Data aggregation in wireless sensor networks

Jin Cui

N°d'ordre NNT : 2016LYSEI065

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
INSA de Lyon

Ecole Doctorale N° ED512
Informatique et Mathematiques

Spécialité de doctorat :
Discipline : Informatique

Soutenue publiquement le 27/06/2016, par :

Jin CUI

Data aggregation in Wireless Sensor Networks

Devant le jury composé de :

MINET Pascale	Directeur de Recherche (Inria)	Rapporteur
DIASDEAMORIM Marcelo	Directeur de Recherche (UPMC)	Rapporteur
BEYLOT André-Luc	Professeur des Universités (ENSEEIHT)	Examinateur
ROUSSEAU Franck	Maître de Conférences (ENSIMAG)	Examinateur
BOUSSETTA Khaled	Maître de Conférences (Université Paris 13)	Examinateur
VALOIS Fabrice	Professeur des Universités (INSA-LYON)	Directeur de thèse

THESIS

DATA AGGREGATION IN WIRELESS SENSOR NETWORKS

defended at

INSA DE LYON

for the degree of

Doctor of Philosophy

École Doctorale : INFORMATIQUE ET MATHÉMATIQUES

Spécialité : Informatique

By

Jin CUI

Defense on 27 June 2016

Abstract

Wireless Sensor Networks (WSNs) have been regarded as an emerging and promising field in both academia and industry. Currently, such networks are deployed due to their unique properties, such as self-organization and ease of deployment. However, there are still some technical challenges needed to be addressed, such as energy and network capacity constraints. Data aggregation, as a fundamental solution, processes information at sensor level as a useful digest, and only transmits the digest to the sink. The energy and capacity consumptions are reduced due to less data packets transmission. As a key category of data aggregation, aggregation function, solving how to aggregate information at sensor level, is investigated in this thesis.

We make four main contributions: firstly, we propose two new networking-oriented metrics to evaluate the performance of aggregation function: aggregation ratio and packet size coefficient. Aggregation ratio is used to measure the energy saving by data aggregation, and packet size coefficient allows to evaluate the network capacity change due to data aggregation. Using these metrics, we confirm that data aggregation saves energy and capacity whatever the routing or MAC protocol is used. Secondly, to reduce the impact of sensitive raw data, we propose a data-independent aggregation method which benefits from similar data evolution and achieves better recovered fidelity. Thirdly, a property-independent aggregation function is proposed to adapt the dynamic data variations. Comparing to other functions, our proposal can fit the latest raw data better and achieve real adaptability without assumption about the application and the network topology. Finally, considering a given application, a target accuracy, we classify the forecasting aggregation functions by their performances. The networking-oriented metrics are used to measure the function performance, and a Markov Decision Process is used to compute them. Dataset characterization and classification framework are also presented to guide researcher and engineer to select an appropriate functions under specific requirements.

Key-words : wireless sensor networks; data aggregation; aggregation functions; energy consumption; capacity saving; classification; performance evaluation.

Résumé

Depuis plusieurs années, les réseaux de capteurs sans fil sont considérés comme un domaine émergent et prometteur tant dans le milieu universitaire que dans l'industrie. De tels réseaux ont déjà été largement déployés en raison de leurs propriétés clés, telles que l'auto-organisation et leur autonomie en énergie. Cependant, il reste de nombreux défis scientifiques telles que la réduction de la consommation d'énergie sur des capteurs de plus en plus petits et la capacité du réseau tenant compte de liens à bande passante réduite. Selon nous, l'agrégation de données apparaît comme une solution pour ces deux défis, car au lieu d'envoyer une donnée, l'agrégation va traiter les informations collectées au niveau du capteur et produire une donnée agrégée qui sera effectivement transmise au puits. L'énergie et la capacité du réseau seront donc économisées car il y aura moins de transmissions de données. Le travail de cette thèse s'intéresse principalement aux fonctions d'agrégation

Nous faisons quatre contributions principales. Tout d'abord, nous proposons deux nouvelles métriques pour évaluer les performances des fonctions d'agrégations vue au niveau réseau : le taux d'agrégation et le facteur d'accroissement de la taille des paquets. Le taux d'agrégation est utilisé pour mesurer le gain de paquets non transmis grâce à l'agrégation tandis que le facteur d'accroissement de la taille des paquets permet d'évaluer la variation de la taille des paquets en fonction des politiques d'agrégation. Ces métriques permettent de quantifier l'apport de l'agrégation dans l'économie d'énergie et de la capacité utilisée en fonction du protocole de routage considéré et de la couche MAC retenue. Deuxièmement, pour réduire l'impact des données brutes collectées par les capteurs, nous proposons une méthode d'agrégation de données indépendante de la mesure physique et basée sur les tendances d'évolution des données. Nous montrons que cette méthode permet de faire une agrégation spatiale efficace tout en améliorant la fidélité des données agrégées. En troisième lieu, et parce que dans la plupart des travaux de la littérature, une hypothèse sur le comportement de l'application et/ou la topologie du réseau est toujours sous-entendue, nous proposons une nouvelle fonction d'agrégation agnostique de l'application et des données devant être collectées. Cette fonction est capable de s'adapter aux données mesurées et à leurs évolutions dynamiques. Enfin, nous nous intéressons aux outils

pour proposer une classification des fonctions d'agrégation. Autrement dit, considérant une application donnée et une précision cible, comment choisir les meilleures fonctions d'agrégations en termes de performances. Les métriques, que nous avons proposé, sont utilisées pour mesurer la performance de la fonction, et un processus de décision markovien est utilisé pour les mesurer. Comment caractériser un ensemble de données est également discuté. Une classification est proposée dans un cadre précis.

Mots-clés : réseaux de capteurs sans fil ; agrégation de données ; agrégation temporelle ; consommation d'énergie; capacité du réseau ; classification ; évaluation de performances.

To my youth that is fading away

Acknowledgements

First and the foremost, I would like to express my sincere gratitude to my advisor Prof. Fabrice VALOIS for his supports and trust. He is most responsible for helping me complete this thesis as well as the challenging research that lies behind it. His wide knowledge and his logical way of thinking have been of great value for me. He was always there to meet and talk about my ideas, to proofread and mark up my papers, and to ask me good questions to help me think through my problems. It was his enlightening supervision that inspired me to explore novel research areas and broadened my views in the research area. It was a wonderful and rewarding journey to learn from him.

I also feel grateful to Dr. Khaled BOUSSETA. Thanks to his professional insights and technical guidance, he has spent very much precious time to discuss my work and to help me analyse the existed problems and so on during the last two years.

I take this opportunity to thank the jury members for spending their precious time to read and review this thesis, for taking a long trip to attend this defence, and for giving me valuable comments. I am grateful for the financial support of the CSC through UT-INSA project.

Additionally, I would also like to thank my dear colleagues at CITI laboratory for sharing the experiences on their studies and works. I would like to thank administrative staffs at CITI for their instant supports, especially for the secretaries who serve for Urbanet team. I would like to extend my thanks to all my friends, they are always standing by me to give me confidence and courage. Here, I should give my special thanks to Dr. ZHANG Biao and CHERKAOUI Soukaina, for giving some suggestions about my manuscript.

As always, I would like to extend my special gratitude to my parents and my family. Their unconditional support has always helped me to bounce back whenever I felt low, and they have always been helping me out of difficulties without a word of complaint.

Villeurbanne, 10.Jun.2016

Contents

List of Tables

List of Figures

1 Introduction

Contents

[1] Wireless Sensor Networks [1, 2] (WSNs) have attracted a lot of attention from both the academic world and industry over the past ten years. Due to the characteristics of flexibility, ease of deployment, self-organization, and so on, a variety of applications are developed base on the technologies of WSN. However, limited battery, massive raw data, and unstable wireless links leads to several constraints (such as energy constraint, memory constraint, and network capacity constraint), which limit the

[1]This thesis was financed and supported by China Scholarship Council (CSC). It is realized at CITI Laboratory of INSA Lyon, in the Urbanet Team of Inria Rhône-Alpes.

performance of wireless sensor network. How to deal with the above constraints has become a hot topic for researchers. Data aggregation [3, 4], which is a way for saving energy and network capacity, has been frequently investigated. By studying the correlations of raw data, data aggregation reduces the data packets in the network, thereby saving communication cost and easing network congestion. In this manuscript, we place our focus on data aggregation of wireless sensor network.

1.1 Global view of Wireless Sensor Networks

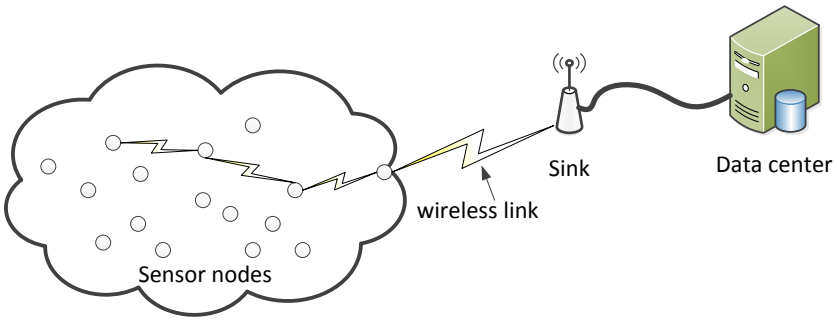


Figure 1.1: General architecture of wireless sensor network.

Typically, a WSN consists of hundreds or thousands of wireless sensor nodes and a sink node, where the sensor nodes own the ability of sensing, processing, communicating, and transmitting. As shown in Fig. 1.1, these sensor nodes sense the environmental factors (temperature, humidity, pressure, motion and other physical variables), communicate with each other, and transmit information. The sink node, like a base station, is deployed to collect the information [5]. The small, distributed, and feasible sensor nodes accelerate the development of WSN. As shown in Fig. 1.2, there are varieties of applications benefiting from WSNs, such as building monitoring [6], health care [7], smart agriculture [8, 9], military surveillance [10], environment monitoring [11, 12], and detection issues [13, 14, 15].

However, due to the limited battery power of sensor nodes, the network lifetime and performance are restricted. Meanwhile, in several applications (e.g. temperature monitoring), sensor nodes are prone to transmit redundant or correlated information to the sink, which wastes the bandwidth, thereby wasting the network capacity and accelerating the battery depletion. Therefore, how to save energy and network capacity are central challenges for researchers regarding this field of research.

By investigating energy consumption in one sensor node, we found that the major power drain occurs from wireless communication [16]. Thus in order to save energy, a

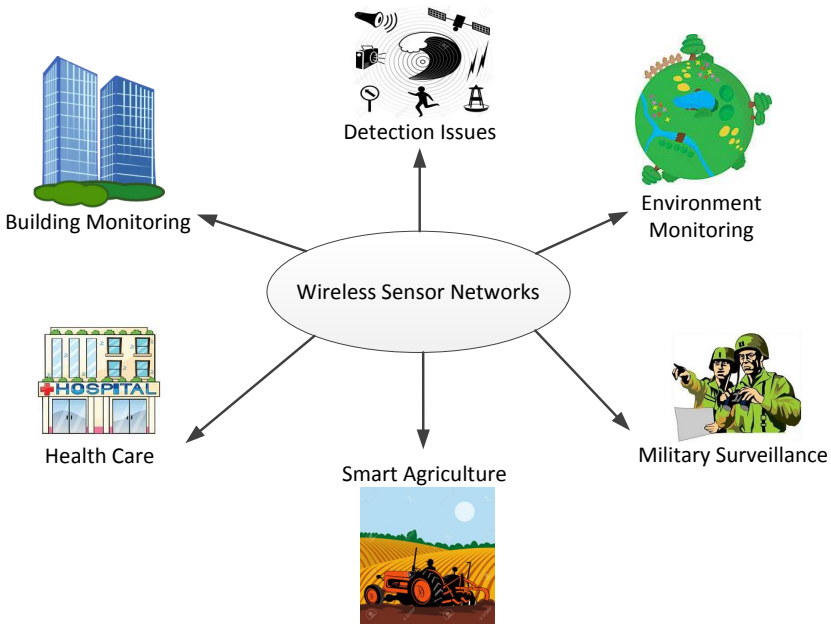


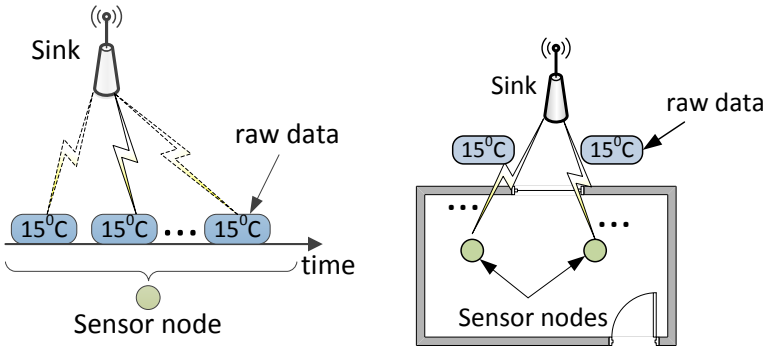
Figure 1.2: Versatile applications based on Wireless Sensor Networks.

reasonable solution is to reduce the communication activity. Data aggregation, which can reduce communication by reducing the number of data packets transmitted in the network, is considered as a fundamental way to save energy [17].

1.2 Data aggregation overview

Without the use of data aggregation in a WSN, sensor nodes will report all the raw data [2] to the sink. While these data tend to be redundant or correlated, leading to several drawbacks: 1) the redundant data is no sense for the application, 2) the chances of network congestion increase dramatically, 3) the network capacity is wasted, 4) energy consumption increases correspondingly. By previous studies [18, 19, 20], temporal and spatial correlations are often based on the raw data. For a given sensor node, temporal correlation exists in the data collected at different time instants, while spatial correlation occurs when the data is collected from the neighboring sensor nodes. As an example shown in Fig. 1.3(a), when a sensor node is used to monitor temperature in an area, the obtained values often keep constant during 30 minutes or even one hour. Furthermore, when two sensor nodes are deployed in a same room to monitor temperature, as shown in Fig. 1.3(b), the data obtained by one node is often similar or even same to another's.

[2]Raw data mentioned in this manuscript denotes the value of the variables, e.g. temperature value is 15^0C.



(a) Temporal correlation (b) Spatial correlation

Figure 1.3: Examples of temporal and spatial correlations.

Data aggregation [3, 4] studies the property of raw data and benefits from the correlations. Using a data aggregation mechanism, sensor nodes process the raw data into a digest, and only send such digest to the sink. As a result of reduced amount of the digest, data aggregation reduces the transmission cost and network overloading.

1.2.1 Goals of data aggregation

Generally speaking, an aggregation protocol should achieve three main objectives, which are:

1) **Energy saving**: Data aggregation reduces the redundant or correlated transmissions in a network, which directly minimizes the energy consumption for the whole network [21]. Since the energy limitation is a main constraint for WSN, the design of data aggregation should put energy saving as the main concern.

2) **Data accuracy**: Data accuracy is the accuracy between the recovered data [3] and raw data. Sensor nodes aggregate raw data into a digest which may lost several information. Thus it is reasonable that the recovered data at sink side has some deviation comparing to raw data [22]. Thus, how to save energy with an acceptable accuracy is a general requirement to be considered for any applications.

