Clustering Mobile Trajectories for Resource Allocation in Mobile Environments*

Dimitrios Katsaros[1], Alexandros Nanopoulos[1], Murat Karakaya[2], Gokhan Yavas[2], Özgür Ulusoy[2], and Yannis Manolopoulos[1]

[1] Department of Informatics,
Aristotle University, Thessaloniki, 54124, Greece
{dimitris, alex, manolopo}@skyblue.csd.auth.gr

[2] Department of Computer Engineering
Bilkent University, Bilkent 06800, Ankara, Turkey
{muratk, gyavas, oulusoy}@cs.bilkent.edu.tr

Abstract. The recent developments in computer and communication technologies gave rise to Personal Communication Systems. Due to the nature of the PCS, the bandwidth allocation problem arises, which is based on the notion of *bandwidth-on-demand*. We deal with the problem of how to predict the position of a mobile client. We propose a new algorithm, called *DCP*, to discover user mobility patterns from collections of recorded mobile trajectories and use them for the prediction of movements and dynamic allocation of resources. The performance of the proposed algorithm is examined against two baseline algorithms. The simulation results illustrate that the proposed algorithm achieves recall that is comparable to that of the baseline algorithms and substantial improvement in precision. This improvement guarantees very good predictions for resource allocation with the advantage of very low resource consumption.

1 Introduction

The recent developments in computer and communication technologies gave rise to Personal Communication Systems (PCS), which ensure ubiquitous availability of services. Unlike ordinary static networks (e.g., public telephone network), PCS allows the dynamic relocation of mobile terminals. This *network mobility* gives rise to some new and important problems. Among them are the *location management* and the *bandwidth allocation* problems. Location management consists of two issues, namely the *location* and *paging* procedures. The former allows the system to keep the user's location knowledge (exact or approximate) in order to be able to locate him. The paging procedure consists of sending paging messages in all the locations (cells) where the mobile client could be located. The bandwidth allocation issues arise due to the nature of the PCS, which is

* This research has been funded through the bilateral program of scientific cooperation between Greece and Turkey. (*Γ.Γ.Ε.Τ.* and from *TÜBITAK* grant no 102E021.)

based on the notion of *bandwidth-on-demand*, that is, allocate network resources only after there exists a request for them.

The issue of location management has mainly focused on the procedure of database updating and relatively little (e.g., [6,7,1]) has been done in the area of location prediction. Location prediction is a dynamic strategy in which the PCS tries to proactively estimate the position of the mobile client. Location prediction could provide substantial aid in the decision of resource allocation in cells. Instead of blindly allocating excessive resources in the cell-neighborhood of a mobile terminal, we could selectively allocate resources to the most "probable-to-move" cells. Thus, we could cut down the resource wastage without introducing an increase in the *call drop ratio*.

Related work. Location prediction is addressed in the following works: [6,7, 5,11,1,4,13]. Nevertheless, they exhibit one or some of the following shortcomings: a) They assume the existence of user movement patterns (by only applying pattern matching techniques), without providing a method to discover them in the first place [7]. b) They rely on knowledge of the probability distribution of the mobile's velocity and/or direction of movement [5,1], which assumes the existence of a sophisticated and costly equipment, e.g., GPS. c) They are susceptible in small deviations in client trajectories, which cause significant degradation in prediction accuracy because they do not distinguish between regular and random movements of a mobile [6]. d) They do not exploit the valuable historical information that in any case is gathered (logged) from the PCS and regards the daily movements of clients from cell to cell (intercell movements), spanning large time intervals. Finally, the works described in [4,13,2] address different problems and their proposed solutions are not applicable in our problem. The work in [4,2] deal with cache relocation and data relocation, respectively, and the work in [13] aims at designing a better paging area for each mobile client for each time region and thus, the focus of the work is the estimation of the time-varying location probability. The only relevant work to ours is that in [11], which proposes recording the "cell-to-cell" transition probabilities in order to make predictive resource reservation. The deficiencies of this method are shown by the experimental results of our work.

