Conception of a wireless sensor network for decision making based on Data mining methods

Massinissa Saoudi

▶ To cite this version:

Massinissa Saoudi. Conception of a wireless sensor network for decision making based on Data mining methods. Networking and Internet Architecture [cs.NI]. Université de Bretagne occidentale - Brest; University College Dublin, 2017. English. NNT : 2017BRES0065 . tel-01688802

HAL Id: tel-01688802
https://theses.hal.science/tel-01688802

Submitted on 19 Jan 2018

THÈSE / UNIVERSITÉ DE BRETAGNE OCCIDENTALE

sous le sceau de l'Université Bretagne Loire

pour obtenir le titre de

DOCTEUR DE L'UNIVERSITÉ DE BRETAGNE OCCIDENTALE

Mention : Informatique

École Doctorale MathSTIC

présentée par

Massinissa SAOUDI

Préparée au Lab-STICC UMR CNRS 6285

Conception d'un réseau de capteurs sans fil pour des prises de décision à base de méthodes du Data mining.

Conception of a wireless sensor network for decision making based on Data mining methods

Massinissa Saoudi, PhD Thesis

This thesis is submitted to University College Dublin
in fulfilment of the requirements for the degree of

Philosophiæ Doctor

College of School of Computer Science
School of Insight Centre for Data Analytics

Head of School:	Professor Pádraig Cunningham
Supervisor:	Professor M-Tahar Kechadi
Supervisor:	Associate Professor Ahcène Bounceur
Joint supervisor:	Professor Reinhardt Euler

September 2017

ACKNOWLEDGEMENTS

It is my pleasure to thank many people who made this thesis possible.

I would like to express my deeply-felt gratitude to my thesis supervisors, Ahcène Bounceur, Reinhardt Euler from UBO in France, and M-Tahar Kechadi from UCD in Ireland, for their warm encouragement and thoughtful guidance. The enthusiasm they have for the research was contagious and motivational to me. Their guidance taught me a lot about conducting good research. Without their support, this dissertation would not have been possible.

I would like to thank all the jury members for reading my manuscript and their interest on my research. Without their help, my thesis couldn't have reached this far. So, I thank Mr. Neil Hurley, Mr. Pietro Manzoni, Mr. Congduc Pham, Mr. Frank Singhoff and Mr. Rahim Kacimi.

I had a great pleasure to thank all my colleagues and friends inside LabSTICC, Insight and also outside for the excellent and truly enjoyable ambiance.

The last but not the least, I would like to thank my family for the unselfish support during the many years of my doctoral study. The dissertation would not have happened without their understanding and sacrifice.

ABSTRACT

Recently, Wireless Sensor Networks (WSNs) have emerged as one of the most exciting fields. However, the common challenge of all sensor network applications remains the vulnerability of sensor nodes due to their characteristics and also the nature of the data generated which are of large volume, heterogeneous, and distributed. On the other hand, the need to process and extract knowledge from these large quantities of data motivated us to explore Data mining techniques and develop new approaches to improve the detection accuracy, the quality of information, the reduction of data size, and the extraction of knowledge from WSN datasets to help decision making. However, the classical Data mining methods are not directly applicable to WSNs due to their constraints. It is therefore necessary to satisfy the following objectives: an efficient solution offering a good adaptation of Data mining methods to the analysis of huge and continuously arriving data from WSNs, by taking into account the constraints of the sensor nodes which allows to extract knowledge in order to make better decisions. The contributions of this thesis focus mainly on the study of several distributed algorithms which can deal with the nature of sensed data and the resource constraints of sensor nodes based on the Data mining algorithms by first using the local computation at each node and then exchange messages with its neighbors, in order to reach consensus on a global model. The different results obtained show that the proposed approaches reduce the energy consumption and the communication cost considerably which extends the network lifetime. The results also indicate that the proposed approaches are extremely efficient in terms of model computation, latency, reduction of data size, adaptability, and event detection.

RÉSUMÉ

Les réseaux de capteurs sans fil (RCSFs) déterminent un axe de recherche en plein essor, puisqu'ils sont utilisés aujourd'hui dans de nombreuses applications qui diffèrent par leurs objectifs et leurs contraintes individuelles. Toutefois, le dénominateur commun de toutes les applications de réseaux de capteurs reste la vulnérabilité des nœuds capteurs en raison de leurs caractéristiques et aussi de la nature des données générées. En effet, les RCSFs génèrent une grande masse de données en continue à des vitesses élevées, hétérogènes et provenant d'emplacements répartis. Par ailleurs, la nécessité de traiter et d'extraire des connaissances à partir de ces grandes quantités de données nous ont motivé à explorer l'une des techniques conçues pour traiter efficacement ces ensembles de données et fournir leurs modèles de représentation. Cependant, parmi les techniques utilisées pour la gestion des données, nous pouvons utiliser les techniques de Data mining. Néanmoins, ces méthodes ne sont pas directement applicables aux RCSFs à cause des contraintes des nœuds capteurs. Il faut donc répondre à un double objectif : l'efficacité d'une solution tout en offrant une bonne adaptation des méthodes de Data mining classiques pour l'analyse de grosses masses de données des RCSFs en prenant en compte les contraintes des nœuds capteurs, et aussi l'extraction d'un maximum de connaissances afin de prendre des décisions meilleures. Les contributions de cette thèse portent principalement sur l'étude de plusieurs algorithmes distribués qui répondent à la nature des données et aux contraintes de ressources des nœuds capteurs en se basant sur les techniques de Data mining. Chaque nœud favorise un traitement local des techniques de Data mining et ensuite échange ses informations avec ses voisins, pour parvenir à un consensus sur un modèle global. Les différents résultats obtenus montrent que les approches proposées réduisent considérablement la consommation d'énergie et les coûts de communication, ce qui étend la durée de vie du réseau. Les résultats obtenus indiquent aussi que les approches proposées sont extrêmement efficaces en termes de calcul du modèle, de latence, de réduction de la taille des données, d'adaptabilité et de détection des événements.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

ONE

INTRODUCTION

Wireless Sensor Networks form an emerging technology of networks that can revolutionize and improve our everyday lives. WSNs comprise large numbers of small wireless devices deployed over a physical environment that actively cooperate in order to accomplish one or more tasks. The sensor nodes compromise a set of sensors which can measure large and diverse information, such as light, temperature, motion, humidity, pressure, and many others, for a better understanding of the environment. The benefits of the WSNs such as low cost, easy deployment, high fidelity sensing, and self-organization allow their use in various applications including environmental monitoring [3], habitat tracking [4], health monitoring [5], and military surveillance [6]. These applications have the ability to generate a large amount of measurements and normally involve sending large sensed data to a base station for post analysis. The collected data could sometimes be locally processed before being sent in the network, and could involve intermediate sensor nodes for further processing of the data. Finally, the sensory data are integrated centrally at the base station to conclude the status of the observed environment at the base station. The base station performs an efficient detection based on traditional processing methods. These conventional approaches, however, suffer from many limitations due to resource constraints of the sensor nodes. The computational power and speed of the base station can create a processing blockage and can cause a failure of the total system if the base station fails. In addition, relaying all sensed data of geographically dispersed sensor nodes to a centralized base station is generally ineffective as it requires a significant communication overhead leading to resource depletion and reduction of lifetime of the network.

1.1 Motivations

The field of wireless sensor networks has become a focus of intensive research in recent years and various theoretical and practical questions have been addressed. The purpose of WSNs is not just the reliable transmission of sensed data from the environment to the base station, but also the detection and the prediction of events from large sets of sensed data, such as forest fire detection. Most of the current methods focus on solving the problem of managing the resources efficiently for a small WSN, with studies conducted on simulation. Interpreting global events from large volume, heterogeneous, and distributed datasets, motivated us to explore the Data mining techniques and to develop new approaches to improve the detection accuracy, the quality of information, the reduction of data size, and extraction of knowledge from WSN datasets for decision making. In addition, to improve the various performances and operations of the WSN. Data mining techniques are traditionally designed to efficiently process static datasets, and to compute their representing model. However, these Data mining approaches despite handling large datasets, and could not be adapted to the distributed nature of the WSNs. Further, online, real-time, and distributed data processing are therefore required taking into account the resource constraints of sensor nodes.

Distributed Data mining algorithms offer an alternative approach to address the problem of processing data using distributed resources. It pays careful attention to distributed resources of data, computing, energy and communication in order to use them in an optimal solution. Distributed Data mining methods provide solutions to these constraints by placing aspects of the Data mining process such as data aggregation, and modeling into individual sensors, as well as clusters of sensors. These activities and the placement in the sensor network vary according to the type of Data mining techniques such as classification, prediction, association rules and clustering.

The main motivation for this thesis is how Data mining techniques can be used to address different issues of WSNs, focused in particular on the tradeoff between dynamic topology, data accuracy, event detection, latency, energy efficiency and network load in data collection tasks. This thesis also investigates a reduction of the amount of data transmitted over the network to eliminate redundancies and to extend WSN applications lifetime.

1.2 Problem statement

Given a distributed wireless sensor network system consisting of resource constrained sensor nodes i.e., with limited processing power, limited energy resource, small storage memory, with dynamic neighbors, organized as a huge network (up to thousands of sensor nodes), and with random deployment. Each node is tasked with processing local data using a distributed Data mining algorithm and updating the system regarding its acquired information, and local decision along with all other nodes in the system. This research aims to achieve many goals coherently, by exploiting the relationships within a large and continuously arriving sensor dataset in a WSN, discovering the hidden relationships between them, reducing the redundancies and data size, and achieving also the energy efficiency, event detection and global knowledge extraction capabilities. This research needs to develop some new distributed algorithms and Data mining techniques which can allow to model, to predict and to evaluate measurements and states of wireless sensor networks, and to detect useful information that exists in the physical environment in order to help decision making. Therefore, our research aims at answering the following questions :

- It is possible to develop a new method that is distributed, consuming less resources, producing high quality results, and that has low complexity ?
- It is possible to develop adaptive Data mining algorithms to address the dynamically changing requirements of a WSN and address the challenges

corresponding to energy efficiency, event detection accuracy, and logical topology ?

- It is possible to update Data mining techniques for extracting useful knowledge and patterns in real-time from the data of all nodes, and to be executed in the system with all nodes participating in a collaborative distributed computation with efficient use of computational power, energy, and communication ?
- It is possible to process huge, heterogeneous, geographically distributed datasets and to reduce redundancy while efficiently using the sensor nodes' resources ?

1.3 Objectives

The ultimate objective of this thesis is to develop decision making systems by means of Data mining techniques and wireless sensor networks in order to improve WSNs operations and extract useful information and knowledge for the end-users. This thesis is open for various application domains and has been achieved with the following objectives :

- Design of an energy-efficient communication algorithm for wireless sensor networks by exploiting local information and cooperative decision of sensor nodes that can adapt to the dynamicity of the network in the case of a node failure, for example.
- Design of an efficient data reduction technique which can represent the information without redundancies within fewer subsets in order to reduce a large amount of sensed data.
- Development of an energy-efficient distributed clustering technique to analyze large, heterogeneous and distributed datasets based on the spatial correlation without any required assumptions during data collection for WSNs.

- Improvement of a WSN topology based on sensor node constraints that can handle the scalable network, data aggregation and communication by dividing the network into different hierarchical levels.

- Development of energy-efficient adaptive classification techniques for event detection in which the sensors are able to ensure a local computation which create the possibility of training, uses predictors in a distributed way and achieves coherently the objectives of low complexity, high accuracy, and low latency.

1.4 Contributions

Our work aims to overcome the challenges imposed by the WSN and the Data mining techniques through data management while extracting useful information for decision making in real life applications. In addition, we favor the distributed processing applied on local data and based on low complexity algorithms and adaptable Data mining techniques for the extension of the network lifetime, event detection, data collection and their reduction. Those are the pillars of interest of this thesis that analyzes the existing research and proposes efficient solutions and improved algorithms. The specific contributions of the thesis are:

- A new generic efficient algorithm, called *Least Polar-angle Connected Node (LPCN)*, is proposed to find the boundary vertices of a connected Euclidean graph (data points), which can be used to design and analyze WSN systems and reduce sensed data in combination with distributed Data mining algorithms. In addition, based on this algorithm, we have proposed a new distributed solution called D-LPCN for boundary nodes detection in WSN, optimizing energy consumption, computational power, and communication bandwidth and monitoring the boundaries of strategic and sensitive sites. We have shown the effectiveness and the low complexity of these algorithms compared to the existing methods. We have also shown that the D-LPCN algorithm can provide the highest communication savings which extends the network lifetime, the adaptability to any topology architecture and fault tolerance.

- A new approach that addresses the data collection, within huge data volumes generated from the WSN is proposed. The most critical design issue at this stage is the data collection among multiple sensor nodes. Existing redundant data at this collection lead to huge packet size which overheads the network and depletes the energy thus reducing the lifetime of the network. Hence, it is essential for sensor networks to be able to detect and clean redundantly transmitted data from the nodes to the sink. This contribution introduces a hierarchical data aggregation model to achieve this goal. Three layers of processing are introduced: the node level (local model), the aggregation level (sub-global model), and the base station (global model). These algorithms aim to optimize the volume of data transmitted by saving energy consumption and reducing communication on the network level. At the first level, the AGNES data clustering algorithm combined with the LPCN algorithm allow to avoid each sensor node to send its whole dataset to the base station. At the second level, a sensor node (cluster-head) collects the contours from its associated nodes where a new part of the merging aggregation techniques is explored. At this level, our approach identifies the similarity between datasets generated by neighboring nodes, by merging the overlapping contours and deleting included or identical contours, and then sending to the higher level within the tree topology. The objective is to detect similarities between nearby sensor nodes, and to integrate their captured data into one record while preserving information integrity. At the last level, the base station merges the received contours to obtain the global view of the network. Our experimental results indicate that the proposed approach is flexible, energy saving, appropriate for the aggregation of large datasets arising at different nodes, and that it reduces the original size of the datasets compared to the existing methods.

- A high spatio-temporal resolution for a forest fire danger monitoring WSN application using the distributed classification Data mining WSN model is proposed. In this kind of time critical applications, the event must be detected early with high accuracy and low latency in order to reduce the threats and damages. The idea is to partition the node set into clusters of sensors so that each node periodically acquires temperature, light, humidity and smoke data to individually detect fires using prediction algorithms. Once a fire is detected, the corresponding node will send an alert to its cluster-head. The cluster-head confirms the alert by receiving alerts from the sensors located at short distance area. This alert will then be routed via gateways and other cluster-heads to the sink in order to inform the firefighters. Using Data mining techniques in the process of pattern discovery in large data sets this is not often so easy. We have focused on a comparative analysis of various Data mining techniques in terms of energy consumption, processing time, and accuracy before to select a suitable algorithm, most appropriate for a particular application of fire detection. In comparison to its best rivals in the literature, extensive simulation results show that our approach provides a fast reaction to forest fires reducing at the same time the overall energy consumption of the network.

These contributions can be applied to WSN design and deployment for online distributed Data mining in WSN applications by an efficient use of resources. Furthermore, this research provides a foundation for future investigation of huge and heterogeneous data processing for environmental monitoring.

1.5 Outline of Thesis

This manuscript is structured in two main parts: the first concerns the state of the art and the second describes our contributions. Figure 1.1 proposes an approach of reading this thesis and indicates the different connections between the chapters. There are 6 chapters, which include Introduction, Background, Boundary Detection, Distributed data clustering approach in WSNs, Data mining techniques for forest fire detection using WSNs, and Conclusion with future scope.

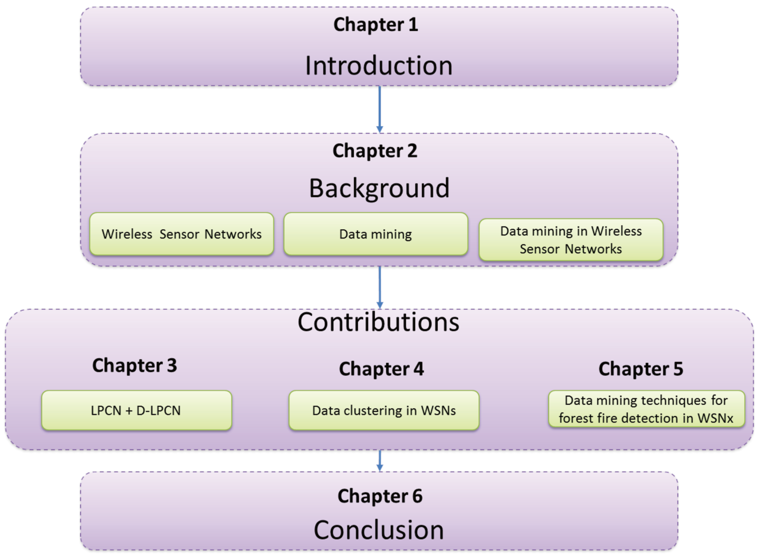


Figure 1.1: Outline of thesis.

In **Chapter one**, we start with a brief introduction of WSNs, Data mining techniques, describe the motivation behind this research, state the specific problem addressed, and highlight the contributions of the thesis and give an overview of the structure of the thesis.

In **Chapter two**, we present an overview of WSN architecture, energy conservation strategies, Data mining techniques and WSN relevant Data mining techniques. We also analyze the state-of-the-art in Data mining techniques literature for WSNs, and highlight the challenges.

In **Chapter three**, we describe the problem of boundary detection in datasets and in WSNs. For these problems, we propose two new solutions, the LPCN algorithm allowing to find the boundary points of a set of connected data points and D-LPCN which represents a new distributed solution in WSNs to detect boundary nodes.

In **Chapter four**, we propose a new distributed data clustering approach for the WSN system. It describes the details of data aggregation in a WSN with an appropriate topology using the distributed data clustering technique combined with the LPCN algorithm in order to cope with a huge and redundant sensed dataset.

In **Chapter five**, we present a new approach to evaluate performance and we analyze the Data mining techniques in WSNs. For these reasons, we have proposed the case study for distributed Data mining techniques for forest fire detection in WSNs.

In **Chapter six**, we conclude this thesis and outline the directions for future research in distributed Data mining for resource constrained systems such as WSNs.

1.6 Publications

During this study, the following seven international peer reviewed publications have been produced that include international journal and conference proceedings.

1. Saoudi, M., Bounceur, A., Euler, R., Kechadi, M. T., & Lalem, F. (2017). Tree-Based Distributed Data Clustering in Wireless Sensor Networks. Ad Hoc Networks, (Submitted).

2. Lalem, F., Bounceur, A., Madani, B., Massinissa, S., Euler, R., Kechadi, M. T., & Sevaux, M. (2017). LPCN: Least Polar-angle Connected Node Algorithm to Find a Polygon Hull in a Connected Euclidean Graph. JNCA, Journal of Network and Computer Applications., DOI: 10.1016/j.jnca.2017.05.005.

3. Saoudi, M., Lalem, F., Bounceur, A., Euler, R., Kechadi, M. T., Laouid, A., ..., & Sevaux, M. (2017). D-LPCN: A Distributed Least Polar-angle Connected Node Algorithm for Finding the Boundary of a Wireless Sensor Network. Ad Hoc Networks, 11/2016; 56:56-71., DOI:10.1016/j.adhoc.2016.11.010.

4. Lalem, F., Bounceur, A., Kacimi, R., Euler, R., & Massinissa, S. (2016). Faulty Data Detection in Wireless Sensor Networks Based on Copula Theory. International conference on Big Data and Advanced Wireless technologies (BDAW'2016), Blagoevgrad, Bulgaria Nov 2016,; 11/2016, DOI:10.1145/3010089.3010114.

5. Saoudi, M., Lounis, M., Bounceur, A., Euler, R., & Kechadi, T. A Parallel Data Mining Algorithm for PageRank Computation. International conference on Big Data and Advanced Wireless technologies (BDAW'2016), Blagoevgrad Bulgaria; 11/2016, DOI:10.1145/3010089.3010118.

6. Saoudi, M., Bounceur, A., Euler, R., Kechadi, T., & Cuzzocrea, A. Energy-Efficient Data Mining Techniques for Emergency Detection in Wireless Sensor Networks. IEEE International Conference on Cloud and Big Data Computing (CBDCom'2016); 07/2016, DOI:10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0123.

7. Saoudi, M., Bounceur, A., Euler, R., & Kechadi, T. Data Mining Techniques Applied to Wireless Sensor Networks for Early Forest Fire Detection. The International Conference on Internet of Things and Cloud Computing (ICC 2016), Cambridge, United Kingdom; 03/2016, DOI:10.1145/2896387.2900323.

8. Bounceur, A., Euler, R., Benzerbadj, A., Lalem, F., Saoudi, M., Kechadi, T., & Sevaux, M. Finding the polygon hull in wireless sensor networks. European Conference on Operational Research, University of Strathclyde, Glasgow, UK, Invited talk, EURO, Glasgow, UK; 07/2015.

CHAPTER

TWO

BACKGROUND

2.1 Introduction

This chapter introduces and analyses the literature on Data mining techniques in the context of Wireless Sensor Networks. Initially, we will highlight several aspects of WSNs, which explain the technical context of our work. We will also summarize the different capabilities, limitations and challenges of WSNs with special attention given to the resource constraints of the sensor nodes. In particular, we focus on the energy constraint to design appropriate methods which maximize the network lifetime. In addition, we will give a general view of the Data mining context and explain some important techniques. Following the description of both the WSN and Data mining contexts, we will present a classification of existing Data mining methods that enable the extraction of knowledge from continuous and rapid WSN datasets. In our work, we will focus on distributed techniques which are suitable to the nature of WSNs. A review of the foundations of each of these fields is presented below.

2.2 Wireless Sensor Networks

Rapid improvements in wireless communication and electronic technologies have enabled the development of tiny, low-cost, low-power, multi-functional devices, known as sensor nodes. These sensor nodes can sense, measure, and collect information from the environment. They can also take decisions based on local processes and transmit a result to the user. Sensor nodes (also known as motes) are battery-powered devices equipped with one or more sensors, a processor, a memory, a power supply, and wireless interfaces with which they can communicate with one another to form a network. A WSN consists of many sensor nodes, which transmit data, collected from the environment, over a network to a base station (e.g., a sink, a handheld device, or an access point to a fixed infrastructure). A base station provides a connection to the wired world where the collected data are processed, analyzed and presented for useful applications. Figure 2.1 illustrates a typical WSN configuration, which consists of several sensor nodes and a sink. The

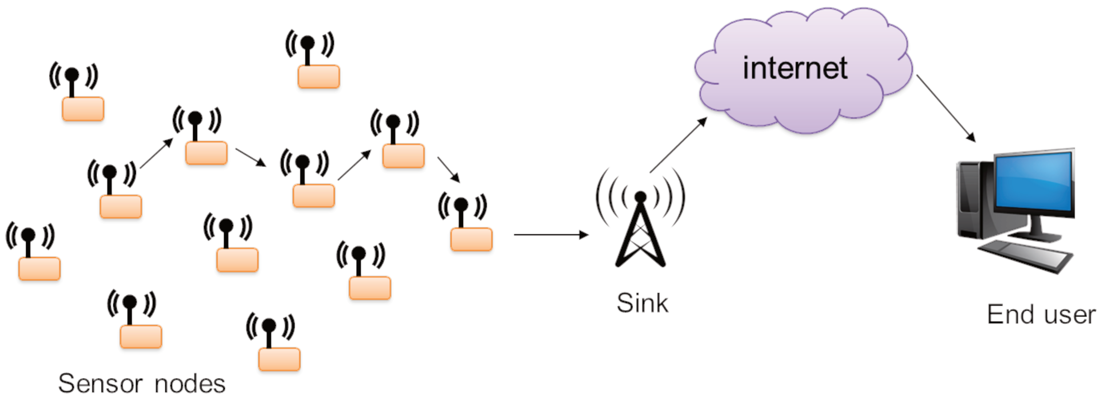


Figure 2.1: A WSN consisting of several sensor nodes and a sink that acts as a gateway between the sensor nodes and an end-user

ultimate objective of this section is to provide a detailed understanding of WSN technology and to present challenges that the field currently undergoes.

2.2.1 The architecture of a sensor node

As shown in Figure 2.2, a sensor node is a tiny device which includes four basic components. A sensing unit composed of one to several sensors, a processing unit, a transceiver unit and a power unit as energy source [7]. The name of each sub-device explains its functionality. In addition to these principal units, the sensor node may also be equipped with a localization system unit such as a Global Positioning System (GPS), a mobilizer allowing its mobility,..., etc.

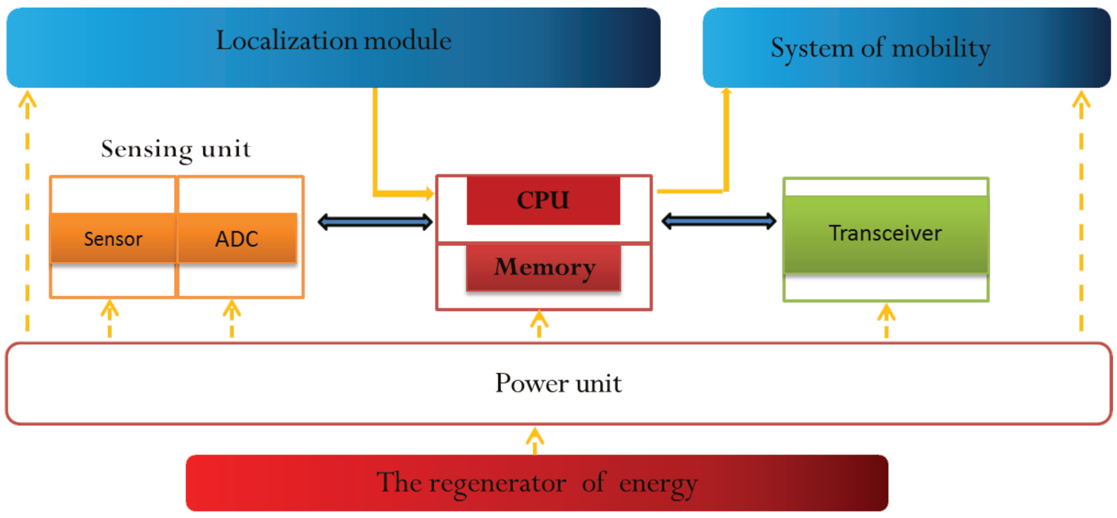


Figure 2.2: Architecture of a sensor node.

1. **Sensing Unit :** is responsible for collecting information from the environment. In the network, sensors can be used to detect or monitor a variety of physical parameters or conditions, for example, light, sound, humidity, pressure, temperature, soil composition, air or water quality, characteristics of objects such as size, weight, position, speed, direction and their motion. The analog signals produced by the sensors are converted into digital signals by the Analog to Digital Communication (ADC) and sent to the processing unit.

2. **Processing Unit :** includes a micro-controller plus a flash memory. The controller performs tasks, processes data and controls other functionalities of a sensor node. Memory is used to store programs (instructions executed by the processor) and data (raw and processed sensor measurements) [8].

3. **Transmission Unit :** The transmitter and receiver are combined into a single device called transceiver. Sensor nodes use bands to communicate between them. The bands are defined by the International Telecommunication Union-Radio that may differ due to variations in national radio regulations. The most specific radio frequencies used in WSNs are : 173, 433, 868 and 915 MHz and 2.4 GHz [7]. In some special cases, a sensor node can have more than one transceiver and thus be able to operate in various wireless

network communications. These nodes usually take more responsibility than the base station node or the gateway.

4. **Power Unit :** One of the most important components of a sensor node is the power unit. It is a source of energy which is produced by tiny batteries. These batteries allow to feed the different units to perform their tasks. The requirement in a WSN is that the size of the battery should be as small as possible and the energy will be efficient, for example, if two small AA sized batteries of $1.2V$ each are employed as energy source.

2.2.2 Standards and simulators for WSNs

Wireless sensor standards have been developed in order to build large low cost products to facilitate the development of different applications for WSNs. These standards present the functions and protocols necessary for sensor nodes to interface with a variety of networks. For this goal, a lot of efforts have been made to build standards such as IEEE 802.15.4 [9], ZigBee [10], Wibree [11], WirelessHART [8]. The following paragraphs describe the standards used in this thesis in more detail.

1. **The IEEE 802.15.4 Standard :** The 802.15.4 [9] is a standard developed by the IEEE 802.15 Task Group 4, which defines the physical (PHY) and Medium Access Control (MAC) layers. It is a simple and flexible protocol which has the following characteristics [12] :

   - Two addressing modes: 16 bit short and 64 bit IEEE addressing.
   - Data rates of 250 kbps (worldwide), 40 kbps, and 20 kbps.
   - CSMA-CA is a protocol which deals with transmissions after a collision has occurred.
   - Automatic network establishment by the coordinator.
   - Fully handshaking protocol for transfer reliability.
   - Some 16 channels in the 2.4 GHz band, 10 channels in the 915 MHz band, and 1 channel in the 868 MHz band.

    - Power management to ensure a low power consumption.

2. **The ZigBee Standard :** The ZigBee standard is maintained by the ZigBee Alliance [10]. The ZigBee standard is developed on top of the IEEE 802.15.4 standard and defines the network and application layers. It is a simple, low-cost, and low-power wireless communication technology used in many distributed applications. The ZigBee protocol provides mesh networking capabilities for 802.15.4 applications. In addition, it is able to deploy hundreds to thousands of devices together with different roles such as coordinator, router or end-device which is very important in WSNs.

There are many simulators in the literature to develop, simulate, and validate the proposed algorithms in WSNs, we can cite CupCarbon [13], Tossim [14], Contiki OS [15], NS-2 [16], Castalia [17], ...,etc. In the following, we describe some of these simulators which are used in our work.

CupCarbon [18] is a Smart City and Internet of Things simulator. Its objective is to design, visualize, debug and validate distributed algorithms for monitoring, environmental data collection, and to create environmental scenarios, generally within educational and scientific projects. CupCarbon offers two simulation environments. The first simulation environment is a multi-agent environment [13], which enables the design of mobility scenarios and the generation of events such as fires and gas as well as the simulation of mobiles such as vehicles and flying objects [19]. The second simulation environment represents a discrete event simulation of wireless sensor networks which takes into account the scenario designed on the basis of the first environment. It allows to generate code for a real Arduino/XBee platform from the simulation.

Tossim is a discrete event simulator for TinyOS sensor networks [14]. It can simulate the behavior of a sensor within a WSN and upload programs ready to be integrated directly into TelosB [20] sensor nodes. In the same way, Contiki-OS [15] is an open source and portable operating system designed specifically for resource limited devices such as sensor nodes. It brings the benefits of both events and thread execution models. It also supports a full TCP/IP stack via uIP and the programming abstraction "Protothreads".

2.2.3 Advantages of WSNs

Wireless sensor nodes offer numerous advantages over conventional wired sensors [21]. Sensor nodes can reduce delays in deployment and be used in many areas especially harsh and hostile environments where wired networks cannot be deployed. The availability of low-cost, reliability, accuracy, flexibility, wireless communication of sensor nodes have allowed their use in a wide range of diverse applications. In the following, a list of the advantages of WSNs is presented [7,21,22].

1. **Monitoring improvement :** WSNs can be deployed without fixed infrastructure, and are able to monitor any type of appliances in which to minimize human effort and working time. This is beneficial in terms of cost and delay to monitor the environment.

2. **Easy deployment and scalability :** A WSN can include hundreds or thousands of nodes which can be deployed in remote or hard environments. The sensor nodes are very small in size which allows to deploy them in any area to collect information that could not have been possible otherwise.

3. **Self-organization :** When sensor nodes are deployed to monitor a target area, they have the ability to organize themselves in a network, to discover the neighbor nodes and to build a network using multi-hop broadcast in a small amount of time.

4. **Flexible network architectures :** A WSN is flexible in random situations, when an additional node is needed, which can join the network without any problem and thus provide applications with high reliability.

5. **Improvement of accuracy and low latency :** In WSNs, the closely located sensor nodes can sense and collect data of the same event which

results in a better accuracy, reduces noise, detects and notifies the events quickly and as soon as possible to the base station and then to the end-user.

6. **Fault Tolerance :** In WSN, it is necessary to develop algorithms that are able to adapt to the presence of node failures resulting from destroyed or dead nodes.

2.2.4 Applications of WSNs

The use of WSNs facilitates many existing applications, in particular in the field of monitoring, and constitutes the foundation of a wide range of applications related to security, surveillance, military, medical, and environmental monitoring. They can significantly improve the accuracy and density of scientific measurements of physical phenomena because large numbers of sensors can directly be deployed where experiments are taking place without human intervention. Some existing real life applications for WSNs can be grouped into 6 main categories, as shown in Figure 2.3.

Figure 2.3: Applications of a WSN.

1. **Environmental Monitoring :** Sensors are used to monitor a variety of environmental parameters or conditions, since they can be deployed in

wild to detect behaviors of animals, the impact of climate changes, or the conditions of a levee.

