



**HAL**  
open science

# Load-Balancing Adaptive Clustering Refinement Algorithm for Wireless Sensor Network Clusters

Gal Oren, Leonid Barenboim, Harel Levin

► **To cite this version:**

Gal Oren, Leonid Barenboim, Harel Levin. Load-Balancing Adaptive Clustering Refinement Algorithm for Wireless Sensor Network Clusters. 15th International Conference on Wired/Wireless Internet Communication (WWIC), Jun 2017, St. Petersburg, Russia. pp.157-173, 10.1007/978-3-319-61382-6\_13 . hal-01675417

**HAL Id: hal-01675417**

**<https://inria.hal.science/hal-01675417>**

Submitted on 4 Jan 2018

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

# Load-Balancing Adaptive Clustering Refinement Algorithm for Wireless Sensor Network Clusters

Gal Oren<sup>1,2</sup>, Leonid Barenboim<sup>3</sup>, Harel Levin<sup>2,3</sup>

<sup>1</sup> Department of Computer Science, Ben-Gurion University of the Negev, POB 653, Be'er Sheva, Israel

<sup>2</sup> Department of Physics, Nuclear Research Center-Negev, P.O.B. 9001, Be'er-Sheva, Israel

<sup>3</sup> Department of Mathematics and Computer Science, The Open University of Israel, P.O.B. 808, Ra'anana, Israel

orenw@post.bgu.ac.il, leonidb@openu.ac.il, harellevin@gmail.com

**Abstract.** Energy efficiency is a crucial performance metric in sensor networks, imminently determining the network lifetime. Consequently, a key objective in WSN is to improve overall energy efficiency to extend the network lifetime. Its conservation influences the topology design of many WSN-based systems, especially the clustering of the network. Unlike other WSN clustering algorithms, that do not re-cluster the network after deployment, our hypothesis is that it is advisable, in terms of prolonging the network lifetime, to adaptively re-cluster specific regions that are triggered significantly more than other regions in the network. By doing so, it is possible to minimize or even prevent the premature death of CHs, which are heavily burdened with sensing and transmitting actions – much more than other parts of the WSN. In order to do so we introduce the Adaptive Clustering Refinement (ACR) algorithm, which is based on the Adaptive Mesh Refinement algorithm by Berger and Olinger [14] and the Hierarchical Control Clustering algorithm by Banerjee and Khuller [13]. We prove that the ACR algorithm complexity is linear in the total size of the graph, and that we manage to optimize the WSN cluster connectivity and prolong its lifetime. We also devise a local version of the algorithm with improved complexity.

**Keywords:** Wireless Sensor Networks, Adaptive Clustering Refinement, Energy Optimization, Networks Connectivity.

## 1 Introduction

### Background

In the past few years, rapid advances in the area of micro and nano technology have taken place with implication to all of the scientific research fields. As a result, micro-sensors have been developed for various needs. Subsequently, this has led to the development of Wireless Sensor Networks (WSNs). WSNs are composed of a variety of nodes, and they include abilities of data sensing and data processing, as well as wireless transmission such as Bluetooth or radio technology. The invention of WSNs

has led to the development of various serviceable applications, including control, tracking and monitoring of large areas [1]. The introduction of WSN presents much superiority over orthodox sensing doctrines. A large-scale, dense spreading improves the spatial coverage and obtains much better resolutions; moreover, it also extends the fault tolerance and sturdiness of such a system. The deployment of sensor nodes is performed in an ad-hoc fashion and occasionally does not include sufficient planning and engineering in many WSN applications. Once the nodes are deployed in their positions, it is essential that the sensors will independently organize themselves into a unified wireless network. The nodes are powered by battery and are designed to operate without supervision for a relatively long duration of time. However, it is often problematic to replace or recharge the sensor node batteries, due to the fact that most of the deployments are in large fields (e.g. animal control) or inaccessible places (e.g. war zones) [2].

The fundamental consideration in some other networks (such as mobile networks) not always take into account the energy consumption, while it is still a significant design factor that directly influences the network lifetime; this is because those energy resources can be easily replaced or rechargeable by the users or the operators. Hence, a higher concern is given in those networks to quality of service such as higher performances. However, energy efficiency is a crucial performance metric in sensor networks, imminently determining the network lifetime. In order to tackle this array of considerations, some protocols may be used to handle the trade-off of performance metrics such as network overall postponement against energy efficiency [3].

Energy may be obtained by utilizing the external environment (for example, by using photovoltaic cells as an energy source). Nevertheless, the behavior of external energy source is usually non-persistent, which makes it unreliable and requires the use of batteries [4]. However, batteries also create a problem because of their finite amount of stored energy and the frequent need to replace or recharge them. Consequently, a main goal in WSN is to improve overall energy efficiency to extend the network lifetime. Energy conservation influences the design of many WSN-based systems. Comprehensive studies have examined this issue [5] and suggested energy optimization techniques in order to manage the WSN topology accordingly.

The most important observation from those studies [6] to our study is that the energy consumption of the transmission unit is significantly higher than the energy needed to make any kind of computation in a node, and the current estimations are that there is an order of magnitude difference in energy consumption between the two. This shows that transmission should always be traded for computation when possible, in order to preserve power. Following this observation, these factors should be considered and are vital for understanding the energy consumption problem. The most basic and common way to implement this knowledge in WSNs is by using clustering, as we explain below.

### **WSN Clustering**

Nodes in multi-hop ad-hoc sensor networks play a dual role as a data originator and data router at the same time. Some of the nodes may not operate properly, which may lead to major topological changes and require the rerouting of some packets and network reconstruction. This further emphasizes the importance of energy efficiency and energy control. Because of this, the focus of many researchers of WSN is to develop

protocols and algorithms that consider scalability and energy efficiency by grouping the network nodes into clusters in order to form a hierarchical topology and eliminating any redundant data. This process consists of the following steps: Nodes transmit their sensed data to a master node in a distributed fashion; the master node aggregates the data; after computation the master node forwards the new calculated data to its master or to the sink node after discarding of the superfluous data.