In this work, we deal with the problem of location prediction for efficient resource allocation. Our contributions are:

- We identify the importance of exploiting historical movement data in the context of predictive resource allocation.
- We develop a new clustering algorithm for discovering regularities in movements of mobile clients to address the large volumes of historical data.
- The design of the proposed algorithm considers the important factor of randomness in the mobile trajectories, which can result in noise within trajectories themselves or to outliers (trajectories that do not follow any regularities). This is addressed by the used distance measure and by the method of representing the clusters.
- The performance of the proposed algorithm is examined against two baseline algorithms, based on simulation. The first of them (EXH) allocates resources

to every neighboring cell and the second one (TM) performs allocation based on cell-to-cell transition probabilities. The simulation results illustrate that the proposed algorithm achieves recall that is comparable to that of the baseline algorithms and substantial improvement in precision (40% over EXH and 25% over TM). This improvement guarantees very good predictions for resource allocation with the advantage of very low resource consumption.

The rest of this article is organized as follows: Section 2 describes the network model assumed in the present study. In Section 3 we give the problem and present its solution in Section 4. Finally, in Section 5 we present the experimental results and in Section 6 we conclude this paper.

2 Preliminaries

In the present work, we consider a wireless PCS (see Figure 1) with architecture similar to those used in *EIA/TIA IS-41* and *GSM* standards [9]. The PCS serves a geographical area, called the *coverage area*, where *mobile users* or *terminals (MU)* or *(MT)* can move. The coverage area served by the PCS is partitioned into a number of non-overlapping regions, called *cells*. At the heart of the PCS lies a fixed backbone (wireline) network. A number of fixed hosts are connected to this network. Each cell is served by one *base station* (BS), which is connected to the fixed network and it is equipped with wireless transmission and receiving capability.

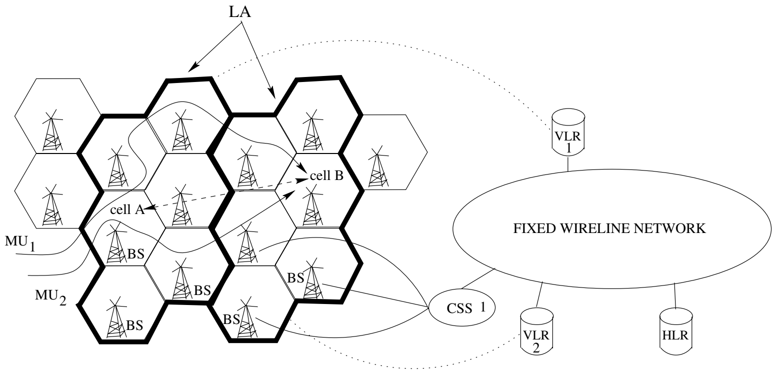

Fig. 1. A typical architecture for a PCS.

Following the settings in [13,4] where each mobile client can report its trajectories to the fixed network, we collect for each user trajectories of the form $T = \langle (id_1, t_1), \ldots, (id_k, t_k) \rangle$. The id_i represents the ID of the cell that the client entered at time t_i $(1 \leq i \leq k)$. The characteristic of this sequence is that for every i $(1 \leq i \leq k-1)$ the id_i and id_{i+1} correspond to neighboring cells. Following related work on session formation, each such trajectory T is divided into subsequences based on a threshold for the difference $t_{i+1} - t_i$ (i.e., when a client stays in a cell for a sufficiently long time).

The procedure of trajectory recording ends up with a host of the fixed network collecting (logging) the movements T of individual mobile clients within the coverage area. Let these trajectories be called *User Actual Paths (UAP)*. UAPs are an invaluable source of information, because the "movement of people consists of random movement and regular movement and the majority of mobile users has some regular daily (hourly, weekly, ...) movement patterns and follow these patterns more or less every day" [7]. In this work, we focus on discovering patterns from multiple users, because it better fits the needs of resource allocation problem.