2. **Military Applications :** Sensor network research was initially driven by military applications [7] for enemy tracking, surveillance, control, communications, and reconnaissance.

3. **Medical Applications :** Sensor nodes are used to improve the existing health care and to monitor patients. Each patient has tiny weight sensor nodes attached to him which allow to locate him within a hospital, at home or outside, for disease diagnosis and to monitor his health.

4. **Industrial Applications :** In industry, WSNs can be used to monitor the factorization processes or the condition of various equipments. Also, WSNs can be used for production processes such as assembly lines, monitoring and control.

5. **Security and Surveillance Applications :** WSNs can be used in many security and surveillance applications such as in movement or video, and other kinds of sensors can be deployed in buildings, airports, subways, and various critical infrastructures. WSNs can adapt to any security application which make their power.

6. **Smart Home Applications :** Smart sensor nodes have become very small and can be integrated into many devices to interact with each other and the external network via the internet, which allows end-users to manage home devices locally and remotely more easily.

2.2.5 Challenges of WSNs

The various range of WSN applications which are described in Section 2.2.4 gives rise to a number of interesting challenges due to the limited resources of the sensor node devices. For instance, nodes have to guarantee a longterm operation. Some other applications might require high data accuracy and reliability. These challenges request a higher cycle of node activity and a higher computing power. However, each sensor node in a WSN must confront the limitations in processing, storage, energy capabilities within single entities in order to assure efficient solutions. In the following, we describe the most relevant challenges in WSNs.

1. **Limited Energy Capacity :** Energy is consumed in data collection, data processing, and data communication. Sensor nodes, however, have very limited energy capacity. This constraint presents many new challenges in the development of software, hardware, and the design of network architectures and protocols for sensor networks. For these reasons, efficient use of this energy should be considered in every aspect of sensor network design in order to extend the operational network lifetime. In most cases, renewing energy is not feasible or even impossible or very costly. This constraint is driving researchers to explore how to extend the lifetime of WSNs.

2. **Limited Hardware Resources :** Sensor nodes have also limited processing and storage capacities which can only perform limited computational functionalities and store a small size of sensing data. These hardware constraints present many challenges in network protocols and algorithms design in relation to sensor networks. For these reasons, algorithms should be of low complexity and low energy consumption.

3. **Random Deployment :** Deployment [23] is the implementation of a WSN in a real world location which can be either manual or random. In the case of an inaccessible or hostile region, sensors are dropped randomly from a helicopter or in some locations sensors are placed according to some topology. In these conditions, the sensor nodes must autonomously organize themselves into a communication network before they start to perform a sensing task. Several deployment issues must be taken into account to minimize energy consumption, minimize data loss and assure self-organization of the sensor nodes without human intervention.

4. **Dynamic Environment :** The network is more vulnerable due to node failures, damages, additions, or energy depletion which cause frequent

changes to the topology. The network connectivity can be disrupted frequently due to noise or error. Therefore, the maintenance of this connectivity is a challenge to be taken into account.

5. **Data Aggregation :** Consists in collecting the sensed data from multiple sensors and to transmit them to the base station for post processing. The data generated from sensors are sometimes redundant and huge which consumes high energy for transmitting them to the base station. So, we must take this constraint into account during data collection and transmission in order to assure high quality of information, reduce energy consumption, and eliminate redundant data.

6. **Interpreting Data and Formation of Knowledge :** The main challenges for data interpretation and knowledge extraction are to apply or to develop new methods able to make prediction and to extract hidden knowledge from the sensed data without being influenced by noisy or missing data, and outliers. The wrong interpretation of data can easily lead users not to understand the system. Therefore, it is necessary to develop methods that can convert the sensed data into usable knowledge [24].

7. **Distributed Management :** The large-scale and resource constraints of WSNs cause various difficulties for centralized algorithms, which are implemented at the base station to assure network management tasks such as routing or topology. If the base station fails, then the entire network will collapse. However, the reliability of the sensor nodes can be increased by using a distributed management. In distributed control, sensor nodes must collaborate with their neighbors to process the data and take decisions locally by themselves without any global knowledge. This method will not necessary lead to optimal algorithms [23], but in comparison to centralized methods, they may be more energy-efficient, lead to better collection of data, assure a higher self-organization.

8. **Heterogeneity :** There are two types of WSN networks, homogeneous and heterogeneous [21]. In homogeneous networks, all nodes have the same capacity in terms of battery, storage, and processing unit. In heterogeneous networks, the nodes are not identical and do not have the same capability i.e., some nodes have more energy than others. The advantage here is an extension of the network lifetime which, beyond, can be increased significantly by forming cluster-heads which have the highest energy within their groups and which collect data from less powerful nodes. Consequently, a WSN should be of heterogeneous nature, for routing, clustering, localization, and target tracking in the aim of maximizing the lifetime of the network.

9. **Mobility :** The nodes in a WSN can move from one location to another [25]. This movement of nodes gives the ability to sense, compute, and communicate like static nodes and to help in self-organization, data collection, and energy efficiency within the network. The mobility is addressed in some WSN applications such as localization of the nodes. Then, the proposed solutions should be adaptive to dynamic networks with appropriate algorithms which take into account the nodes leaving and joining the network.

10. **Scalability :** The increase in the number of sensor nodes in the network causes many changes in node density, network size and topology. If the number of sensor nodes deployed in the sensing area is of important order, the challenge is to keep the network functioning appropriately.

2.3 Data mining

In recent years, Data mining has attracted a great attention in processing of data, due to the wide availability of enormous amounts of data, the imminent need to develop powerful means for analysis, and the interpretation of such data in order to extract interesting knowledge that could help in decision making. This knowledge can be used for many applications such as market analysis, fraud detection, production control and science exploration. Data mining, as defined by [26,27], is a non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. It solves problems by analysing data that already exists in databases and discovering hidden patterns and information. Data mining is also defined as a process of exploring large datasets in order to bring out knowledge, significant information and to uncover useful patterns from databases [28]. Data mining is a synonym for another popularly used term, Knowledge Discovery from Data (KDD). Alternatively, others view Data mining as simply an essential step in the process of KDD. Figure 2.4, shows a KDD process.

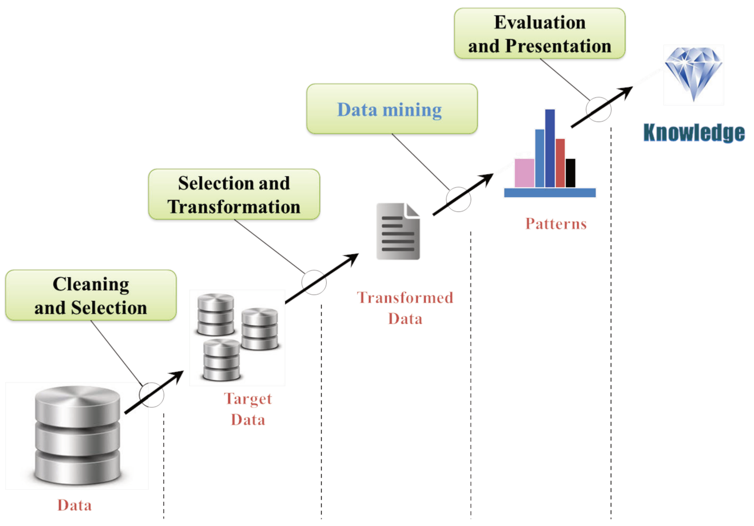


Figure 2.4: KDD (Knowledge Discovery in Data).

The KDD process consists of an iterative sequence of the following steps [28]:

- *Step 1:* **Identifying the goals :** this step develops an understanding of the application domain and the relevant prior knowledge and identifies the goal of the user's viewpoint.

- *Step 2:* **Creating a target data :** consists in selecting a dataset by focusing on a subset of variables or data samples, on which discovery is performed.

- *Step 3:* **Data cleaning :** represents basic operations which include removing noise or inconsistent data, data reduction or handling missing data.

- *Step 4:* **Data integration :** is used where multiple data sources may be combined.

- *Step 5:* **Data selection :** is used to retrieve data relevant to the analysis task from the database.

- *Step 6:* **Data transformation :** is used to transform or consolidate data into appropriate forms by performing summary or aggregation operations.

- *Step 7:* **Data mining :** is an essential process where intelligent methods are applied in order to extract data patterns from datasets, including classification rules or trees, regression, and clustering. The user can significantly help the Data mining techniques by correctly performing the preceding steps.

- *Step 8:* **Pattern evaluation :** allows to identify the truly interesting patterns representing knowledge based on some interesting measures. This step can also involve visualization of the extracted models.

- *Step 9:* **Knowledge presentation :** are visualization and knowledge representation techniques which are used to present the mined knowledge to the users. This knowledge is used to understand a system and to predict the actions, or simply for documenting and reporting a system to interested parties. This process also includes checking for resolving potential conflicts with previously obtained (or extracted) knowledge.

The Data mining step presents the interesting patterns to the user which may be stored as new knowledge. Note that according to this view, Data mining is only one step in the entire process, albeit an essential one because it uncovers hidden patterns for evaluation.

In the following, we discuss the general Data mining techniques and functionalities, with a focus on the Data mining techniques related to our area of research.

2.3.1 Techniques of Data mining

There are many different methods used to perform Data mining tasks [29]. These techniques involve different algorithms to accomplish these different tasks. All these algorithms attempt to fit a model to the data. The algorithms examine the data and determine a model that is closest to the characteristics of the data being examined. The techniques used in Data mining are categorized into two classes [30]. Figure 2.5 shows a general classification of Data mining methods.

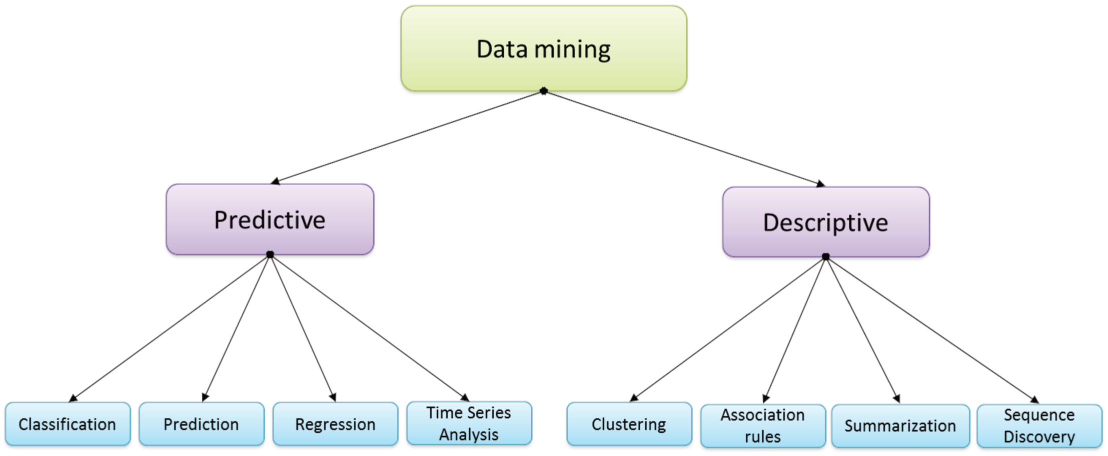


Figure 2.5: Classification of Data mining methods.

From Figure 2.5 it is clear that techniques fall into one of two categories, predictive or descriptive :

1. **Predictive algorithms :** The objective of these algorithms is to construct a model based on a set of input and output observations, in order to predict the value of a new variable from the known values of other variables. In other words, these algorithms look at historical data to predict the future.

2. **Descriptive algorithms :** The purpose of these techniques is to characterize the general properties of given data and to find models. In other words, they describe the data concisely and present their interesting properties. These algorithms make an inference on current data in order to make predictions.

In the following, we will present some Data mining techniques related to our work.

2.3.2 Classification and Prediction

Classification and prediction are two forms of data analysis that can be used to extract models describing important data classes or to predict future data trends which can provide a better understanding of the large data.

Classification is defined as the process of discovering a model that describes and distinguishes the data classes [31]. The classification uses given class labels to order the objects in the data collection. This approach makes use of a training dataset, in which all objects are already associated with known class labels, to build a model which can be used to predict the classes of future data instances which have unknown class labels. A variety of data classification techniques are developed in the literature such as Decision Tree (DT), Naïve Bayesian (NB), Artificial Neural Network (ANN), Support Vector Machine (SVM), K-Nearest Neighbor (KNN), ..., etc. [32].

Prediction can be viewed as a type of classification which can explain one thing based on the relationship between other things. The difference is that prediction means predicting a future state rather than a present state. In addition, predicted values are usually continuous whereas classifications are discrete [28]. Among the prediction techniques, we can cite the regression method [28].

2.3.3 Clustering

Similar to classification, clustering is the organization of data in classes. However, unlike classification, in clustering, class labels are unknown and it is up to the clustering algorithm to discover acceptable classes. Clustering is also called unsupervised classification, because the classification is not dictated by given class labels. The main aim of the clustering methods is to group the data objects according to some measure of similarity, so that the data objects in one group or cluster are highly similar to one another while being very different from the data objects belonging to the remaining clusters. In other words, the aim is to minimize the intra-class distance and maximize the inter-class distance [28] as shown in Figure 2.6. The similarity or dissimilarity between data objects based on distance is mesured by using a formula such as Euclidean, Manhattan, and Minkowski [29].

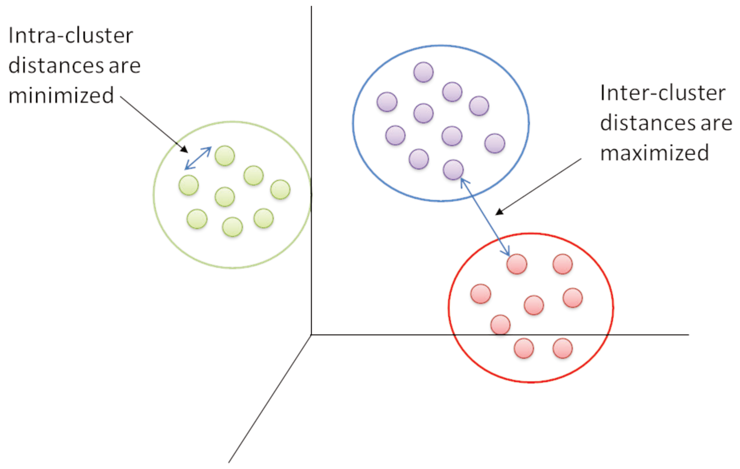

Intra-cluster distances are minimized
Inter-cluster distances are maximized

Figure 2.6: Principle of Clustering.

Generally, clustering methods are classified into different categories [30] :

- Partitioning methods : this type of clustering allows to construct various clusters and then evaluates them by some criterion by minimizing a similarity function based on distance or the sum of square errors. The typical methods of this category are : k-means, k-medoids, and CLARANS [33].

- Hierarchical methods : a hierarchical clustering algorithm [34] creates a decomposition of the set of data (or objects) into a tree of clusters, also known as a dendrogram by an agglomerative (bottom-up) or divisive (top-down) approach. An agglomerative clustering starts with one-point clusters and recursively merges two or more most appropriate clusters. A divisive clustering starts with one cluster of all data points and recursively splits the most appropriate cluster. The process continues until a stopping criterion is achieved. The typical methods of this category are : DIANA, AGNES, BIRCH, ROCK, and CAMELEON [28].

- Density-based methods : this type of clustering algorithms is based on connectivity of objects and density functions. The typical methods of this category are : DBSACN, OPTICS, and DenClue [35].

2.3.4 Association rules

Frequent patterns are patterns that occur frequently in datasets [28]. There are many types of frequent patterns, including itemsets, subsequences, and substructures. A frequent itemset typically refers to a set of items that frequently appear together in a transactional dataset. The aim of mining frequent patterns is to discover interesting associations and correlations within data. Association rules study the frequency of items occurring together in transactional datasets based on a threshold called *support*, and identify the frequent itemsets. Another threshold called *confidence* is used to identify the accuracy of association rules, which represents the conditional probability that an item appears in a transaction when another item appears. The association rules have to satisfy the minimum support and the minimum confidence. Many techniques for frequent itemset mining are proposed in the literature which are divided into three major approaches : Apriori algorithm [36], frequent-pattern growth [37], and vertical data format [38].

2.4 Data mining techniques for WSNs

WSNs are an emerging area of research applied in many application domains. These applications require real-time monitoring and generate huge volumes of dynamic, geographically distributed and heterogeneous data. The collected data can be of potentially large quantity, redundant, of high dimensionality, and of distributed nature. For these reasons, Data mining algorithms play an important role for data management and decision making. Therefore, these data must be analyzed and transformed to usable information efficiently. In this section, we would like to survey some previous works related to this topic. We select some of them to introduce and categorize them by Data mining tasks that are applied in different applications of WSN to mine sensed data. We note that these categories do not cover all the aspects in WSNs, but they are fundamental to all other works in the field. In addition, we also present the challenges and the limitations of Data mining techniques applied to WSNs.

2.4.1 Challenges of Data mining in WSNs

Sensor nodes generate high speed arriving data that need to be processed in real time by Data mining algorithms. Due to the nature of sensor data and their specific characteristics, conventional Data mining techniques are not directly applicable to WSNs. Table 2.1 shows the global differences between traditional and WSN data processing [1]. As we can see from Table 2.1, traditional Data mining techniques are highly computational, executed in centralized nature and designed for analyzing static datasets. The data is collected at the central computer which has no resource constraints. In comparison with the central computer, sensor nodes have limited storage and processing capabilities, and it is impossible to store the entire WSN data or to run highly complex algorithms. These characteristics of collected data and the special design of sensor nodes make classical Data mining techniques challenging. Therefore, it is necessary to develop Data mining techniques which can process and analyze sensed data in multi-level, locally in nodes, and on-line manner. In the following, we discuss the different challenges addressing the development of Data mining techniques for WSNs.

Table 2.1: Difference between traditional and sensed data processing.

Characteristics	**Traditional data**	**WSN data**
Processing architecture	Centralized	Distributed Data
Data type	Static	Dynamic
Memory usage	Unlimited	Restricted
Processing time	Unlimited	Restricted
Computational power	High	Weak
Energy	No constraints	Limited
Data flow	Stationary	Continuous
Data length	Bounded	Unbounded
Response time	Non-real-time	Real time
Update speed	Low	High

1. **Resource Constraint :** The sensor nodes have low resources in terms of power, memory, communication bandwidth, and computational capability. The main challenge of Data mining techniques for WSNs is to satisfy the mining accuracy requirements with a minimum resource consumption of sensor nodes.

- Energy efficiency : Sensor nodes consume a huge percentage of their energy by sending and/or receiving datasets [39]. Therefore, a local processing of sensed data or the use of efficient data aggregation techniques are crucial to assure the operability of the WSN. The energy issue has been the main motivating research challenge in data processing in WSNs where most of the proposed techniques in the literature have addressed this issue.
- Communication efficiency : The exchange of data is limited by the lack of bandwidth or unreliable connection. Then an appropriate data aggregation technique is important to reduce the communication overhead.
- Processing power : Sensor nodes have limited processing capacity to perform large computational tasks. Therefore, the selected Data mining techniques must be of low complexity according to the processing capability of a sensor node. We note that most of the research studies done in this area have used simulation techniques due to the limitations of the documentation of processors of sensor nodes.
- Memory size : As mentioned above, the traditional Data mining techniques process data which are stored in a large memory. However, limited availability of memory on sensor nodes makes these techniques unsuitable for WSNs, and we have to find the best processing techniques to solve this issue.

2. **Huge and fast collected data :** In many WSN applications, data arrive with higher speed than we are able to process them. In addition, the spatial and temporal nature of sensor data play an important role in these applications. These reasons cause many challenges for the Data mining techniques to process sensed data. Therefore, we must use Data mining techniques which can cope with continuous, rapid, and changing data streams.

3. **On-line Data mining techniques :** A WSN is deployed through thousands of nodes in an environment. Then data are geographically distributed, arrive continuously, and are able to be scale. Most of the Data mining techniques analyze data in an off-line way which is not compatible with the requirement of handling distributed data streams. Therefore, the challenge is how to process distributed data streams in an on-line way.

4. **Data delivery latency :** Real-time processing and delivery are very important in most WSN applications. Therefore, reducing the latency of response is very important. The sensed data or alerts generated by the sensor nodes have to be delivered to the end-user with a smallest possible latency. The major challenge of Data mining techniques for sensor networks is to guarantee a small latency by taking into account the presence of constraints and limitations of connectivity. The processed data need to be delivered via a multi-hop communication with latency requirements, but simultaneously in the aim to maximize the network lifetime.

5. **Dynamic Network Topology :** WSNs are able to have a dynamic network topology in potentially harsh, uncertain, heterogenic, and dynamic environments. The topology is variable because sensor nodes may move among different locations over time which increases the complexity of designing appropriate Data mining techniques for WSNs.

   To deal with these challenges, researchers have modified the classical Data mining techniques and also proposed new Data mining algorithms to handle the data generated from sensor nodes. In the following section, we will provide a general view of these techniques.

2.4.2 Taxonomy of Data mining techniques for WSNs

In this section, different classification schemes for existing approaches designed for Data mining techniques in WSNs are presented. The first classification is given by [1]. As shown in Figure 2.7, the highest level classification is based on the general Data mining classes used such as frequent pattern mining, sequential pattern mining, clustering, and classification. The second level of classification is based on the ability of each approach to process data in a centralized or distributed way. The third level of classification is selected according to two separate aspects of issues, namely, WSN performance issues and WSN application issues. As sensor nodes are limited in resources, algorithms are needed which maximize performance of WSNs. On the other hand, WSN applications require data precision, fault tolerance, event prediction, scalability, and an efficient use of energy, communication, and processing.

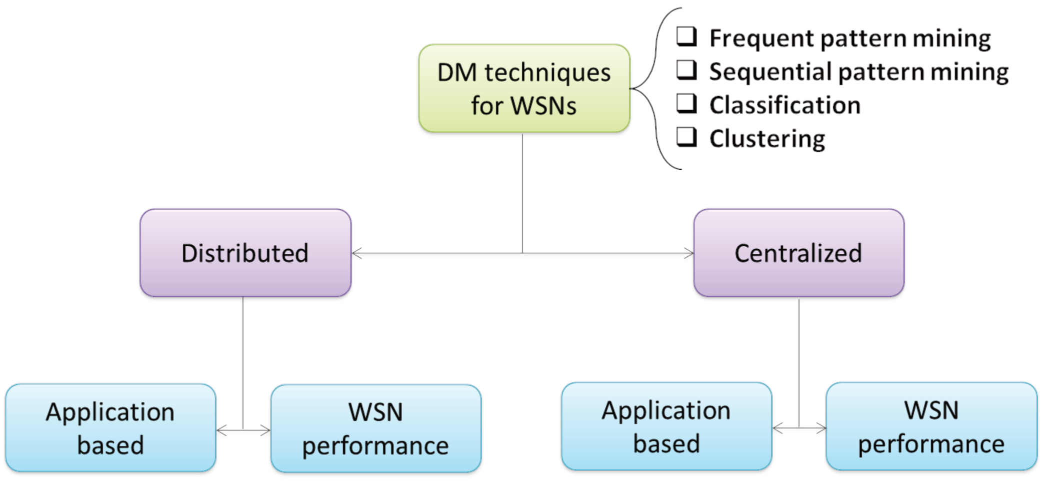


Figure 2.7: Classification tree presented in [1].

The second classification consists of the distributed Data mining techniques which are given in [2]. As shown in Figure 2.8, the highest level of classification follows the Data mining classes as follows : Clustering, Association Rule Mining, Classification, and Regression. This classification is justified by [2], as the algorithms belonging to one of the defined classes have very different roles, which has a major impact on their design and principles of operation. On the other hand, when selecting an algorithm with an application in mind, it has to define its role in terms of one of these classes, from which point the rest remains of little interest. For these reasons, this division was selected as the primary, top-level classification. The second level classification is based on the ability of the methods to handle node mobility in the network. WSNs can be of static or mobile nature. For both natures, Data mining algorithms must help the sensor nodes in addressing the problems by themselves. The third level of classification is selected according to the two issues energy awareness and performance of a WSN system. The first is energy which should be conserved as much as possible. This energy is consumed on radio transmission, algorithm running which needs to be as short as possible so as to allow nodes to sleep for longer periods of time, reducing redundancies,..., etc. The second issue is the performance of a WSN system which is presented by precision and accuracy, fault tolerance and robustness that requires efficient solutions for communication and redundancies. This leads to find a compromise between the system's performance and energy consumption to ensure a longer network lifetime with the best performance. For these reasons, division energy or performance oriented approaches have been selected as the lowest level classification criteria.

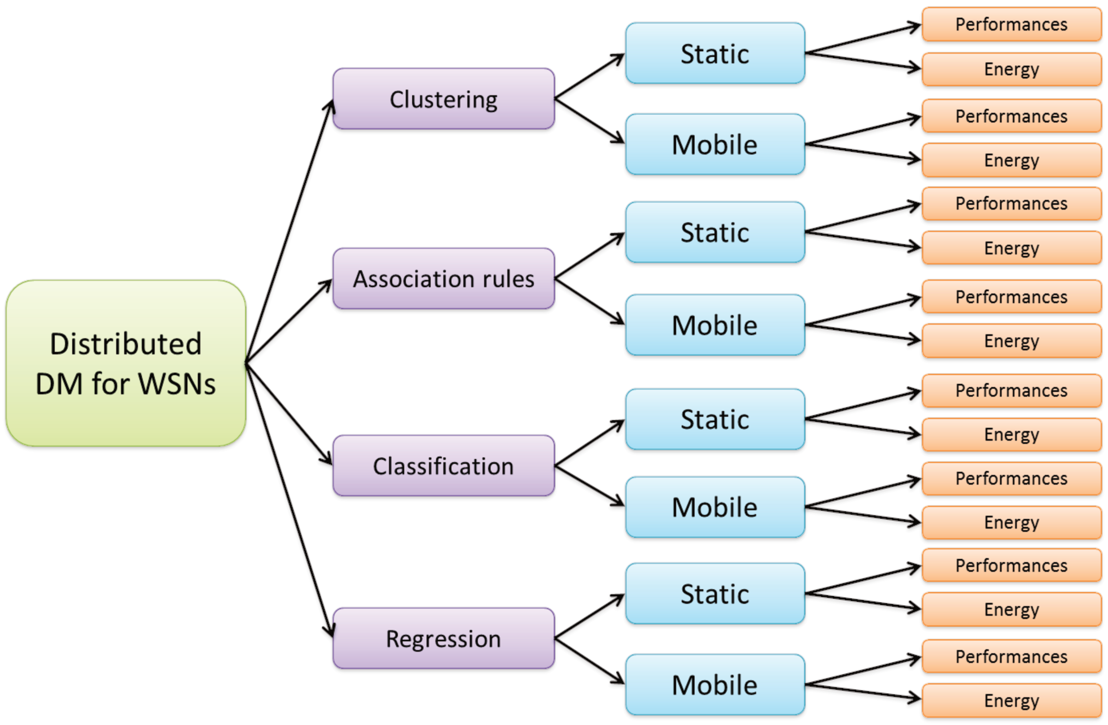


Figure 2.8: Classification tree for the Distributed Data mining approaches developed for use in WSNs. [2].

The third classification is presented in [2] which consists in classifying the Data mining algorithms following the criteria shown in Table 2.2. This classification starts from the major four algorithm types for Data mining in WSNs, with a subsequent special emphasis on optimization criteria such as the used algorithm type, mobility type, and the attitude toward energy awareness. Therefore, three criteria are applied, and 16 classes of approaches are obtained. According to the

Table 2.2: Classification criteria

Criteria	Type
Algorithm type	Classification
	Clustering
	Regression
	Association rule mining
Mobility type	Mobile
	Static
Attitude towards energy awareness	Approaches characterized by energy awareness and efficiency awareness alone (EE)
	Approaches characterized by multi -parameter efficiency optimizations, overall optimization (OO)

second and the third criterion, these can be organized into the four basic groups Mobile Energy Efficient (MEE), Static Energy Efficient (SEE), Mobile Overall Optimized (MOO), and Static Overall Optimized (SOO), as presented in Figure 2.9.

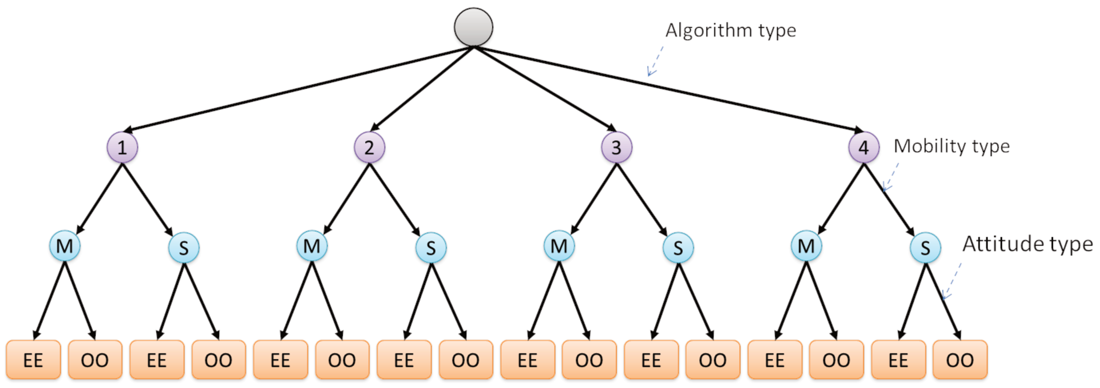


Figure 2.9: Classification tree presented in [2].

In our thesis, we focus on distributed approaches in order to overcome the problems of the limited resource constraints, and it is easy to see that the standard Data mining algorithms are not directly applicable to a WSN setting. Therefore, several researchers have proposed various algorithms for WSNs which take into account some or all of these constraints. One of the main challenge to consider in WSNs is to assure a distributed and local processing, and to reduce the sensor communication for aggregating the whole data or the results when sent to the base station. In this context, the distributed Data mining techniques can likely play a major role. The major objective of such distributed approaches is to develop algorithms which can perform some local computation on its own data, limit the messages and communication energy of sensor nodes with nearby nodes while transferring data to the base station to compute a global model. In more detail, Data mining techniques have to be used off-line with distributed and on-line data processing, real-time analysis for decision making, knowledge extraction, and adaptation to the dynamic nature of WSNs data. Also, we must split the data processing into smaller parts executed at multiple levels and in parallel to deal with large-scale data from WSNs. We must also define a role for each node based on its capacities and reduce the size of the data to be transmitted in order to maximize the lifetime of the network. Consequently, we can help to improve the WSN lifetime, to deal with large amounts of data obtained from WSNs, and to extract a maximum information and knowledge from the environment. However, in centralized approaches, a WSN generates a huge amount of data, and communication can create a loss of communication bandwidth and bottlenecks. In our research, we take into account the data processing in terms of processing time, energy consumption, storage and communication and their impact on the choice of Data mining algorithms. In addition to all criteria of WSNs, we take into account the nature and the type of WSN applications in order to decide on the architecture for choosing the Data mining algorithm.

In the following, we present a sampling of two important topics of distributed Data mining algorithms for WSNs which are presented on data and node clustering and data classification.

Classification

Classification is one of the most important tasks in WSNs. The objective of this technique is to develop energy efficient distributed classification algorithms for large-scale WSNs. These techniques are able to train classifiers which are a good optimization in decision-making processes and transmit the results to the base station. This greatly reduces the communication and improves energy efficiency.

However, we must choose the best classification or prediction accuracy, and the most appropriate for a particular WSN application, and cope to large quantities of sensed data. This kind of analysis provides an opportunity for Data mining researchers to develop more advanced methods for handling some of the specific issues related to sensed data. Table 2.3 shows and discusses already existing distributed classification algorithms for WSNs.

Table 2.3: Literature on classification techniques in WSNs.

Name	Techniques Used	Objective in WSNs	Issues
HDT [40]	Decision tree	Aims to construct a spanning tree, encompasses all the nodes in the system.	Offers better accuracy and energy consumption. Requires synchronization in every time step which is expensive to deploy large WSNs.
IDS [41]	K-nearest neighbor	Intrusion detection system	High detection rate and speed. But has also a high false positive rate.
Online learning [42]	SVM (support vector machine)	Incremental classification	Energy consumption decreases when the SVM is trained incrementally. But its computational complexity is very high.
EEA [43]	Neural network (ANN)	Detection mechanism of energy exhaustion attacks	Increases the lifetime of sensor nodes.
BN-LEACH [44]	Realizes Bayesian network model	Realized uniform distribution of cluster heads that is not guaranteed in LEACH to extend the network lifetime.	Increases the lifetime of sensor nodes.

