Efficient Location Area Planning for Personal Communication Systems

Yigal Bejerano* Nicole Immorlica† Joseph (Seffi) Naor‡ Mark Smith*

ABSTRACT

A central problem in personal communication systems is to optimize bandwidth usage, while providing Quality of Service (QoS) guarantees to mobile users. Network mobility management, and in particular, location management, consumes a significant portion of bandwidth, which is a necessary overhead for supporting mobile users. We focus our efforts on minimizing this overhead. Unlike previous works, we concentrate on optimizing existing schemes, and so the algorithms we present are easily incorporated into current networks. We present the first polynomial time approximation algorithms for minimum bandwidth location management. In planar graphs, our algorithm provably generates a solution that uses no more than a constant factor more bandwidth than the optimal solution. In general graphs, our algorithm provably generates a solution that uses just a factor $O(\log n)$ more bandwidth than optimal where n is the number of base stations in the network. We show that, in practice, our algorithm produces near-optimal results and outperforms other schemes that are described in the literature. For the important case of the line graph, we present a polynomial-time optimal algorithm. Finally, we illustrate that our algorithm can also be used for optimizing the handoff mechanism.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations — Network management; G.2.2 [**Discrete Mathematics**]: Graph Theory — Graph algorithms

General Terms

Algorithms, Design, Management, Performance, Theory

Keywords

Personal Communication System, Cellular Systems, Mobility Management, Location Area Planning, Approximation Algorithms.

1. INTRODUCTION

Personal Communication Service (PCS) networks enable people to communicate independently of their location and while they are moving. To provide this capability, each PCS network is equipped with a *mobility management* mechanism that consists of two major components. The *location management* component maps subscriber numbers to the current locations of the corresponding users when calls arrive. The *handoff management* component maintains ongoing calls with ensured quality of service (QoS) while their end-users change their attachment points. The current standards for location management in present PCS networks such as GSM [1], IS-41 [2] and UMTS [3] use similar schemes for mobility management. The basic idea for these schemes is as follows. The coverage area of the system is divided into *location areas* (LAs), where each LA consists of a group of cells that forms a continuous geographic area. The cells of one or few LAs are associated with a single mobile switching center (MSC) that connects them to a fixed infrastructure. The system keeps the LA identifier of the location for each user. When a mobile user crosses an LA boundary, the user updates the system with its new location. In this way, the system is able to maintain the current location of each user. When a call comes in for a particular user, the system simultaneously pages the mobile user in all the cells of the user's recorded current LA. The called user replies to this paging message and the system establishes a connection between the originator of the call and the called user. In current systems, LAs are determined in advance based on static movement probabilities and, in general, remain unchanged.

In recent years, PCS networks have faced rapid increase in the number of mobile users. The main solution for supporting the growing population is to reduce the cell sizes and to increase bandwidth reuse [4, 5]. These changes cause the number of call deliveries, location update operations, and handoff operations to increase dramatically and result in high loads on mobility management mechanisms. Moreover, frequent handoff operations affect the user's QoS perception, especially when moving from one MSC-area to another

*Y. Bejerano and M. Smith are with Bell Labs, Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974.
†N. Immorlica is with the Laboratory for Computer Science, MIT, NE43-334, 200 Technology Square, Cambridge, MA 02139. Part of this work was done while visiting Bell Labs.
‡J. Naor is with the Computer Science Department, Technion - Israel Institute of Technology, Haifa 32000, Israel. Part of this work was done while visiting Bell Labs.

MobiCom'03, September 14–19, 2003, San Diego, California, USA.
Copyright 2003 ACM 1-58113-753-2/03/0009 ...$5.00.

where inter-MSC handoff operations are performed. This motivates extensive research efforts for reducing the overhead of the mobility management mechanisms from both the network resources and wireless bandwidth perspectives. These efforts have lead to the development of numerous new mobility management schemes on one hand and new clustering algorithms for optimizing the LA planning (LAP) on the other hand.

1.1 Related Work

There are a variety of new location management schemes in the literature. Several proposals introduce incremental improvements to the current LA approach. For instance, overlapping LA systems were introduced to reduce the number of update operations that result from users moving near the LA boundaries [6],[7]. Sequential paging methods were suggested to reduce the amount of paging induced by an incoming call. In sequential paging, the system sequentially pages subareas of the current LA of a called-user based on its location probability [8],[9], [10]. Both updating and paging costs can be reduced by using a system of moving LAs. This approach can be useful in the case of tracking fast moving users on highways [11].

Other work in this area consists of dynamic mobility management schemes that are based on users' profile information. Since the latter are customized for the individual mobility patterns of each user, they are more efficient than generalized schemes for all users. In this category we find proposals like the personal LA approach, in which the LA sizes are defined on a per-user basis for reducing the signaling overhead of each individual user [12],[13], [14],[15]. In [16], [17], [18],[19], the users are allowed to skip some update operations when they cross the LAs' boundaries. When a call comes in, the system uses the user profile information to estimate the probability for each LA that it is the current LA of the user. The system then pages the LAs in order of decreasing probability. In other schemes, there is no notion of location areas, and the mobile users perform update operations based on either the elapsed time [20], [21], number of crossed cells [20], [22] or the traveled distance since the previous update operation [20], [23]. The selected thresholds for performing update operations are adapted to the individual mobile user mobility patterns and communication traffic. Other schemes [24], [25], [26], [14], also consider the user speed and the trajectory during its last update for predicting its location when needed. Comprehensive surveys of location management schemes can be found in [2],[27]. While the new schemes, especially the profile-based methods, offer better utilization of wireless network resources than the standard LA-approach, they suffer from some inherent deficiencies. They tend to add significant complexity to the management of the networks. The network is required to keep track of significantly more information per user and each mobile user needs to keep track of its mobility patterns. Moreover, they require modification of the current standards and so require changes of the wireless network infrastructures as well as updating all the handsets. This complicates the incorporation of recent schemes into the current infrastructures.

Other research directions include developing new partitioning algorithms that compute efficient LA plans. These LA plans are evaluated by their total signaling costs, *i.e.,* the cost of all the induced paging and update operations in the system. Since the paging cost depends on the number of cells that are paged when calls come in, while the update cost results from LA boundary crossings, there is a clear trade-off between the two costs. By selecting large LAs, the number of LA boundary crossings and hence the system update cost is reduced, while selecting small LAs produces low paging cost. The LA planning methods can be classified into two categories. The papers in the first category [28], [29], [30],[31] assume uniform user distribution and intercell movement rate, and with these strong assumptions they derive an optimal LA planning. However, these results are not applicable to most practical cases where the network usage is heterogeneous [30, 32]. In the second category, the network is modeled as a graph where each node represents a cell, its weight specifies the cell population and there is an edge between every pair of adjacent nodes that defines the user movement rate between the corresponding cells. The LA planning is mapped to a graph partitioning problem. Since this partitioning problem is NP-hard, different heuristics are proposed for obtaining efficient solutions. Most of the proposed schemes limit the LA size in order to bound the LA paging cost and seek heuristics for LA planning that minimize the system overall update cost. These heuristics employ different algorithmic tools like genetic algorithms [33],[34], taboo search [34] and simulated annealing [34],[32], [35]. In [35] the authors provide a comprehensive problem formulation that also consider size and connectivity constraints, enabling network designers to represent some practical constraints of a cellular network. Other papers [36],[37], [38] address the LA planning problem from a graph theoretic approach. In [36] the author presents an algorithm that contains two phases. A merge phase in which the cells are merged for constructing a collection of LAs, and an exchange phase in which cells are exchanged between the LAs for reducing the update cost. The algorithm in [37] uses planar graph bisection (partitioning into two equal parts), and in [38] the partitioning algorithm is based on the constrained maximal spanning tree algorithm. In all the papers mentioned, simulations constitute the main tool for evaluating the quality of the solutions produced by the algorithms, and no worst-case guarantees are presented (e.g., as compared to an optimal solution). In [39] the authors address the one-dimensional LA planning problem for covering highways and railroads. Since highways and railroads carry a major portion of user traffic, an LA planning that efficiently covers them can significantly reduce the signaling overhead of the entire system. This work presents a heuristic which constructs the optimal LA planning in the case of homogeneous traffic and user density, but yields suboptimal results in general.

1.2 Our Results

We concentrate our efforts on optimizing existing mobility management schemes and we present a novel algorithm for location area planning (LAP). Such planning can be used for both efficient location management as well as handoff management. Since the second can be viewed as the special case of the first, throughout the paper we mainly consider efficient LAP for location management. Then, we present a practical design for optimizing both the location and the handoff mechanisms.

We cast the LAP problem in graph-theoretic terms and then define several algorithms that attempt to minimize the update and paging costs subject to a variety of system-imposed constraints like maximum LA size. In the case of one-dimensional networks such as highways and railroads, we present a polynomial time algorithm that finds the optimal LA planning even for general traffic and user density patterns. In general networks, we show that it is computationally difficult to find the LA planning with the lowest update and paging costs. We formulate the problem in general graphs as a linear program and use an optimal (fractional) solution to this linear program as a lower bound on the optimal (integral) solution to the LA planning problem. Our approximation algorithms round an optimal fractional solution to the linear program into an approximate integral solution. Specifically, we provide a polynomial time algorithm that computes an LA planning whose cost is no more than $O(\log n)$ times the optimal cost (where n is the number of base stations). In the special case of planar networks (*i.e.,* networks in which users only move between adjacent base stations), a highly practical instance, we can find in polynomial time an LA planning with cost at most a constant times the optimal cost. Unlike existing heuristic-based schemes, our algorithms have the advantage that (i) they provide a worst-case guarantee on the performance; (ii) we can use the optimal fractional solution to bound the actual performance of the algorithms on any problem instance.