A formalized approach [7] to demonstrate this concept assumes that a WSN cluster includes two main components: A base station (BS) and a number of sub-clusters (which also can have sub-clusters). Each of the sub-clusters have a leader (usually referred to as the cluster head, or CH), as well as other nodes (non-cluster-head nodes, or NCHs) that are all within a same transmission range around the CH. The transmission range is defined as the maximal distance between a receiver and a sender – in this case, CH and NCH respectively – and there is a correlation between the length of the transmission range and the energy consumed in this topology.

A CH has various responsibilities: It gathers data from NCHs, processes it in order to minimize its volume as much as possible and transmit it to the master node or the base station. A CH forwards the information in two possible ways: Directly or by using numerous relay nodes. The relay nodes focus on carry forward the data transmitted from other nodes, rather than local sensing. The CH in each sub-cluster may be chosen in distributed fashion by the sensors themselves, or pre-determined by the network configurator. These approaches can be classified into two groups: static clustering and dynamic clustering. The difference between these two clustering algorithms is that static clustering-based algorithms do not modify clusters after formation, while dynamic algorithms may choose a new CH after a given period of time.

Another vital factor in determining the performance and network lifetime is based on the location that each CH is set in. As previously mentioned, the energy consumption is highly derived by the transmission span, and if a proper CH position is chosen, the CH node may not be forced to communicate with the master node / BS directly over a distant distance; instead, it will be able to preserve its stored energy for a long period of time. Other studies and developments in the field show heterogeneous network topologies, in which the CHs have additional energy capacity compared to NCHs, which obviously increases the network lifetime. These fields also introduce network topologies with multiple layers ( $L > 2$ ), such as adding a meta-CH layer, which may help reducing the load of the other CHs in the layer above and so forth [8]. In this paper we adopt the idea of multi-layers CHs with a uniformed energy capacity, and devise an algorithm that turns NCHs to CHs and vice versa, in order to preserve energy.

The lifetime maximization problem has been addressed in many algorithms, and the current outcomes can be classified into two main groups: centralized doctrines vs. distributed doctrines. For centralized doctrines, the sensors' position must be available and known in order to obtain comprehensive optimizations according to some performance metrics. In contrast, distributed doctrines make resolutions according to the local data that approximate nodes manage to know about each other. In recent years, various WSN algorithms have been introduced, in attempt to minimize energy consumption using central clustering doctrine in order to extend the network lifetime [9]. These studies, however, mainly focused on reducing the number of CHs or their energy consumption, rather than focusing on NCHs. In contrast, in this paper we adopt the distributed fashion from the same reasons mentioned before: our suggested

algorithm has to consider the two entities equally because of the fluidity of rules of the nodes in the WSN, from NCHs to CHs and vice versa.

## 2 The Adaptive Clustering Refinement (ACR) Algorithm

### Energy-Efficient Schemes and Factors

Energy-efficient schemes can be mainly divided into two classes according to their purposes: reducing the energy consumption of all nodes in a WSN, and increasing the WSN lifespan and connectivity. While these two objectives are highly interdependent, they are not the same. Indeed, decreasing overall energy consumption is solely a minimization problem, while increasing network lifespan is a min-max problem, as network lifespans may often be fixed or influenced by the network nodes that have the shortest lifespan [3]. The nodes with the shortest lifespan are referred to as *bottleneck nodes* and in the case of clustered WSNs they are often the CHs. These two objectives of energy-efficient schemes derive different optimizations. The term *network lifespan* has been given various definitions that mostly fit into two categories [2]: (1) The amount of time until some  $X$  percentage of nodes consume most or all of their energy supply, and (2) The amount of time until a specific coverage or connectivity setting conditions in a specific region cannot be realized.

In this paper, we adopt the idea of minimizing the overall energy consumption in conjunction with the idea of maximizing the network lifetime. We investigate the amount of time until some  $X$  percentage of nodes consume most or all of their energy supply in a **specific region**, and how is that affects the specific coverage or connectivity of the region nodes - which in this case are probably the most triggered ones. By adopting such a definition, maximizing network lifetime leads to a **localized** max-min problem with the objective of making those sensors with shortest lifespan survive as long as possible, while trying to minimize the overall energy consumption of the specific triggered area.

### Research Goals

Unlike other WSN clustering algorithms, which cannot or prefer not to re-cluster the network after deployment except the necessary case of death of nodes, our hypothesis is that it is advisable, in terms of prolonging the network lifetime, to adaptively re-cluster specific regions that are **triggered significantly** more than other regions in the network in a distributed fashion. By doing so, it is possible to minimize or even prevent the premature death of CHs (in comparison to the other nodes in the network), which are heavily burdened with sensing and transmitting actions – much more than other parts of the WSN.

For example [10], imagine that the South-African government decided to hermetically map the movement of the Blue Wildebeest in Kruger national park, one of the largest game reserves in Africa, which covers an area of 19,485 square kilometers in the provinces of Limpopo and Mpumalanga in northeastern South Africa. It is known that the Blue Wildebeest take part in a long-distance migration, synchronize to overlap

with the yearly pattern of rainfall and herbage sprouting on some several specific plains where they can trace the nutrient fodder [11]. Because of the huge masses of the Blue Wildebeest herds, it is impracticable to collar their members with wireless sensors. Consequently, in order to do trace those herds, at the center of each 100 square meters a wireless sensor with animal sound recognition [12] has been placed, and initially all of those nodes created a uniform hierarchical WSN cluster. Now, because the Blue Wildebeest tend to move in large-scale herds and not in individual fashion, it means that soon there is going to be a massive load on parts of the network, while other parts of the network will not be triggered at all; a state which shortly will cause, as previously explained, the premature death of CHs in this region and the reduction of the connectivity in the network. Therefore, in order to ease the load on the burdened CHs, a re-clusterization of those specific triggered sub-clusters of the whole WSN can prolong the total lifespan of the network. This way it is possible to achieve the goal of minimizing the overall energy consumption in conjunction with the goal of maximizing the network lifetime. We refer to our re-clustering algorithm as the *Adaptive Clustering Refinement* (henceforth, ACR), which is based on the Hierarchical Control Clustering (HCC) Algorithm.

Through this paper  $n$  will represent the number of CHs and  $m$  will represent the number of sub-WSN grids to be refined.

