the little information in the map for the less known areas.

b) Network Routing: As aforementioned, we are interested in using PheroCast in optimizing the routing in vehicular *ad hoc* networks – VANET. Once a node knows where it will be going, it can attach such information to outgoing requests so that it is used by the network to route responses back to the node. For example, the forecast may be used to create flows in OpenFlow[10] or reserving channels in the MPLS[11] boxes it crosses.

c) Smart Cities' Traffic Planning: In a smart city scenario, nodes could periodically report their position and forecast to some common entity. This way, the entity could then correlate the information to estimate and forecast traffic. This would allow, for example, for detecting and preventing traffic jams. Moreover, if nodes are taking unusual routes, other nodes, with similar mobility patterns, may be advised to take detours before they reach the area with anomaly.

The same mechanism may be used while in areas not previously (or recently) visited by the node. That is, the forecasts may be accomplished based on the average mobility pattern in the area.

VII. Related Work

[2] have shown that human mobility is predictable at several scales. We have explored such characteristic and presented, in this paper, a method for forecasting the position of nodes in a mobile network environment. In this section we discuss other works related to forecasting in mobile networks.

In [3], time series analysis of user mobility traces have been combined with social information from the users' smart phones to provide very accurate forecasts. It seems reasonable

that such approach must be realized in a logically centralized way, where data from "friends" is aggregated and correlated. Differently, the approach we introduce here only considers single user's traces and may be executed unilaterally by each node in a mobile network. Our approach is very light if compared to the time series analysis in [3] and well fit for processing and power constrained devices.

More focused on vehicular networks, the work in [4] has focused on communication with buses on a well known route. In the work, buses update a central service with their current position whenever they handshake with an infrastructure device and, whenever a message has to be sent to the bus, the authors propose to send replicated messages to the bus' last know position plus the following three access points in the route. Differently from our work, no forecasting is made in [4].

We envision that forecasting the position of nodes will allow for new routing mechanisms in VANET, allowing messages to be targeted to where the node will be instead of where it is. [12] has explored the same problem, using GPS trajectories in the forecasting. We believe that most GPS owners will seldom plan their trips within areas they know well using the device. In such areas, PheroCast will use the user's history to forecast; if a GPS route is available, then it may be also taken into account, but this is not a requirement.

PheroCast has been inspired by ant colonies and stigmergy, the ability to indirectly communicate by modifying elements in the environment [13]. While ants use pheromone to communicate with each other, PheroCast enabled nodes "leave" marks mostly for themselves, and therefore may be seen as independent one-individual colonies. In Section VI we discussed some of the interesting applications that emerge once the forecasts are combined.

VIII. Conclusion and Future Work

In this paper we have presented an algorithm for forecasting location of nodes in mobile networks. The algorithm is based on a compact representation of nodes' recent mobility history and inspired by the ant colony abstraction. We have also described some experiments and shown that the algorithm performs well in forecasting the location of buses in a metropolitan area.

Our strongest motivation for developing PheroCast lies in developing novel routing mechanisms for mobile *ad hoc* networks, in special vehicular *ad hoc* networks, VANET. Specifically, we are interested in proactively adapting the network infrastructure using Software Defined Networks[1] and enabling Delay Tolerant Networks [6] with better QoS.

Currently we are working on several improvements to PheroCast. For example, we are extending the algorithm evaluation in emulated mobility scenarios using SUMO[14]. We expect that in such a controlled scenario we will be able to better measure the effects of the aging strategy adopted. We are also developing an application to collect mobility traces from smartphone users. With such traces we will be able to test the forecasting also in a new set of realistic scenarios, ranging from users in closed locations such as malls, to drivers travelling within and between cities.

As next steps we plan on improving the algorithms by including standard deviation statistics in our phero map data structure, which would allow searches not for a simple set of sites that might be reached, but for a set of ranges over the possible routes. For example, one will be able to search for all locations reachable with 10 seconds, give or take one standard deviation from the expected location. We also plan on investigating how to take into account the temporal context in which the trips happen (e.g., day of the week and time of the day) as well as other context information. In this sense, we will investigate the use of more complex time series analysis to determine if they may render better results than our current approach while being lightweight enough to run in a car on-board system or a smart phone.

References

[1] N. McKeown, "Software-defined networking," *INFOCOM keynote talk, Apr*, 2009.

[2] D. Brockmann, L. Hufnagel, and T. Geisel, "The scaling laws of human travel," *Nature*, vol. 439, no. 7075, pp. 462–465, 2006.

[3] M. De Domenico, A. Lima, and M. Musolesi, "Interdependence and predictability of human mobility and social interactions," *arXiv preprint arXiv:1210.2376*, 2012.

[4] V. B. da Silva, F. O. da Silva, M. E. M. Campista, and L. H. M. Costa, "A trajectory-based approach to improve delivery in drive-thru internet scenarios," in *Workshop on Emerging Vehicular Networks: V2V/V2I and Railroad Communications, IEEE International Conference on Communications ICC 2013*, 2013.

[5] M. Dorigo, "Optimization, learning and natural algorithms," Ph.D. dissertation, Politecnico di Milano, Italy, 1992.

[6] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2003, pp. 27–34.

[7] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *International Journal of Forecasting*, vol. 20, no. 1, pp. 5 – 10, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169207003001134

[8] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," RFC 5348 (Proposed Standard), Internet Engineering Task Force, September 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5348.txt

[9] J. G. Jetcheva, Y.-C. Hu, S. PalChaudhuri, A. K. Saha, and D. B. Johnson, "CRAWDAD data set rice/ad_hoc_city (v. 2003-09-11)," Downloaded from http://crawdad.cs.dartmouth.edu/rice/ad_hoc_city, Sep. 2003.

[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[11] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031 (Proposed Standard), Internet Engineering Task Force, January 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3031.txt

[12] J. Jeong, S. Guo, Y. Gu, T. He, and D. H. Du, "Tsf: Trajectory-based statistical forwarding for infrastructure-to-vehicle data delivery in vehicular networks," in *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*. IEEE, 2010, pp. 557–566.

[13] M. Dorigo, E. Bonabeau, and G. Theraulaz, "Ant algorithms and stigmergy," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 851 – 871, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X0000042X

[14] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo - simulation of urban mobility: An overview," in *SIMUL 2011, The Third International Conference on Advances in System Simulation*, Barcelona, Spain, 2011.