3) **Network capacity saving**: Bandwidth constraints of sensor nodes limit the network capacity of WSNs, so how to save the capacity has also investigated frequently. By sending less packets to the sink, data aggregation can save network capacity. Furthermore, how much network capacity has been saved can be seen as a metric to evaluate an aggregation protocol [23]. Thus we also need to consider the goal of saving network capacity when designing an aggregation protocol.

According to the requirements of specific applications, data aggregation may need

[3] We define recovered data as the data computed at the sink level.

to realize other goals, such as minimum latency [21, 24], security guarantees [25], and privacy protection [26, 27].

1.2.2 Categories of data aggregation

Based on the state of the art, we can classify three categories of data aggregation: aggregation structure, aggregation function, and aggregation scheduling.

1) Where to aggregate is the objective of **aggregation structure**. This category defines how the aggregated data is routed towards the sink by using or creating a network structure, and the structure should prompt the in-network aggregation. Basically, aggregation structure finds several sensor nodes as aggregators or gateway nodes to process raw data, and the aggregated information will be transferred through the structure. Aggregation structure is more like a data-centric routing, which makes nodes relay data with more information. Literature-related aggregation structure proposes hierarchy or backbone based structures to forward aggregated data[4, 28, 29].

2) How to aggregate is defined using **aggregation function**. This issue is the most important part for data aggregation, which focuses on how sensor nodes aggregate raw data into a digest. An efficient and useful aggregation function helps sensor nodes to reduce energy consumption dramatically. We have mentioned above that there are two correlations (i.e. temporal and spatial correlations) in raw data, and data aggregation function mainly benefits from these correlations. Basically, simple operations are used as aggregation functions (based on temporal correlation), such as Average, MAX(MIN), SUM, COUNT, and Median [4]. As the requirements of recover fidelity, more complicated aggregation functions are proposed [30, 31, 32, 33]. Chapter 4 and chapter 5 belong to this category.

3) When to aggregate is concerned by **aggregation scheduling**. Data aggregation needs to process the data and then transmit them, thus it leads to delay between source node and sink. Aggregation scheduling is proposed to solve this problem. More precisely, aggregation scheduling defines when a sensor node should aggregate data and when the node should forward data. The purpose of aggregation scheduling is to reduce the delay caused by the data aggregation. There are also many works focusing on this issue [34, 35, 36, 37]. The theoretical work in Chapter 6 discuss when to aggregate in detail.

Besides the categories we have listed, there are other issues also regarding data aggregation, such as security issues [38] and mobility issues [39]. As wireless sensor networks may be deployed in hostile areas such as battlefields, the raw data should be confidential. Under such situation, data aggregation mechanisms should work with

communication security protocols, as any conflict between these protocols might create loopholes in the network security. Mobility issues are considered frequently in mobile Ad Hoc Networks. For example, how to aggregate data through mobile vehicular? when to transfer it to another vehicular? Both of them are challenges for data aggregation.

1.3 Motivations of data aggregation

We claim that data aggregation is a key mechanism to reduce energy in WSNs. However, there are still some challenges needed to be overcome, in order to improve the performance of data aggregation.

In the existing contributions, several aggregation schemes rely on the raw data to group sensor nodes, in order to aggregate information. However, abnormal data often appears in raw data. Thus the data instability definitely impacts the performance of such schemes. In addition, several aggregation functions are specified for a certain data (e.g. temperature data) or a type of network property (e.g. grid network), which limits the adaptivity of these functions. Therefore, we are motivated to propose data-independent and property-independent aggregation solutions.

The first objective of this thesis is to investigate the existing aggregation works, and the second one is to deal with the performances of data aggregation functions. There are many contributions dealing with aggregation functions, however, there is no guideline for researchers or users on how to select one of them for a given application. Even though the application type and accuracy requirement are provided, users do not know which aggregation function is suitable. For example, if we deploy a WSN to monitor number of vehicles, a given function A performs better than a given function B, but when we use the same WSN to monitor temperature, maybe function B performs better than function A. Is it caused by different types of these two raw data? If we release the accuracy requirement, will the performance of aggregation functions change? These questions motive us to study the performance of different aggregation functions, and to propose guideline for selecting aggregation functions.

1.4 Contributions and organization

1.4.1 Contributions

The main contributions of this thesis are in four-fold:

- Networking-oriented metrics are proposed to evaluate the impact of data aggregation at networking level: *aggregation ratio* and *packet size coefficient* respectively. Aggregation ratio describes the ability of saving energy of an aggregation function, and packet size coefficient is used to demonstrate the network capacity saved when using an aggregation function.

- An efficient data-independent aggregation method, Similar-evolution Based data Aggregation (denoted as Simba), is provided to avoid the use of sensitive raw data. We investigate real datasets, and find that several nodes show similar evolutions even though their data are different. Thus we propose Simba to group the nodes having similar evolutions together, and data aggregation is achieved by the unit of group. Our experiments show that Simba reaches higher recover fidelity than the method which is based on raw data only. Meanwhile, Simba saves more energy.

- Without considering the data property, we propose a new aggregation function: Agnostic Aggregation (A^2). A^2 can go against non-anticipated data variations (like data type changes). More specifically, A^2 allows sensor nodes to dynamically adjust the aggregation function to adapt the latest data. To the best of our knowledge, A^2 is the first aggregation function which is able to self-adapt to the data and to the environment.

- Considering a given application and the associated accuracy constraint, the question is how to select the optimal forecasting data aggregation function. To answer this question, we propose a framework to classify aggregation functions. To evaluate the performances of aggregation functions, we compute networking-oriented metrics by means of building a stochastic model. Finally, we classify the aggregation functions under the light of three perspectives: performances, data and accuracy constraints.

1.4.2 Organization of the manuscript

This thesis is structured in six main chapters:

We firstly review the state-of-the-art data aggregation works in Chapter 2. We mainly review the contributions about aggregation structures and aggregation functions.

In Chapter 3, we introduce two new networking-oriented metrics: *aggregation ratio* and *packet size coefficient*. These metrics are associated to evaluate the impact of data aggregation at the networking level.

A new aggregation mechanism is proposed in Chapter 4. Instead of considering sensitive raw data, we propose a data-independent aggregation: Similar-evolution Based Aggregation (Simba). Considering data evolution, Simba groups the nodes having similar evolution together, and execute aggregation functions in the unit of group. The design of Simba avoids the sensitivity of raw data, guarantees the recover fidelity, and also saves more energy.

In Chapter 5, an Agnostic Aggregation function (A^2) has been proposed to save energy and network capacity. As a data property-independent function, A^2 keeps the recover accuracy regardless of data type. Most of data aggregation functions are designed for only one type of data property, our A^2 can adjust the function to fit the latest data.

To classify the forecasting aggregation functions, a whole framework is provided in Chapter 6. Firstly, we build a stochastic model to compute networking-oriented metrics for different aggregation functions; secondly, we characterize different datasets; and finally, we classify the functions from three perspectives: performances, data type, and accuracy constraints.

In Chapter 7, we summarize the main contributions of this manuscript, and we discuss several possible extensions of the work covered in this thesis.

2

State-of-the-art data aggregation

Contents

As we illustrated in Chapter 1, the techniques of data aggregation can be mainly classified in 3 categories: aggregation structure, aggregation function and aggregation scheduling. Aggregation scheduling deals with the problems about how long a node should wait before aggregating and forwarding received data [34]. As this category is highly related with aggregation function, many state-of-the-art aggregation functions are designed to reduce the latency. That is to say these functions focus on the questions

that when to aggregate data and when to forward it. Hence, we consider aggregation scheduling as a part of aggregation function.

Aggregation structure [40] organises the sensor nodes to do in-network aggregation. It mainly defines the path of aggregating data and the locations of aggregators. The data packets from source nodes are relayed on a structure to the sink, and the structure should make more data can be aggregated along this structure. In fact, the simplest way to aggregate data from source to sink is to pre-choose nodes that worked as *aggregator* or *gateway* nodes, and pre-define a preferred direction to be followed when forwarding data.

Aggregation function [4] is the computation part of data aggregation, which is responsible for how to do aggregation. It defines the methodology of executing aggregation at sensor node level. At the initial studies of aggregation functions, simple operations (MAX, MIN, Average etc) are used to process data. As the improvement of accuracy requirements, more complex aggregation functions are proposed. They are based on mathematical theories, and can achieve higher recover fidelity than the simple operations.

The objective of this chapter is to review the existing works on two indispensable mechanisms: aggregation structures and aggregation functions.

2.1 Data aggregation structures

In the viewpoint of aggregation structure, if some special nodes are pre-defined to aggregate information, other nodes can relay data to these nodes, and in-network aggregation will be prompted. There are many works proposing aggregation structures in this way, such as hierarchy-based structures (tree-based or cluster-based structures), and backbone-based structures (backbone, dominating set).

2.1.1 Hierarchy-based structures

Hierarchy-based structures make sensor nodes form a hierarchial shape, such as tree or cluster. Data is transferred from lower-level nodes to higher-level nodes, and aggregation will be achieved by the higher-level nodes. In tree-based structure, children nodes send information to parent nodes, and aggregation is achieved by parent nodes. In cluster-based structure, cluster member nodes report data to cluster head, and aggregation will be performed by the cluster head. In the following, we mainly review tree-based and cluster-based structures.

Tree-based structures

The Tiny AGgregation (TAG) [41], a data-centric protocol, is based on a tree structure and specifically designed for monitoring applications. There are two phases for TAG, one is the distribution phase, another is the collection phase. During the distribution phase, sink broadcasts queries to the target sensor node. From the query message, the route from sink to sensor is constructed. During the collection phase, each parent has to wait for data from all of its children before it can send its aggregated information up the tree. Time slot is used for each node, once time is over, node can go to sleep to save energy. Data aggregation can be achieved on intermediate nodes by simple operations (e.g., Average, Max, Min). As discussed above, we know TAG has two requirements: first, sink can deliver query requests to all the network nodes; second, each node has at least one route to the sink. Thus TAG may be inefficient for dynamic topologies or link failures.

Energy Aware Data Aggregation (EADA) [42] combines a grid structure with on-demand data dissemination tree. In each grid cell, the node with the maximum residual energy is selected as the gateway node which is responsible for aggregating the data generated within the grid cell. The sink floods data queries through gateways restricted on a circle sector towards the interest zone. Once the query enters the interest zone, the entry gateway becomes the root of a newly constructed tree which covers all the nodes in the interest zone. The aggregated data are then disseminated through the reverse of the query route to the sink. If the energy of the root gateway goes below a certain threshold, the gateway with the maximum remaining energy in the interest zone is selected as the new root and a new tree is formed. However, establishing and maintaining a separate tree for each interest zone may increase the overall energy consumption in the network.

The authors of [43] construct a data aggregation tree (MECAT) that minimizes the total energy cost of data transmission, and propose algorithms to solve it. They analyze the tree construction under some certain aggregation ratio (the size of reports that can be aggregated into one packet). They prove that every shortest path tree has an approximation ratio of 2. When there are pure relay nodes (only forward data) in the network, the problem is proved to be NP-complete and a seven-approximation algorithm is proposed. Recently, there have some works about constructing a tree routing structure with maximum lifetime, such as [44].

Cluster-based structures

Low Energy Adaptive Clustering Hierarchy (LEACH) [29] is an energy conserving cluster formation method. LEACH uses randomization to distribute the energy expenditure

among the sensor nodes. Cluster-based structure is exploited to perform data aggregation, and cluster head (CH) is treated as aggregator. At beginning, each node elects itself to be the local CH for the current round R. The algorithm aims to have a percentage P of the nodes acting as CH, where P has to be optimally chosen according to the nodes density. Node i calculates a threshold $T(i)$:

$$T(i) = \begin{cases} \frac{P}{1-P(Rmod(1/P))} & if\ i \in G \\ 0 & otherwise \end{cases} \tag{2.1}$$

where P is the desired percentage of CH, R is the round number, and G is the set of nodes that have not been CH during the last $1/P$ rounds. A node i picks a random number from 0 to 1 and decides to be a CH if this number is lower than $T(i)$. A cluster head sends advertisement to its neighbours, and the neighbours decide to join a cluster by signal power of the advertisement. For aggregation, the nodes in one cluster send data to CH in TDMA transmission slot, and then move to sleep until next slot for them. An aggregation function can be achieved by CH. LEACH is a completely distributed clustering algorithm, but maintaining cluster needs more energy.

Based on LEACH, *energy-efficient mobile sink routing protocol algorithm* (EEMSRA) [45] makes cluster heads create a TDMA schedule informing each node in the cluster of when they can transmit data. The CHs also perform aggregation before transmitting data to the sink. The sink broadcasts its next projected cluster visit in order to enable the network to update routes prior to the sink's actual arrival at the cluster. This approach, while saving significant amounts of energy, requires the sink to at least have knowledge about its short-term trajectory (mobile sink). EEMSRA forms clusters with enforced TDMA schedules to increase energy-efficiency, thus it has MAC layer requirements, so it might not be applicable to a wide range of sensor network.

Hybrid Energy Efficient Distributed clustering approach (HEED) [46] is proposed as an improvement over LEACH. The enhancement is done in the cluster head selection method. In HEED, the CH selection is not random, it is based on residual energy and node density. Each sensor node sets the probability P_{CH} of becoming a CH as follows:

$$P_{CH} = P \cdot \frac{E_{residual}}{E_{max}} \tag{2.2}$$

where P is the initial percentage of CHs required by application, $E_{resedual}$ is the current residual energy of node, and E_{max} is its initial energy (maximum one). HEED is a fully distributed routing technique and achieves load balancing and uniform cluster head distribution. However, in HEED, nodes having less energy expire faster than nodes with higher energy level.

In [47], the authors present a spatial clustering algorithm to achieve data aggregation. The contribution is mainly made for environmental surveillance applications in high density sensor networks. The aggregation algorithm constructs a dominating set by exploiting the spatial correlation between data measured by different sensors. The dominating set is further considered to execute data aggregation on the basis of information summarization of the dominator nodes.

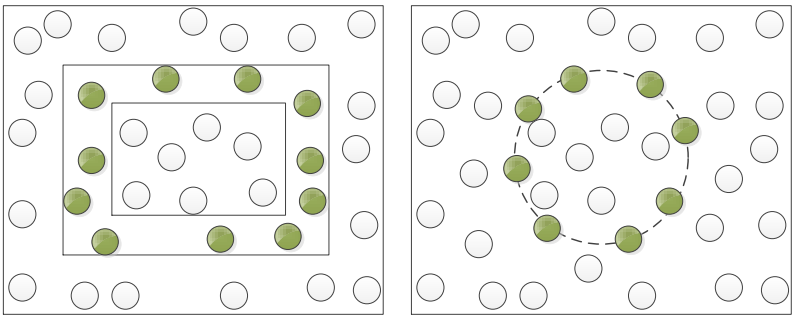
The authors of [48] focus on evaluating performances of data aggregation at data fusion center for a WSN which is divided into sensor clusters. A dynamic clustering and aggregation strategy are investigated for local data at the sensor node and global data aggregation at the cluster head. They propose a clustering algorithm comprising two phases. In first phase, sensor nodes which sense the same category of data are grouped in the same cluster. In second phase, it guarantees that all sensor nodes are assigned to the clusters (formed in the first phase), based on the least-divergent clusters.