Clustering

In the literature, clustering algorithms are addressed by node clustering or data clustering. Node clustering consists in creating efficient communication topologies by organizing sensor nodes into clusters. This method is required for large number of sensor nodes to resolve the energy constraint and to aggregate collected data. Clustering also consists in aggregating data in order to summarize the overall transmitted data and to find data correlations among the nodes. Table 2.4 shows and discusses already existing distributed clustering algorithms for managing data, lowering the communication overhead, enabling improved traffic control, and improving energy efficiency and network stability.

Table 2.4: Literature on clustering techniques in WSNs.

Name	Techniques Used	Objective in WSNs	Issues
LABDC [45]	Location Aware Based Data Clustering algorithm.	Reduces the number of transmissions as well as the transmission cost.	Efficient in highly correlated sensor data.
DVKM [46]	Distributed version of the K-Means algorithm.	Transmission time and power consumption of sensor nodes reduces.	Requires extra memory for cluster head and high computation power. Not efficient for high sensor areas.
CAG [47]	Distributed correlation based Clustering.	Suggests generating clusters of nodes sensing similar values within the threshold.	Spatial and temporal correlation is used for grouping the data. Significant resource savings. But most sensory readings are never reported (lower precision).
KMDR [48]	K-Means Data Relay clustering algorithm.	Reduces the communication overhead and increases the network performance.	Works well in homogeneous networks.
LEACH [49]	Clustering Hierarchy nodes for aggregating data.	Generates clusters based on the size of the sensor network.	A drawback in this protocol is the prior knowledge of the topology of the network.
CODA [50]	Cluster based self organizing. Data Aggregation.	Aims to aggregate sensor data in clusters. The nodes are trained to have the ability to classify the sensor data.	Increases the quality of data and reduces data traffic, and is energy-conserving.

2.5 Conclusion

In this chapter, we have introduced the background of the Data mining techniques in the context of WSNs. First, we have presented the architecture of WSNs. Second, the hardware platforms, operating systems, wireless advantages, challenges and applications of the WSN are described. We have also described some strategies and constraints to take into account for energy conservation which can make an efficient WSN. Following the WSN description, we have the reviewed Data mining techniques, specifically those relevant to the WSN field. Finally, we have analyzed the different classifications of existing Data mining methods with a focus on distributed techniques based on the context of WSN. In the next chapter, a two new solutions are proposed. The first one allows to detect boundary points in datasets. The second one consist of distributed algorithm which allows to detect boundary nodes in the WSNs.

CHAPTER

THREE

BOUNDARY DETECTION ALGORITHMS

3.1 Introduction

The improvement of information technology has led to the continuous, rapid and huge collection of data. Knowledge discovery is a relevant process to identify valid, interesting and potentially valuable patterns in data [51]. The urgent need for efficient analysis tools to discover information from these data led to develop many Data mining techniques such as data classification, association rule, and clustering, as well as data cleaning and preparation techniques to improve the quality of the data by removing anomalies and noises. In this chapter, we examine the problem of boundary points detection. Boundary points are data points that are located at the margin of a group of distributed data. This situation is modeled as the problem of finding a polygon hull in a connected Euclidean graph, which represents a set of connected vertices that envelops the whole data points. Boundary points are useful in Data mining applied for WSN applications, since they can be used to represent a subset of sensed data that have possibly overlapping classes, to detect shape of the extracting clusters in a set of sensed data, to detect anomalies, to make classification, and to reduce the transmitted data which extends the lifetime of the WSNs. The knowledge of these points is also useful to resolve real problematics of many applications which can be modeled as a connected Euclidean graph, like detecting boundaries of WSNs, drawing contours in medical and biological images, localizing of interest event or region,..., etc. [52–56]

This chapter is organized into two sections. The first section is destinated to present an efficient algorithm called *Least Polar-angle Connected Node (LPCN)* [57] allowing to find the boundary vertices of a connected Euclidean graph. This algorithm will be used in our next contribution in order to reduce huge transmitted data which are generated by WSNs (cf. 4). The second section is devoted to present a new distributed algorithm called *Distributed Least Polar-angle Connected Node (D-LPCN)* [58] which aims to find the boundary of a WSN in order to help decision making with respect to low complexity and low energy consumption. In addition, we will review related work and present the simulation results of both algorithms.

3.2 LPCN

Finding the polygon hull (contour) in a connected Euclidean graph can be considered as the problem of finding a concave hull, with the exception that at any iteration a vertex can be chosen only if it is connected to the vertex chosen at the previous iteration. One of the methods that can be used for this kind of problems is Jarvis' algorithm [59] which allows to find the convex hull and which must be adapted because it does not take into account the connections of the vertices (data points). In this work, we propose a new algorithm allowing to find the boundary vertices of a connected Euclidean graph, where we try to find a set of vertices allowing to represent the geometric shape of the graph in the form of a polygon hull, i.e., a simple polygon formed by edges of the graph such that all vertices of the graph are either on the polygon or surrounded by it. More precisely, we are looking for a closed cycle of minimum length in the graph such that all vertices are either on or surrounded by that cycle.

3.2.1 Related work

In the literature, there are many useful algorithms allowing to find either a convex or a polygon (concave) hull for a given set of points in the plane. Table 3.1 summarizes the major approaches found in the literature regarding convex or concave hull determination algorithms, also, in terms of time complexity and dimensionality, where n is the total number of points, k the maximum degree of a point, h the number of points on the convex hull, d the dimension of the considered space, and where r is a number that depends on the dimension d of the considered space, which is equal to $\frac{d}{2}$ for a 3-dimensional space.

Table 3.1: Comparison with existing algorithms.

Algorithm	Convex hull	Concave hull	Complexity	Dimension	Observations
Graham [60]	✓		$O(n \log n)$	$2D$	
Jarvis [59]	✓		$O(nh)$	$2D$	
Quick-hull [61]	✓		$O(n^2)$	$2D$	
Incremental [62]	✓		$O(n^{(d+1)/2})$	Multidimensional	
TORCH [63]	✓		$O(n \log n)$	$2D$	
NICP [64]	✓		$O(n + n \log n)$	$2D$	Requires the Quick-hull algorithm.
Mei [65]	✓		$O(n \log n)$	$2D$	GPU-accelerated convex hull algorithm.
Ruano et al. [66]	✓		[not given]	Multidimensional	Time complexity depends on many parameters.
S-CH [67]	✓		$O(n \log n)$	$3D$	Requires a standard convex hull algorithm.
Split and Merge [68]		✓	$O(nh)$	$2D$	Starts from the convex hull of points.
Alpha-Shape [69]		✓	$O(n \log n)$	$2D$	Depends on a parameter α.
PBE [70]		✓	$O(n)$	$2D$	
KNN [71]		✓	$O(nh^2)$	$2D$	
CM [72]		✓	$O(n \log n + rn)$	Multidimensional	
Braune et al. [73]		✓	$O(n \log h)$	$2D$	Starts from the convex hull of points.
Gheibi et al. [74]		✓	$O(n \log n)$	$2D$	Starts from the convex hull of points.
EC-Shape [75]		✓	$O(n \log n)$	$2D$	Requires Delaunay Triangulation (DT)
Gift Opening [76]		✓	$O(n)$	$2D$	Starts from the convex hull of points.
RGH [77]		✓	$O(n^3)$	$2D$	
LPCN		✓	$O(kh^2)$	$2D$	Requires only a starting point.

3.2.2 LPCN algorithm

In this section, we will present the proposed LPCN algorithm. The problem of the boundary detection can be formulated as follows. Given an undirected graph $G = (V, E)$, where $V = \{v_0, v_1, ..., v_{n-1}\}$ is the set of vertices of G and $E = \{e_0, ..., e_{m-1}\}$ its set of edges, the objective is to determine a smallest closed polygon enveloping all the vertices of G. In other words, the goal is to find a finite and minimum set of vertices, able to contain all other vertices within the envelope formed by them. This envelope is called a polygon hull which is defined by two sets: a set of vertices $\mathbb{B}_V$ of G and a set of edges $\mathbb{B}_E$ of G that can be given as follows:

$$\mathbb{B}_V = \{v_0, v_1, ..., v_{i-1}, v_i, v_{i+1}, ..., v_h\} \subseteq V,$$

$$\mathbb{B}_E = \{\{v_0, v_1\}, \{v_1, v_2\}, \cdots, \{v_{h-1}, v_h\}\} \subseteq E$$

with $v_0 = v_h$.

The LPCN algorithm allows to find a polygon hull (or a boundary) of a given Euclidean graph. The main advantage of this algorithm is the way it defines the boundary. In addition, our algorithm differs from Jarvis' algorithm in the selection of that point which follows the current one. Its main idea is given as follows. The algorithm starts from a vertex $v = v_0$ with minimum x-coordinate, and determines the vertices v_1, ..., v_{h-1} of the polygon hull. In each iteration i of the algorithm, the next vertex v_{i+1} in the polygon hull is determined by the edge $\{v_i, v_{i+1}\}$ that has the minimum angle $\varphi min(v_{i-1}, v_i, v_{i+1})$ with the edge $\{v_i, v_{i-1}\}$ (cf. Figure 3.1), where v_i is the vertex of the polygon hull selected in the current iteration i and v_{i-1} is the vertex of the polygon hull found in the previous iteration $i - 1$. This algorithm covers some problematic situations whose treatment is illustrated

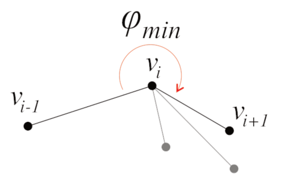


Figure 3.1: Angle formed by three vertices.

in our paper [57]. One of these situations is given by a pair of intersecting edges in the graph G. By definition of a polygon hull, no two edges of such a hull should intersect. Whenever such a crossing is detected by the algorithm, it will be eliminated as follows:

Intersecting edges: we require that two edges of the polygon hull must not intersect outside the endpoint corresponding to the next vertex v_{i+1} chosen by the algorithm. As an example, Figure 3.2(a) shows that accepting the intersecting edges $\{A, B\}$ and $\{E, F\}$ of the polygon hull will lead to non-visited vertices on the polygon hull, which are I , J and G. However, if this intersection is considered then, as shown by Figure 3.2(b), all the vertices of the polygon hull are visited (i.e., A, B, C, D, E, G, H, I, J). In addition, since the next chosen point is B, this intersection must not consider the endpoint B of the edge $\{D, B\}$. That is to say, if we have two edges $\{A, B\}$ and $\{C, D\}$ and if the intersection with B results in one of the vertices A, B, C or D then it will not be considered as an intersection. This situation is justified by Figures 3.2(c) and (d). In the first one, if we accept the normal intersection $\{B, C\} \cap \{D, B\} = \{B\}$, the algorithm will choose a vertex which is different from B. Then, it will choose the next vertex C, which will lead to an infinite loop B, C and D. However, if we consider intersection without the endpoints of the edges, then when the algorithm comes back to B, no intersection is detected and the algorithm will choose vertex A as shown by Figure 3.2(d).

The pseudo-code of the LPCN algorithm is given by Algorithm 1. First, it starts from a vertex having the minimum x-coordinate which is always part of the solution (lines 2 and 3). In each iteration of the repeat loop section that starts from line 7, the algorithm will search for the next vertex v_{min} of the polygon hull that will follow the currently found vertex v_c of the current polygon hull. This is done by calculating the angles formed by the edge $\{v_c, v_p\}$ and all edges $\{\{v_c, v\}/v \in \mathbb{A}\}$, where v_p is the previously found vertex and $\mathbb{A}$ is the set of neighbors of $v \in \mathbb{A}$ of v_c which verify the condition that each edge $\{v_c, v\}$ with v element of A does not intersect the currently found polygon hull.

3.2.3 Simulation results

Let us consider a graph with n vertices, and a boundary of h vertices, and let k denote the maximum degree of G. Then the complexity of the LPCN algorithm is $O(kh^2)$, since in each iteration we are searching for an intersection with the edges of the polygon found in previous iterations, i.e., we have to include a factor of h.

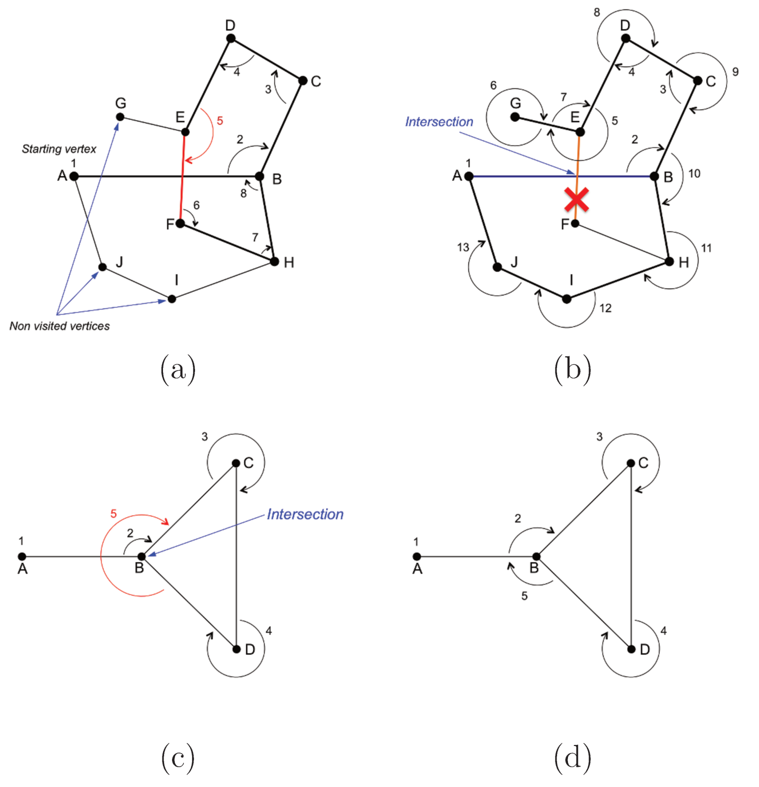


Figure 3.2: Edges' intersection.

Algorithm 1 LPCN Algorithm

```
 1: procedure LPCN($(V, E)$)
 2:     $v_c \leftarrow$ vertex having the minimum x-coordinate
 3:     $\mathbb{B}_V \leftarrow [v_c]$
 4:     $\mathbb{B}_E \leftarrow \emptyset$
 5:     $v_{first} \leftarrow v_c$
 6:     $v_p \leftarrow$ fictive vertex situated on the left of $v_c$
 7:     repeat
 8:         $\mathbb{A} \leftarrow \{v \in N(v_c) / \mathbb{B}_{\mathbb{E}} \cap \{v_c, v\} = \emptyset\}$
 9:         $v_{min} \leftarrow \underset{v \in \mathbb{A}}{\operatorname{argmin}} \{\varphi(v_p, v_c, v)\}$
10:         $\mathbb{B}_V \leftarrow [\mathbb{B}_V, v_{min}]$
11:         $\mathbb{B}_E \leftarrow \mathbb{B}_E \cup \{v_c, v_{min}\}$
12:         $v_p \leftarrow v_c$
13:         $v_c \leftarrow v_{min}$
14:     until $v_{min} = v_{first}$
15:     return $\mathbb{B}_V, \mathbb{B}_E$
16: end procedure
```

The algorithm has been programmed in Java and implemented in a software simulator *CupCarbon* [78]. This software offers an API that facilitates the development of algorithms and the visualization of their results. It offers the possibility to simulate mobiles and targets.

In the following, we will discuss two test examples for the proposed algorithm LPCN. The first example represents a random network with 100 points (cf. Figure 3.3(a)). The points are located in a plan of 1200×600 with the same distances between them. In the second example, we have 1500 points (cf. Figure 3.4) that are located in a plan of 2500×1500. The number of the obtained points on the boundary is 62 in the first example, and 85 in the second one. The maximum degree is 17 in the first example, and 50 in the second one. The theoretical complexity in each example is, respectively, $O(17 \times 62^2) = O(65k)$, and $O(50 \times 85^2) = O(361k)$. The real number of iterations for each example is, respectively, $9k$, and $53k$.

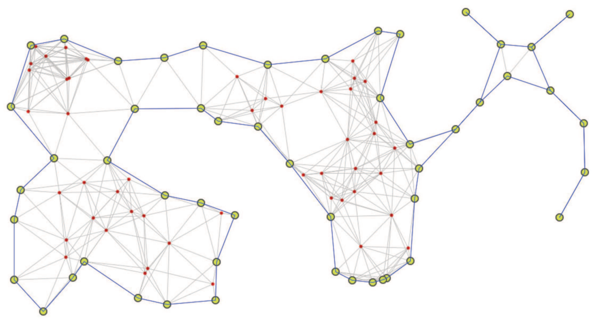


Figure 3.3: Example to execute LPCN with 100 points.

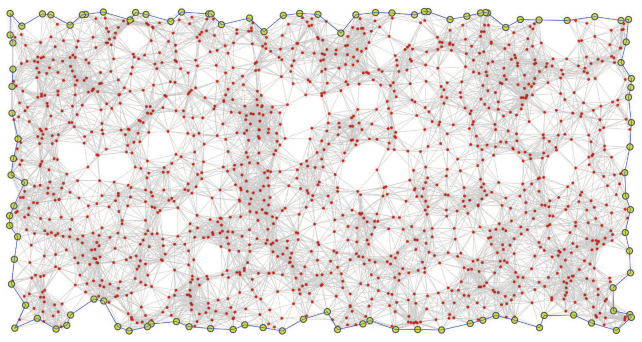


Figure 3.4: Example to execute LPCN with 1500 points.

3.3 D-LPCN

In this section, we present a new distributed algorithm called D-LPCN which allows to find the boundary nodes of a WSN in order to make a network operating efficiently taking into account the limited resources and the need to exchange information with minimum communication. As energy consumption is a limiting factor for the lifetime of a sensor node, the communication between nodes has to be minimized. To overcome this limitation, we propose in this solution to activate the sensing units of the whole network, to activate the radio module of the network's boundary nodes to ensure the communication between them, and to keep the radio module of the other nodes asleep periodically (duty cycling) in order to preserve their energy. Indeed, if all sensor nodes of the inner region participate actively in communication, an important size of data transmitted between nodes leads to an important energy consumption, and thus to an important reduction of the network's lifetime.

The proposed algorithm is based on the localization information of the nodes that only need to communicate with their one-hop neighbor nodes. Many existing methods can be used for the location information. Traditionally, a GPS receiver is the main device used to localize sensor nodes [79], especially when high accuracy is needed. However, it is very energy consuming. Many other localization techniques have been presented in the literature, such as TOA, TDOA, RSSI, AOA, HIRLOC, SERLOC, and ADLA [80] which use the power of the received signals in order to determine the location. Other methods use hybrid sensor nodes where only some sensor nodes are equipped with GPS, called anchor nodes [81] or beacon nodes [79]. These nodes will be used to determine the location of other un-localized nodes. The accuracy of these methods depends on the number of anchor nodes.

3.3.1 Related Work

In order to find the boundary nodes of a wireless sensor network several methods have been developed as [4, 82–99]. They can be classified into three categories: geometrical, statistical, and topological. The geometrical methods [83, 85, 92] are the most accurate because they use the location information of each node to determine the boundary node. Statistical methods make assumptions on the probability distribution of the node deployment, and identify the boundary nodes based on statistical properties under certain network conditions. The topological methods [82, 84, 100] use the connectivity information to determine the boundary nodes [88]. These methods allow the exchange of connectivity information among neighbor nodes, and they detect the boundary nodes without any information on their location. Generally, they outperform the statistical methods [4, 88, 101].

We have observed that some of these algorithms work with planar networks or that they do not take into account the crossings between edges which can lead to blocking situations arising from specific subgraphs or to cycles producing an infinite loop. We have illustrated these situations as part of our work in [102] and shown how to avoid them. We have also observed that in general they do not give the optimal boundary in terms of the number of nodes. Table 3.2 summarizes 8 boundary determination algorithms in terms of message complexity (number of exchanged messages per node) and accuracy which represents the percentage of the real boundary nodes, where n is the number of nodes, n_b is the number of the boundary nodes, k is the number of nodes which execute simultaneously local computation, d is the maximum degree of the network, h the size of the table of the 3-hop neighbors and h_{max} the maximum hop-count of a node as given by the user. We added some observations on the communication overhead in the last column. We have compared our algorithm with only those for which the energy consumption is provided in their paper. Another comparison based on the energy consumption is presented in Section 3.3.3.

3.3.2 D-LPCN Algorithm

In this section, we will present the main steps of the proposed distributed algorithm for the discovery of the boundary nodes in a WSN. First, we introduce some definitions and primitives. Then, we present our Start-node algorithm allowing to launch the proposed D-LPCN algorithm, which will be presented thereafter.

Table 3.2: Comparison of D-LPCN with existing algorithms.

Algorithm	Message complexity	Accuracy	Observations
ABBD [89]	$O(n/k + n_b/k)$ = number of rounds $O(kd^3)$ = number of messages per round	$\simeq 88\%$	Communication overhead.
EBDG [99]	$O(d\ h_{max})$	$\simeq 92\%$	Requires a dense network.
LVP [83]	$O(n\ d)$	$\simeq 100\%$	Computation overhead.
PDiscovery [98]	$O(n\ d\ h)$	$\simeq 95\%$	Communication overhead.
DBD [92]	$O(n(d^3 + d^2 + d))$	$\simeq 90\%$	Requires 3-hop neighbors' information.
DBNS [91]	$O(n/k)$ = number of rounds $O(k\ d)$ = number of messages per round	$\simeq 100\%$	Required to remove redundant nodes to form a complete boundary.
Hop-based [97]	$O(n\ d\ h_{max})$	3% to 99%	Depends on the list of x-hop neighbors. Communication overhead.
D-LPCN	$O(d\ n_b + n)$	$= 100\%$	Communication and energy are reduced. Only the boundary nodes and their neighbors are used.

Definitions and Primitives

We assume that the communication between two sensor nodes is symmetrical. In this case, a WSN can be modeled as an undirected graph $G = (S, E)$, where $S = \{s_1, s_2, ..., s_n\}$ is the set of sensor nodes, $n =\mid S \mid$ their total number and E the set of communication links. The link between two nodes can be defined as follows:

$$e_{ij} = \begin{cases} 1 & if\ the\ node\ s_i\ communicates\ with\ s_j \\ 0 & otherwise \end{cases} \tag{3.1}$$

The neighbor nodes $N(s_i)$ of a given node s_i are the nodes that communicate with it.

$$N(s_i) = \{s_j/e_{ij} = 1,\ \ j = 1, ..., n\ \ and\ \ j \neq i\} \tag{3.2}$$

For a better understanding of the proposed algorithm, we define in Table 3.3 some message primitives and their definitions and in Table 3.4 the functions used in the algorithm. Note, that the proposed algorithm works in the case of bidirectional communication in order to obtain the optimal boundary nodes and every node knows its location in space in terms of (x, y) coordinates.

Table 3.3: Message primitives and their definitions

Primitive	Definition
AC	ask for coordinates
CS	send coordinates
SN	select a node

Table 3.4: Functions of the D-LPCN algorithm

Function	Definition
`getId()`	returns the node identifier
`getCoord()`	returns the node coordinates (x, y)
`getNumberOfNeighbors()`	returns the number of neighbors of the node
`send(`a`,` b`)`	sends the message a to the sensor node having the identifier b, or in a broadcast $(b = *)$
`read()`	waiting for receipt of messages

Determining the Starting Node

In this section, we will present a distributed algorithm that allows to determine a node with minimum x-coordinate in a network. This algorithm is based on the Minimum Finding algorithm presented in [103, 104] which relies on the tree-based broadcast algorithm. The aim of this algorithm is to determine the starting node of the D-LPCN algorithm which will be presented in Section 3.3.2. The principle of the Start-node algorithm can be described as follows. First, each node of the network determines locally its x-coordinate and assigns it to the variable x_{min} assumed to represent the minimum x-coordinate of the network. Then, it will broadcast it and wait for other x_{min} values coming from its neighbors. If a received value x_{min} is less than its local x_{min} value then this one will be updated and broadcasted again. This process is repeated by each node as long as a received value is less than its local x_{min} value. After a certain time t_{max}, there will be only one sensor node that has not received a value that is less than its local x_{min} value. This node is at the extreme left of the network. It will be considered as the starting node of the D-LPCN algorithm. The pseudo-code of this process is given by Algorithm 2, where t_0 is the time of the first execution of the algorithm, which can correspond to the first powering-on of a sensor node, t_c the current local time of a sensor node, and t_{max} the maximally tolerated running time of the algorithm from the first execution to the current time of a sensor node.

Algorithm 2 *MinFind*: The pseudo-code of determining the starting node.

```
1: first_node = TRUE;
2: $t_0$ = getCurrentTime();
3: $x_{min}$ = getX();
4: send($x_{min}$, *);
5: **repeat**
6:     $x$ = read();
7:     **if** ($x < x_{min}$) **then**
8:         first_node = FALSE;
9:         $x_{min} = x$;
10:        send($x_{min}$, *);
11:    **end if**
12:    $t_c$ = getCurrentTime();
13: **until** ($t_c - t_0 > t_{max}$)
```

Calculating the time complexity of this *MinFind* algorithm will help to set the value of t_{max}. To do this, let us consider a linear network with n nodes representing the worst case. The node with minimum x-coordinate, which is on the extreme left, will send only 1 message and will receive only 1 message. However, the node which is on the extreme right will receive $n-1$ messages and will send $n-1$ messages to send the received and assumed x_{min} coordinate, because it is the node with the largest x-coordinate, and thus, each of the other nodes, except the extreme left one, has at least one node on its left.Therefore, these nodes will systematically send a message to broadcast the newly received x_{min}. Altogether, the message complexity is equal to $M[MinFind] = 2(n-1) = 2n-2$ and if we consider that a sensor can send and receive messages at the same time (full-duplex communication), the time complexity is equal to $T[MinFind] = n-1$. This complexity is reduced in the case where the network is not linear. Since the time complexity is known, it is possible to estimate the value of t_{max}, the required time to find the starting node. As an example, for a network with 100 nodes, where the size of each message is equal to 1024 bits, sampled with a frequency rate of 250 kb/s (case of the 802.15.4 standard) we need 406 ms to find the starting node. Using the CupCarbon simulator, we have simulated two networks with 100 sensor nodes. The first one is linear (cf. Figure 3.5) and the second one is random (cf. Figure 3.6). The simulation results show that the starting node is obtained in 406 ms with a consumption of $1J$ to $9J$ per node for the linear network and in 70 ms with a consumption of $1J$ to $5J$ per node for the second one. In this simulation, the serialization of messages from the microcontroller to the radio module is neglected. However, if the serialization time is taken into account and if we assume that it is equal to 38400 b/s then to determine the starting node, we need 1.5 s for the linear and 190 ms for the random network. Determining an accurate estimator of the value of t_{max} in the case of random networks could be a topic for future work.

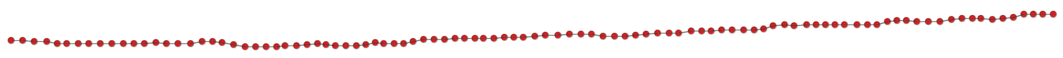


Figure 3.5: A linear network with 100 sensor nodes.

The D-LPCN algorithm

The D-LPCN algorithm uses the same principle as LPCN presented by Algorithm 1 with the exception that the program is written in a distributed form. This means that each sensor node will execute its own program, and by communicating with its neighbors, it can decide if it is a boundary node or not. Algorithm 3 shows the main steps of the D-LPCN algorithm that can be described as follows:

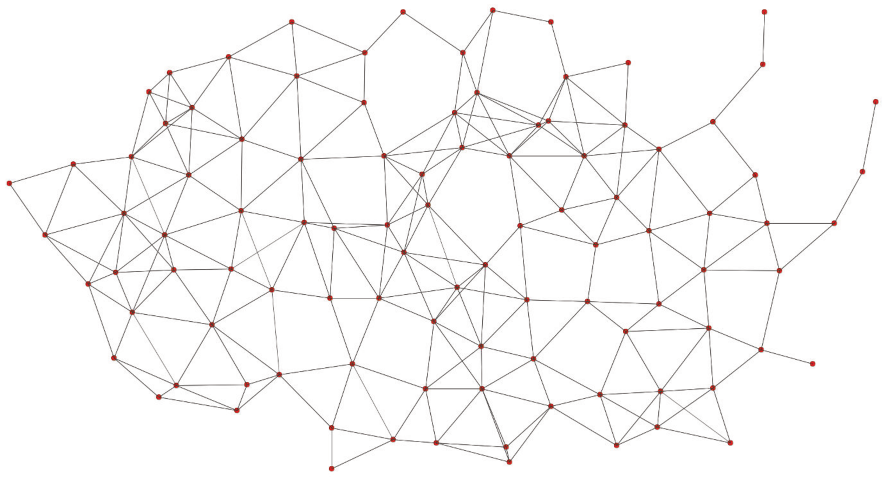


Figure 3.6: A random network with 100 sensor nodes.

- *Step 1 (lines 1 to 4)*: Initialization.
- *Step 2 (line 5)*: Run Algorithm 2 to determine whether the current node is a starting node or not.
- *Step 3 (lines 6 to 10)*: If the current node is a starting node than it launches the D-LPCN algorithm by broadcasting the "AC" message to its neighbors in order to ask for their coordinates.
- *Step 4 (lines 12 to 13)*: Each node is waiting to receive a message.
- *Step 5 (lines 14 to 17)*: The variable i determines the number of received "CS" messages. Receiving n "CS" messages means that the coordinates of all neighbors have been received.
- *Step 6 (lines 18 to 20)*: If the node receives an "AC" message then it will send a "CS" message containing its coordinates to the transmitter having the identifier *id*.
- *Step 7 (lines 21 to 27)*: For all received "CS" messages, the node calculates the minimum angle formed with them by taking into account the intersections with the received boundary set (in line 15). This calculation is done using the function angleWI (angle without intersections).

- *Step 8 (lines 28 to 23)*: The reception of an "SN" message by a node means that this node has been selected as a boundary node by its previous boundary neighbor node. This node will then restart the process of finding the next boundary node by broadcasting a "CS" message.

Finally, Algorithm 3 stops when the first boundary node is selected a second time with an "SN" message. This algorithm, as it is presented, will continue to determine the boundary nodes continually. If we want to add the stop condition, we have just to add the following code after line 28:

```
if (first_node) then
    STOP;
end if
```

To better explain how this algorithm works, we will use the example of Figure 3.7 which represents a WSN with eight sensor nodes. Let us consider the set $S = \{S1, S2, ..., S8\}$ of these nodes and the set B of the boundary nodes, which initially is empty.

First, after the initialization (*Step 1*), we run Algorithm 2 (*Step 2*). The only node which will be considered as the starting node is $S1$, and thus the boundary set is updated to *boundary_set* $= \{S1\}$.

Next, the node $S1$ broadcasts an "AC" message to its neighbors $N(S1) = \{S2, S3, S4, S7\}$ (*Step 3*) to ask for their coordinates (cf. Figure 3.7(a)) while the other nodes are waiting for the receipt of messages (*Step 4*).

Next, each neighbor node $S2, S3, S4$ and $S7$, which receives the "AC" message, sends a "CS" message (*Steps 5 and 6*) to the boundary node $S1$ (cf. Figure 3.7(b)) which will calculate the angle formed by the fictive node $S1'$ with itself and with each of its neighbor nodes $S2, S3, S4$ and $S7$ (*Step 7*). This situation is illustrated by Figure 3.7(c).

Then, as shown by Figure 3.7(d), the boundary node $S1$ will send an "SN" message to the new boundary node $S3$ which will update its boundary set to *boundary_set* $= \{S1, S3\}$ (*Step 8*). The obtained situation is shown by Figure 3.7(e).

The next boundary node $S3$ will then perform the same procedure from *Step 2* on. Figure 3.7(f) shows the final iteration of the D-LPCN algorithm.

