Towards Mathematical Programming Methods for Predicting User Mobility in Mobile Networks

Alberto Ceselli and Marco Premoli

Abstract Motivated by optimal orchestration of virtual machines in mobile cloud computing environments to support mobile users, we face the problem of retrieving user trajectories in urban areas, when only aggregate information on user connections and trajectory length distribution is given. We model such a problem as that of finding a suitable set of paths-over-time on a time-dependent graph, proposing extended mathematical programming formulations and column generation algorithms. We experiment on both real-world and synthetic datasets. Our approach proves to be accurate enough to faithfully estimate mobility on the synthetic datasets, and efficient enough to tackle real world instances.

1 Problem Statement and Modeling

Motivated by optimal orchestration of virtual machines in mobile cloud computing environments to support mobile users [1], we face the problem of retrieving user trajectories in urban areas. We partition the region covered by a mobile network into cells, one for each Access Point (AP), and we suppose to be given: (a) the adjacency matrix between cells, and (b) the demand in each cell at each point in time, that is the number of users connected to the corresponding AP. We also assume that an aggregated information about user mobility is given, namely the probability distribution of trajectory lengths. Our aim is to find an estimate on the trajectory, and more in general on the path of each user, in terms of sequence of cells traversed by the user during the considered time horizon. Since demand is usually easy to forecast, e.g. by time series analysis, our methods can be seen in the long term as means of *predicting* the corresponding user mobility. Our modeling approach (Sect. 1) is the following: first, we perform a time discretization and a trajectory length categorization. Then, we

A. Ceselli (✉) · M. Premoli
Dipartimento di Informatica, Università Degli Studi di Milano, via Bramante 65, Crema, Italy
e-mail: alberto.ceselli@unimi.it

M. Premoli
e-mail: marco.premoli@unimi.it

A. Fink et al. (eds.), *Operations Research Proceedings 2016*,
Operations Research Proceedings, DOI 10.1007/978-3-319-55702-1_7

introduce extended mathematical programming formulations, inspired by flows over time models, having a polynomial number of constraints, but an exponential number of variables, and two hierarchical objectives. We devise column generation algorithms (Sect. 2): pricing problems are resource constrained minimum cost path problems, for which we provide ad-hoc dynamic programming procedures. We experiment on both synthetic datasets, obtained through generative models from the literature, and real world datasets from a major mobile carrier in Paris for which ground truth is not available (Sect. 3). Our approach proves to be accurate enough to faithfully estimate mobility on the synthetic datasets, and efficient enough to tackle real world instances. Our model is the following.

Data Let $T = \{1, \dots, |T|\}$ be a set of time slices and N be a set of APs, each lying at coordinates (x_i, y_i) in a plane that models our urban area. For each $t \in T$ and $i \in N$, let $d_i^t \in \mathbb{Z}^*$ be the number of users connected to AP i during time slice t. We denote as Ω the set of feasible *paths-over-time* (paths in the remainder), each being a sequence of APs whose cells are adjacent, and which are assumed to be visited by users in consecutive time slices. Notation-wise, for each $p \in \Omega$, we indicate with $p(t)$ the AP visited at time t in path p, and we suppose $p(t)$ to be set to a dummy value "–" if path p starts after, or ends before t. Let $l(p)$ be the total length of each path $p \in \Omega$, that is the sum of euclidean distances between consecutive APs in the path. The starting and ending APs of each path (the first and last values of $p(t)$ which are different from "–") identify a trajectory; the same trajectory can be identified by many feasible paths. Let $K = \{1, \dots, |K|\}$ be a set of classes, obtained by partitioning Ω according to the length of its paths. For each $k \in K$, let l_k (resp. l_{k-1}) be the upper (resp. lower) bound on the length of each path in class k, with $l_0 = 0$; let also $n_k \in \mathbb{Z}^*$ be the number of users whose path is in class k. From an application point of view, we assume (x_i, y_i) and d_i^t to be given, e.g. by a telecommunication operator, Ω to be easily definable, e.g. by Voronoi tessellations and street maps, and l_k and n_k to be estimated by previous knowledge on users travel distance distributions like [2].