We use the results of the approximation algorithm as the basis for a simple heuristic that further improves the LA planning solution. When we run our algorithms on a realistic network, we observe that their solutions cost just 6% to 71% more than the optimal fractional solution of the linear program, a lower bound on the optimal integral solution. Note that this lower bound may be significantly lower than the actual optimal solution for the LA planning problem. Consequently, we conclude that, in practice, our algorithm finds solutions that are very close to the optimum. Moreover, by simulations we show that the proposed algorithm outperforms other LAP schemes described in the literature.

This work is organized as follows. Section 2 presents the network model and provides a formal definition of the LA planning problem. Section 3 introduces a scheme for finding the optimal LA planning for linear graphs, and Section 4 presents approximation algorithms for general graphs and for planar graphs. Section 5 introduces an efficient heuristic for improving the performance of the proposed approximation scheme, and we address the optimization of the hand-off mechanism in Section 6. Section 7 evaluates the performance of our algorithms by simulations and concluding remarks are given in Section 8.

2. MODEL AND PROBLEM STATEMENT

In this section we present the network model and provide a formal definition of the Location Area Planning (LAP) problem. We also address the question whether a given instance of the problem has a feasible solution at all and we prove that finding an optimal one is NP-hard even when the given graph topology is planar or a star.

2.1 Network Model

We represent a cellular network by a graph $G(V, E)$, where $|V| = n$ and $|E| = m$. Each cell in the cellular network is represented by a node with a unique identification number $i \in [1..n]$ and a weight, w_i, that reflects the cell user population[1]. The edges denote cell adjacency, and every edge $(i, j) \in E$ has a weight f_{ij} that specifies the user traffic (flow) between its end-points (in both directions) during a time unit. For completeness, let $f_{ij} = 0$ for every pair of nodes $i, j \in V$ such that $(i, j) \notin E$.

2.2 The Problem Statement

For location management, the nodes are clustered into disjoint sets called *locations areas* (LAs), $\mathcal{L} = \{S_1, \cdots S_k\}$, which contain all the graph nodes. The LA containing node i is denoted by $LA(i)$. For each mobile user, the system keeps a record of the current LA where it resides. Each time a user crosses an LA boundary, it updates the system with its new location. When an incoming call arrives, the system simultaneously pages the user in all the cells of its current LA. After receiving the user reply, the system establishes a connection between the call originator and the called user. Thus, the location management scheme produces two types of signaling costs, where the costs are considered from both the wireless and wired network perspective. An *update cost* that reflects the cost of all the update operations performed by the users during a time unit, and a *paging cost* that results from all the paging operations during a time unit. To define these costs, we use the following notation. Consider an LA planning $\mathcal{L} = \{S_1, \cdots S_k\}$. Let λ be a user incoming call rate and C_p be the cost of paging a single cell. The paging cost of a single LA, $S \in \mathcal{L}$, with $|S|$ cells, is the product of the incoming call rate $\lambda \cdot \sum_{i \in S} w_i$, times the cost of paging all the LA cells $C_p \cdot |S|$. Consequently, $Page\_Cost(S) = \lambda \cdot C_p \cdot |S| \cdot \sum_{i \in S} w_i$, and the overall paging cost is,

$$Page\_Cost(\mathcal{L}) = \lambda \cdot C_p \cdot \sum_{S \in \mathcal{L}} |S| \cdot \sum_{i \in S} w_i$$

Similarly, we define the update cost. Recall that a user induces an update operation whenever it moves from a cell i in one LA to a cell j in a another LA. The amount of traffic per time unit between the two cells is f_{ij}, so the update cost caused by traffic between the cells i and j is $C_u \cdot f_{ij}$ where C_u is the cost of a single update operation. In our calculations we charge each of the nodes j and i for half of this update cost. Therefore, the cost of update operations in the system as a whole is simply the amount of traffic between the LAs times C_u. The total update cost is,

$$Update\_Cost(\mathcal{L}) = \frac{1}{2} \cdot C_u \cdot \sum_{i \in V} \sum_{j \notin LA(i)} f_{ij}.$$

Since the efficiency of a given LA planning, $\mathcal{L}$, is determined by its overall signaling cost, an *optimal LA planning*, $\mathcal{L}_{OPT}$, is a graph partition with the minimal signaling cost among all the feasible LA plans, *i.e.,* ,

$$Cost(\mathcal{L}_{OPT}) = \min_{\mathcal{L}} \{Update\_Cost(\mathcal{L}) + Page\_Cost(\mathcal{L})\}.$$

We note that in actual cellular systems, not every LA planning is feasible. Geographical considerations and network infrastructure may impose certain size and connectivity constraints on the system. For instance, as all the cells of an LA are connected to a single mobile switching center (MSC), neither an LA size nor its total population should exceed the MSC capacities. We enforce these *size constraints*

[1] Practically, the node's weight represents the average user population during the rush hours.

by introducing two bounds $K_{\max}$ and $W_{\max}$ on the maximal cell number and maximal population size of an LA, respectively, *i.e.,* for every $S \in \mathcal{L}$ we require that $|S| \leq K_{\max}$ and $\sum_{i \in S} w_i \leq W_{\max}$. In fact, our schemes can easily deal with more general constraints that bound, for each vertex $i \in V$, the size and weight of the LA containing i, i.e., $|LA(i)| \leq K_i$ and $\sum_{j \in LA(i)} w_j \leq W_i$, as we show later.

As a result of topological considerations, it is also possible that certain cells must reside in the same LA or other cells must reside in separate LAs. These *connectivity constraints* are define by constants b_{ij} for every pair of cells $i, j \in V$, such that

$$b_{ij} = \begin{cases} 1 & \text{If } i \text{ and } j \text{ must be in different LAs.} \\ -1 & \text{If } i \text{ and } j \text{ must be in the same LAs.} \\ 0 & \text{Otherwise.} \end{cases}$$

These constants can be represented by a connectivity matrix $B = \{b_{ij}\}$. Any LA planning $\mathcal{L}$ that satisfies both the size and the connectivity constraints is called a *feasible LA planning* and the LAP problem is defined as follows.

Definition 1 (Location Area Planning Problem). *Given a graph $G(V, E)$ with weights w_i and f_{ij} for every node $i \in V$ and edge $(i, j) \in E$, LA size bounds $K_{\max}$, $W_{\max}$ and connectivity matrix B, find an LA planning $\mathcal{L}$ such that,*

$$Cost(\mathcal{L}) = \min \left\{ \begin{array}{c} \frac{1}{2} \cdot \sum_{i \in V} \sum_{j \notin LA(i)} f_{ij} \ + \\ \lambda \cdot C_p \cdot \sum_{S \in \mathcal{L}} (|S| \cdot \sum_{i \in S} w_i) \end{array} \right\}$$

subject to:

$$\begin{array}{rc} \forall S_1, S_2 \in \mathcal{L}: & S_1 \bigcap S_2 = \emptyset \\ \forall i \in V: & 1 \leq |LA(i)| \leq K_{\max} \\ \forall i \in V: & \sum_{j \in LA(i)} w_j \leq W_{\max} \\ \forall i, j \in V, b_{ij} = 1: & LA(i) \neq LA(j) \\ \forall i, j \in V, b_{ij} = -1: & LA(i) = LA(j) \end{array}$$

Note that the proposed problem seeks an LA planning that simultaneously minimizes both the update and search costs. Unlike standard partitioning problems, our minimization objective does not include just the cost of the cut separating the LAs, but also includes a cost dependent on the size and the weight of the components. Thus, our problem is fundamentally different from standard partitioning problems, and requires new algorithms to solve it.

2.3 Hardness of the LAP Problem

Theorem 1. *The LAP Problem is NP-hard, even when the given instance $G(V, E)$ is a star.*

Proof. We prove this theorem by presenting a polynomial reduction from the *partition* problem [40] to the LAP problem. Consider a set $\mathcal{A}$ of $m > 2$ elements where each element $a_i \in \mathcal{A}$ has size $s_i \in \mathcal{Z}^+$, and let $X = \sum_{a_i \in \mathcal{A}} s_i/2$. The partition problem looks for a subset $\mathcal{A}' \subset \mathcal{A}$ such that $\sum_{a_i \in \mathcal{A}'} s_i = \sum_{a_i \in \mathcal{A} - \mathcal{A}'} s_i = X$. Our reduction constructs a star graph $G(V, E)$ that contains the following nodes and edges. A hub node h whose weight is $w_h = 0$. For every element $a_i \in \mathcal{A}$, we define a node i adjacent to the hub node that satisfies $w_i = s_i$ and $f_{ih} = s_i$. Let $C_u = 2mX$ and let $C_p \cdot \lambda = 1$. We impose a constraint $W_{\max} = X$ on the maximal weight of an LA. Intuitively, due to the high update cost, the optimal LA planning of this graph is obtained when the LA that contains node h, $LA(h)$, contains as many other nodes as possible without violating the weight constraints.

