Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computer Networks 45 (2004) 715–730

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Efficient location area planning for cellular networks with hierarchical location databases

Shi-Wu Lo [a], Tei-Wei Kuo [a], Kam-Yiu Lam [b,*], Guo-Hui Li [c]

[a] *Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, ROC*

[b] *Department of Computer Science, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon Tong, Hong Kong, PR China*

[c] *School of Computer Science and Technology, Huazhong University of Science & Technology, Wuhan, Hubei, PR China*

Received 10 July 2003; received in revised form 20 February 2004; accepted 25 February 2004
Available online 7 April 2004

Responsible Editor: J.Y.B. Lin

Abstract

Location area planning (LAP) is an important issue in the design of high-performance PCS networks. It could have a serious impact on the total mobility management cost of mobile terminals. Most of the previous works either explored the LAP problem as a 0–1 linear programming problem or used adopted techniques, such as simulated annealing, taboo search, and genetic algorithms [IEEE Trans. Vehicular Technol. 49 (2000) 1678; Proceedings of 1999 Vehicular Technology Conference, vol. 4, 1999, pp. 2119–2123; IEEE Vehicular Technol. Conf. 3 (1996) 1835; Proceedings of IEEE INFOCOM'01, Anchorage, Alaska, April 2001; IEEE Trans. Vehicular Technol. 47 (1998) 987], to derive a solution to minimize the location update cost. In this paper, we model and resolve the LAP problem as a *set-covering problem*. The main advantage of this approach is that it can adapt to the changing mobility patterns of the mobile terminals. We propose the *set-covering-based location area planning (SCBLP)* algorithm to minimize the total number of location updates, in which the cost-benefit functions are defined based on the coupling and cohesive functions among neighboring cells. We then apply SCBLP to the location database system with a hierarchical structure to further improve the overall system performance in searching and updating the location databases. Extensive simulation experiments have been conducted, and the experimental results show that our proposed algorithms can significantly reduce the location management costs, compared to the greedy algorithm and the random algorithm.

Keywords: Location area planning; Location updates; Location management; Set covering; PCS

1. Introduction

The number of personal communication service (PCS) subscribers increases rapidly in recent years. With limited radio resources in a cell, one of the major solutions to support the growing number of mobile terminals (MTs) is to use small cells so as to increase the degree of radio frequency reuses. For example, the Taiwan Cellular Corporation has built 2500 base stations in Taiwan [19], and the number of base stations keeps growing rapidly. An important problem associated with the use of

* Corresponding author.
E-mail addresses: d89015@csie.ntu.edu.tw, ktw@csie.ntu.edu.tw (T.-W. Kuo), cskylam@cityu.edu.hk (K.-Y. Lam).

small cells is the high cost in mobility management.

In a PCS network, location databases are used to record the locations of MTs. The location information is important to efficient delivery of connection calls. The whole system service area of a PCS network is divided into location areas (LAs), and each LA consists of a group of neighboring cells. A location database is responsible for recording the MTs residing in an LA or within a group of neighboring LAs. The data items in a location database are real-time data since their values may change rapidly with time due to the movement of the MTs. In order to maintain the validity of location data such that they truly reflect the current locations of the MTs, the MTs have to generate location updates periodically, especially when an MT enters into a new LA.

The cost in mobility management can be divided into two major parts: *location update cost* and *searching cost (or paging cost)*. In general, the minimizing of the location update cost might increase the paging cost and paging delay. If the location update frequencies of MTs are low, then the database may contain out-dated location information of the MTs. Thus, the paging cost and paging delay for searching the exact locations of the MTs will be large since the system needs to search a large number of cells before it can identify the current locations of the MTs. High paging cost and delay is very much undesirable to future PCS networks. High update frequencies of MTs can also be a heavy burden to the PCS networks since the number of MTs is increasing rapidly. How to provide a balance between these two factors is an important system design issue, and it has attracted a lot of research work in recent years [24–26].

Two important factors affecting the number of location updates and the update processing costs are: (1) how the cells are grouped into location areas, and (2) how the location databases are organized. Apparently, the solutions for solving these two problems depend on how the MTs move among the cells. It is the focus of this paper to resolve these two problems to minimize the total location management cost. As shown in the previous work that if the cells are not grouped appropriately, the number of location updates will be very large. The problem how to group the cells into location areas is called the *location area planning (LAP)* problem. It is a NP-complete problem [4]. In last decade, various methods have been proposed to resolve this problem. Some of them resolve it using 0–1 linear programming algorithms while the others adopt techniques, such as simulated annealing, taboo search, and genetic algorithms [3,4,8,10,14], with the objective to derive a feasible solution to minimize the location update cost without increasing the paging cost.

Another way to resolve the problem is to minimize the cost for update processing which is highly related to how the location databases are organized. In existing PCS networks, location databases are organized in a two-tier structure: the *home location register (HLR)* and a number of *visitor location registers (VLRs)* [12]. The *home location register (HLR)* contains the locations of all the MTs subscribed to the system. A *visitor location register (VLR)* maintains the location information of the MTs currently within the group of cells managed by it. In searching an MT, the current VLR of the caller will be searched first and then HLR will be looked up if the MT cannot be found in the current VLR.

Although a two-tier structure [2,6] is simple and easy to implement, it could have a serious performance problem in the next generation PCS networks such as the scalability problem and the heavy workload problem at the HLR (which includes the workload for processing a location update signal and the network traffic), especially if the number of MTs is large and their mobility is high [21–23]. The processing cost of a location update is not only the cost of executing the database operations for installing the update but also the transmission overheads of the update in the network. The transmission overheads really depend on the location of a mobile terminal from the location registers. In addition to that, the searching delay for a mobile terminal could be long when there is a large location database in the system. Because of that, researchers started proposing different location database structures to improve the scalability of the system in recent years. The objective is to minimize the processing cost and searching cost for updates. One of the promising methods is to organize the location database hierarchically. In a hierarchical location database structure, location databases are organized into a tree structure with each leaf node of a location database responsible for managing the location information of the MTs within an LA (or a group of LAs) [1]. It has been shown being a good solution for the next generation PCS networks [1]. Compared to the two-tier structure, the hierarchical structure is more reliable and scalable because a hierarchical structure tends to distribute the location updating and searching workloads over multiple servers. The searching cost is smaller since mobile users tend to call and move among geographically neighboring areas, and most connections are between mobile terminals maintained at lower-level databases in the hierarchical tree.

In this paper, we propose to resolve the LAP problem as a *set-covering (SC) problem*. We propose an algorithm, called *set-covering-based location area planning (SCBLP)*, to group the cells into location areas. The objective is to minimize the total location update cost. The main advantage of the proposed scheme is its adaptability to dynamic mobility patterns of mobile terminals. Note that the mobility patterns of MTs can be highly dynamic and may change rapidly with time. If we can reduce the update cost, more updates may be generated to improve the accuracy in tracking the mobile terminals. The consequence will be lower paging cost in searching a mobile terminal when a connection call request is received. We then apply SCBLP to organize the location databases in a hierarchical location database structure to further minimize the processing costs for updating and searching the location databases. Simulation results have shown that the proposed algorithms can significantly reduce the PCS network traffics and location management cost, compared to other methods, i.e., random and greedy algorithms.

The rest of this paper is organized as follows: Section 2 summarizes the related work. Section 3 defines the LAP problem. In Section 4, we propose our approach in location planning and define the benefit functions for determining how to group the cells into location areas. Section 5 extends the results in Section 4 for a hierarchical database structure to minimize the update processing and searching costs. Section 6 shows the experiment results on evaluating the performance of the proposed algorithms as comparing with other algorithms. Section 7 is the conclusions.