Variables Our aim is to assess how many users are expected to follow a path over our time horizon, that we indicate as x_p for each $p \in \Omega$. We also consider the possibility that users enter or quit the system, or that simply data d_i^t is approximate, allowing a positive (resp. negative) correction $\bar{\varepsilon}_i^t$ (resp. $\underline{\varepsilon}_i^t$) for each $i \in N, t \in T$.

Constraints A feasible solution respects the following constraints:

$$d_i^t - d_i^{t-1} = \sum_{j \in N} \sum_{\substack{p \in \Omega \\ |p(t-1)=j \\ \wedge p(t)=i}} x_p - \sum_{j \in N} \sum_{\substack{p \in \Omega \\ |p(t)=j \\ \wedge p(t-1)=i}} x_p + \bar{\varepsilon}_i^t - \underline{\varepsilon}_i^t \qquad \begin{matrix} \forall i \in N, \\ \forall t \in T, t>1 \end{matrix} \qquad (1)$$

$$\sum_{\substack{p \in \Omega \\ |l(p)<l_k}} x_p \geq \sum_{k' \leq k} n_{k'} \qquad \forall k \in K \qquad (2)$$

$$x_p \geq 0, \bar{\varepsilon}_i^t \geq 0, \ \underline{\varepsilon}_i^t \geq 0 \qquad (3)$$

Constraints (1) resemble flow conservation, imposing the expected variation at time t in the number of users connected to AP $i \in N$ at time t to be consistent with the number of users arriving in i and those leaving i, potentially with corrections given by $\bar{\varepsilon}_i^t$ and $\underline{\varepsilon}_i^t$. We experimented on variants of (1), including a pure flow conservation formulation, without improvements. Constraints (2) imply that the number of users following a path in class k is at least the estimated one: cumulative values are used.

Objective We adopt a hierarchical bi-objective approach. Our primary objective is to find a setting of the variables explaining our data with minimum absolute value correction, that is we optimize the following linear program (LP):

$$\min\ \varepsilon = \sum_{t \in T} \sum_{i \in N} (\bar{\varepsilon}_i^t + \underline{\varepsilon}_i^t) \quad \text{s.t.} \quad (1), (2), (3)$$

Once an optimal ε value is found, as a secondary objective we try to match the path lengths distribution as close as possible; that is, we minimize the maximum difference between the number of users migrating on paths of each class k according to our solution, and the estimated one:

$$\begin{aligned} &\min\ \eta && (4)\\ &\text{s.t.} \sum_{\substack{p \in \Omega \\ |l(p) \in [l_{k-1}, l_k)}} x_p - n_k \leq \eta,\ \forall k \in K && (5)\\ &\sum_{i \in N} \sum_{t \in T} \bar{\varepsilon}_i^t + \underline{\varepsilon}_i^t \leq \varepsilon && (6)\\ &(1), (2), (3) \end{aligned}$$

2 Algorithms

Both problems are LPs. However, as the cardinality of Ω grows combinatorially, it is computationally infeasible to solve them directly. Instead we perform column generation on the set of variables x_p.

For the primary objective problem, let λ_i^t and μ_k be the dual variables associated to constraints (1) and (2), resp. The reduced cost of a variable x_p is

$$\bar{c}_p = - \sum_{t \in T\ |p(t) \neq '' - ''} (\lambda^t_{p(t-1)} - \lambda^t_{p(t)}) - \sum_{k \in K\ |l_k > l(p)} \mu_k.$$

For each $k \in K$, the search for the most negative reduced cost variable encoding a path in class k can be mapped into the problem of finding a minimum cost path in a time-expanded directed graph G = {N', A}, that has one node (i, t) for each pair of AP $i \in N$ and time slice $t \in T$, together with two additional dummy nodes acting as origin and destination; i.e. $N' = (N \times T) \cup \{(o, t_{-1}), (d, t_{T+1})\}$. The set A includes one arc $(i, t-1; j, t)$ connecting nodes $(i, t-1)$ and (j, t) if and only if the cells of

APs i and j are adjacent. The dummy origin (resp. destination) has an outgoing (resp. incoming) arc to (resp. from) every other node. Each arc $(i, t-1; j, t)$ has cost $w_{i,j}^{t-1,t} = \lambda_j^t - \lambda_i^t$ and length $l_{i,j}^{t-1,t} = \|(x_i, y_i) - (x_j, y_j)\|$, except those incident to either the origin $(o, 0)$ or the destination $(d, T+1)$, whose cost and length are set to 0. Indeed, the graph nodes are organized in layers, one for each time slice; paths in G can only be composed by nodes of different layers, and by arcs connecting one layer with the subsequent one. Modeling of waiting decisions is included, as represented by arcs $(i, t-1; i, t)$.

