

Efficient Integration of Multi-hop Wireless and Wired Networks with QoS Constraints

Yigal Bejerano

Bell Laboratories, Lucent Technologies,
600 Mountain Avenue, Murray Hill, NJ 07974

Abstract—This work considers the problem of designing an efficient and low-cost infrastructure for connecting static multi-hop wireless networks with wired backbone, while ensuring QoS requirements such as bandwidth and delay. This infrastructure is useful for designing low cost and fast deployed access networks in rural and suburban areas. It may also be used for providing access to sensor networks or for efficient facility placement in wireless networks. In these networks some nodes are chosen as access points and function as gateways to access a wired backbone. Each access point serves a cluster of its nearby user and a spanning tree rooted at the access point is used for message delivery. The work addresses both the design optimization and the operation aspects of the system. From the design perspective, we seek for a partition of the network nodes into minimal number of disjoint clusters that satisfy multiple constraints; Each cluster is required to be a connected graph with an upper bound on its radius. We assume that each node has a weight (representing its bandwidth requirement) and the total weight of all cluster nodes is also bounded. We show that these clustering requirements can be formulated as an instance of the Capacitated Facility Location problem (CFLP) with additional constraints. By breaking the problem into two sub-problems and solving each one separately, we propose polynomial time approximation algorithms that calculate solutions within a constant factor of the optimal ones. From the operation viewpoint, we introduce an adaptive delivery mechanism that maximizes the throughput of each cluster without violating the QoS constraints.

Keywords— Wireless Access Network, Sensor Networks, Unit Disk Graphs, Clustering, Facility Location, Approximation Algorithms.

I. INTRODUCTION

The tremendous advance of wireless communication technologies has prompted the development of new wireless applications. Among these new applications we found the broadband wireless access (BWA) systems [1], [2] and the wireless sensor networks [3], [4]. BWA systems provide broadband access for residential customers for both telephone and Internet connectivity and they are easily deployed in areas where the existing communication networks cannot support the bandwidth demands. Wireless sensor networks, like the Wireless Integrated Network Sensors (WINS) [3], provide monitoring and control capacities over large areas for monitoring applications, like transportation, manufacturing, health care and security monitoring. These networks are composed of large number of low power sensors that collect measurement, acoustic or video information and send it to monitoring centers. In both applications, most of the communication is between the wireless nodes and stations outside the wireless networks, *i.e.*, monitoring centers or Internet servers. This raises the need for efficient connectivity between the wireless nodes and the fixed networks that minimizes the number of direct connections while satisfying *Quality of Service* (QoS) constraints.

Currently, in sensor networks some nodes are selected as access points between the sensor network and the fixed backbone [4]. However, to the best of our knowledge there is no rigorous method for selecting these nodes. BWA systems use a cellular-like approach, where a single access point provides wireless connectivity to a group of static users in its transmission range.

Such an approach is suitable for urban areas with dense user population. However, it may be inefficient in rural and suburban areas with sparse population, where the number of users served by a single access point is low. Although the user to access point ratio can be improved by increasing the transmission power of both the access points and the users, the latter causes significantly higher energy consumption by the wireless stations, increases the wireless channel interference and requires more expensive hardware. These drawbacks make the cellular-like approach inefficient solution for sensor networks and for BWA systems in rural areas. In [5], Beyer *et al.* presented a low-cost Internet access infrastructure for geographically scattered communities termed *rooftop community networks*. In their scheme, the community houses are equipped with simple antennas and create a multi-hop wireless network. Some of the nodes, termed *access points*, are directly connected to a backbone network and function as gateways to access the Internet. This approach gains a lot of attention in the recent years and some community networks have already been launched in several cities in U.S.A. [6].

This work uses the concept of *community access network* for constructing an efficient and low-cost infrastructure that connects static wireless networks with wired backbone network while ensuring QoS requirements such as bandwidth and delay. The proposed infrastructure can be used for broadband access control, collecting monitoring information from sensor networks, or for facility placement when the provided services need to satisfy QoS constraints. Our scheme divides the network nodes into *clusters* and selects a single access point node at each cluster. The access point functions as gateways to access the wired backbone for all its cluster nodes. At each cluster, a spanning tree rooted at the access point is used for message delivery, where the tree's internal nodes (beside the access point) serve as relays for the benefit of the distinct stations. This tree-based approach simplify the routing [7] and enables us to maximize the network utilization while ensuring the QoS constraints. By selecting a small number of clusters, the number of required access points as well as the system cost are reduced. However, for operational considerations and for QoS assurance the clusters must satisfy several constraints: The clusters should be disjoint and contain all the network nodes. Each cluster must induce a connected graph with fixed bound, R , on its radius for multi-hop routing with bounded propagation delay. For QoS based access, there is a bound on the total bandwidth requirement of all the nodes in a cluster and a node may serve as relay only for a limited number of users. Our work addresses both the design and the operation aspects of the proposed infrastructure. We consider the *clustering problem* of minimizing the number of clusters that satisfy the above constraints and we present polynomial-time approximation algorithms that guaran-

tee solutions within a constant factor of the optimum ones. We also introduce an adaptive delivery mechanism that maximizes the throughput of each cluster without violating the QoS constraints.

Clustered and hierarchical structures are widely used in communication networks [8] and numerous algorithms for low-diameter graph decomposition are described in the literature [9], [10], [11]. Most of these studies model the communication networks as arbitrary graphs and they do not consider practical topologies of wireless networks. In wireless networks clustering techniques are extensively used for routing [12], [13], [14], [15], location management and facility location [16] (see also [17]). Most of these schemes proposed heuristics for dividing the wireless network into clusters with bounded diameter when ignoring the clusters' sizes. It is usually the case that the number of calculated clusters is evaluated by simulations and no guarantee, in comparison to the optimal solution, is proven. Recently, in [18], Banerjee and Khuller presented a clustering scheme that partitions a given wireless network to connected clusters that satisfy minimum and maximum size constraints. In their scheme, a node may be included in a constant number of clusters and the intersection of two clusters may include a small number of nodes. To the best of our knowledge, there is no clustering algorithm that satisfies all the requirements of our clustering problem.

In this paper, we present the clustering problem as a variant of the capacitated facility location problem (CFLP) [19], [20] with additional constraints and an integer programming formulation is given. Rather than addressing the combined problem, we break it into two sub-problems, which are both NP-hard. The first one seeks to find minimal number of disjoint connected clusters that contain all the nodes and satisfy the radius constraint. For this problem we present "heavy" and "light" algorithms, in terms of time complexity, with different approximation ratios. The "heavy" algorithm uses the *shifting strategy* of Hochbaum and Maass [21], [22] and allows us to control the complexity and the quality of the solution. The "light" algorithm is based on geometric properties of unit disk graphs and guarantees an approximation factor that is proportional to the maximal radius R , independent of the network size. After finding a set of clusters with bounded radius, clusters that violate the weight constraint are further divided by a weight partition algorithm based on the scheme of Banerjee and Khuller in [18]. The weight partition algorithm guarantees a 3-approximation factor. As a result, the final solution satisfies the clustering constraints and its size is within a small constant factor of the optimum solution. Our simulations show that the proposed algorithms yield efficient partitions which are close to the optimum.

The rest of this paper is organized as follows. Section II presents the network model and needed definitions. Section III provides a general description of the proposed systems. This description is required for understanding the constraints of the clustering problem presented in Section IV. Section V describes the clustering algorithms while their approximation ratios are given in Section VI. Simulation results are presented in Section VII and Section VIII concludes this study. In the appendix, we present the proofs of some auxiliary lemmas of our approximation analysis.

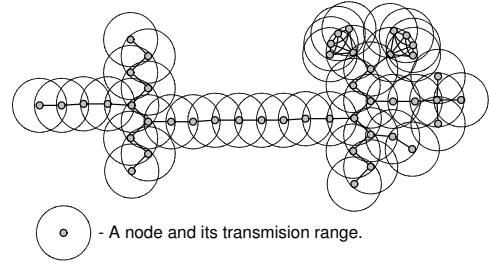


Fig. 1. An example of a wireless network and its unit disk graph representation.

II. THE NETWORK MODEL

We consider a static multi-hop wireless network represented by an undirected graph $G(V, E)$. Each node $v \in V$ represents a radio transceiver with a unique identification number (ID). At any given time a transceiver may either transmit or listen to a single wireless channel. The bandwidth of every wireless channel is W and it is divided into time slots of length Δ . Each node $v \in V$ has a circular transmission range with radius 1 and the *neighborhood* of v , denoted by $N(v)$, is the set of nodes that reside within its transmission range. Thus, there is a bi-directional wireless link between v and every node $u \in N(v) - \{v\}$, which is represented as an edge $(u, v) \in E$. Consequently, our network model is similar to the proximity model of unit disk graphs that is defined in [23]. An example of such graph is given in Figure 1. In the following, we use this simple network for illustrating the properties of the proposed algorithms.

In our model, the *shortest path* between nodes $u, v \in V$ is the path with the minimum number of hops between them and their *distance*, $d(u, v)$, is the number of hops in this path. Any subset of nodes $C \subseteq V$ is called a *cluster* and assume that there is a cluster-head $y \in C$ at each considered cluster C . The cluster nodes and the links between them define the *cluster graph* $G_C(C, E_C)$, where $E_C = \{(u, v) | u, v \in C \wedge (u, v) \in E\}$. A cluster is called *connected* if its corresponding cluster graph is connected. We denote by $d_C(v, u)$, $v, u \in C$, the shortest path between u and v in the cluster graph and the *cluster radius* is the maximal distance from y to any other node $v \in C$, i.e., $\max_{v \in C} d_C(y, v)$. In addition, every node $v \in V$ has a positive weight $w_v \ll W$ that represents its bandwidth requirement and we denote the total weight of a cluster C by $W_{sum}(C) = \sum_{v \in C} w_v$.

Let T be a spanning tree of a cluster graph $G_C(C, E_C)$ rooted at the cluster-head $y \in C$. The i -th *level* of the tree T is the set of nodes with hop distance i from y along the tree T and the *depth* of T is the index of its farthest level. A tree T is called a *shortest-path (SP) tree* if it is composed of the shortest paths from its root, y , to any other node $v \in C$. In this case the depth of T is also the cluster radius.

For any positive integer k , the k -*neighborhood* of a node v , denoted by $N_k(v)$, is the set of nodes with distance at most k from v in G , i.e., $N_k(v) = \{u | u \in V \wedge d(u, v) \leq k\}$. We define the k -th *power graph* of G , denoted by $G^k(V, E^k)$, as the graph with the nodes V and an edge between every pair of nodes $u, v \in V$ such that $d(u, v) \leq k$ in G . Recall that the k -neighborhood of a node v in G is the neighborhood of v in G^k .

For the sake of completeness we present here several graph theory definitions that we use in the paper. Given an undirected

graph $G(V, E)$, an *independent set* (IS) of G is a subset $S \subset V$ such that there is no edge between any pair of nodes in S . In unit disk graph the geographic distance between every pair of nodes in S is more than 1. A *dominating set* (DS) of G is a subset $S \subset V$ such that every node in $V - S$ has at least one neighbor in S and a *dominating independent set* (DIS) of G is a set $S \subset V$ that is both dominating and independent set. An approximation algorithm for an optimization problem Π provides an *approximation factor* of δ if for every instance of Π the solution returned by the approximation algorithm is within a factor of δ from the optimal solution. A *polynomial time approximation scheme* (PTAS) for problem Π is a polynomial time algorithm which for any given instance I of Π and $\epsilon > 0$, returns a solution within a factor of $(1 + \epsilon)$ times of the optimal solution for I .