2. Related work

Location area planning (LAP) is an important issue to mobility management. Some researchers modeled the LAP as a 0–1 linear programming problem [3,4,8,10,14], in which searching techniques, such as taboo search, genetic algorithms, and simulated annealing, were employed to derive a proper planning for location areas to minimize the total number of location updates.

The simulated annealing (SA) [4,10] randomly moves a cell from one location area to another area to generate a neighboring solution and compares the neighboring solution to the local optimal solution. The neighboring solution will be accepted if it improves the local optimal solution. The taboo search (TS) [3,4] can be used to resolve the 0–1 linear programming problem. The TS treats a solution space as a connected graph, in which each node denotes a feasible solution. The genetic algorithms (GA) [4,8,14] are similar to the SA and TS in the searching mechanisms, except that they generate a neighboring solution using two existing feasible solutions. One of the major problems of these algorithms is the heavy processing overhead due to the high complexity of the algorithms and the difficulty in determining the optimal parameters for calculating the optimal solution. If the mobility patterns of the mobile terminals change rapidly, these algorithms are usually not suitable to location area planning, especially in an on-line fashion [7,13].

A greedy algorithm was proposed by Plehn [15] to merge two LAs (i.e., groups of cells) in each iteration using the maximum profit to minimize the number of location updates. Munguia-Macario et al. [13] proposed to use a constrained maximum spanning tree to partition a geographic area into LAs. Varsampoulos and Gupta [12] proposed to dynamically adjust the coverage of location areas based on mobility patterns of the mobile terminals. Researchers in location planning, in general, agree that the roadmap and the traffic flows have significant impacts on the mobility patterns. For example, mobile terminals that move in the same direction on a highway tend to have a similar mobility pattern. Bejerano and Cidon [6] proposed to dynamically adjust the coverage of LAs by predicting the speed of the mobile terminals on highways. Although these approaches may reduce the network traffic by moving or overlapping the coverage areas of LAs to fit the traffic speed and movement of mobile terminals, how to partition the coverage area appropriately is still an open problem.

The previous works on location area planning, e.g., [2–6,8,10–15], mainly assumed the use of the traditional HLR–VLR database structure to manage the location databases. To minimize the workload at the HLR and to improve the system performance, a hierarchical location database structure is proposed to organize the location databases in the system [20]. However, it is still an open problem on how to organize the databases into a hierarchical structure with the objective to minimize the update processing cost. It has been shown that variations in the number of neighboring nodes at different levels may seriously affect the system efficiency in processing location updates. Li et al. [17] proposed and summarized methods for location updates over a hierarchical location database structure. They explored the possibility in delaying the location updates of mobile terminals to reduce update traffics and database workloads. In order to reduce the location update cost, some previous works suggested to using location pointers [20]. However, the main concern of this method is that it has additional problem for managing the location pointers. Furthermore, the forwarding pointers may be maintained at any level of the hierarchical location databases. It is difficult to determine the best level to set the forwarding pointers.

3. The location area planning (LAP) problem

In this section, we first use an example to illustrate the *location area planning (LAP)* problem and show how the LAP affects the location update cost in a hierarchical location database system. In the next section, we will propose the SCBLP algorithm to resolve the problem. In Section 5, we will discuss how to apply the SCBLP algorithm to organize the hierarchical databases to minimize the update and searching cost in mobility management.

Consider the example shown in Fig. 1, in which the location databases are organized in a three-layer hierarchy. Let each leaf node (a location database) contain the location information of the MTs in an LA. As shown in Fig. 1, if an MT moves from location area B to location area F, the system has to insert the location information of the MT into the location databases labeled as 1.2.2 and 1.2, and then delete the corresponding records in the databases labeled as 1.1 and 1.1.1. If an MT at location area A requests a connection call to the MT located at location area B, the system has to query the location databases upward from the location database responsible for location area A to the *least common ancestor (LCA)* of location areas A and B, and then go along the proper links of the tree to the leaf node which is responsible for location area B (in which the called MT currently

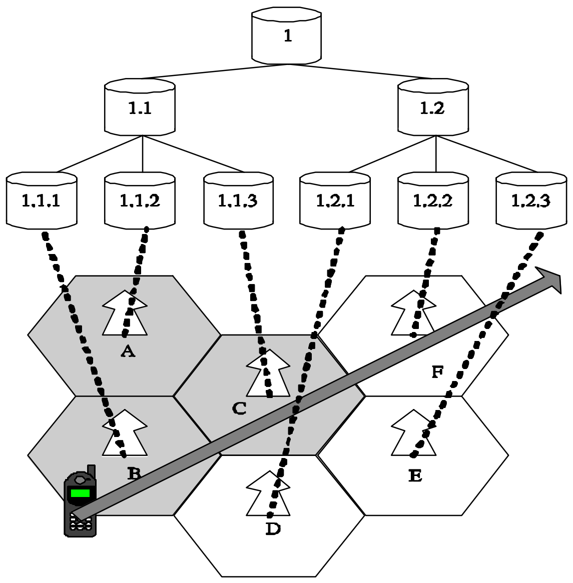

Fig. 1. Location updates and searching of MTs in a hierarchical location database.

resides). We call the length of the path traveling from location areas A to B as the distance between the two location areas, i.e., $\mathrm{dis}(A,B)$. The location update cost and the searching cost depends value of $\mathrm{dis}(A,B)$. If the distance is larger, the searching cost will be heavier.

As shown in Fig. 1, the distance between two location areas in a location database tree depends on how the location databases are organized in the hierarchy. For example, although location area C is geographically next to location area F, the location update cost could be high for an MT moving from location area C to location area F, i.e., the system needs to update the location databases labeled as 1.1.3, 1.1, 1.2 and 1.2.2. This example shows that in order to minimize the mobility management cost, in organizing the cells into LAs (and the location database tree), the system needs to consider the mobility patterns of the MTs, the population of MTs in each LA, and the call frequency of the MTs in each LA. In principles, at each level of a hierarchy, we should group the nodes which have a large number of crossover traffics (due to the movement of mobile terminals) together into the same hierarchy (or sub-tree) to minimize the location update and searching costs.

4. The set-covering-based location area planning (SCBLP) algorithm

4.1. The basic approach

We propose to resolve the LAP problem as a *set-covering* problem in which the relationships among the cells are represented by a *graph*. We apply the *greedy set-covering approximation* algorithm [9,16,18] with proper benefit functions to determine how to group the cells into location areas to minimize the number of boundary crossings among the LAs. Each edge in the graph is associated with a *weight* to denote the traffic volume of MTs between the two neighboring cells. It is assumed that the base stations in the system periodically generate the boundary crossing information of the MTs in the cells, and the statistics are used to define the weight of each edge in the graph. The objective of the LAP problem is to find a *location area design* to minimize the total number of boundary crossings among the LAs such that the number of location updates can be minimized under the constraints of the system and user requirements, such as the maximum number of cells in each location area. For location area planning, limiting the maximum number cells in a location area can limit the paging cost. Once the location area of a mobile terminal is identified, the next job is to page the cells in the location area. If a location area contains more cells, the paging cost will be higher.