### **The Hierarchical Control Clustering (HCC) Algorithm**

In order to devise the ACR algorithm, we need to determine which clustering scheme will take place along with the refinement method. Unlike most of the published schemes, the goal of Banerjee and Khuller scheme is to form a multi-tier hierarchical clustering using proximity-traversing-based algorithm named Hierarchical Control Clustering [13] (henceforth, HCC). HCC is a distributed multi-hop hierarchical clustering algorithm which also effectively manages to create a multi-level cluster hierarchy. The algorithm works in a distributed fashion, meaning that each node in the WSN can initiate the cluster formation process. The HCC progress in two main sub-processes, when the first is the Tree Discovery process and the second is the Cluster Formation process.

The first process is essentially a distributed formation of a BFS tree, which is rooted is the initiator node. In this process, every single node broadcasts a message signal (which includes the parent identification, the BFS tree root identification, and the sub-tree size) at every predetermined unit of time, transporting the data regarding its shortest hop distance to the BFS tree root. This is done by the following routine: A node  $n_i$  which is adjacent to node  $n_j$  will select  $n_j$  to be its parent, and also will bring up-to-date its hop distance to the root of the BFS tree, if the route via  $n_j$  is shorter. Obviously, each node brings up-to-date its sub-tree size when its children sub-tree size modifies. The second process is initiated when a sub-tree on a node rise above the size parameter  $k$ . Then the node starts cluster formation on its sub-tree. If the sub-tree size is less than  $2k$  it will form a single cluster for the whole sub-tree. Otherwise, it will form multiple clusters.

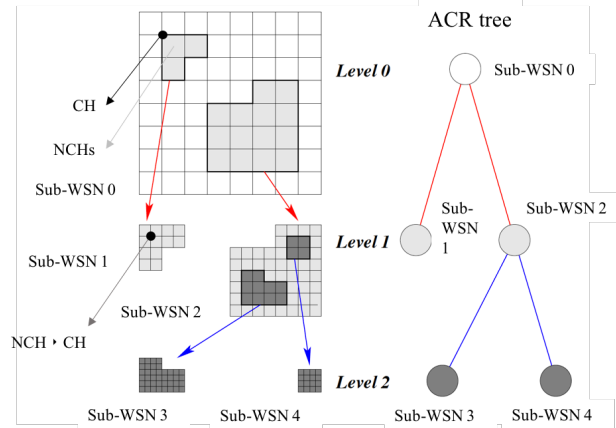
This two step process has a time complexity of  $O(n)$ . Nevertheless, it has managed to obtain balanced clustering, and additionally to deal with non-stable environments

quite effectively. This time complexity is calculated as follows: The BFS tree formation of the first process of GRAPH CLUSTER ( $T \leftarrow \text{BFS tree of } G$ ), takes  $O(|E|)$ . The computation time at every vertex  $n_i$ , in post-order traversal, is  $O(\text{deg}_T(n_i))$ , meaning the degree of  $n_i$  in the tree. Therefore, the total cost for the whole post-order traversal is  $\sum_n \text{deg}_T(n_i) = O(|V|)$ . Thus, the complexity of the algorithm is  $O(|E| + |V|)$ .

### The Adaptive Mesh Refinement (AMR) and Adaptive Clustering Refinement (ACR) Algorithms

The Adaptive Mesh Refinement (AMR) is a type of multi-scale algorithm that achieves high resolution in localized regions of dynamic, multi-dimensional numerical simulations [14][15]. The AMR algorithm has been implemented with large success to model large-scale scientific simulations in a variety of disciplines, mainly astrophysics. In principle, the AMR algorithm manages to place large high-resolution grids exactly where they are needed, meaning where the high computational cost and overheads requires. The AMR algorithm adaptability achieve a state in which it is possible to simulate multi-scale resolutions that are impossible otherwise because of computational power limits with the traditional techniques which use a global uniform fine grid.

The motivation of combining the AMR concept into clustering comes from the observation that a very fine mesh can be required for clustering on a highly irregular or concentrated data distribution, if a grid-based clustering algorithm that employs a single uniform mesh is used. This motivation also exists in the case of clustering WSNs, which are under fluid amount of load and need to re-cluster themselves in order to create a more uniformed load-balancing among the CHs nodes. In this paper, we demonstrate this re-clustering by using the HCC algorithm [13] as a building block, especially because of its hierarchical clustering fashion and the dynamic and distributed abilities to ‘refine’ areas when reaching some parameter ( $k$ ), which for simplicity of explanation will be set to 2 through this paper.



**Figure 1.** An ACR WSN-Graph Tree formation example with 2 levels of refinement. A finer resolution WSN-Graph is applied each time a sub-WSN is created.

Figure 1 shows an example of an ACR WSN-Graph tree formation in which every intersection represents a CH and every square represent the field in which the NCHs nodes are located. Each refined CH creates  $k^2$  child-CHs. It is possible to see that each tree node uses a different resolution WSN-Graph. The root WSN-Graph with the coarsest granularity (i.e. the WSN-Graph cluster at the beginning) covers the entire domain, which contains two sub-WSN-Graphs, sub-WSN 1 and sub-WSN 2, which were refined – meaning that from each square at the previous step, an NCH turns into a CH in order to ease the load pressure of the other CHs around it (i.e. the corners of the previous square). Sub-WSN 2 at level 1 also contains two sub-WSN-Graphs that are discovered using a finer graph. The deeper the node is located in the tree, the finer the WSN-Graph is used.