2.1.2 Backbone-based structures

Backbone-based structures are also investigated frequently in WSN. The underlying idea is defining backbone path or connected dominating set in the network. All the other nodes can forward data from it, and the nodes on the backbone are responsible for aggregating information. This type of structures also facilitates the process of data dissemination, and is useful for mobile sink applications.

Directed Diffusion (DD) [28] is a reactive data-centric protocol. The routing is specifically tailored for the situation when sink is interested in collecting data from some nodes in the network. Sink propagates an interest message to the source in an interesting area. Each node rebroadcasts the message to its neighbours once receiving it. In addition, node sets up gradient to sink. As the gradient setup is finished, only a single path from source is reinforced and used to route packet toward the sink as the reverse direction of gradient.

Rail road [49] adopts a virtual infrastructure called a rail that is placed in the middle area of the network. As illustrated in Fig. 2.1(a), the nodes inside this rail are called rail nodes. When a source generates a new information, it sends data to the rail nodes, and rail nodes will aggregate information. As soon as a sink node needs to collect the generated data reports, a query message is sent into the rail until it reaches the rail nodes that store the relevant source node information. However, the delay is a problem for rail road protocol. Similarly, Ring Routing [50] proposes a ring structure which is a closed loop of single-node width (see Fig. 2.1(b)). The ring encapsulates a globally predetermined network center, the aggregation and dissemination are achieved by the



(a) Rail road structure (b) Ring structure

Figure 2.1: Various structures.

nodes on the ring. The drawback of this routing is the overhead of ring construction for large or sparse networks.

Dynamic Directed Backbone (DDB) [51] organizes sensor nodes into one-hop clusters. A sensor node in each cluster is selected as a Leader (namely, cluster head) and the other sensor nodes in each cluster become members. Some members which can connect between leaders become gateway nodes, as shown in Fig. 2.2. DDB constructs a backbone through leaders and gateway nodes. When a mobile sink enters the network, it tries to attach to a member or a gateway and sends a query to its leader through it. This query is then flooded along all nodes in the backbone until reaching the leader receiving the reporting data of a source node. If the mobile sink moves, it attaches to a new leader through a new member or gateway and refloods the query in the whole backbone through the new leader. However, by using one-hop clustering, DDB cannot have enough nodes to substitute for a failure CH and enough links to substitute for a failure link between CHs.

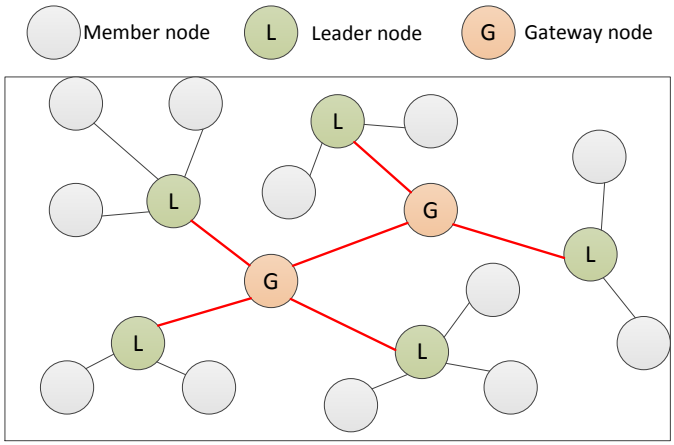


Figure 2.2: Backbone-based structure.

Data Quality Maximization (DQM) [52] is another protocol based on a backbone consisting of gateways. This protocol assumes predictability of the sink movement and selects gateways adjacent to the predicted path of the sink. The sensors establish shortest path routes with the gateways using Floyd-Warshall algorithm. Gateways aggregate incoming data and wait for the sink. However, DQM is only applicable in delay-tolerant applications since the selected gateways disseminate the aggregated data only when the sink is nearby.

Besides hierarchy-based and backbone-based aggregation structures, there are also some works using other structures, e.g. map. The map here is marked by isolines, the nodes holding similar raw data are considered on the same isoline. In [53], the authors propose an isoline aggregation to reduce redundant transmissions. The isoline is a pre-defined value range. Taking temperature as an example, if the isolines measure multiples of 10^0C, then a node sensing 25^0C and a neighbor whose sensed value is 32^0C are able to detect that there is (at least) one isoline of value 30^0C passing between them. Nodes only report to the sink when there are new isolines nearby. Isoline aggregation uses local information from neighbors to group nodes that report similar data. Energy efficiency is achieved by having only a subset of the nodes, that is, the ones detecting the isolines, report to the sink. Authors of [54] improve the isoline aggregation, they propose a contour map consisting of isolines. Essentially, a node only reports to the sink if it detects an isoline between itself and its adjacent nodes and waits long enough time, to avoid the dynamic of raw data. The authors evaluate map construction protocols in [55], and highlight that: the performances of map construction approaches depend on physical phenomena properties, such as the dynamic data value and spatial coverage, and on network properties such as the communication range and failures.

2.1.3 Structure-free data aggregation

Either hierarchy-based or backbone-based structures make the data easily be aggregated along the predefined way, but they all need to build and maintain the structures, which consumes more energy. To avoid maintenance overhead, structure-free methods [56, 57, 58] are proposed.

The authors of [56, 59] propose a structure-free scheme for data aggregation which does not require any predefined structures. The protocol consists of two mechanisms: *Data-Aware Anycast* (DAA) at MAC layer and *Randomized Waiting* (RW) at the application layer. DAA uses anycast to forward packets to one-hop neighbors that have packets for aggregation and RW is used at the source nodes for each packet to introduce artificial delays and increase temporal convergence. With higher temporal

convergence, nodes can aggregate more information in one aggregated packet. This method does not produce any communication overhead for structure maintenance, thus it is appropriate for dynamic networks. However, the randomized waiting may introduce long delay from source node to sink.

In [57], the authors propose a structure-free Real-time data Aggregation protocol (RAG). RAG makes use of two methods for temporal and spatial convergence of packets: *Judiciously Waiting policy* and *Real-time Data-aware Anycasting policy*. On the one hand, Judiciously Waiting policy is used to satisfy the on-time delivery of data packets. In this policy, the End-to-End delay (EED) is calculated by measuring estimated one-hop delay, including channel contentions, packet transmissions and queuing delay by using a time-stamping method. The Waiting Timeout (WT, the maximum waiting time) for a packet at a node with H hops away from the sink is calculated as:

$$WT = \frac{TTD - EED}{1 + (\frac{H-1}{H})} \cdot \alpha \tag{2.3}$$

where TTD (Time-to-deadline) indicates the remaining time of the packet to be received at the destination, and α is a constant factor. On the other hand, Real-time Data-aware Anycasting policy decides which next hop node achieves better aggregation performance while satisfying real-time requirements. In this policy, node A computes the required velocity based on the progress made toward the sink and the packet's TTD before forwarding its data to the next hop as:

$$V_{req} = \frac{D(A, Sink)}{TTD} \tag{2.4}$$

where $D(A, Sink)$ is the Euclidean distance between node A and the sink. Therefore, by satisfying the required velocity of each hop, the end-to-end deadline is addressed.

The authors of *Structure-Free and Energy-Balanced* scheme (SFEB) [58] introduce a two-phase structure-free aggregation scheme. This method enables both efficient data gathering and balanced energy consumption. In phase one, using the concept of *gather before transmit*, data collecting nodes (aggregators) are selected first to gather their neighbors sensing data as many packets as possible. Then, these aggregators send the collected packets to the sink at phase two. However, the existing of aggregators makes SFEB not be a total structure-free method.

2.2 Data aggregation functions

Aggregation functions focus on how to do aggregation, which is the way to process raw data into a digest. As we mentioned in Sec.1.1, there are two types of correlations in raw data: temporal correlation and spatial correlation. Temporal correlation exists in data collected by one sensor node at different time instants; and spatial correlation often occurs when the data are collected from local neighboring sensor nodes. Corresponding to the two correlations, we define two types of aggregation functions: forecasting and compressing functions.

Forecasting data aggregation uses prediction models or empirical data to predict the next one, while compressing aggregation is committed to compress information. Several simple operations, such as Average, MAX, MIN, COUNT, SUM, etc, can be treated as any types of forecasting or compressing functions. Take Average as an example, the average value of certain data can be used to predict the next data, and also can be seen as a compressed data.

2.2.1 Basic data aggregation functions

Basic aggregation functions are simple operations. We introduce usual operations that are often used in practical applications. First, average, it is ease to implement on sensor node. Supposing that raw data is represented by v_t at time t, denoted as (v_t,t).The function of average can get an aggregated value as:

$$\mathcal{F}_{average} \equiv \bar{v} = \frac{v_1 + \cdots + v_t}{t} \tag{2.5}$$

where $\bar{v}$ is the average value of t raw data.

Secondly, MAX (*resp.* MIN), maximum (*resp.* minimum) is mainly used in alarming applications. These applications do not need to know raw data, but if the raw data is too high (*resp.* too low) for their requirements, they need to trigger alert. Function of MAX (*resp.* MIN) can be formulated as:

$$\mathcal{F}_{max} \equiv v_{max} = MAX\{v_0, \cdots, v_t\} \tag{2.6}$$

$$\mathcal{F}_{min} \equiv v_{min} = MIN\{v_0, \cdots, v_t\} \tag{2.7}$$

where v_{max} (*resp.* v_{min}) denotes the MAX (*resp.* MIN) value of previous t raw data.

Thirdly, SUM, it represents the summation of raw data, sum can be calculated by:

$$\mathcal{F}_{sum} \equiv v_{sum}(t) = \sum_{i=1}^{t} v_i \tag{2.8}$$

where v_{sum} is the sum value of $\{v_1, \cdots, v_t\}$. There are also operations like COUNT (which is used to count how many data have been collected), Median (the median value of collected data) and so on. The drawbacks of these functions are that they cannot guarantee the accuracy according to a given error threshold. Application can choose one of them to achieve energy efficiency and network capacity saving if accuracy is not a hard requirement.

2.2.2 Forecasting data aggregation

Forecasting aggregation functions benefit from temporal correlations of the raw data. The data series of a sensor node (such as temperature, humidity) often show that the current data value is related with the previous one (same or similar to it), and mean and variance of the whole dataset often do not change. Such data series is stationary, which can be predicted accurately.

Forecasting aggregation is a generic name for the functions that predict data using several models or methods. As sink can recover the high-fidelity data from the models or methods, sensor nodes do not needs to send all the raw data to the sink, thereby reducing the number of transmissions and then saving energy.

Theoretical background

The Auto Regression Moving Average (ARMA) [60] model is a widely-used forecasting model for time series analysis. It uses the historical data to develop a model to predict the future data. Many environmental physical quantities, with stationary property, can be modelled by this way. The model incorporates two terms, the Auto-Regression(AR) term, and the Moving Average (MA) term.

The AR term is a linear regression which represents the self-deterministic part of the time series. AR term forecasts the current data $\dot{v_t}$ with p prior data:

$$\dot{v_t} = \varphi_0 + \varphi_1 \times v_{t-1} + \cdots + \varphi_p \times v_{t-p} \tag{2.9}$$

An AR(p) model is characterized by the $p+1$ coefficients: $\varphi_0, \varphi_1, \cdots, \varphi_p$.

The MA term captures the influence of random shocks which is independent from autoregressive process. The model consists of random shocks on q prior elements:

$$\dot{v_t} = \vartheta_1 \times \varepsilon_{t-1} + \cdots + \vartheta_q \times \varepsilon_{t-q} \tag{2.10}$$

A MA(q) model is characterized by the q coefficients: $\vartheta_1, \cdots, \vartheta_q$, and ε is the white

noise. Thus, the formation of ARMA(p,q) is the addition of Eq. 2.9 and Eq. 2.10, i.e.,

$$\mathcal{F}_{arma} \equiv \dot{v_t} = \varphi_0 + \varphi_1 \cdot v_{t-1} + \cdots + \varphi_p \cdot v_{t-p} + \vartheta_1 \cdot \varepsilon_{t-1} + \cdots + \vartheta_q \cdot \varepsilon_{t-q} \tag{2.11}$$

where (p, q) is the order for an ARMA model, the bigger the order is, the higher the algorithmic complexity is, and the higher the accuracy should be. Several information criteria are used to select the optimal order, e.g. Akaike's Information Criteria (AiC) [61] and Bayesian Information Criteria (BiC) [62].

Since ARMA model is suitable for time series within stationary property, not all time series are certainly stationary. Another model, Auto Regressive Integrated Moving Average model(ARIMA) [63], is used for the non-stationary scenarios. ARIMA supposes that the integrate value between time series should be stationary.
ARIMA(p,d,q) means that the model have p AR term, q MA term, and the number of difference needed for stationarity is d, the AR term and MA term here are same with ARMA (Eq. 2.11). For clarity, we assuming that y denotes the difference of v, thus there are:

$$\text{If } d\text{=0: } y_t = v_t$$

$$\text{If } d\text{=1: } y_t = v_t - v_{t-1}$$

$$\text{If } d\text{=2: } y_t = (v_t - v_{t-1}) - (v_{t-1} - v_{t-2}) = v_t - 2 \cdot v_{t-1} + v_{t-2}$$

In terms of y, the equation of ARIMA model is:

$$\mathcal{F}_{arima} \equiv \dot{y_t} = \varphi_0 + \varphi_1 \cdot y_{t-1} + \cdots + \varphi_p \cdot y_{t-p} - \vartheta_1 \cdot \varepsilon_{t-1} - \cdots - \vartheta_q \cdot \varepsilon_{t-q} \tag{2.12}$$

Actually, for the non-stationary series, ARIMA highlights that the difference y between the adjacent data is stationary. We can see that the formation of Eq. 2.12 is similar to Eq. 2.11, while the variable in Eq. 2.12 is y.

Polynomial regression is also an usual prediction model, which searches for a relationship between v and t. It is formulated as:

$$\mathcal{F}_{poly} \equiv \hat{v} = a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n \tag{2.13}$$

where n is the order of the polynomial, and coefficients $a_0, \cdots, a_n$ can be calculated by least squares method.

Forecasting data aggregation functions

Based on the theoretical forecasting models, researchers propose alternative data aggregation functions which can be achieved on sensor nodes.

In [32], the authors use ARIMA model in wireless sensor networks. They make a sink-driven method, i.e. firstly sink computes the coefficients (φ and ϑ in Eq. 2.12) and sends them to corresponding sensor nodes. When accuracy cannot satisfy the threshold, sink recomputes coefficients. Moreover, this work does not rely on AiC or BiC due to the huge computation, the author defines a new metric C, which is:

$$C = \alpha \times MAE + (1 - \alpha) \times r_{tran} \tag{2.14}$$

where MAE is the Mean Absolute Error defined by the threshold, and the r_{tran} is the ratio of the number of data transmitted over the total number of data. In Eq. 2.14, $\alpha(0 \leq \alpha \leq 1)$ is used to trade off between MAE and r_{trans}, i.e. between energy consuming and data accuracy. [32] simplifies the utilisation of ARIMA in WSNs, however, sink needs a preparation phase to collect the data and compute the parameters, which is not convenient in some cases. Additionally, it is a centralized method, which is not suitable for self-organized sensor networks.