We claim that there is a subset $\mathcal{A}' \subset \mathcal{A}$ with $\sum_{a_i \in \mathcal{A}'} s_i = X$ if and only if there is an LA planning $\mathcal{L}$ with cost less then $2mX(X+1)$. Recall that the paging cost of the system is at most $2mX$, which happens when all the nodes are included in a single LA. Suppose that there is such a partition $\mathcal{A}'$. Then we construct an LA planning $\mathcal{L}$ with two LAs. The first LA, denoted by S_1, contains node h and all the nodes $i, \ \ i \in \mathcal{A}'$, while the second LA, denoted by S_2 contains all other nodes. Since $\sum_{a_i \in \mathcal{A}'} s_i = \sum_{a_i \in \mathcal{A} - \mathcal{A}'} s_i = X$, the two LAs satisfy the weight constraints. The update cost of this LA planning comes only from the separation of node h from the nodes in S_2. Thus, the update cost is $2mX \cdot \sum_{a_i \in \mathcal{A} - \mathcal{A}'} s_i = 2mX^2$, and therefore the cost of this LA planning is no more than $2mX(X+1)$ as claimed.

We assume now that there is a feasible LA planning $\mathcal{L}$ such that its cost is at most $2mX(X+1)$. Since the update cost must be an integer multiple of $2mX$, the update cost of $\mathcal{L}$ is at most $2mX^2$. Thus, the total weight of all the nodes that are not included in $LA(h)$ is at most X. Since the total weight of every LA is at most X, the total weight of all the nodes in $LA(h)$ must be X, yielding that the total size of all the elements in the set $\mathcal{A}' = \{a_i | i \in LA(h)\}$ is X. This completes the proof. □

Since the LAP problem is NP-hard, rather than finding optimal solutions, we construct a polynomial-time approximation algorithm. Such an algorithm has an approximation factor α if, for every instance of the LA problem, it finds a solution whose cost is at most α time the cost of an optimal solution.

3. OPTIMAL SOLUTION FOR A LINE

In the following, we present an efficient algorithm based on dynamic programming for finding an optimal LA planning when the given LAP instance, $G(V, E)$, is a line. This algorithm can be used, for instance, in the LA planning of highways. Since highways carry a large portion of the user traffic, in some cases it is more economic to treat them separately. Cost-effective solutions for highway LA planning reduce the overall LA planning cost. For clarity of presentation, we first ignore connectivity and size constraints.

A *line* is formally defined as a graph where exactly two nodes have degree 1 and the remaining nodes have degree 2. Consider a line $G(V, E)$, connectivity matrix B and size bounds $K_{\max}$ and $W_{\max}$. We assume that the line nodes are indexed adjacently in increasing order from 1 to n, *i.e.,* for every i, j, $f_{ij} > 0$ and $b_{ij} \neq 0$ only if $|i - j| = 1$. We denote by $Cost(i)$ the cost of the optimal LA planning on the line graph G_i induced by the first i nodes, $[1..i] \in V$, where $Cost(0) = 0$ is the cost of the empty line.

We note that $Cost(i)$ (and the actual LA planning itself) can be computed recursively. Suppose the optimal LA planning in the graph G_i is $\{S_1, \ldots, S_k\}$ for some k. Let q be the index of the left border node in S_k (*i.e.,* the node with the lowest index in S_k). Then the optimal plan for G_{q-1} must have the same cost as $\{S_1, \ldots, S_{k-1}\}$ or else $\{S_1, \ldots, S_k\}$ would not be optimal for G_i. Thus $Cost(i)$ is the sum of $Cost(q-1)$ and the cost incurred by S_k:

$$Cost(i) = \min_{q=1}^{i} \left\{ \begin{array}{l} \lambda \cdot C_p \cdot |i-q+1| \cdot \sum_{j=q}^{i} w_j \\ + \; C_u \cdot f_{q-1,q} \; + \; Cost(q-1) \end{array} \right\} \tag{1}$$

where we take $f_{01} = 0$. Using dynamic programming, we can find the optimal LA planning for a line graph in time $O(n^2)$ where n is the number of nodes in the line. We simply calculate $Cost(i)$ for i from 1 to n and store the result of each iteration in a table so it may be used in the next iteration. As a technical detail, we must also store the sum of weights to avoid an extra factor of n in the running time (see Figure 1).

Finally, we address the connectivity and size constraints. Connectivity constraints $b_{ij} = 1$ split the line graph into multiple line graphs, and we can solve the problem optimally on each instance separately. In the presentation of our algorithm we assume such preprocessing has been done and $b_{ij} = 0$ or -1 for all i, j. Connectivity constraints of type $b_{ij} = -1$ are handled by the Line algorithm and any node q that is associated with a constraint $b_{q-1,q} = -1$ is disqualified to serve as a left border node. The size constraints are easily accommodated as well. We will discuss only the maximal LA size constraint, $K_{\max}$, as the weight constraint $W_{\max}$ is analogous. Let $x = \max\{1, i - K_{\max} + 1\}$. Then the size constraint forces the border node q to be amongst nodes $x, \ldots, i$, and so the minimization in Equation (1) is just taken over this range.

A formal description of our dynamic programming scheme is given in Figure 1. In this description, $\mathcal{L}$ and C are two arrays that store the optimal LA planning and its cost for the segment $[1..i]$, respectively. The variable W_{LA} records the total weight of the nodes $q, \ldots, i$. It is initialized by $W_{LA} = \sum_{j=x}^{i} w_j$, and it is decreased by w_q before increasing the border node index, q.

Theorem 2. *Given a line $G(V, E)$, connectivity matrix B and maximal LA size $K_{\max}$, the Line algorithm returns an optimal LA planning of $G(V, E)$ and its cost.*

Proof. We prove the correctness of the Line algorithm by induction on the number of nodes in the considered line. For an empty line without nodes there are no paging or update costs and its total cost is $Cost[0] = 0$. Let us assume that the theorem is valid for lines with $i - 1$ nodes and consider a line with i nodes. The algorithm calculates the cost of different solutions when checking each one of the feasible border nodes q of $LA(i)$ (the last $\max\{1, i - K_{\max} + 1\}$ nodes of the line), by using the equation,

$$\lambda \cdot C_p \cdot (i - q + 1) \cdot W_{LA} + C_u \cdot f_{q-1,q} + C[q-1].$$

By inductive assumption, $C[q]$, $q < i$, is the cost of the optimal LA planning for the line graph induced by nodes $[1..q]$. Therefore, the algorithm finds the cost of the optimal LA planning when a given node q is forced to be the border node of $LA(i)$. As it takes the minimum of these values over all feasible border nodes q, the algorithm finds both the optimal LA planning and its cost. □

```
Algorithm **Line**$(G(V,E), B, K_{\max})$
            // *Variable Initialization.*
  $C[0] = 0$
  $\mathcal{L}[0] = \emptyset$
  $f_{0,1} = 0$
            // *Main loop from 1 to n.*
  **for** $i = 1$ to $n$ do
    $C[i] = \infty$
            // *Updating the max weight of LA LA(i).*
    $x = \max\{1, i - K_{\max} + 1\}$
    $W_{LA} = \sum_{j=x}^{i} w_j$
            // *Loop for checking all border nodes.*
    **for** $q = x$ to $i$ do
            // *Checking if q can be a border node.*
      **if** $b_{q-1,q} = 0$ then
        $tmp = \lambda \cdot C_p \cdot (i - q + 1) \cdot W_{LA} +$
              $+ C_u \cdot f_{q-1,q} + C[q-1]$
        **if** $tmp < C[i]$ then
              // *A cheaper LA planning was found.*
          $C[i] = tmp$
          $\mathcal{L}[i] = \mathcal{L}[q-1] \bigcup \{\{q..i\}\}$
        **end-if**
      **end-if**
      $W_{LA} = W_{LA} - w_q$
    **end-for**
  **end-for**
  **return** $C[n], \mathcal{L}[n]$
**end**
```

Figure 1: A formal description of the Line Algorithm

4. APPROXIMATION ALGORITHMS

In this section we present approximation algorithms for LAP in both general graphs and planar graphs. We start by providing an integer programming formulation. Then, we relax the integrality constraints and obtain a linear program. We solve the linear program and obtain an optimal fractional solution for the problem, i.e., a solution where the variables can assume non-integral values. This solution serves as our lower bound on the value of an optimal solution. Finally, we round the fractional solution and obtain a near-optimal integral solution. We provide an upper bound on the ratio between the value of the near-optimal integral solution computed and the value of an optimal fractional solution. This bound is our *approximation factor*. For general graphs, this bound turns out to be logarithmic and for planar graphs it is a constant.

4.1 The Integer Program Formulation

Recall that LAP is defined as a clustering problem with a non-linear objective function. In order to formulate LAP as a linear integer problem, we take a different approach. A pair (V, d), where V is a set and d is a non-negative function $d : V \times V \to \mathbb{R}$, is called a *semi-metric* [41] if and only if d satisfies the following three conditions. (i) $d_{ij} = d_{ji}$ for all $i, j \in V$ (*symmetry*). (ii) $d_{ii} = 0$ for all $i \in V$. (iii) $d_{ij} \leq d_{ik} + d_{jk}$ for all $i, j, k \in V$ (*triangle inequality*). Consider a partition of the graph into LAs. Define for every pair of nodes $i, j \in V$ a variable $d_{ij} \in \{0, 1\}$ such that,

$$d_{ij} = \begin{cases} 1 & \text{If } i \text{ and } j \text{ belong to different LAs.} \\ 0 & \text{If } i \text{ and } j \text{ belong to the same LA.} \end{cases}$$

We claim that the variables d_{ij} induce a semi-metric. Conditions (i) and (ii) above are obviously satisfied. The triangle inequality is also satisfied, as can be seen by a simple case analysis.