In general, the ACR load-balancing-based algorithm tries to find the load burdened regions, and the number of the discovered load burdened regions determines the number of regions that will need to re-cluster themselves. Because the refinement is based on the load factor, the ACR method can recursively identify the load burdened regions and represent them in a hierarchical tree structure in which the tree nodes near the leaves indicate the more load burdened regions and the nodes close to the root have lower rates of load burden. The ACR tree construction is a top-down process starting from the root node that covers the entire problem volume

#### The Adaptive Clustering Refinement Algorithm

**ACR**(WSN-Graph, level)

1. Traverse the WSN-Graph using BFS: Calculate the top  $k^2$ -quantile of the load on the entire WSN-Graph and set it to be the threshold.
2. **For** each non-marked CH:
  - 2.1. **If** the CH-load is greater than the threshold:
    - 2.1.1. Mark this CH to be refined
    - 2.1.2. Connect this CH to all of its neighbor CHs in the WSN-Graph that are also greater than the threshold and mark them too
    - 2.1.3. Create a new sub-WSN-Graph containing all of these nodes
  - 2.2. **Else if** the CH-load is smaller than half of the threshold:
    - 2.2.1. Mark this CH to be coarsen
    - 2.2.2. Connect this CH to all of its neighbor CHs in the WSN-Graph that are also smaller than half of the threshold and mark them too
    - 2.2.3. Create a new sub-WSN-Graph containing all of these nodes
3. **For** each sub-WSN-Graph:
  - 3.1. **If** sub-WSN-Graph set to be refined:
    - 3.1.1. **Refine** sub-WSN-Graph
    - 3.1.2. **Call** **ACR**(sub-WSN-Graph, level+1)
  - 3.2. **If** sub-WSN-Graph set to be coarsen:
    - 3.2.1. **Coarse** sub-WSN-Graph



Given a WSN-Graph (initially it assumed to be the HCC output tree), the ACR tree construction starts at the BS or the main CH node that uses the WSN-Graph with an initial granularity to cover the entire problem domain as given from the hierarchical WSN clustering algorithm [13]. While traversing the WSN-Graph in a BFS fashion, a calculation is being made in order to determine the average load on the WSN-Graph. Afterwards, each node is examined to check if the load exceeds the given threshold (we chose a  $k^2$ -quantile of the nodes because this ensures that the number of CH nodes produced at each round is less than the nodes that originally exceed the threshold. This fact helps us to bound the size of the produced tree). The nodes whose load is larger than the threshold are marked to be refined, and a new sub-WSN-Graph is created from all marked nodes that are connected (adjacent) with each other. The algorithm recursively refines the sub-WSN that has been found, and goes to the child nodes while a hierarchical tree is built. Whenever the load is lower than some pre-defined lower bound, a coarsening (un-refinement) should be performed. The algorithm stops when the maximum level of tree depth is reached or there are no nodes with load that are larger than the threshold. The process of constructing the ACR tree is a top-down operation. This is also the main difference of our ACR approach from the other grid-based algorithms whose hierarchical trees are built in a bottom-up fashion, especially because of the need to start at the BS or the main CH, which receives most of the information regarding the load. Next, we analyze the running time of the algorithm in Theorem 1, and the construction complexity of the algorithm in Theorem 2. An upper bound on the total number of CHs in the ACR tree is provided by Theorem 3.

**Theorem 1:** *The running time of the algorithm is  $O(dm \frac{1-p^h}{1-p})$ :*

*Proof.* Assuming  $m$  is the number of the sub-WSNs grids which need to be refined, the dimensionality of those grids is fixed to 2 dimensions of an Euclidean space ( $d=2$ ),  $h$  is the ACR tree height, and  $p$  represents the average percentage of data objects to be refined at each level, the complexity for scanning the database is almost the same as in the AMR algorithm, and it is bounded by  $O(dm + dmp + dmp^2 + \dots + dmp^{h-1}) = O(2m \frac{1-p^h}{1-p}) \leq O(\frac{2m}{1-p})$ . ■

**Theorem 2:** *The construction complexity of the algorithm is  $O(dm \frac{1-p^h}{1-p} + 6^d n \frac{1-q^h}{1-q})$ .*

*Proof.* The complexity of finding the sub-WSN-Graphs highly depends on the size of the graph in each sub-WSN-Graph. We assume the graph size at the root is  $n$  and  $q$  is the average ratio of graph sizes between two levels of graphs. Assuming  $m$  is the number of the sub-WSNs grids which need to be refined. The complexity for marking the CHs nodes that exceed the threshold and connecting the other marked CHs to form the sub-WSN-Graph is  $O(2^d 3^d n q + 2^d 3^d n q^2 + \dots + 2^d 3^d n q^{h-1}) = O(6^d n \frac{1-q^h}{1-q})$  assuming the refinement factor is 2, meaning that we split any cluster to  $2^d$  sub-clusters, and each node must check at most its  $(3^2 - 1)$  neighbors for connected sub-WSN-

Graph. Therefore the complexity for constructing the AMR tree is  $O(2m \frac{1-p^h}{1-p} + 6^2 n \frac{1-q^h}{1-q})$ . ■

**Theorem 3:** *The total number of CH nodes in the ACR is bounded by  $O(n^2)$ , where  $n$  is the number of CHs in the input ACR tree.*

*Proof.* At every step, there are less than  $\frac{n'}{k^2}$  CHs chosen to be refined (by the threshold quantile definition), where  $n'$  is the number of CHs received as an argument and  $k$  is the refinement factor. Each refined CH creates  $k^2$  child-CHs. It means that at every iteration, the algorithm will create  $(\frac{n'}{k^2} - 1)k^2 = n' - k^2$  new CHs. It is clear that after  $i$  iterations, the algorithm will produce  $n' - ik^2$  new CHs, which means that after  $\frac{n'}{k^2}$  iterations it will stop producing new CHs. Hence, the total number of CHs after the algorithm run is at most  $n' + \sum_{i=1}^{\frac{n'}{k^2}} (n' - ik^2) = n' + \left( \frac{n'^2}{k^2} - \frac{n'}{2k^2} (k^2 + n') \right) = n' + \left( \frac{n'^2}{2k^2} - \frac{n'}{2} \right) = \frac{n'}{2} + \frac{n'^2}{2k^2}$ , and therefore the total complexity of the ACR tree is  $O(n^2)$  while  $k$  is a constant. ■