3 Problem Formulation

Let $\mathcal{D}$ be a collection of logged user actual paths (UAPs), each one of the form $U = \langle c_1, \ldots, c_n \rangle$, where c_i is the ID of the i-th cell in the two dimensional grid of the coverage area U. The discovery of patterns in clients' movement can be attained by organizing the logged UAPs according to their *in-between similarity.* The similarity considers the network topology [7]. Such a procedure identifies groups of similar UAPs. These groups represent a kind of aggregation of large numbers of UAPs into a set of much less patterns, which correspond to the expected regularities which will be called *user mobility patterns* (UMPs).

Having a set of discovered UMPs, let a client that up to the current time point has visited cells described in UAP $C = \langle c_1, \ldots, c_t \rangle$. According to the similarity of UAP C with the discovered UMPs, the movement of the client can correspond to one (or possibly more) discovered regularity. Based on this correspondence, the mobile system can predict the forthcoming movements of the client.

In summary, what is required first is a scheme for the organization of similar UAPs into UMPs. Due to the large sizes of logged UAP collections and the possibility of the existence of noise and outliers, the paradigm of data mining will be followed for this purpose. Second, based on the discovery scheme, we need a method for the dynamic prediction of forthcoming clients' movements and the allocation of resources.

4 Dynamic Clustering-Based Prediction of Mobile-Client Movements

Let $A = \langle a_1, \ldots, a_m \rangle$ and $B = \langle b_1, \ldots, b_n \rangle$ be two cell sequences. To measure their similarity we are based on the well known metric of *edit distance*. However, to reflect the geometry of the cell grid, we use weighted edit distance measures [7], considering that it suffices to limit the effect of temporal deviation from a specific cell only within the cells *around* it. We extend [7] so that, for a cell c, we examine all the cells with a city-block distance from c, denoted as CB[1], less than or equal

[1] Given a cell c_1 with coordinates (x_1, y_1) and c_2 with (x_2, y_2), $CB(c_1, c_2) = \max\{|x_1 - x_2|, |y_1 - y_2|\}$.

to t_d. Henceforth, for two UAPs A and B, the weighted edit distance is denoted as $d_e(A, B)$. The selection of t_d is value is discussed in Section 5.

The UAPs and the aforementioned weighted edit distance form a metric space which, however, may not be Euclidean. Although several clustering algorithms have been proposed for Euclidean spaces (e.g., [3] and the references therein), we have to focus on algorithms that can be used for general metric spaces. Hierarchical agglomerative algorithms belong to the latter category, because they are not based on coordinates of an Euclidean space. Plain hierarchical clustering uses one medoid object for each cluster. Other approaches extended this case and represent a cluster with more objects, e.g., [8] which represents clusters by a subset of distant objects within them (this approach was also adopted in [3, 10]).

For a cluster c, let c.rep denote the set of representative UAPs. Given two clusters c and c', their distance $d(c, c')$ is defined as

$$d(c, c') = \min_{\forall u \in c.rep, v \in c'.rep} \{d_e(u, v)\}$$

A general scheme of hierarchical algorithm to cluster UAPs is given Figure 2. For an efficient implementation of HierCluster, we use a heap to maintain the distances between the clusters. Following the approach of [3,10], we sample the initial collection of UAPs, to be able to efficiently process, with the clustering algorithm, the selected sample of UAPs in main memory. The merging of representative points can be done in several ways. Based on [3], when merging two clusters c and c' into a new cluster n, we initially set n.rep $=$ c.rep $\cup$ c'.rep. If $|n|$.rep $\geq r$, then the subset of the r most distant members of n.rep can be found.[2] The aforementioned approach is denoted as FU (selects the Farthest UAPs). Although FU captures the shape of a cluster, it may be sensitive to outliers (i.e., UAPs that do not to belong to clusters and tend to form isolated small neighborhoods), because they can become representatives (since projection cannot be applied) and impact the correct cluster detection.