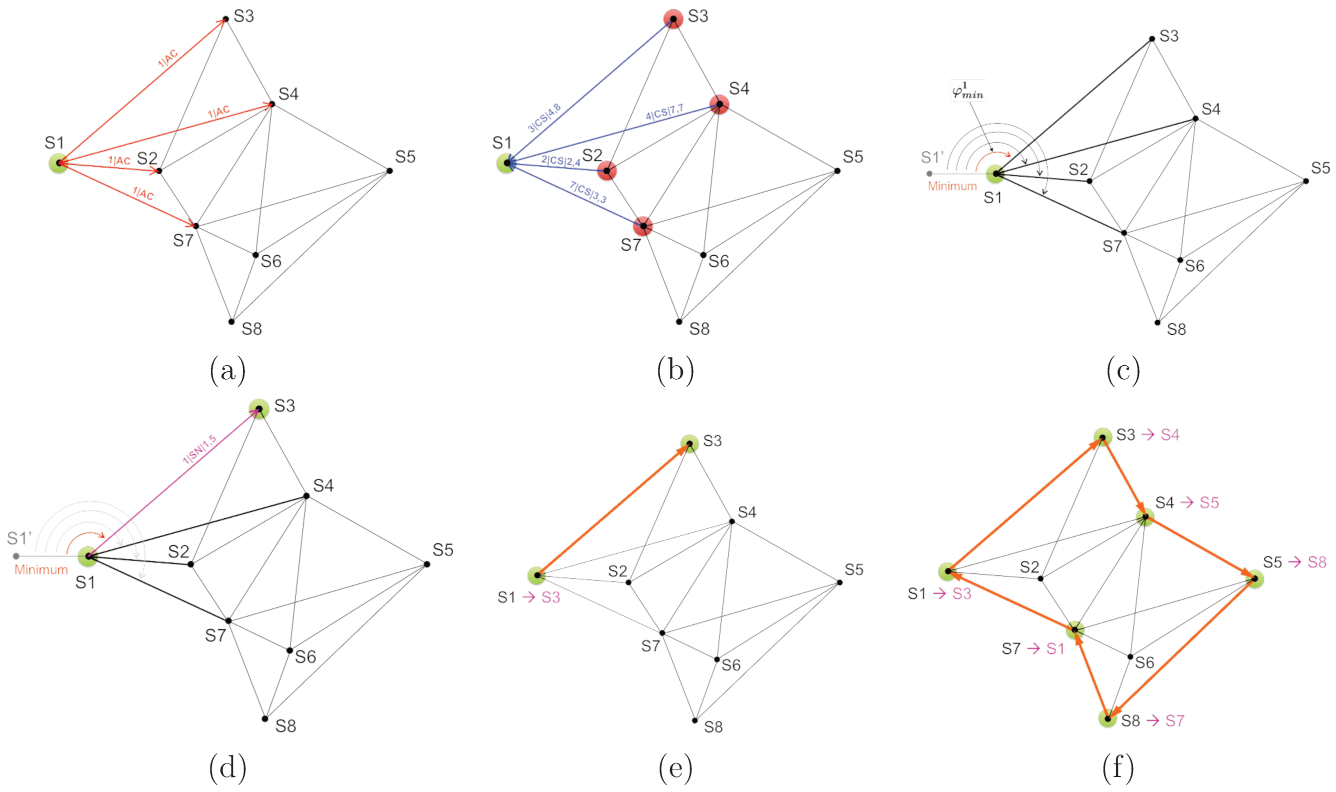


Figure 3.7: An example for the D-LPCN algorithm.

Altogether, the pseudo-code of D-LPCN is given by Algorithm 3.

3.3.3 Implementation and Simulation Results

In this section, we give a brief description of simulators and platforms that we have used to simulate and to implement the proposed algorithm. In particular, we will present some simulation results in terms of energy consumption. The remainder is then divided into three main parts. The first part is dedicated to present the simulation of the D-LPCN algorithm under the CupCarbon simulator. The second part is dedicated to its real implementation on the Arduino/XBee and TelosB platforms. This part allows to validate the proposed algorithm. In the last part, we will study the energy consumed by the proposed algorithm.

Algorithm 3 The D-LPCN algorithm

```
1: boundary = false; phi_min = 10;
2: c_id = getId(); c_coord = getCoord();
3: boundary_set = $\emptyset$;
4: n = getNumberOfNeighbors(); i=0;
5: Run Algorithm 2 to determine the value of first_node;
6: if (first_node) then
7:     boundary = true;
8:     p_coord = (c_coord.x−1, c_coord.y);
9:     send(c_id+"|"+"AC", *);
10: end if
11: repeat
12:     id = read();
13:     type = read();
14:     if (i==n)  then
15:         boundary_set = boundary_set $\cup$ {c_id};
16:         send(c_id+"|"+"SN"+"|"+c_coord+"|"+boundary_set, n_id);
17:     end if
18:     if (type=="AC") then
19:         send(c_id+"|"+"CS"+"|"+c_coord, id);
20:     end if
21:     if (type=="CS") then
22:         n_coord = read(); i=i+1;
23:         phi = angleWI(p_coord, c_coord, n_coord, boundary_set);
24:         if (phi<phi_min)  then
25:             phi_min = phi; n_id = id;
26:         end if
27:     end if
28:     if (type=="SN") then
29:         boundary = true; phi_min = 10; i=0;
30:         p_coord = read();
31:         boundary_set = read();
32:         send(c_id+"|"+"AC", *);
33:     end if
34: until false
```

Simulation

To validate the proposed D-LPCN algorithm, we have used three simulators: CupCarbon [13], Tossim [14], and Contiki OS [15]. We use the Contiki simulator in our work because it is able to accurately measure the energy consumed by real sensor nodes when executing the D-LPCN algorithm. This option is done using the *Powertrace* tool [105].

The red nodes of Figure 3.8, highlighted by the arrows, show the nodes with the minimum x-coordinate in each sub-network of a network containing several connected components. This result is obtained by Algorithm 2 (Minimum Finding algorithm) presented in Section 3.3.2, which constitutes the necessary step before running the D-LPCN algorithm. The boundary nodes, highlighted in yellow, are found by Algorithm 3 (D-LPCN algorithm) presented in Section 3.3.2 which starts from the red nodes determined in the previous step.

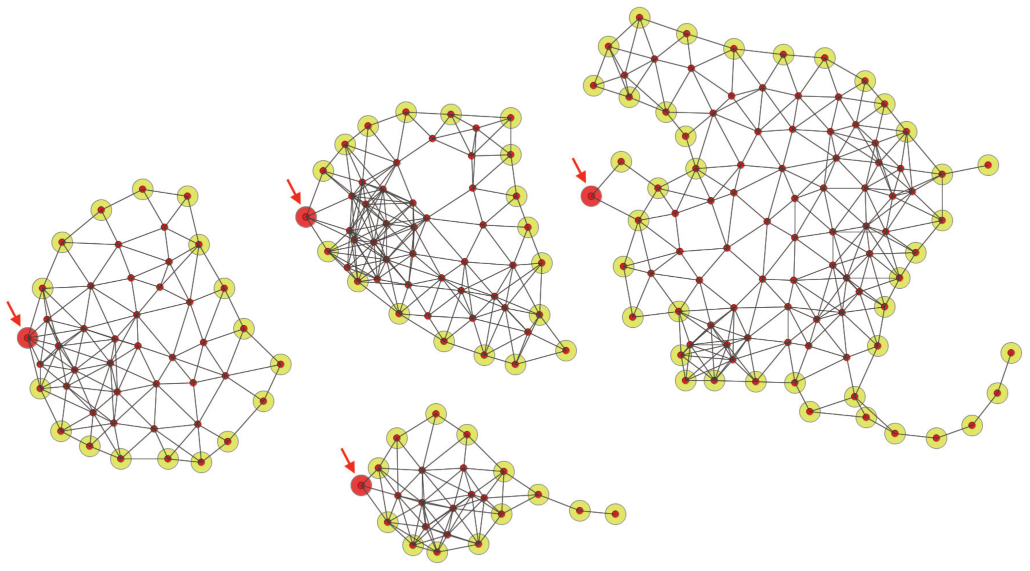


Figure 3.8: CupCarbon simulation of Minimum Finding and D-LPCN algorithms for a disconnected network.

Figure 3.9 shows an example of detecting boundary nodes using the ring topology. The boundary nodes are represented in yellow. The red nodes represent the neighbors of the boundary nodes. In this example, the algorithm is executed periodically for instance to recalculate the new boundary if some of the boundary nodes fail. As shown by Figures 3.9(b) and (c), a new boundary is found after injection of faulty nodes. Figure 3.9(c) also shows that a ring topology is useful for monitoring a sensitive site since D-LPCN can continue to work on the basis of the inner nodes even if a complete part of the ring is broken or destroyed.

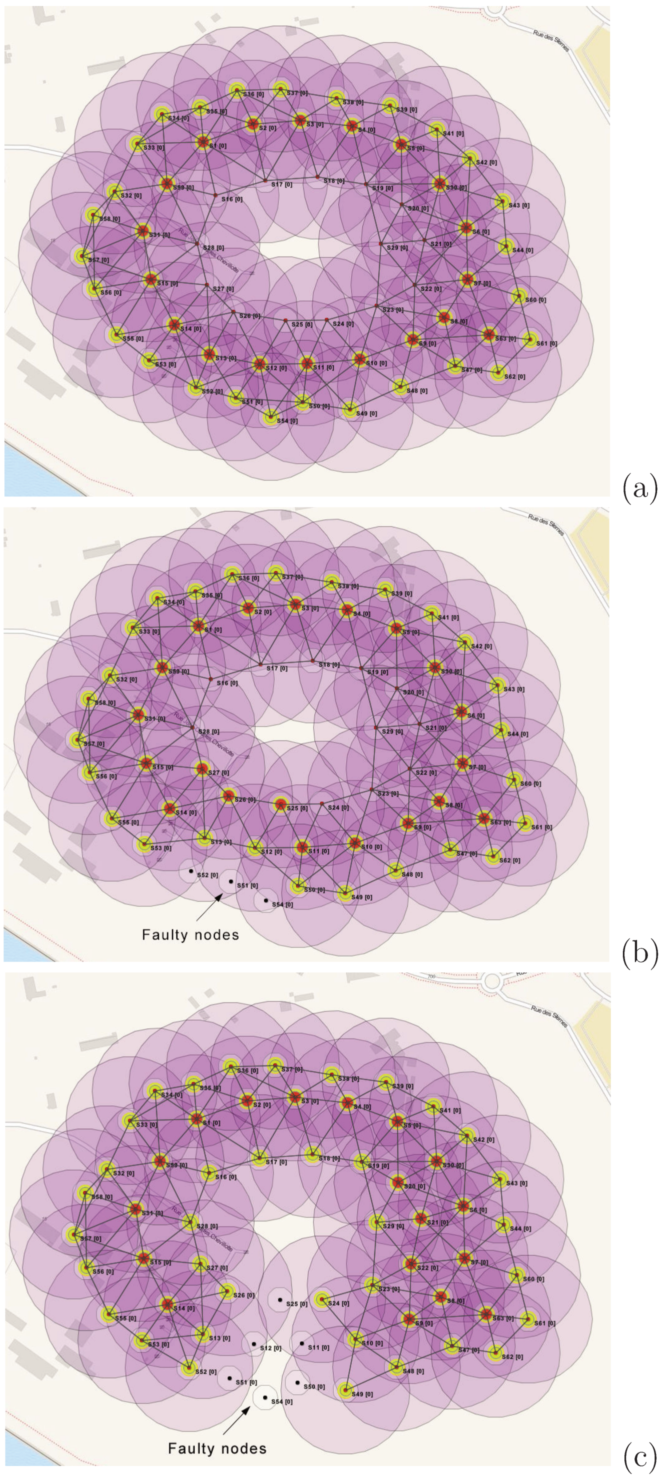


Figure 3.9: Simulation of D-LPCN using the CupCarbon simulator.

Implementation

Once the algorithm validated by simulation, we have implemented it using two real WSN platforms. Figure 3.10 shows the first WSN which is based on the

Arduino cards [106] and the XBee (Series 1) radio module working with the 802.15.4 protocol. Figure 3.11 shows the second WSN which is based on the TelosB sensor nodes. In each figure, the boundary nodes are represented by the nodes with switched-on leds that are designated by arrows.

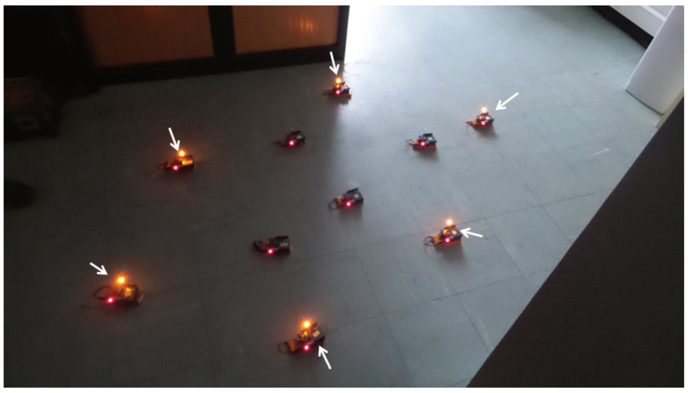


Figure 3.10: Execution of D-LPCN using real Arduino/XBee sensor nodes.

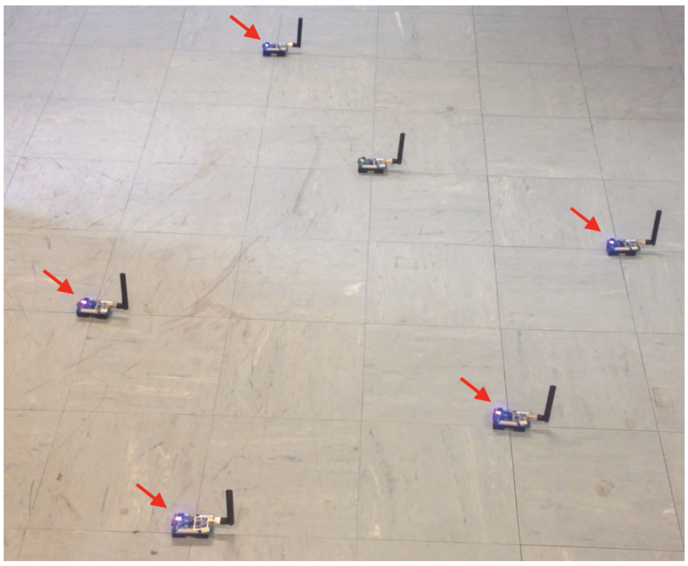


Figure 3.11: Execution of D-LPCN using real TelosB sensor nodes.

Energy consumption

To estimate the energy consumption of the proposed algorithm, we use in this work the energy model of the TelosB sensor nodes which is described in [107]. Its energy consumption is estimated to $59.2\mu J$ for the transmission of one byte and to 28.6μJ for the reception of one byte. To profile the power consumed by the sensor nodes, we have used the *Powertrace* tool [105] which estimates the energy consumption with an accuracy very close to the real one. Experiments in [15] show that the energy consumption obtained by the *Powertrace* tool are very close with 94% to the energy consumption of a real device. *Powertrace* calculates the power consumption of the local node based on the monitoring of the power state. Then, this value is encapsulated according to different activities like reception or transmission of packets, computation, idleness,..., etc. The energy consumption is computed in mW as follows:

$$E = \frac{Energest\_Value \times Current \times Voltage}{RTIMER\_SECOND \times Runtime} \tag{3.3}$$

where $Energest\_Value$ is the difference between the number of ticks in two time intervals (for example difference between CPU in time interval 10 and 20), $Current$ is equal to 17.4mA for packet reception, to 18.8mA for packet transmission and to 0.33mA for CPU operations (computing). The variable $Voltage$ is set to 3V and $RTIMER\_SECOND$ is set to 32KHz which represents the number of ticks per second and which is predefined in the RTIMER-library of the Contiki software. It provides a scheduling and execution of real-time tasks (with predictable execution times).

We use Equation (3.3) to study the energy consumption of the proposed algorithm. To this end, we have considered two main cases to estimate the energy consumption of a boundary node. In the first case, we consider a boundary node which is connected to only two boundary nodes and to other non-boundary nodes. In the second one, we consider a boundary node that is connected only to boundary nodes. Figure 3.12 shows the first case where a boundary node S_1 is connected to two boundary neighbor nodes, to zero such nodes, one (S_7), or many other non-boundary neighbor nodes ($S_7, ..., S_m$).

Figure 3.13 shows the evolution of the energy consumption of a boundary node S_1 as a function of the number of its neighbors including two boundary neighbor nodes and other non-boundary neighbor nodes. We consider the case where this

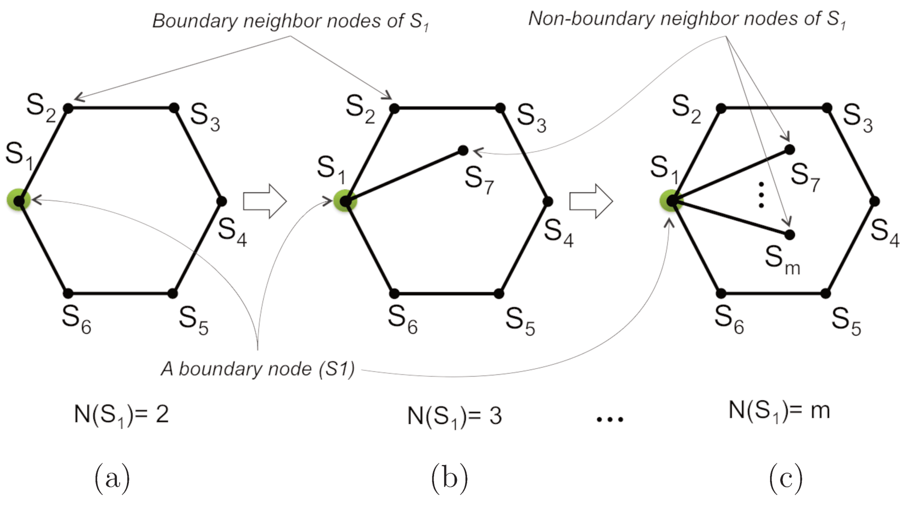


Figure 3.12: A boundary node S_1 connected to two boundary nodes (S_2 and S_6) and to non-boundary nodes (S_7), ($S_7, ..., S_m$).

boundary node S_1 is connected only to two boundary nodes, i.e., it is not connected to any other non-boundary neighbor node (cf. Figure 3.12(a)). Then, we observe from the first marker of the curve of Figure 3.13 that the energy consumed by node S_1 when running continually the D-LPCN algorithm is equal to $8.6\mu J$ each second. If we assume that this node has a source of two Super Alkaline AALR6 batteries with a capacity of $9580J$ each (i.e., a total of $19160J$), we can conclude that this node consumes $45 \cdot 10^{-6}\%$ of energy each second. This means that the lifetime of this node is around 25 days. If we consider the case where a boundary node is connected to 10 non-boundary neighbor nodes (cf. Figure 3.12(c)), the energy consumption ($\simeq 12,000\mu J$) increases by $18 \cdot 10^{-6}\%$ (cf. the 9^{th} marker of the curve of Figure 3.13). This means that the lifetime of this node decreases to 18 days. Therefore, we can conclude that each additional 10 non-boundary neighbor nodes lead to a reduction of 7 days of the lifetime of a boundary node.

Now, if we consider the case where the neighbors of S_1 are only boundary nodes, too, as shown by Figure 3.14, then if the neighbor list table is determined by each sensor node, the situation is identical to the one explained above.

However, if the neighbors must be determined periodically or before each angle calculation, as it is the case in Algorithm 3, then the consumption will increase. This situation is useful in order to take into account the possibility of faulty nodes

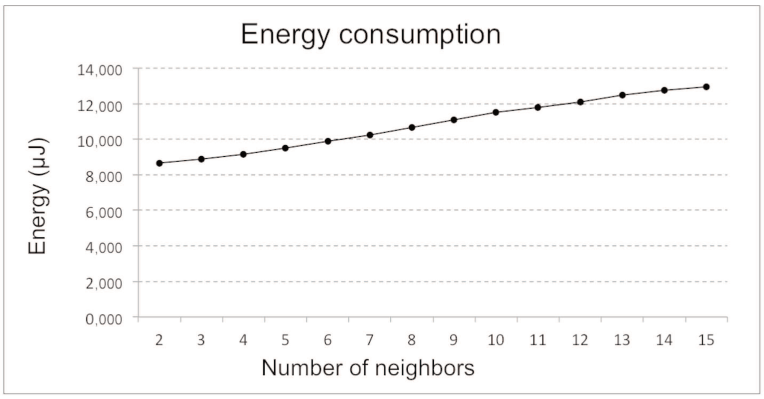


Figure 3.13: Energy consumption of a boundary node S_1.

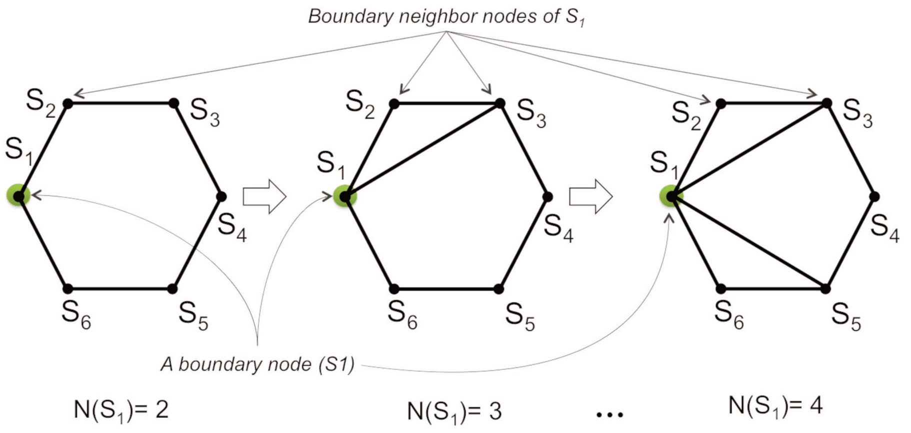


Figure 3.14: A boundary node S_1 connected to up to 4 boundary nodes (S_2, S_3, S_5, S_6).

(cf. Figure 3.9), but it is energy consuming. Figure 3.15 gives an illustration where we consider a network with 35 nodes including 17 boundary nodes (highlighted in yellow) and 11 nodes that are connected to the boundary nodes (highlighted in red). The remaining nodes are not connected to the boundary nodes.

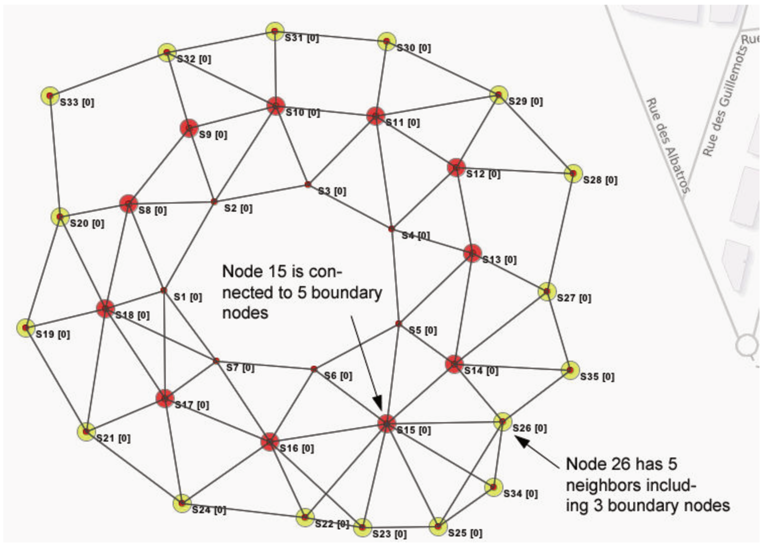


Figure 3.15: A Wireless Sensor Network with 35 nodes.

Figure 3.16 shows the energy consumption of these nodes. The green bars represent the energy consumption of the yellow nodes, the red bars represent the energy consumption of the red nodes and the black bars, which are equal to zero, represent the energy consumption of the other nodes. As we can see, node $S15$ is the node that consumes the maximum energy among the neighbors of the boundary nodes because it has the maximum number of boundary neighbors. We can also observe that even if this node is not a boundary node it consumes more energy than the boundary node $S33$, for instance. In the same way, node $S26$ is the node that consumes the maximum energy among the boundary nodes because it has the maximum number of neighbors. Furthermore, if we compare the energy consumption of these two nodes, we find that $S26$ consumes more energy than $S15$ even though $S15$ has more neighbors. This is justified by the fact that the node $S15$ is only asked 5 times by its 5 boundary neighbors to send its coordinates. However, while the node $S26$ is asked to send its coordinates by only 3 boundary neighbors, it has to send a message to ask the other nodes for their coordinates and to send another message to choose the next boundary node. Moreover, this node has received a message from the previous boundary node saying that it has been chosen as a boundary node. Finally, the inner nodes that are not connected to the boundary nodes will consume a negligible energy since they do not communicate with them. These nodes can be useful for a replacement of possibly faulty nodes, which ensures a real fault-tolerant wireless sensor network.

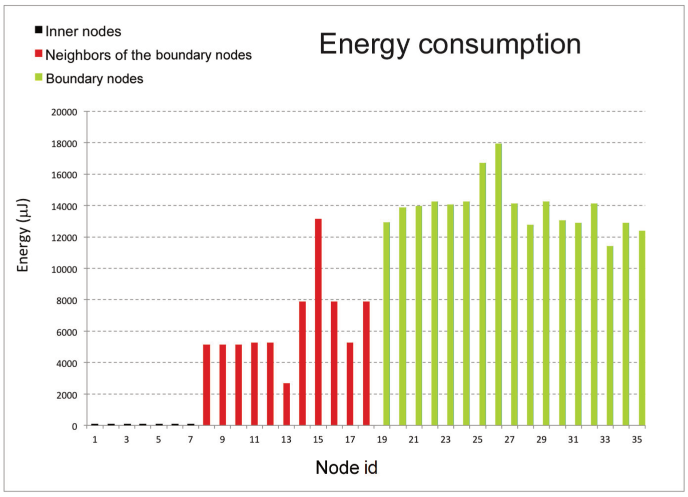


Figure 3.16: Energy consumption of the nodes from Figure 3.15.

We conclude that the energy consumption does not depend on the number of nodes of the network nor on the number of the boundary nodes, but instead depends on the number of the neighbors of the boundary nodes.

Comparison with existing methods

Figure 3.17 presents the energy consumption of the boundary nodes with respect to the number of their neighbor nodes given by some existing algorithms [91] [95] [96]. It shows that the energy consumption increases with the number of neighbor nodes (degree).

As we can see, the energy consumptions obtained by LVP, DBD, D-LPCN algorithms are close to each other (between $10mJ$ and $150mJ$). However, the energy obtained by the Hop-based algorithm is important (between $10,000mJ$ and $100,000mJ$). This is due to the number of hops required by this algorithm. The D-LPCN algorithm consumes less energy than all the others. This can be explained for the case of the DBD algorithm by the necessity to use 3-hop communications, in the case of the LVP algorithm by the important local computations (use of Voronoi diagrams) and for the case of the Hop-based algorithm by the important number of multi-hop communications. In addition, the D-LPCN algorithm uses only the boundary nodes and their neighbors. The other nodes remain stable in terms of energy.

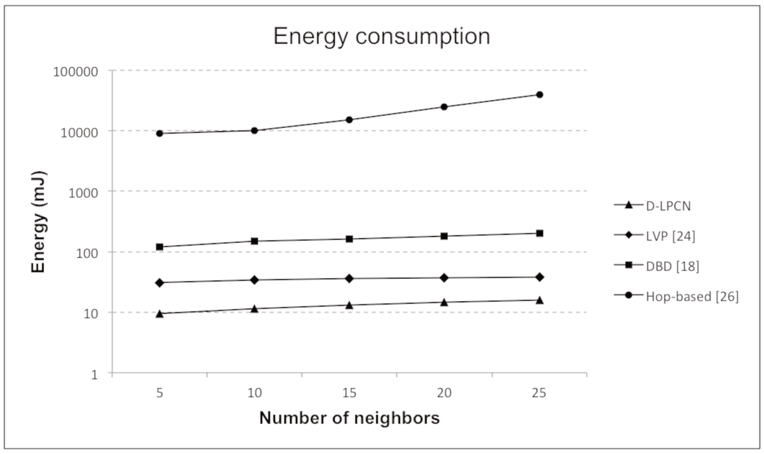


Figure 3.17: Comparison of the energy consumption with those of existing methods.

3.3.4 Comparison with the centralized version

In this section we will compare the non-centralized version of the D-LPCN algorithm with the centralized one. The comparison will be done according to the message complexity (the number of messages exchanged between nodes) and the energy consumption. The centralized algorithm is designed to work on a powerful central machine which is considered as a sink. Each sensor node will send its identifier and its position to the sink. For the case that the sensors cannot communicate directly with the sink, these informations will be sent over multiple hops. The sink will then run the classical LPCN algorithm [102] to find the boundary nodes and send the obtained solution to the nodes. The energy consumption of this version increases with the network size. The nodes close to the sink will be called often by the other sensor nodes, but then they will die quickly. For example, we can see in Figure 3.18 that the nodes S_{12} and S_{13} will be called for each message sent by any other node to the sink. Also, in this version, all sensor nodes are working.

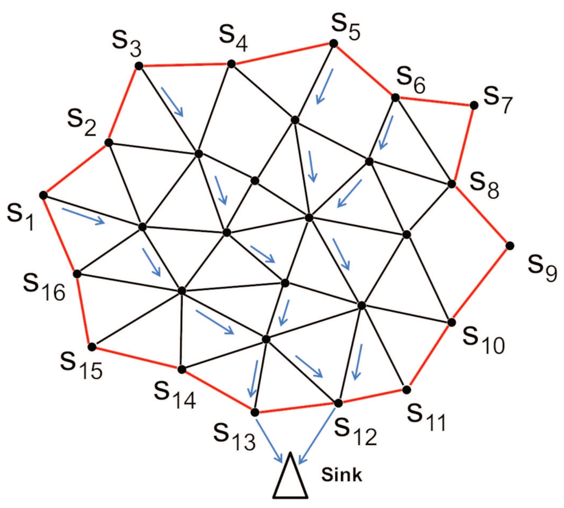


Figure 3.18: The centralized version of the D-LPCN algorithm.

In the distributed version, however, each node communicates only with its neighbors and takes decisions locally to determine the next boundary node. In this case, the sink is not required. As we can see in Figure 3.19, some nodes are not called at all. The only nodes that are necessary to run the algorithm are the boundary nodes and their neighbors. Therefore, the non-called nodes can be either removed or used to replace possibly faulty nodes.

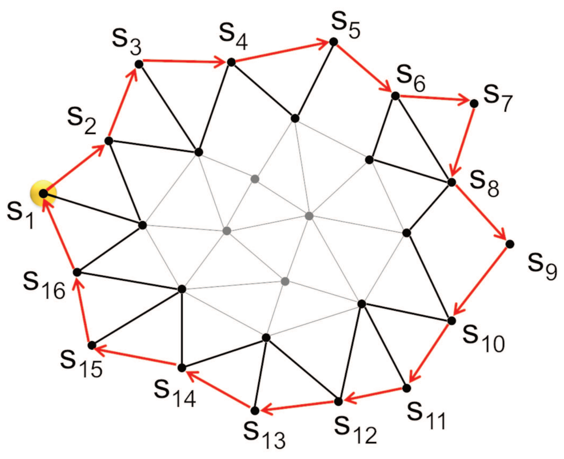


Figure 3.19: The distributed version of the D-LPCN algorithm.

Overall, we can conclude that the distributed version is more useful than the centralized one. The calculations that are done in each sensor node are not energy consuming. The energy consumption is significantly improved by reducing the communications between sensor nodes. According to [108], the energy required for transmitting a single bit could be used to execute 1000 to 2000 instructions. The complexity of the centralized version is equal to $O(ln^2)$ where n is the total number of sensor nodes, l the number of messages sent in multiple hops from the node s_i to the sink or from the sink to any other node. The square comes from the exchanged messages that can be done in both directions. However, the complexity of the distributed algorithm is $O(kh)$, where k is the maximum degree of the network and h the number of the boundary nodes. This complexity can be further improved by increasing the number of starting nodes. If this number is equal to m then the new complexity is given by $O(kh/m)$.

3.3.5 Other variants

The D-LPCN algorithm can be implemented with some minor modifications allowing to increase the speed of the algorithm which can be very useful in the case of applications where this parameter is important. We propose to start the algorithm from several starting nodes instead of just one, which all have to be boundary nodes. As shown by Figure 3.20, the network is divided into two parts and two starting nodes are chosen (S_1 and S_9). The algorithm stops when all starting nodes have been visited twice. The same process can be duplicated. Finally, it is possible to choose any number of starting nodes. However, we may end up with situations where the algorithm stops without finding the final boundary nodes.

Another variant of the D-LPCN algorithm starts with a boundary node from which two other boundary nodes will be chosen. As shown by Figure 3.21, the first one is chosen using the minimum polar angle (in a clockwise direction) and the second one is chosen using the maximum polar angle (which also represents the node with the minimum angle in the anticlockwise direction).

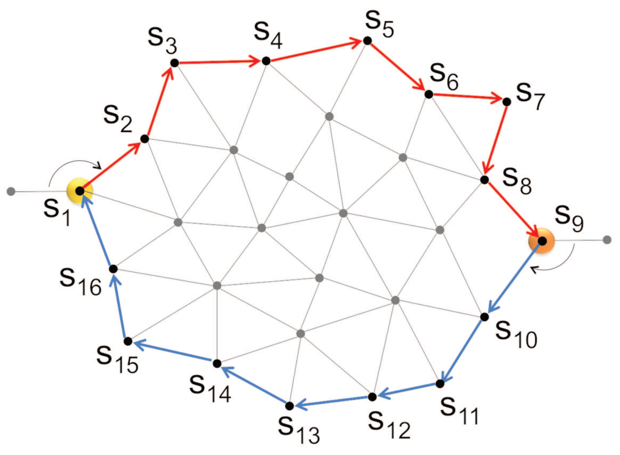


Figure 3.20: D-LPCN with two starting nodes.