Consider an assignment to the variables d_{ij} that induces a semi-metric. This assignment defines a partition of the graph into LAs in a natural way. For every node $i \in V$, the LA containing it, $LA(i)$, is defined by, $LA(i) = \{j | j \in V \wedge d_{ij} = 0\}$, i.e., all nodes that are in zero distance from node i. We refer to edges $(i,j) \in E$ for which $d_{ij} = 1$ as *cut edges*. Denote by $\mathcal{L}$ the partition into LAs induced by variables d_{ij}. Our integer program will have variables $d_{ij} \in \{0,1\}$, where d is required to be a semi-metric.

Recall the connectivity matrix B defined in Section 2. For every pair $i,j \in V$ we add the constraints, $b_{ij} \leq d_{ij} \leq b_{ij}+1$. If nodes i,j should be in the same LA then $b_{ij} = -1$, and the non-negativity constraint on d_{ij} yields that $d_{ij} = b_{ij}+1 = 0$. Similarity, if nodes i,j are required to be in different LAs then $b_{ij} = 1$, and since $d_{ij} \leq 1$, it follows that $d_{ij} = b_{ij} = 1$. If nodes i,j are not constrained then $b_{ij} = 0$, and $d_{ij} \in \{0,1\}$.

We now state the objective function, as well as the size and connectivity constraints, in terms of the variables d_{ij}. We first address the paging cost. A node i pages whenever there is an incoming call to a user belonging to its LA, $LA(i)$, and hence it performs $\lambda \cdot \sum_{j \in V}(1 - d_{ij}) \cdot w_j$ paging operations in a time unit. Thus, the paging cost of the entire system is

$$Page\_Cost(\mathcal{L}) = \lambda \cdot C_p \cdot \sum_{i,j \in V} (1 - d_{ij}) \cdot w_j .$$

The update cost is simply the cost of the cut edges times the cost of a single update operation, yielding

$$Update\_Cost(\mathcal{L}) = \frac{1}{2} \cdot C_u \cdot \sum_{i,j \in V} d_{ij} \cdot f_{ij} .$$

Hence, our objective function is,

$$\min \quad \lambda \cdot C_p \cdot \sum_{i,j \in V} (1 - d_{ij}) \cdot w_j + \frac{1}{2} \cdot C_u \cdot \sum_{i,j \in V} d_{ij} f_{ij} .$$

We now address the size constraints. Recall the bounds $K_{\max}$ and $W_{\max}$ defined in Section 2. Since, for each node $i \in V$, $LA(i) = \{j | j \in V \wedge d_{ij} = 0\}$, $|LA(i)| = \sum_{j \in V}(1 - d_{ij})$ (*number of nodes in* $LA(i)$), and $w(LA(i)) = \sum_{j \in V}(1 - d_{ij}) \cdot w_j$ (*weight of nodes in* $LA(i)$). Thus, we can enforce the size constraints on each LA by adding for each node $i \in V$ the constraints, $|LA(i)| \leq K_{\max}$ and $w(LA(i)) \leq W_{\max}$.

We are now ready to present the integer programming formulation.

$$\min \quad \lambda \cdot C_p \cdot \sum_{i,j \in V} (1 - d_{ij}) \cdot w_j + \frac{1}{2} \cdot C_u \cdot \sum_{i,j \in V} d_{ij} f_{ij}$$

subject to:

$$\begin{aligned}
\forall i,j,k \in V : &\quad d_{ij} + d_{jk} \geq d_{ik} \\
\forall i,j \in V : &\quad b_{ij} \leq d_{ij} \leq b_{ij} + 1 \\
\forall i \in V : &\quad \textstyle\sum_{j \in V}(1 - d_{ij}) \leq K_{\max} \\
\forall i \in V : &\quad \textstyle\sum_{j \in V}(1 - d_{ij}) w_i \leq W_{\max} \\
\forall i,j \in V : &\quad d_{ij} \in \{0,1\}
\end{aligned}$$

LEMMA 1. *An optimal solution to the above integer program defines an optimal solution to LAP.*

PROOF. It follows from the preceding discussion that a partitioning into LAs defines a feasible assignment to the variables d_{ij} and vice versa. The discussion also shows that size constraints are maintained by the constraints of the integer program. □

Since solving integer programs is an NP-hard problem, we relax the integrality constraints, i.e., we only require that for all $i,j \in V$, $d_{ij} \in [0,1]$. The linear program we obtain contains only $O(n^2)$ variables and $O(n^3)$ constraints, and therefore an optimal fractional solution can be computed in polynomial time. Clearly, the value of an optimal fractional solution is a lower bound on the value of an optimal integral solution.

4.2 Rounding Algorithm for General Graphs

In this section we present our rounding algorithm. We assume that an optimal fractional solution to the above linear program has been computed. We now show how to round the fractional solution to a near-optimal integral solution, while ensuring an $O(\log n)$-approximation factor in the worst case. Furthermore, we observed that, in practice, the fractional solution is very close to an integral solution. In other words, most of the d_{ij} parameters have values of 0 and 1 also in the fractional solution. Our rounding algorithm preserves these integral values and guarantees that every pair of nodes $i,j \in V$ such that $d_{ij} = 0$ are assigned to the same LA, while pairs with $d_{ij} = 1$ are assigned to two different LAs. This property enables us to find near optimal solution in most cases, as we illustrate by simulations in Section 7.

4.2.1 The Algorithm

Our rounding algorithm uses a technique known as *region growing* (or ball growing) [42]. We iteratively grow balls of at most some fixed radius around nodes of the graph with respect to the semi-metric defined by the variables d_{ij}. The balls are grown until all nodes are included in some ball, and these balls define the LAs in the final solution. The intuition is that large d_{ij} values indicate that i and j should be in separate LAs, and small d_{ij} values indicate that they can be in the same LA. The importance of the fixed radius is: (i) it guarantees that size bounds are "almost" satisfied; and (ii) it yields an approximation factor bound on the paging component of the objective function. The region growing technique itself enforces an approximation factor bound on the update component of the objective function.

First, we present some notation that we will need in order to define the algorithm. A *ball* $b(i,r)$ of radius r around node i is the subgraph that consists of all nodes j such that $d_{ij} < r$, their connecting edges and the fraction $(r - d_{ij})/(d_{ik} - d_{ij})$ of any edge (j,k) with only one endpoint, say j, belonging to the ball, i.e., $d_{ij} \leq r$ (note that d_{ij} is defined for all nodes i,j). Thus, the ball $b(i,0)$ around node i contains node i and all the nodes $j \in V$ such that $d_{ij} = 0$. The *cut* of a ball b is the set of edges with precisely one endpoint in b and its weight, denoted by $\text{cutweight}(b)$, is defined to be $\sum_{|\{i,j\} \cap b| = 1} f_{ij}$. Finally, the *volume* of a ball $b(i,r)$, $\text{vol}(b(i,r))$, is defined to be the weighted distance of the edges belonging to the ball. Each internal edge (j,k) contributes $f_{jk} \cdot d_{jk}$ to the ball volume and every cut edge (j,k), with $d_{ij} < r$, contributes $f_{jk} \cdot d_{jk} \cdot (r - d_{ij})/(d_{ik} - d_{ij})$ to $\text{vol}(b(i,r))$. For technical reasons, we also include an ini-

```
Algorithm Round($G(V,E), \{d_{ij}\}$)
  // Variable Initialization.
  $H \leftarrow G$
  $\mathcal{L} \leftarrow \emptyset$
  // Main loop
  while $\exists$ a node $i \in H$
    $S \leftarrow \emptyset$
    $r \leftarrow 0$
    // Grow ball
    repeat
      $S \leftarrow S \cup b(i,r)$
      $r \leftarrow r + \Delta$
    until cutweight$(b(i,r)) \leq c\ln(n+1) \cdot$ vol$(b(i,r))$
    $\mathcal{L} \leftarrow \mathcal{L} \cup S$
  end
```

Figure 2: A formal description of the ball growing algorithm

tial volume (*seed*) I to the volume of every ball (i.e. ball $b(i,0)$ has volume I).

We are now ready to present the algorithm for rounding a fractional solution to an integral solution. The input to the algorithm is a complete graph $G(V,E)$, $|V| = n$, with a fractional assignment to the variables d_{ij} obtained from the linear program. Suppose the volume of the entire graph is $F \equiv \frac{1}{2}\sum_{i,j\in V} d_{ij} f_{ij}$. Note that the update cost of the fractional solution is $C_u F$. Let the initial volume of the balls defined in the algorithm be F/n. The algorithm iteratively grows balls around arbitrary nodes of the graph until it finds a ball such that the weight of the cut defined by the ball is at most $c\ln(n+1)$ times the volume of the ball. It then creates a location area consisting of the nodes belonging to this ball and removes these nodes from the graph. This algorithm terminates when all nodes are removed from the graph. The pseudo-code for this algorithm can be found in Figure 2. Note that the order in which the algorithm considers nodes is indeed arbitrary, and thus the following analysis applies to any node selection heuristic.