III. THE SYSTEM DESCRIPTION

A. The Multi-Hop System

The aim of the proposed system is providing low-cost connection between a wireless network $G(V, E)$ and a fixed backbone network that satisfies *quality of service* (QoS) constraints. The QoS constraints are given in the form of minimal bandwidth requirements¹, w_v , of every node $v \in V$ and a bound D_P on the message propagation delay. The connection is obtained by equipping some nodes with direct links with a backbone network. Such node, termed *access point*, serves as a gateway between the nodes in its vicinities and the backbone. The efficiency of a proposed system is evaluated by the number of access points that it requires.

In this work we propose to minimize the number of access points by using multi-hop systems. We partition the wireless network into connected clusters and select a *cluster-head* at each cluster that serves as the access point of all its cluster nodes. For message delivery, we construct a spanning tree, termed a *delivery tree*, rooted at the cluster-head, where internal nodes serve as relays for connecting distinct nodes with the access-point. The tree-based delivery mechanism simplifies the message routing [7] and enables us to maximize the network utilization while satisfying the QoS constraints. In Example 1, we illustrate that the multi-hop systems need significantly less access points than the single-hop systems (where each node is included in the transmission range of its access point), even when an access point can serve only small number of users. This observation is also confirmed by our simulations in Section VII.

Example 1: Figure 2 presents the optimal single-hop and multi-hop systems for the graph presented in Figure 1. For the multi-hop systems, the cluster radius is bounded by $R = 5$ and the number of nodes in a cluster is bounded by 100 and 10, respectively. In this example, the optimal solutions were found by checking all the possibilities. \square

For designing effective multi-hop systems we should construct two mechanisms: (i) An efficient clustering algorithm that produces a small number of delivery trees that cover all the nodes and satisfy the given QoS constraints. (ii) A delivery mechanism at each cluster that maximizes the cluster throughput without violating the QoS constraints and with efficient channel utilization. In the rest of this section, we describe the delivery mechanism. This description is essential for understanding the constraints of the clustering algorithm, e.g., the maximal radius

¹Similarly, the weight may represent a lower bound on the average bandwidth requirement of a considered node.

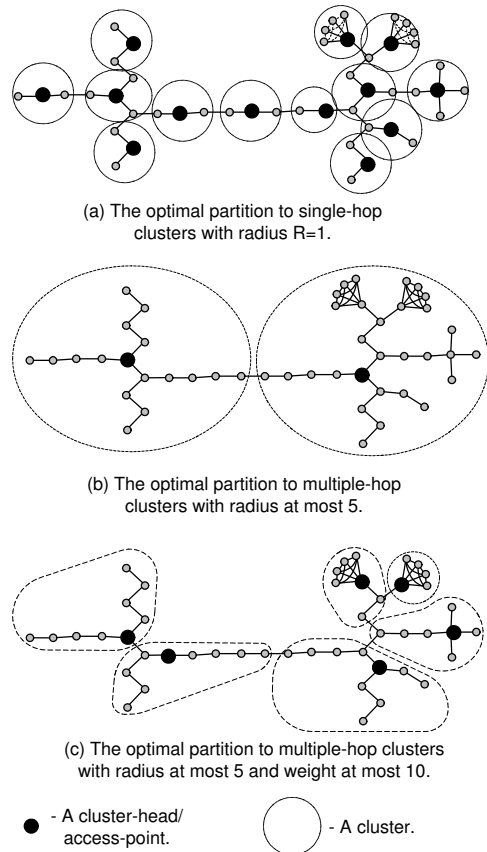


Fig. 2. Examples of single-hop and multi-hop systems.

of a cluster, R . For simplicity, we assume that the upstream and downstream bandwidth requirements of every node are the same² and the weight of each node $v \in V$, w_v , denotes its aggregated bandwidth requirements.

B. The Delivery Mechanism

Consider a cluster $C \subseteq V$ with a delivery tree $T(C, E')$ rooted at an access point $y \in C$ and assume that the tree depth is R . Since, all messages traverse through the access point, the throughput of the cluster is maximized when the access point is busy all the time with efficient data transmission. This goal is obtained by employing a collision free transmission scheme, where the access point³ coordinates all the transmissions inside the cluster by using slotted channels and polling mechanism. For the clarity of the presentation we assume that the slots are enumerated. Moreover, each node knows its parent and its descendant nodes in the delivery tree.

B.1 Downstream and Upstream Delivery

For efficient and simple message delivery, the slots are divided into upstream and downstream slots. Odd slots are used only for downstream data transmission, while upstream data is

²In practice, one of the traffic flows may dominant the other. Such asymmetry can be easily resolved in our scheme, for instance, by allocating slots with different sizes for upstream and downstream traffic flows.

³By definition, the access point is a single point of failure since it is the only node in the cluster that is connected to the backbone network. Therefore, a complicated distributed management is not justified.

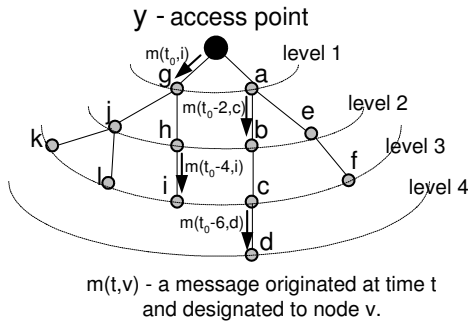


Fig. 3. An example of downstream transmissions at a given time slot t_0 .

forwarded only in even slots. Initially, let us describe the downstream transmissions and recall that every downstream message is originated by the access point. Consider a message from the access point y to node u that was originated at time t_0 and traverses along the path $P_{y,u} = \{v_0 = y, v_1, v_2, \dots, v_k = u\}$ in T . At each successive downstream slot, the message is forwarded from its current location to its successive nodes in the path $P_{y,u}$, until it reaches its destination. Thus, at slot t_0 the message is received by node v_1 , at slot $t_0 + 2$ the message is forwarded to node v_2 and so on, until it reaches node u at slot $t_0 + 2 \cdot (k - 1)$. The total propagation delay of the message in the system is $2 \cdot k - 1$ time slots. Since, a node may either receive or transmit at any given time, the delivery mechanism must ensure that when node v_i sends the message to node v_{i+1} , it is not suppose to receive another message. This property is obtained if the access point does not send two messages to the same child, v_1 , in two successive downstream slots.

Example 2: Figure 3 presents an example of a delivery tree in which four messages are forwarded simultaneously at a given time slot t_0 . Each message $m(t, v)$ is indicated by its first transmission time t (by the access point) and its destination node v . \square

Upstream transmission is done in a similar manner. The main difference is that upstream messages are originated by cluster nodes, $u \neq y$, and designated to the access point, y . For guaranteeing collision free operation, the access point allocates for each node *reserved slots* for its upstream transmission. This reserved slots' information is piggybacked to the downstream messages sent to each node and it specifies the number of slots that the node should wait before sending its next upstream message. This dynamic slot allocation method enables adaptive and fair bandwidth management that maximizes the cluster throughput. Moreover, when transmission collisions are prevented by deploying an efficient channel allocation method (as described in Sub-Section III-B.2), then the next property is satisfied.

Property 1: The maximal propagation delay of a message in the wireless network is at most $(2 \cdot R - 1) \cdot \Delta$, when Δ is the duration of a time slot.

Example 3: Figure 4 presents an example of message delivery along a given path $P = \{y, a, b, c, d\}$ during a period of 30 time slots. The access point manages two sessions with nodes c and d , and the bandwidth of each session is $W/4$. Recall that downstream messages carry the destination IDs, while upstream messages contain the source IDs. \square

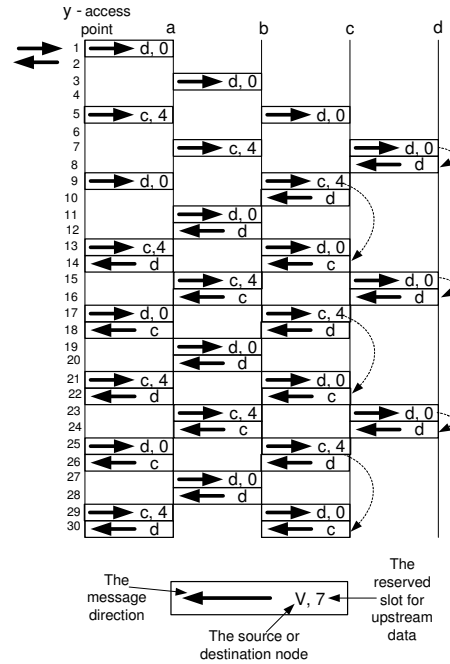


Fig. 4. An example of the message delivery of two active sessions during 30 time slots.

B.2 Synchronization and Channel Allocation

The proposed system maximizes the system throughput by preventing transmission collisions. Thus, at every time slot, each node has to know either its transmission or listening channel. Moreover, two nodes are not allowed to transmit simultaneously on the same channel if they are both neighbors of the same receiving-node. These requirements force us to make both inter-cluster and intra-cluster channel synchronization, where the first prevents interference between adjacent clusters and the second addresses interference from nodes in the same cluster. While inter-cluster synchronization can be easily obtained by using disjoint sets of channels for adjacent clusters, the intra-cluster channel allocation raises a challenging problem of synchronizing all the transmitting/receiving pairs at each slot without interfering to each other. This channel allocation, however, can be simplified by using the next property of the delivery mechanism.

Property 2: At any given time there is at most one transmitting-node at each level of the delivery tree.

This property results from the fact that at each time slot the access point can either transmit or receive a single message. Thus, for each level i of the tree, we allocate a single channel, F_i , that is used as the transmission channel by all the nodes in level i . Every non-transmitting node in level i , listens to channel F_{i-1} during downstream slots and to channel F_{i+1} during upstream slots. From Property 2 follows that there are no transmission collisions and each node know to which channel it has to listen at any time. A node u that received a message originated by or designated to one of the nodes in its sub-tree, transmits the message in the next upstream or downstream slot, respectively. In this method the number of required channel is at most R . Note that "channel reuse" techniques can be applied for further reducing the number of required channels. If, for instance, the delivery tree is also a shortest path tree and non-neighbor nodes do not interfere to each other, it can be shown that only three

wireless channels are required for each cluster.

B.3 Admission Control

In the proposed system a node is allowed to originate a new message only if it has acquired a reserved slot. Reserved slots are obtained only when a node receives a message from the access point. This raises the problem of initiating new sessions by the cluster nodes. To overcome this problem the access point periodically polls the cluster nodes that are not involved with active sessions. Since, the poll request and replay messages are very short, the access point may poll simultaneously several nodes for minimizing the polling overhead, *i.e.*, a set of nodes with the same parent v in T . The access point sends a single poll message to node v . Upon reception of the poll message, node v polls all its children by transmitting a single poll message. Each child returns a short reply in a corresponding mini-slot. This reply message indicates whether the child wants to initiate a new session and the required bandwidth. Node v collects the replies and originates a single reply message that is sent toward the access point. When the access point receives the reply message, it initiates the requested sessions and allocates the required resources.

This polling mechanism may also be used for time slot synchronization as every node needs to be synchronized only with its parent in the tree. Moreover, The polling mechanism is also used for fault detection as we describe in Section III-B.4.

B.4 Fault Management

In the considered system, each cluster is managed by a single access point. We assume that the access point is backed-up and it is connected to a reliable source of energy, while the other nodes are prone to failures. For failure recovery, the system is equipped with failure detection and failure recovery mechanisms. The failure detection mechanism also uses the polling method for failure detection. If the access point does not receive any reply after sending few poll messages, it assumes that one of the nodes along the message path has failed, and it sends relevant poll messages for identifying the malfunctioning node. Similarly, a node that has not received a poll message during a give time period, assumes that it is no longer connected with the access point and it starts listening to a pre-defined control channel for recovery messages.

Upon a failure, the access point modifies the delivery tree for reconnecting the detached nodes and sends proper recovery messages for reconstructing the delivery tree. If the malfunctioning node is a cut-node of the cluster graph⁴, the system attempts to attach the disconnected nodes to adjacent clusters.