4.2. Set covering problem

The LAP problem could be formally defined as follows:

Definition 1. *The location area planning problem.* An instance $G = (V, E, m)$ of the LAP problem consists a set of vertices which are numbered from 1 to n (i.e., $V = \{1, 2, 3, \ldots, n\}$). An edge (i, j) in E indicates that an MT may move from cell i to cell j directly, and vice versa. Each edge is associated with a weight $m(i, j)$ to denote the traffic volume between cell i and cell j. A location area planning problem is to determine a *location area design* to partition V into disjoint subsets $v_1, v_2, \ldots$, and v_n, where $\bigcup_{i=1}^{n} v_i = V$, so that the total value of $\sum_{LA_i \neq LA_j} \sum_{x \in LA_i, y \in LA_j} m(x, y)$ is minimized.

We propose to resolve the LAP problem as a weighted set-covering problem which can be formally defined as follows.

Definition 2. *The weighted set-covering problem.* An instance (S, C, W) of the weighted set-covering problem consists of a finite set S of *items*, a *collection* C of subsets of S, and a weight function $W : C \rightarrow Z_0^+$ such that each *component* of C has a positive weight. The problem is how to find a subset c' of C such that the union of all the subsets in c' is equal to S, and the total weight of c' is minimized.

Let us take Fig. 2 as an example to illustrate the LAP problem. Suppose we have six cells in the

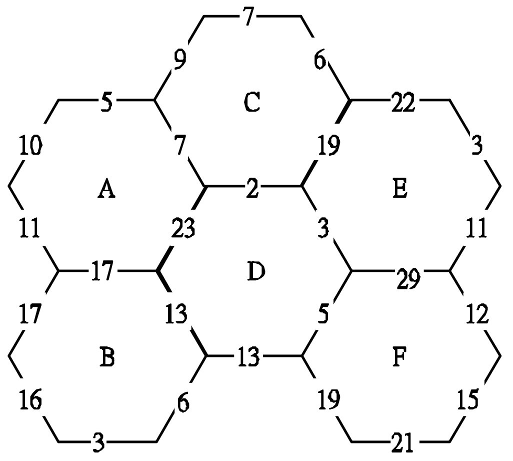

Fig. 2. The number of boundary crossings between cells.

service area under consideration. Each number on the boundary between two neighboring cells represents the average number of boundary crossings between the two cells per unit time. It is assumed that each LA may have three cells in maximum. Therefore, there are $C_3^6/2$ possible combinations for grouping the cells into two LAs with each one has three cells $\left(C_3^6/2 = 6!/(3! \times 3! \times 2)\right)$. For example, we may group them into two LAs, e.g., $\{\{A,B,D\},\{C,E,F\}\}$. The update cost for different location area designs are listed in Table 1. Note that the number of possible combinations is an exponential function of the number of cells.

4.3. The algorithm

The weighted set-covering problem is a well-known NP-complete problem [9,16,18]. It is difficult to get the optimal solution and the computation overhead will be expensive since the number of cells in the service area could be large. In this section, instead of finding an optimal solution in LAP, we propose a heuristic algorithm called *set-covering-based location area planning (SCBLP)* to resolve the problem in polynomial time so that the average performance measured in terms of number of locations updates can be significantly reduced comparing to other existing techniques. We will study the performance of the proposed algorithm in Section 6 through a series of simulation experiments in which we had varied the system parameters to test its performance under different system workload characteristics and system settings as compared with other algorithms, i.e., greedy algorithm. The proposed algorithm is designed based on the greedy-set-cover [9,18] algorithm. It consists of two phases:

1. In the first phase, each location area consists of one cell initially, i.e., one vertex. SCBLP then nine iteratively adds a cell into a location area to maximize the total benefit of the set of location areas under the system constraints, e.g., the maximal number of cells in an LA. Note that there may be a non-empty intersection between two location areas in some cases.
2. In the second phase, we apply the greedy-set-cover algorithm [7] to remove any non-empty intersection between any two LAs. The details of the algorithm are shown in Fig. 3.

The time complexity of the first phase of SCBLP is $O(n * m)$ where $m = \{\max(| V_i |) \mid i = 1, \ldots, | V |\}$. The time complexity of the second phase is $O(n)$. As a result, the time complexity of SCBLP is $O(n * m)$. Because there is usually a constant to bound the maximum number of cells in an LA, m is bounded by the constant. In 10 other words, the time complexity of SCBLP should be $O(n)$.

4.4. Benefit functions

As shown in Fig. 3, Step 4 determines which cells in the neighboring cells of a location area V_i should be merged into V_i. Because there is an intersection area between neighboring location

Table 1
The total location update costs of grouping the six cells into two LAs

LAs	Number of updates
$\{\{A,B,D\},\{C,E,F\}\}$	$324-17-23-13-19-29=223$
$\{\{A,D,C\},\{B,F,E\}\}$	$324-23-2-7-29=263$
$\{\{A,D,F\},\{B,C,E\}\}$	$324-23-5-19=277$
$\{\{C,E,D\},\{A,B,F\}\}$	$324-19-2-3-17=283$

324 is to total sum of all the boundary crossings amongst the cells in the figure.

```
    Algorithm SCBLP
    INPUT:G=(V,E,m)
    OUTPUT: A set of location areas

    Phase1:
1.   FOR each vertex v_i in V
2.       Vi = { v i}
3.       WHILE Vi does not violate the systemconstraints
4.           Select a vertex vj such that the benefit of (Vi∪ {vj}) is maximized
5.           Vi = Vi∪ {vj}
    Phase2:
6.     result-set = ∅
7.     size-of-result-set = 0
8.     WHILE size-of-result-set? |V|
9.       FOReach Vi
10.        IF benefit(Vi) has the maximum benefit
11.          result-set = result-set ∪ Vi
12.        FOR any two different Vj and Vi
14.          IF (Vi ∩ Vj) ≠ ∅
15.            Vj = Vj- Vi
16.            Re-calculate the benefit of Vj
17.            IF Vj = ∅
18.              DELETE Vj
19.          DELETE Vi
20.        size-of-result-set = size-of-result-set + | Vi |
21.    OUTPUT result-set
```

Fig. 3. The SCBLP algorithm.

areas, Step 15 in Fig. 3 re-calculates the benefit of the grouping. In this section, we will propose three benefit functions for cell grouping in SCBLP: *cohesion function, coupling function* and *cost-benefit function.*

4.4.1. Cohesion function

The objective of the cohesion function is to group the cells which have a large number of boundary crossings into the same location area:

$$\text{benefit}_{\text{coh}}(V_x) = \{\text{sum}(m(i,j)) \text{ for different cells } v_i \text{ and } v_j \in V_x\}$$

where $m(i,j)$ is the number of boundary crossings between cell i and cell j. $\text{benefit}_{\text{coh}}(V_x)$ is the total number of boundary crossings amongst the cells in V_x. It denotes the benefits in mobility management if the cells in V_x are in the same group. (Note that an MT which crosses a cell boundary within a location area does not need to report its new position.) Our objective is to maximize $\text{benefit}_{\text{coh}}(V_x)$ since a larger value of $\text{benefit}_{\text{coh}}(V_x)$ implies a greater saving in location updates (smaller update cost).

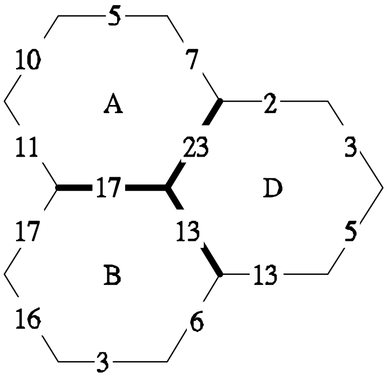

Fig. 4. The example of cohesion function.