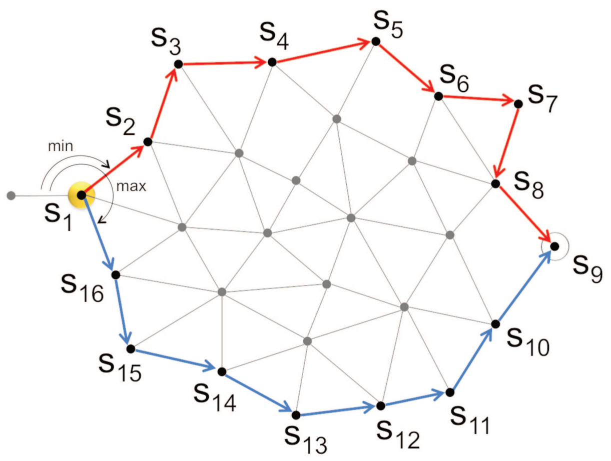


Figure 3.21: D-LPCN with two boundary nodes chosen from a starting node.

We conclude this section with the remark that starting from many nodes can improve the speed of finding the boundary nodes. However, as the stopping condition is based on visiting the starting node twice, it is possible to get such a situation for all starting nodes without finding the final boundary nodes.

3.4 Conclusion

In this chapter, we have examined the problem of boundary point detection which can help the distributed Data mining techniques to reduce a huge volume of data generated by sensor nodes, and improve the WSN operations by reducing the different resource constraints and extending its lifetime. We have divided the chapter into two main sections. In the first section, we have proposed the LPCN algorithm allowing to find the boundary points for a set of data points. The main idea of this algorithm is to choose in each iteration and for each boundary vertex its nearest polar angle vertex with respect to the vertex found in the previous iteration. Its complexity is $O(kh^2)$ where k is the maximum degree of the graph and h the number of vertices on the polygon hull. The LPCN algorithm is useful in Data mining applied for WSN applications, since it can be used to represent a subset of sensed data that have possibly overlapping classes, to detect the shape of the extracting clusters in a set of sensed data, to detect anomalies, classification, hereby reducing the transmitted data which extends the lifetime of WSNs. In the second section, we have proposed the D-LPCN algorithm for finding a minimal set of connected boundary nodes in a WSN. The main advantages of this algorithm are that it works with any type of network, planar or non-planar, relies exclusively on the nodes of the boundary and their neighbors, works with any topology even disconnected (i.e., with many connected sub-networks) and with any density of the network. In the case of a disconnected network, it can determine the boundary nodes of each connected component. Furthermore, it takes into account any blocking situation, which it is able to overcome. Its complexity is $O(kh)$ where k is the maximum degree of a sensor node in the network and h the number of boundary nodes of the network. This complexity is reduced compared to the LPCN algorithm by the mean of the distribution method. The proposed algorithm has been implemented using real sensor nodes based on the TelosB and Arduino/XBee platforms. We have shown that the distributed version is less energy consuming than the centralized one. This is due to the important number of messages that are exchanged between the sink and the nodes in the centralized version, while in the distributed version only the boundary nodes and their neighbors are communicating. On the basis of our results, we may even conclude that the energy consumption is reduced with respect to the energy consumption of the existing algorithms. The simulation results also indicate that the proposed algorithm can be applied to very large sensor networks and it is quite optimal in terms of energy consumption.

CHAPTER

FOUR

A DISTRIBUTED DATA CLUSTERING APPROACH

4.1 Introduction

Wireless Sensor Networks consist of a distributed autonomous system of sensor nodes that have limited capacities [109]. These sensor nodes are responsible to cooperatively monitor physical or environmental conditions. They have no fixed or predefined topology, thus allowing an autonomous self-organization based on the communication between them [110]. The possibility of self-organization in WSNs favors distributed approaches which allow the sensor nodes, or clusters of them, to perform localized sensing and processing [109]. This self-organization is still a challenge which requires the development of new methods allowing to collect and send data from sensor nodes of the network to the base station (BS). Then these data will be further processed to provide all kinds of service or strategic analysis for the users. The difference between such enormous data transmissions and the limited energy with bandwidth constraints on sensors motivates the need to combine all data into a high-quality but small set by using data aggregation or data clustering in WSNs [111].

One simple way to process the data collected from the sensors is to perform a post analysis on the centralized task in the base station. However, the transmission of all these redundant data to the base station, which can be obtained by traditional algorithms, increases the energy consumption, the communication costs, and the computational complexity. Therefore, the application of distributed techniques is required. Many distributed Data mining techniques are proposed to deal with this problem such as distributed association rules [1,112] and distributed classification [113–115]. However, only few studies concern distributed clustering for the analysis of large, heterogeneous and distributed datasets in WSNs [116,117], which motivate us to search for a new approach using clustering algorithms and information sharing with single-hop neighbors only in order to process and analyze the dataset more accurately and efficiently.

In this chapter, we propose a new distributed data clustering approach for WSNs, in which the network is flexible and self-organized forming a tree topology. Rather than communicating each individual sensor measurement to a central node for analysis, each sensor node will cluster its local dataset using the hierarchical agglomerative clustering to get the clusters´ contours. Then it can send the local results representing the boundary points of its clusters rather than the whole dataset to its parent (cluster-head). Moreover, each intermediate sensor node (parent) merges the collected contours from its child nodes. This aggregation is repeated by each cluster-head in the tree topology of the network until the base station is reached to build global results. Using this approach, our distributed data clustering technique can minimize communication overhead, which is a major source of energy consumption for sensor nodes, and also avoid to process the redundant data due to their spatial correlation [110].

This chapter is organized as follows: the next section discusses related work on data clustering in WSNs. Section 4.3 presents and discusses the proposed approach. Section 4.4 presents the implementation and the discussion of our experimental results. Finally, we conclude the chapter with Section 4.5.

4.2 Related Work

There is already a substantial body of research related to clustering in sensor networks. Most of these studies are focused on clustering sensor nodes such as LEACH [49], HEED [118], TBC [119], PEDAP [120], and HEEC [121], and all try to design a better network topology for the purpose of extending the network lifetime rather than clustering sensor data for future analytical purposes. Our work is different from clustering sensor nodes as we focus on the clustering of sensed data to aggregate them according to their similarity. Another strategy to aggregate data is based on techniques employing compressed sensing, as is the case in [122–126].

Let us review next the data aggregation strategies based on data clustering techniques that are related to our work.

The authors of [116] have proposed a data correlation-based clustering scheme (DCC) based on the similarity of sensor data along a spatial suppression scheme in order to reduce the data size. The network is organized as clusters of sensor nodes which have similar measurements. Spatial suppression is performed at a cluster-head which computes the difference between sensor reading and representative value. If a cluster-head has redundant data, it will remove them except for the node identification. The results show that DCC reduces 40% of the data size through suppression and extends the network lifetime by 20% to 30%. However, a cluster-head needs more energy to collect similar data and also to communicate with several nodes. DCC is ineffective due to a higher rate of cluster-head creation with low percentage of similar data.

In [127, 128], distributed, hierarchical clustering (DHC) and summarization algorithms (DHCS) for online data analysis and mining are proposed in wireless sensor networks. These methods cluster sensor nodes based on their current data values as well as their geographical proximity, and they compute a summary for each cluster based on a dissimilarity metric called expansion and the dissimilarity between two clusters based on a cluster distribution feature (CDF). Initially, each node considers itself as a cluster and treats its data by computing the cluster data range which is the smallest closed region in the data space to finally calculate the spherical cluster data range represented by the tuple (Center, Radius). Then similar adjacent clusters are merged into larger clusters round by round. In each round, each cluster will try to combine with its most similar adjacent cluster simultaneously. Two clusters can be merged only if both consider one another as the most similar neighbor. These methods terminate when no merging happens any more. However, in DHCS [127], a merging operation can only happen between two clusters which have been resolved in DHC [128]. However, these methods do not to take into account the energy consumption of the nodes.

The authors of [129] have proposed a two-level data aggregation technique based on a clustering architecture in which data are sent periodically from nodes to their cluster-heads. The first level of data aggregation is applied at the node itself to eliminate redundancy from the collected raw data. At the second level, nodes generate redundant datasets based on the variance study with three different Anova tests in order to eliminate duplication before sending them to the sink.

The authors of [117] have proposed a distributed algorithm for clustering sensory data called H-cluster. This algorithm aims to summarize the input dataset as a set of cluster features. The authors used HilbertMap to map d-dimensional sensory data into a two-dimensional area covered by a sensor network. H-cluster has two phases: the first phase consists in merging grid features with local cluster features at each destination node. The second phase is designed to combine the connected local clusters to global clusters. This method suffers from a high rate of data loss and a decrease in energy efficiency as soon as the WSN size increases.

The authors of [46] have proposed a communication-efficient distributed algorithm for clustering sensory data. This method is based on the distributed version of the K-Mean clustering algorithm to send summarized data to the base station. The sensor network is organized as clusters and cluster-heads will only communicate with the base station. Initially, the base station transmits current center locations to cluster-heads. A cluster-head collects data from its sensor nodes and sends them to the base station including count and vector sum of its local data points as well as the sum of the squared distance from each local point to its center. When the base station receives data, it updates the cluster mean, and the algorithm continues until the function convergence is met. The cluster-heads require high memory and high computation power for the summarization of data before their transmission to the base station. Also, the algorithm requires multiple rounds of message exchanging between cluster-heads and the base station which leads to high communication and high energy consumption especially when the number of nodes is relatively high.

A distributed K-Mean clustering (DKC) method for WSN is proposed in [130–132]. The authors develop a network data aggregation mechanism based on adaptive weighted allocation of WSN. First, the sink extracts randomly k records from the data in the WSN. Then each node will divide local data into k clusters according to these centroids and send them to the parent nodes. When the parent nodes receive the information from all the sub-nodes, they will combine the local information with the cluster information and continue to transmit it to an upper node, until the sink has received all the information from sub-nodes. Then the sink combines the information of k clusters and recalculates the mean value of each cluster for a number of iterations until the centroids of k global clusters will be created. To measure the accuracy of data, the feature of weight is introduced. The DKC algorithm is mainly used to process the testing of data at bottom nodes in order to reduce the data redundancy. However, it may generate data loss in the process of iteration which decreases the accuracy of global clustering results.

4.3 Proposed Approach

In this section, we will present the main steps of the proposed distributed data clustering approach in a hierarchical WSN topology. First, we introduce some definitions and primitives. Then we present an algorithm to find the tree topology which is necessary for the aggregation of a clustered dataset realized by the AGNES algorithm [133] where the contours of clusters are found using the LPCN algorithm [134]. Both these algorithms will also be reviewed.

To obtain the appropriate tree topology of a WSN, we use a self organization of the nodes achieved by the modified BSF algorithm in a distributed computing system. The main goal of this step is to improve the WSN lifetime by a reduction of the communication between the sub-clusters. Figure 4.1 gives an overview of the proposed approach which operates in three main steps. The first step, called the parallel step, is executed in parallel without communication between the sensor nodes, where the local clustering is performed using the Hierarchical Agglomerative Clustering (HAC) algorithm [135]. Each sensor node s_i executes the AGNES algorithm on its local dataset to produce m_i local clusters. Once all local clusters determined, their contours are determined using the LPCN algorithm. These contours will be used as representatives of their corresponding clusters. The second step consists in sending the contours from each sensor node to its cluster-head located on the previous level of the tree topology. This also allows to check whether there are any overlapping contours (clusters) to reduce. In the third step, the cluster-heads of a sub-tree attempt to merge overlapping contours of their group. The cluster-heads are elected among the nodes of each group forming the tree topology. Therefore, each cluster-head generates new contours (new clusters). The second and third steps are repeated to merge clusters until we reach the root node (base station) which will contain the global models. The sub-cluster aggregation is done following a tree structure and the global results are located at the top level of the tree (base station). As in all clustering algorithms, the variability in cluster shapes and in densities causes a problem for their detection. Therefore, we will show in Section 4.4 that the LPCN algorithm can efficiently detect the contours of separated clusters having any shape. Moreover, the AGNES algorithm can dynamically determine the number of the clusters without an a priori knowledge of the dataset or an estimation of the number of clusters. In the following sections, we will describe the main features and requirements of these algorithms.

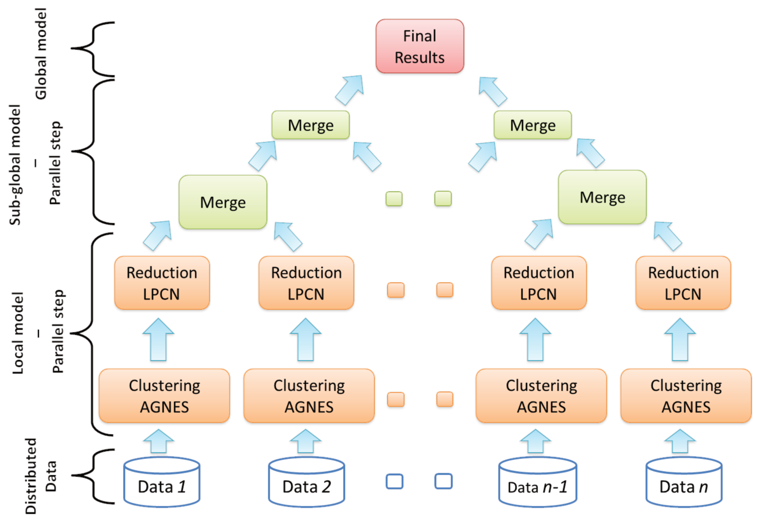


Figure 4.1: Design of the proposed Distributed Data Clustering Approach (DDCA)

The nodes are organized to form a tree topology using the modified BFS algorithm [136], and they work as follows :

1. Each sensor node contains a sensed dataset representing a portion of the overall dataset.
2. Each sensor node executes the AGNES algorithm on its local dataset without any parameters.
3. Each sensor node computes the envelopes of its obtained clusters using the LPCN algorithm.
4. Each sensor node sends its envelopes of clusters to its cluster-head in order to form larger clusters using the merge technique.
5. The cluster-head merges the received clusters and obtains the results of its group.
6. Each cluster-head sends the merged clusters to its previous level cluster-head in the tree.
7. Steps 3, 4, 5 and 6 are repeated until the base station (root node of the tree) is reached.

4.3.1 Definitions and Primitives

The approach presented is based on the distributed computing system inherent to the WSN, which can be modeled by an undirected graph $G = (S, E)$, $S = \{s_1, s_2, ..., s_n\}$ representing the set of sensor nodes, $n =| S |$ their total number and E the set of communication links. We assume that the communication between two sensor nodes is symmetrical. Then the link between two nodes can be defined by a $0-1$ vector e of length $\frac{n\times(n-1)}{2}$ as follows:

$$e_{ij} = \begin{cases} 1 & if\ node\ s_i\ communicates\ with\ node\ s_j \\ 0 & otherwise \end{cases} \tag{4.1}$$

The neighbor nodes $N(s_i)$ of a given node s_i are the nodes that communicate with it.

$$N(s_i) = \{s_j/e_{ij} = 1,\ \ j = 1, ..., n\ \ and\ \ j \neq i\} \tag{4.2}$$

We also assume that the sensor nodes are randomly distributed in the area to be monitored which is mapped into a two dimensional space. Furthermore, each sensor node s_i has a unique identifier given by ID_{s_i} and it can monitor its residual energy level given by E_{s_i}.

For a better understanding of the proposed approach, we define in Table 4.1 some primitives and in Table 4.2 the functions used in the proposed approach.

Table 4.1: Primitives and their definitions

Primitive name	Definition
START	start to construct tree message(ID_{s_0} root node)
NEW	parent message(ID_{s_i}, E_{s_i}, ID_{s_0}, level)
CHILD	child message(ID_{s_i})
ACK	refuse parent message(ID_{s_i})
DONE	confirm tree construction message(ID_{s_i})

Table 4.2: Functions of the proposed approach

Function name	Definition
`getId()`	returns the node identifier
`send(`a`,`b`,`e`)`	send the message a to the sensor node having the identifier b, or in a broadcast ($b = *$) or except the sensor node having the identifier e
`getNumberOfNeighbors()`	returns the number of neighbors of the node
`read()`	waits to receive messages
getCurrentTime()	gets the current time
length(l)	gets the size of liste l

4.3.2 Tree Topology

Building a tree topology rooted at a sink node (base station) for dataset collection is a fundamental method for dataset aggregation in wireless sensor networks. However, due to the nature of the sensor networks, the tree topology should be formed in a distributed way. To build this topology, nodes are based on the quality of signal (RSSI) to identify the minimum-hop distance to the sink, and also on their residual energy which allows to label nodes with less energy as leaves. Hence, we modify the Breadth-First Search (BFS) algorithm to construct an efficient tree topology for WSNs, so that clusters which are subtrees are also formed with energy and quality of signal considerations of the nodes. The algorithm can work with mobile sensor nodes and it can recover from network failures. It achieves the self-organization of sensor nodes into a multi-hop network, reduces transmission distances, since parent nodes are chosen by the highest known signal to the root, and increases the lifetime of the network, as parent nodes are chosen also by the highest residual energy.

The distributed BFS algorithm can be started by an arbitrary node (in our case: the sink) of the network, which becomes the root node of a tree. The main steps of the modified BFS algorithm are described as follows:

- *Step 1 (lines 1 to 3)*: Initialization.

- *Step 2 (lines 4 to 6)*: If the current node is a sink node s_0 than it starts the algorithm by broadcasting the "START" message to its neighbors.

- *Step 3 (lines 9 to 10)*: The node is waiting to receive a message.

- *Step 4 (lines 11 to 15)*: The node which receives the "START" message updates its parent and sends the NEW message to its neighbors except its parent.

- *Step 5 (lines 17 to 27)*: For all received "NEW" messages, the node s_i calculates the maximum performance of the sender based on signal strength (RSSI) and the residual energy of $N(s_i)$ (line 23). The transmitter node is accepted to be parent if it has the maximum performance among the nodes located at the previous level l. This calculation is done during a certain time t_Δ in order to ensure that all "NEW" messages are received.

- *Step 6 (lines 28 to 32)*: When the time t_Δ is over, the node will send a "CHILD" message to its parent found previously (line 22) and it sends an "ACK" message to other nodes. In addition, it sends on its turn the "NEW" message in order to find its children.

- *Step 7 (lines 33 to 35)*: When the node receives a "CHILD" message, it saves the ID of the transmitter in its list of children.

- *Step 8 (lines 36 to 38)*: When the node receives $n-1$ "ACK" messages it sends the "DONE" message to its parent. Notice, that $n-1$ means that all neighbor nodes have their parents or $n-1$ is equal to 0, i.e, there is just one neighbor (its parent).

- *Step 9 (lines 39 to 44)*: The reception of the message "DONE" from all children of a node means that his sub-tree is built. Then this node will send the message "DONE" to its parent.

Finally, this algorithm stops when the root node (sink) has received all "DONE" messages from its neighbors.

Theorem 4.1. *The time complexity of the modified BFS algorithm is $O(n)$ and its message complexity is $O(n \cdot m)$.*

Proof. As the longest path in a network may contain $n - 1$ nodes, the time complexity to build the tree topology is $O(n)$ where n is the number of sensor nodes. Since at every step, there are a maximum of m messages, the message complexity is $O(n \cdot m)$. □

Figure 4.2 shows a wireless sensor network divided into several levels forming the tree topology obtained by the modified BFS algorithm. Each node in the sensor network is assigned its own level based on the signal strength from the base station. The number of these levels depends on various parameters such as density of the sensor network, number of nodes, location of the base station or performance of nodes.

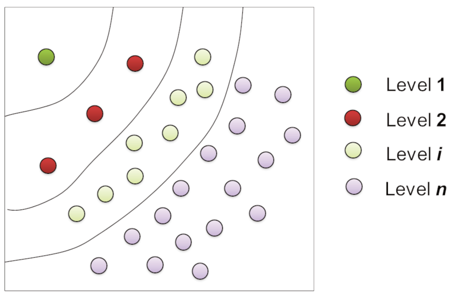


Figure 4.2: Levels of the tree topology

Altogether, the pseudo code of the modified BFS is given by Algorithm 4.

Algorithm 4 Modified BFS algorithm

```
**Input** $t_\Delta$ , $id_{sink}$, $\alpha$, $\beta$
parent = 0;   children = $\emptyset$;   acks = $\emptyset$;   performance = 0;   level = 1000;
$c\_id$ = getId();
$n$ = getNumberOfNeighbors();
**if** ($c\_id == id_{sink}$) **then**
    send($START$, $*$);
**end if**
$t_0$ = getCurrentTime();
**repeat**
    id = read();
    type = read();
    **if** (type == $"START"$) **then**
        level = 1;
        parent = $id$;
        send($NEW$, $*$ , $parent$);
    **end if**
    $t_c$ = getCurrentTime();
    **if** (type == $"NEW"$ && $(t_c - t_0) > t_\Delta$) **then**
        l = read();
        rssi = read();
        energy = read();
        $acks = acks \cup \{id\}$;
        **if** ($\alpha$ rssi + $\beta$ energy > performance && l < level) **then**
            performance = $\alpha$ rssi + $\beta$ energy;
            parent = $id$;
            level = l + 1;
        **end if**
    **end if**
    **if** (parent != $\emptyset$) **then**
        send($CHILD$, $parent$));
        send($ACK$, $acks/\{parent\}$);
        send($NEW$, $*$ , $parent$));
    **end if**
    **if** (type == $"CHILD"$) **then**
        $children = children \cup \{id\}$;
    **end if**
    **if** (length(acks) == $n - 1$) **then**
        send($DONE$, $parent$);
    **end if**
    **if** (type == $"DONE"$) **then**
        num_done = num_done+1;
        **if** (num_done == length(childs)) **then**
            send($DONE$, $parent$));
        **end if**
    **end if**
**until** false
```

In the following, we will explain in more details the design of each phase described above.

4.3.3 Local Models

The local clusters are highly dependent on the clustering techniques used by their corresponding nodes. In this work, we use the HAC algorithm [137] in order to cluster the dataset located at each sensor node. Our goal is to provide an efficient clustering without the global knowledge of the data in the network by reversing the clustering approach from down to up. Our work is based on the HAC algorithm because it is a flexible solution to process the different datasets without requirements, and it is a conceptually and mathematically simple algorithm.

The next step consists in sending all local clusters to the cluster-heads. As the whole dataset is very large, this operation will increase the communication which saturates and reduces very quickly the lifetime of the network. Hence, sending the whole original dataset to the base station should be avoided. In our approach, we only send the clusters' representatives, which constitutes a small part of the original dataset. Any cluster consists of an internal dataset and of boundary points. Therefore, we can use the LPCN algorithm to find the polygon hull of a connected Euclidean graph [102, 134]. This is an efficient algorithm for constructing boundaries, able to accurately detect the shape of a wide range of different point distributions and densities with a reasonable time complexity of $O(kh^2)$, where k is the maximum degree of the graph and h the number of the nodes on the polygon hull.

In the following, we present the algorithm used in HAC Clustering combined with the LPCN algorithm in order to realize the local phase at each sensor node.

HAC Clustering

The first step of the HAC algorithm [137] consists in forming the clusters and the second in joining two clusters based on their similarity. The algorithm is repeated until all clusters are joined into a single cluster. The dissimilarity between clusters is expressed as the distance between two clusters which can be computed using the Euclidean distance between two objects.

The degree of similarity between two clusters is computed with different methods [138] :

- Single LINKage (SLINK) [138], also called the nearest neighbor method, which is given by the minimum distance between two objects contained in the clusters. For example, for two clusters A and B with object a in A and object b in B: $d(A, B) = min_{a \in A, b \in B} d(a, b)$, $d(a, b)$ representing the Euclidean distance between a and b.

- Complete LINKage (CLINK) [139], also called the farthest neighbor method which is given by the maximum distance between two objects contained in the clusters.

- Un-weighted Pair-Group Method using arithmetic Averages (UPGMA) [140] is given by the average distance between two objects contained in the clusters.

For the HAC algorithm, we use AGglomerative NESting (AGNES) [133] which starts with the definition of each object as a single cluster, then joining two clusters at a time until the stopping criterion is accomplished or until we get a single cluster containing all objects. In our case, we fixed a certain distance as stopping criterion in order to obtain separate clusters allowing the LPCN algorithm to determine the contours.

The following Algorithm 5 shows the main steps of the AGNES algorithm :

Algorithm 5 AGNES algorithm

```
1: Input: $n$ objects
2: Output: $k$ clusters
3: Step 1: Label the objects with numbers 1 to $n$. Each object represents a cluster.
4: Step 2: Compute the distance $d(r,s)$ between objects r and s with $r$, $s = 1,2,...,n$; let $D = (d(r,s))$ denote the similarity matrix.
5: Step 3: Find most similar clusters $r$, $s$, by minimizing $d(r,s)$ within $D$.
6: Step 4: Join $r$ and $s$ into a single new cluster $c$. Compute $d(c,k)$ for all clusters $k \neq r$, $s$. Delete the row and column for clusters $r$ and $s$ from $D$ and add a new column and a new row for the new cluster $c$.
7: Step 5: Repeat steps 3 and 4 until the stopping criterion is satisfied.
```

The structure of the AGNES clustering result is hierarchical in the sense that the more we are on a higher level, the bigger and more general are the clusters. The result of the algorithm represents a dendrogram. The AGNES clustering algorithm does not require the number of clusters being specified at the beginning. To find the contours enveloping the clusters of data found by the AGNES algorithm, we use the LPCN algorithm which has been presented in a previous chapter (cf. 3.2).

4.3.4 Global Models

The global model phase is divided into two main parts (cf. our model depicted in Figure 4.1). At each step, sub-global models are generated during the sending of the contours within a sub-tree (from children to parents). This step is executed in a distributed way. Each cluster-head in a sub-tree collects the contours of the local clusters of its children and merges them using the corresponding technique. This will allow the identification of overlapping contours (clusters). Therefore, each cluster-head generates new contours (new clusters). The merge procedure will continue until there are no overlapping contours in the cluster-head. Then the node sends its local contours to its parent (a cluster-head of the previous level) which repeats to merge clusters until we reach the root node (sink) to get the global models of the whole network. The root node will then contain the global clusters (see Figure 4.1). Two cases may occur when running the merge procedure: crossing or inclusion contours. In the crossing case, we found two contours that have at least two edges which intersect. Then we take the smallest $x-$coordinate among the two contours and we apply again the LPCN algorithm. The result of this procedure is a new contour which envelopes the two previous contours. In the second case, the algorithm tests the inclusion of clusters to merge. The inner contour will be removed and the bigger one will be saved. The merge procedure will continue until there are no crossings or including contours. When the merge procedure finishes at each cluster-head, only the representative points for the new merged cluster are selected to be sent rather than all the points in the original clusters. This allows to save energy and to reduce communication time.

4.4 Experimental Results

In this section, we give a brief description of the simulator and the platform we used to simulate and to implement the proposed approach. In particular, we will present some simulation results in terms of energy consumption and execution time. The remainder is then divided into two main parts. The first part is dedicated to the validation of the proposed approach. The second is on the evaluation of our approach within a real platform (TelosB) in terms of energy consumption and execution time.

4.4.1 Validation

Following the general structure of the approach illustrated in Figure 4.1, the AGNES algorithm is executed by each node s_i of the network in parallel, where each node processes its local data to produce l_i local clusters. To give an example, Figure 4.3 illustrates the AGNES clustering results after application to the dataset of one sensor node where bold points represent the centers of the clusters.

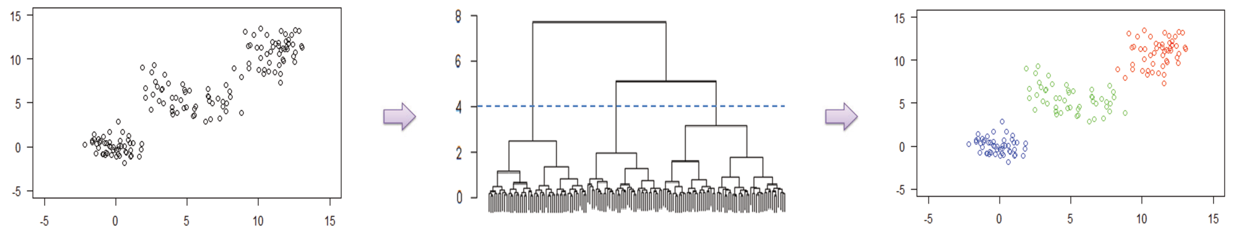


Figure 4.3: Application of the AGNES algorithm.

In the next step, each node finds its local clusters and calculates their contours using the LPCN algorithm. After application to the dataset of the example shown in Figure 4.3, Figure 4.4 depicts the locally obtained contours.

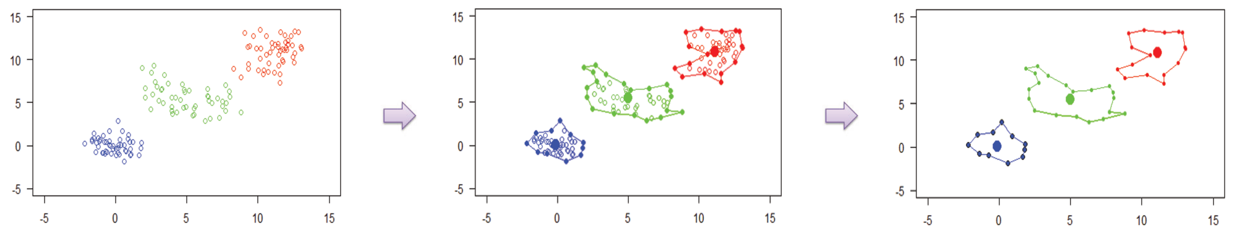


Figure 4.4: Application of the LPCN algorithm.

The process of merging clusters contains the two cases, crossing and inclusion. Figure 4.5 shows the results obtained for three nodes, where each one has already detected its contours and sent these contours to its cluster-head (parent).

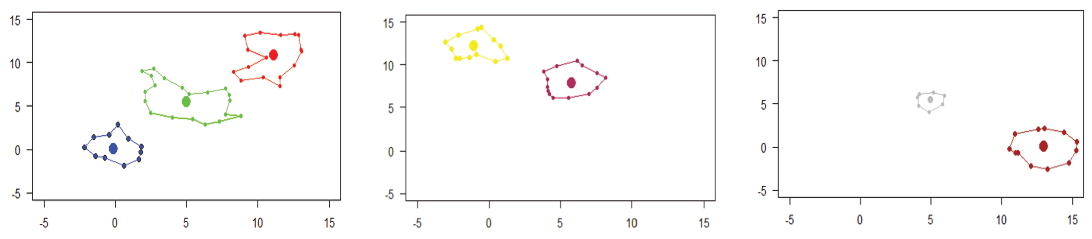


Figure 4.5: The contours of clusters for three nodes.

In the third step, the cluster-head attempts to merge the overlapping contours of its group and it removes the included contours. First, the cluster-head merges the received clusters from node 1 and 2, which are intersecting as shown in Figure 4.6(a), and obtains the new merged cluster by the union of the contours of the two original ones, as shown in Figure 4.6(b).

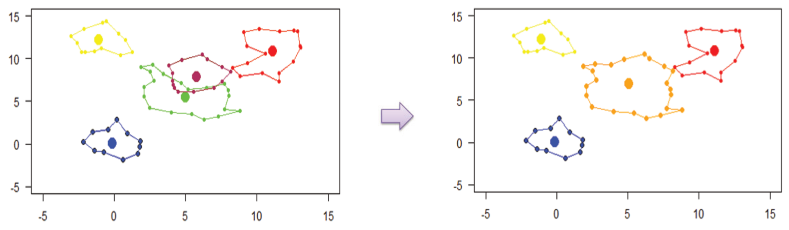


Figure 4.6: Merge of the contours of node 1 with those of node 2.

Second, the cluster-head detects the inclusion of one of the clusters of node 3 within the obtained cluster, as shown in Figure 4.7(a). Then, as indicated in Figure 4.7(b), it removes this cluster and saves the larger one.

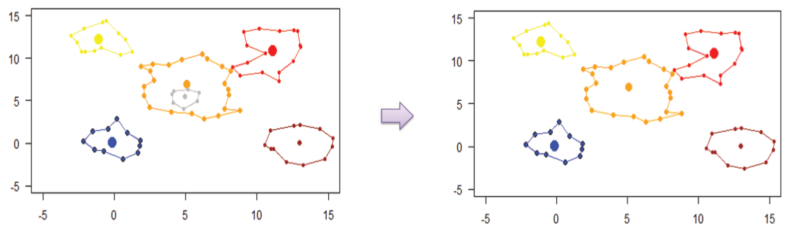


Figure 4.7: Merge of the contours from Figure 4.6(b) with the contours of node 3.