### 3 The ACR Algorithm Energy Consumption Model

#### WSN Energy Consumption Model

We consider two different types of energy consumption for data transmission and receiving, respectively: a transmitter consumes energy to run both the radio electronics and the power amplifier, while a receiver only consumes energy to drive the radio electronics. The mobile radio channels on typical sensor nodes are predominantly in the VHF (frequency from 30MHz to 300MHz, wavelength from 1m to 10 m) and UHF (frequency from 300MHz to 3GHz, wavelength from 10cm to 1m), respectively [1][2]. We employ the free space ( $fs$ ) fading channel model for wireless communication that incurs a  $d^2$  power loss, some outdoor deployments [2]. In a real communication system, the transmission power could be adjusted by suitably configuring the power amplifier. Therefore, the energy dissipation in transmitting one unit of data message over a directed wireless communication link can be modeled as  $E_t(i)$ , when  $E_t(i) = E_{elec} + E_{amp}(d_{i,j}) = E_{elec} + \epsilon_{fs} \cdot d_{i,j}^2$ , where  $E_{elec}$  denotes the energy for driving the electronics, which depends on various factors including digital coding, modulation, filtering, and spreading of the signals, for both transmitter electronics and receiver electronics; and  $\epsilon_{fs}$  is the coefficient for calculating the amplifier energy  $E_{amp}$ , which depends on the Euclidean distance  $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$  between transmitter  $v_i$  located at  $(x_i, y_i)$  and receiver  $v_j$  located at  $(x_j, y_j)$  as well as the acceptance bit-error rate. The energy consumed by a sensor  $v_i$  in receiving one unit of

data packet is denoted as  $E_r(i) = E_{elec}$ . Note that the above transmission and receiving energy models assume a contention free MAC protocol, where interferences from simultaneous transmission can be avoided.

A CH, which also collects environment sensing data, receives data messages from NCHs within the cluster and sends all the data to a main CH or BS after performing a certain type of data processing (such as data aggregation and data compression). We use a constant  $E_p$  to represent the energy spent in processing each unit of received or sensed data. We assume that the CH performs complete data aggregation, that is, an input of two k-bit messages produces an output of one k-bit message after aggregation. Furthermore, we use a parameter  $\alpha$ ,  $0 < \alpha \leq 1$ , to denote the data compression ratio: an input of k bits results in an output of  $\alpha \cdot k$  bits after compression.

### **Problem Formulation**

The problem of determining the optimal number and location of CHs for minimum TEC in sensor networks is formulated as follows. We consider a WSN where n sensor nodes have been deployed in a bounded  $L \times L$  ( $m^2$ ) square. The location of each sensor  $v_i, i \in 0, 1, \dots, n - 1$ , is denoted as  $(x_i, y_i)$ . We assume a one-hop communication model, and the transmission energy is calculated using the free space (fs) model mentioned above. We consider a sensor deployment scenario in a uniform node distribution. The optimization problem is to strategically refine parts of the entire WSN by designating an appropriate subset of sensor nodes in the network as CHs base on the sensing load, each of which forms a cluster with its neighbor nodes, such that there will be a reduce of pressure on the entire CHs in the WSN. Thus, (1) the total energy consumption for the transmission of each unit of data message from all NCHs to CHs and so forth is minimized, and (2) the total energy consumption of the entire WSN will reduce, and it will be achievable to maximize the WSN lifetime and connectivity at once.

We consider the following general conditions or assumptions in our problem formulation: All sensors are pre-deployed and have constrained energy supply; The network is static, that is, neither the sensors nor the CHs has mobility once deployed; The total number of sensors is known; Each CH forms exactly one cluster, and besides data processing, also performs the same task of environmental sensing and data collection as a regular sensor node; There exists a contention free MAC protocol for wireless communication. We consider the energy consumption for data transmission of each NCH, and for data receiving, processing, and transmission of each CH. Since the energy cost for environment sensing is generally much less than communication and processing tasks, we do not consider sensing energy cost here. Obviously, the total energy consumption depends on the network distribution, the number and location of CHs, and the compression ratio  $\alpha$  at CHs.

### **ACR Algorithm Energy-Efficiency Proof**

In this paper, we use an analytical formula for calculating the optimal value of refinement of loaded parts of the WSN in order to achieve the minimum total energy consumption of data transfer from NCHs to the BS or the main CH through their

corresponding CHs. The optimal number of refinement determined by our approach can be used to guide the execution of the HCC clustering algorithm that requires such information.

The total energy consumption per round, denoted by  $E_{Tot}$ , is the sum of the energy consumption  $E_{NCH}$  of all NCHs for data transmission and the energy consumption  $E_{CH}$  of all CHs for data receiving, processing, and transmission in one round, which can be defined as  $E_{Tot} = E_{NCH} + E_{CH}$ . The  $E_{NCH}$  only includes transmission energy cost  $E_t$ , when  $E_{CH}$  includes the energy cost  $E_r$  for receiving,  $E_p$  for processing, and  $E_t$  for transmission. Each of NCHs transfers one unit of data to its corresponding CH, which performs processing (aggregation and compression) on the received data and its own sensing data, and sends the compressed aggregated result to other CH or BS.

In order to prove that our algorithm reduces the total energy consumption of the whole WSN and also increases the connectivity of the network we need to show that (I) the total energy consumption of the refined zone is actually lower than the previous state, and (II) that the distances of the NCHs to their new CHs in the refined zones were reduced and became more uniform than the previous state. Therefore, we need first to formulate the energy consumption of the NCHs and the CHs in our model. Based on the previous knowledge of  $E_{Tot}$ , for each NCH in our model the energy consumption per bit will be:

$$E_{NCH} = E_t = E_{elec} + \epsilon_{fs} \cdot d_{NCH \rightarrow CH}^2 \quad (1)$$

When  $d_{NCH \rightarrow CH}$  is the distance between the NCH to its CH, and for each CH in our model the energy consumption will be:

$$\begin{aligned} E_{CH} &= n_{NCH \rightarrow CH} E_r + (n_{NCH \rightarrow CH} + 1) E_p + \alpha E_t \\ &= n_{NCH \rightarrow CH} E_{elec} + (n_{NCH \rightarrow CH} + 1) E_p + \alpha (E_{elec} + \epsilon_{fs} \cdot d_{CH \rightarrow InitCH}^2) \end{aligned} \quad (2)$$