In contrary, [30] presents adaptive-ARMA (A-ARMA) in WSNs. A-ARMA reduces the computation in every sensor nodes, and it does not require pre-computation phases. The basic idea of A-ARMA is that each node computes an ARMA model based on a fixed-size window of W consecutive data, and note that W is also a sliding window. By merely sending only the coefficients (φ and ϑ in Eq. 2.11) of the ARMA model to the sink for rebuilding data, the temporal correlation of these data within each window is explored. The model coefficients are used at the sink level for data forecasting (using Eq. 2.11), unless it receives new model updates from the sensor nodes. As illustrated in Fig. 2.3, each node locally verifies the accuracy of the forecasted data ($\hat{v}_i$) with raw data (v_i). If the accuracy (RMS error) is acceptable according to a given threshold, the node assumes that the sink is able to rebuild the data correctly and there is hence no need to report the data. Otherwise, it computes a new model based on the latest W data, and communicates the new parameters to the sink so as to adjust the forecasting. In order to reduce the complexity in the model estimation process whilst still achieving a high accuracy, a moving window technique is introduced. This means that the verification is required every time the window moves a step ahead. The adaptive nature of the technique relyes on the use of this moving window. Moreover, it also decreases the computation complexity.

Polynomial regression [31] is used in the project $LiveE!$ [18]. The project deploys a global infrastructure aiming at collecting and distributing environmental information, which includes 106 weather stations across 13 countries. [31] considers a dataset from 25 weather stations. These weather stations collect environmental data during a time

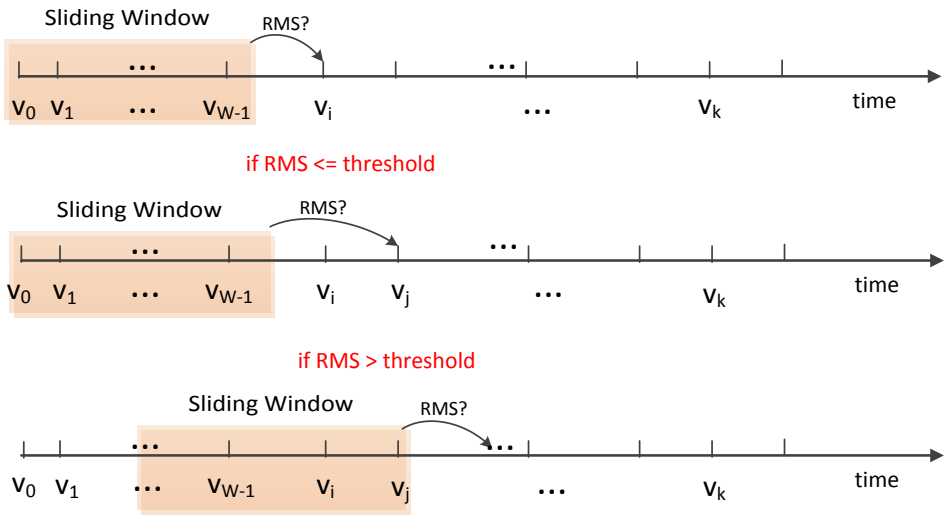


Figure 2.3: The introduction of A-ARMA on a sensor node.

window W, and then the stations determine the coefficients of the polynomial function with degree n that fits the collected data in a least squares method. And finally, the stations transmit the corresponding polynomial coefficients ($n+1$ coefficients) instead of raw data to the server, and the original time series can be recovered. Obviously, if the order n is lower than the number of data collected during the time window W, the overall data traffic can be significantly reduced. However, this method requires the stations to compute the coefficients, if the stations are changed as sensor nodes, the computation is too complex for them.

2.2.3 Compressing data aggregation

Compressing redundant information is the initial idea for aggregation, and researchers try to seek better way to compress the raw data. By doing so, it can reduce the energy consumption, and thus extend the lifetime of wireless sensor network. Most of the data compressing algorithms for internet [64, 65] are not feasible for WSNs. This is because the computation complexity, which is too high to implement on sensor nodes. Another reason is that those algorithms need high PC speed, whereas the MCU of sensor node cannot run the algorithms at all. Thus recently, low-complexity and light compression algorithms are proposed [66, 67] for WSNs.

Traditional Compressing Methods

To compress information, sensor nodes can drop redundant packets before transmitting to save energy. Coding by Ordering [68] is a representative of this idea, which

is introduced as part of Data Funneling Routing. The authors point out that the aggregator can drop data packets, and use other packets' permutation to express the dropped packet. Theoretically, when there are 2^7 sensor nodes, each node generates a 4-bit value, and an aggregator can process 100 packets, approximately 44% of packets can be dropped. However, if using Coding by Ordering, sensor nodes require a mapping table to store the packets' permutation. Obviously, as the number of sensor nodes increases, the size of table will increase exponentially.

Finding the redundancy and then dropping packets are investigated in Pipelined In-Network COmpression [69], denoted as PINCO. The raw data is stored at the buffer of aggregator for a certain duration. During this time, arrived data packet are compressed into one packet, and redundancies in the data packet will be removed. The compressed packet has a "shared prefix", including the suffix value (e.g. zero). When new packet arrives, the aggregator uses this value to check if the new packet is redundant (e.g. check if the suffix value of the new measurement is zero), if it is redundant, node drops this packet. PINCO is a simple compression scheme, however, the length of shared prefix may limit the algorithm efficiency. Moreover, large buffer in aggregator is not available due to the limited memory on a sensor node.

Compressive Sensing theory

The state-of-the-art compressing method is Compressive Sensing (denoted as CS) [70]. Shannon sampling theorem defines that the sampling rate should be more than twice of the maximum frequency of a signal. This sampling rate is a worst case bound. Compressive sensing [71] asserts that certain signals can be recovered from fewer samples than Shannon sampling used, by solving a programming optimization problem. Suppose that a signal $d \in \mathbb{R}^N$ can be represented as a sparse signal $x \in \mathbb{R}^N$ in orthonormal basis $\Psi \in \mathbb{R}^{N \times N}$, the signal can be recovered from M ($M \ll N$) measurements. The sampled signal via CS is presented as:

$$y = \Phi d + e = \Phi \Psi x + e \tag{2.15}$$

where $\Phi \in \mathbb{R}^{M \times N}$represents a sensing matrix and e is an unknown additive noise during acquisition.

Using CS needs two conditions: sparsity and incoherence. Sparsity means signal d should be sparse in some domain, and sparsity makes it possible to abstract the signal with small samples than the Shannon sampling theory used. We say a signal d is k-sparse in the Ψ domain if the number of non-zero coefficients are small and equal to k. Incoherence means the domain Ψ should be incoherence with sensing matrix

Φ. In general, the sensing matrix can be choose randomly, and random matrices are normally incoherent with any fixed basis [72]. Therefore incoherence between the sensing matrix Φ and transform basis Ψ can be achieved.

Restricted isometry property (RIP) is an usual criteria to detect if the sensing matrix is satisfied to recover the sparse signal, and due to RIP, we know when the number of measurements M satisfies:

$$M \geqslant const \cdot k \log \frac{N}{k} \tag{2.16}$$

$\mathcal{O}(k \log \frac{N}{k})$ random measurements are enough to recover a signal (when the signal is k-sparse), and in general, $M = 3k \sim 4k$.

The recovery procedure is a linear program, l_1 minimization is widely used for CS signal reconstruction, and can be expressed ($\beta \geqslant 0$):

$$\min_x \quad \frac{1}{2}\|\Phi\Psi x - y\|_2^2 + \beta\|x\|_1 \tag{2.17}$$

where defining the l_p norm of the vector $\boldsymbol{x}$ as

$$\|\boldsymbol{x}\|_p = (\sum_{p=1}^{N} |x_i|^p)^{\frac{1}{p}} \tag{2.18}$$

and N is the length of vector $\boldsymbol{x}$. To solve this problem in Eq. 2.17, there are several algorithms can be used, such as convex optimization algorithms or greedy algorithms [73].

Compressive Sensing in WSNs

Due to the characteristics of compressive sensing, the researchers find that CS is a good way to compress information in WSNs. Theoretically, CS can be used to compress spatial raw data in WSN, and CS theory shifts the energy consumption into the decoder (e.g. the sink node in WSN), which considerably solves or relieves the energy consumption on sensor nodes. Moveover, CS reduces the traffic (transmit measurements instead of raw data), which saves network capacity. Fig. 2.4 shows an example of CS theory in WSN, we can see that raw data d multiplying with an orthonormal matrix Ψ and a random sparse matrix Φ, will be a sparse data y, the sink can recover y to d by compressing sensing theory.

However, using CS for data aggregation in WSN also faces to several challenges, such as sensing matrix choice (how to choose the matrix to improve accuracy and energy efficiency), application implementation (how to use CS in real application?),

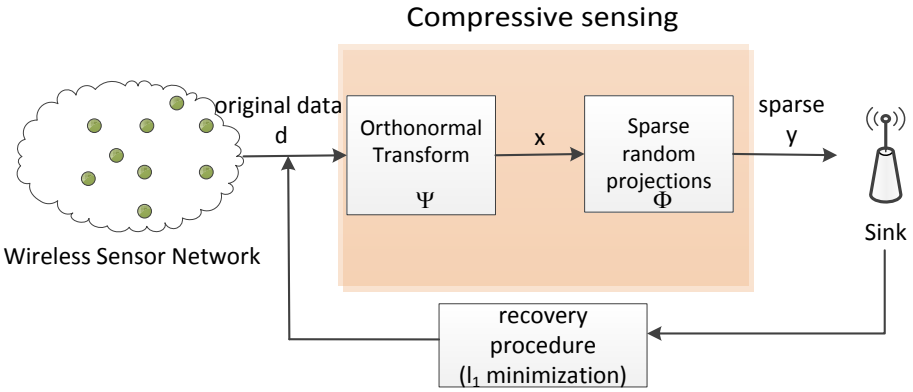


Figure 2.4: The procedure of compressive sensing in Wireless Sensor Network.

routing and compression (how to use CS considering with a routing protocol?), performance (if CS indeed improves the aggregation performance?).

As discussed in CS theory part, sensing matrix $\Phi(M \times N)$ is used to sample data, it is like a projection to connect data and random measurements. Generally, we use random generator to select a projection. While the authors in [74] concern on finding such a projection as little energy as possible. Sink node is in charge of determining new projection, and this projection should use less energy and get more information. In the beginning, each sensor node randomly sends its reading to the sink (sink has the whole view of the network). When the sink is not satisfied with the data, it determines the projection, sends the projection and waits for the reply. The main contribution of this paper is how to determine the projection. Suppose that p is the projection, the reduction of differential entropy $\triangle H(p)$ and energy $E(p)$ (measured by the number of transmissions) are used to determine which is a better projection. To choose the p which make $\frac{\triangle H(p)}{E(p)}$ maximum (lower energy consumption, higher entropy of projection p), they propose heuristics. However, such projection's generation method can be used only for small-scale network, otherwise the energy consumption may exceed expectation due to the projection dissemination.

The authors of [75] propose a simple projection method, it makes sensors use projection with the scheduled time slot. For example, nodes $1, 3, 5$ send the reading to sink in slot t_i, and the nodes $2, 4$ report in t_{i+1}. This method uses the fast fading wireless channels for generating random projections, which is simple to implement. However the recovery accuracy may not be guaranteed.

In terms of the application using CS, authors of [76] propose a compressive data gathering method from a large-scale wireless sensor network. They propose that data can be compressed in each transmission, which reduces the possibility of bottleneck and improves the network capacity. For example, without considering CS, sensor

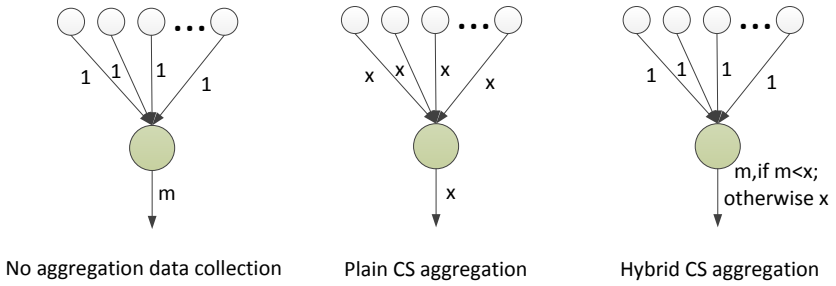


Figure 2.5: Illustration of plain CS aggregation and hybrid CS aggregation. The link labels correspond to the carried traffic.

node s_1 sends reading d_1 to node s_2, and s_2 transmits both its reading d_2 and the relayed reading d_1 to the next node. while in [76], it works like this: s_1 multiplies its reading d_1 with a random coefficient Φ_{i1}, and sends $\Phi_{i1}d_1$ to s_2, and similarly, s_2 will send $\Phi_{i1}d_1 + \Phi_{i2}d_2$ to the next node. Finally, the sink receives $\sum_{j=1}^{N} \Phi_{ij}d_j$ (N is the total number of nodes), a weighted sum of all readings. This process is repeated using M sets of different weights so that the sink will receive M weighted sums. With these sum, sink can recover data by using CS theory. In [33], the authors propose a compressive sensing algorithm to improve the recover accuracy, which introduce autoregressive AR model into the reconstruction of the sensed data.

The authors propose a CS framework that includes random projection, CS construction, and the sampling frequency feedback in [77]. Sampling frequency feedback means setting a parameter which enable the sensor to adjust its sampling rate to keep the reconstruction quality, which is called SRI (sampling rate indicator) feedback. The authors use L additional reserved data to evaluate the reconstruction performance. The sink compares the reserved data with the corresponding reconstructed data for calculating the reconstruction quality indicator (RQI). If the RQI is below (*resp.* above) the acceptable reconstruction quality, the sink will send an SRI message to make sensor increases (*resp.* decreases) the sampling.

[78] and [79] combine the compress sensing and PCA (Principal Component Analysis) to recovery the original data. PCA is a statistical procedure that uses an orthogonal transformation to convert a set of possibly correlated variables into a set of linearly uncorrelated variables (i.e. principal components). In [78, 79], the authors use mean of data $\bar{x}$ and covariance matrix $\widehat{\Sigma}$ as principal components to recover the original data, the accuracy is improved. This method firstly needs to collect enough data to compute the parameters, and then collects data as CS, thus it wastes some energy for data collecting procedure.

In terms of the relationship between routing and compression, the authors in [80]

present two aggregation mechanisms within CS, one is plain CS aggregation, another is hybrid CS aggregation, they are illustrated by Fig. 2.5. Plain aggregation denotes that the situation of forcing every link to carry x data, and hybrid CS aggregation means starting CS coding only when the outgoing data becomes no less than x. And the result shows that the hybrid CS aggregation is more energy efficient. In the meanwhile, performances of the two mechanisms are investigated in [81], they prove that applying CS naively (plain-CS) may not bring any improvement, and the hybrid-CS can achieve significant improvement in throughput.