Following the approach of [10] when merging two clusters, if $|n|$.rep $\geq r$, then the medoid representative can be found and then the procedure selects the $r - 1$ nearest neighbors of it. Thus, the medoid and its nearest neighbors form the n.rep set. We denote this method as NN. Its objective is to reduce the sensitivity of the clustering algorithm against the outliers, since it reduces the probability of outliers to become representatives.

The result of HierCluster is the set of UMPs, each one consisting of a group of representatives. Given a mobile client that currently follows a UAP, the purpose of prediction is to first find the UMPs that best match the client's current UAP. Then, based on the matched UMPs, the most probable next movements are predicted, and resource allocation is applied for them. Let A be a client's UAP (its recently crossed cells), where $A = \langle a_1, \ldots, a_k \rangle$. A is examined

[2] The shrinking and projections (towards the center) of the selected subset of representatives can be applied for Euclidean spaces (they use the coordinates), thus not for the problem examined herein.

```
Algorithm HierCluster
Input: The collection $D$ of UAPs
       The number $k$ of required UMPs
       The max number $r$ of repr.
Output: The discovered UMPs
1. Preprocess($D$)
2. while $c > k$ {
3.     extract heap root with two clusters $c$, $c'$
       for which $d(c,c') \leq \min_{x,y \neq c,c'}\{d(x,y)\}$
4.     merge $c$ and $c'$ into a new cluster $n$
5.     merge $c$.rep and $c'$.rep to form $n$.rep
6.     update the heap
7.     $c = c - 1$
8. }
9. return the set of discovered clusters
end of HierCluster
```

```
Procedure Preprocess
1. foreach $u \in D$ {
2.     form a $c_u$ cluster
3.     set $c_u$.rep $= \{u\}$
4.     build the heap
5. }
6. set $c = |D|$
end of Preprocess
```

Fig. 2. The clustering algorithm.

against each UMPs B to find those with distance $d_m(A, B) < t_m$. In such a case we denote that B is matched against A. The calculation of d_m must take into account that a UAP A can be part (i.e., subsequence) of a UMP B. The solution of this problem, which is based on dynamic programming, can be found in [12]. Therefore, assuming that $B = \langle b_1, \ldots, b_j, \ldots b_n \rangle$, the remaining cells in the part $\langle b_j, \ldots, b_n \rangle$ of B can be used for the prediction of the next movements of the client.

The number of predictions that can be made from each matched UMP is set as a parameter. Assume that the remaining part of a matched UMP consists of p cells $\langle b_j, \ldots, b_n \rangle$ (i.e., $n - j = p$). We can use up to a maximum number m ($m \leq p$) cells from this part, by maintaining the order of their appearance, i.e., cells b_j, b_{j+1}, b_{j+m-1} can be predicted as the next movements of the client. The aforementioned cells are denoted as the *predicted trajectory* of the client. Evidently, one may expect that cell b_j has larger probability of correct prediction than b_{j+1}, b_{j+1} than b_{j+2} and so on. For the tuning of m it has to be considered that the smaller m is the more accurate predictions are made. However, the total number of correct predictions is going to be smaller. The experimental results in Section 5 elaborate further on this issue.

Finally, it has to be noticed that the current UAP of a mobile client can be matched by one or more UMPs (i.e., representatives) in each cluster. Thus, one may select the best matched UMP (with the minimum d_m) or all the matched UMPs. We have examined both options and we choose the latter one, because we found that it leads to better performance, since the number of correct predictions increases without significant increase in the total number of predictions made (that corresponds to resource consumption).

Symbol	Definition	Default
m	max length of predicted trajectory	2
r	max number of representatives	20
t_d	cut-off CB distance for d_e	2
L	average length of UAPs	5
c	corruption factor	0.4
o	outlier percentage	30%

Fig. 3. Symbol table.

5 Performance Study

This section presents the empirical evaluation of the proposed method (DCP), which is compared against TM-k and EXH. TM-k makes predictions based on the cell-to-cell transition matrix, where k denotes the number of the most probable-to-visit cells. As described, EXH is the method that allocates resources to every neighboring cell. We first describe the simulation model for the mobile system considered. Next, we give the results for the sensitivity of DCP, by comparing it with TM-k and EXH. Finally, we examine the efficiency of the prediction procedure of DCP with respect to execution time.