When  $n_{NCH \rightarrow CH}$  is the number of NCH that communicate with the CH, and  $d_{CH \rightarrow HigherCH}$  is the distance between the CH to its higher CH in the hierarchy (the BS is the top node in the hierarchy). Hence, a refinement of the WSN is always worthwhile if the current energy consumption in the intended to refinement zone is higher than the energy consumption of the same zone after the refinement. A formulation of this condition, based on the previous formulas to  $E_{NCH}$  and  $E_{CH}$  will be:

$$\sum_{i=1}^{n_{NCH \rightarrow InitCH}} E_{NCH_i} + E_{InitCH} > \sum_{i=1}^{n_{CHs}} (E_{CH_i} + \sum_{j=1}^{n_{NCH \rightarrow CH_i}} E_{NCH_j}) + E_{InitCH} \quad (3)$$

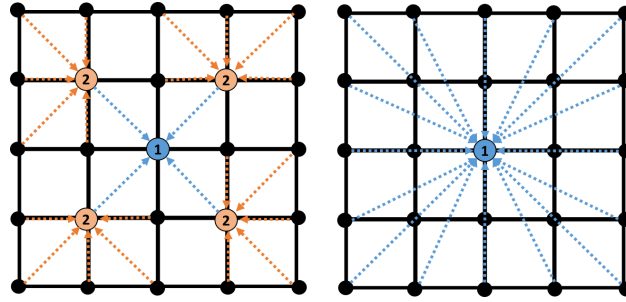
When the  $InitCH$  is the CH in the hierarchy from which a refinement should start, and  $\{CH_i \mid i \in \{1, \dots, n_{CH \rightarrow InitCH}\}\}$  is the group of NCHs which turns into CHs in the refinement process.

It is possible to see that although each refinement shortens the distance between the NCHs and the CHs, an  $E_p$  overhead accumulates due to the additional CHs in the WSN. Base on the known energy consumption parameters [7],  $E_{elec} = 5 \cdot 10^{-8} J/bit$ ,  $E_p =$

$5 \cdot 10^{-9} J/bit/signal$ , and  $\epsilon_{fs} = 10^{-10} J/bit/m^2$ , although there is an order of magnitude difference between  $E_p$  and  $\epsilon_{fs}$ , which makes it look like it is not energetically efficient to add CHs to the WSN,  $E_p$  is multiplied by  $(n_{NCH \rightarrow CH} + 1)$ , while  $\epsilon_{fs}$  is multiplied by  $d_{CH \rightarrow HigherCH}^2$ , hence even a relatively small distance between the NCHs to their CH can overcome the data aggregation overhead. This means that a good refinement using the ACR algorithm, which will take into consideration this balancing, will achieve a reduction in the total energy consumption and an increase in the connectivity of the WSN simultaneously. We prove this in Theorem 4.

**Theorem 4:** *The total squared distances in the entire WSN constantly reduced by 70% using the refinement algorithm in WSN grid.*

*Proof.* An exemplification of the balancing that the ACR algorithm performs, and the energy factors outperformances, are demonstrated in WSN in figure 2. The figure presents a 5X5 WSN grid before and after an ACR refinement, with one CH to 24 NCHs before refinement with 2 hierarchy level (right), and the same grid after refinement with 4 NCHs turned into CHs with 3 hierarchy levels.



**Figure 2.** A 5X5 WSN grid with one CH to 24 NCHs before refinement (right), and the same grid after refinement with 4 NCHs turned into CHs.

We reached those results using the following formulas (I-IV), which present the calculation for the pre-refine squared distances (I), the post-refine squared distances (II), the value of the distances saving (III) and the percentage of the distance saving (IV).

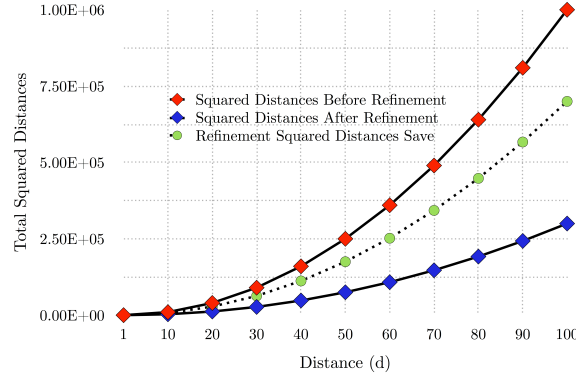
$$\text{PreRefineSqDist}[d, n] = 4 \cdot \sum_{i=1}^{\lfloor n/2 \rfloor} ((i \cdot d)^2 \cdot (1 + 2) + 2 \cdot \sum_{j=1}^{i-1} ((d \cdot i)^2 + (d \cdot j)^2)) \quad (\text{III})$$

$$\text{PostRefineSqDist}[d, n] = 4 \cdot \left( \text{PreRefineSqDist} \left[ d, \left\lfloor \frac{n}{2} \right\rfloor \right] - \left( \left\lfloor \frac{n}{4} \right\rfloor d \right)^2 - 2 \cdot \sum_{i=1}^{\lfloor \frac{n}{4} \rfloor} \left( (i \cdot d)^2 + \left( \left\lfloor \frac{n}{4} \right\rfloor \cdot d \right)^2 \right) \right) \quad (\text{IV})$$

$$\begin{aligned} \text{DistanceSavingVal}[d, n] \\ = \text{PreRefineSqDist}[d, n] - \text{PostRefineSqDist}[d, n] \end{aligned} \quad (\text{V})$$

$$\text{DistanceSavingPer}[d, n] = \frac{\text{DistanceSavingVal}[d, n]}{\text{PreRefineSqDist}[d, n]} \quad (\text{VI})$$

Where  $d$  is the height (and width) of each tile in the mesh and  $n^2$  is the amount of tiles. Hence, the reduction of the total squared distances in the example above can be easily calculated. For example, if  $d = 1$  the total squared distances of the pre-refinement WSN are equal to 100, while the total distances of the post-refinement are equal to 30, meaning 70% reduce. This reduction exists in all positive optional value of  $d$  as shown in Figure 3. Figure 3 shows that this reduction of percentage is constant, and that the total squared distances before and after the refinement are linear. ■



**Figure 3.** The total squared distances in the WSN as function of the distance coefficient ( $d$ ) before the ACR-HCC refinement (red), after the ACR-HCC refinement (blue) and the saving between the two (green).