Regarding to performance of CS, the capacity and delay of data gathering using are investigated in [82]. The gathering scheme is based on cell (they separate grid network as many cells, each cells have same number of nodes), the first step is that a cell head is designated to collect the data from the member nodes in the same cell, and second step is gathering from the column of the cells, and then forward to the sink. Under this scheme, the capacity can be considerably improved, and within TDMA, the delay is also bounded. However, this method looks only suitable for grid topology, and how to separate the cell is not mentioned.

2.3 Conclusion

In this chapter, we review the state-of-the-art aggregation works from two perspectives: aggregation structure and aggregation functions. Aggregation structure organises sensor nodes following a logical way to allow in-network aggregation, and we investigate: hierarchy-based structures, backbone-based structures and structure-free aggregation. Aggregation function is the computation part concerning how to aggregate information. Similarly, we review the associated literature by two types: forecasting aggregation and compressing aggregation.

By investigating the state of the art, we highlight that there are several issues have not been sufficiently discussed. First, some works construct aggregation structure relying on raw data (e.g. isoline aggregation), which makes these methods strongly depend on the raw data. When abrupt variation occurs in raw data, the performance of such works are affected. Secondly, the current aggregation functions are designed for targeted applications. For example, compressive sensing is more suitable for dense network, whereas A-ARMA keeps fixed model order: in both cases, if the network topology evolves or if the behaviour of the application changes, such aggregation functions will not be able to capture the new context. Thirdly, to the best of our knowledge, in area of sensor network, there is no work concerning the comparison of performances of different functions, and there is no guideline for how to select a function considering an application and targeting accuracy. These three comments will drive our motivations and contributions, which are discussed in the following chapter.

3

Networking-oriented metrics for data aggregation functions

Contents

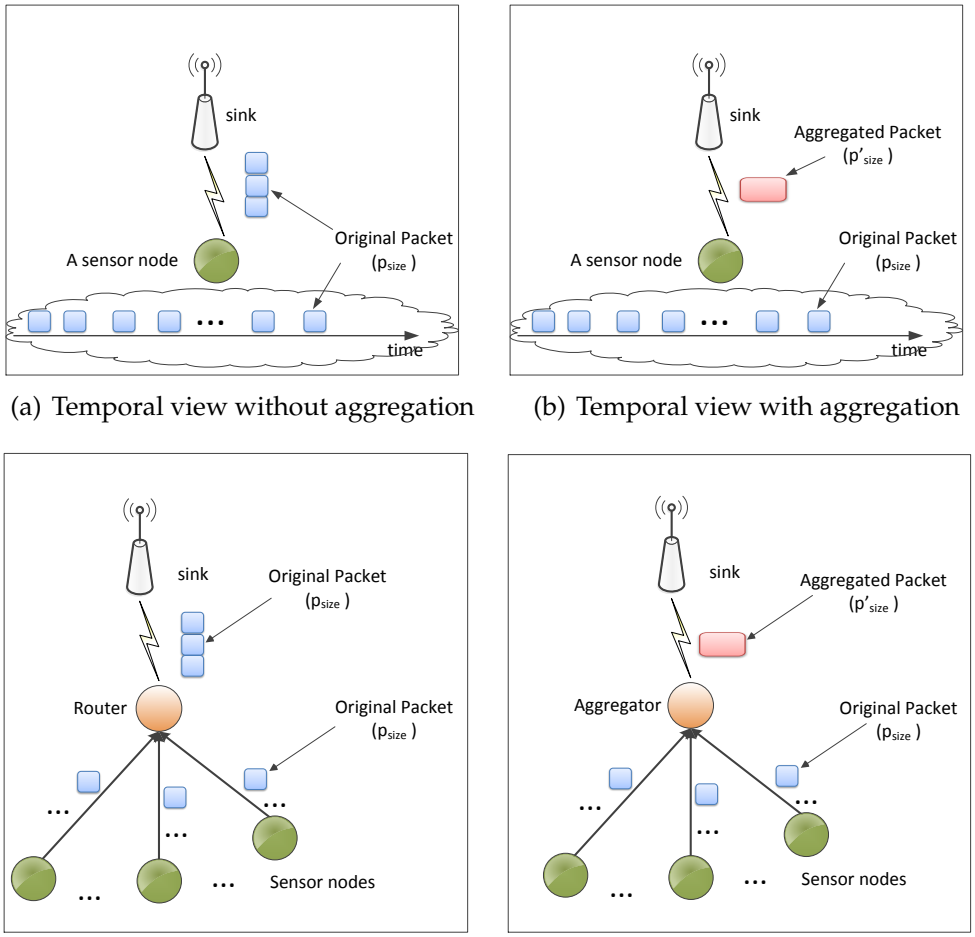
In wireless sensor networks, longer network lifetime and more network capacity are required in variety of applications, such as long-term monitoring. Thus the problems of energy and network capacity consumption are considered central to the sensor research theme. In previous studies, several researchers demonstrate routing and MAC (media access control) protocols [83, 84] can append feature of saving energy. However, under general assumptions, these protocols exhibit similar performances.

We know that the energy cost of a given protocol includes quite fixed cost and variable cost. The fixed one is the energy cost of transmitting data packets, variable one is related with the design of the protocol (such as energy cost of control packets in routing protocol). When the protocols own the same number of data packets, their fixed cost will be same, and the only difference is the variable cost. Actually, regarding routing and MAC protocols, they are responsible for transmitting a data packet efficiently to the destination. But they never consider whether the data packet is needed to be sent or not. While data aggregation focuses on this question, it reduces the redundant or correlated packets to save energy and capacity. What can be benefited by data aggregation, and how to evaluate the benefits at networking level are discussed in this chapter.

3.1 Networking-oriented metrics

As discussed in Sec. 1.2, temporal and/or spatial correlations exist in raw data of sensor nodes. Temporal correlation exists in data collected by one sensor node at different time instants, while spatial correlation occurs when the data are collected from local neighbouring sensor nodes. In Fig. 3.1, we give examples of temporal and spatial correlations, and show the impacts of data aggregation. From temporal view, without aggregation, shown by Fig. 3.1(a), a sensor node generates original packets (assuming the original data packets have same size as p_{size}) to sink without any computation. While with aggregation, shown by Fig. 3.1(b), the sensor node aggregates several original data packets into an *aggregated packet*, and only transmits the aggregated packet to sink. Regarding to spatial view, without aggregation, shown by Fig. 3.1(c), sensor nodes generate original packets to a router, and the router forwards all the packets to the sink. While with aggregation, shown by Fig. 3.1(d), the router becomes aggregator, it aggregates the received packets as an aggregated packet, and transmits the packet to sink.

Thus from Fig. 3.1, we can see the differences between the situations with aggregation and without aggregation. The first difference is that the number of transmitted packets changes. With aggregation, the number of aggregated packets is lower than (or equal to) the number of original packets. The second difference is that packet size changes. With aggregation, the aggregated packet size is p'_{size}, while the original packet size is p_{size}. If we are able to compute how many packets are saved by an aggregation function, we will be able to conclude about the energy efficiency of the aggregation function. Similarly, if we are able to compute the packet size change due to an aggregation function, we will be able to detect the effect of this function on the



(a) Temporal view without aggregation (b) Temporal view with aggregation

(c) Spatial view without aggregation (d) Spatial view with aggregation

Figure 3.1: Effects of data aggregation on Wireless Sensor Networks.

network capacity. Thus we claim that it is necessary to propose new metrics to evaluate the impacts of aggregation on energy efficiency and network capacity. Next, we define the networking-oriented metrics.

Definition 3.1.1. *Aggregation Ratio, denoted as ω, is the ratio of aggregated packets respect to the total packets generated.*

Thus, aggregation ratio $\omega = \frac{n}{N}$, where n is the number of aggregated packets, while N is the total packets generated, as shown in Fig. 3.2. Using aggregation, only $\omega \cdot N$ aggregated packets are transmitted. If $\omega \geq 1$, it means the aggregation scheme does not save any packets. Thus, the range of aggregation ratio is $\omega \in (0, 1]$, the smaller aggregation ratio ω is, the smaller number of aggregated packets is, the more original packets are aggregated.

Definition 3.1.2. *Packet size coefficient, denoted as λ, presents the packet size change due to*

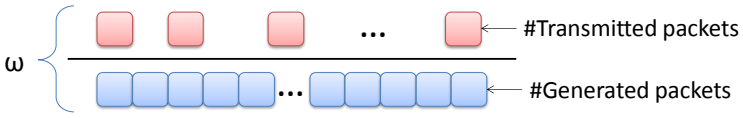


Figure 3.2: Illustration of aggregation ratio ω.

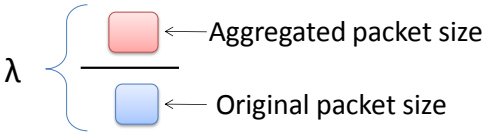


Figure 3.3: Illustration of packet size coefficient λ.

the aggregation function.

Packet size coefficient is the rate of packet size change. $\lambda = \frac{p'_{size}}{p_{size}}$, where p'_{size} is the size of aggregated packet, and p_{size} is the size of original packets that have been aggregated, as shown in Fig. 3.3. Note, packet size coefficient λ is an useful metric to evaluate the network capacity when considering aggregation. If an aggregation function can process more original packets into one aggregated packet, we consider it saves network capacity better. We define the packet size coefficient range as $\lambda \geq 1$.

Using these networking-oriented metrics, we can evaluate the impacts of data aggregation on WSNs. In the next, we show the impacts on routing and MAC layers.

3.2 Impacts of data aggregation on routing layer

In this section, assuming an ideal MAC layer, we consider the energy consumptions at routing layer. We will discuss the impacts on MAC layer in the next section.

3.2.1 Basic topology analysis

To analyse the impacts of data aggregation on routing layer, firstly we consider basic topologies: 1-hop network, 1D network and 2D network. In 1-hop network, sensor nodes (assuming 5 sensors) are directly connected to the sink. In 1D network, sensor node communicates only with its direct neighbours. In 2D network, we use a grid network. These topologies are shown in Fig. 3.4.

For numerical results, we consider a data sheet from real sensor node. Giving the power of transmission is $P_{tx} = 62.5$mW, the power of reception is $P_{rx} = 53.7$mW, and assuming an ideal scheduling leads to no collision and no interfere (ignoring the energy for over-hearing). The data packet size is $p_{size} = 36$bytes.

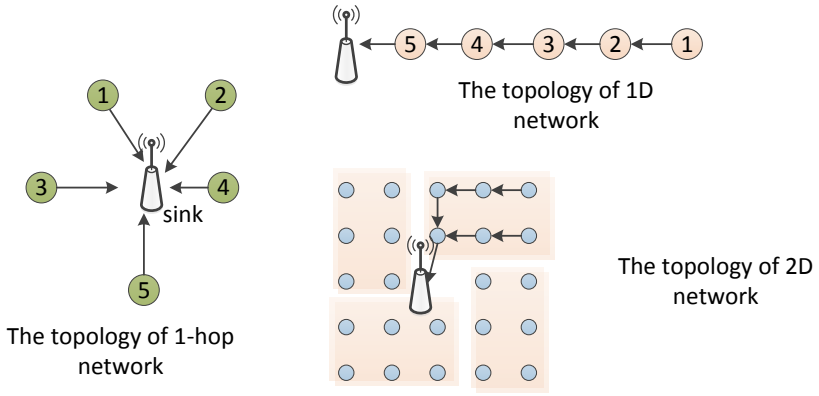
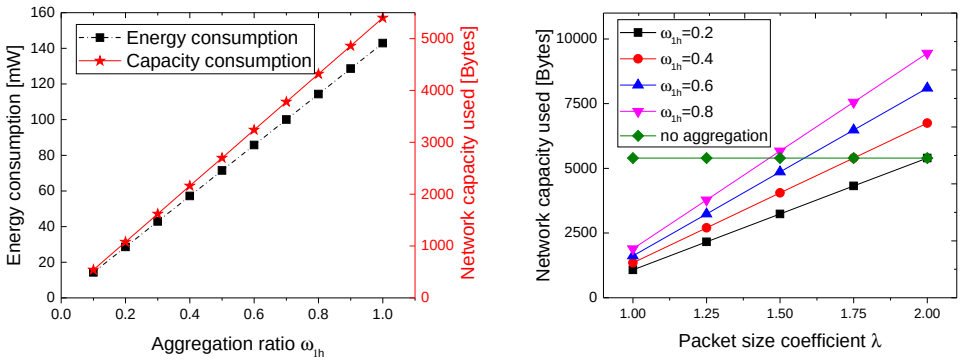


Figure 3.4: Several basic topologies.

In 1-hop network, assuming each sensor node generates 30 packets, and a given aggregation function with aggregation ratio (*resp.* packet size coefficient) is ω_{1h} (*resp.* λ_{1h}). Without aggregation, the energy consumption for 1-hop network can be written as $E^{1h} = 5 \cdot 30 \cdot P_{tx}$, that means 5 nodes generate $5 \cdot 30$ packets. Similarly, the network capacity consuming is $C^{1h} = 5 \cdot 30 \cdot p_{size}$. While with aggregation, energy consumption changes as $E^{1h}_{agg} = 5 \cdot 30 \cdot \omega_{1h} \cdot P_{tx}$, where $5 \cdot 30 \cdot \omega_{1h}$ is the number of aggregated packets. Using aggregation, the network capacity used is $C^{1h}_{agg} = 5 \cdot 30 \cdot p_{size} \cdot \omega_{1h} \cdot \lambda_{1h}$. We plot the impact of aggregation ratio on 1-hop network in Fig. 3.5(a), here we set packet size coefficient $\lambda_{1h} = 1$. We can see as aggregation ratio decreases, the energy consumption and network capacity consumption all decrease. Comparing to the situation without aggregation (i.e. $\omega_{1h} = 1$), energy consumption and capacity used within aggregation are lower. The smaller aggregation ratio is, the more energy is saved.



(a) Energy and network capacity with different ω_{1h} (here $\lambda_{1h} = 1$).

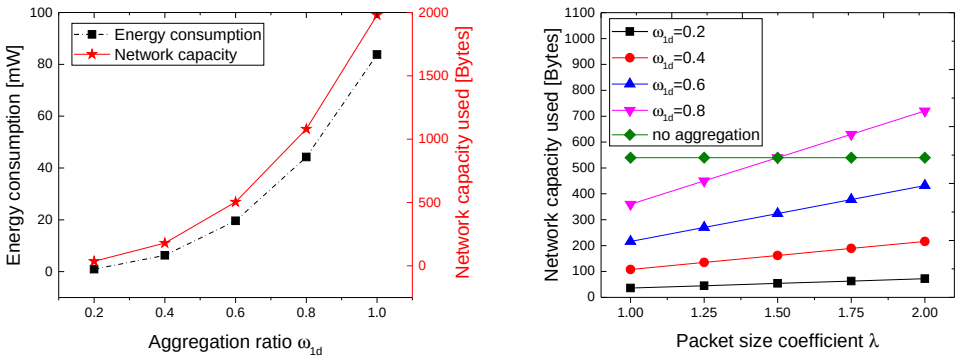
(b) Network capacity with different ω_{1h} and λ_{1h}.

Figure 3.5: Energy and network capacity consumptions in 1-hop network.