IV. THE SYSTEM DESIGN PROBLEM

We now present the clustering problem that is raised from the design of the multi-hop systems. Consider a graph $G(V, E)$ as describe in Section II, with bandwidth requirements w_v for every node $v \in V$, capacity W to every wireless channel and a bound D_P on the message propagation delay. The *clustering problem* looks for the minimal number of disjoint delivery trees that cover all the nodes and satisfy three additional design requirements. For presenting these constraints we need the next definition.

Definition 1 (Relay-Load) : Consider a delivery tree T and for every non-root node v let $T_v(C', E')$ be the sub-tree of T

rooted at node v . The *relay-load* of node v in T , denoted by $h_v(T)$, is the total weight of all its descendant nodes in the delivery tree T , *i.e.*, $h_v(T) = \sum_{u \in C' - \{v\}} w_u$.

From the description of the delivery mechanism in Section III-B results that every delivery tree T must satisfy the following constraints. The aggregated bandwidth requirements of all the tree nodes must not exceed the capacity W of a wireless channel. According to Property 1, the depth of the delivery tree is at most $R = \lfloor \frac{D_P + \Delta}{2 \cdot \Delta} \rfloor$ for satisfying the delay constraint. As, every non-root node v needs to receive and retransmit every message that is originated by or designated to one of its descendant nodes, its relay-load is at most $(W - w_v)/2$.

Definition 2 (The clustering problem) : Given a graph $G(V, E)$, a maximal tree depth $R \in \mathcal{Z}^+$ and a maximal weight $W \in \mathcal{R}^+$. Find the smallest collection of sub-graphs (trees) $\mathcal{T} = \{T_1(C_1, E_1), \dots, T_m(C_m, E_m)\}$, $C_i \subseteq V$, $E_i \subseteq E$ that satisfy the following requirements.

1. **Tree Topology** - Every sub-graph $T_i(C_i, E_i)$ has a tree topology and let us denote by $y_i \in C_i$ the tree root.
2. **Coverage** - The trees contain all the graph nodes, *i.e.*, $\bigcup_{j=1}^m C_j = V$.
3. **Disjoint Trees**- Every node is included in a single tree, *i.e.*, $C_i \cap C_j = \emptyset$, for every $i \neq j$.
4. **Depth** - The depth of every tree $T_i(C_i, E_i) \in \mathcal{T}$ is at most R .
5. **Weight** - For every tree, $T_i(C_i, E_i) \in \mathcal{T}$, the total weight of all its nodes is at most W . *i.e.*, $W_{sum}(C_i) = \sum_{v \in C_i} w_v \leq W$.
6. **Relay-Load** - For every tree $T_i(C_i, E_i)$ and for every node $v \in C_i - \{y_i\}$, its relay-load is bounded by $h_v(T_i) \leq (W - w_v)/2$.

For the sake of simplicity, the set of access points is selected from the network nodes. This limitation can be easily resolved by adding an auxiliary node to the graph at each point where an access point can be deployed. The latter can be used only as a root of a delivery tree. This modification of the problem can be easily addressed by the algorithms presented in Section V.

Theorem 1: The clustering problem is NP-hard.

Proof: We prove this theorem by reducing any instance of the *dominating set* (DS) problem of unit disk graphs to the clustering problem. Consider a unit disk graph $G(V, E)$. We claim that the graph $G(V, E)$ has a dominating set of size k if and only if there is a partition of the graph to k trees that satisfy the requirements of Definition 2 with $W = \infty$ and $R = 1$. If the graph has a dominating set $S \subseteq V$ of size k , then by assigning every node $v \in V - S$ to one of its neighbor in S we obtain k trees that satisfy the requirements of Definition 2. If there is a partition of the graph into k trees then the set of all the tree roots defines a dominating set of size k . Since, the DS problem is NP-complete also for unit disk graphs [23] the theorem is satisfied. \square

A graph decomposition \mathcal{T} that satisfies the requirements of Definition 2 is called a *feasible partition* and every tree $T_i \in \mathcal{T}$ is called a *feasible delivery tree*. From the clustering problem formulation it is clear that the root of every feasible delivery tree can serve as an access point. However, the calculated trees are not necessarily the ones that minimize the relay-load of the cluster nodes. Minimizing the relay-load of all the non-root nodes is important for reducing the energy consumption of the nodes (especially in sensor networks) and for failure recovery. Therefore, after calculating the trees $\mathcal{T} = \{T_i(C_i, E_i)\}$ and their corresponding clusters, C_i , we would like to find at each cluster C_i

⁴A node is termed a cut-node if its removal leaves the graph disconnected.

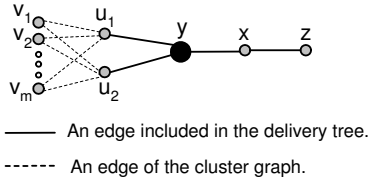


Fig. 5. The cluster graph for the proof of Theorem 1.

the access point and its corresponding delivery tree that minimize the maximal relay-load of its cluster nodes.

Definition 3 (The relay-load problem) : Given a cluster C , a corresponding cluster graph $G_C(C, E_C)$ and maximal depth R . Find a node $y \in C$ and a spanning tree T of $G_C(C, E_C)$ rooted at node y with depth at most R that minimizes the function, $\min_{T, y} \{ \max_{v \in C_i - \{y\}} h_v(T) \}$.

Theorem 2: The relay-load problem is NP-hard.

Proof: We prove this theorem by presenting a polynomial reduction from the *partition* problem [24] to the relay-load problem. Consider a set \mathcal{A} of $m > 2$ elements where each element $a_i \in \mathcal{A}$ has size $s_i \in \mathbb{Z}^+$, and let $X = \sum_{a_i \in \mathcal{A}} s_i / 2$. The partition problem looks for a sub set $\mathcal{A}' \subset \mathcal{A}$ such that $\sum_{a_i \in \mathcal{A}'} s_i = \sum_{a_i \in \mathcal{A} - \mathcal{A}'} s_i = X$. Consider the cluster graph $G_C(C, E_C)$ depicted in Figure 5 with $R = 2$ and let the weight of the nodes $\{y, u_1, u_2, x, z\}$ be 1. For every element $a_i \in \mathcal{A}$ we define a node v_i with weight $w_{v_i} = s_i$ that is adjacent to nodes u_1 and u_2 . 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 a spanning tree T of G with depth 2 such that the relay-load of every non-root node is at most X . Recall that node y must be the root of every calculated tree T for satisfying the depth requirement and the relay-load of node x , $h_x(T) = 1 < X$. Suppose that there is such a partition \mathcal{A}' . Then we construct the tree T such that every node v_i is attached to node u_1 if $a_i \in \mathcal{A}'$ and to node u_2 otherwise. Thus, $h_{u_1}(T) = h_{u_2}(T) = X$. Now, suppose that there is a tree T with maximal relay-load X . Thus, $h_{u_1}(T) = h_{u_2}(T) = X$. Let \mathcal{A}' be the set of elements represented by children of u_1 in T . The sum $\sum_{a_i \in \mathcal{A}'} s_i = X$ and this complete the proof. \square

The clustering and the relay-load problems can be formulated as a single integer program (IP). We view the integer program as an extended instance of the capacitated facility location problem (CFLP) [19] with additional constraints. The CFLP problem seeks for a minimum cost way for connecting clients to open facilities. A facility is termed *open* if it serves a non-empty set of clients and its "capacity" allows it to serve no more than a limited number of clients. In our instance every open facility represents a selected access point and it is associated with feasible delivery tree. Unlike the original CFLP, the selected facilities and the associated trees are required to meet also the other constraints of Definition 2.

Let $F \subseteq V$ and $C = V$ be the set of facilities and clients, respectively. We introduce a 0-1 variable y_i to indicate whether facility i is open. To indicate facility assignment, we define a 0-1 variable $x_{i,j}$ which has value 1 if and only if client (node) $j \in C$ is assigned to facility $i \in F$. We denote by P the set of paths in the graph with at most R hops. Let $P_{i,j} \subset P$ be the set of paths between facility $i \in F$ and client $j \in C$. For unique identification of the paths in $P_{i,j}$, each path $\pi_{i,j}^k \in P_{i,j}$ is associated with a unique index k . For every path $\pi_{i,j}^k \in P_{i,j}$,

$i \neq j$, let $\pi_{i,j'}^{k'} = \text{prefix}(\pi_{i,j}^k)$ be the path obtained by removing client j from the path $\pi_{i,j}^k$. Moreover, for every client j , let $Q_j \subset P$ be the set of paths that contain client j as an internal node. For constructing the delivery trees, we introduce a 0-1 variable $z_{i,j}^k$ that indicates whether path $\pi_{i,j}^k$ is selected to connect client $j \in C$ with facility $i \in F$. Among all the solutions that minimize the number of open facilities we select the one that also minimizes the maximal relay-load. For obtaining the secondary goal, we introduce the variable h that upper bounds the maximal relay-load. Since $h < W$, we set the cost of opening a facility to W . The objective function of our IP formulation is defined as follows.

$$\min \left\{ h + W \cdot \sum_{i \in F} y_i \right\}$$

s.t.

$$\forall j \in C : \sum_{i \in F} x_{i,j} = 1 \quad (1)$$

$$\forall i \in F : \sum_{j \in C} w_j x_{i,j} \leq W \quad (2)$$

$$\forall i \in F, j \in C : y_i \geq x_{i,j} \quad (3)$$

$$\forall j \in C : \sum_{i \in F, \pi_{i,j}^k \in P_{i,j}} z_{i,j}^k = 1 \quad (4)$$

$$\forall \pi_{i,j}^k \in P, \pi_{i,j'}^{k'} = \text{prefix}(\pi_{i,j}^k) : z_{i,j'}^{k'} \geq z_{i,j}^k \quad (5)$$

$$\forall j \in C : \sum_{\pi_{i,u}^k \in Q_j} w_u \cdot z_{i,u}^k \leq (W - w_j) / 2 \quad (6)$$

$$\forall j \in C : \sum_{\pi_{i,u}^k \in Q_j} w_u \cdot z_{i,u}^k \leq h \quad (7)$$

$$\forall i \in F : y_i \in \{0, 1\} \quad (8)$$

$$\forall i \in F, j \in C : x_{i,j} \in \{0, 1\} \quad (9)$$

$$\forall i \in F, j \in C, \pi_{i,j}^k \in P_{i,j} : z_{i,j}^k \in \{0, 1\} \quad (10)$$

We now show that the CFLP formulation above represents our clustering problem. The objective function is minimized by the collection \mathcal{T} with minimum cardinality that also minimizes the maximal relay-load of the non-access-point nodes. The first inequality ensures that each node is assigned to exactly one access point. (i.e., the coverage and disjoint trees requirements). Inequality (2) ensures that the total weight of a tree does not exceed the channel capacity (i.e., the weight requirement). Inequality (3) guarantees that a facility is open before assigning nodes to it. Inequality (4) ensures that there is a selected path for each client and inequality (5) guarantees that also the prefix-path of every selected path is also selected. As a result of these two inequalities, the set of paths initiated at an open facility i has a tree topology, i.e., the tree topology requirement is satisfied. Inequality (6) ensures that the aggregated weight of all the descendants of a given node will not exceed its maximal relay-load. Thus, the relay-load constraint also fulfilled. Since the considered paths in P have at most R hops, The calculated solution satisfies the depth requirement as well. Finally, Inequality (7) guarantees that h is an upper bound of the relay-load of every node that is not an open facility.

Unfortunately, due to the strict constraint on the maximal tree depth, we cannot use the various techniques for solving CFLP

problems with constant approximation factor, like the ones described in [19], [20]. From Theorem 1, we conclude that we cannot find algorithm with approximation factor better than the lower bound of the dominating set problem. Generally, this lower bound is $\ln(\Lambda)$, where Λ is the maximal degree of the graph nodes. The minimum dominating set problem remains NP-complete also for unit disk graphs [23]. However, in such graphs a solution within a constant factor of the optimum can be found.