Not all paths are considered feasible for each class k, but only those starting from $(o, 0)$ and ending in $(d, T+1)$ whose sum of arc lengths falls into the range $[l_{k-1}, l_k)$. In principle, performing column generation means to solve a *resource constrained* minimum cost path problem for each $k \in K$. However, we propose an ad hoc dynamic programming algorithm, that optimize over all classes simultaneously, working as follows. We consider *labels* of the form $(C, L, (i, t))$, encoding partial paths starting from $(o, 0)$, ending in (i, t), whose sum of arc prizes and lengths are C and L resp. We *initialize* the algorithm, creating a single starting label $(-\sum_{k \in K} \mu_k, 0, (i, t))$ for each $i \in N, t \in T$; then, we proceed layer by layer and node by node, that is, for each $t \in T$ and for each $i \in N$, we iteratively *select* each label $(C, L, (i, t))$ and *extend* it to all the nodes $(j, t+1)$ having $(i, t; j, t+1) \in A$, creating a new label $(C', L', (j, t+1))$ for each of them that has $L' = L + l_{i,j}^{t,t+1}$ and

$$C' = C + w_{i,j}^{t,t'} + \sum_{k \in K \,|\, L < l_k \wedge L' \geq l_k} \mu_k.$$

The creation of labels having $L' \geq l_{|K|}$ is skipped, as encoding infeasible paths. After treating each label we *check dominance* rules: if any label $(C'', L'', (j, t+1))$ has already been created, having $C'' \leq C'$ and $L'' \leq L'$, at least one inequality being strict, then $(C', L', (j, t+1))$ is fathomed; similarly, if $C'' \geq C'$ and $L'' \geq L'$, at least one inequality being strict, then $(C'', L'', (j, t+1))$ is fathomed. We stop when all pairs (i, t) have been considered. All labels whose cost C is negative encode paths of negative reduced cost. We remark that, given the laminar structure of constraints (2), this aggregated dynamic programming algorithm is able to produce in a single run the labels of all non dominated paths for each class k; a formal proof is omitted. This allows us on one side to improve efficiency, since only one resource constrained minimum cost path problem needs to be solved at each column generation iteration, and on the other side to obtain an effective multiple pricing strategy, that consists in enlarging the set $\bar{\Omega}$ at each column generation iteration with the minimum reduced cost path for each class $k \in K$, if any of negative reduced cost exists.

The same algorithm is used for the secondary objective problem (1)–(6). Formally, since the structure of constraints (5) is not laminar anymore, the dominance rules need to be slightly relaxed to take into account of the contribution of the new dual variables. In our implementation, instead, we found it computationally useful to keep the original rules and resort to heuristic pricing. As discussed in Sect. 3 the routine obtained in this way proved to be able to produce high quality solutions with limited effort.

3 Dataset Generation and Experiments

Unfortunately, no ground truth is available on our real-world dataset. Therefore, in order to test both the computational viability and the prediction accuracy of our methods, we proceed as follows. First, we draw APs coordinates at random, and we generate instances as collections of user paths-over-time on this set of APs; we refer to such a collection as the *original* paths. Then the number of users d_i^t connected to each AP $i \in N$ at time $t \in T$, and the details l_k and n_k of path length classes $k \in K$, are computed and used as sole input of our methods. Therefore the full collection of *original* paths is kept only for cross-checking (as post-processing) the quality of *predicted* paths, that are those produced as output solutions of our methods.

We propose two generative models of original paths. The first is a simple *ad-hoc* model: given the number of users U as input, for each of them we create a path whose length is drawn from a power law distribution, and whose starting time is chosen uniformly at random. We assume that one hop is made in each time slice, in a graph having one node for each AP, and one edge between each pair of APs whose Voronoi cells are adjacent. The second is a *Point of Interest* (*POI*) generative model, reproducing the behavior of users during rush hours [2]: we randomly define a set $S \subseteq N$ of residential points and a set $D \subseteq N$ of destination POIs. We randomly draw the starting (resp. final) position of each user from bivariate normal distributions centered in a user residential point of S (resp. POI of D). Attractiveness of APs and transition probabilities are built following [2]. One path is finally generated for each user, choosing a residential point uniformly at random, a destination AP at random according to the transition probabilities, computing the shortest path in the adjacency graph described previously, assuming one hop for each time slice.