The cohesion function can be illustrated with the example shown in Fig. 4 (which is extracted from Fig. 2). The benefit in grouping cells A, B, and D into the same group is $\text{benefit}_{\text{coh}}(\{A, B, D\}) = 17 + 13 + 23 = 53$. Notice that the cohesion function only considers the internal traffic within a group. Similarly, we have the following values for different groupings for the cells in Fig. 2: $\text{benefit}_{\text{coh}}(\{C,E,F\}) = 19 + 29 = 48$, $\text{benefit}_{\text{coh}}(\{A,C,E\}) = 7 + 19 = 26$, and $\text{benefit}_{\text{coh}}(\{B,D,F\}) = 13 + 5 = 18$.

The objective of SCBLP using cohesion function is to increases the degree of cohesion amongst the cells within the same location area. Although the cohesion function does not directly address the main purpose in location area planning, i.e., to minimize the total number of boundary crossings among location areas, the cohesion function minimizes the degree of coupling among the location areas.

4.4.2. Coupling function

The coupling function is to minimize the total number of boundary crossings of the cells in an LA with the cells in the other LAs:

$$\text{cost}_{\text{cou}}(V_x) = \{1/\text{sum}(m(i,j)) \mid v_i \in V_x \text{ and } v_j \notin V_x\}$$

where $m(i,j)$ is the number of cell boundary crossings between cell i and cell j. $\text{cost}_{\text{cou}}(V_x)$ is the total number of boundary crossings between a cell in V_x and a cell outside V_x. $\text{cost}_{\text{cou}}(V_x)$ is the reciprocal of sum$(m(i,j))$. It should be maximized such that the total number of boundary crossings is minimized. Note that this benefit function addresses the objective of SCBLP directly.

The coupling function can be illustrated with the example shown in Fig. 5. The benefit in grouping cells A, B, and D into the same group is $\text{cost}_{\text{cou}}(\{A,B,D\}) = 1/(7+2+3+5+13+6+3+16+17+11+10+5) = 1/98$. Note that the coupling function only considers the outward traffic of a group. Similarly, we have the following values for different groupings for the cells in Fig. 2: $\text{cost}_{\text{cou}}(\{C,E,F\}) = 1/(7+2+3+5+19+21+15+12+11+3+22+6+7+9) = 1/142$, $\text{cost}_{\text{cou}}(\{A,C,E\}) = 1/(17+23+2+3+29+11+3+22+6+7+9+5+10+11) = 1/158$, and $\text{cost}_{\text{cou}}(\{B,D,F\}) = 1/(17+23+2+3+29+12+15+21+19+13+6+3+16+17) = 1/196$.

4.4.3. Cost-benefit function

The cost-benefit function is an integration of the cohesion function and the coupling function. It has the advantages of the two functions. It is defined as:

$$\text{benefit}_{c\text{-}b}(V_x) = \text{benefit}_{\text{coh}}(V_x) \times \text{cost}_{\text{cou}}(V_x)$$

$\text{benefit}_{c\text{-}b}(V_x)$ is the multiplication of $\text{benefit}_{\text{coh}}$ to cost_{cou} to balance the benefits of boundary crossings within a location area and amongst location areas. The determination of the values for $\text{benefit}_{\text{coh}}(V_x)$ and $\text{cost}_{\text{cou}}(V_x)$ can be done according to the formulas introduced in Sections 4.4.1 and 4.4.2.

The cost-benefit function may be illustrated using the example shown in Fig. 2. The benefit of

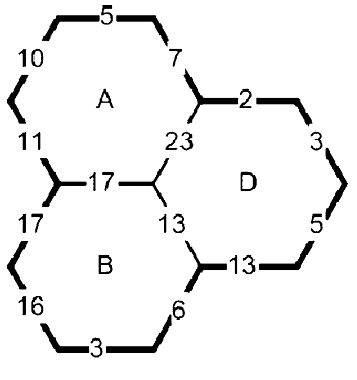

Fig. 5. The example of the coupling function.

Table 2
The location update cost of each configuration

	$\{\{A,B,D\},\{C,E,F\}\}$	$\{\{A,C,E\},\{B,D,F\}\}$
Cohesion	53 + 48 = 101	26 + 18 = 44
Coupling	1/98 + 1/142 = 0.017	1/158 + 1/196 = 0.011
Cost-benefit	0.541 + 0.338 = 0.879	0.165 + 0.092 = 0.256

grouping cells A, B, and D into the same group is $\text{benefit}_{c\text{-}b}(\{A,B,D\}) = \text{benefit}_{\text{coh}}(\{A,B,D\}) \times \text{cost}_{\text{cou}}(\{A,B,D\}) = 53/98 = 0.541$. Similarly, we have the following values for the cost-benefit functions for different groupings: $\text{benefit}_{c\text{-}b}(\{C,E,F\}) = \text{benefit}_{\text{coh}}(\{C,E,F\}) \times \text{cost}_{\text{cou}}(\{\text{C, E, F}\}) = 48/142 = 0.338$; $\text{benefit}_{c\text{-}b}(\{A,C,E\}) = \text{benefit}_{\text{coh}}(\{A,C,E\}) \times \text{cost}_{\text{cou}}(\{A,C,E\}) = 26/158 = 0.165$; $\text{benefit}_{c\text{-}b}(\{B,D,F\}) = \text{benefit}_{\text{coh}}(\{B,D,F\}) \times \text{cost}_{\text{cou}}(\{B,D,F\}) = 18/196 = 0.092$.

The three benefit functions introduced in this Section 4.4 are to group the cells into location areas to minimize the total cost in location updates. The rationale behind the cohesion function $\text{benefit}_{\text{coh}}$ is to have two cells in the same location area if the number of boundary crossings between them is high. The coupling function cost_{cou} is to group two adjacent cells into different location areas if the number of outward boundary crossings is small. The cost-benefit function $\text{benefit}_{c\text{-}b}$ is to balance the cell grouping benefits considered in the cohesion function $\text{benefit}_{\text{coh}}$ and the coupling function cost_{cou}. Based on the examples in Section 4.4, it has been shown that all benefit functions suggest that cells A, B and D should be in the same group since the traffics amongst them are heavy. However, in the other cases, the three cost functions might suggest different cell groupings. As a result, SCBLP using different benefit functions may result in different location area designs. The values of the benefit functions for the example in Fig. 2 are summarized in Table 2. The values of the benefit functions for the two groupings of the cells in the example are calculated. We use a pair of brackets to denote the cells in the same location area.

5. Hierarchical location database organization

In this section, we propose an algorithm which applies SCBLP to organize location databases into a hierarchical structure to minimize the total cost in location update cost and in location searching. In a hierarchical location database structure, each leaf node database maintains the location information of the mobile terminals within an LA or several LAs. We propose to organize the leaf nodes into groups (sub-trees) to minimize the cost in processing location updates. The grouping process is repeated until there is only one group left, and it is the root node of the location database tree. We call a node as an *internal node* if it is not a leaf node. If we group several nodes into a new group, we call the node that denotes the new group as the *parent node* of the nodes being grouped together, which are called *child nodes* of the parent node. The coverage area of an internal node is the union of the cover areas of all its child nodes.

The technical question, which we are interested in, is how to group the cells (location databases) to form a hierarchical location database tree with minimum processing cost. We extend the SCBLP algorithm iteratively to group the cells as described in the previous paragraphs. The hierarchical version of the SCBLP algorithm is called the H_SCBLP algorithm as shown in Fig. 6.

Steps 2–3: SCBLP derives a location area design as the input for H_SCBLP and the output of the location area design from SCBLP is a set of location areas. In the third statement, each location area is labeled from P_i to $P_{|SLAD|}$ where $|SLAD|$ is the size of the set *SLAD*.