To simulate the mobile computing system, we assumed a square-shaped mesh network consisting of 225 cells. To generate UAPs, we first formed a number of UMPs whose length is uniformly distributed with mean equal to L and max equal to $L + 3$. Each UMP is taken as a random walk over the mesh network. We had two types of UAPs, those following a pattern (i.e., a UMP) and those not. The latter are called *outliers*, and their ratio to the number of UAPs that follow a UMP is denoted as o. The formers are formed as follows: First, a UMP is selected at random. Then, with a corruption ratio c, we intervene random paths through the nodes of the selected UMP. The total number of the nodes in these random paths are equal to c times the size of the UMP. The default number of generated UAPs is 100,000. From this set of UAPs we form the train and an evaluation data set. The sampling for the clustering algorithm is done according to the results in [3,10]. The performance measures we used are the following: a) *recall*, which represents the number of correctly predicted cells divided by the total number of requests and b) *precision*, which represents the number of correctly predicted cells divided by the total number of predictions made. Figure 3 summarizes the symbols used henceforth, and, for each one, gives the default value used throughout the performance evaluation.

We saw that the DCP has some parameters, namely the cluster-merging method, the number m of maximum predictions made at each time, and the number r of representatives used for clusters, which require tuning. Similar tuning for the k is required also for the TM-k algorithm. We performed extensive experiments in order to select appropriate values for these parameters. Due to the limited space we do not present here the experimental results for the tuning of the algorithms.

5.1 Sensitivity Experiments

In this series of experiments, we examine the sensitivity of DCP with respect to the pattern length L, the percentage c of corruption, and the ratio o of outliers. DCP is compared with TM-2 and EXH.

Initially we examined the impact of L. In the left part of Figure 4 it can be clearly observed that, for very small L values (i.e., 2), the recall of all methods is reduced significantly. Evidently, the first cell-visit in a UAP may not be predicted by any method. Therefore, when $L = 2$, there is not enough room for correct predictions (even for EXH). With increasing L, recall is increased. EXH clearly attains the best recall in all cases.

However, it has to be considered that EXH is, in fact, an upper-bound for the recall of *any* prediction scheme, because it is not dynamic and bases its prediction on all possible forthcoming cells visits (EXH may miss only the first visit in each UAP). Based on this fact, the dynamic methods, i.e., DCP and TM-2, achieve only around 10% smaller recall than EXH. The recall of DCP and TM-2 are comparable with each other. However, the good performance of DCP is not to the cost of resource consumption.

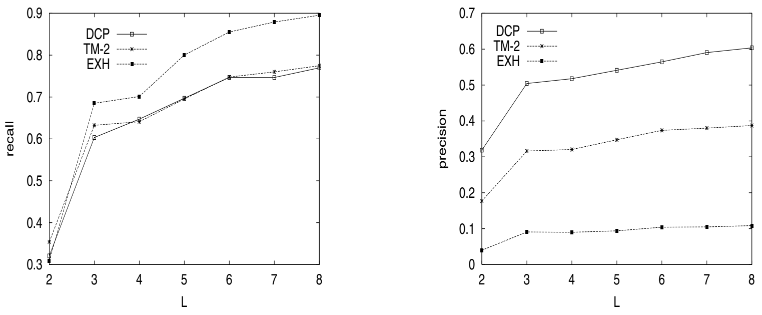

Fig. 4. (Left) Recall vs L and (Right) Precision vs L.

This is shown in the right part of Figure 4, which illustrates precision. Clearly, EXH has the lowest precision (around 0.1), since it allocates every possible cell. This indicates the ineffectiveness of EXH in realistic cases, due to its high resource consumption. TM-2 is second best, but significantly outperformed by DCP in all cases. Since L is the only factor that affects EXH (for the reasons explained earlier), we do not further examine this method, because it achieves constant recall (80%) and precision (1%) with respect to all the following factors. This helps for a more clear comparison between the two dynamic methods, i.e., DCP and TM-2.