4.4.2 Evaluation

To evaluate the proposed distributed data clustering approach, we have used the Contiki-OS simulator [15]. To estimate the energy consumption of the proposed approach, we use the TelosB sensor node (TPR2420CA) as described in [141]. The time and the energy consumption of our approach are computed according to different activities such as reception or transmission of packets, and computation.

- The time taken to execute the algorithm depends on the frequency of the sensor node processor. In our case, the frequency is 8 Mhz.

- The time needed to transmit a packet from the source node to the destination depends on the node capacities. We adopt the transmission time from the 802.15.4 frame which can be calculated using the formula (4.3) [142]:

$$T = T_{Bo} + T_{Frame}(x) + T_{TA} + T_{ACK} + T_{IFS}(x) \tag{4.3}$$

  where T_{Bo} is the duration of the channel access, $T_{Frame}(x)$ is the transmission time for a payload of x bytes, T_{TA} is the time to receive a response (turnaround time), T_{ACK} is the transmission time for an acknowledgment and $T_{IFS}(x)$ is the IFS time, which is the time to process data received from the physical layer.

- The communication energy cost (E_c) is the energy consumed by transmission (E_{tx}) and reception (E_{rx}). It can computed as in [143]:

$$\begin{aligned} E_c &= E_{tx} + E_{rx} \\ E_{tx} &= V_{dc} * I_{tx} * P_b * T_b \\ E_{rx} &= V_{dc} * I_{rx} * P_b * T_b \end{aligned}$$

  where I_{tx} and I_{rx} are the current consumptions in the transmission and the reception mode, respectively, P_b is the bit length of the packet, T_b is the transmission or reception time of a single bit and V_{dc} is the used voltage.

- The computational energy consumption (E_{comp}) of a sensor node includes the MCU active mode and other modes (e.g., standby/idle/sleep) which is computed as in [143]:

$$E_{comp} = V_{dc} * I_{mcu\_active} * T_{mcu\_active} + V_{dc} * I_{mcu\_sleep} * T_{mcu\_sleep} \tag{4.4}$$

  where I_{mcu_active} and I_{mcu_sleep} are the MCU active and sleep mode currents, respectively. T_{mcu_active} and T_{mcu_sleep} are the MCU active and sleep mode durations, respectively, and V_{dc} is the used voltage. In our case, we assume that the sleep mode duration is equal to 0.

Figure 4.8 shows the evolution of the energy consumption of the modified BFS algorithm as a function of the number of neighbors of each node. In this simulation, we assume that the binary tree topology is built in such a way that each cluster-head (parent) has two children. Then we observe from this figure that the energy consumed by a node in the modified BFS algorithm increases with the number of neighbors. If we assume that the node has 5 neighbors, then it consumes $1.2J$. If we assume further that the node has a source of two Super Alkaline AALR6 batteries with a capacity of $9580J$ each (i.e., a total of $19160J$) and the algorithm is run each 10 minutes in order to rebuild the tree if some nodes fail, then we can conclude that the node consumes 0.0063% of energy each 10 minutes. This means that the lifetime of this node is around 110 days.

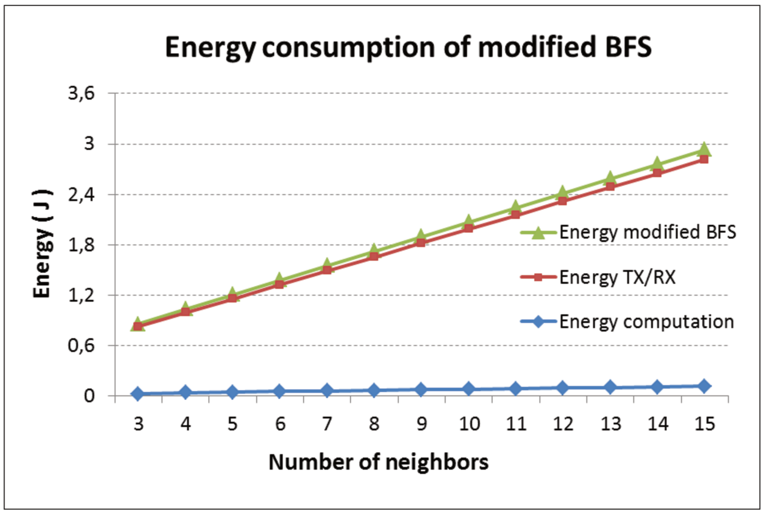
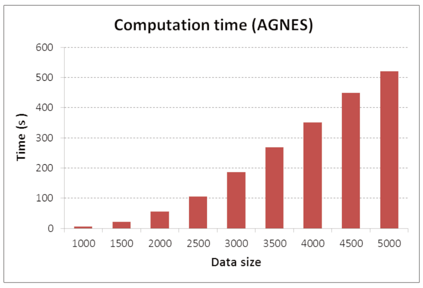
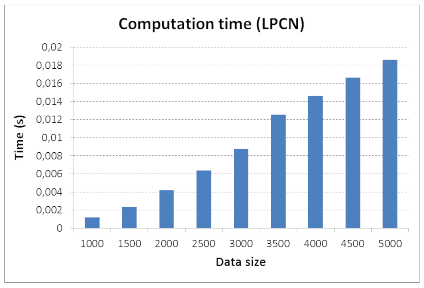


Figure 4.8: Energy consumption of the BFS algorithm in terms of the number of neighbors.

Figure 4.9 shows the computational time as a function of the data size (number of data points) at a given sensor node of the network. Our algorithms based on AGNES and LPCN took only a few seconds to cluster 1000 to 5000 data points. We observe that AGNES takes more time then LPCN (0.001 to 0.02 seconds for LPCN and 6 to 500 seconds for AGNES) because it processes the whole dataset in order to determine the different clusters. However, LPCN takes much less time because it only uses the boundary data points and their neighbors to detect the contours of the clusters. This computation is done within a distributed system, and for a network containing 100 sensor nodes, knowing that each one contains 2000 data points. Then the computation will be done in parallel (and locally) by each node which takes $55,92$ seconds. This shows that the algorithms can be executed comfortably in small sensor nodes with limited capacities because these are low complexities.





(AGNES) (LPCN)

Figure 4.9: Computational time of AGNES & LPCN in terms of data size.

Figure 4.10 illustrates the energy consumption of the AGNES and LPCN algorithms as a function of the size of the dataset. We can observe that the energy consumed by both algorithms is very small, representing $8.77 * 10^{-7}\%$ from the total power in the case of 5000 data points, for example. This shows that our approach can handle high-dimensional data due to a low energy consumption.

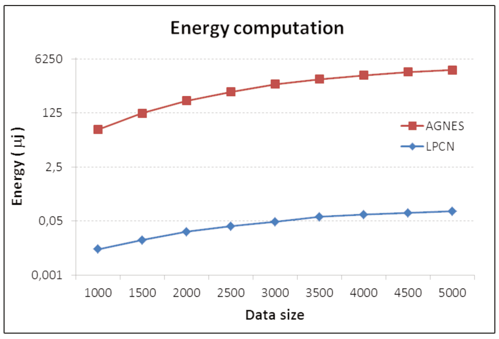
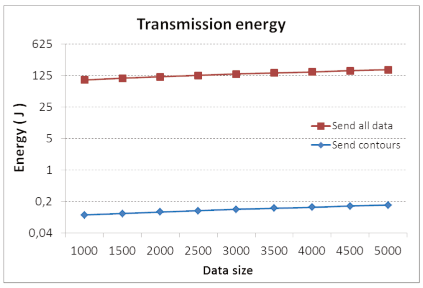
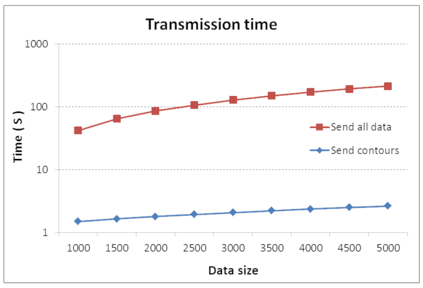


Figure 4.10: Energy consumption of AGNES & LPCN in terms of data size.

Figure 4.11(a) shows a comparison between the energy consumed in the transmission of the whole dataset and the one consumed in the transmission of only the contours of clusters (boundary points). We observe in both cases that the energy consumption increases with the size of the dataset but the energy consumed to transmit the whole dataset is about 1018 times higher than the energy consumed to transmit the contours. A similar result is obtained for the time taken by the transmission of the whole dataset and the one taken for only the contours as shown by Figure 4.11(b). We observe that the time taken to transmit the contours is very much below the time to transmit all data. Our approach allows to reduce the transmission time by a factor of 81 and hereby to increase the lifetime of the network by saving transmission energy.



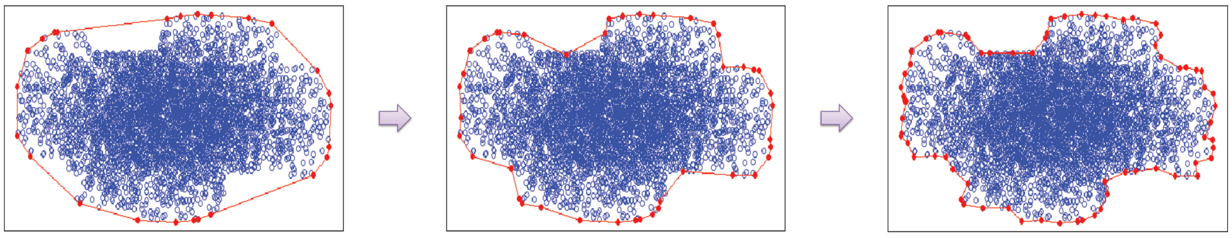
(a)



(b)

Figure 4.11: Transmission energy and transmission time of all data versus boundary.

Figure 4.12 illustrates the performance of the proposed LPCN algorithm to detect shapes of clusters according to the given distance between data points. In this simulation example, the size of data points is 5000. The more we decrease the distance between data points, the more accuracy we obtain in the shape of the cluster.



distance = 1.8 distance = 0.8 distance = 0.4

Figure 4.12: Accuracy of the proposed LPCN algorithm.

As can be seen in Figure 4.13, the percentage of the data points transmitted to the corresponding cluster-head depends on the distance given by users. For more accuracy, we decreased the distance in order to obtain an accurate shape of the clusters. This leads to the transmission of more data points but gives the best global results. However, there is a higher consumption of transmission energy. Otherwise, if we want to save transmission energy, we can increase the distance in order to obtain a small number of data points to transmit. This, however, leads to results with less precision. Figure 4.13 indicates that the percentage of the sent data points varies from 0.5% to 5% of the whole dataset which shows that our proposed approach is less energy consuming in comparison to the existing methods.

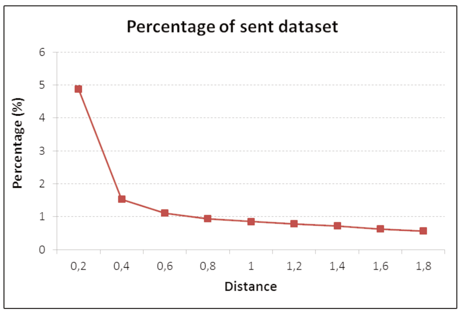


Figure 4.13: Percentage of sent dataset in terms of distance.

Figure 4.14 shows a comparison between our approach (DCCA) and some existing methods in terms of the reduction of data size. We have compared our approach with methods based on data clustering DCC [116] and DAPSN [129], and based on data compression EBDC-DTS [123]. The obtained results show that our approach reduces by more than 95% the amount of collected data after the reduction step, while it sends all the collected data (100%) without applying this reduction step. Therefore, our technique can successfully eliminate redundant measures at each node and reduce the amount of data sent to the base station. We can also observe that our method outperforms the DCC, DAPSN, and EBDC methods by 55%, 30%, and 27%, respectively. Altogether, we can conclude that our approach clearly improves the WSN lifetime compared to the existing methods by considerably reducing the data to be transmitted.

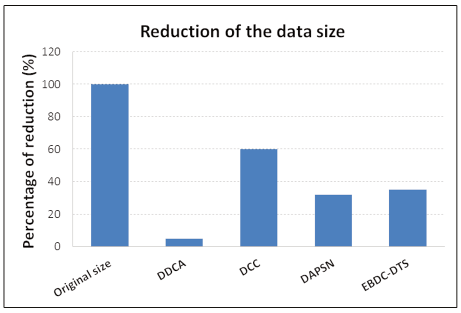


Figure 4.14: Comparison in terms of reduction of the data size.

4.5 Conclusion

In this chapter, we have proposed a new distributed data clustering approach for a hierarchical wireless sensor network topology, dealing with the aggregation of spacial and huge datasets. Our approach is focused on the processing power of the distributed method by maximizing parallelism and minimizing communications and reducing the size of the data to be exchanged among the nodes of the network. It is also based on the tree topology formed by the modified BSF algorithm and based on the received signal strength indication to estimate the best neighbors and the residual energy of nodes in order to build an efficient network with maximum lifetime. The local models are generated using a hierarchical agglomerative clustering algorithm together with the LPCN algorithm at each sensor node, and the local results are then merged to build the global clusters. We also observe that the local models represent only the contours of clusters which are small size datasets representing 0.5% to 5% of the total size of the original dataset, to be exchanged through the network in order to save energy and to maximize the lifetime of the network. The proposed approach is validated using the Contiki simulator and real sensor nodes based on the TelosB platform. Finally, experimental results are presented and discussed. They indicate that the proposed approach can be applied to sensor networks with large datasets for which it is quite efficient in terms of energy consumption. Our results also show the effectiveness of our approach in terms of the quantity of clusters generated and the computational time. Finally, our approach clearly outperforms the existing methods with respect to the network lifetime and the size of the data to be transmitted. In the next chapter, we will propose distributed Data mining techniques applied to a WSN for forest fire detection.

CHAPTER

FIVE

DATA MINING TECHNIQUES FOR FIRE DETECTION USING WSNS

5.1 Introduction

Accurate and timely event detection is an important part in many WSN applications, such as forest fire and environmental pollution. In this kind of time critical applications, the event must be detected early in order to reduce the threats and damages. In this chapter, we propose a new WSN system for early forest fire detection, which is based on the integration of Data mining techniques into sensor nodes. Forest fires are a fatal threat in the world: it is reported that over 8388 forest fires have been counted in the Mediterranean region in France between January 2010 and October 2015 [144]. Forest fires can be deadly threats to the environment and human life. In some of these fires, large areas of forests of more than 21 954.61 hectares have been destroyed [144] and many people or animals have died. Therefore, the monitoring and early detection of forest fires is an invaluable tool for the prevention or reduction of damage to the environment and the protection of human lives.

Number of detection and monitoring technologies and systems have been proposed to detect fire, e.g. : systems employing charge-coupled device optical cameras and infrared detectors, satellite monitoring and images [145, 146] and Wireless Sensor Networks [147–149].

Wireless sensor network technology has been put forward in the literature as a suitable candidate for forest fire detection and monitoring. Its low cost, self-configuring and rapid deployment features makes it an ideal candidate. A WSN is usually composed of a few sinks and a large quantity of small sensor nodes, which are able to sense, process and communicate data [109]. To detect fire, a sensor node can be deployed in the forest and collect data such as temperature or humidity, and deliver these data to the base station (sink node) where they can be processed and analyzed automatically without requiring operations performed by humans. Data sets are growing rapidly because there is a need to monitor a forest as long as possible over time. Since classical algorithms are not designed for the processing of heterogeneous, geographically distributed and complex data collected from the environment, new algorithms have emerged to deal with this problem in the data management. In addition, the design and deployment of sensor networks create many challenges due to their large size (up to thousands of sensor nodes), random deployment, lossy communication environment, limited battery power, limited processing unit, small memory, and high failure rate. Energy consumption is a particularly limiting factor for the lifetime of a node in a WSN. Therefore, processing and communication should be minimized and there is a permanent need to balance the power consumption on all nodes, based on their residual energy. Data mining techniques are a suitable solution to be integrated into the sensor nodes in order to solve the above limitations, e.g., to reduce the size and to improve the quality of the collected data in an intelligent way.

Data mining is a process of extracting hidden patterns from large data sets and a critical component of the knowledge discovery process [150]. This process needs to coordinate predictive analysis and decision support systems in real-time. The purpose of this study was to develop a resource efficient online distributed Data mining approach for WSNs. The approach must minimize inter-node communications and optimize local computation and energy efficiency without compromising fire application requirements. The objectives were to address the

WSN energy constraints, network lifetime, and distributed processing of streaming data. Another objective was to develop a new high spatio-temporal resolution version of forest fire detection.

In this chapter, we use Data mining techniques to process the sensing data within the sensor nodes by taking into account their limited computing and storage capabilities. We are interested in techniques which can find and describe structural patterns in data as a tool to help to explain the data and to make prediction from them. Classification is one of the popular Data mining techniques [151] that consists in predicting the probability of a new instance to belong to a predefined class using the set of attributes describing this instance. We have already presented part of this work in [115].

To this aim, we propose a new approach based on Data mining techniques and a clustered architecture. We consider a WSN deployed to monitor forest fire and alert fire in real-time. The sensor nodes continuously measure weather parameters and feed into the Data mining model, which triggers fire alarm when fire is detected. Our approach does not tolerate offline data analysis. Therefore, the data streaming has to be processed on the fly and in real-time using the distributed Data mining WSN model. Each sensor node can individually decide whether there is a fire or not using a model of Data mining technique created through learning from real historical data in offline mode. When a fire is detected online, the corresponding node sends an alert through its cluster-head, which will pass through gateways and other cluster-heads until it reaches the sink in order to inform the firefighters.

This chapter is organized as follows. In Section 5.2, related work is presented. The design of our approach will be discussed in Section 5.3. Then Section 5.4 exhibits and discusses the obtained results and, finally, Section 5.5 concludes the chapter.

5.2 Related work

In this section, we summarize the various studies which have been proposed regarding the detection of forest fires.

The authors of [152] present a brief overview of detection techniques and monitoring systems for forest fire. They summarize all the methods used for forest fire detection into three main methods which are authorities' techniques, satellite systems, optical cameras, each method being presented with its own advantages and disadvantages. The authorities' techniques are based on the probability of human observation for forest fires, but it is the most expensive proposition to solve this problem. The satellite system is based on images gathered by satellites used for forest fire detection, but it suffers from serious limitations due to a weakness in rapid and accurate surveillance of forest areas. The optical cameras use a digital camera to gather the image for early recognition and forest fire detection, but the difficulty of this method is the processing of landscape images due to the large number of dynamic events that can appear under different illumination conditions, depending on weather, distance and time of the day. WSNs are based on a number of sensor types to sense different parameters of fire and to detect it. This solution can provide all information of the environment at any moment accurately by establishing many kinds of sensors to measure different parameters, it can cover any area size even in scalable networks and it can be deployed anywhere even in inaccessible regions. It is for these reasons that we adopt WSNs for our approach in order to benefit from their advantages.

The authors of [146] propose a combination of a Wireless Local Area Network (WLAN) and sensor-node technology for fire detection. The system is comprised of multi-sensor nodes and IP cameras in a wireless mesh network in order to detect and verify a fire in rural and forest areas of Spain. When a fire is detected by a wireless multi-sensor node, an alert generated by the node is propagated to a central server on which a software application runs for selecting the closest wireless camera(s). Then real-time images from the zone are streamed to the sink. In this study, the sensor nodes are deployed with a large distance between the nodes where the data from sensor and cameras is collected and processed at a base station. However, our proposed approach considers a clustered deployment strategy where the distances between neighboring sensor nodes are rather short. In this way, our goal is to detect forest fire in a more rapid way and to send the related information to a base station as fast as possible.

In [145], the authors study the behavior of forest fires rather than fire detection. They developed a multi-tiered portable wireless system for monitoring environmental conditions, especially for forest fires. The system is comprised of sensor nodes, web-cams, and base stations, which are capable of long distance communication of up to ten kilometers range. The sensor nodes measure the temperature, relative humidity, wind speed and direction of fire. Web-cams provide continuous visual data of the fire zone to the base station. The main drawback of the above reviewed systems is that they all assume resource-rich nodes that are equipped with long range radio. However, our proposed system considers a clustered deployment strategy with rather short distances between neighboring sensor nodes allowing to detect forest fire more quickly and to send the related information to a base station as rapidly as possible.

The authors of [149] combine a WSN with an artificial neural network (ANN) for forest fire detection. The system they propose collects data from a region via a WSN by sensors, such as temperature, light and smoke. All readings are transmitted (after being transformed into information and then into knowledge) to an already trained ANN situated at the central processing unit (base station). This ANN uses the received information to test whether some part of it belongs to the fire class for fire detection. This method suffers from high energy consumption and high response time due to the routing of all data to the base station.

The authors of [148] present a WSN for forest fire detection which is based on the Fire Weather Index (FWI) system, one of the most compressive rating systems for forest fire danger in the USA. This system determines the risk of propagation of a fire according to several index parameters. In this study, weather data are collected by the sensor nodes, and the data collected at the center are analyzed according to FWI. A distributed algorithm is used to minimize the error estimation for spread direction of forest fire.

In [153–156], the authors propose an intelligent method to make a decision using a fuzzy logic system in WSNs for forest fire prediction. This technique allows to describe the fire event in ways that sensor nodes would be able to understand. It uses linguistic variables whose values are not numbers but sentences or words in a natural or codified language. Fuzzy logic is based on rules which are conditional statements in the form of if-then. However, in our work, we use real data-gathering from the environment in the two cases that a fire is present or not, and fuzzy logic cannot really be used when parameters of a fire are numbers.

In [150], the authors present a comparative analysis of various DM techniques on WSN fire detection data using the WEKA tool. The goal was to see which of them has the best classification accuracy of fuzzy logic generated data and which is the most appropriate for a particular application of fire detection. In our system, we use real sensing data and we simulate under conditions close to reality.

A skyline approach for early forest fire detection is proposed in [157]. Skyline is built using greater values, e.g., sensor readings in a two-dimensional attribute space (large temperature and high wind speed). Only data on skyline are sent to a sink to be used for fire detection. The sink processes the data according to the proposed algorithm and eventually detects fire.

[158] presents an extensive literature review over the period 2002-2013 of machine learning methods that were used to address common issues in Wireless Sensor Networks (WSNs). They propose the advantages and disadvantages of each algorithm and provide a comparative guide to aid WSN designers in developing suitable machine learning solutions for specific applications such as fire detection.

In [159], the authors present a guideline for choosing the most optimal sensor combinations for accurate residential fire detection. Additionally, the applicability of a Feed Forward Neural Network (FFNN) and a Naïve Bayes Classifier is investigated and results are analyzed in terms of detection rate and computational complexity. This study considers and handles a single aspect of forest fire detection. In our proposed approach, we use multiple parameters. We aim at an efficient energy consumption for early fire detection, to include a comparison of some DM techniques, to incorporate environmental conditions and to make use of the simulator, which offers an environment close to reality, to validate our approach.

In [160], the authors present a method which applies neural network techniques for in-network data processing in the context of environmental sensing applications of Wireless Sensor Networks in order to detect forest fire. The sensor nodes are organized into clusters. Sensor nodes can measure environmental temperature, relative humidity and smoke. Each sensor node sends the sensing data and its GPS coordinates to the corresponding cluster-head. The cluster-head computes the weather index using a neural network method and sends the weather index to the manager node via the sink. The manager node provides two types of information to users: the report for an abnormal event and the danger rate of forest fire.

In [147], the authors present a framework for forest fire detection, which includes a clustered network architecture for the deployment of sensor nodes, as well as an interaction protocol of intra- and inter-clusters. They develop a simulator to perform simulation tests in order to examine the proposed system protocols and components. In the end, their system manages to provide effective and efficient operation that consumes less energy without disturbing the rapid reaction capability. In this study, the fire detection is done on a cluster-head level. In our approach, we suppose that every node in the WSN contains all the required functions. In this way, communication overhead between neighboring nodes is avoided and each sensor node can detect fire locally by itself. This allows to reduce the energy consumption and to improve the performance of the WSN.

5.3 A Data mining Approach for Intelligent Forest Fire Detection

In this section, we describe our proposition for fire detection based on WSN and Data mining techniques. We first introduce some assumptions and primitives and then identify the important design features that a wireless sensor network should possess, as well as the various Data mining techniques used in order to be able to successfully monitor a forest and to detect fires.

- *Early Detection and Accurate Localization:* An early detection and an accurate localization of forest fire are necessary for a rapid intervention of firefighting personnel at the correct place.
- *Energy efficiency:* The deployment of a WSN for fire detection should consume energy very efficiently because the replacement of batteries may be too costly, impractical or even not possible. The energy consumption should also be balanced among nodes in order to maximize the lifetime of the WSN.
- *DM techniques comparison:* The aim is to study and to compare some techniques of Data mining applied to detect fire in terms of precision, response time and energy consumption.

5.3.1 Assumptions and Primitives

In our proposed approach, we consider a WSN with one base station and hundreds of multi-sensor nodes. More precisely, there are n sensor nodes in the WSN, denoted by s_i, $1 \leqslant i \leqslant n$, and identified by a unique identifier id_i. We assume that any two sensor nodes can directly exchange messages if the Euclidean distance between them is not greater than their communication range R_c. Hence, the set of neighbor nodes $N(s_i)$ of a given node s_i can be defined as follows:

$N(s_i) = \{s_j : dist(s_i, s_j) <= R_c,\ \ j = 1, ..., n\ \ and\ \ j \neq i\}$.

Fires can differ in size and shape which can influence the possibility to detect them. Therefore, it is necessary to find the optimum size of the target area coverage by a single node. For simplicity, we assume that a planar area can be covered by sensor nodes if their Euclidean distance is not greater than the sensing range R_s. A forest fire f can be seen as a function $f(x_1, x_2, .., x_m)$, where the variables x_i represent m attributes such as temperature, humidity,... each of them being sensed by a sensor unit in the node. We also assume that e_i is the current remaining energy level of the node s_i and r_i^h is the risk level of season h, $1 \leqslant h \leqslant 4$, of the node s_i. Each cluster-head has its specific fire threshold $FT\{low, medium, high\}$ and a_i is the number of received alerts from each of its member nodes for the specified period.

5.3.2 Main Phases

In this subsection we describe the proposed architecture for forest fire detection. A large number of sensor nodes are randomly deployed in the forest. These sensor nodes are organized as clusters so that each node has a corresponding cluster-head. Each sensor node can measure environmental temperature, relative humidity, smoke and light. Consequently, the communication overhead between neighboring nodes is avoided and each sensor node can detect fire locally by itself. Each sensor node predicts the fire using a Data mining technique and sends the alert containing its id_i to the corresponding cluster-head. The cluster-head calculates the danger rate and sends the id_i and damage rate to the sink via the gateway node. The sink detects the location of fire using the stored coordinate that corresponds to the received id_i for possible actions, such as alerting local residents or fire-fighting personal, and stores the alert in the server for the sake of statistical analysis.

The proposed approach can be divided into three main phases : a clustered network architecture, route discovery to the sink, fire detection and routing alerts to the sink. These phases are the following:

1. Clustering: this phase consists in subdividing the set of sensor nodes into clusters in order to obtain more efficient network processing and data fusion. Cluster-heads can be used as the important points in the network to achieve data processing, and to provide cooperation and coordination. We will show that this network architecture is appropriate for both early fire detection and energy conservation.

2. Routing: this phase consists in building routing tables in cluster-heads and gateway nodes within the clustered network. The aim is to maximize the

lifetime, to ensure the efficient performance of the network and to route the alert from a fire detecting node to the sink as rapidly as possible.

3. Fire Detection: this phase consists in applying Data mining techniques at the member node level in order to detect fire. The Data mining technique is learned in offline mode in order to create a model from historical fire data. Using the model obtained from classifier learning, the fire will be detected in online mode.

5.3.3 The Techniques Used in Detail

In this section we explain in more detail the design of each of the phases described above.

Clustering Step

An efficient functioning of a WSN depends on the topology of the network. An architecture based on a clustered topology provides important advantages for forest fire detection. Hence, it is possible to benefit from rapid detection of fire danger, to maximize the lifetime of the network, to achieve connectivity and fault-tolerance.

To ensure the maximum lifetime of a WSN, it is necessary to perform a good energy management in order to cope with depletion of sensor nodes. The objective of connectivity is to guarantee that the most important nodes of the network can communicate with other nodes that are located in their clusters. We also grant particular attention to low computational complexity and high accuracy. These properties are achieved by Data mining techniques that efficiently detect the fire with as minimal computation as possible.

We have chosen a distributed clustering algorithm [161] which is based only on neighborhood information, as preferable for WSNs, and which can be described as follows :

1. Each node s_i broadcasts the information to its neighbors $N(s_i)$.

2. Each node decides to be cluster-head or not according to its local topology information.

3. The node selected as a cluster-head broadcasts its status to its neighbors and invites them to join his cluster.

4. If a node receives at least two messages to join two different clusters then it declares itself as a gateway, otherwise it declares itself as a member node.

The selection as cluster-head (CH) is based on weight (given by the residual energy e_i and a few parameters such as the node degree $|N(s_i)|$). The node having the highest weight within this neighborhood is declared as a cluster-head. The gateway nodes in the cluster are used to relay data among cluster-heads. The member nodes just treat the fire detection and send alerts to the corresponding cluster-head.

It is necessary to re-select a new cluster-head among nodes in order not to overload a few nodes with respect to others. There are several studies for cluster-head rotation in [161]. The best way is to use the remaining battery power for triggering the clustering algorithm at local regions. When the battery level of the cluster-head is below a specified threshold then it broadcasts a message to its neighbors to select a new cluster-head among them.

Routing Step

Several routing algorithms have been presented in the literature [162]. For our work, we adapt an algorithm based on the cluster network to maximize the lifetime, to provide a best performance of the network and to allow the routing of an alert from the node to the sink as rapidly as possible.

After applying the clustering algorithm, each node is declared as a cluster-head or as a gateway including a routing table. At the beginning, the routing table is empty. When the sink propagates the discovery route message which contains its id_i, the concerned gateways will receive the message and save the identifier of the sink in their routing table. Each gateway node of the sink forwards the discovery route message which contains their identifiers to the next cluster-heads except the sink. When the cluster-heads receive the discovery route message, they save the gateway identifier in the routing table in chronological order. In the same way, each cluster-head forwards the discovery message to the next gateway with the exception of the previous one. As soon as all cluster-heads and gateways have received a discovery message, they are ready to route the message to the sink. With this technique, the cluster-heads and gateways can use multiple paths to route messages to the sink in the network. These multi-path communications are aimed to improve the reliability, fault-tolerance and performance of the network. For that, the first recorded node is established as the active communication routing node, while the others are stored for future need, e.g., when the current active node is broken or fails. This also allows to use the other nodes for routing data.

DM Step

Our work is based on the measurement of real data by sensors (temperature, humidity, light and smoke) and a prediction of fire using classification techniques of Data mining at the member node level, discarding normal values and transmitting only abnormal values to the cluster-head. This process reduces the number of exchanged messages, removes redundancy, improves the system speed and decreases the potential network traffic, extends network lifetime and thus makes early fire detection possible. Also observe, that the rate of sensing data varies according to year seasons: The sensing rate is high in summer, average in spring and autumn and low in winter. In order to reduce sensing energy consumption, we use an intelligent method which is based on the risk level r_i^h of the node i. The node computes its r_i^h for each season h according to the number of fires detected in the season of the previous year. According to [144], in summer, between June 21, 2015 and August 21, 2015, there have been 956 forest fires with 743 fires between 7 am and 9 pm and 213 fires between 9 pm and 7 am. In this case, the rate for sensing data p_i for the next summer is computed as follows :

$$p_i = r_i^h / t \tag{5.1}$$

where t is the number of hours in a day when a fire is detected. In our example, 743 fires are detected on the 92 summer days in the period of 14 hours a day. The rate for the sensing of node i in this period is : $p_i = (743/92)/14 \simeq 0.58$ fire/h or 1 fire in 1 hour and 45 minutes. Therefore, the sensor node declared as member turns periodically on and off its radio and its multiple sensors according to p_i. On the other hand, the node can sense within the determinate period of 20 minutes every $1.25h$.

There are many classification/predictive methods and in this study, we will focus on three particular ones.