In Fig. 3.5(b), we can note that aggregation ratio ω_{1h} and packet size coefficient

λ_{1h} all impact the network capacity. When $\omega_{1h} = 0.2$, the network capacity has been saved even though λ_{1h} equals 2. As the increase of ω_{1h}, the 1-hop network consumes more capacity. Especially, under our assumption, when $\omega_{1h} > 0.6$ and $\lambda_{1h} > 1.5$, the network consumes more capacity than the situation without aggregation. That is to say, there should is a trade-off between ω and λ to guarantee that the network capacity can be saved.

In the case of 1D network, if there is no aggregation, the energy consumption is $E^{1D} = \sum_{i=1}^{5}\{[i \cdot P_{tx} + (i-1) \cdot P_{rx}]\} \triangleq \sum_{i=1}^{5} f_{hop}(i)$, where i is the number of packets, and we define $f_{hop}(i) = \{[i \cdot P_{tx} + (i-1) \cdot P_{rx}]\}$. Setting aggregation ratio $\omega_{1D} \in [0.2, 1]$, thus the energy consumption with aggregation for 1D network is $E^{1D}_{agg} = \sum_{i=1}^{\lceil 5 \cdot \omega_{1d} \rceil} f_{hop}(i)$. Similarly, the network capacity used without aggregation can be formulated as $C^{1D} = \sum_{i=1}^{5} i \cdot p_{size}$; the capacity used with aggregation is $C^{1D}_{agg} = \sum_{i=1}^{\lceil 5 \cdot \omega_{1d} \rceil} i \cdot p_{size} \cdot \lambda_{1d}$. Fig. 3.6(a) shows the energy consumption with different ω_{1d} (here $\lambda_{1d} = 1$), and the details of the capacity consuming with ω and λ are shown in Fig. 3.6(b).



(a) Energy and network capacity with different ω_{1d} (here $\lambda_{1d} = 1$).

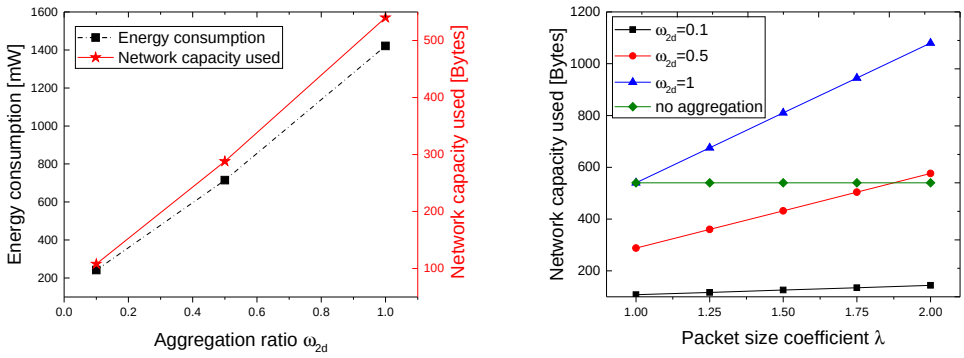
(b) Network capacity with different ω_{1d} and λ_{1d}.

Figure 3.6: Energy and network capacity consumptions in 1D network.

In Fig. 3.6(a), we can see that the smaller aggregation ratio ω_{1d} is, the more energy and capacity saves. Similar as in 1-hop network, when the ω_{1d} or λ_{1d} are beyond certain range, the network cannot save capacity, even the network consumes more capacity than the situation without aggregation (see Fig. 3.6(b)).

For 2D network, each sensor node routes packet to the sink using a shortest path protocol (e.g., Dijkstra). We can divide the traffic flows into 4 parts (see 2D network in Fig. 3.4). For each part, the topology can be seen as a combination of 1D networks, i.e. two 1D topologies of 3 nodes in a line. The energy consumption for each part

of 2D network without aggregation is $E^{2D} = 2 \cdot \sum_{i=1}^{3} f_{hop}(i) + 3 \cdot (P_{tx} + P_{rx})$, correspondingly, the network capacity consuming of each part without aggregation is $C^{2D} = 2 \cdot \sum_{i=1}^{3} i \cdot p_{size} + 3 \cdot p_{size}$. If an aggregation function is used for 2D network, the energy consumption for each part can be formulated as $E^{2D}_{agg} = 2 \cdot \sum_{i=1}^{\lceil 3 \cdot \omega_{2d} \rceil} f_{hop}(i) + \lceil 3 \cdot \omega_{2d} \rceil \cdot (P_{tx} + P_{rx})$, meanwhile, the capacity consumption for each part is $C^{2D}_{agg} = 2 \cdot \sum_{i=1}^{\lceil 3 \cdot \omega_{2d} \rceil} i \cdot p_{size} \cdot \lambda_{2d} + (\lceil 3 \cdot \omega_{2d} \rceil) \cdot p_{size} \cdot \lambda_{2d}$. Similar with 1D network and 1-hop network, the aggregation ratio ω_{2d} is a fraction, no more than 1, thus the energy can be saved (see Fig. 3.7(a)). Besides, the network capacity can be saved if λ_{2d} and ω_{2d} are in certain range (see Fig. 3.7(b)).



(a) Energy and network capacity used with different ω_{2d} (here $\lambda_{2d} = 1$).

(b) Network capacity used with different ω_{2d} and λ_{2d}.

Figure 3.7: Energy and network capacity consumptions in 2D network.

Considering the above results (Fig. 3.5, Fig. 3.6 and Fig. 3.7,), we found that efficient aggregation function can save energy and network capacity. But when the networking-oriented metrics exceed some range, the associated aggregation function is not so powerful to save energy and capacity. Thus, there should be a trade-off between ω and λ, and we will discuss in Sec. 3.4.

3.2.2 Routing protocols analysis

In the context of wireless sensor networks, many researchers think that routing protocols should save energy and extend the network capacity [85, 86], except the basic function of routing. However, by modelling the energy consumptions for several routing protocols, we find that data aggregation is a more efficient way to save energy.

Several analytical methods are proposed to model energy consumption in battery-powered WSNs. We use the model proposed in [87], to analyse the energy cost by routing protocols. The model supposes a formula as:

$$E_b = E_{Tx} + \Gamma \cdot E_{Rx} \tag{3.1}$$

where E_{Tx} is the energy consumed to transmit 1 bit, E_{Rx} is the energy consumed to receive the same bit at targeted receivers, Γ is the neighborhood size degree. Thus, E_b denotes the total energy cost of a single bit in 1-hop communication, including transmission and reception costs. To model the energy consumption of routing protocols, we use parameters as: f_{xx} is the number of packets of type xx, S_{xx} is the corresponding size of packet xx in bits. We introduce the energy model following, which is an extension of [88].

OLSR (Optimized link state routing protocol) [89] is one of the most popular proactive routing protocol that is used in MANETs. It consists to periodically exchange topology information in *Topology Control (TC)* messages in order to establish a route to any destination. Multipoint relays (MPR) concept is used to optimize TC message flooding [90]. Thus in OLSR, there are two types of control messages: TC message (a global broadcast message), and *hello* message. **Energy Cost of *hello* message** is expressed as:

$$E_H^{OLSR} = f_H^{OLSR} \cdot S_H^{OLSR} \cdot E_b \tag{3.2}$$

where f_H^{OLSR} is the number of hello messages, and S_H^{OLSR} is the size of the hello message, E_b is given by Eq. 3.1. Similarly, **Energy cost of *TC* message** is expressed as:

$$E_{TC}^{OLSR} = f_{TC}^{OLSR} \cdot S_{TC}^{OLSR} \cdot E_b \cdot B_{TC}^{OLSR} \tag{3.3}$$

where B_{TC}^{OLSR} denotes the ratio of MPRs to neighbours.**Energy cost of Data message** can be expressed as follow:

$$E_D^{OLSR} = f_D^{OLSR} \cdot (S_D^{OLSR} + S_{ACK}^{OLSR}) \cdot L_{OLSR} \cdot N_{Tx} \cdot E_b \tag{3.4}$$

where L_{OLSR} is the average number of hops in protocol OLSR, N_{Tx} denotes the average number of retransmissions until the packet is successfully transmitted.

With the above information, the **energy consumption of OLSR protocol** can be calculated, that is the sum of all energy costs:

$$E_{OLSR} = E_H^{OLSR} + E_{TC}^{OLSR} + E_D^{OLSR} \tag{3.5}$$

GPSR (Greedy perimeter stateless routing) [91] is a geographic routing. In GPSR,

nodes periodically send *hello* messages in order to get its 1-hop nodes locations. To send data to destination, node selects its neighbor which will minimize the distance to the destination. Therefore, similar as energy cost of OLSR, the **energy consumption of GPSR protocol** is expressed as follows:

$$E_{GPSR} = E_H^{GPSR} + E_D^{GPSR} + E_{PU} \tag{3.6}$$

where $E_{PU} = f_{PU} \cdot S_{PU}$ is energy cost of location informations, and the computation of E_D^{GPSR} is similar to the computation in OLSR (Eq. 3.4).

In simple gradient-based routing (GBR) [92], only one control message is periodically broadcasted in the network to compute the gradient of nodes: *Advertisement message*(ADV). **Energy cost of ADV message** can be expressed as:

$$E_{ADV} = f_{ADV}^{GBR} \cdot S_{ADV}^{GBR} \cdot E_b \cdot B_{ADV}^{GBR} \tag{3.7}$$

where B_{ADV} is the number of nodes that broadcast the ADV message. In a simple GBR (without optimisation like Neighbour Elimination Scheme [93]), B_{ADV} is equal to the number of nodes in the network. Thus, **energy consumption of GBR** is:

$$E_{GBR} = E_{ADV}^{GBR} + E_D^{GBR} \tag{3.8}$$

A simple random walk (SRW) [94] routing is defined as: firstly, nodes use *hello* message to set their 1-hop neighbours information; secondly, nodes randomly select the next hop among its 1-hop neighbours before relaying the data to the selected node. The **energy consumption of SRW** can be formulated as:

$$E_{SRW} = E_H^{SRW} + E_D^{SRW} \tag{3.9}$$

By studying the energy costs of such routing protocols (Eq. 3.5, Eq. 3.6, Eq. 3.8 and Eq. 3.9), we find that, although these protocols use different routing strategies (proactive routing, gradient routing or geographic routing), the formation of energy consumption is similar (the differences are the control messages), and all the energy consumptions are related to E_D. E_D is the energy cost for data packet. Thus with less data packets, all the energy consumptions can be reduces. Note that data aggregation is used to reduce the number of data packets.

If we assume that there is an aggregation function with a given aggregation ratio ω and a given packet size coefficient λ, thus the energy cost of the previous routing

protocols are expressed as follows:

$$
\begin{aligned}
E^{agg}_{OLSR} &= E_H + E_{TC} + \omega\lambda \cdot E_D \\
E^{agg}_{GPSR} &= E_H + \omega\lambda \cdot E_D + E_{PU} \\
E^{agg}_{GBR} &= E_{ADV} + \omega\lambda \cdot E_D \\
E^{agg}_{SRW} &= E_H + \omega\lambda \cdot E_D
\end{aligned}
\tag{3.10}
$$

As we defined, aggregation ratio ω is no more than 1, packet size coefficient λ is related with the given aggregation function, the total energy cost will decrease if $\omega \cdot \lambda \leq 1$. Especially, when the energy cost of transferring data is the main part in a routing protocol, the energy can be saved with data aggregation.

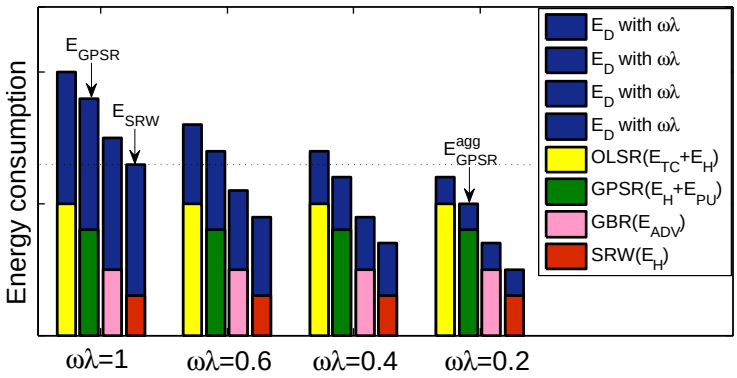


Figure 3.8: Energy consumption comparison for different routing protocols with and without aggregation.

To compare the energy cost of the above routing protocols, we separate their energy consumption as two parts: energy cost for data message and energy cost for control messages. Also, we assume that these protocols consume the same energy for data message.We plot a schematic diagram of energy consumptions for the four routing protocols in Fig. 3.8, and different colors are used to show the energy for data message (blue) and energy for control message (other colors in Fig. 3.8). We first shou the energy consumption of the routing protocols without aggregation (the first group of bars when $\omega\lambda = 1$), and then we consider an aggregation function holds $\omega\lambda$ as 0.6, 0.4 and 0.2 respectively. The energy consumption of data messages decreases respectively, and the energy consumption of the protocols decreases according to Eq. 3.10. Because ω represents the ratio of aggregated packets to original packets, if $\omega = 0.2$, it means that only 20% packets are transferred. Taking GPSR as an example, without aggregation, GPSR (marked by E_{GPSR} in Fig. 3.8) consumes more energy than SRW (E_{SRW}) even though they consume same energy in data message. Considering an aggregation function with $\omega\lambda = 0.2$, the energy cost of GPSR (E^{agg}_{GPSR}) is reduced

because less data packets are needed to be transmitted, and the energy consuming is less than the energy cost of SRW when there is no aggregation. It means that an appropriate aggregation scheme is more useful and efficient to reduce the energy cost than choosing different routing protocols. If we can select an efficient aggregation function regardless which routing protocols, more energy can be saved since the energy cost of data packets are reduced.

3.3 Impacts of data aggregation on MAC Layer

MAC protocols [84, 95] are used to share the medium and avoid collisions between neighbours. Generally, MAC protocols for WSN usually use a duty cycle mechanism to save energy. Since the receiving, sending and listening energy costs are approximately the same for usual radio chips, the way to save energy is to turn off the radio (e.g. switch to sleep mode).

Fig. 3.9 shows the different MAC mechanisms (BMAC[96], XMAC[97], SMAC[98]), which describes the process of sender (*resp.*receiver) successfully sends (*resp.* receives) a data packet using the three MAC protocols.