V. THE CLUSTERING ALGORITHM

This section describes our clustering algorithm that satisfies all the requirements of Definition 2. The algorithm guarantees a solution within a constant factor of the optimum and it uses the following property,

Property 3: Consider a cluster C with cluster-head $y \in C$ and cluster graph $G_C(C, E)$. If for every node $v \in C$ the distance $d_C(y, v) \leq R$, then the depth of any SP tree rooted at node y is at most R , and we say that the cluster C satisfies the depth constraint.

The algorithm contains four stages. Initially, we select a small set of cluster-heads such that their R -neighborhoods cover all the nodes. From Property 3 follows that these clusters satisfy the depth requirement. In the second stage, each node is assigned to a single cluster without increasing the radius of each cluster graph. Then, at each cluster we select an SP tree rooted at the cluster head. If the selected tree violates the weight or the relay-load requirements, it is further divided into smaller trees (clusters) that satisfy all the requirements of Definition 2. Finally, at each cluster we use a simple heuristic for selecting an access point that reduces the maximal relay-load of the cluster nodes. We use the graph $G(V, E)$ depicted in Figure 1 for demonstrating the different stages of the algorithm, where $R = 5$, and $W = 10$ and the weight of each node $v \in V$ is $w_v = 1$. Correctness proof, approximation factor calculations and run-time complexity of the proposed algorithms are provided in the appendix.

A. Node Coverage with Bounded Radius Clusters

Consider a unit disk graph $G(V, E)$ as described in Section II. The *coverage algorithm* looks for the minimal number of cluster-heads such that their R -neighborhoods cover all the nodes. This is the same as looking for the minimum dominating set of the power graph $G^R(V, E^R)$. In this section we consider three coverage algorithms that guarantee different approximation ratios.

The first approach employs the greedy set cover (SC) algorithm of Chvatal [25] and it is termed the *greedy SC coverage algorithm*. This is an iterative algorithm that selects at each iteration the node whose R -neighborhood contains the maximal number of uncovered nodes. This algorithm guarantees a solution with in a factor of $\log(\Lambda)$ from the optimum, where Λ is the maximal degree of the nodes in the graph $G^R(V, E^R)$.

In the following we present two additional approximation algorithms that are based on Property 4.

Property 4: The R -neighborhood of any node $v \in V$ of the graph $G(V, E)$ [the neighborhood of node v in $G^R(V, E^R)$] is concentrated inside a circle of radius R around node v .

These algorithms guarantee approximation ratios that do not depend on the number of nodes in the graph or its topology. One algorithm is a polynomial time approximation scheme (PTAS)

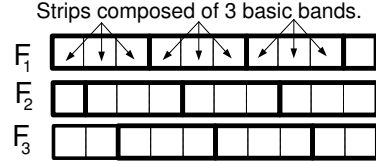


Fig. 6. An example of strip sets for $L = 3$ and 10 basic-bands.

based on the *shifting strategy* of Hochbaum and Maass [21], [22], which allows us to strike a balance between quality of approximation and its running time. The other algorithm is based on geometric properties of independent sets in unit disk graphs. The latter provides a fast solution with an error bound that linearly grows with R .

A.1 The Shifting Strategy Approach

The *shifting strategy* is a useful divide-and-conquer method for bounding the solution error when dealing with geometric problems that consider bounded size objects. In [21], [22], it is used for point-covering problems with bounded size disks. We describe this method for a plane, objects with diameter⁵ at most D and a shifting parameter L . Initially, the shifting strategy divides the plane into vertical *basic-bands* of width D . Then it uses the basic-bands to construct L different sets of *strips*, denoted by F_1, F_2, \dots, F_L . Each set contains strips of width $D \cdot L$ (or less) that cover the entire area. The set F_1 is defined by grouping every L consecutive basic-bands to a single strip of width $D \cdot L$ starting from the left most basic-band, as shown in Figure 6. The other sets are defined by shifting each time the $D \cdot L$ strips one basic-band to the right. Note that after L shifts the first set is received. Let A be an algorithm for calculating a solution in any strip of width at most $D \cdot L$. We use algorithm A for calculating a solution for every set $F_l, l \in [1, L]$. Among all the calculated solutions the best one is selected. As proven in [21], the shifting strategy ensures the following upper bound.

Theorem 3 (The Shifting Theorem) : Let δ_A be the approximation factor of algorithm A and let $\delta_{A,L}^{SH}$ be the approximation factor of the shifting strategy when using algorithm A and shifting parameter L . Then $\delta_{A,L}^{SH} \leq \delta_A \cdot (1 + \frac{1}{L})$.

Let us now describe our *shifting strategy coverage algorithm*. The latter uses a nested application of the shifting strategy for calculating a small set of cluster-heads of the graph G . Initially, It selects $D = 2 \cdot R$ as the width of a basic-band. From Property 4 results that each cluster is covered by at most two basic-bands. The algorithm calculates L sets of vertical strips of width $L \cdot D$, as we describe above. Then, it applies the shifting strategy again on the other dimension, and each strip is further divided into sets of squares each one of size $L \cdot D \times L \cdot D$. At each square, the algorithm finds the optimal set of cluster-heads by performing an exhaustive search, and selects the set of squares (strips) $F_l, l \in [1, L]$ with the minimal number of cluster-heads.

We turn to describe the approximation factor and the time complexity of the shifting strategy coverage algorithm. For the clarity of the presentation, the proofs of all the auxiliary lemmas are deferred to the appendix. According to the Shifting Theorem, at each strip a solution within a factor of $(1 + \frac{1}{L})$ is found.

⁵An object diameter is the maximal distance between any two point on the consider object.

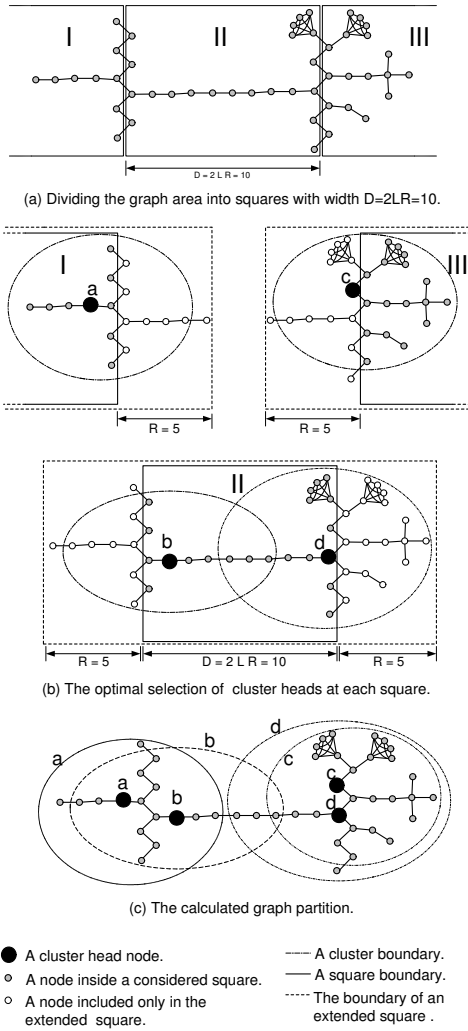


Fig. 7. An execution example of the shifting strategy with $R = 5$ and $L = 1$.

Since we use a nested application of the shifting strategy follows that,

Lemma 1: The approximation factor of the coverage algorithm is $(1 + \frac{1}{L})^2$.

We show, now, that for a fix fixed L and R the algorithm has a polynomial running time. Recall that the area of each square is $2 \cdot L \cdot R \times 2 \cdot L \cdot R$. Therefore, the size of the minimum cluster-head set is upper bounded.

Lemma 2: For every square of size $2 \cdot L \cdot R \times 2 \cdot L \cdot R$ there is a set with at most $(2^{\frac{3}{2}} \cdot L \cdot R + 1)^2$ clusters that covers all its nodes.

As a result, for every square the search routine needs to check at most $(2^{\frac{3}{2}} \cdot L \cdot R + 1)^2$ nodes, which can be done in a polynomial time for a fixed L and R . So, the described algorithm is a polynomial time approximation scheme (PTAS) as stated in Lemma 3.

Lemma 3: The complexity of the coverage algorithm is $O\left(L^2 \cdot |V|^{(2^{\frac{3}{2}} \cdot L \cdot R + 1)^2 + 2}\right)$.

Example 4: Consider the graph shown in Figure 1 and an execution of the shifting strategy coverage algorithm with shifting parameter $L = 1$, so there is only one set of squares to consider. The graph is divided into 3 squares with $D = 2 \cdot L \cdot R =$

$2 \cdot 1 \cdot 5 = 10$, as depicted in Figure 7-(a). Figure 7-(b) presents the optimal sets of cluster-heads that cover the nodes of each square. As illustrated by square III, such set may include nodes outside the considered square boundaries (node c in this example). However, the geometric distance of such cluster-heads from the square boundary is at most R . Therefore, each square is surrounded by a ring of width R around it, that defines an *extended square*, and its optimal cluster-head set is selected from the nodes in this square. Figure 7-(c) depicts the final partition with 4 clusters, which is $1 + \frac{1}{L} = 2$ times the size of the optimal partition. \square

Finally, we present an improvement for the shifting strategy, termed the *overlap coverage improvement*. The later does not affect the approximation factor, but it significantly improves its average performance, as shown in Section VII. Let $F = \{f_1, f_2, \dots, f_k\}$ be a set of disjoint strips (or squares) that covers the considered area and let Y_i be the selected cluster-heads for strip f_i . Recall that for any adjacent pair of strips f_i and f_{i+1} , the clusters defined by Y_i may also cover some nodes of f_{i+1} . Thus, if the sets Y_i and Y_{i+1} are calculated sequentially, the size of Y_{i+1} can be reduced by selecting the minimal set of clusters that covers "uncovered" nodes in f_{i+1} .

Example 5: Consider the partition to squares that is described in Example 4, and let us illustrate an execution of the overlap coverage improvement method for this case. Like in Figure 7-(b), node a is selects as the head of the single cluster that covers all the nodes in Square I. Since, this cluster covers also some nodes in Square II, it is sufficient to choose node d as a cluster head for covering the remaining nodes in Square II. Moreover, the latter covers also all the nodes in Square III. Thus, the overlap coverage improvement method has found a solution with only 2 clusters, which is an optimal solution. \square

A.2 The Dominating Independent Set Approach

The shifting strategy enables us to find a near optimal solution with the expense of a high running time. In this section, we present a "fast" greedy algorithm that selects a dominating independent set (DIS) of the power graph $G^R(V, E^R)$ as the set of cluster-heads. The *greedy DIS coverage algorithm* guarantees an approximation factor that is linear with R .

Let H_v be the cluster graph defined by the R -neighborhood of any node $v \in V$ and let H_v^R be the R -th power graph of H_v . We define by $M(R)$ the upper bound on the size of any independent set of H_v^R . As we prove in the appendix,

Lemma 4: The size of any independent set of the graph H_v^R is at most $M(R) \leq 8 \cdot R + 10$.

In [18] the authors show that when $R = 1$ then $M(1) = 5$.

Lemma 5: Let IS be any independent set of the graph $G^R(V, E^R)$ and let DS^* be its minimum dominating set. Then, $|IS| \leq M(R) \cdot |DS^*|$.

From Lemma 5 follows that any dominating independent set of G^R is within a factor of $M(R)$ from the minimum dominating set of G^R .