Naïve Bayes Classifier This method uses Bayesian statistics and Bayes' theorem to find the probability of each instance to belong to a specific class. The training data contain attributes x_i and are split into two classes C_k (Fire, Non), $k \in \{1, 2\}$. The learning of Gaussian naïve Bayes algorithm relies on the computation of the mean μ_k and the variance σ_k^2 of each attribute x_i in each class C_k. To find the probability of a new sensing instance $I(x_1, x_2, ..., x_m)$ to belong to a specific class C_k, the following formula is applied :

$$P(C_k|x_1, x_2, ..., x_m) = \frac{P(C_k)\prod_{i=1}^{m} P(x_i/C_k)}{evidence} \tag{5.2}$$

where

$$evidence = \prod_{i=1}^{m} P(x_i/Fire) + \prod_{i=1}^{m} P(x_i/Non) \tag{5.3}$$

and

$$P(x_i/C_k) = \frac{1}{\sqrt{2\pi\sigma_{C_k}^2}} e^{-\frac{(x_i-\mu_{C_k})^2}{2\sigma_{C_k}^2}} \tag{5.4}$$

Consequently, a fire is detected if the probability of the $Fire$ class $P(Fire|x_1, x_2, ..., x_m)$ is greater than the probability of the $NonFire$ class $P(Non|x_1, x_2, ..., x_m)$. In case, that these two probabilities are equal, we cannot distinguish the presence or absence of fire. The solution is to launch another classifier in order to detect the fire.

Artificial Neural Network Classifier (ANN) We use the multi-layer back-propagation artificial neural network. The algorithm is learned by adjusting the weights W_{ij} of the ANN for each input training data $X_l(x_1, x_2, ..., x_m)$. This adjustment is performed in order to minimize the error E_l between the expected output Y_l and the obtained output $Y_l^{'}$ of the ANN, for $1 \leqslant l \leqslant r$, where r is the number of instances in the training data. Since the output of ANN is a Boolean value ($Fire = 1$, $Non = 0$), we choose the log-sigmoid function $f(I)$ given by formula (5.5) as the transfer function.

$$f(I) = \frac{1}{(1 + e^{-I})} \tag{5.5}$$

A method for measuring the discrepancy between the expected output Y_l and the obtained output $Y_l^{'}$ is to use the absolute error measure as follows:

$$E_l = \frac{1}{n} \sum_{l=1}^{n} |Y_l^{'} - Y_l| \tag{5.6}$$

After the learning, ANN constructs a mathematical relationship between the sensing data and the correct class ($Fire$ or $NonFire$). Then, the ANN can take an accurate decision.

Figure 5.1 illustrates the neural network method used to detect fire. There are four neurons in the first layer, as the forest fire is represented by multiple attributes which are sensed by different types of sensing units (temperature, humidity, light, smoke).

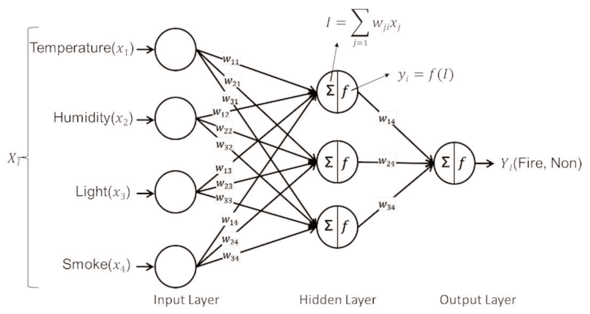


Figure 5.1: Fire detection using a neural network.

K-nearest neighbors Classifier The aim of this method is to classify a new event according to its similarity with the examples of the learning data. Let D be the set of learning data, d a distance function and k a small positive integer. For any new sensing instance $I(x_1, x_2, ..., x_m)$, the classifier has to take a decision. In D, the algorithm searches the k nearest examples of I using the distance d, and assigns I to the class $C(Fire, NonFire)$ that is the most frequent among these k neighbors. There are various methods to compute the distance between learning data and new sensing data such as the Euclidean, Manhattan and Minkowski distance. For our case, the Euclidean distance will be appropriate.

In our work, the fire detection will be categorized into two phases, offline and online as shown in Figure 5.2. The offline process produces predefined patterns (the model) from the forest environment for the two situations that fire is present or not, using classification techniques from Data mining and learning from historical data. The model obtained from learning needs to be known before the detection. This phase cannot operate online because wireless sensor nodes have resource limitations: energy, memory and computation. The model as the output from this process is stored in member sensor nodes. The second phase (online process) consists in finding the correspondence between the predefined model from previous processes and sensor reading instances. This process provides a fast detection and reduces the response time. The output from this process gives the possibility to detect fire or not.

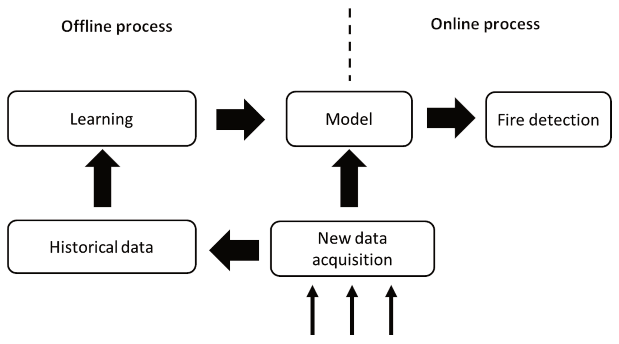


Figure 5.2: The phase of fire detection.

The main steps of the proposed approach, can be described by the following algorithm:

Algorithm 6 Proposed approach

```
1: **Input** Create a local model in each sensor
2: Sense environment according to $p_i$
3: Classify each new reading according to the local model
4: if member node && fire is detected then
5:     Send alert to cluster-head
6: end if
7: if cluster-head && receive alerts then
8:     Compute damage rate and combine the predictions from the sensors.
9:     Send alert to gateway using routing table
10: end if
11: if gateway && receive alerts then
12:     Route alert to other cluster-head using routing table
13: end if
14: if base-station && receive alerts then
15:     Localize fire
16:     Inform fire-fighting personal
17: end if
```

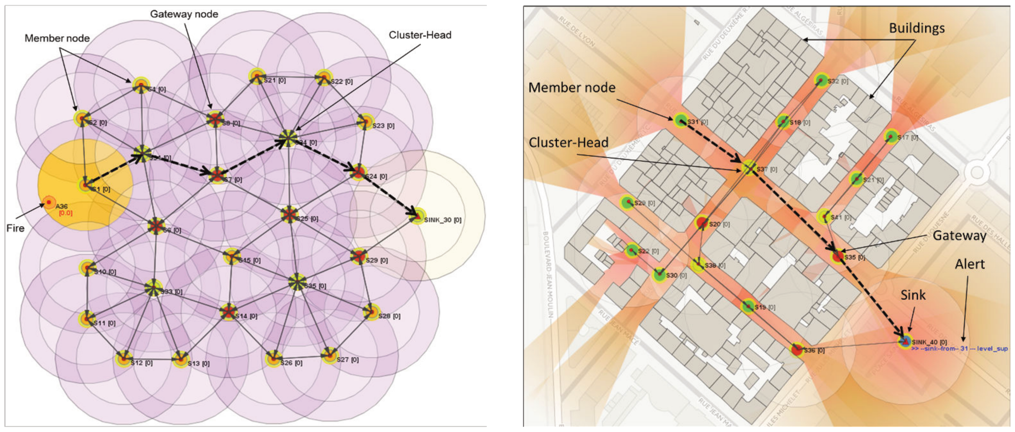
According to the algorithm 6, when a fire is detected by member nodes, they send the alert message to their respective cluster-head. This will reduce the processing cost of all data by the cluster-head and also reduce the communication between the member nodes and their cluster-head. Therefore, the energy consumption is reduced. In addition, the cluster-heads can apply smart scheduling and adaptive transmissions to reduce the overhead on most sensor nodes near the sink. When cluster-heads receive an alert from their members, they compute the number of the received alerts a_i from each of their members and they use a fire threshold FT to determine the current risk level of fire. Then a cluster-head will send one alert to the sink via its gateway using its routing table.

The risk level of fire is determined by comparing a_i with FT. The cluster-head sends one alert to the sink containing the id_i of the corresponding node and the risk level of fire (low, medium, high). Note, that a_i is re-initialized whenever the member nodes are in sleeping mode. This way, the energy consumption is further reduced.

5.4 Results

To evaluate our proposed approach, we have implemented and performed extensive simulation experiments using the CupCarbon simulator [18]. In this section, we present our results and discussions.

Figure 5.3 shows an example of detecting fire with our approach using the CupCarbon simulator.



(a) (b)

Figure 5.3: CupCarbon Simulator for fire detection.

Figure 5.3(a) shows an example of forest fire detection. In this figure, we notice that the member node s_1 detects fire and sends the alert to its cluster-head s_{32}, which itself sends the alert to the gateway s_7, that corresponds to the first node recorded in its routing table. In the same way, the gateway s_7 forwards the alert to the next cluster-head s_{34}, from s_{34} the alert is sent to s_{24} and finally, the gateway s_{24} forwards the alert to the sink.

Figure 5.3(b) shows an example of fire detection in a city. In this case, some constraints have been imposed by the nature of WSN in terms of radio aspects. This figure shows the area of visibility for each sensor node (the orange area around each node). The possible connections are represented by the arrows between the nodes. It turns out that there exist several possible connections for some nodes.

In order to evaluate the performance of our approach for forest fire detection, the nodes are deployed in the plane region representing a forest. The maximum communication range R_c of each node is set to $100m$. Each sensor node is equipped with battery and multi-sensor devices, which are used to collect data such as temperature, humidity, light and smoke, i.e., TMP36, 808H5V5, GL5537 IDR and MQ-135, respectively. The MAC protocol used in the simulation is 802.15.4 as implemented in the CupCarbon simulator.

To estimate the energy consumption of the proposed approach, we compute the energy consumption in transmission/reception, sensing and computation. First, to estimate the transmission/reception energy consumption, we use the energy model of the TelosB sensor node. Its energy consumption is estimated as $59,2\mu J$ to transmit one byte and as $28,6\mu J$ to receive one byte [107]. We have used the Super Alkaline AALR6 battery which is a portable energy source with a capacity of 9580 Joules. Second, to estimate the sensing energy consumption, we use the data given in the following Table [143].

Table 5.1: Sensing energy of sensors

Type of sensor	**Energy consumption**
Temperature	270 μJ
Humidity	72 μJ
Light	0.123 μJ
Smoke	225 μJ

Finally, to estimate the computational energy, we use the energy model of the TelosB sensor node. The energy consumed in computing 1 time clock at 4 mhz is 1.2 nJ on the TelosB [163].

The details of the general simulation parameters are depicted in Table 5.2.

Figure 5.4 shows the comparison between the energy consumption with our proposition, which respects the environment conditions, and simple sensing. The energy consumption with simple sensing remains at similar levels because the rate of sensing is fixed to one threshold throughout the year. With our proposition, however, the energy consumption changes depending on the season because our approach adapts the rate of sensing to the history of the number of fires detected in each season of the last year. We note that energy consumption in summer is

Table 5.2: Simulation parameters

Parameters	**Values**
Transmitter range	100 m
Number of nodes	100
Environment size	500 m. x 500 m.
MAC protocol type	802.15.4
Card type	Arduino UNO
Temperature sensor	TMP36
Humidity sensor	808H5V5
Light sensor	GL5537 IDR
Smoke sensor	MQ-135
Energy model	TelosB
Battery type	Super Alkaline AALR6
Battery capacity	9580 J
Energy transmission	59,2 μJ/byte
Energy reception	28,6 μJ/byte
Energy processing	1.2 nJ/1 time clock

higher using our approach, which however detects quickly and efficiently a fire compared to the simple detection technique, due to the adapted rate of sensing.

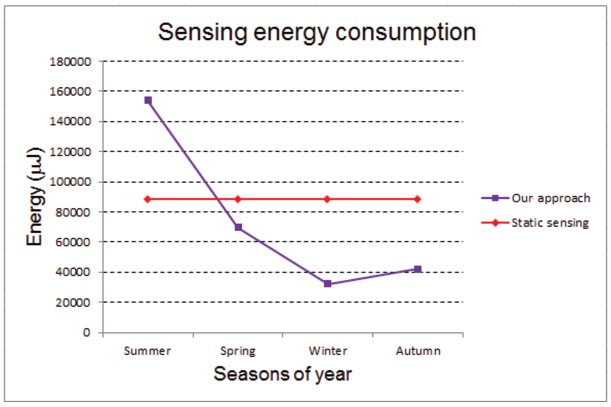


Figure 5.4: Sensing energy consumption (μJ)

Figure 5.5 shows the energy consumption in terms of our clustered and the non-clustered topology. As can be seen in this figure, clustering reduces the energy consumption compared to the non-clustered topology. This consumption depends on the communication rate in the network, and particularly on the number of alerts. In our approach and upon receipt of many alerts from a same node in short time, the cluster-head sends just one alert to the sink. Therefore, the energy consumed by our approach is less than that with non-clustered topology.

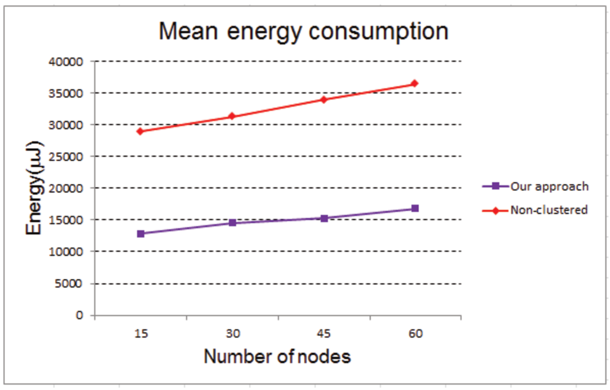


Figure 5.5: Comparison with non-clustered solution in terms of energy consumption (μJ)

Figure 5.6 shows a comparison between the energy consumption of fire detection at the cluster-head level and at the member node level in terms of node numbers in the same cluster. In order to make this comparison, we have fixed the cluster numbers to 2 and we have varied the number of nodes in the cluster. Each node senses 1000 instances containing 4 attributes (temperature, humidity, smoke, light) and we assume that 18% of the instances represent fire. We also use the Naïve Bayes classifier as a technique for fire detection. Figure 5.6 also shows that the energy consumed in computing the data at the cluster-head level is higher than the energy consumed at the member node level in terms of number of nodes in the same cluster. When the nodes send all data to the cluster-head it is difficult to synchronize the reception of data from a high number of member nodes. Also observe, that a high traffic in the network may increase the time of response of fire detection and lead to a rapid decrease of the cluster-head's energy. Therefore, the solution is to make fire detection at the member node level which is achieved in our approach.

Figure 5.7 shows the time taken to detect fire and to route an alert from the corresponding node to the sink, in terms of the number of clusters. We performed experiments with up to 10 clusters. We observe, that as the number of clusters in the network increases, our approach provides an improvement in performance.

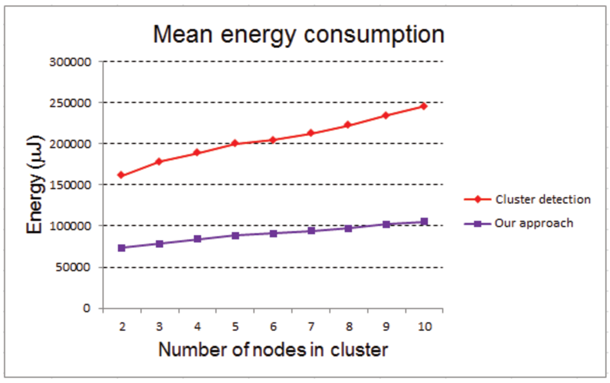


Figure 5.6: Comparison with cluster detection in terms of energy consumption(μJ)

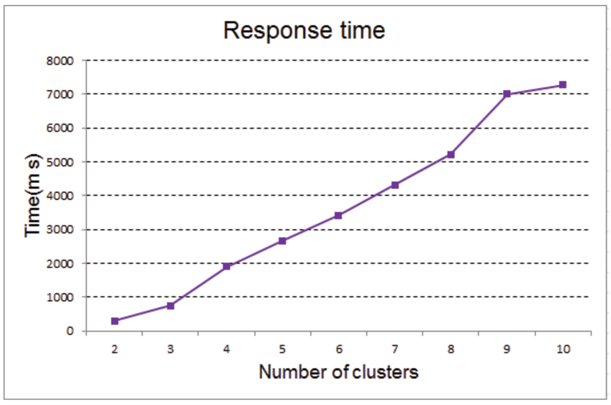


Figure 5.7: Average response time (milli-seconds)

Figure 5.8 shows a comparison between our approach and the paper [149] in terms of the time taken to detect fire and to route an alert from the corresponding node to the sink. We observe, that our approach provides a clear improvement in performance.

In this simulation, we performed experiments with 5500 instances including temperature, humidity, smoke and light of data and among which 2400 instances represent fire. After the simulation, the Artificial Neural Network classifies 2120 instances as fire, containing the same 1980 instances representing fire (true positives) at the beginning and in which 140 instances are classified as fire without representing it (false positives). The Naïve Bayes classifies 2400 instances as fire which represent real fire (true positives), and 570 instances as fire which do not represent fire at the beginning (false positives). The KNN classifies 1930 instances

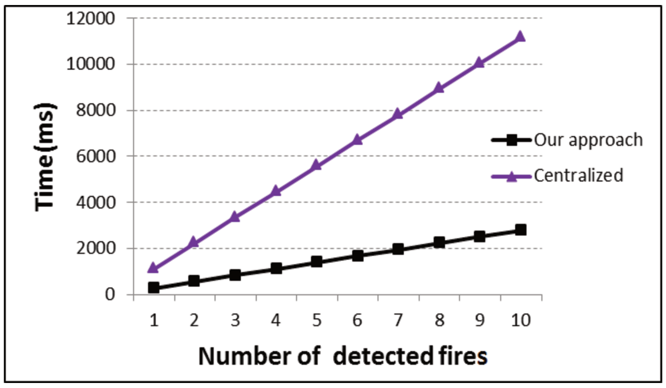


Figure 5.8: Average detection and routing time (milli-seconds)

as fire which represent real fire (true positives), and 70 instances as fire which do not represent fire at the beginning (false positives). Table 5.3 shows the results obtained after the comparison of the above classifiers applied to detect fire in terms of : precision, recall, energy consumption and response time. The precision P and the recall R of the Data mining techniques are measured by the following formulas [126]:

$$P = \frac{TP}{TP + FP} \tag{5.7}$$

$$R = \frac{TP}{TP + FN} \tag{5.8}$$

where TP, FP and FN are the numbers of true positives, false positives and false negatives, respectively.

Table 5.3: Data mining techniques comparison

Technique	NB	ANN	KNN
Energy (nJ)	222,4	290,8	70323,52
Time (S)	0,000972	0,00127	0,307358
Precision	0.808	0.934	0.965
Recall	1	0.825	0.804

As shown in Table 5.3, the Naïve Bayes (NB) classifier provides higher classification accuracy, detects fire rapidly in 0.000972 seconds and consumes less energy in the computational task to detect fire. This energy takes the value of 2,22e-7 joule which is 2,32e-09% from the total battery capacity.

According to the result obtained in Table 5.3, we notice that the NB classifier achieves 80.8% of precision and detects 100% of fire which is shown by the value of recall but there are some false alerts. It is clearly better to have a false alert rather than not to detect a true fire.

Table 5.3 also shows that both Artificial Neural Network (ANN) and Naïve Bayes classifiers (NB) have low computational complexity which explains the rapidity of fire detection. Moreover, they consume less energy than K-Nearest Neighbors (KNN) which needs to compute the distance between the new sensing instance and each instance in the training data and to chose the K smallest distances which causes the high computational effort and the high energy consumption. It can be seen that the ANN classifier shows better prediction on larger training data sets than KNN while in case of small data sets, KNN gives quite good results and provides higher classification accuracy than NB. We also notice that ANN and KNN provide higher classification accuracy than NB because NB has one false alert. In case of fire detection and for precautionary measures, we can tolerate the sending of a false alert by NB. In other applications, it is up to the user to choose the criteria for selecting the classifiers that are most appropriate for this application in terms of energy consumption, precision and time of response.

5.5 Conclusions

In this work, a new approach for predicting forest fire has been proposed, which is based on Data mining techniques and wireless sensor networks. Our approach takes into account all characteristics of a WSN that regards low energy capacity, computing limitation, low memory capacity of sensor nodes, and environmental conditions which can affect fire detection and the performance of a WSN. Our work is based on measuring and combining real data from different sensors (temperature, humidity, light and smoke) and on selecting the best classifier applied to data for fire detection. A node detects fire locally by itself, then it discards normal values and transmits only abnormal values to the sink for fire localization and information to the firefighters. The obtained results show that applying different

Data mining techniques can reduce the size of data, delete redundancy, improve the WSN speed and decrease the network traffic which extends the lifetime of the network and guarantees a short time of decision for fire detection as early as possible. In the next chapter, we will conclude our report of the thesis with a conclusion, a discussion, and some perspectives.

CHAPTER

SIX

CONCLUSIONS

In recent years, the field of Data mining techniques in WSNs has seen considerable research interest. The development of resource efficient WSNs has been the central focus of most research in this field. The advancement of WSNs has allowed to deploy wireless sensors to address many distributed monitoring application challenges. As a result, WSN applications are increasingly generating a continued and large volume of data which should be processed efficiently. However, WSNs raise difficult issues and challenges for the research community, since application requirements such as accuracy, lifetime, latency, are very often in conflict with the limited computational, memory, communication and energy resources of the network. This thesis has investigated how Data mining techniques could be used to address some of these issues, and focused in particular on the tradeoff between data accuracy, energy efficiency, and network load in data collection tasks. It also presents some of the challenging areas of WSNs and distributed Data mining techniques. This chapter summarizes the main results of the thesis, and opens the topic to further research directions.

6.1 Summary

This research has investigated the possibilities of employing distributed Data mining techniques for WSNs to extract useful information and to help in decision making. The aim is to use the individual data processing capabilities of the sensor nodes by implementing distributed algorithms and distributed Data mining approaches in order to make efficient WSN applications.

In Chapter 2, we have presented a general view of the existing Data mining techniques in the context of WSNs. We have presented the architecture of WSNs. In addition, we have described the hardware platforms, operating systems, wireless advantages, challenges and applications of WSNs. As the energy is a major limiting factor in a WSN, we have described some strategies for its conservation that can make a WSN efficient. Following the WSN description, Data mining techniques, specifically those relevant to the WSN field are also reviewed. Finally, we have analyzed and presented the different classifications of the existing methods with a focus on distributed Data mining techniques related to our research area.

In Chapter 3, we have resolved the problem of boundary point detection which can help the distributed Data mining techniques reduce a huge volume of data generated by sensor nodes, and improve the WSN operations. We have divided this chapter into two main sections. In the first section, we have proposed the LPCN algorithm allowing to find the boundary points for a set of data points. We have compared this algorithm with the existing algorithms in the literature. This algorithm can be useful in many real applications, especially Data mining applied for WSN applications, since it can be used to detect the shape of the extracting clusters in a set of sensed data, to detect anomalies, classification, and medical imaging. Finally, we have presented the implementation and the results obtained from the CupCarbon simulator. In the second section, we have proposed the D-LPCN algorithm for finding a minimal set of connected boundary nodes in a WSN. The main advantages of this algorithm when compared with the existing algorithms are that it works with any type of network, planar or non-planar, relies exclusively on the nodes of the boundary and their neighbors, works with any topology even disconnected (i.e., with many connected sub-networks) and with any density of the network. In the case of a disconnected network, it can determine the boundary nodes of each connected component. Furthermore, it takes into account any blocking situation, which it is able to overcome. Its complexity is reduced when compared with the LPCN algorithm by the mean of the distribution method. The proposed algorithm is validated using the CupCarbon, Tossim and Contiki simulators. It has also been implemented using real sensor nodes based on the TelosB and Arduino/XBee platforms. We have estimated the energy consumption of each node and we have found that the consumption of the network depends on the number of the boundary nodes and their neighbors. The simulation results show that the proposed algorithm is less energy consuming than the existing algorithms and its distributed version is less energy consuming than the centralized version. The simulation results also indicate that the proposed algorithm can be applied to very large sensor networks and it is quite optimal in terms of energy consumption.

In Chapter 4, we have proposed a new distributed data clustering approach (DCCA) for a hierarchical wireless sensor network topology for addressing WSN challenges, aggregating spacial and huge datasets, and reducing the amount of data in the network data. Our approach is focused on the processing power of the unsupervised, online, distributed methods by maximizing parallelism and minimizing communications and to reduce the size of the data to be exchanged among the nodes of the network. The proposed approach organizes the network in a tree topology using the modified BSF algorithm and based on the received signal strength indication estimates the best neighbors and the residual energy of nodes in order to build an efficient network with maximum lifetime. A hierarchical agglomerative clustering algorithm together with the LPCN algorithm remain embedded within the individual nodes to analyze the locally generated data by its heterogeneous sensors in order to compute the local models. This phase is computed in parallel by all the leaf nodes. The local results represent the contours of clusters and are transmitted to the cluster-heads in the tree topology which merge them to build the global clusters. This process is repeated at each level of the tree until to reach the base station which creates the global model of the network. The proposed approach is validated using the Contiki simulator and real sensor nodes based on the TelosB platform. Experimental results indicate that the local models which represent only the contours of clusters are small size datasets representing 0.5% to 5% of the total size of the original dataset, which are exchanged through the network and allow to save energy and maximize the lifetime of the network. Experimental results have also highlighted the capability of the proposed approach in the WSN to produce very good clustering results on the distributed, large, and heterogeneous data as well as to greatly minimize the quantity of data that has to be transmitted by using the LPCN algorithm for which it is quite efficient in terms of energy consumption. In contrast to the other clustering algorithms, the AGNES algorithm used does not require prior knowledge of the number of clusters within the dataset. Then our results show the effectiveness of our approach in terms of the quantity of clusters generated and the computational time. Finally, our approach clearly outperforms the existing methods with respect to the network lifetime and the size of the data to be transmitted.

In Chapter 5, a new event detection approach was proposed for predicting forest fire, which is based on machine learning techniques and wireless sensor networks. As the approach is based on sensed data, and tries to learn the relationship between sensor data and event detection capability, different types of techniques available to process datasets using machine learning (classification). Here, modeling of sensors in WSN is done with features/attributes of a dataset, the output classes or variables as the application events, and sensor measurements modeled with instances of dataset. An extensive experimental evaluation with several collected datasets show, that proposed the proposed approach allows learning from historical data, and meets the operational/functional WSN challenges such as energy efficiency, and event detection (prediction accuracy). As reiterated before, due to resource constraints on WSNs and its sensor nodes, there is a need to select the best classifier to be implemented for the data of sensor nodes in a distributed way for fire detection. A node detects fire locally by itself, then it discards normal values and transmits only abnormal values to the sink for fire localization and information to the firefighters. Applying Data mining techniques reduces the size of data, deletes redundancy, improves the WSN speed and decreases the network traffic to extend the lifetime of the network to allow for a short time of decision and fire detection as early as possible.

In Chapter 6, we present a synthesis of the realized work and the results obtained during this thesis, and we conclude with some perspectives for our future work.

6.2 Future work

While this study delivered promising results within some particular goals, further study should explore other ways to extend performance, scope, and application of this research. The following is a list of suggestions for future work in this research:

Combination of Data mining techniques: This study explored two of the most prominent Data mining techniques, namely, clustering and classification. There are several techniques discussed in the literature such as association rule, prediction, and more machine learning which are adopted for WSN requirements. Combining more than one Data mining technique at different levels and exploring their potential in a distributed way is recommended.

Adaptation to mobile nodes: This study considers stationary WSN nodes to perform the distributed Data mining methods. With the advancement of WSN applications such as the sensor nodes used for monitoring animal movement, the use of mobile networks will be necessary. Therefore, we recommend to extend the proposed approaches to mobile nodes.

Another promising direction for further extension is an improvement of the proposed solutions. In the LPCN algorithm, we could consider its use with many other data clustering algorithm in order to detect the best solution for reducing the data size. In the D-LPCN algorithm, we are looking to improve our method to find the starting node respecting a low energy and a low complexity. In addition, we want to use this algorithm in other WSN applications to detect holes in a WSN, to route messages efficiently using only the boundary nodes, and to detect intrusion by using Data mining techniques in the boundary nodes. In our proposed data aggregation approach (DCCA), we intend to explore the various local data clustering techniques in order to guarantee an efficient clustering of large and distributed datasets. In addition, we want to use a prefix filtering algorithm to eliminate outliers and missing values in oder to improve the quality of the results (local and global models).