Our algorithms are implemented in C++ using CPLEX 12.6 as LP solver; the tests are performed on a PC with i7 4.0 GHz CPU and 32 GB RAM. For experiments we use a synthetic set of 300 APs with coordinates randomly drawn from a single bivariate normal distribution, and considering 15 time slices. These values match well those of real applications [1]. Given this fixed set of APs, we create 5 instances with $U = 40000$ for each generative model. Given the lengths of all paths in each instance, we compute 100 path length classes, with l_k values given by the percentiles of lengths distribution. We first assess the computational viability of our methods. Table 1 reports, for each stage of our algorithm (column blocks) and for each generative model (table rows), the avg. number of column generation iterations,

Table 1 Computational efficiency

Gnr. model	1st stage				2nd stage				Total t.
	CG iter	Master t.	Pricer t.	n. paths	CG iter	Master t.	Pricer t.	n. paths	
Ad-hoc	81.0	0.66	1.27	59.08	32.8	8.61	2.35	68.12	516.4
POI	121.4	1.96	1.75	57.89	19.8	26.94	2.82	60.45	1019.4

Table 2 Prediction accuracy

δ	1st stage					2nd stage				
	3%	5%	10%	20%	40%	3%	5%	10%	20%	40%
Ad-hoc	17.81%	24.99%	34.81%	41.19%	56.33%	24.91%	35.03%	57.54%	79.30%	97.55%
POI	4.67%	8.29%	21.77%	50.63%	78.64%	5.60%	9.91%	26.16%	61.11%	94.20%

the avg. execution time of each master LP optimization (in sec.), the avg. execution time of each dynamic programming pricing algorithm (in sec.), the avg. number of paths added at each column generation iteration; the total execution time (in sec.) is also reported. Values are averaged over the 5 instances of each generative model. Our methods show to be computationally stable, the most critical point being the master LP optimization during second stage optimization. Affordable computing times are also observed on a real-world dataset concerning about 600 APs in Paris [1].

Then we assess the accuracy of our methods in rebuilding mobility patterns from demand and path length distributions. Here we focus only in rebuilding user trajectories in terms of origin and destination, being the target of both the original application and related works in the literature [2]. We assume each prediction to be correct if both origin and destination APs of predicted paths fall within distance δ from origin and destination APs of original ones. We designed a maximum likelihood procedure, that is based on flow computations, and outputs the best matching between predicted and original paths. Let $\mathcal{N}(i,j)$ (resp. $\widetilde{\mathcal{N}}(i,j)$) be the number of users whose origin is i and destination is j in the original paths (resp. predicted paths according to such a maximum likelihood matching). As accuracy measure we consider $\sum_{i\in N, j\in N} \min(\mathcal{N}(i,j), \widetilde{\mathcal{N}}(i,j))/U$. Table 2 reports, for each stage of our algorithm (column blocks) and for each generative model (table rows), the average accuracy obtained when different δ correction levels are allowed; δ values are reported as percentage of the radius of the instance region. As expected, exploiting second stage optimization substantially improves accuracy. Values of δ as low as 10% are enough to make predictions on ad-hoc models reach 57.5% accuracy, and values of δ of 20% yield average prediction accuracy of almost 80%. POI models are harder to predict. Still 60% accuracy can be achieved when $\delta = 20\%$.

Acknowledgements The project has been partially funded by Regione Lombardia—Fondazione Cariplo, grant n. 2015-0717, project REDNEAT.

References

1. Ceselli, A., Premoli, M., Secci, S.: Cloudlet Network Design Optimization. In: Proceedings of 2015 IFIP Networking, Toulouse (2015)
2. Liang, X., Zhao, J., Dong, L., Xu, K.: Unraveling the origin of exponential law in intra-urban human mobility. Nature—Scientific Reports, vol. 3 (2013)