To select a small dominating independent set Y we use a greedy algorithm similar to the greedy SC algorithm presented in [25]. Let Y be the set of selected cluster-heads, let $\bar{V} \subseteq V$ be the set of uncovered nodes and let $n_v = |N_R(v) \cap \bar{V}|$ be the number of uncovered nodes in the R -neighborhood of v . Initially, $Y \leftarrow \emptyset$, $\bar{V} \leftarrow V$ and for every $v \in V$, $n_v \leftarrow |N_R(v)|$. At each iteration, the algorithm selects the node $u \in \bar{V}$ with

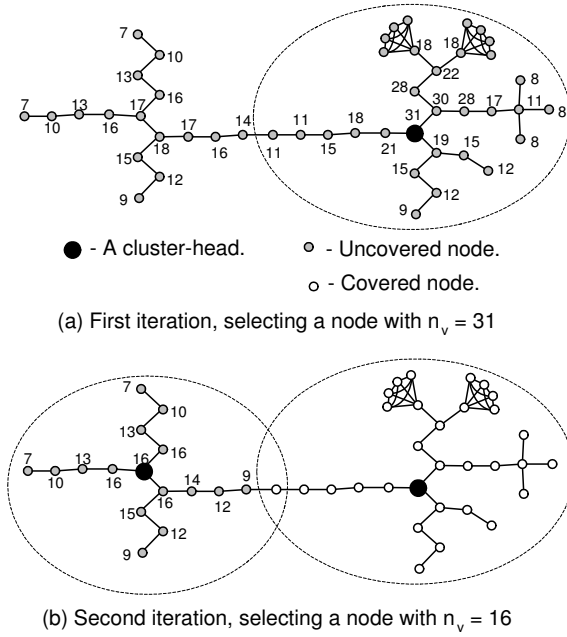


Fig. 8. An execution example of the greedy DIS coverage algorithm.

the maximal n_u value and adds it to Y . Recall that the R -neighborhood of u contains the maximal number of uncovered nodes from all the nodes in \bar{V} . Then, the R -neighborhood of u is removed from the set \bar{V} and for every remaining node $v \in \bar{V}$ its n_v value is recalculated. This step is done until all the nodes are covered, i.e., $\bar{V} = \emptyset$.

Example 6: For the graph depicted in Figure 1, the greedy algorithm calculates a set with 2 cluster-heads, as depicted in Figure 8. The number near each node represents its current n_v value. \square

Recall that at each iteration, the algorithm selects a node $v \in \bar{V}$ from the set of uncovered nodes. Thus, the distance of v from any other node $y \in Y$ is more than R , i.e., $d(v, y) > R$. Given that the algorithm ends when $\bar{V} = \emptyset$, every node is included in the R -neighborhood of a node in Y . As a result, the final set Y is indeed a DIS of $G^R(V, E^R)$ and greedy DIS algorithm satisfies Corollary 1.

Corollary 1: The approximation factor of the greedy DIS coverage algorithm is $M(R)$ and its time complexity is $O(|V|^3)$.

B. Creating Disjoint Clusters

The coverage algorithms in the first stage finds small number of clusters that contain all the graph nodes, however these clusters may overlap. In the second stage, the *cluster refinement algorithm* constructs disjoint connected clusters out of them. Let Y be the set of cluster-heads and assume that every cluster is represented by its cluster-head $y_i \in Y$. The algorithm associates each node with its closest cluster-head $y_i \in Y$, and breaks ties by selecting the cluster-head with the lowest ID. Since, all the nodes are covered by the R -neighborhoods defined by Y , the selected cluster-head of each node is included in its R -neighborhood and their distance is at most R . Hence, as we prove in the appendix,

Lemma 6: The cluster refinement algorithm defines $|Y|$ disjoint connected clusters that contain all the graph nodes and satisfy the depth requirement.

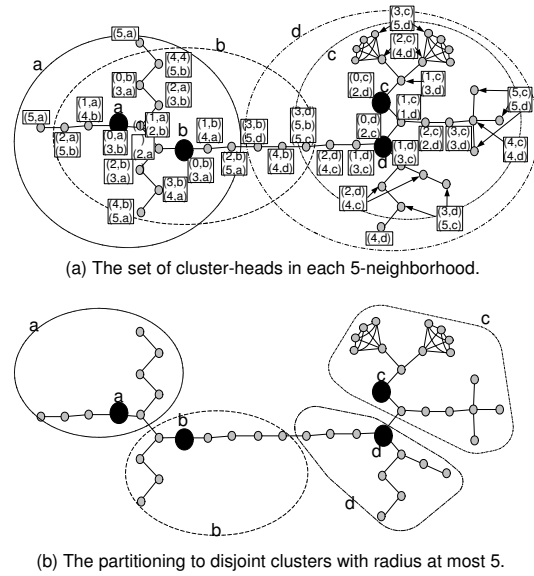


Fig. 9. An execution example of the partition refinement algorithm.

Example 7: In Figure 9 we demonstrate an execution of the cluster refinement algorithm when the input is the four cluster-heads described in Figure 7-(c). Figure 9-(a) presents, for each node v , the list of cluster-heads in its R -neighborhood sorted according to their distance from v . The partition to disjoint clusters is presented in Figure 9-(b). \square

C. Satisfying the Weight and the Relay-load Requirements

In the third stage, the *weight partition algorithm* selects an SP tree for each cluster rooted at the cluster-head. If the SP tree violates either the weight or the relay-load constraints it is divided into smaller sub-trees that meet all the requirements. Our algorithm extends the clustering scheme of Banerjee and Khuller [18] and uses the next property of unit disk graphs, proven in [18].

Property 5: The maximum independent set of the graph defined by the neighborhood, $N(u)$, of any node u has at most 5 nodes, i.e., $M(1) = 5$.

Consider a cluster C with cluster head $y \in C$. Initially, the algorithm calculates an SP tree, T , of the cluster graph rooted at node y . Recall that the depth of T is at most R and let us denote by T_u the sub-tree of T rooted at node u . Then, the algorithm performs a post-order tour of the tree, in which it calculates the total weight, $W_{tree}(u)$, of every sub-tree T_u , $u \in C$. Each time, a sub-tree T_u that violates the relay-load requirement is found, i.e., $W_{tree}(u) > (W + w_u)/2$, it is removed from T and becomes a separate delivery tree rooted at node u . Since, the relay-load constraint is specified only for non-cluster-head nodes, the new tree satisfies the relay-load requirement. If, in addition, the total weight of T_u exceeds the weight constraint, we use a *local pruning routine* for dividing T_u into minimal number of trees that satisfies also the weight requirement.

Consider a sub-tree T_u with weight more than W and let K_u be the children of u . We denote by $W_{tree}(v) = \sum_{x \in T_v} w_x$ the total weight of the nodes in the sub-tree T_v . For every set of nodes S let $W_{tree}(S) = \sum_{v \in S} W_{tree}(v)$. The local pruning routine minimizes the number of newly generated trees in the following way. It divides K_u into a small number of sets, such

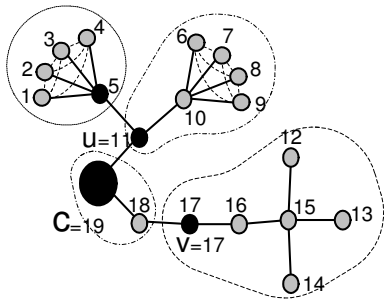


Fig. 10. An execution example of the weight partition algorithm.

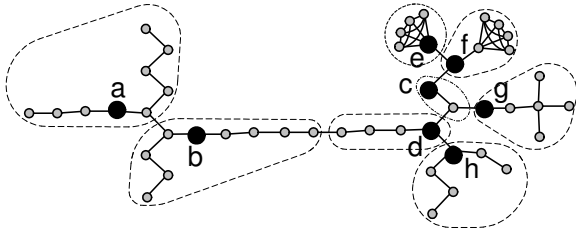


Fig. 11. The final partition of the graph.

that each set S and its related sub-trees T_v , $v \in S$ induce a tree that meets the depth, weight and relay-load requirements. While the weight of the tree T_u is more than W , $W_{tree}(u) > W$, the routine selects the set S with the maximal $W_{tree}(S)$ weight, removes S and its associated sub-trees from T_u and updates the tree weight, *i.e.*, $W_{tree}(u) \leftarrow W_{tree}(u) - W_{tree}(S)$. The set S and its attached sub-trees become a new tree (cluster).

It is clear, that an efficient partition of K_u into a small number of sets is essential for achieving low approximation factor. The local pruning routine minimizes the numbers of sets by using an iterative method that creates a single set at each iteration. Let $Q \subseteq K_u$ be the set of nodes that are not assigned to any set S . At the beginning $Q \leftarrow K_u$. While Q is not empty the routine performs the following loop. It selects a *seed node*, $v \in Q$, removes it from Q and creates a new set $S = \{v\}$ with $W_{tree}(S) \leftarrow W_{tree}(v)$. While there is any node $x \in N(v) \cap Q$ such that $W_{tree}(x) + W_{tree}(S) \leq W$, the routine adds x to the set S , removes x from Q and updates $W_{tree}(S) \leftarrow W_{tree}(S) + W_{tree}(x)$. At the end, by connecting node v with the other nodes in S , a new tree is defined. Thus, the trees calculated by the weight partition algorithm satisfy all the requirements of the clustering problem in Definition 2.

Example 8: Consider the 4 clusters calculated in Example 7 and let $W = 10$. In this example, cluster c violates the weight requirement and the SP tree of cluster d violates the relay-load constraint. We illustrate the weight partition of cluster c in Figure 10. The solid lines represent the SP-tree T and the nodes are enumerated according to the post-order-tour starting from node c . The sub-tree, T_u , routed at node u (number 11) has a weight $W_{tree}(u) = 11 > W$. Therefore, it violates both the relay-load and the weight requirements. The local pruning routine overcomes this weight violation by removing T_u from T and dividing it into two smaller clusters. Similarly, the tree T_v (node $v = 17$) violates the relay-load requirement, as $W_{tree}(T_v) = 6 > (10 + 1)/2 = 5.5$, and it is removed from T . Consequently, the entire graph is divided into eight clusters that meet all the requirements, as depicted in Figure 11. \square

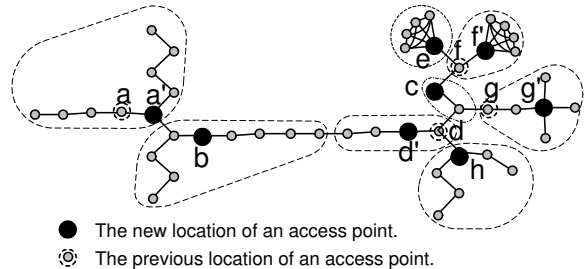


Fig. 12. The final partition of the graph.

| Evaluated alg. ($R = 1$) | $L = 1$ | $L = 2$ | $L = 3$ |
|----------------------------|---------|---------|---------|
| Shift alg. | 277 | 252 | 231 |
| Improved Shift alg. | 243 | 229 | 220 |

TABLE I

THE NUMBER OF CALCULATED CLUSTERS BY THE SHIFTING ALGORITHM FOR $R = 1$ AND $L \in [1, 3]$ WITHOUT WEIGHT CONSTRAINT.

We now present the approximation factor of the weight partition algorithm. This factor is independent of the used heuristics for selecting the SP tree or the seed nodes and the proofs of Lemmas 7 and 8 are given in the appendix.

Lemma 7: The weight partition algorithm divides the cluster graph into disjoint trees that contain all the cluster nodes and satisfy the depth, relay-load and the weight constraints.

Lemma 8: Let assume⁶ that for every $v \in V$ $w_v \leq W/3$. Then the approximation factor of the weight partition algorithm is 3.

Corollary 2: The weight partition algorithm creates at most $3 \cdot OPT$ new clusters, where OPT is the optimal cluster partition of the given graph $G(V, E)$.

Moreover, the algorithm running time is $O(|V|^3)$.