4.2.2 The Approximation Factor Analysis

In this algorithm, c is some constant which we will determine later, and $\Delta = \min\{(d_{ij} - r) : j \notin b(i,r),\ (d_{ij} - r) > 0\}$ is the remaining distance to the nearest vertex (among those with distance greater than zero) outside the current ball. This algorithm clearly runs in polynomial time. We must show it terminates with a solution $\mathcal{L}$ that satisfies the constraints, and has a cost which is not much more than the fractional volume F.

Notice the region-growing procedure's termination condition guarantees an $O(\log n)$ approximation to the update component of the objective function. Let $\alpha = c\ln(n+1)$.

$$\begin{aligned}
Update\_Cost(\mathcal{L}) &= \frac{1}{2}C_u \sum_{\text{balls } b} \text{cutweight}(b) \\
&\leq \frac{1}{2}C_u\alpha \sum_{\text{balls } b} \text{vol}(b) \\
&\leq \frac{1}{2}C_u\alpha\left(\frac{1}{2}\sum_{i,j\in V} d_{ij}f_{ij} + \sum_{\text{balls } b}\frac{F}{n}\right) \\
&\leq \frac{1}{2}C_u\alpha(2F) \\
&\leq C_u\alpha F
\end{aligned}$$

where the second line follows from the fact that the balls found by the algorithm are disjoint. Note that $C_u F$ is precisely the update cost of the fractional solution.

The rest of our analysis hinges on the fact that the balls returned by this algorithm have radius at most $1/c$. This fact follows from the following known lemma [43, 42].

LEMMA 2. *For any vertex i and family of balls $b(i,r)$, the condition* cutweight$(b(i,r)) \leq c\ln(n+1) \cdot$ vol$(b(i,r))$ *is achieved for some $r \leq 1/c$.*

PROOF. We proceed by contradiction. Set $\alpha = c\ln(n+1)$. Consider growing the ball continuously from $r = 0$ to $r = 1/c$ and suppose throughout this process, cutweight$(b(i,r)) > \alpha \cdot vol(b(i,r))$. Notice that due to this assumption, the incremental change in the volume is

$$\begin{aligned}
d(\text{vol}(b(i,r))) &= d(\sum_{j,k\in b} f_{jk}d_{jk} + \\
&\quad \sum_{j\in b,k\notin b} f_{jk}\ d_{jk}\ (r - d_{ij})/(d_{ik} - dij)) \\
&= \sum_{j\in b,k\notin b} d\,(f_{jk}\ d_{jk}\ (r - d_{ij})/(d_{ik} - dij)) \\
&= \sum_{j\in b,k\notin b} d(f_{jk}\ d_{jk}\ r/(d_{ik} - dij)) \\
&\geq \sum_{j\in b,k\notin b} f_{jk}\ dr \\
&= \text{cutweight}(b(i,r))\ dr \\
&> \alpha\text{vol}(b(i,r))\ dr
\end{aligned}$$

The step $d(\text{vol}(b(i,r))) \geq \sum_{j\in b,k\notin b} f_{jk}dr$ results from the fact that $d_{jk} \geq (d_{ik} - dij)$ and therefore, $d_{jk}/(d_{ik} - dij) \geq 1$. The initial volume of a ball is, by definition, F/n, and the final volume is at most $F + F/n$ if the ball covers the entire graph. Therefore

$$\int_{F/n}^{F+F/n} \frac{1}{\text{vol}(b(i,r))} d(\text{vol}(b(i,r))) > \int_0^{1/c} \alpha dr$$

and so $\ln(n+1) > \frac{1}{c}\alpha = \ln(n+1)$. □

We now bound the paging cost of our rounding.

$$\begin{aligned}
Page\_Cost(\mathcal{L}) &= \lambda C_p \sum_{\text{balls } b}\sum_{i,j\in b} w_j \\
&= \frac{c}{c-2} \cdot \lambda C_p \sum_{\text{balls } b}\sum_{i,j\in b}(1 - 2/c)w_j
\end{aligned}$$

Note that $\lambda C_p \sum_b \sum_{i,j\in b}(1 - 2/c)w_j$ is a lower bound on the paging cost of the fractional solution. This is true since the radius of the balls is at most $1/c$ and therefore by the triangle inequality $1 - d_{ij} \geq 1 - 2/c$ for any nodes i and j that belong to the same ball. This implies that our solution gives a $\frac{c}{c-2}$-approximation to the paging cost.

The final approximation factor of our algorithm is the maximum between the approximation factors of the two components. Thus,

THEOREM 3. *The approximation factor of our algorithm is* $\max(c\ln(n+1), \frac{c}{c-2}) = O(\log n)$.

We note that the integral solution that our approximation algorithm computes preserves all the integral components of the fractional solution. Consider a variable d_{ij} with an integral value in the fractional solution. Suppose that $d_{ij} = 1$. Since the diameter of a ball is at most $2/c < 1$, the two nodes belong to separate LAs in the integral solution. Now, suppose that $d_{ij} = 0$. From the triangle inequality it follows that $d_{ik} = d_{jk}$ for every node $k \in V$. Thus, every ball $b(k, r)$, either contains both nodes i and j, or none of them.

4.2.3 Satisfying Connectivity and Size Constraints

We turn to prove that the algorithm returns a solution which satisfies the constraints. It easy to see that the rounding algorithm fulfills the connectivity constraints. For the case when a connectivity constraint $b_{ij} = 1$ is given, the fractional solution enforces $d_{ij} = 1$ (distance between i and j). Since the diameter of a ball is at most $2/c < 1$, the two nodes are in two separate LAs. When connectivity constraint $b_{ij} = -1$ is given, the fractional solution enforces $d_{ij} = 0$. As described above, the rounding algorithm keeps nodes with zero distance in the same ball and consequently in the same LA. We now handle the size constraints. Actually, we will prove something slightly weaker. We will show our algorithm is a pseudo-approximation algorithm. A pseudo-approximation algorithm gives an approximate solution to a problem with slightly different parameters. In our case, we must perturb the size bound parameters, $K_{\max}$ and $W_{\max}$, slightly. Specifically, our algorithm finds a set of LAs such that each LA has size at most $\frac{c}{c-1}K_{\max}$ and weight at most $\frac{c}{c-1}W_{\max}$. We prove the first of these statements. The proof of the second is similar. We know that $\forall i \in V : \sum_{j\in V}(1 - d_{ij}) \leq K_{\max}$. Fix i. Since the maximal radius of a ball $r \leq \frac{1}{c}$, follows that distance $d_{ij} \leq \frac{1}{c}$. Therefore, for each i,

$$\begin{aligned} K_{\max} &\geq \sum_{j\in V}(1 - d_{ij}) \\ &\geq \sum_{j\in LA(i)}(1 - d_{ij}) \\ &\geq \sum_{j\in LA(i)}\left(1 - \frac{1}{c}\right) \\ &= \frac{c-1}{c}\sum_{j\in LA(i)} 1 = \frac{c-1}{c}LA(i) \end{aligned}$$

Thus, the maximal LA size is at most $\frac{c}{c-1}K_{\max}$. We note that c is an arbitrary constant. The larger we take c, the closer our solution will be to the true size bounds. However, our approximation factor grows like $c\ln(n + 1)$, and so our overall costs may get worse. This parameter is a tradeoff that the user can specify. We also note that if the bounds $K_{\max}$ and $W_{\max}$ are not specified (i.e. can be arbitrarily large), then the algorithms we present are exact approximation algorithms in the standard sense of the term.

Finally, we show that our scheme can actually deal with more general constraints on the size and weight of LAs. For example, we can adjust our scheme to accommodate a constraint for each vertex $i \in V$ on the size and weight of the LA containing i, i.e., $|LA(i)| \leq K_i$ and $\sum_{j\in LA(i)} w_j \leq W_i$. without increasing the total number of constraints. The latter is obtained by replacing the two size constraints of each node $i \in V$ with the following constraints, $\sum_{j\in V}(1 - d_{ij}) \leq K_i$ and $\sum_{j\in V}(1 - d_{ij})w_i \leq W_i$. However, as a result of this modification, it can be shown that the LA size or weight may be as high as $\frac{c}{c-2}$ times the required size, K_i, or weight W_i.

4.3 Rounding Algorithm for Planar Graphs

In this section we prove that for planar graphs we can change the region growing algorithm and obtain a constant factor approximation. We use the following technique developed by Klein, Plotkin, and Rao [44] (see also [45]). The *weak diameter* of a subset S of nodes is r if every pair of nodes in S is at distance at most r in the original graph (and not necessarily in the graph induced by S).

THEOREM 4 (KPR). *Given a planar graph with capacities u on its edges and parameter p, one can find, in polynomial time, an edge separator of total capacity $O(U/p)$ whose removal yields components of weak diameter at most $O(p)$ where U is the sum of all capacities.*

We will use this theorem to find an LA design with components of radius at most $1/c$ and with update cost just a constant factor more than the optimal fractional solution. The other results required, i.e. the constant factor approximation for the paging cost and the constraint-satisfaction argument, follow from the $1/c$ radius guarantee.

The next Corollary follows from Theorem 4 by creating from G another planar graph G', mapping edge (i, j) to a chain of length $\lceil Bd_{i,j}\rceil$, where each link in the chain is an edge of weight f_{ij}, for some appropriate large B. We can then find the required cut by applying Theorem 4 to G' and scaling down the result.