Steps 4–6: Create a vertex for each element in *SLAD*. The set V' of vertices will be used to invoke H_SCBLP.

Steps 7–13: Construct E' and assign $m'(i,j)$ for all edges in E'. H_SCBLP creates an edge $e'_{i,j}$ in E' whenever the nodes which correspond to vertices $v'_i \neq v'_j$ in V' have a common edge between them. The weight $m'(i,j)$ of $e'_{i,j}$ is the sum of the crossing edges which connect nodes in v'_i and v'_j, i.e., P_i and P_j.

Step 14: Return a collection of *location areas* (i.e., a set of groups). Note that the result of each invocation creates a level in the hierarchy. The later invocations create upper levels of in the hierarchical tree. The first *location area* is derived from this invocation, and the others are produced by the remaining recursive calls.

```
Algorithm H_SCBLP
INPUT: G = (V, E, m)
OUTPUT: A set ofLocation Areas

1.  IF|V| = 1 THENRETURN∅
2.  SLAD = SCBLP(V, E, m)
3.  Label each element of SLAD from P1 to P|SLAD|
4.  FOR each Pi in SLAD
5.    Create a vertex v'i for Pi
6.    V' = V'∪ {v'i}
7.  FOR each v'i in V'
8.    FOR each v'jin V'
9.      IFv'i≠v'j
10.       Create an edge e'i,j which connects v'itov'j
11.       m'(i, j)= {sum(m(x, y)) for different vertex vx∈ Piand vy∈ Pj }
12.       IFm'(i, j) ≠ 0
13.         E' = E'∪ {ei,j}
14. RETURN {SLAD} ∪ H_SCBLP(V', E', m')
```

Fig. 6. The H_SCBLP algorithm.

6. Performance evaluation

6.1. Simulation model, parameters setting, and performance metrics

In this section, we report the performance evaluation of SCBLP and H_SCBLP when different benefit functions were adopted. We compared their performance with that of a random algorithm and a greedy algorithm. In the *random algorithm*, each location area initially contains only one cell. It then *randomly* merges neighboring location areas into LAs under the constraints of an LA, i.e., the maximum number of cells in an LA. Note that the random algorithm is easy in implementation and is served as the baseline method for performance comparison. The *greedy algorithm*, which is also called *hill-climbing* algorithm in [14] or LOAD-CMST in [2], is similar to the random algorithm. Initially each location area contains only one cell. In each iteration, two neighboring cells with the maximum number of boundary crossings are merged into a location area. Note that the heuristics adopted by the greedy algorithm are similar to the cohesion function introduced in Section 4.

In the experiments, the whole service area was partitioned into a number of interconnected cells. We used an $n \times n$ array to represent the cells in the service area. It is assumed that the cells had the same size. Each cell had six neighboring cells except the cells at the system boundary. In the experiments, we had studied the impacts of varying the number of neighboring cells on the performance of the algorithms. We modeled the movements of MTs by varying the number of cell boundary crossings using a random number generator. The number of cell boundary crossings for each pair of neighboring cells was generated randomly within the range of 4 and 36 following a uniform distribution.

The primary performance measure was the total number of location area boundary crossings per unit time in the whole system (U_T). A location update was generated whenever an MT entered into a new location area. Thus, U_T was an indicator of the number of location updates generated, and the location update cost was proportional to the value of U_T. A smaller value of U_T implied a better performance and a better location area design. In the experiments, we used the random algorithm as the baseline method. We defined the improvement percentage (improve_%) of an algorithm (e.g., SCBLP, H_SCBLP and the greedy algorithm) as the difference in U_T between the algorithm and the random algorithm:

$$\text{Improve\_\%}_{\text{SCBLP}} = (U_{\text{T,SCBLP}} - U_{\text{T,random}})/U_{\text{T,random}}.$$

We use the notations "*costBenefit*", "*cohesion*", and "*coupling*" to denote the uses of the cost-benefit, cohesion, and coupling functions, respectively, in SCBLP and H_SCBLP. Table 3 summarizes the model parameters and their baseline values. In order to improve the confidence level of the results, we repeated the simulation runs for the same setting using different random number seeds.

We had performed two sets of simulation experiments. The first set of experiments evaluated the 16 performance of SCBLP when different benefit functions were adopted as comparing with the greedy algorithm and the random algorithm (Section 6.2). In the experiments, we had studied the impacts of the important factors, i.e., the maximum number of cells in an LA, the maximum number of neighboring cells of a cell, the size of the system service area, and the bound of the weights for generating cell boundary crossings for each pair of neighboring cells, on the performance of SCBLP, the greedy algorithm, and the random algorithm. In addition, we had investigated the performance of the algorithms when the generation of boundary crossings was following the Weibull distribution [27] instead of a uniform distribution. The purpose was to simulate some hot cells where the traffic workloads were heavier. The second set of experiments studied the performance of H_SCBLP when it was applied to organize a hierarchical location database (Section 6.3).

Table 3
Parameters of experiments

Parameters	Baseline value
The maximum number of cells in an LA	6
The maximum number of neighboring cells of a cell	6
The size of the service area	30 cells × 30 cells
The weight distribution of cell boundary crossings for each pair of neighboring cells	[4–36] uniformly distributed
The number of simulation runs	200
Confidence level	±1% with 99.95% level of confidence

6.2. Performance evaluation of SCBLP

Figs. 7 and 8 shows the values of U_T (location update cost) for the five location area planning algorithms when the number of cells in the system are varied. As shown in Fig. 7, the SCBLP algorithms, combined with different benefit functions, significantly outperformed the random algorithm and the greedy algorithm. The improvement percentages are shown in Fig. 8, compared to the baseline method, i.e., the random algorithm. As shown in Fig. 8, although the performance of the greedy algorithm is better than the random algorithm, the degree of improvement is marginal. The improvement is only around 10% for different numbers of cells in the system. Amongst the three benefits functions, *costBenefit* gives the best performance, and the cohesion method has the least improvement. However, it still has an improvement of about 24% for different numbers of cells in the system. As shown in Fig. 8, the improvement percentage decreases slightly with an increase in the number of cells in the system. It is due to the decrease in the average number of neighboring cells of a cell when the system has more cells. If a cell is connected to a smaller number of neighboring cells, the performance difference among the algorithms is smaller as the choices for cell groupings are smaller.

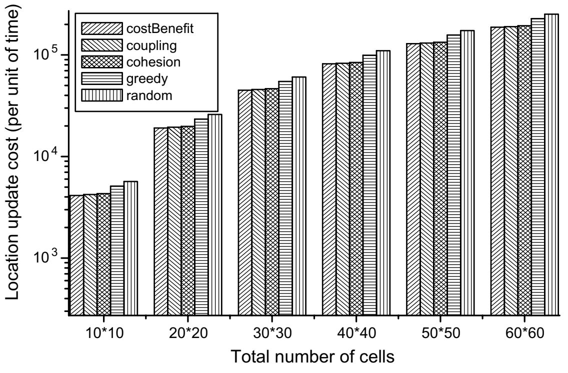

Fig. 7. U_T vs. different number of cells in the system.

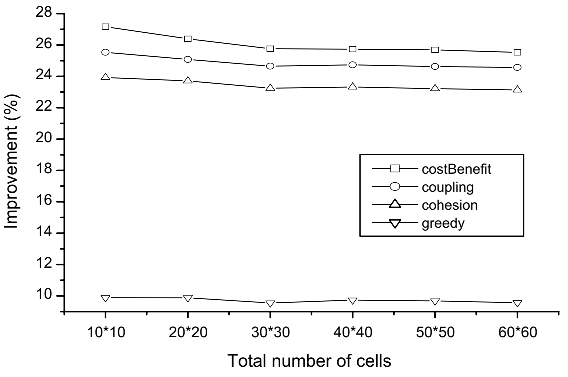

Fig. 8. Improve_% vs. number of cells in the system.