D. Reducing the Maximal Relay Load

So far, we described the algorithms for partitioning the graph into small number of trees that satisfy the clustering requirements of Definition 2. Each tree $T_i(C_i, E_i)$ represents a cluster of nodes C_i served by an access point that is located at the tree root, $y_i \in C_i$. However, a calculated tree T_i is not necessarily the optimal tree for minimizing the maximal relay-load of the cluster nodes. For instance, in Figure 11, each one of the tree roots of the clusters d , f and g has a single child. Thus, we can reduce the maximal relay-load at each one of these clusters by selecting as an access point a node that is closer to the tree center. According to Theorem 2, finding the optimal delivery tree of a cluster is NP-hard. Here, we propose a simple heuristic for reducing the maximal relay load. The heuristic considers all the nodes $v \in C_i$ that can serve as root of the tree T_i without violating the depth requirement, and selects as the access point the one that minimizes the maximal relay-load.

Example 9: Figure 12 presents the new locations of the access points. The current maximal relay-loads of clusters e and g are 0 and 2, respectively (instead of 4 before). \square

Another possibility is to employ the load-balancing heuristic that is presented in [7].

⁶If this assumption is not satisfied it can be shown that the algorithm approximation factor is 4.

VI. THE APPROXIMATION FACTOR OF THE CLUSTERING ALGORITHMS

This section presents the approximation ratios of the proposed clustering algorithms. The algorithms are termed according to the coverage scheme that they apply; the *shifting clustering algorithm*, the *greedy DIS clustering algorithm* and the *greedy SC clustering algorithm*.

Theorem 4: The approximation factor of the shifting clustering algorithm is $(1 + \frac{1}{L})^2 + 3$, where L is the shifting parameter.

Proof: Let OPT be the size of the optimal partition. According to Lemma 1, the number of clusters produced by coverage algorithm is at most $(1 + \frac{1}{L})^2 \cdot OPT$. From Lemma 6, the cluster refinement algorithm does not increase the number of clusters. From Lemma 8 and Corollary 2 follows that the number of clusters added by the weight partition algorithm is at most $3 \cdot OPT$. Thus, the total number of clusters is at most $[(1 + \frac{1}{L})^2 + 3] \cdot OPT$. \square

Moreover, the time complexity of shifting clustering algorithm is $O\left(L^2 \cdot |V| (2^{\frac{R}{2}} \cdot L \cdot R + 1)^2 + 2\right)$.

Theorem 5: The approximation factor of greedy DIS clustering algorithm is $M(R) + 3 = 8 \cdot R + 13$, where R is the maximal depth.

Theorem 6: The approximation factor of greedy SC clustering algorithm is $\log(\Lambda) + 3$, where Λ is the size of the maximal R -neighborhood.

The proofs of Theorems 5 and 6 are similar to the proof of Theorem 4 and the running time of the greedy algorithms is $O(|V|^3)$. Although, these algorithms have different approximation ratios, on average they yield similar results, as we show in Section VII.

VII. SIMULATION RESULTS

We compared by simulations the performance of the three algorithms proposed in this work and two other algorithms described in the literature. The algorithms are evaluated by the number of clusters that they produce. In our simulation we evaluated first the efficiency of the coverage algorithms and then we evaluated the sizes of the final solutions that satisfy also the weight and the relay-load constraints. The simulated coverage algorithms are; the *greedy SC algorithm* presented in Section V-A, the *shifting algorithm* described in Section V-A.1 with and without the *overlap coverage improvement*⁷, the *greedy DIS algorithm* described in Section V-A.2, a *naive DIS algorithm* that finds arbitrary DIS sets, and the *D-cluster algorithm* presented in [15]. The latter uses the node IDs for selecting the cluster-heads and constructing disjoint clusters that each one is included in the D -neighborhood of its clusters head. Selected typical results of our simulations are given in Figures 13–15 and Table 1 for different values of L , R and W . We present the number of clusters produced by the different algorithms for a single network with 1000 nodes, that are uniformly distributed over a square area of size 30×30 . The transmission range of very node is a circle with radius 1 and the bandwidth requirement of each node, w_v , is also 1. Additional simulations with various sizes produced very similar results.

Initially, we evaluated the effect of the shifting parameter on the performance of the two variants of the shifting algorithm with $R = 1$ and unbounded cluster weight. Table 1 summarizes

⁷The “improved shift alg” denotes the shifting algorithm with the overlap coverage improvement.

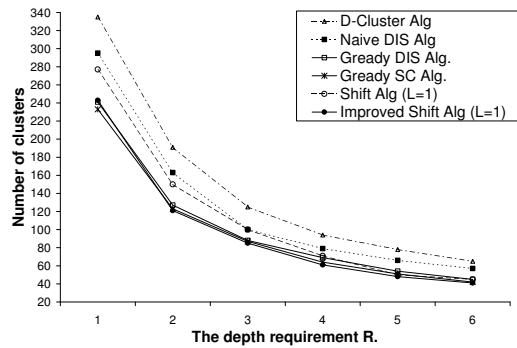


Fig. 13. A comparison between the different coverage algorithms (without weight constraint).

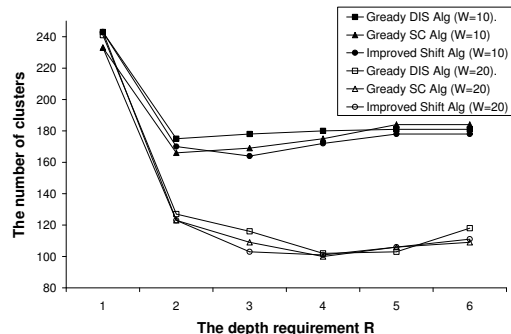


Fig. 14. A comparison between the different algorithms with weight constraint $W = 10$ and 20 .

our results and shows that indeed better results are obtained by increasing the shifting parameter L . However, this improvement is minor relative to the significantly expansion of the simulation running time. From Table 1, we conclude that the improved shifting algorithm obtains near optimal results also for $L = 1$ and we use this value in the rest of our discussion.

Figure 13 presents the number of clusters calculated by the different coverage algorithms without the weight constraint. It shows that the greedy SC, the greedy DIS and the improved shifting algorithms yield very similar results close to the optimal ones and outperform the D-cluster and the naive DIS algorithms (therefore, we omit the latter two from the following charts). Moreover, the improved shifting algorithm finds slightly better solution than the two greedy algorithms. Figure 14 confirms this observation also when $W = 10$ and 20 . Figure 14 also shows that the number of required access points is significantly smaller in multi-hop systems than in single-hop systems, even if W is small. Finally, Figure 15 presents the number of clusters calculated by the greedy DIS alg. for various values of W and R .

Moreover, Figures 14 and 15 present an unexpected result. Intuitively, for a given weight W , we expect the number of clusters to decrease by increasing the value of R until it converges to some value. In practice, our simulations show that for every W there is an optimal depth, R^* , that minimizes the number of clusters and after this point more clusters are created. As an example, for $W = 10$ the optimal depth is $R^* = 2$ or 3 (depend on the applied algorithm), while for $W = 20$ the optimal depth is $R^* = 4$ or 5 . This phenomenon results from the efficiency of the weight partition algorithm. Since, the weight partition algorithm

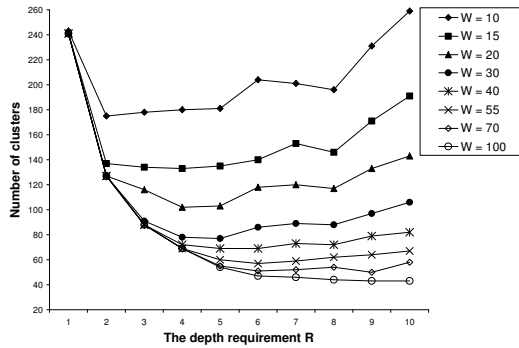


Fig. 15. Performance evaluation of the greedy DIS algorithm for different values of R and W .

allows the average cluster size to be as small as $W/3$, it produces clusters that are small in comparison to those of the optimal partition. As a result, a large value of R yields smaller number of clusters with radius R (calculated by a coverage algorithm), but these clusters exceed the weight constraint and are further divided into many small clusters. Consequently, we conclude that efficient solution requires a balance between the weight and the depth parameters.

VIII. SUMMARY AND FUTURE WORK

This work presents an efficient and low-cost infrastructure for connecting a static multi-hop wireless network with a fixed backbone network. This infrastructure can be used for reducing the cost of BWA systems and for accessing sensor networks. It presents clustering algorithms that partition a given wireless network into a forest with small number of trees, when each tree serves as delivery tree of a single access point. For QoS assurance, these trees also satisfy depth, weight and relay-load constraints. The work also describes an efficient delivery mechanism that maximizes the cluster throughput without violating the given QoS constraints. Although, the paper consider most of the design aspects of the proposed infrastructure, it leaves some open questions. For instance, the design of a recovery mechanism and finding an efficient algorithm for the relay-load problem.

IX. ACKNOWLEDGEMENT

I would like to thank Bulent Yener for useful discussions and to Eran Gabber, Rajeev Rastogi and Mark Smith for their helpful comments.

REFERENCES

- [1] H. Bolcskei, A. J. Paulraj, K. V. S. Hari, R. Nabar and W. W. Lu. "Fixed Broadband Wireless Access: State of the Art, Challenges and future Directions". *Proc. IEEE Comm. Magazine*, Vol. 39, No. 1, January 2001.
- [2] O. Momtahan And H. Hashemi. "A Comparative Evaluation of DECT, PACS and PHS Standards for Wireless Local Loop Applications". *Proc. IEEE Comm. Magazine*, Vol. 39, No. 5, May 2001.
- [3] G. Asada, M. Dong, T. S. Lin, F. Newberg, G. Pottie, W. J. Kaiser, and H. O. Marcu. "Wireless Integrated Network Sensors: Low Power Systems on a Chip", *Proc. of European Solid State Circuits Conference*, 1998.
- [4] W. R. Heinzelman, J. Kulik and H. Balakrishnan. "Adaptive Protocol for Information Dissemination in Wireless Sensor Networks", *Proc. of ACM MOBICOM'99*, 1999.
- [5] D. A. Beyer, M. D. Vestrich and J. J. Garcia-Luna-Aceves, "The Rooftop Community Network: Free, High-Speed Network Access for Communi-

ties", a chapter in "The First 100 Feet: Options for Internet and Broadband Access" Ed. D. Hurley and J. H. Keller. *MIT Press: Cambridge* 1999.