References

REFERENCES

[1] A. Mahmood, K. Shi, S. Khatoon, and M. Xiao, "Data mining techniques for wireless sensor networks: A survey," *International Journal of Distributed Sensor Networks*, vol. 2013, 2013. xii, 28, 30, 31, 70

[2] S. V. Stanković, G. Rakočević, N. Kojić, and D. Milićev, "A classification and comparison of data mining algorithms for wireless sensor networks," in *2012 IEEE International Conference on Industrial Technology*, March 2012, pp. 265–270. xii, 31, 32, 33

[3] P. Sharma, "Wireless sensor networks for environmental monitoring," *International Journal of Scientific Research Engineering and Technology*, pp. 2278–0882, 2014. 1

[4] L. Sitanayah, A. Datta, and R. Cardell-Oliver, "Heuristic algorithm for finding boundary cycles in location-free low density wireless sensor networks," *Computer Networks*, vol. 54, no. 10, pp. 1630–1645, 2010. 1, 45, 46

[5] R. Jafari, A. Encarnacao, A. Zahoory, F. Dabiri, H. Noshadi, and M. Sarrafzadeh, "Wireless sensor networks for health monitoring," in *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on.* IEEE, 2005, pp. 479–481. 1

[6] M. P. DJurivsić, Z. Tafa, G. Dimić, and V. Milutinović, “A survey of military applications of wireless sensor networks,” in *Embedded Computing (MECO), 2012 Mediterranean Conference on.* IEEE, 2012, pp. 196–199. 1

[7] S. K. Gupta and P. Sinha, “Overview of wireless sensor networks: A survey,” *Telos*, vol. 3, no. 15μW, p. 38mW, 2014. 12, 13, 16, 18

[8] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, “Wireless sensor networks: a survey on recent developments and potential synergies,” *The Journal of supercomputing*, vol. 68, no. 1, pp. 1–48, 2014. 13, 14

[9] J. A. Gutierrez, “Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (lr-wpans),” *IEEE Standard for Information Technology*, vol. 802, no. 4, 2003. 14

[10] Z. Alliance, “Zigbee 2007 specification,” *Online: http://www. zigbee. org/Specifications/ZigBee/Overview. aspx*, vol. 45, p. 120, 2007. 14, 15

[11] J. Yick, B. Mukherjee, and D. Ghosal, “Wireless sensor network survey,” *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008. 14

[12] S. Chatterjea, P. Havinga, and S. Dulman, “Introduction to wireless sensor networks,” in *Embedded Systems Handbook.* CRC Press, 2005, pp. 31–1. 14

[13] K. Mehdi, M. Lounis, A. Bounceur, and T. Kechadi, “Cupcarbon: A multi-agent and discrete event wireless sensor network design and simulation tool,” *In IEEE 7th International Conference on Simulation Tools and Techniques (SIMUTools’14), Lisbon, Portugal*, March 17-19 2014. 15, 54

[14] P. Levis, N. Lee, M. Welsh, and D. Culler, “Tossim: accurate and scalable simulation of entire tinyos applications,” *in Proc. of the 1st int. conf. on embedded networked sensor systems, ser. SenSys 03. New York, NY, USA: ACM*, pp. 126–137, 2003. 15, 54

[15] A. Dunkels, B. Gronvall, and T. Voigt, “Contiki-a lightweight and flexible operating system for tiny networked sensors,” in *Local Computer Networks,*

2004. 29th Annual IEEE International Conference on. IEEE, 2004, pp. 455–462. 15, 54, 58, 86

[16] T. Issariyakul and E. Hossain, *Introduction to network simulator NS2.* Springer Science & Business Media, 2011. 15

[17] G. Fortino, R. Greco, and A. Guerrieri, "Modeling and evaluation of the building management framework based on the castalia wsn simulator," in *Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on.* IEEE, 2013, pp. 668–674. 15

[18] A. P. PERSEPTEUR. (2015) Cupcarbon simulator. [Online]. Available: http://www.cupcarbon.com 15, 110

[19] M. Lounis, K. Mehdi, and A. Bounceur, "A cupcarbon tool for simulating destructive insect movements," *1st IEEE International Conference on Information and Communication Technologies for Disaster Management (ICT-DM'14), Algiers, Algeria*, March 24-25 2014. 15

[20] *Wireless Sensor Networks, Product Reference Guide.* Crossbow, 2007. 15

[21] S. D. Patil and B. Vijayakumar, "Overview of issues and challenges in wireless sensor networks," *Target*, vol. 5, no. 5, 2016. 16, 21

[22] S. D. Indu, "Wireless sensor networks: Issues & challenges," *International Journal of Computer Science and Mobile Computing (IJCSMC)*, vol. 3, pp. 681–85, 2014. 16

[23] K. Gupta and V. Sikka, "Design issues and challenges in wireless sensor networks," *International Journal of Computer Applications*, vol. 112, no. 4, 2015. 19, 20

[24] J. A. Stankovic, A. D. Wood, and T. He, "Realistic applications for wireless sensor networks," in *Theoretical aspects of distributed computing in sensor networks.* Springer, 2011, pp. 835–863. 20

[25] T. Das and S. Roy, "Coordination based motion control in mobile wireless sensor network," in *Electronic Systems, Signal Processing and Computing Technologies (ICESC), 2014 International Conference on*. IEEE, 2014, pp. 231–236. 21

[26] G. Piateski and W. Frawley, *Knowledge discovery in databases*. MIT press, 1991. 22

[27] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI magazine*, vol. 17, no. 3, p. 37, 1996. 22

[28] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011. 22, 25, 26, 27

[29] M. H. Dunham, *Data mining: Introductory and advanced topics*. Pearson Education India, 2006. 24, 26

[30] S. Wu and D. Clements-Croome, "Understanding the indoor environment through mining sensory data—a case study," *Energy and Buildings*, vol. 39, no. 11, pp. 1183–1191, 2007. 24, 26

[31] G. K. Sorot, "Data mining techniques and research challenges and issues." 25

[32] R. M. Balabin, R. Z. Safieva, and E. I. Lomakina, "Gasoline classification using near infrared (nir) spectroscopy data: comparison of multivariate techniques," *Analytica Chimica Acta*, vol. 671, no. 1, pp. 27–35, 2010. 25

[33] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping multidimensional data*. Springer, 2006, pp. 25–71. 26

[34] F. Murtagh, "A survey of recent advances in hierarchical clustering algorithms," *The Computer Journal*, vol. 26, no. 4, pp. 354–359, 1983. 26

[35] R. Chauhan, H. Kaur, and M. A. Alam, "Data clustering method for discovering clusters in spatial cancer databases," *International Journal of Computer Applications (0975–8887) Volume*, 2010. 26

[36] R. Agrawal, R. Srikant *et al.*, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, 1994, pp. 487–499. 27

[37] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *ACM Sigmod Record*, vol. 29, no. 2. ACM, 2000, pp. 1–12. 27

[38] M. J. Zaki and C.-J. Hsiao, "Charm: An efficient algorithm for closed itemset mining," in *Proceedings of the 2002 SIAM international conference on data mining*. SIAM, 2002, pp. 457–473. 27

[39] V. Cantoni, L. Lombardi, and P. Lombardi, "Challenges for data mining in distributed sensor networks," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 1. IEEE, 2006, pp. 1000–1007. 29

[40] X. Cheng, J. Xu, J. Pei, and J. Liu, "Hierarchical distributed data classification in wireless sensor networks," *Computer Communications*, vol. 33, no. 12, pp. 1404–1413, 2010. 35

[41] W. Li, P. Yi, Y. Wu, L. Pan, and J. Li, "A new intrusion detection system based on knn classification algorithm in wireless sensor network," *Journal of Electrical and Computer Engineering*, vol. 2014, 2014. 35

[42] K. Flouri, B. Beferull-Lozano, and P. Tsakalides, "Distributed consensus algorithms for svm training in wireless sensor networks," in *Signal Processing Conference, 2008 16th European*. IEEE, 2008, pp. 1–5. 35

[43] N. A. Alrajeh, S. Khan, J. L. Mauri, and J. Loo, "Artificial neural network based detection of energy exhaustion attacks in wireless sensor networks capable of energy harvesting." *Ad Hoc & Sensor Wireless Networks*, vol. 22, no. 1-2, pp. 109–133, 2014. 35

[44] H. Ghasemzadeh, M. Rezaeian, F. D. Touranposhti, and M. M. Ghasemian, "Bn-leach: an improvement on leach protocol using bayesian networks for energy consumption reduction in wireless sensor networks," in *Telecommunications (IST), 2014 7th International Symposium on.* IEEE, 2014, pp. 1138–1143. 35

[45] T. Wang and Z. Yang, "A location-aware-based data clustering algorithm in wireless sensor networks," in *Communication Systems, 2008. ICCS 2008. 11th IEEE Singapore International Conference on.* IEEE, 2008, pp. 1–5. 36

[46] A. Taherkordi, R. Mohammadi, and F. Eliassen, "A communication-efficient distributed clustering algorithm for sensor networks," in *Advanced Information Networking and Applications-Workshops, 2008. AINAW 2008. 22nd International Conference on.* IEEE, 2008, pp. 634–638. 36, 72

[47] S. Yoon and C. Shahabi, "The clustered aggregation (cag) technique leveraging spatial and temporal correlations in wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 3, no. 1, p. 3, 2007. 36

[48] S. Nithyakalyani and S. S. Kumar, "Data relay clustering algorithm for wireless sensor networks: a data mining approach," *Journal of Computer Science*, vol. 8, no. 8, p. 1281, 2012. 36

[49] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *System sciences, 2000. Proceedings of the 33rd annual Hawaii international conference on.* IEEE, 2000, pp. 10–pp. 36, 70

[50] S. Lee and T. Chung, "Data aggregation for wireless sensor networks using self-organizing map," in *International Conference on AI, Simulation, and Planning in High Autonomy Systems.* Springer, 2004, pp. 508–517. 36

[51] C. Xia, W. Hsu, M.-L. Lee, and B. C. Ooi, "Border: efficient computation of boundary points," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 3, pp. 289–303, 2006. 38

[52] J. O. Cadenas, G. M. Megson, and C. L. L. Hendriks, "Preconditioning 2d integer data for fast convex hull computations," *PloS one*, vol. 11, no. 3, p. e0149860, 2016. 39

[53] J. Fan, J. Yang, M. Goyal, and Y. Wang, "Rigid registration of 3-d medical image using convex hull matching," in *Bioinformatics and Biomedicine (BIBM), 2013 IEEE International Conference on.* IEEE, 2013, pp. 338–341. 39

[54] M. Jayaram and H. Fleyeh, "Convex hulls in image processing: A scoping review," *American Journal of Intelligent Systems*, vol. 6, no. 2, pp. 48–58, 2016. 39

[55] Y. Li, S. Wang, Q. Tian, and X. Ding, "A survey of recent advances in visual feature detection," *Neurocomputing*, vol. 149, pp. 736–751, 2015. 39

[56] F. Sheeba, R. Thamburaj, J. J. Mammen, M. Kumar, and V. Rangslang, "Convex hull based detection of overlapping red blood cells in peripheral blood smear images," in *7th WACBE World Congress on Bioengineering 2015.* Springer, 2015, pp. 51–53. 39

[57] F. Lalem, A. Bounceur, M. Bezoui, M. Saoudi, R. Euler, T. Kechadi, and M. Sevaux, "Lpcn: Least polar-angle connected node algorithm to find a polygon hull in a connected euclidean graph," *Journal of Network and Computer Applications*, vol. 93, pp. 38 – 50, 2017. 39, 41

[58] M. Saoudi, F. Lalem, A. Bounceur, R. Euler, M.-T. Kechadi, A. Laouid, M. Bezoui, and M. Sevaux, "D-lpcn: A distributed least polar-angle connected node algorithm for finding the boundary of a wireless sensor network," *Ad Hoc Networks*, vol. 56, pp. 56 – 71, 2017. 39

[59] R. A. Jarvis, "On the identification of the convex hull of a finite set of points in the plane," *Information Processing Letters*, vol. 2, no. 1, pp. 18–21, 1973. 39, 40

[60] R. L. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," *Information Processing Letter*, vol. 1, no. 4, pp. 132–133, 1972. 40

[61] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software (TOMS)*, vol. 22, no. 4, pp. 469–483, 1996. 40

[62] M. Kallay, "The complexity of incremental convex hull algorithms in r^d," *Information Processing Letters*, vol. 19, no. 4, p. 197, 1984. 40

[63] A. J. Gomes, "A total order heuristic-based convex hull algorithm for points in the plane," *Computer-Aided Design*, vol. 70, pp. 153–160, 2016. 40

[64] C. Xing, Z. Xiong, Y. Zhang, X. Wu, J. Dan, and T. Zhang, "An efficient convex hull algorithm using affine transformation in planar point set," *Arabian Journal for Science and Engineering*, vol. 39, no. 11, pp. 7785–7793, 2014. 40

[65] G. Mei, "Cudachain: an alternative algorithm for finding 2d convex hulls on the gpu," *SpringerPlus*, pp. 1–26, 2016. 40

[66] A. Ruano, H. R. Khosravani, and P. M. Ferreira, "A randomized approximation convex hull algorithm for high dimensions," *IFAC-PapersOnLine*, vol. 48, no. 10, pp. 123–128, 2015. 40

[67] V. Skala, Z. Majdisova, and M. Smolik, "Space subdivision to speed-up convex hull construction in e3," *Advances in Engineering Software*, vol. 91, pp. 12–22, 2016. 40

[68] G. Garai and B. B. Chaudhuri, "A split and merge procedure for polygonal border detection of dot pattern," *Image and Vision Computing*, vol. 17, no. 1, pp. 75–82, 1999. 40

[69] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *Information Theory, IEEE Transactions on*, vol. 29, no. 4, pp. 551–559, 1983. 40

[70] A. R. Chaudhuri, B. B. Chaudhuri, and S. K. Parui, "A novel approach to computation of the shape of a dot pattern and extraction of its perceptual border," *Computer Vision and Image Understanding*, vol. 68, no. 3, pp. 257–275, 1997. 40

[71] A. Moreira and M. Y. Santos, "Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of points," 2007. 40

[72] J.-S. Park and S.-J. Oh, "A new concave hull algorithm and concaveness measure for n-dimensional datasets," *Journal of information science and engineering*, vol. 29, no. 2, pp. 379–392, 2013. 40

[73] C. Braune, M. Dankel, and R. Kruse, "Obtaining shape descriptors from a concave hull-based clustering algorithm," in *International Symposium on Intelligent Data Analysis*. Springer, 2016, pp. 61–72. 40

[74] A. Gheibi, M. Davoodi, A. Javad, F. Panahi, M. M. Aghdam, M. Asgaripour, and A. Mohades, "Polygonal shape reconstruction in the plane," *IET Computer Vision*, vol. 5, no. 2, pp. 97–106, March 2011. 40

[75] S. Methirumangalath, A. D. Parakkat, and R. Muthuganapathy, "A unified approach towards reconstruction of a planar point set," *Computers & Graphics*, vol. 51, pp. 90–97, 2015. 40

[76] E. Rosén, E. Jansson, and M. Brundin, "Implementation of a fast and efficient concave hull algorithm," University of Uppsala, Sweden, Tech. Rep., 2014. 40

[77] M. Körner, M. V. Krishna, H. Süße, W. Ortmann, and J. Denzler, "Regularized geometric hulls for bio-medical image segmentation," *The Annals of the BMVA,(4)*, pp. 1–12, 2015. 40

[78] K. Mehdi, M. Lounis, A. Bounceur, and T. Kechadi, "Cupcarbon: A multi-agent and discrete event wireless sensor network design and simulation tool," *In IEEE 7th International Conference on Simulation Tools and Techniques (SIMUTools'14), Lisbon, Portugal*, March 17-19 2014. 43

[79] J. Xu and H.-Y. Qian, "Localization of wireless sensor networks with a mobile beacon," *Information Technology Journal*, vol. 12, no. 11, p. 2251, 2013. 45

[80] M. B. Alrajeh, Nabil Ali Bashir, "Localization techniques in wireless sensor networks," *International Journal of Distributed Sensor Networks*, 2013. [Online]. Available: http://dx.doi.org/10.1155/2013/304628 45

[81] A. Laouid, H. Moumen, and K. Chait, "A distributed localization protocol for wireless sensor networks," in *Computer and Information Technology (WCCIT), 2013 World Congress on.* IEEE, 2013, pp. 1–5. 45

[82] T. Le Dinh, "Topological boundary detection in wireless sensor networks." *JIPS*, vol. 5, no. 3, pp. 145–150, 2009. 45, 46

[83] C. Zhang, Y. Zhang, and Y. Fang, "Detecting coverage boundary nodes in wireless sensor networks," in *2006 IEEE International Conference on Networking, Sensing and Control.* IEEE, 2006, pp. 868–873. 45, 47

[84] A. Kröller, S. P. Fekete, D. Pfisterer, and S. Fischer, "Deterministic boundary recognition and topology extraction for large sensor networks," in *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm.* Society for Industrial and Applied Mathematics, 2006, pp. 1000–1009. 45, 46

[85] S. Funke and C. Klein, "Hole detection or: how much geometry hides in connectivity?" in *Proceedings of the twenty-second annual symposium on Computational geometry.* ACM, 2006, pp. 377–385. 45

[86] K. Luthy, E. Grant, N. Deshpande, and T. C. Henderson, "Perimeter detection in wireless sensor networks," *Robotics and Autonomous Systems*, vol. 60, no. 2, pp. 266–277, 2012. 45

[87] P. K. Sahoo, K.-Y. Hsieh, and J.-P. Sheu, "Boundary node selection and target detection in wireless sensor network," in *IFIP International Conference on Wireless and Optical Communications Networks (WOCN'07).* IEEE, 2007, pp. 1–5. 45

[88] B. Huang, W. Wu, and T. Zhang, "An improved connectivity-based boundary detection algorithm in wireless sensor networks," in *In 38th IEEE Conference on Local Computer Networks (LCN).* IEEE, 2013, pp. 332–335. 45, 46

[89] S. Shukla and R. Misra, "Angle based double boundary detection in wireless sensor networks," *Journal of Networks*, vol. 9, no. 3, pp. 612–619, 2014. 45, 47

[90] C. Lara-Alvarez, J. J. Flores, and C.-C. Wang, "Detecting the boundary of sensor networks from limited cyclic information," *International Journal of Distributed Sensor Networks, Hindawi Publishing Corporation, Volume 2015, Article ID 401838*, 2015. 45

[91] P. K. Sahoo, J.-P. Sheu, and K.-Y. Hsieh, "Target tracking and boundary node selection algorithms of wireless sensor networks for internet services," *Information Sciences*, vol. 230, pp. 21–38, 2013. 45, 47, 62

[92] W.-C. Chu and K.-F. Ssu, "Location-free boundary detection in mobile wireless sensor networks with a distributed approach," *Computer Networks*, vol. 70, pp. 96–112, 2014. 45, 47

[93] ——, "Decentralized boundary detection without location information in wireless sensor networks," in *Wireless Communications and Networking Conference (WCNC), 2012 IEEE.* IEEE, 2012, pp. 1720–1724. 45

[94] S. Das, I. Banerjee, and T. Samanta, "Sensor localization and obstacle boundary detection algorithm in wsn," in *Advances in Computing and Communications (ICACC), 2013 Third International Conference on*. IEEE, 2013, pp. 412–415. 45

[95] L.-H. Zhao, W. Liu, H. Lei, R. Zhang, and Q. Tan, "The detection of boundary nodes and coverage holes in wireless sensor networks," *Journal of Mobile Information Systems, Volume 2016 (2016)*, vol. 2016, 2016. 45, 62

[96] N. Senouci, M. K. El Ouahed, and H. Haffaf, "Detecting boundary nodes in wsn," in *Proceedings of the International Conference on Wireless Networks (ICWN)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2014, p. 1. 45, 62

[97] I. M. Khan, N. Jabeur, and S. Zeadally, "Hop-based approach for holes and boundary detection in wireless sensor networks," *IET Wireless Sensor Systems*, vol. 2, no. 4, pp. 328–337, 2012. 45, 47

[98] A. M. Khedr, W. Osamy, and D. P. Agrawal, "Perimeter discovery in wireless sensor networks," *Journal of Parallel and Distributed Computing*, vol. 69, no. 11, pp. 922–929, 2009. 45, 47

[99] S. Kundu and N. Das, "Event boundary detection and gathering in wireless sensor networks," in *Applications and Innovations in Mobile Computing (AIMoC), 2015*. IEEE, 2015, pp. 62–67. 45, 47

[100] R. Jing, L. Kong, and L. Kong, "Boundary detection method for large-scale coverage holes in wireless sensor network based on minimum critical threshold constraint," *Journal of Sensors*, vol. 2014, pp. 1–13, 2014. 46

[101] B. Huang, W. Wu, G. Gao, and T. Zhang, "Recognizing boundaries in wireless sensor networks based on local connectivity information," *International Journal of Distributed Sensor Networks*, vol. 2014, 2014. 46

[102] A. Bounceur, R. Euler, A. Benzerbadj, F. Lalem, M. Saoudi, T. Kechadi, and M. Sevaux, "Finding a polygon hull in wireless sensor networks," in *European Conference on Operational Research, University of Strathclyde, Glasgow, UK.* Invited talk, EURO 2015, July 2015. 46, 63, 81

[103] N. Santoro, *Design and analysis of distributed algorithms.* John Wiley & Sons, 2007, vol. 56. 48

[104] N. A. Lynch, *Distributed algorithms.* Morgan Kaufmann, 1996. 48

[105] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes, "Powertrace: Network-level power profiling for low-power wireless networks," 2011. 54, 58

[106] J. Oxer and H. Blemings, *Practical Arduino: cool projects for open source hardware.* Apress, 2009. 57

[107] A. S. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on.* IEEE, 2005, pp. 324–328. 57, 111

[108] G. Mao, *Localization Algorithms and Strategies for Wireless Sensor Networks: Monitoring and Surveillance Techniques for Target Tracking.* IGI Global, 2009. 65

[109] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002. 69, 95

[110] Y. Ma, Y. Guo, X. Tian, and M. Ghanem, "Distributed clustering-based aggregation algorithm for spatial correlated sensor networks," *IEEE Sensors Journal*, vol. 11, no. 3, pp. 641–648, 2011. 69, 70

[111] R. Rajagopalan and P. K. Varshney, "Data-aggregation techniques in sensor networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 8, no. 4, pp. 48–63, Fourth 2006. 69

[112] A. Das, "A novel association rule mining mechanism in wireless sensor networks," in *Emerging Trends and Applications in Computer Science (NCETACS), 2012 3rd National Conference on.* IEEE, 2012, pp. 274–276. 70

[113] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 14, no. 2, pp. 70–87, April 2007. 70

[114] M. Saoudi, A. Bounceur, R. Euler, T. Kechadi *et al.*, "Intelligent data mining techniques for emergency detection in wireless sensor networks," in *2nd IEEE International Conference on Cloud and Big Data Computing (CBDCom 2016), Toulouse, France, 18-21 July.* IEEE, 2016. 70

[115] M. Saoudi, A. Bounceur, R. Euler, and T. Kechadi, "Data mining techniques applied to wireless sensor networks for early forest fire detection," in *Proceedings of the International Conference on Internet of things and Cloud Computing.* ACM, 2016, p. 71. 70, 96

[116] M. H. Yeo, M. S. Lee, S. J. Lee, and J. S. Yoo, "Data correlation-based clustering in sensor networks," in *Computer Science and its Applications, 2008. CSA'08. International Symposium on.* IEEE, 2008, pp. 332–337. 70, 71, 91

[117] L. Guo, C. Ai, X. Wang, Z. Cai, and Y. Li, "Real time clustering of sensory data in wireless sensor networks." in *IPCCC*, 2009, pp. 33–40. 70, 72

[118] O. Younis and S. Fahmy, "Heed: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on mobile computing*, vol. 3, no. 4, pp. 366–379, 2004. 70

[119] S. S. Satapathy and N. Sarma, "Treepsi: tree based energy efficient protocol for sensor information," in *Wireless and Optical Communications Networks, 2006 IFIP International Conference on.* IEEE, 2006, pp. 4–pp. 70

[120] M. Liu, J. Cao, G. Chen, and X. Wang, "An energy-aware routing protocol in wireless sensor networks," *Sensors*, vol. 9, no. 1, pp. 445–462, 2009. 70

[121] P. Rajeshwari, B. Shanthini, and M. Prince, "Hierarchical energy efficient clustering algorithm for wsn," *Middle East Journal of Scientific Research*, vol. 23, pp. 108–117, 2015. 70

[122] R. Masiero, G. Quer, D. Munaretto, M. Rossi, J. Widmer, and M. Zorzi, "Data acquisition through joint compressive sensing and principal component analysis," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE.* IEEE, 2009, pp. 1–6. 71

[123] M.-H. Li, C.-C. Lin, C.-C. Chuang, and R.-I. Chang, "Error-bounded data compression using data, temporal and spatial correlations in wireless sensor networks," in *Multimedia Information Networking and Security (MINES), 2010 International Conference on.* IEEE, 2010, pp. 111–115. 71, 91

[124] X. Xu, R. Ansari, and A. Khokhar, "Adaptive hierarchical data aggregation using compressive sensing (a-hdacs) for non-smooth data field," in *Communications (ICC), 2014 IEEE International Conference on.* IEEE, 2014, pp. 65–70. 71

[125] A. Morell, A. Correa, M. Barceló, and J. L. Vicario, "Data aggregation and principal component analysis in wsns," *IEEE Transactions on Wireless Communications*, vol. 15, no. 6, pp. 3908–3919, 2016. 71

[126] R. Houari, A. Bounceur, M.-T. Kechadi, A.-K. Tari, and R. Euler, "Dimensionality reduction in data mining: A copula approach," *Expert Systems with Applications*, 2016. 71, 115

[127] X. Ma, S. Li, Q. Luo, D. Yang, and S. Tang, "Distributed, hierarchical clustering and summarization in sensor networks," in *Advances in Data and Web Management.* Springer, 2007, pp. 168–175. 71, 72

[128] X.-L. Ma, H.-F. Hu, S.-F. Li, H.-M. Xiao, Q. Luo, D.-Q. Yang, and S.-W. Tang, "Dhc: Distributed, hierarchical clustering in sensor networks,"

Journal of Computer Science and Technology, vol. 26, no. 4, pp. 643–662, 2011. 71, 72

[129] H. Harb, A. Makhoul, D. Laiymani, O. Bazzi, and A. Jaber, "An analysis of variance-based methods for data aggregation in periodic sensor networks," in *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXII*. Springer, 2015, pp. 165–183. 72, 91

[130] P. Zou and Y. Liu, "A data-aggregation scheme for wsn based on optimal weight allocation." *JNW*, vol. 9, no. 1, pp. 100–107, 2014. 73

[131] M. Bendechache, M.-T. Kechadi, and N.-A. Le-Khac, "Efficient large scale clustering based on data partitioning," in *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 612–621. 73

[132] N.-A. Le-Khac, L. M. Aouad, and M.-T. Kechadi, "Knowledge map: Toward a new approach supporting the knowledge management in distributed data mining," in *Autonomic and Autonomous Systems, 2007. ICAS07. Third International Conference on*. IEEE, 2007, pp. 67–67. 73

[133] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344. 73, 82

[134] F. Lalem, A. Bounceur, M. Saoudi, M. Bezoui, R. Euler, and M. Sevaux, "Lpcn: Least polar-angle connected node algorithm to find the polygon hull in a connected Euclidean graph," (Submitted). 73, 81

[135] P. Willett, "Recent trends in hierarchic document clustering: a critical review," *Information Processing & Management*, vol. 24, no. 5, pp. 577–597, 1988. 74

[136] B. Awerbuch and R. Gallager, "A new distributed algorithm to find breadth first search trees," *IEEE Transactions on Information Theory*, vol. 33, no. 3, pp. 315–322, 1987. 75

[137] C.-H. Lung and C. Zhou, “Using hierarchical agglomerative clustering in wireless sensor networks: An energy-efficient and flexible approach,” *Ad Hoc Networks*, vol. 8, no. 3, pp. 328–344, 2010. 81

[138] A. Bouguettaya, Q. Yu, X. Liu, X. Zhou, and A. Song, “Efficient agglomerative hierarchical clustering,” *Expert Systems with Applications*, vol. 42, no. 5, pp. 2785–2797, 2015. 82

[139] F. Nielsen, “Hierarchical clustering,” in *Introduction to HPC with MPI for Data Science*. Springer, 2016, pp. 195–211. 82

[140] P. Azad and V. Sharma, “Pareto-optimal clustering scheme using data aggregation for wireless sensor networks,” *International Journal of Electronics*, vol. 102, no. 7, pp. 1165–1176, 2015. 82

[141] B.-C. Renner, *Sustained Operation of Sensor Nodes with Energy Harvesters and Supercapacitors*. BoD–Books on Demand, 2013. 86

[142] X. Liang and I. Balasingham, “Performance analysis of the ieee 802.15. 4 based ecg monitoring network,” in *Proceedings of the 7th IASTED International Conferences on Wireless and Optical Communications*, 2007, pp. 99–104. 87

[143] M. A. Razzaque and S. Dobson, “Energy-efficient sensing in wireless sensor networks using compressed sensing,” *Sensors*, vol. 14, no. 2, pp. 2822–2859, 2014. 87, 111

[144] PROMETHEE. (2015) Database on mediterranean forest fires in france. [Online]. Available: http://www.promethee.com/default/incendies 94, 105

[145] C. Hartung, R. Han, C. Seielstad, and S. Holbrook, “Firewxnet: A multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments,” in *Proceedings of the 4th international conference on Mobile systems, applications and services*. ACM, 2006, pp. 28–41. 95, 98

[146] J. Lloret, M. Garcia, D. Bri, and S. Sendra, "A wireless sensor network deployment for rural and forest fire detection and verification," *Sensors*, vol. 9, no. 11, pp. 8722–8747, 2009. 95, 97

[147] Y. E. Aslan, I. Korpeoglu, and Ö. Ulusoy, "A framework for use of wireless sensor networks in forest fire detection and monitoring," *Computers, Environment and Urban Systems*, vol. 36, no. 6, pp. 614–625, 2012. 95, 100

[148] M. Hefeeda and M. Bagheri, "Forest fire modeling and early detection using wireless sensor networks." *Ad Hoc And Sensor Wireless Networks*, vol. 7, no. 3-4, pp. 169–224, 2009. 95, 98

[149] H. Soliman, K. Sudan, and A. Mishra, "A smart forest-fire early detection sensory system: Another approach of utilizing wireless sensor and neural networks," in *Sensors, 2010 IEEE*. IEEE, 2010, pp. 1900–1904. 95, 98, 114

[150] M. Maksimović and V. Vujović, "Comparative analysis of data mining techniques applied to wireless sensor network data for fire detection," *JITA-Journal of Information Technology and Applications (Banja Luka)-APEIRON*, vol. 6, no. 2, 2013. 95, 99

[151] E. W. Ngai, L. Xiu, and D. C. Chau, "Application of data mining techniques in customer relationship management: A literature review and classification," *Expert systems with applications*, vol. 36, no. 2, pp. 2592–2602, 2009. 96

[152] A. A. Alkhatib, "A review on forest fire detection techniques," *International Journal of Distributed Sensor Networks*, vol. 2014, 2014. 97

[153] A. Singh and H. Singh, "Forest fire detection through wireless sensor network using type-2 fuzzy system," *International Journal of Computer Applications*, vol. 52, no. 9, 2012. 99

[154] V. Khanna and R. K. Cheema, "Fire detection mechanism using fuzzy logic," *International Journal of Computer Applications (0975–8887) Volume*, 2013. 99

[155] P. Bolourchi and S. Uysal, "Forest fire detection in wireless sensor networks using fuzzy logic," in *Computational Intelligence, Communication Systems and Networks (CICSyN), 2013 Fifth International Conference on.* IEEE, 2013, pp. 83–87. 99

[156] G. Demin, L. Haifeng, J. Anna, and W. Guoxin, "A forest fire prediction system based on rechargeable wireless sensor networks," in *Network Infrastructure and Digital Content (IC-NIDC), 2014 4th IEEE International Conference on.* IEEE, 2014, pp. 405–408. 99

[157] K. Pripužić, H. Belani, and M. Vuković, "Early forest fire detection with sensor networks: sliding window skylines approach," in *Knowledge-Based Intelligent Information and Engineering Systems.* Springer, 2008, pp. 725–732. 99

[158] M. Abu Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 4, pp. 1996–2018, 2014. 99

[159] M. Bahrepour, N. Meratnia, and P. J. Havinga, "Use of ai techniques for residential fire detection in wireless sensor networks," 2009. 99

[160] L. Yu, N. Wang, and X. Meng, "Real-time forest fire detection with wireless sensor networks," in *Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on*, vol. 2. IEEE, 2005, pp. 1214–1217. 100

[161] O. Younis, M. Krunz, and S. Ramasubramanian, "Node clustering in wireless sensor networks: recent developments and deployment challenges," *Network, IEEE*, vol. 20, no. 3, pp. 20–25, 2006. 103, 104

[162] S. K. Singh, M. Singh, D. Singh *et al.*, "Routing protocols in wireless sensor networks–a survey," *International Journal of Computer Science & Engineering Survey (IJCSES) Vol*, vol. 1, pp. 63–83, 2010. 104

[163] G. De Meulenaer, F. Gosset, F.-X. Standaert, and O. Pereira, “On the energy cost of communication and cryptography in wireless sensor networks,” in *Networking and Communications, 2008. WIMOB’08. IEEE International Conference on Wireless and Mobile Computing,*. IEEE, 2008, pp. 580–585. 111

Conception d’un réseau de capteurs sans fil pour des prises de décision à base de méthodes du Data mining

Résumé

Les réseaux de capteurs sans fil (RCSFs) déterminent un axe de recherche en plein essor, puisqu’ils sont utilisés aujourd’hui dans de nombreuses applications qui diffèrent par leurs objectifs et leurs contraintes individuelles. Toutefois, le dénominateur commun de toutes les applications de réseaux de capteurs reste la vulnérabilité des noeuds capteurs en raison de leurs caractéristiques et aussi de la nature des données générées.
En effet, les RCSFs génèrent une grande masse de données en continue à des vitesses élevées, hétérogènes et provenant d’emplacements répartis. Par ailleurs, la nécessité de traiter et d’extraire des connaissances à partir de ces grandes quantités de données nous ont motivé à explorer l’une des techniques conçues pour traiter efficacement ces ensembles de données et fournir leurs modèles de représentation. Cependant, parmi les techniques utilisées pour la gestion des données, nous pouvons utiliser les techniques de Data mining. Néanmoins, ces méthodes ne sont pas directement applicables aux RCSFs à cause des contraintes des noeuds capteurs. Il faut donc répondre à un double objectif : l’efficacité d’une solution tout en offrant une bonne adaptation des méthodes de Data mining classiques pour l’analyse de grosses masses de données des RCSFs en prenant en compte les contraintes des nœuds capteurs, et aussi l’extraction du maximum de connaissances afin de prendre des décisions meilleures. Les contributions de cette thèse portent principalement sur l’étude de plusieurs algorithmes distribués qui répondent à la nature des données et aux contraintes de ressources des noeuds capteurs en se basant sur les techniques de Data mining. Chaque noeud favorise un traitement local des techniques de Data mining et ensuite échange ses informations avec ses voisins, pour parvenir à un consensus sur un modèle global. Les différents résultats obtenus montrent que les approches proposées réduisent considérablement la consommation d’énergie et les coûts de communication, ce qui étend la durée de vie du réseau. Les résultats obtenus indiquent aussi que les approches proposées sont extrêmement efficaces en termes de calcul du modèle, de latence, de réduction de la taille des données, d’adaptabilité et de détection des événements.

Mots clés : Réseau de capteurs sans fils, Data mining, clustering, algorithmes distribués.

Conception of a wireless sensor network for decision making based on Data mining methods

Abstract

Recently, Wireless Sensor Networks (WSNs) have emerged as one of the most exciting fields. However, the common challenge of all sensor network applications remains the vulnerability of sensor nodes due to their characteristics and also the nature of the data generated which are of large volume, heterogeneous, and distributed. On the other hand, the need to process and extract knowledge from these large quantities of data motivated us to explore Data mining techniques and develop new approaches to improve the detection accuracy, the quality of information, the reduction of data size, and the extraction of knowledge from WSN datasets to help decision making. However, the classical Data mining methods are not directly applicable to WSNs due to their constraints.
It is therefore necessary to satisfy the following objectives: an efficient solution offering a good adaptation of Data mining methods to the analysis of huge and continuously arriving data from WSNs, by taking into account the constraints of the sensor nodes which allows to extract knowledge in order to make better decisions. The contributions of this thesis focus mainly on the study of several distributed algorithms which can deal with the nature of sensed data and the resource constraints of sensor nodes based on the Data mining algorithms by first using the local computation at each node and then exchange messages with its neighbors, in order to reach consensus on a global model. The different results obtained show that the proposed approaches reduce the energy consumption and the communication cost considerably which extends the network lifetime.
The results also indicate that the proposed approaches are extremely efficient in terms of model computation, latency, reduction of data size, adaptability, and event detection.

Keywords : Wireless Sensor Networks, Data mining, clustering, distribued algorithms.