Next, we examined the impact of corruption in the data set, measured through parameter c. The results are illustrated in Figure 5 (c is given as a percentage). Focusing on recall (left part of Figure 5), both methods are not affected significantly by increasing c, and their recall is comparable. The precision of DCP (right part of Figure 5) for very low c values is very high, reaching 90% for the case of no corruption. However, with increasing c, the precision of DCP reduces, since the clustering, performed by it, is affected. However, for c larger than 40% the reduction is less significant.

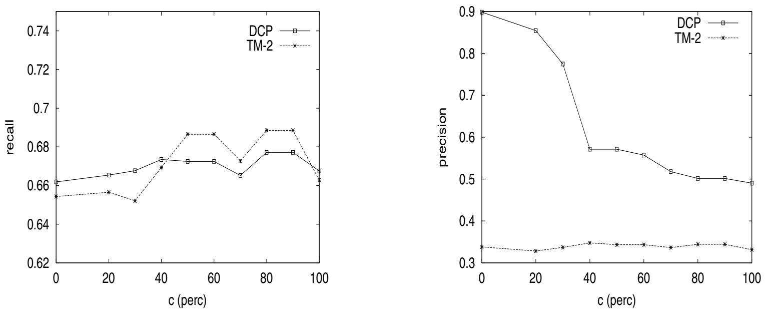

Fig. 5. Comparison of DCP and TM-2 w.r.t. c: (Left) Recall. (Right) Precision.

This is explained from the fact that before this value, DCP performs almost unhindered matchings of UAPs against patterns, reaching the aforementioned very high values of precision. However, these cases does not correspond to realistic ones, because the absence of corruption is not probable[3]. But for larger values of c, DCP manages to maintain a good value of precision (around 50%) without being affected by further increase in c. Evidently, DCP outperforms TM-2, which is less affected by c but does not attain precision larger than 35%.

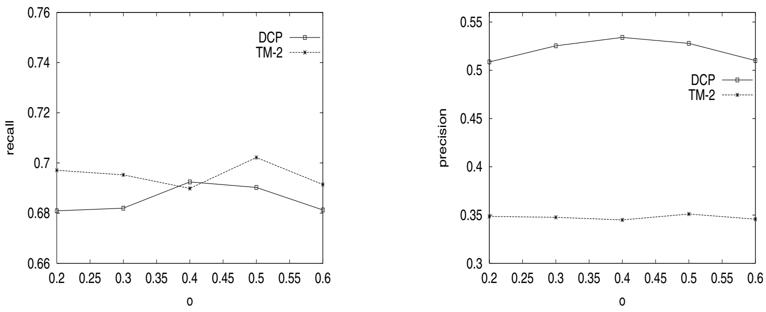

Fig. 6. Comparison of DCP and TM-2 w.r.t. o: (Left) Recall. (Right) Precision.

Finally, we examined the impact of outliers in the data set. The results are illustrated in Figure 6 with respect to the outlier percentage o. The recall (left part of Figure 6) of both methods is not affected significantly by increasing o,

[3] This is the reason why throughout the rest experiments we do not use c smaller than 40%.

where DCP and TM-k attain comparable recall values. The precision of both methods (right part of Figure 6.b) is also not affected significantly. For DCP this indicates that the filtering used by its clustering procedure is effective in reducing the impact of outliers (TM-k does not use clustering, thus is not directly affected by outliers). However, DCP compares favorably with TM-2 in all cases.

5.2 Execution Time

For DCP, the on-line computation of predicted trajectories is performed each time a client moves to a new cell. For this reason, we examined the execution time required for this procedure. Results are depicted in Figure 7 (time is measured in ms) with respect to the average UAP length L. More precisely, the depicted times correspond to the application of prediction procedure at each cell, where the UMPs have been pre-computed. As shown, execution time increases with increasing L, since for larger UAPs the computation of d_m distance requires more steps. However, in all cases, the prediction requires only few ms. Assuming that a client that reaches a cell may stay in it for a period that is much larger than few ms, the prediction time is a negligible overhead.