COROLLARY 1. *Given a planar graph G with distances d_{ij} and weights f_{ij} on its edges, one can find, in polynomial time, a cut of weight $O(\text{vol}(G)/p)$ which yields components of radius at most $O(p)$, where $\text{vol}(G) = \sum_{i,j} d_{ij} f_{ij}$.*

Given an optimal fractional solution, we can use Corollary 1 with an appropriate setting of constants to obtain an LA design $\mathcal{L}$, where components have radius at most $1/c$ for any constant c (note p will be constant too). As the update cost of the fractional solution is $\text{vol}(G)$ and the update cost of $\mathcal{L}$ is the cost of the cut, Corollary 1 gives a constant-factor approximation guarantee for $\mathcal{L}$. We note that this constant may be very large. Due to the $1/c$ radius guarantee, all previous results in the discussion on general graphs follow, yielding a constant factor approximation for the LA design problem in planar graphs.

5. HEURISTICS

The region growing algorithm presented in Section 4.2 provides an LA planning with bounded cost and LA sizes. Specifically, for any LAP instance and maximal region radius $1/c$, for a given $c > 2$, the algorithm guarantees a solution such that its update cost is at most $c\cdot\ln(n+1)\cdot OPT$ and its paging cost is at most $\frac{c}{c-2}\cdot OPT$, where OPT is the cost of the optimal solution. Thus, for $c = 3$ the scheme ensures $(3\cdot\ln(n+1), 3/2)$-approximation factor. In other words, the calculated solution will be within a factor of $3\cdot\ln(n+1)$ from the optimal solution and the sizes and weights of its LAs are at most $1.5K_{\max}$ and $1.5W_{\max}$, respectively. These bounds are ensured by prudently balancing between the update and the paging costs with respect to the fractional solution.

```
Algorithm $\mathbf{Exchange}(\mathcal{L})$
  madeChange = true
  while(madeChange)
    madeChange = false
    foreach LA $S \in \mathcal{L}$
      foreach cut edge $(u, v)$ s.t. $u \in S, v \in S'$
        if (size$(S')$ $+ 1 \leq K_{max}$ and
                weight$(S')$ + weight$(u) \leq W_{\max}$)
          oldCost = cost$(S)$ + cost$(S')$
          newCost = cost$(S' \cup u)$ + cost$(S \setminus u)$
          if (newCost < oldCost)
            madeChange = true
            $S = S \setminus u$
            $S' = S' \cup u$
          end-if
        end-if
      end-foreach
    end-foreach
  end-while
```

Figure 3: A formal description of the cell exchanging heuristic

We develop a simple heuristic that improves the overall cost of the LAP solution produced by the region growing algorithm. The heuristic uses a greedy strategy, and is also an extension of the scheme presented in [36]. The pseudocode for the heuristic is shown in Figure 3. The *Exchange* heuristic takes the initial LA planning solution produced by the region growing algorithm and tries to improve the cost by exchanging cells between neighboring LAs. The idea of the heuristic is to move nodes between neighboring LAs, where the move results in a decrease in overall cost without causing any constraint violation. We continue this process until there is no more cost improvements. It is clear that the process will terminate because a local minima will be reached after which the exchange process will give no further improvement in cost.

Depending on how tight the size and weight constraints are for the PCS network, a violation of these constraints may not be acceptable. We have two proposals to fix this problem. The first is to set the constraints in the LP formulation so that the real constraints are not violated in the final solution. The second is to do a cell exchange similar to the exchange heuristic above. For each LA S which is in violation of the size or weight constraints, look at its neighboring LAs and see which neighbor S' can take a cell from S without violating the constraints and causing the minimal increase in overall cost. The process is repeated until S is no longer in violation of the size and/or weight constraints. The first heuristic guarantees a feasible solution, while the second one may give better results in practice, but does not guarantee feasibility.

6. HANDOFF MANAGEMENT

So far we have addressed location management. However, our algorithms can also be used for the planning of MSC-domains for constructing efficient handoff mechanisms. Handoffs that occur between cells in different MSCs tend to cause degradation in the quality of the provided service, in the forms of higher delays, increased data lost and connection drop-offs, while the handoffs are happening. Thus, to improve the quality of service provided to the users, we would like to partition the network cells into disjoint MSC-domains that reduce the number of inter-MSC handoff operations. It is clear that inter-MSC handoffs are eliminated if all the cells are associated with a single MSC. However, due to physical and performance constraints, each MSC can be connected only to a limited number of base stations (cells) and it can support a bounded number of connections simultaneously. Consequently, we define an efficient *MSC-domain planning* to be a partition of the networks cells into a small number of clusters, so called MSC-domains, that minimize the total number of inter-MSC handoffs, while the size and the user population of each MSC-domain are bounded by $K_{\max}$ and $W_{\max}$, respectively. Recall that MSC-domain planning can be viewed as a special case of the LA-planning problem, described in Definition 1, where the cost of a single paging operation $C_p = 0$ and the cost of an update operation $C_u = 1$. This implies that our LA-planning algorithm can be used for the determining MSC-domains.

We now describe a combined approach for planning both the MSC-domains for efficient handoff management and the LAs for cost-effective location management. Our approach is based on the following two observations; Since, each LA and each MSC-domain is associated with a single MSC, it follows that both LAs and MSC-domains have to satisfy the same size constraints. Moreover, as LA-planning considers also the paging cost, LAs are in general smaller than the MSC-domains. Consequently, we view the LA-planning as a refinement of the MSC-domain partition. We start with calculating an efficient MSC-domain planning. Then, we further divide each MSC-domains to several LAs, by employing LA-planning. This approach optimizes the two components of the mobility management mechanism.

7. SIMULATION RESULTS

In this section we present the results of our experiments to evaluate how well our algorithm works in practice. We start by describing the experimental set up.

7.1 Methodology

The region growing algorithm described in Section 4.2 grows regions continuously around an initial node (seed). To implement the algorithm, we grow regions in discrete steps. The discrete algorithm is based on the process described in [42]. The $O(\log n)$ approximation bound also holds for this method. The discrete algorithm builds shortest path trees from the seeds and grows regions by adding nodes in increasing distance from the seeds. At each iteration, nodes with the same distance from the current seed are added together.

For our simulations we use data that was collected from a big wireless service provider in the United States. The data covers several MSCs and is for a region in New Jersey. We had two sets of data to work with. One set is a handoff matrix giving the number of handoffs between cells for a particular day. The other set of data gives a measure of the number of incoming calls for each cell during the busy period. The data we worked with is not complete in that it does not have information for all the cells of the MSCs represented, and the set of cells covered by the handoff matrix does not intersect fully with the set of data containing the measure of the number of incoming calls.

The number of cells for which we have both handoff information and incoming call data is only 39, and we present simulation results for this set of cells. However, this is quite small and may not be representative of a full network. Since we use the handoff data to determine cell connectivity, we wanted to use the cells for which we have handoff information. For the cells in this set that we had incoming call data for, we use that data. To generate incoming call data for the rest of cells of this set, we used *non-parametric bootstrap* re-sampling [46]. This is a standard statistical technique for filling in missing data and gives good results in practice. Using this method, we generate two other networks, one with 76 cells and the other with 128 cells.

Since paging and update costs are not typically measured in comparable units, having an objective function that sums these two costs can be problematic. The standard solution used in location area planning research is to make an assumption on the relative cost of the these units. For example, in [31] the authors use a 17:1 ratio for update to paging cost. For our experiments we make a similar assumption, but we present a range of values for this ratio. We use a range from 10:1 to 30:1 in increments of 5. The other parameters we need to consider for our experiments are size and weight restrictions. In the following we only present results where there are size constraints. For our experiments we use a size constraint of about 20% of the total number of cells in the network.

To show the effectiveness of our region growing algorithm (RG), we compare our results to the optimal fractional solution of the LP formulation of the problem as well as other methods described in the literature. As summarized in Section 1.1, there is a large body of work on the issue of LA-planning. In this body, some papers [28], [29], [30],[31] assume uniform user distribution and inter-cell movement rate, and with these strong assumptions they derive optimal LA plans. However, these results are not applicable to practical cases where the network usage is heterogeneous and therefore they are not suitable for comparison with our method. Other papers utilize sophisticated variants of exhaustive search and employ different algorithmic tools such as genetic algorithms [33],[34], taboo search [34] and simulated annealing [34],[32], [35]. These algorithms do not have polynomial running time and the quality of the found solutions depend on the duration of the execution. For our comparisons, we would like to consider only algorithms with polynomial running time. However, in the literature, only a few polynomial time LAP algorithms are presented. Most of them consider relaxed versions of the LAP where the number of LAs is fixed [37] or the cost of the paging is ignored [38], [36]. Thus, to the best of our knowledge, there is not any polynomial time algorithm that considers the comprehensive LAP problem that we address in this work. Moreover, none of the above papers compared its results with the optimal solutions.