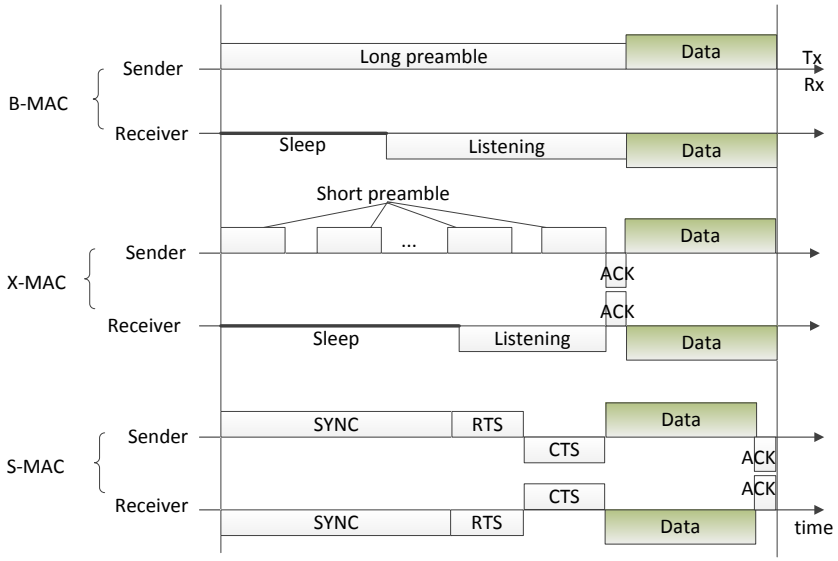


Figure 3.9: Schematic comparison of the timelines between B-MAC, X-MAC and S-MAC.

We present our model notations in Tab. 3.1. To compare the energy consumption of the three MAC protocols more clearly, we assume that the total time of a successful transmission is the same for them (assuming 75ms), and their common notations have same value (such as d_{tx} d_{rx} and d_s for three of them, d_{ACK} for XMAC and SMAC).

Meanwhile, the data packet has same size and same duration, 36bytes, and 15ms (for a data rate of 19.2kb/s).

Table 3.1: Model notations and target values.

		Target Value		
Parameter	Definition	BMAC	XMAC	SMAC
P_{tx}[1]	Power required for transmission	62.5	62.5	62.5
P_{rx}	Power required for receive	53.7	53.7	53.7
P_s	Power required for sleep	0.02	0.02	0.02
d_p[2]	Preamble duration	60	NA[3]	NA
d_{sp}	Short preamble duration	NA	7.8	NA
d_s	Sleep duration	20	42.8	NA
d_l	Listening duration	40	10	NA
d_{data}	Data transmission duration	15	15	15
d_{ACK}	Acknowledgement duration	NA	7.2	7.2
d_{SYNC}	Synchro. duration	NA	NA	25
d_{RTS}	RTS duration	NA	NA	13.9
d_{CTS}	CTS duration	NA	NA	13.9
α	short preamble repetition required in XMAC	NA	6	NA

[1] the unit for P_{xx} is mW,
[2] the unit for d_{xx} is ms,
[3] NA denotes non applicable,
[4] Assuming in an ideal PHY Layer.

BMAC [96] is an asynchronous media access control protocol, relies on preamble sampling. Idle listening is reduced in asynchronous protocols by shifting the burden of synchronization to the sender. When a sender has data, the sender transmits a preamble that is at least as long as the sleeping period of the receiver. The receiver will wake up, detect the preamble, and stay awake to receive the data (see BMAC in Fig. 3.9). Thus for B-MAC, successfully transmitting a packet in one pair of sender-receiver consumes E_{BMAC}, which includes preamble energy ($P_{tx}d_p$), sending data energy ($P_{tx}d_{data}$), sleep energy ($P_s d_s$), listening energy ($P_{rx}d_l$) and receiving data energy ($P_{rx}d_{data}$):

$$E_{BMAC} = P_{tx}d_p + P_{tx}d_{data} + P_s d_s + P_{rx}d_l + P_{rx}d_{data} \tag{3.11}$$

We can see that the energy consumptions of BMAC can be separated as two parts:

energy cost of data (E^B_{data}) and configurable part E_B (including long preamble, listen and sleep energy), denoted as $E_{BMAC} \triangleq E^B_{data} + E_B$, where

$$E^B_{data} = P_{tx}d_{data} + P_{rx}d_{data}$$

and

$$E_B = P_{tx}d_p + P_s d_s + P_{rx}d_l$$

To obtain the numerical results, we list the values in Tab. 3.1. For BMAC, the long preamble is $d_p = 60$ms (transmission duration 75ms, and data duration is equal to 15ms). Meanwhile, in BMAC, receiver randomly wakes up, thus we assume $d_l = 40ms$ and $d_s = 20ms$.

XMAC [97] is an extension of B-MAC. X-MAC employs a strobed preamble approach by transmitting a series of short preamble packets, each containing the ID of the receiver. The series of short preamble packets approximates a continuous preamble. Small pauses between preamble packets permits the potential receiver to send an acknowledgment that stops the sequence of preamble packets (see XMAC in Fig. 3.9). X-MAC uses strobe preamble to reduce the energy consumption of the long preamble, thus it has a expected number of iterations (denoted as α) required to determine the preamble frequency, the energy of transmitting a packet using X-MAC E_{XMAC} can be expressed as:

$$\begin{aligned} E_{XMAC} &= (P_{tx}d_{sp} + P_{rx}d_{ACK}) * \alpha + P_{tx}d_{data} \\ &\quad + P_{rx}d_l + P_s d_s + P_{rx}d_{data} + P_{tx}d_{ACK} \\ &\triangleq E_X + E^X_{data} \end{aligned} \tag{3.12}$$

Similar to BMAC, the compositions of energy cost for XMAC are energy of data E^X_{data} and configurable part E_X. We assume $d_{sp} = 7.8ms$ and $d_{ACK} = 7.2$ms for XMAC as it is proposed in [97]. To guarantee the receiver can listen a whole short preamble, the listening duration of XMAC should meets $d_{sp} \leq d_l < 2 \cdot d_{sp}$, so we set $d_l = 10ms$ in XMAC. Thus $d_s = 42.8$ms, because under our assumption (same time of transmission), the sleep duration plus ACK duration plus listen duration (in XMAC) should equal the long preamble duration (in BMAC). And correspondingly, we set the expected number of iterations $\alpha = 6$.

SMAC [98] is a low power RTS-CTS scheme for wireless sensor networks inspired by 802.11. BMAC is a synchronization MAC protocol. A node using SMAC periodically sleeps, wakes up, listens to the channel, and then returns to sleep. At the beginning of each active period, nodes exchange synchronization information. Following the SYNC period, data may be transferred for the remainder of the active period using

RTS-CTS (see SMAC in Fig. 3.9). Before sending packets to the receiver, SMAC needs to synchronize the neighbors. The energy consumption is denoted as E_{SMAC}, which can be formulated as:

$$\begin{aligned} E_{SMAC} &= P_{tx}d_{SYNC} + P_{tx}d_{RTS} + P_{tx}d_{data} + P_{rx}d_{CTS} \\ &+ P_{rx}d_{ACk} + P_{rx}d_{SYNC} + P_{tx}d_{CTS} \\ &+ P_{rx}d_{data} + P_{tx}d_{ACk} \\ &\triangleq E_S + E^S_{data} \end{aligned} \tag{3.13}$$

Similarly, the energy consumption of SMAC is composed of data energy E^S_{data} and other part E_S (including synchronization energy and RTS-CTS energy). Analysing the schematic diagram of SMAC which is illustrated in Fig. 3.9, the active period $75ms$ is used for SYNC, RTS, CTS, data and ACK durations. According to [97], we set $d_{ACK} = 7.2$ms (same to XMAC), and assume RTS and CTS need same time, thus we set $d_{SYNC} = 25$ms and $d_{RTS} = d_{CTS} = 13.9$ms.

Supposing this pair of sender-receiver communicates 30 packets, with all the target values (in Tab.3.1), we calculate the energy consumptions for the three protocols (by Eq. 3.11, Eq. 3.12 and Eq. 3.13), and the results are presented in Fig. 3.10 (shown by blue bars). We can see that the energy consumption among different protocols are different. SMAC consumes more energy than others because synchronization process and RTS-CTS mechanism all spend energy. Comparing to BMAC, XMAC saves more energy because the short preamble can save some energy.

Assuming an aggregation function which holds $\omega\lambda = 0.2$, the energy consumption for the three protocols can be formulated as:

$$\begin{aligned} E^{agg}_{BMAC} &= E_B + \omega\lambda E^B_{data} \\ E^{agg}_{XMAC} &= E_X + \omega\lambda E^X_{data} \\ E^{agg}_{SMAC} &= E_S + \omega\lambda E^S_{data} \end{aligned} \tag{3.14}$$

The results of Eq. 3.14 are also shown in Fig. 3.10 (shown by yellow bars), we can see the energy consumption of all protocols reduce because the energy of data transferring reduce. Taking SMAC as an example, without aggregation, SMAC consumes much more energy comparing to XMAC. While with aggregation, the energy cost of SMAC substantially decreases. That is to say, if there is an efficient aggregation method, there is no need to consider which MAC protocol can save more energy, because any protocol can achieve better performance coupled with an aggregation scheme.

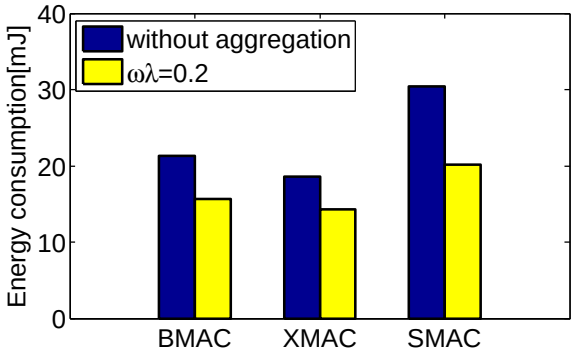


Figure 3.10: Energy consumption comparison for different MAC protocols with and without aggregation, where assuming $\omega \cdot \lambda = 0.2$.

3.4 The trade-off between networking-oriented metrics

As highlighted above, data aggregation in WSNs can save energy regardless of which routing or MAC protocols, and it can save the network capacity. All of the benefits are derived from the feature of data aggregation. Data aggregation discovers the correlations of different packets (the correlated packets, redundant packets), and then essentially reduce the number of packets to save both energy and network capacity.

However, as illustrated in Fig. 3.1, we know that aggregation functions may change the aggregated packet size. Also, under several situations, a given aggregation function may lead to the node consume more capacity (Sec. 3.2.1). Thus, in this section, we discuss the trade-off between aggregation ratio and packet size coefficient.

Supposing that the sensor nodes in a network generate N packets, where the average packet size is p bits, the energy requested to transmit 1 bit is E_{bit}. Thus, to transmit N packets, the energy consuming is:

$$E = N \cdot p \cdot E_{bit} \tag{3.15}$$

If considering an aggregation function which has aggregation ratio ω, thus the energy consumption with aggregation is:

$$E_{agg} = \omega \cdot N \cdot p \cdot \lambda \cdot E_{bit} \tag{3.16}$$

if $E_{agg} \leq E$, it illustrates that the aggregation function leads to decrease energy consumption, i.e.

$$\begin{aligned} \omega \cdot N \cdot p \cdot \lambda \cdot E_{bit} &\leq N \cdot p \cdot E_{bit} \\ \omega \cdot \lambda &\leq 1 \end{aligned} \tag{3.17}$$

This is to say, if an aggregation function has an aggregation ratio ω and a packet size coefficient λ, it can save energy as much as $(1 - w \cdot \lambda)$.

Similarly, without aggregation, the network capacity (C) and link capacity (C^{link}) can be formulated as:

$$\begin{aligned} C^{link} &= N \cdot p, \\ C &= \sum_{\forall link} C^{link} \end{aligned} \tag{3.18}$$

If considering an aggregation function, the aggregated packets is $\omega \cdot N$, and the aggregated packet size is $\lambda \cdot p$, thus the aggregated maximum link capacity and network capacity are given by:

$$\begin{aligned} C^{link}_{agg} &= \omega \cdot N \cdot p \cdot \lambda \\ C_{agg} &= \sum_{\forall link} C^{link}_{agg} \end{aligned} \tag{3.19}$$

Also, the link capacity can be saved as much as $1 - \omega \cdot \lambda$.

To illustrate Eq. 3.17 more clearly, we set $\omega \in [0, 1)$, $\lambda \in [1, 2]$, and present the result in Fig. 3.11. We plot λ as a function of ω in this figure, and use different colors to describe the value of $1 - \omega \cdot \lambda$. That is to say: we use warmer color (e.g. red) to denote that the value of $1 - \omega \cdot \lambda$ is higher, and colder color (e.g. blue) to show the value is lower. In this way, we can separate Fig. 3.11 into 3 areas. Specifically, if $\omega \leq 0.2$, no matter how to change λ from 1 to 2, the energy and capacity savings can reach about 80% (in area 1). If $\lambda \geq 1.8$ and $w = 0.4$ (top of area 2), the energy and capacity savings can reach 40%. While when $\lambda \geq 1.8$ and $\omega \geq 0.5$, the energy and capacity savings are only about 20% (top of area 3). We need to highlight that in the range of area 3, the ability of saving energy and capacity is not optimistic (just $\leq 20\%$). Thus, if ω and λ of an aggregation function are in area 1 (left part of Fig. 3.11, red color), the function can save more energy and capacity. While when them are in the area 3, it means the the associated function can not save energy and capacity too much. Therefore, for a given aggregation function, we should investigate the trade-off between the aggregation ratio ω and packet size coefficient λ firstly, in order to achieve better performance of saving energy and capacity.

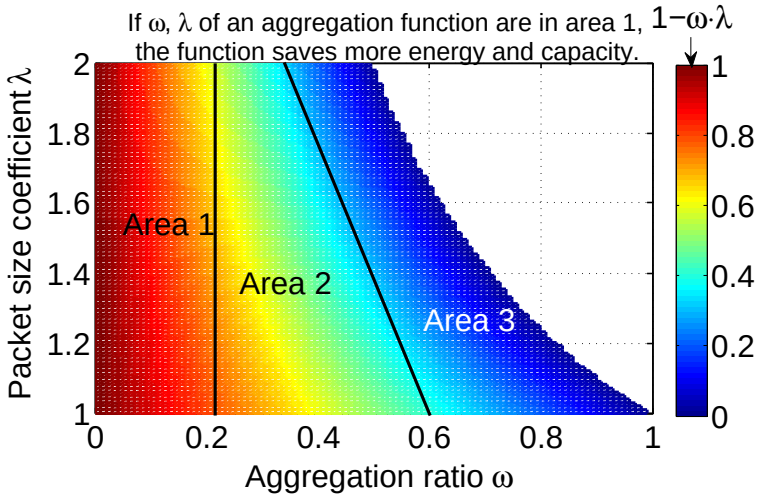


Figure 3.11: The trade-off between aggregation ratio ω and packet size coefficient λ.

3.5 Conclusion

In this chapter, we highlight the benefits taken by data aggregation in wireless sensor networks. Firstly, we propose two new networking-oriented metrics, and then we use them to evaluate the aggregation impacts on routing, MAC protocols. Finally, we discuss the trade-off between the new metrics.

Networking-oriented metrics include *aggregation ratio* and *packet size coefficient*. Aggregation ratio ω is the ratio of aggregated packet number to generated packet number, which is used to evaluate the energy saving by an aggregation function. Packet size coefficient λ is the ratio of aggregated packet size to original packet size, it is used to measure the capacity change due to an aggregation function.

With these two metrics, we investigate the impact of an aggregation function on routing and MAC protocols. We find that, aggregation function focuses on reducing the number of data packets; with less data packets, energy consumptions of all protocols reduce correspondingly. Clearly, data aggregation is a fundamental way to save energy due to less data packets regardless which routing or MAC protocols are. Meanwhile, since ω and λ impact energy and network capacity together, it is better to evaluate an aggregation function using two metrics in order to select a better aggregation function.