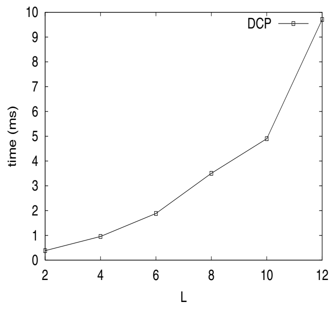

Fig. 7. Execution time (in ms) per cell of the prediction procedure of DCP.

6 Conclusions

We considered the problem of predictive resource allocation in cellular mobile environments. We proposed a new algorithm called *DCP*, to discover user mobility patterns from collections of recorded mobile trajectories and to use them for the prediction of movements and dynamic allocation of resources.

The design of the proposed algorithm considers the important factor of randomness in the mobile trajectories, which can result in noise within trajectories themselves or to outliers (trajectories that do not follow any regularities).

Using simulation, the performance of *DCP* was compared against that of *EXH* and *TM*. Our experiments showed that for a variety of trajectory length, noise and outliers, *DCP* achieves an very good tradeoff between prediction recall and precision.

References

1. A. Aljadhai and T. Znati. Predictive mobility support for QoS provisioning in mobile wireless environments. *IEEE Journal on Selected Areas in Communications*, 19(10):1915–1930, 2001.
2. W.-C. Chen and M.S. Chen. Developing data allocation schemes by incremental mining of user moving patterns in a mobile computing system. *IEEE Transactions on Knowledge and Data Engineering*, 15(1):70–85, 2003.
3. S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *Proceedings of the ACM Conference on Management of Data (ACM SIGMOD'98)*, pages 73–84, 1998.
4. S. Hadjiefthymiades and L. Merakos. Using proxy cache relocation to accelerate Web browsing in wireless/mobile communications. In *Proceedings of the World Wide Web Conference (WWW'01)*, pages 26–35, 2001.
5. B. Liang and Z. Haas. Predictive distance-based mobility management for PCS networks. In *Proceedings of the IEEE Conference on Computer and Communications (IEEE INFOCOM'99)*, pages 1377–1384, 1999.
6. G.Y. Liu and M.Q. Gerald. A predictive mobility management algorithm for wireless mobile computing and communications. In *Proceedings of the IEEE International Conference on Universal Personal Communications*, pages 268–272, 1995.
7. T. Liu, P. Bahl, and I. Chlamtac. Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks. *IEEE Journal on Selected Areas in Communications*, 16(6):922–936, 1998.
8. R. Michalski, R. Stepp, and E. Diday. A recent advance in data analysis: Clustering objects into classes characterized by conjuctive concepts. In *Progress in Pattern Recognition*, volume 1. North Holland Publishing, 1983.
9. S. Mohan and R. Jain. Two user location strategies for Personal Communication Systems. *IEEE Personal Communications Magazine*, pages 42–50, First Quarter 1994.
10. A. Nanopoulos, Y. Theodoridis, and Y. Manolopoulos. C^2P: Clustering with closest pairs. In *Proceedings of the 27th Conference on Very Large Data Bases (VLDB'01)*, pages 331–340, 2001.
11. S. Rajagopal, R.B. Srinivasan, R.B. Narayan, and X.B.C. Petit. GPS-based predictive resource allocation in cellural networks. In *Proceedings of the IEEE International Conference on Networks (IEEE ICON'02)*, pages 229–234, 2002.
12. P.H. Sellers. An algorithm for the distance between two finite sequences. *Journal of Algorithms*, pages 359–373, 1980.
13. H.-K. Wu, M.-H. Jin, J.-T. Horng, and C.-Y. Ke. Personal paging area design based on mobile's moving behaviors. In *Proceedings of the IEEE Conference on Computer and Communications (IEEE INFOCOM'01)*, pages 21–30, 2001.