Consequently, we compare our region growing algorithm (RG) to the optimal fractional solution, denoted by LP, as well as to two polynomial time greedy heuristics, described in the literature. The first heuristic, *greedy1* (Gr1), is a simple an intuitive greedy algorithms similar to the one used in [35] as a benchmark, while the second heuristic, *greedy2* (Gr2), is based on the scheme proposed in [36]. Both greedy heuristics first perform a merge-phase and then an exchange-phase. The exchange-phase for both algorithms is the same as described in Section 5, but the algorithms differ in the initial merge-phase. In the first algorithm, *greedy1*, the merge-phase of this scheme is done as follows:

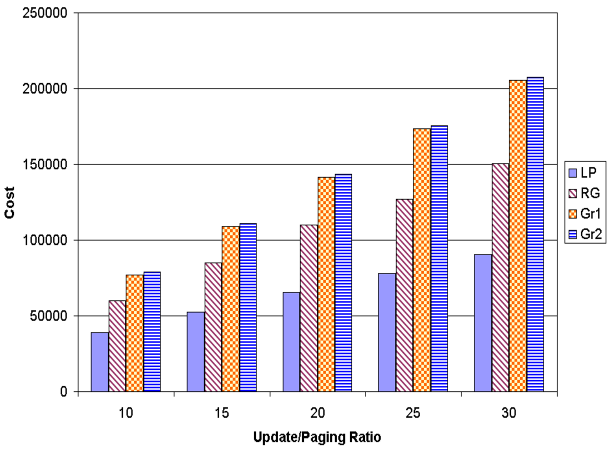

Figure 4: 39 cells: Cost comparisons of region growing to other methods.

> Initially, each cell induces a separate LA. The algorithm considers every pair of LAs and calculates the cost reduction obtained by merging them, then it selects the pair that yields the maximal cost reduction without violating the size or weight constrains and merges them. The process continues as long as there are pairs of LAs where merger reduces the system cost.

The second greedy algorithm, *greedy2*, employs the following merge-phase:

> Each cell is initially a separate LA. A profit metric is defined based on the update and paging costs of cells in the system. In each round of the merge-phase, each pair of LAs is tested to see which pair maximizes the profit metric and does not violate the size or weight constraints. This pair is merged, and then the process is repeated. The process continues until no more merges can be made.

7.2 Results

The results of our simulations are presented in the histograms and tables below. The histograms and tables show the overall system cost when the networks are partitioned using the different algorithms. The tables also show the relative costs of the different methods.

For the 39 cells case (Figure 4 and Table 1) the RG algorithm was on average 63% from the optimal fractional solution, whereas the greedy methods were more than twice as bad as the LP solution on average. For this case, the RG algorithm was on average better than Gr1 and Gr2 by 31% and 34% respectively. For the 76 cells network(Figure 5 and Table 2) all the algorithms were closer to the optimal solution than for the 39 cells network. However, the RG algorithm was much closer that the greedy methods. On average our method was 20% from the optimal solution, and for the case where the update to paging ratio was 25, it was

Ratio	LP	RG	Gr1	Gr2	RG/LP	Gr1/LP	Gr2/LP	Gr1/RG	Gr2/RG
10	39029	59975	76831	79187	1.54	1.97	2.03	1.28	1.32
15	52353	85173	109137	111247	1.63	2.08	2.12	1.28	1.31
20	65259	109905	141253	143307	1.68	2.16	2.20	1.28	1.30
25	78022	126923	173363	175367	1.63	2.22	2.25	1.37	1.38
30	90665	150493	205473	207427	1.66	2.27	2.29	1.37	1.38

Table 1: Results: 39 cells with maximum LA size 8

Ratio	LP	RG	Gr1	Gr2	RG/LP	Gr1/LP	Gr2/LP	Gr1/RG	Gr2/RG
10	123069	141309	190925	190925	1.15	1.55	1.55	1.35	1.35
15	155151	185357	250875	250875	1.19	1.62	1.62	1.35	1.35
20	183738	232607	319168	310825	1.27	1.73	1.69	1.37	1.34
25	210705	222392	395536	370775	1.06	1.88	1.76	1.78	1.67
30	237001	315870	460326	430725	1.33	1.94	1.81	1.46	1.36

Table 2: Results: 76 cells with maximum LA size 16

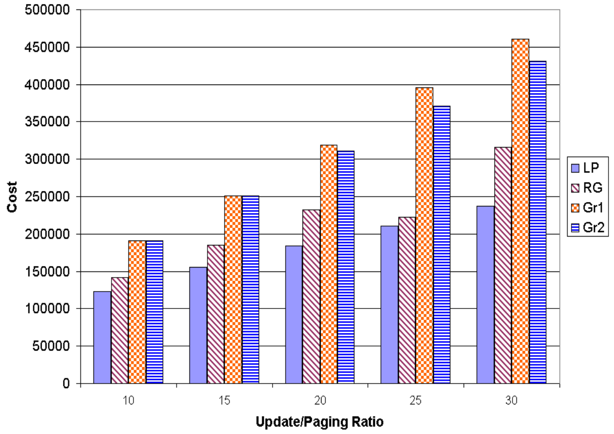

Figure 5: 76 cells: Cost comparisons of region growing to other methods.

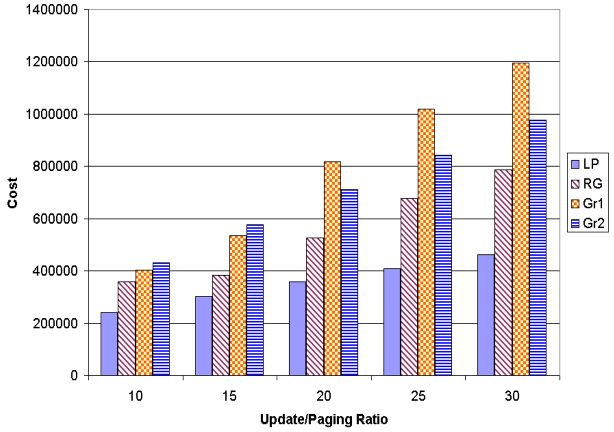

Figure 6: 128 cells: Cost comparisons of region growing to other methods.

only 6% from the optimal fractional solution. For this network the Region Growing algorithm was on average 46% and 41% better than Gr1 and Gr2 respectively. Finally, for the 128 cells network (Figure 6 and Table 3) the performance of the RG growing algorithm was on average 51% from the LP solution, and better that Gr1 and Gr2 by 42% and 31% respectively.

Over all our experiments, the RG algorithm came as close as 6% to the optimal fractional solution of the LAP problem, and was never more than 71% from this optimal solution. The RG algorithm was also consistently and significantly better than both greedy heuristics. Our experiments demonstrates that our method gives very good results, which in practice are much better than the $O(\lg n)$ worst-case bound.

8. CONCLUDING REMARKS

In this work, we described new clustering algorithms for LA planning that minimize both the update and paging costs from the wireless and wired network perspective. We presented a polynomial-time algorithm that finds an optimal LA planning for one-dimensional networks such as highways and railroads. In general networks, we have formulated the problem as an integer program. Our formulation is very flexible and it allows the incorporation of constraints that can capture a variety of system-imposed constraints, such as maximum LA size. Due to the NP-hardness of the problem, we resorted to polynomial time approximation algorithms that compute an LA planning whose cost is no more than $O(\log n)$ times the optimal cost (for planar graphs the algorithm achieves a constant approximation factor). We also simulated the rounding algorithm and coupled it with a heuristic. The results of our experiments on a realistic network indicate that our algorithms give results that are close to the the optimal solution and superior to the existing greedy heuristics to which we compare them. We also described how our algorithms can be used for MSC-domain planning, which is essential for improving the user perceived QoS.

The main theoretical contribution of this work — a method to balance between some property of the clusters and the weight of the cut — is applicable in other settings as well. Recent work [47, 48, 49] uses similar techniques to give an

Ratio	LP	RG	Gr1	Gr2	RG/LP	Gr1/LP	Gr2/LP	Gr1/RG	Gr2/RG
10	240868	357439	402660	432341	1.48	1.67	1.79	1.13	1.21
15	302595	382487	536027	577102	1.26	1.77	1.91	1.40	1.51
20	357798	525951	818519	710212	1.47	2.29	1.98	1.56	1.35
25	409998	676806	1019530	843322	1.65	2.49	2.06	1.51	1.25
30	460908	787941	1196650	976432	1.71	2.60	2.12	1.52	1.24

Table 3: Results: 128 cells with maximum LA size 26

$O(\log n)$ approximation for another clustering problem, *correlation clustering* [50]. It is conceivable that there are other problems with similar LP formulations that can benefit from our techniques, and applying our techniques to these problems is an interesting extension of this work.

Another remaining open question is the existence of a better, say constant-factor, approximation algorithm for the LAP problem in general graphs. In fact, due to the apparent similarity of the LAP problem and known hard problems such as *minimum multicut* [42], we believe it is unlikely that a $o(\log n)$ approximation algorithm exists. However, we do not know of a reduction or an $\Omega(\log n)$ LP-gap to support this claim. In a similar vein, it would be interesting to find examples for which the existing greedy heuristics perform poorly, returning a solution that is $\omega(\log n)$ more costly than the optimal one.

9. ACKNOWLEDGMENTS

We would like to thank Georg K. Hampel and John M. Graybeal for providing us with network data, and Duncan Temple Lang for insights on how to generate missing data.

10. REFERENCES

[1] M. Mouly and M. B. Pautet. *The GSM System For Mobile Communication*. Telecom Publishing, France, June 1992.

[2] I. F. Akyildiz, J. McNair, J. S. M. Ho, H. Uzunalioglu, and W. Wang. Mobility management in next generation wireless systems. *IEEE Proceedings Journal*, 87(8):1347–1385, August 1999.