Fig. 9 depicts the performance results when the bounds for generating the weights of cell boundary crossings between neighboring cells are varied. As shown in Fig. 9, except the random algorithm, the value of U_T of the algorithms decreases with an increase in the value between the lower and upper bounds for generation of the weights. That is consistent with our expectation. If the range of the bounds is larger, the improvement due to a better grouping of cells will be higher. Similar to the results shown in Fig. 7, the performance of the SCBLP algorithms is significantly better than that of the random algorithm and the greedy algorithm as shown in Figs. 9 and 10. The performance of the greedy algorithm is consistently better than that of the random algorithm as shown in Figs. 9 and 10. Higher improvement percentages are obtained when the range between the bounds is larger. Again, *costBenefit* gives the best performance, especially when the range of the bounds is large. The improvement percentage of the *costBenefit* is about 30% when the bounds are 0–40.

Fig. 11 shows the impacts of the constraint on the maximum number of cells in an LA on the performance of the algorithms. If the maximum number of cells in an LA is larger, the improvement percentages of the SCBLP algorithms are greater. Consistent with the results shown in Figs. 8 and 10, *costBenefit* gives the best improvement especially when the number of cells in a location area is larger as shown in Fig. 11. The improvement percentage of *costBenefit* is close to 28%

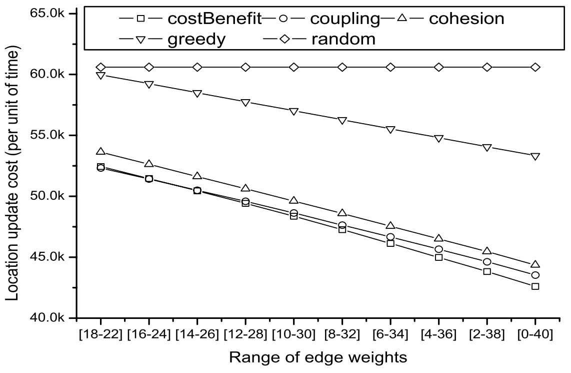

Fig. 9. U_T vs. different upper and lower bounds for weight generation.

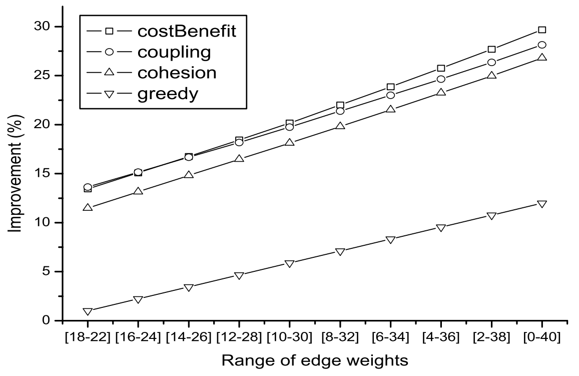

Fig. 10. Improve_% vs. different upper and lower bounds for weight generation.

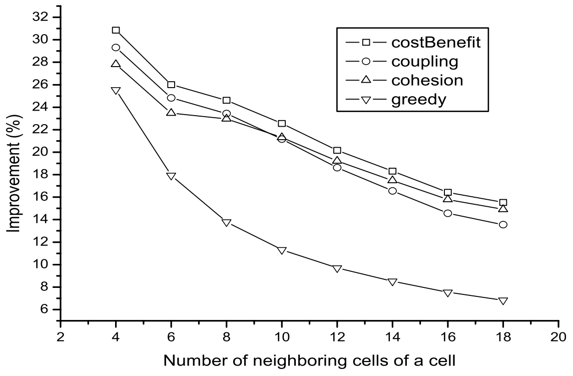

Fig. 12. Improve_% vs. number of neighboring cells.

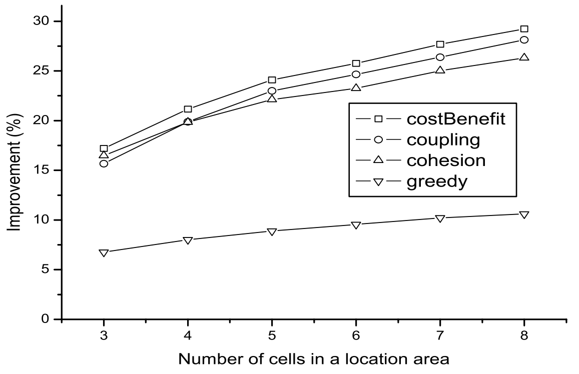

Fig. 11. Improve_% vs. maximum number of cells in an LA.

when the maximum number of cells in an LA is 8 while that of the greedy algorithm is only about 11%. The performance of *coupling* is 19 better than that of *cohesion*, especially when the maximum number of cells in an LA is larger. It is because when a location area is larger, the number of LA boundary crossings will be smaller.

Fig. 12 shows the performance results when the maximum number of neighboring cells of a cell is varied. The improvement percentage of the algorithms decreases with an increase in the number of neighboring cells. *costBenefit* still gives the best performance results when different number of neighboring cells are used. As shown in Fig. 12, the improvement percentage of the greedy algorithm drops rapidly with an increase in number of neighboring cells compared to the SCBLP algorithms. In [14], Wang et al. proposed a genetic algorithm for location area planning. The performance of the proposed algorithm [14] was compared with the *greedy* algorithm which was also tested in this paper. It was shown in [14] that the improvement of the proposed algorithm was only 10% when the number of cells was 91. Under the similar constraints (the size of location area, the variance of cell boundary crossing, and the coverage area size), we found that the improvement of our proposed *costBenefit* can be close to 20%, compared to the results in [14].

Another important factor on the performance of the SCBLP algorithms is the distribution of the traffic workload between neighboring cells. Figs. 13 and 14 show the results when the traffic workloads between neighboring cells are following the Weibull distribution [27]. Compared to uniform distribution, Weibull distribution has a higher probability in generating extreme values (extreme values mean hot-spots). The scale and the shape parameters of Weibull distribution are 20 and 1, respectively, and we set the mean value to be 20. As shown in Figs. 13 and 14, the improvement ratios comparing to the random algorithm are higher comparing to the results from a uniform distribution of traffic workloads (Figs. 7 and 8). *costBenefit* is still the best algorithm and its improvement is up to 40%. The trend of the results of the algorithms is similar to the results showed in Figs. 7 and 8. We conjecture that if the system contains more hot-spots (a cell with a heavy traf-

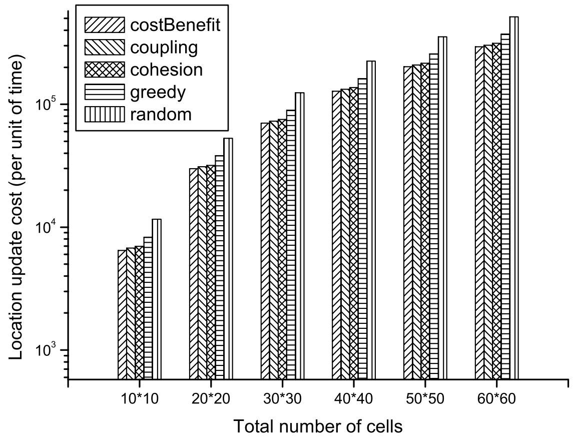

Fig. 13. U_T vs. different number of cells in the system (Weibull distribution).