- [6] "Wireless Networking Reference - Community Wireless/Rooftop Systems", http://www.practicallynetworked.com/tools/wireless_articles_community.htm.
- [7] P.-H. Hsiao, A. Hwang, H. T. Kung and Dario Vlah. "Load Balancing Routing for Wireless Access Networks". *Proc. of IEEE INFOCOM '01*, 2001.
- [8] R. Krishnan, R. Ramanathan, and M. Steenstrup, "Optimization Algorithms for Large Self-Structuring Networks", *Proc. of IEEE INFOCOM '99*, 1999.
- [9] B. Awerbuch, B. Berger, L. Cowen, and D. Peleg, "Low Diameter Graph Decomposition is in NC", *Springer-Verlag, Lecture Notes in Computer Science*, Volume LNCS-621, pp. 83-93, 1992.
- [10] N. Linial and M. Saks, "Low Diameter Graph Decompositions", *Combinatorica*, Vol. 13, No. 4, 1993.
- [11] B. Awerbuch, B. Berger, L. Cowen, and D. Peleg, "Fast distributed network decomposition and covers", *Journal of Parallel and Distributed Computing*, Vol. 39, No. 2, 1996.
- [12] D. J. Baker and A. Ephremides. "The Architectural Organization of a Mobile Radio Network via Distributed Algorithm", *IEEE Trans. on Comm.*, Vol. 29(11), pp. 1694-1701, 1981.
- [13] C. R. Lin and M. Gerla, "Adaptive Clustering for Mobile Wireless Networks", *IEEE Journal of Selected Areas in Communications*, Vol. 15, No. 7, 1997.
- [14] R. Ramanathan and M. Steenstrup, "Hierarchically-Organized, Multihop Mobile Wireless Networks for Quality-of-Service Support", *Mobile Networks and Applications*, Vol. 3, No. 1, 1998.
- [15] A. D. Amis and R. Prakash and T. H. V. Vuong and D. T. Huynh. "Max-Min D-Cluster Formation in Wireless Ad Hoc Networks", *Proc. of IEEE INFOCOM 2000*, 2000.
- [16] B. Liang and Z. J. Haas, "Virtual Backbone Generation and Maintenance in Ad Hoc Network Mobility Management", *Proc. of IEEE INFOCOM*, 2000.
- [17] *IEEE Journal of Selected Areas in Communications*, Vol. 17, No. 8, Aug. 1999.
- [18] S. Banerjee and S. Khuller. "A Clustering Scheme for Hierarchical Control in Wireless Networks". *Proc. IEEE INFOCOM 2001*.
- [19] D. B. Shmoys, E. Tardos and K. Aardal. "Approximation Algorithms for Facility Location Problems", *Proc. of ACM Symposium on Theory of Computing*, 1997.
- [20] K. Jain and V. V. Vazirani. "Primal-Dual Approximation Algorithms for Metric Facility Location and k -Median Problems", *Proc. of IEEE Symposium on Foundations of Computer Science*, 1999.
- [21] D. S. Hochbaum and W. Maass, "Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI", *Journal of the ACM*, 32-1, 1985.
- [22] D. S. Hochbaum, editor, "Approximation Algorithm for NP-Hard Problem", *PWS Publishing Company*, 1997.
- [23] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit Disk Graphs. *Discrete Mathematics*, 86:165-177, 1990.
- [24] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-completeness. *Freeman Publication*, New York, 1979.
- [25] V. Chvatal, "A Greedy Heuristic for the Set-Covering Problem", *Math. of Operation Research*, Vol. 4, No. 3, 1979.

APPENDIX

I. CORRECTNESS AND APPROXIMATION ANALYSIS

This appendix proves the correctness of the clustering algorithms and presents their running time and approximation ratios.

A. The Shifting Strategy Approach

Consider the *shifting strategy coverage algorithm* given in Section V-A.1, with a basic band width $D = 2 \cdot R$ and a shifting parameter L .

Lemma 1: The approximation factor of the coverage algorithm is $(1 + \frac{1}{L})^2$.

Proof: According to the Shifting Theorem [21], the approximation factor of the shifting strategy is $\delta_A \cdot (1 + \frac{1}{L})$, where δ_A is the

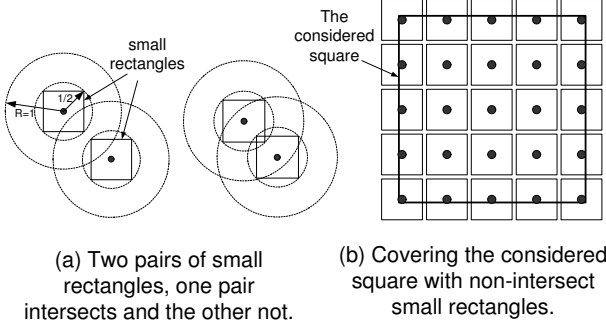


Fig. 16. The maximal number of rectangles in a 3×3 -square.

approximation factor of algorithm A , used for finding the solutions for strips of width $L \times D$. The proposed algorithm is based on nested application of the shifting strategy of depth 2. Each strip is further divided into squares. Thus, the calculated solution is within a factor of $\delta_B \cdot (1 + \frac{1}{L})^2$ of the optimal solution, where δ_B is the approximation factor of the square search routine. Since, the square search routine finds the optimal cluster-head set of any given square, results that $\delta_B = 1$. \square

Lemma 2: For every square of size $2 \cdot L \cdot R \times 2 \cdot L \cdot R$ there is a set with at most $(2^{\frac{3}{2}} \cdot L \cdot R + 1)^2$ clusters that covers all its nodes.

Proof: Our proof is based on geometric considerations. We claim that the maximal number of points with distance greater than 1 between each other that can be marked in a square of size $k \times k$ is at most $(\sqrt{2} \cdot k + 1)^2$. We encapsulate every point inside a small rectangle of size $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$, where the consider point is in its center. Note that if two small rectangles intersect then the distance between their center is at most 1, $((\frac{1}{\sqrt{2}})^2 + (\frac{1}{\sqrt{2}})^2 = 1)$. The maximal number of non-intersecting rectangles that can be arranged inside a square of size $k \times k$ is at most $(\sqrt{2} \cdot k + 1)^2$ when we arrange them in straight row and columns, as depicted in Figure 16. Now consider the optimal cluster-head selection of a given square. The number of nodes in this set is maximized if there is a single node in every R -neighborhood, which is the cluster-head. As a result, the maximum number of clusters in the optimal partition of any square of size $2 \cdot L \cdot R \times 2 \cdot L \cdot R$ is at most $(2^{\frac{3}{2}} \cdot L \cdot R + 1)^2$. \square

Lemma 3: The complexity of the coverage algorithm is $O(L^2 \cdot |V|^{(2^{\frac{3}{2}} \cdot L \cdot R + 1)^2 + 2})$.

Proof: Consider a given square. Basing on Lemma 2, the square search routine checks all the combination of at most $(2^{\frac{3}{2}} \cdot L \cdot R + 1)^2$ nodes for finding the optimal cluster-head set that covers the square's nodes. Thus, the complexity of the square partition algorithm is $O(|V|^{(2^{\frac{3}{2}} \cdot L \cdot R + 1)^2 + 1})$. The square partition algorithm is called at most $|V| \cdot L^2$ times. Hence, the complexity of the algorithm is $O(L^2 \cdot |V|^{(2^{\frac{3}{2}} \cdot L \cdot R + 1)^2 + 2})$. \square

B. The Dominating Independent Set Approach

In the sequel we calculate the approximation factor of the greedy DIS coverage algorithm described in Section V-A.2. Consider a graph $G(V, E)$ and a depth constraint R . Let $H_v(N_R(v), E(v))$ be the cluster graph defined by the R -

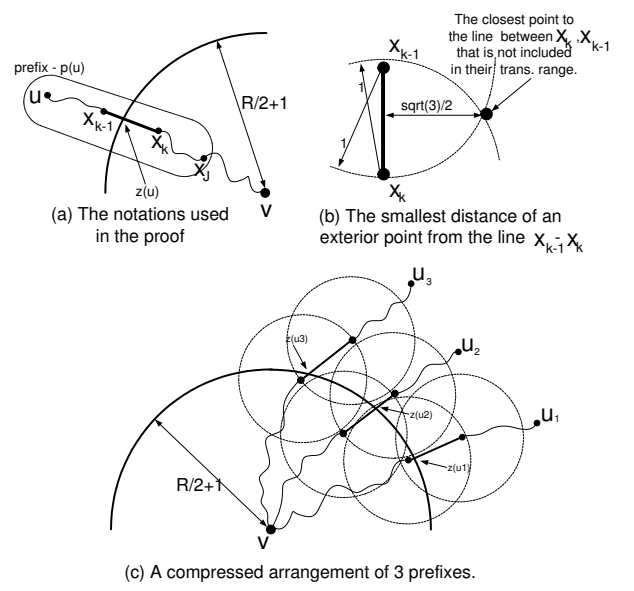


Fig. 17. Notations and figures used in the proof of Lemma 4.

neighborhood of any node $v \in V$ and let H_v^R be the R -th power graph of H_v .

Lemma 4: The size of any independent set of the graph H_v^R is at most $M(R) \leq 8 \cdot R + 10$.

Proof: In [18], the authors have shown that $M(1) = 5$. Thus we need to prove this lemma for $R > 1$. First we prove this lemma for the case of $R = 2$, then we extend this result for and $R \geq 3$. We turn to the case of $R = 2$. Consider a graph H_v and lets IS be its maximum independent set of its power graph H_v^2 . recall that the distances between any pair of nodes in IS is at least 1. We assume that every node in IS is surrounded with a small circle radius $1/2$ and area $\pi/4$. These circles are disjoint and included inside a big circle with radius $5/2$ around v , that its area is $\pi \cdot (2\frac{1}{2})^2 = 6\frac{1}{4}\pi$. By considering the area differences of the small and the big circles we get an upper bound on the size of IS as follows;

$$M(2) \leq \frac{6.25 \cdot \pi}{0.25 \cdot \pi} = 25 < 8 \cdot R + 10$$

Now, we address the case of $R \geq 3$. We consider a graph H_v and lets IS be its maximum independent set of its power graph H_v^R . Thus, for every pair of nodes $u_1, u_2 \in IS$, their hop count distance in H_v is $d(u_1, u_2) > R$. Our calculation of the upper bound of $|IS|$ is composed of two steps. In the first step, we found an upper bound of the number of nodes in IS that their geometric distance from node v is at least $\lfloor \frac{R}{2} \rfloor + 1$. Let IS_{far} denote this set of nodes and let $M_{far}(R)$ denote the upper bound. Then we calculate an upper bound on the numbers on node in IS that their geometric distance from node v is at most $\lfloor \frac{R}{2} \rfloor + 1$. This set is denoted by IS_{near} and its upper bound is denoted by $M_{near}(R)$.

In our proof we use the following notations, described in Figure 17-(a). For every node $u \in IS$ let $P(u) = \{u = x_0, x_1, \dots, x_k = v\}$, $k \leq R$ be its shortest path to node v , and let $p(u) = \{u = x_0, x_1, \dots, x_J\}$, be the prefix of size $J = \lfloor \frac{R-1}{2} \rfloor$ of the path $P(u)$. Recall that the length of every path $P(u)$, $u \in IS$, except at the most one, is at least

$|P(u)| > R/2$, thus it has a prefix of size J . We say that two prefixes $p(u_1)$ and $p(u_2)$ are *remote* if the distance of every pair of nodes $x_1 \in p(u_1)$ and $x_2 \in p(u_2)$ is at least 1. Since the hop count distance of every pair $u_1, u_2 \in IS$ is $d(u_1, u_2) > R$, their prefixes must be remote. Thus, the maximum possible independent set is obtained by maximizing the number of remote prefixes.

Claim 1: For $R \geq 3$, $M_{far}(R) \leq 4 \cdot R$.

The distance of v from the J -th node, $x_J \in p(u)$, of every $u \in IS_{far}$ is at most $R - \lfloor \frac{R-1}{2} \rfloor = \lfloor \frac{R}{2} \rfloor + 1$. Thus, the size of IS_{far} is bounded by the number of suffixes that their last nodes are included inside the circle of radius $\lfloor \frac{R}{2} \rfloor + 1$ around node v . This number is maximized by arranging the prefixes as described in Figure 17-(c). Thus, for every node $u \in IS_{far}$ there is a node $x_k \in p(u)$ that is included in this circle but the node $x_{k-1} \in p(u)$ is not. Let $z(u)$ be the point where the line defined by (x_k, x_{k-1}) , intersects with the circle line. The minimal distance between two points $z(u_1), z(u_2)$ that represents two remote prefixes $p(u_1), p(u_2)$ is at least $\frac{\sqrt{3}}{2}$, as shown in Figure 17-(b). Thus, the maximal number of points, $z(u)$, that can be arranged around a circle of radius $\lfloor \frac{R-1}{2} \rfloor + 1$ is at most $\left\lfloor \frac{4}{\sqrt{3}} \cdot \pi \cdot (\lfloor \frac{R}{2} \rfloor + 1) \right\rfloor$. Thus for $R \geq 3$, $M_{far}(R) \leq 4 \cdot R$.