[3] 3GPP, Boston, MA, USA and London, UK. *3GPP TS 23.012 V5.0.0, 3rd Generation Partnership Project, Technical Specification Group Core Network, Location Management Procedures (Release 5)*, March 2001.

[4] B. Jabbari, G. Colombo, A. Nakajima, and J. Kulkarni. Network issues for wireless communication. *IEEE Communications Magazine*, 33(1):88 –99, January 1995.

[5] R. Steele, J. Whitehead, and W. C. Wong. Network issues for wireless communication. *IEEE Communications Magazine*, 33(1):80–87, January 1995.

[6] S. Okasaka, S. Onoe, S. Yasuda, and A. Maebara. A new location updating method for digital cellular systems. In *Proceeding of the IEEE* 41^{st} *Vehicular Technology Conference, VTC'41*, pages 345–350, St. Louis, Missouri, May 1991.

[7] P. G. Escalle, V. C. Giner, and J. M. Oltra. Reducing location updates and paging costs in a pcs network. *IEEE Transaction on Wireless Communications*, 1(1):200–209, January 2002.

[8] M. Shirota, Y. Yoshida, and F. Kubota. Statistical paging area selection scheme (spas) for cellular mobile communication systems. In *Proceeding of the IEEE* 44^{th} *Vehicular Technology Conference, VTC'44*, volume 1, pages 367–370, 1994.

[9] S. Mishra and O. K. Tonguz. Most recent interaction area and speed-based intelligent paging in pcs. In *Proceeding of the IEEE* 47^{th} *Vehicular Technology Conference, VTC'47*, volume 2, pages 505–509, 1997.

[10] C. Rose and R. Yates. Minimizing the average cost of paging under delay constraints. *ACM-Baltzer Journal of Wireless Networks*, 1(2):211–219, July 1995.

[11] Y. Bejerano and I. Cidon. Efficient location management based on moving location areas. In *Proceedings of IEEE INFOCOM'01*, pages 3–12, Anchorage, Alaska, USA, April 2001.

[12] Hai Xie, Sami Tabbane, and David J. Goodman. Dynamic location area management and performance analysis. In *Proceeding of the IEEE* 43^{th} *Vehicular Technology Conference, VTC'43*, pages 536–539, May 1993.

[13] Z. Lei and C. Rose. Wireless subscriber mobility management using adaptive individual location areas for pcs systems. In *IEEE International Conference on Communications, ICC'98*, volume 3, pages 1390 –1394, 1998.

[14] Z. Lei, C. U. Saraydar, and N. B. Mandayam. Paging area optimization based on interval estimation in wireless personal communication networks. *ACM-Baltzer Journal of Mobile Networks and Applications*, 5(1):85–99, March 2000.

[15] J. Ming-Hui, H. Jorng-Tzong, and H-K. Wu. Personal paging area design based on mobiles moving behaviors. In *Proceedings of IEEE INFOCOM'01*, volume 1, pages 21–30, Anchorage, Alaska, USA, April 2001.

[16] S. Tabbane. An alternative strategy for location tracking. *IEEE Journal on Selected Areas in Communications, JSAC*, 13(5):880–892, June 1995.

[17] G. P. Pollini and I. Chih-Lin. A profile-based location strategy and its performance. *IEEE Journal on Selected Areas in Communications, JSAC*, 15(8):1415 –1424, October 1997.

[18] S. K. Sen, A. Bhattacharya, and S. K. Das. A selective location update strategy for pcs users. *ACM-Baltzer Journal of Wireless Networks*, 5(5):313–326, October 1999.

[19] A.Bhattacharya and S. K. Das. Lezi-update: An information-theoritic approach to track mobile users in pcs networks. In *Proceedings ACM/IEEE MobiCom '99*, pages 1–12, Sealtte, WA, August 1999.

[20] A. Bar-Noy, I. Kessler, and M. Sidi. Mobile users: To

update or not to update? *ACM-Baltzer Journal of Wireless Networks*, 1(2):175–185, July 1995.

[21] I. F. Akyildiz and J. S. M. Ho. Dynamic mobile user location update for wireless pcs networks. *ACM-Baltzer Journal of Wireless Networks*, 1(2):187–196, July 1995.

[22] I. F. Akyildiz, J. S. M. Ho, and Y. Lin. Movement-based location update and selective paging for pcs networks. *IEEE/ACM Transactions on Networking, ToN*, 4(4):629–638, August 1996.

[23] I. F. Akyildiz and J. S. M. Ho. A mobile user location update and paging mechanism under delay constrains. *ACM-Baltzer Journal of Wireless Networks*, 1(4):413–425, December 1995.

[24] B. Liang and Z. J. Haas. Predictive distance-based mobility management for pcs networks. In *Proceedings of IEEE INFOCOM'01*, volume 3, pages 1377–1384, New York, NY, USA, March 1999.

[25] G. Wan and E. Lin. Cost reduction in location management using semi-realtime movement information. *ACM-Baltzer Journal of Wireless Networks*, 5(2):245–256, July 1999.

[26] Z. Lei and C. Rose. Probability criterion based location tracking approach for mobility management of personal communication systems. In *IEEE Global Telecommunications Conference, GLOBECOM '97*, volume 2, pages 977–981, 1997.

[27] V. Wong and V. Leung. Location management for next-generation personal communciations network. *IEEE Network*, pages 18–24, September/October 2000.

[28] R. Thomas, H. Gilbert, and G. Mazziotto. Influence of the moving of the mobile stations on the performance of a radio mobile cellular network. In *Proceedings 3rd Nordic Seminar*, Copenhagen, Denmark, September 1998.

[29] E. Alonso, K. S. Meier-Hellstern, and G. P. Pollini. Influence of cell geometry on handover and registration rates in cellular and universal personal telecommunications networks. In *Proceedings 8th Int. Teletraffic Seminar*, pages 261–270, Genova, Italy, October 1992.

[30] M. Vudali. The location area design problem in cellular and personal communications systems. In *5th IEEE International Conference on Universal Personal Communications*, volume 2, pages 591–595, 1996.

[31] Chih-Lin I, Gregory P. Pollini, and Richard D. Gitlin. Pcs mobility management using the reverse virtual call setup algorithm. *IEEE/ACM Transactions on Networking, ToN*, 5(1):13–24, February 1997.

[32] C. U. Saraydar and C. Rose. Location area design using population and traffic data. In *Proceedings of Conference on Information Science and Systems, CISS 1998*, pages 739–744, Prinston, NJ, USA, March 1998.

[33] P. R. L. Gondim. Genetic algorithm and the location area partition problem in cellular networks. In *Proceeding of the IEEE* 46^{th} *Vehicular Technology Conference, VTC'46*, volume 3, pages 1835 –1838, 1996.

[34] P. Demestichas, E. Tzifa, V. Demesticha, N. Georgantas, G. Kotsakis, M. Kilanioti, M. Striki, M. E. Anagnostou, and M. E. Theologou. Control of the location update and paging signaling load in cellular systems by means of planning tools. In *Proceeding of the IEEE* 49^{th} *Vehicular Technology Conference, VTC'49*, volume 4, pages 2119–2123, 1999.

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

[36] J. Plehn. The design of location areas in a gsm-network. In *Proceeding of the IEEE* 45^{th} *Vehicular Technology Conference, VTC'45*, volume 2, pages 871–875, 1995.

[37] I. G. Tollis. Optimal partitioning of cellular networks. In *IEEE International Conference on Communications, ICC'96*, volume 3, pages 1377 –1381, 1996.

[38] M. Munguia-Macario, D. Munoz-Rodriguez, and C. Molina. Optimal adaptive location area design and inactive location areas. In *Proceeding of the IEEE* 47^{th} *Vehicular Technology Conference, VTC'47*, volume 1, pages 510–514, 1997.

[39] C. U. Saraydar, O. E. Kelly, and C. Rose. One-dimensional location area design. *IEEE Transactions on Vehicular Technology*, 49(5):1626 –1632, September 2000.

[40] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* Freeman Publication, New York, 1979.

[41] M. M. Deza and M. Laurent. *Geometry of cuts and metrics.* Springer-Verlag, Berlin-Heidelberg, 1997.

[42] V. V. Vazirani. *Approximation algorithms.* Springer-Verlag, 2001.

[43] V.V. Vazirani N. Garg and M. Yannakakis. Approximate max-flow min-(multi) cut theorems and their applications. *SIAM J. Computing*, 25:235–251, 1996.

[44] S. Plotkin P. Klein and S. Rao. Excluded minors, network decomposition, and multicommodity flow. In *Proceeding of the 25th ACM symposium on Theory of Computing*, pages 682–690, 1993.

[45] E. Tardos and V.V. Vazirani. Improved bounds for the max-flow min-multicut for planar and $k_{r,r}$-free graphs. *Information Processing Letters*, 47:77–80, 1993.

[46] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap.* Chapman & Hall, New York, NY, first edition, 1993.

[47] Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *IEEE Symp. on Foundations of Computer Science*, 2003.

[48] Erik D. Demaine and Nicole Immorlica. Correlation clustering with partial information. *Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, 2003.

[49] Dotan Emanuel and Amos Fiat. Correlation clustering — minimizing disagreements on arbitrary weighted graphs. *European Symp. on Algorithms*, 2003.

[50] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *IEEE Symp. on Foundations of Computer Science*, 2002.