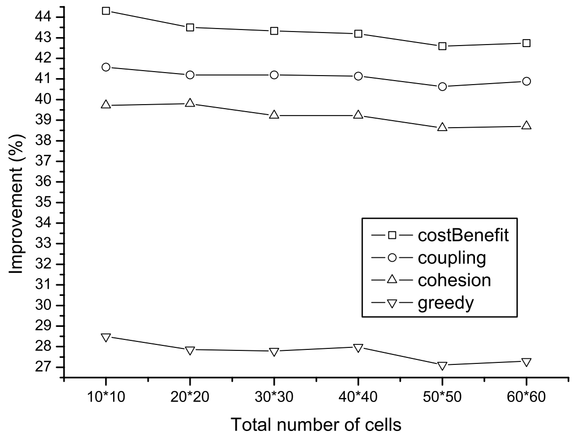

Fig. 14. Improve_% vs. number of cells in the system (Weibull distribution).

fic), the improvement will be higher using the proposed algorithm, SCBLP.

6.3. Performance evaluation of H_SCBLP

The purpose of this section is to compare the performance of H_SCBLP to that of the H_random and H_greedy algorithms in organizing a hierarchical location database with different benefit functions. In the experiments, the number of cells was set to be 3600 (60 × 60). Each internal node had up to six child nodes following the grouping policy of child nodes for the H_SCBLP algorithm defined in Section 5. The grouping policies for the H_greedy algorithm and the H_random algorithm were similar to that of H_SCBLP. The major difference was that H_SCBLP invoked SCBLP to derive a location area design in Step 2, and H_greedy and H_random greedily and randomly group child nodes, respectively. As a result, H_SCBLP, H_greedy, and H_random may have different number of neighboring nodes at different levels in a hierarchical database structure.

Two major performance metrics were considered. The first metric was the total number of location area boundary crossings per unit time in each level of a hierarchy. The second metric was the height of a hierarchical location database tree because it affected the maximum propagation delay in searching the hierarchical location database tree. Figs. 15 and 16 show the values of U_T and the improve_% of the algorithms for updating the databases at different levels in the hierarchical database tree. Since *coupling* and *costBenefit* tend to have a hierarchy with a smaller height, the results for *coupling* (and *costBenefit*) are only up to level 5.

As shown in Figs. 15 and 16, *costBenefit* is the best algorithm when updates are installed at lower levels. *Coupling* is slightly better than *costBenefit* when updates are installed at levels higher than 4. The performance improvement of *costBenefit* and *coupling* comparing with H_random is significant for updates installing at all levels in a hierarchical

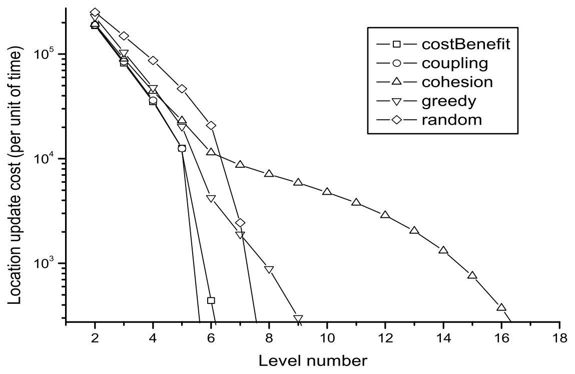

Fig. 15. U_T of the algorithms when different levels of hierarchy are used.

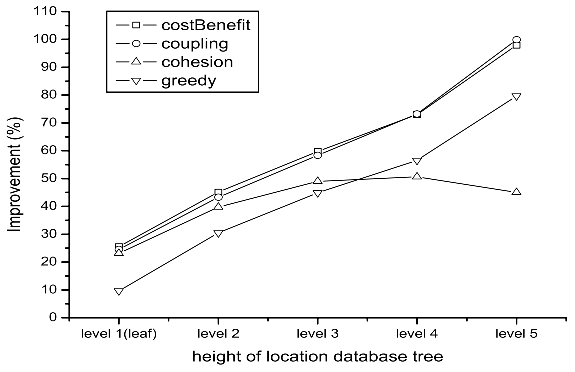

Fig. 16. Improve_% of the algorithms when different levels of hierarchy are used.

database structure. We must point out that when the number of internal nodes is small, a brute force approach may be used to obtain an optimal solution. In the experiments, we did not take any brute force approach for the algorithms in order to have a fair comparison among them.

Fig. 17 shows that *coupling* and *costBenefit* can build up hierarchies with no more than six levels for most cases. Hierarchies may grow up to more than seven levels, when greedy, random, and *cohesion* are used. Since Figs. 15 and 16 have shown that no algorithm can outperform any other algorithm for all cases, we surmise that the best strategy is to adopt *costBenefit* in constructing internal nodes at lower levels of a location database tree. Brute force algorithm can then be taken

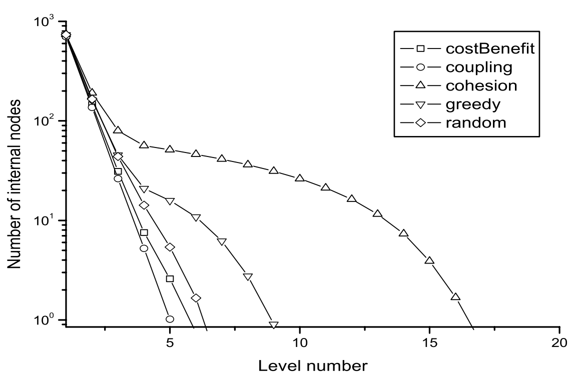

Fig. 17. Number of internal nodes in each level of the hierarchy.

to have an optimal grouping quality of nodes at higher levels, i.e., close to the root.

7. Conclusions

The number of personal communication service (PCS) subscribers is increasing very rapidly. How to resolve the traffics and performance problems for mobility management becomes a critical issue. The solutions should not just rely only on deploying more base stations. Instead, the proposal of an intelligent way for mobility management is of paramount importance. In this paper, we propose an algorithm for location area planning (LAP) to minimize the total number of location updates. We consider the mobility patterns of mobile terminals in deciding how to group the cells into location areas and how to organize the location areas into a hierarchical location database structure to further reducing the update processing and searching costs. Different from the previous works in the area, we model the location area planning (LAP) problem as a set-covering-problem. The main advantage of using a set-covering approach is on the efficiency of the approach for applications that need a dynamic adaptability to the mobility patterns of mobile terminals. We propose an efficient algorithm for location update planning called set-covering-based location area planning (SCBLP), and three benefit functions are proposed for determining how to group the cells in SCBLP. We also have extended the SCBLP algorithm for a hierarchical location database structure. Simulation results have shown that our proposed algorithms can significantly reduce the PCS network traffics and the mobility management cost, compared to the greedy algorithm and the random algorithm.

For the future work, we will derive approximation bounds for various benefit functions to provide guidelines for location area planning. We shall further explore set-covering approaches in the handling of the migrations between hierarchical structures so that the system could be quickly adaptive to the mobility patterns of mobile terminals.

Acknowledgement

The work described in this paper was partially supported by a grant from CityU (Project No. 7001259).

References

[1] E. Pitoura, I. Fudos, An efficient hierarchical scheme for locating highly mobile users, in: Proceedings of the 6th ACM International Conference on Information and Knowledge Management (CIKM98), November 1998, pp. 218–225.

[2] A. Gamst, Application of graph theoretical methods to GSM radio network planning, in: Proceedings of 1991 IEEE International Symposium on Circuits and Systems, pp. 942–945.