As previously explained, in order to examine if the refinement achieved its goals, we need to focus on the distances between the nodes in the WSN. In this case study, each black line between two nodes is equal for ease of explanation ( $d$ ), and we assume the *InitCH* transmitting to a specific space in both cases (distance  $x$ ). According to formulas (1) and (2) before refinement of the WSN is equal to:

$$\begin{aligned} 24 \cdot (E_{elec} + \epsilon_{fs} \cdot d_{NCH \rightarrow InitCH}^2) + n_{NCH \rightarrow InitCH} E_{elec} + (n_{NCH \rightarrow InitCH} + \\ 1) E_p + \alpha (E_{elec} + \epsilon_{fs} \cdot d_x^2) \end{aligned} \quad (\text{I})$$

While after refinement, the WSN is equal to:

$$\begin{aligned} (5 \cdot 4) \cdot (E_{elec} + \epsilon_{fs} \cdot d_{NCH \rightarrow CH}^2) + 4 \cdot (n_{NCH \rightarrow CH} E_{elec} + (n_{NCH \rightarrow CH} + \\ 1) E_p + \alpha (E_{elec} + \epsilon_{fs} \cdot d_{CH \rightarrow InitCH}^2)) + (n_{CH \rightarrow InitCH} E_{elec} + \\ (n_{CH \rightarrow InitCH} + 1) E_p + \alpha (E_{elec} + \epsilon_{fs} \cdot d_x^2)). \end{aligned} \quad (\text{II})$$

It is possible to see that even if  $\alpha = 1$ , although the total amount of  $E_{elec}$  is the same in both calculation, the transformation of the 4 NCHs into CHs created an additional overhead of  $5E_p$  ( $((5 + 1) \cdot 4)E_p + (4 + 1)E_p$  instead of  $(24 + 1)E_p$ ). Nevertheless, Because of the ACR-HCC algorithms hierarchy creation, the total squared distances in the entire WSN constantly reduced by 70%.

Therefore, in order to determine if an ACR refinement is worthwhile, the difference between the total  $E_t$  of the WSN before and after a refinement should be bigger than the difference between the total  $E_p$  of the WSN before and after a refinement. This refinement balance can be formalized as the following formula (4):

$$\epsilon_{fs} \cdot (\sum d_{Before Refinement}^2 - \sum d_{After Refinement}^2) > (4 \cdot ((n_{NCH \rightarrow CH} + 1) + (n_{CH \rightarrow InitCH} + 1)) - (n_{NCH \rightarrow InitCH} + 1)) \cdot E_p \quad (4)$$

It is possible to see that the saving between before the refinement and after the refinement cross the  $4E_p$  (i.e.  $4 \cdot 5 \cdot 10^{-9}$ ) at the very beginning of the measurements, when  $d = \frac{4 \cdot 5 \cdot 10^{-9}}{10^{-10}} = \sqrt{200} = \sim 14$  (m), meaning that the ACR-HCC refinement algorithm proven to be efficient in many cases.

#### 4 A Local Version of the Adaptive Clustering Algorithm

In this section we present an alternative algorithm that makes decision based on local behavior of clusters, rather than taking into account the behavior of the entire network. In this way we can significantly improve the running time of our algorithm without compromising energy efficiency significantly. The algorithm is executed by all CHs in parallel. Initially, each CH is provided with a threshold value  $t$  of a plausible energy use. If these values are unknown in the beginning of the execution, they can be computed using a single execution of the ACR algorithm of Section 2. This combination of an initial global execution with numerous local executions following it, is still more efficient than performing several executions of the global ACR algorithm of Section 2. In the initial configuration all these values  $t$  are the same, and represent a balanced environment. The algorithm can start from any cluster-hierarchy tree, where the simplest configuration is a single CH, which is the root (equivalently, a single leaf). Once an energy use of a leaf  $v$  reaches  $3t$ , we perform a local refinement in the cluster of  $v$ . This results in adding new CHs to the tree as leaves that become the children of  $v$ . These new leaves correspond to the newly formed clusters. This refinement results in a better energy use in each such newly formed cluster, specifically, bounded by  $3t \cdot \frac{3}{10} < t$  instead of  $3t$  (See Theorem 4). In other words, we balance clusters of excess energy-use by decomposing them into smaller clusters that require less energy.

Once the average energy consumption in the children of a CH node  $u$  whose all children are leaves becomes less than  $t/5$ , a coarsening operation is performed. (This operation is the opposite of refinement). Specifically, the clusters represented by  $u$  and its children are merged into a single cluster. Then  $u$  becomes its CH, and former children of  $u$  become NCHs. Consequently, the energy use of the newly-formed larger

cluster grows, but the tree-distance between the root to some leafs decreases. This completes the description of the algorithm. Its pseudo-code is provided below, and it is executed periodically by each CH.

The Local Version of the Adaptive Clustering Algorithm

**Local-ref**(Node  $v$ , Threshold  $t$ )

1. **If**  $v$  is a leaf and  $\text{energy\_use}(v) \geq 3t$ 
  - 1.1 **Refine**( $v$ )
  - 1.2 **Add** new CHs as children of  $v$
2. **If** all children of  $v$  are leafs and  $\text{average\_energy}(\text{children}(v)) < t/5$ 
  - 2.1 **Coarsen**( $v$ )
  - 2.2 **Remove** the children of  $v$  from the tree and mark  $v$  as a leaf.

**Theorem 5:** *Each execution of Local-ref requires a constant number of communication rounds.*

*Proof.* A refinement operation adds a set of children to the tree. All these children have a common parent, which is the executing cluster, and thus their creation requires a constant number of communication rounds. A coarsening operation is performed on a node whose all children are leafs. Therefore, the node can communicate with all nodes in its sub-tree within a single round. Hence, a constant number of rounds is required to complete the operation. ■

**Theorem 6:** *Local-ref preserves a balance of energy use between  $t/5$  and  $3t$  in each cluster.*

*Proof.* Each cluster whose energy use exceeds  $3t$  performs the refinement procedure which improve energy use by 70% (See Theorem 4). Thus, after refinement of a cluster, the energy use in all its sub-clusters reduces.