In Claim 1, we calculated the maximal number of prefixes that may cross the circle with radius $\lfloor \frac{R}{2} \rfloor + 1$ around v . This claim is based on the observation that one node of every prefix must be located inside the ring bounded by the two circles with radii $\lfloor \frac{R}{2} \rfloor + 1$ and $\lfloor \frac{R}{2} \rfloor$ around v . Moreover, under the assumption that IS_{far} is maximized and due to geometric considerations, it can be shown that the prefixes of the independent set, IS_{near} , must be concentrated inside a circle with radius $\lfloor \frac{R}{2} \rfloor$ around node v . With this assumption⁸, we calculate the maximal number of nodes in the set IS_{near} .

Claim 2: For $R \geq 3$, $M_{near}(R) \leq 4 \cdot R + 10$.

Suppose that every node $u \in N(v)$ is the center of a *small circle* with radius $\frac{1}{2}$ and area $\pi/4$. Two nodes $u_1, u_2 \in N(v)$ are *disjoint*⁹ in the graph H_v only if their small circles are not intersecting. Let the *area of a prefix*, $A(u)$, of a node $u \in IS_{near}$ be the total area covered by the small circle induced by the nodes of its prefix $p(u)$. Consider the prefix $p(u) = \{u = x_0, x_1, \dots, x_J\}$, $J = \lfloor \frac{R-1}{2} \rfloor$ of node $u \in IS_{near}$. Every pair of nodes with even index from $p(u)$, $\{x_0, x_2, x_4, \dots, x_J\} \subset p(u)$, are disjoint. Thus, $A(u)$ is at least the area of $\lfloor J/2 \rfloor + 1$ small circles. So,

$$A(u) \geq \frac{\pi}{4} \cdot (\lfloor \frac{J}{2} \rfloor + 1) \geq \frac{\pi}{4} \cdot (\lfloor \frac{R-1}{4} \rfloor + 1) \geq \frac{\pi \cdot R}{16}$$

Recall that the regions of the prefixes $p(u)$, $u \in IS_{near}$, do not overlap. Moreover, every prefix is included inside a circle of radius $R/2$ around v . Thus, the number of nodes is IS_{near} is bounded by

$$M_{near}(R) \leq \frac{\pi \cdot [(R+1)/2]^2}{\pi \cdot R/16} \leq \frac{4}{R} \cdot (R^2 + 2 \cdot R + 1)$$

Thus, for $R \geq 3$, $M_{near}(R) \leq 4 \cdot R + 10$, and from claims 1 and 2, we get that $M(R) \leq 8 \cdot R + 10$. \square

⁸Recall, that every prefix of a node in IS_{near} that is located in the considered ring implies reduction of the bound of IS_{far} at least by one.

⁹They are not connected by an edge in the graph H_v .

Lemma 5: Let IS be any independent set of the graph $G^R(V, E^R)$ and let DS^* be its minimum dominating set. Then, $|IS| \leq M(R) \cdot |DS^*|$.

Proof: Since DS^* is a dominating set of G^R each node $v \in V$ is included in the R -neighborhood of at least one node in DS^* . From Lemma 4, the set IS contains at most $M(R)$ nodes from the R -neighborhood of any node in DS^* . As a result, $|IS| \leq M(R) \cdot |DS^*|$. \square

Since greedy DIS algorithm selects as cluster heads a dominating independent set of the graph G^R , Corollary 1 is satisfied.

Corollary 1: The approximation factor of the greedy DIS coverage algorithm is $M(R)$.

We turn to calculate the complexity of the greedy DIS algorithm. Recall that the greedy algorithm may have at most $|V|$ iterations. At each iteration the value n_v of every node $v \in \bar{V}$ is calculated. Thus, the complexity of every iteration is $O(|V|^2)$ and the algorithm running time is $O(|V|^3)$.

C. The Cluster Refinement Algorithm

Consider the *cluster refinement algorithm* described in Section V-B. Let Y be the set of cluster-heads calculated by any coverage algorithm. The algorithm associates the node $v \in V$ to the cluster heads Y according to the following *assignment rule*: Every node v is associated with its closest cluster-head $y_i \in Y$, and ties are broken by selecting the cluster-head with the lowest ID.

Lemma 6: The cluster refinement algorithm defines $|Y|$ disjoint connected clusters that contain all the graph nodes and satisfy the depth requirement.

Proof: Recall that every node $v \in V$ is associated with a single cluster-head $y_i \in I_v$. Consequently, at most $|Y|$ disjoint clusters are created and they contain all the graph nodes. Now, we have to show that each cluster C with cluster-head y is connected and $d_C(y, v) \leq R$ for every $v \in C$. Suppose, that node v is assigned to the cluster C_i of cluster-head y_i and let $P(v, y_i)$ be any shortest path between them in the graph $G(V, E)$. We claim that all the nodes in $P(v, y_i)$ are also assigned to the same cluster. Suppose in contrast that there is a node $x \in P(v, y_i)$ that is assigned to the cluster of node y_j , $j \neq i$. Note that both $y_i, y_j \in N_R(x)$ since node x is included in the shortest path between nodes v and y_i . This assignment occurs only in one of the two conditions exists. (i) If $d(x, y_j) < d(x, y_i)$ or (ii) If $d(x, y_j) = d(x, y_i)$ and $y_j < y_i$, where y_i and y_j are the identification numbers of these nodes. However, these conditions cannot be satisfied without violating the assignment rule of node v . If condition (i) is satisfied then node v is included in $N_{Rmax}(y_j)$, since node x is included in the shortest path between nodes v and y_i . Moreover, $d(v, y_j) < d(v, y_i)$. Therefore according to the assignment rule node v should be assigned to node y_j . If condition (ii) is satisfied then also node $v \in N_{Rmax}(y_j)$ and $d(v, y_j) = d(v, y_i)$ which also contradict the assignment condition. So, every node in $P(v, y_i)$, is also assigned to cluster of node y_i . Therefore, $d_{C_i}(y_i, v) \leq R$ for each $v \in C_i$. \square

The cluster refinement algorithm calculates $|Y|$ Breadth-First-Search trees for identifying the closest cluster-head to each node. The complexity of a BFS tree is $O(|V|^2)$. Hence the algorithm complexity is $O(|V|^3)$.

D. The Weight Partition Algorithm

In this part we analyze the correctness and the approximation factor of the *weight partition algorithm* described in Sec-

tion V-C. Consider a connected cluster graph $G_C(V_C, E_C)$ that satisfy the depth requirement and let y be its cluster-head. We denote by T the SP-tree calculated by the algorithm, let $W_{tree}(v) = \sum_{x \in T_v} w_x$ denotes the total weight of the nodes in the sub-tree T_v and for every set of nodes S let $W_{tree}(S) = \sum_{v \in S} W_{tree}(v)$.

Lemma 7: The weight partition algorithm divides the cluster graph $G_C(V_C, E_C)$ to disjoint connected clusters that contain all the graph nodes and satisfy the depth, the relay-load and the weight constraints.

Proof: The partition algorithm defines a SP-tree T and divides it into smaller clusters. Since each node is included in exactly one cluster the clusters are disjoint and their union contains all the graph nodes. Each cluster defines a connected graph since it is composed of a single sub-tree of T or several sub-trees that their roots create a connected graph. Recall that every new tree rooted at any node u is created by merging several sub-trees T_v , $v \in N(u)$. Since the depth of every sub-tree T_v , $v \in C - \{y\}$ is at most $R - 1$, The depth of the resulting tree is at most R and the depth requirement is satisfied. From the description of the weight partition algorithm it is clear that it also satisfies the weight and the relay-load constraints, since every sub-tree that violates these constraints is removed from the original SP-tree and divided into smaller cluster that meet these requirements. \square We turn to calculate the approximation factor of the weight partition algorithm as stated by Lemma 8. For that propose we first prove the following auxiliary claims.

Claim 3: The maximum independent set of the graph induced by the set of children K_u , $u \in V_C - \{y\}$, has at most 4 nodes.

Proof: This is a direct result from Property 5. Suppose in contrast that there is an independent set $S \subseteq K_u$ of size 5 and let v be the parent of u in T . Since T is a SP-tree the geometric distance from v to any nodes in S is at least 1. Thus the set $S \cup \{v\}$ is an independent set of size 6 of the graph defined by $N(u)$. This contradicts Property 5. \square

Claim 4: Let \mathcal{S} be a collection of sets calculated by the local pruning routine. Then \mathcal{S} contains at most 4 sets $S \in \mathcal{S}$ with $W_{tree}(S) \leq W/2$.

Proof: Consider, in contrast, that there is an execution of the local pruning routine for a given sub-tree T_u where 5 sets S_1, S_2, \dots, S_5 with $W_{tree}(S_i) \leq W/2$ where created. Let v_1, v_2, \dots, v_5 be the seed nodes of these sets. Assume an iteration in which a set S with seed node v is created. According to the local pruning routine, the iteration terminates when there is no node $x \in N(v)$ that can be added to S without violating the radius and the weight constraints. Since $W_{tree}(S_i) \leq W/2$, for every set S_i , $1 \leq i \leq 5$, their seed nodes must define an independent set of the graph defined by K_u (otherwise at least two seed nodes should be included in the same set). This contradicts Claim 3. \square

Claim 5: Consider a partition of a given sub-tree T_u , $u \neq y$, to k clusters by the local pruning routine. Then $k < \frac{3 \cdot W_{tree}(u)}{W}$.

Proof: Let C_1, \dots, C_k be the clustered created by the local pruning algorithm labeled according to their creation order. Each cluster C_j , $1 \leq j < k$ is associated with a single set $S_j \subseteq K_u$, hence $W_{sum}(C_j) = W_{tree}(S_j)$. Regard that these sets are sorted in increasing order according to their $W_{tree}(S_j)$ values. The last cluster C_k contains the node u and may include several sets calculated by the local pruning routine. According to Claim 4 the routine may define at most 4 sets S such that

$W_{tree}(S) \leq W/3$. In such case the two smallest sets remain connected to node u , (assuming $w_u \leq W/3$). From this we conclude that for every $1 \leq j \leq k - 3$, $W_{sum}(C_j) > W/3$ and $W_{sum}(C_{k-2}) + W_{sum}(C_{k-1}) + W_{sum}(C_k) > W$. Since, $W_{tree}(u) = \sum_{j=1}^k W_{sum}(C_j)$, the average $\overline{W}_{sum}(C_j) > W/3$ and $k < (3 \cdot W_{tree}(u))/W$. \square

Corollary 2: Let y be the root of the original SP-tree T and consider a partition of the tree T to k clusters by the local pruning routine. Then $k < \frac{3 \cdot W_{tree}(y)}{W} + 1$.

Lemma 8: Let assume that for every $v \in V$ $w_v \leq W/3$. Then the approximation factor of the weight partition algorithm is 3.

Proof: Let OPT_W be the number of clusters in the optimal partition of $G_C(V_C, E_C)$ to connected clusters that satisfy the weight requirement, and let RES_W be the number of clusters created by the weight partition algorithm. Recall that $RES_W - 1$ clusters are created by the local pruning routine and an additional cluster contains the cluster-head y . According to Claim 5 and Corollary 2, $RES_W < \frac{3 \cdot W_{sum}(T)}{W} + 1$. Moreover, $OPT_W \geq \lceil W_{sum}(T)/W \rceil$. Since, RES_W and OPT_W are integers $RES_W \leq 3 \cdot OPT_W$. \square

We show now that the algorithm running time is $O(|V|^3)$. The weight partition algorithm is called at most $|V|$ times. It contains two steps, in the first step it constructs a SP-tree and then a post-order-tour is done for discovering over-weight components. The complexity of these steps is $O(|V|^2)$. Therefore, the complexity of the third stage is $O(|V|^3)$.