[3] C.Y. Lee, H.G. Kang, Cell planning with capacity expansion in mobile communications: a tabu search approach, IEEE Transactions on Vehicular Technology 49 (5) (2000) 1678–1691.

[4] P. Demestichas, E. Tzifa, V. Demesticha, N. Georgantas, G. Kotsakis, M. Kilanioti, M. Striki, M.E. Anagnostou, M.E. Theologou, Control of the location update and paging signaling load in cellular systems by means of planning tools, in: Proceedings of 1999 Vehicular Technology Conference, vol. 4, 1999, pp. 2119–2123.

[5] K. Tutschku, Demand-based radio network planning of cellular mobile communication systems, in: Proceedings of 1998 INFOCOM, 1998, pp. 1054–1061.

[6] Y. Bejerano, I. Cidon, Efficient location management based on moving location areas, in: Proceedings of 2001 INFOCOM, 2001, pp. 3–12.

[7] J.G. Markoulidakis, G.L. Lyberopoulos, D.F. Tsirkas, E.D. Sykas, Evaluation of location area planning scenarios in future mobile telecommunication systems, Wireless Networks 1 (1) (1995) 17–29.

[8] P.R.L. Gondim, Genetic algorithms and the location area partitioning problem in cellular networks, in: Proceedings of IEEE Vehicular Technology Conference, vol. 3, 1996, pp. 1835–1838.

[9] T.H. Corman, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., The MIT Press and McGraw-Hill, Cambridge, Massachusetts and New York, 2001.

[10] I. Demirkol, C. Ersoy, M.U. Caglayan, H. Delic, Location area planning in cellular networks using simulated annealing, in: Proceedings of IEEE INFOCOM'01, Anchorage, Alaska, April 2001.

[11] S.Z. Ali, Location management in cellular mobile radio networks, in: Proceedings of 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, vol. 2, 2002, pp. 745–749.

[12] G. Varsampoulos, S.K.S. Gupta, On dynamically adapting registration areas to user mobility patterns in PCS networks, in: Proceedings of 1999 International Workshop on Parallel Processing, 1999, pp. 108–113.

[13] M. Munguia-Macario, D. Munoz-Rodriguez, C. Molina, Optimal adaptive location area design and inactive location areas, in: Proceedings of 1997 Vehicular Technology Conference, 1997, pp. 510–514.

[14] T.-P. Wang, S.-Y. Hwang, C.-C. Tseng, Registration area planning for PCS networks using genetic algorithms, IEEE Transactions on Vehicular Technology 47 (3) (1998) 987–995.

[15] J. Plehn, The design of location areas in a GSM-network, in: Proceedings of 1995 Vehicular Technology Conference, 1995, pp. 871–875.

[16] S. Ceria, P. Nobili, A. Sassano, Solving large-scale sparse semi-definite programs for combinatorial optimization, SIAM Journal on Optimization 10 (2000) 443–446.

[17] G.-H. Li, K.-Y. Lam, T.-W. Kuo, S.-W. Lo, Location management in cellular mobile computing systems with dynamic hierarchical location databases, Journal of Systems and Software 69 (1–2) (2004) 159–171.

[18] V.V. Vazirani, Approximation Algorithms, Springer, New York, 2001.

[19] Taiwan Cellular Corp. Available from <http://www.twngsm.com.tw/company/mind/index.htm>.

[20] E. Pitoura, G. Samaras, Locating objects in mobile computing, IEEE Transactions on Knowledge and Data Engineering 13 (4) (2001) 571–592.

[21] I.F. Akyildiz, J. McNair, J.S.M. Ho, H. Uzunalio glu, W. Wang, Mobility management in current and future communication networks, IEEE Network 12 (4) (1998) 39–49.

[22] R. Jain, Reducing tra.c impacts of PCS using hierarchical user location databases, in: Proceedings of IEEE International Conference in Communications, vol. 2, 1996, pp. 1153–1157.

[23] C. Eynard, M. Lenti, A. Lombardo, O. Marengo, S. Palazzo, A methodology for the performance evaluation of data query strategies in universal mobile telecommunication systems (UMTS), IEEE Journal Selected Areas in Communications 13 (5) (1995) 893–907.

[24] M. Vudali, The location area design problem in cellular and personal communications systems, in: Proceedings of 5th IEEE International Conference on Universal Personal Communications, vol. 2, 1996, pp. 591–595.

[25] D. Saha, A. Mukherjee, P.S. Bhattacharya, Mobility analysis of personal communication services (PCS) networks for optimum location area design, in: Proceedings of 1997 IEEE Region 10 Annual Conference TENCON '97, Brisbane, Australia, December 1997.

[26] A. Pal, D.S. Khati, Dynamic location management with variable size location areas, in: Proceedings of the 2001 International Conference on Computer Networks and Mobile Computing, Beizing, October 2001, pp. 73–78.

[27] N. Johnson, S. Kotz, N. Balakrishnan, Continuous Univariate Distributions, vol. 1–2, Wiley, New York, 1994.

Shi-Wu Lo received the BSCS and MSCS degrees from Yuan-Ze University and National Chung-Cheng University, respectively. He is currently a Ph.D. candidate at the Department of Computer Science and Information Engineering, National Taiwan University. He has done various research work on system design and development, including workload scheduling on disk arrays, flash memory management, location management for mobile services, and real-time scheduling problems over simultaneous-multithreaded (SMT) processors.

Tei-Wei Kuo received his BSE degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, ROC, in 1986. He received the MS and PhD degrees in computer sciences from the University of Texas at Austin in 1990 and 1994, respectively. He is currently a Professor in the Department of Computer Science and Information Engineering of the National Taiwan University, Taipei, Taiwan, ROC. He was an Associate Professor in the Department of Computer Science and Information Engineering of the National Chung Cheng University, Taiwan, ROC, from August 1994 to July 2000.

The research interest of Professor Kuo includes real-time process scheduling, real-time operating systems, embedded systems, and real-time databases. He was the Program Co-Chair of IEEE 7th Real-Time Technology and Applications Symposium, 2001, and an associate editor of the Journal of Real-Time Systems since 1998. He has consulted for government and industry on problems in various real-time and embedded systems design and received several research awards, including the Distinguished Research Award from the Taiwan National Science Council in 2003, from the government, research institutes, and foundations. Dr. Kuo is a senior member of IEEE.

Kam-Yiu Lam received the B.Sc. (Hons) degree in Computer Studies with distinction and Ph.D. degree from the City University of Hong Kong in 1990 and 1994, respectively. He is currently an Associate Professor in the Department of Computer Science, City University of Hong Kong. He has served as a program committee member and reviewer for conferences on real-time systems, mobile computing and databases. He is also a reviewer for international journals including the IEEE Transactions on Computers, IEEE Transactions on Data and Knowledge Engineering, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Software Engineering, IEEE Transactions on Vehicular Technology, Computer Networks, Real-Time Systems, Information Systems, IEEE Multimedia, Journal of Real-Time Systems, The Computer Journal, Information Processing Letters and the Journal of Systems and Software. His current research interests are real-time database systems, real-time active database systems, location-dependent continuous query processing, mobile computing and distributed multimedia systems. He was the Guest Editor of the Journal of Systems and Software on the special issue of Real-time Active Database Systems. Dr. Lam is a member of the IEEE.

Guo-hui Li was born in April 1973. He graduated from Huazhong University of Science & Technology (HUST) in China and got his Ph.D. degree in 1999. Currently he is an associate professor in the School of Computer Science & Technology in HUST. His main research interests include mobile computing, real-time database systems, active database systems and transaction processing systems.