If the average energy use of the leafs of a cluster goes below  $t/5$ , it means that each of the four leafs has energy use at most  $4t/5$ . Therefore, after coarsening, we have a single cluster instead of four clusters, and its energy use is at most  $3t$ . ■

**Theorem 7:** *All longest paths from the root to a leaf contain a cluster with energy use at least  $t/5$ .*

*Proof.* Suppose for contradiction that there is a longest path  $P$  from the root to a leaf, for which all clusters have energy use less than  $t/5$ . Then there is a node on the path  $P$  whose all children are leafs. Since the energy use of all these children is less than  $t/5$ , the coarsen procedure would be invoked, which would eliminate these leaf. In particular, one of these leafs is an endpoint of  $P$  which would be eliminated. Hence the path  $P$  does not remain in the tree. This is a contradiction. ■

When  $L$  is the maximum load in the network, consider the following theorem.



**Theorem 8:** *The maximum depth of a tree is bounded by  $O\left(\log \frac{L}{t}\right)$ , and the maximum tree size is bounded by  $2^{O\left(\log \frac{L}{t}\right)}$ .*

*Proof.* Suppose we have maximum load  $L$ . The load may be divided to sub-clusters. Every such step will decrease the load, allowing no more than  $L/3$  load. The algorithm will repeat this division  $q$  times, until the load will be less than  $3t$ . Thus, we obtain the following formula:  $L \cdot \left(\frac{1}{3}\right)^q \leq 3t \Rightarrow q \leq O\left(\log \frac{L}{t}\right)$ . Because each node may have only 4 children, the total size of the tree is bounded by  $2^{O\left(\log \frac{L}{t}\right)}$ . ■

These theorems demonstrate a correlation between the size of the tree and the load in the network. Specifically, whenever the load is light, the tree remains small. If the loads of certain areas become heavy, the tree grows branches that correspond to these areas. Therefore, the size of the tree corresponds to the load in the entire network. In other words, the tree grows only when needed, which allows reducing the cost of maintaining the tree structure. This is in contrast to a fixed tree that does not take into account the load in the network.

## 5 Conclusions and Future Work

Those results and analysis lead to the conclusion that it would be beneficial to use the ACR algorithm along with the HCC algorithm in WSNs with differential load for maximization of the network lifetime as well as connectivity.

This paper opens a number of prospective directions for future research. One immediate direction is to explore how the ACR algorithm is reacting along with different WSN hierarchical clustering algorithms, and what exactly does that mean in aspects of cost, complexity, energy-efficiency and connectivity of the network. Another direction is to understand how to optimize other WSN clustering algorithms which are not based on a hierarchical formation of the WSN using ACR algorithm.

Finally, we also expect that in the near future the ACR-HCC will be implemented for the specific problem that the algorithm was designed for. A comparison of the empirical benchmark results to those presented in this paper would be a fertile ground for further research and development.

## Acknowledgments

This work was supported by the Lynn and William Frankel Center for Computer Science, the Open University of Israel's Research Fund, and ISF grant 724/15.

## References

- [1] El Emary IM, Ramakrishnan S, editors. **Wireless Sensor Networks: From Theory to Applications**. CRC Press; 2013 Aug 28.

- [2] Dargie W, Poellabauer C. **Fundamentals of wireless sensor networks: theory and practice**. John Wiley & Sons; 2014 Nov 5.
- [3] Rault T, Bouabdallah A, Challal Y. **Energy efficiency in wireless sensor networks: A top-down survey**. *Computer Networks*. 2014 Jul 4;67:104-22. Huang P, Xiao L, Soltani S, Mutka MW, Xi N.
- [4] Basagni S, Naderi MY, Petrioli C, Spenza D. **Wireless sensor networks with energy harvesting**. *Mobile Ad Hoc Networking: The Cutting Edge Directions*. 2013 Mar 5:701-36.
- [5] Tuna G, Gungor VC, Gulez K. **Energy harvesting techniques for industrial wireless sensor networks**. In *Industrial Wireless Sensor Networks: Applications, Protocols, Standards, and Products*, GP Hancke and VC Gungor, Eds. 2013 Apr 4:119-36.
- [6] Sachan V, Imam S, Beg M. **Energy-efficient communication methods in wireless sensor networks: A critical review**. *International Journal of Computer Applications*. 2012 Feb 29;39(17).
- [7] Zhang P, Xiao G, Tan HP. **Clustering algorithms for maximizing the lifetime of wireless sensor networks with energy-harvesting sensors**. *Computer Networks*. 2013 Oct 4;57(14):2689-704.
- [8] Younis M, Senturk IF, Akkaya K, Lee S, Senel F. **Topology management techniques for tolerating node failures in wireless sensor networks: A survey**. *Computer Networks*. 2014 Jan 15;58:254-83.
- [9] Li M, Li Z, Vasilakos AV. **A survey on topology control in wireless sensor networks: Taxonomy, comparative study, and open issues**. *Proceedings of the IEEE*. 2013 Dec;101(12):2538-57.
- [10] Viswanathan S, Jayakumar D. **The Role of Wireless Sensor Network in Tracking Wild Animals Crossing Forest Boundaries**. *Programmable Device Circuits and Systems*. 2016;8(5):105-7.
- [11] Voeten MM, Van De Vijver CA, Olf H, Van Langevelde F. **Possible causes of decreasing migratory ungulate populations in an East African savannah after restrictions in their seasonal movements**. *African Journal of Ecology*. 2010 Mar 1;48(1):169-79.
- [12] Liu NH, Wu CA, Hsieh SJ. **Long-term animal observation by wireless sensor networks with sound recognition**. In *International Conference on Wireless Algorithms, Systems, and Applications 2009 Aug 16* (pp. 1-11). Springer Berlin Heidelberg.
- [13] Banerjee S, Khuller S. **A clustering scheme for hierarchical control in multi-hop wireless networks**. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE 2001 (Vol. 2, pp. 1028-1037)*. IEEE.
- [14] Berger MJ, Olinger J. **Adaptive mesh refinement for hyperbolic partial differential equations**. *Journal of computational Physics*. 1984 Mar 31;53(3):484-512.
- [15] Liao WK, Liu Y, Choudhary A. **A grid-based clustering algorithm using adaptive mesh refinement**. In *7th Workshop on Mining Scientific and Engineering Datasets of SIAM International Conference on Data Mining 2004 Apr 22* (pp. 61-69).