4

Similar-evolution based data aggregation

Contents

Several aggregation works benefit from data collecting structure, and the structure is organized based on similar raw data. However, by observing real datasets, we find that abnormal data often appears, while the data evolution is more stable. Therefore, to avoid the impact of instability of raw data, we propose a data-independent aggregation scheme: Similar-evolution based data aggregation (denoted as Simba). Our proposal organizes sensor nodes which own similar evolutions into a group to perform data aggregation, targeting at improving the accuracy and saving more energy.

4.1 Related works

As the literature [53, 54] that we already reviewed in Sec. 2.1, several recent aggregation schemes [99, 100] also highlight that similar raw data can be used to build groups of sensor nodes, and then perform data aggregation. The authors of [99] proposed a characteristic correlation to cluster nodes to achieve data aggregation. The characteristic correlation is defined by the data value and gradient (the difference between consecutive data), which is more precise than spatial distance of nodes. The similarity of value is pre-defined by a categorizing range. For example, node 1 and node 2 obtain temperature as 15^0C and 15.5^0C [1] at the same time respectively; and at next time, they sense temperature as 15.5^0C and 16.5^0C respectively. If the categorizing range is 0.5^0C, the two nodes have similar data; and their gradient are same (1^0C), thereby being characteristic correlated. An energy efficient framework for data compression method (EFFECT) [100] used bucket approximation to compress data in a tree-based wireless sensor network. A bucket is a segment of time series, and the value of a bucket is represented by the average of the maximum and minimum entries in the bucket.

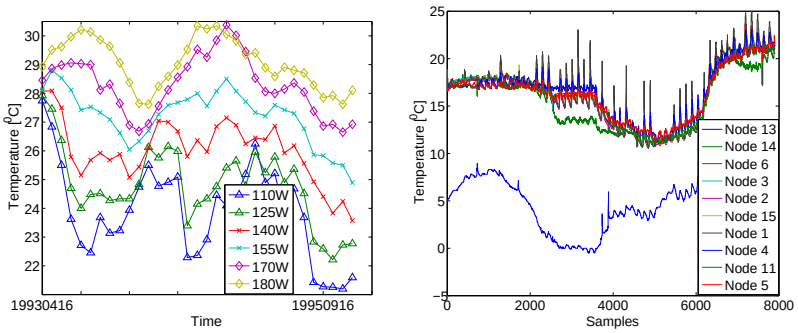
All such associated works achieve aggregation relying on raw data. However, the raw data of WSN is error-prone and dynamic, which definitely effect their performances. By observations, we find that data evolution is more stable and persistent than the raw data. Thus we propose Simba, which groups nodes by evolution rather raw data. Our experiment shows that Simba can considerably improve recovered accuracy comparing to the related work. In the following, we firstly highlight our observations, and then give our design in details.

[1] "^0C" denotes centigrade.

4.2 Observation of data evolution

We analyse two datasets to observe the evolution of raw data. The first dataset is from Tropical Atmosphere Ocean (TAO) Project [101] (denoted as $\mathcal{T}_o$, see Appendix A). The project focuses on monitoring ocean surface temperature for understanding and predicting $EIN\tilde{i}no$ and $LaN\tilde{i}a$. As shown in Fig. 4.1(a), we observed that several sensors express similar evolutions, whereas the data are dynamic (note that all the sensors are deployed in North Latitude 0^0, and W represents West Longitude).

The second dataset is provided by a custom project, which is temperature monitoring in a house (denoted as $\mathcal{T}_h$, see Appendix B). The deployed nodes report temperature every 10 minutes, and we consider 7900 data to construct our dataset $\mathcal{T}_h$, which are collected in Fig. 4.1(b).



(a) Dataset $\mathcal{T}_o$. (b) Dataset $\mathcal{T}_h$.

Figure 4.1: Datasets $\mathcal{T}_o$ and $\mathcal{T}_h$.

From Fig. 4.1, we obtain two properties: 1) abnormal data often appears in the raw data even though the monitored physical phenomena is stationary (i.e. temperature); 2) several nodes show similar evolution despite the raw data are different. Moreover, comparing to the dynamic raw data, the evolution is more stable and persistent. Thus, data aggregation achieved by the similar evolutions (instead of the raw data) has the potential to improve the accuracy.

4.3 Similarity of data evolution

Several researchers highlight that the nodes owning similar raw data can be grouped together to achieve data aggregation, thereby reducing energy consumption [53, 54, 55, 99, 100]. However, from our previous observations, we find that data evolution should be more suitable for data aggregation. That is to say, if nodes owning similar evolution can be organized to perform data aggregation, the recovered accuracy and

energy efficiency will be improved. Thus, we propose to measure the similarity of data evolutions. In this section, we firstly detail how to capture the data evolution, and then demonstrate how we measure the similarity.

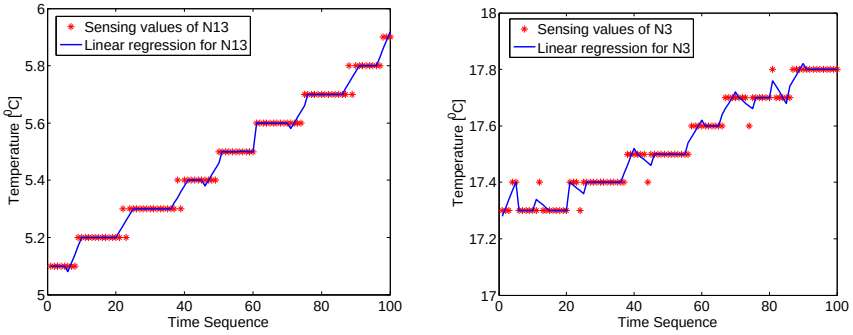
4.3.1 Capturing the evolution

Linear regression investigation [102] indicates that straight lines can be used to approximate time series, and the straight lines can be seen as evolution in our case. The raw data of a sensor node can be seen as a set of (y, i), in which y is the data value and i is the corresponding time sequence. Thus, linear regression can be simply expressed as: $y = Rc \cdot i + \beta$, where Rc is the slope of the straight line, and β is the intercept. In our case, we consider Rc to demonstrate the evolution.

In order to precisely capture the evolution, we use piecewise linear regression [103]. That is to say the linear regression will be computed in segments. Regarding a segment, regression coefficient Rc is computed by:

$$Rc_{S_{eg}} = \frac{\sum_{i=1}^{S_\iota}(i-\bar{i})(y_i-\bar{y})}{\sum_{i=1}^{S_\iota}(i-\bar{i})^2} \tag{4.1}$$

where S_ι is the segment length (number of data in a segment), $\bar{i} = \frac{1}{S_\iota}\sum_{i=1}^{S_\iota} i$, $\bar{y} = \frac{1}{S_\iota}\sum_{i=1}^{S_\iota} y_i$, and S_{eg} denotes the current segment. With more segments, node will generate a coefficient vector $\vec{Rc} = [\cdots, Rc_{S_{eg}-1}, Rc_{S_{eg}}]$, which can characterize the data evolution more precisely.



(a) Raw data $1-100$ from N.13 and the piecewise linear regression result. (b) Raw data 1-100 from N.3 and the piecewise linear regression result.

Figure 4.2: 20-segments linear regression, using data of N3 and N13 from dataset $\mathcal{T}_h$, where segment length $S_\iota = 5$.

Fig. 4.2 demonstrates the piecewise liner regression result from given sensor nodes (N3 and N13 of dataset $\mathcal{T}_h$). We can see that the lines in blue can express the evolution

of data (red start) in the associated segment. Thus, data evolution of sensor node is approximated by the vector of regression coefficients $\vec{Rc}$.

4.3.2 Measuring the similarity of data evolution

Cosine similarity [104] is a usual measurement, which shows the cosine of the angle between two vectors. The cosine similarity between $\vec{Rc}^{\,i}$ and $\vec{Rc}^{\,j}$ can be formulated as follow:

$$Cs(\vec{Rc}^{\,i}, \vec{Rc}^{\,j}) = \frac{\vec{Rc}^{\,i} \cdot \vec{Rc}^{\,jT}}{\sqrt{(\vec{Rc}^{\,i} \cdot \vec{Rc}^{\,iT})(\vec{Rc}^{\,j} \cdot \vec{Rc}^{\,jT})}} \tag{4.2}$$

where $\vec{Rc}^{\,T}$ states the transpose vector of $\vec{Rc}$. The cosine similarity values range between $[0, 1]$, where a value of 0 means that the vectors are dissimilar and a cosine similarity value close to 1 means that the vectors are similar. Giving a threshold th_{cs}, cosine similarity can provide us an explicit index to show whether two vectors are similar or not. For clarity, we provide the following definition:

Definition 4.3.1. *We define the similarity between two vectors ($\vec{Rc}$) as:*

$$sim(\vec{Rc}^{\,i}, \vec{Rc}^{\,j}) = \begin{cases} 1 & if \;\; Cs(\vec{Rc}^{\,i}, \vec{Rc}^{\,j}) \geq th_{cs}, \\ 0 & otherwise. \end{cases} \tag{4.3}$$

Since vector $\vec{Rc}$ represents the data evaluation, we consider that two data evolutions are similar if and only if their $sim(\cdot, \cdot)$ value is equal to 1. Note that th_{cs} should be bigger than 0.5 to avoid two orthogonal vectors being considered similar.

Theorem 4.3.1. *The similarity has transitive property. If $\vec{Rc}^{\,i}$ is similar to $\vec{Rc}^{\,j}$ and $\vec{Rc}^{\,k}$, thus $\vec{Rc}^{\,j}$ is also similar to $\vec{Rc}^{\,k}$, i.e.,*

$$\begin{aligned} &\forall i, sim(\vec{Rc}^{\,i}, \vec{Rc}^{\,j}) = 1 \;\&\; sim(\vec{Rc}^{\,i}, \vec{Rc}^{\,k}) = 1 \\ &\Rightarrow sim(\vec{Rc}^{\,j}, \vec{Rc}^{\,k}) = 1, \forall j, k \end{aligned} \tag{4.4}$$

Proof. From Definition 4.3.1, when $sim(\vec{Rc}^{\,i}, \vec{Rc}^{\,j}) = 1$ & $sim(\vec{Rc}^{\,i}, \vec{Rc}^{\,k}) = 1$, it means:

$$Cs(\vec{Rc}^{\,i}, \vec{Rc}^{\,j}) = \frac{\vec{Rc}^{\,i} \cdot \vec{Rc}^{\,jT}}{\sqrt{(\vec{Rc}^{\,i} \cdot \vec{Rc}^{\,iT})(\vec{Rc}^{\,j} \cdot \vec{Rc}^{\,jT})}} = x \geq th_{cs},$$

$$\text{and } Cs(\vec{Rc}^{\,i}, \vec{Rc}^{\,k}) = \frac{\vec{Rc}^{\,i} \cdot \vec{Rc}^{\,kT}}{\sqrt{(\vec{Rc}^{\,i} \cdot \vec{Rc}^{\,iT})(\vec{Rc}^{\,k} \cdot \vec{Rc}^{\,kT})}} = y \geq th_{cs},$$

$$\text{so, } \frac{\vec{Rc}^{\,i}}{\sqrt{\vec{Rc}^{\,i}\,\vec{Rc}^{\,iT}}} = \frac{1}{x} \cdot \frac{\vec{Rc}^{\,jT}}{\sqrt{\vec{Rc}^{\,j}\,\vec{Rc}^{\,jT}}} = \frac{1}{y} \cdot \frac{\vec{Rc}^{\,kT}}{\sqrt{\vec{Rc}^{\,k}\,\vec{Rc}^{\,kT}}},$$

multiplying $\frac{\vec{Rc}^j}{\sqrt{\vec{Rc}^j \vec{Rc}^{jT}}}$ to both sides of above equation, it will have:

$$\frac{1}{x} \cdot \frac{\vec{Rc}^{jT}}{\sqrt{\vec{Rc}^j \vec{Rc}^{jT}}} \cdot \frac{\vec{Rc}^j}{\sqrt{\vec{Rc}^j \vec{Rc}^{jT}}} = \frac{1}{y} \cdot \frac{\vec{Rc}^{kT}}{\sqrt{\vec{Rc}^k \vec{Rc}^{kT}}} \cdot \frac{\vec{Rc}^j}{\sqrt{\vec{Rc}^j \vec{Rc}^{jT}}},$$

$$\text{i.e., } \frac{1}{x} \cdot Cs(\vec{Rc}^j, \vec{Rc}^j) = \frac{1}{y} \cdot Cs(\vec{Rc}^j, \vec{Rc}^k),$$

$$\text{because } Cs(\vec{Rc}^j, \vec{Rc}^j) = 1,$$

$$\text{so } Cs(\vec{Rc}^j, \vec{Rc}^k) = \frac{y}{x} \geq \frac{th_{cs}}{x},$$

$$\text{because } 0.5 < th_{cs} \leq x \leq 1,$$

$$\text{thus } Cs(\vec{Rc}^j, \vec{Rc}^k) \geq \frac{th_{cs}}{x} \geq th_{cs},$$

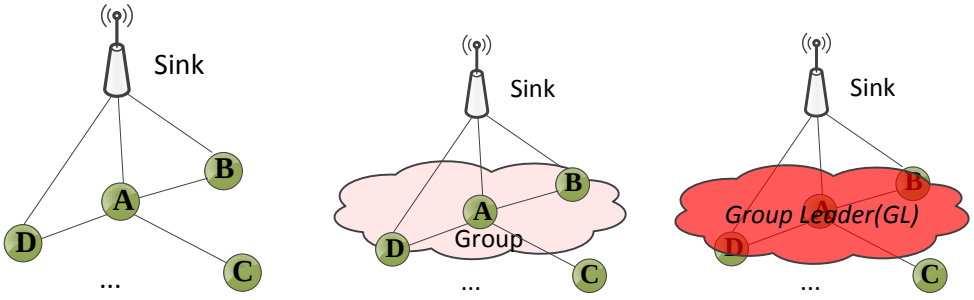
$$\text{i.e., } sim(\vec{Rc}^j, \vec{Rc}^k) = 1.$$

□

4.4 Similar-evolution based data aggregation

We claim that if we are able to measure the similarity of data evolutions, we can design a new data aggregation method which relies on evolution rather than the raw data. The stable evolution is more reliable than dynamic raw data to guarantee the recovered accuracy. Thus, we describe *Sim*ilar-evolution *B*ased data *A*ggregation (denoted as Simba) in this section.

Fig. 4.3 demonstrates the two phases of Simba: set-up phase and aggregation phase. Set-up phase (Fig. 4.3(b)) focuses on organizing nodes: nodes having similar evolutions are grouped together. Aggregation phase (Fig. 4.3(c)) focuses on executing data aggregation, where a Group Leader (GL) represents the group and is able to transmit the aggregated packet. We introduce the set-up phase firstly, and then describe the process of aggregation phase.



(a) Original network. (b) Set-up phase. (c) Aggregation phase.

Figure 4.3: The two phase of Simba, set-up phase and aggregation phase.

4.4.1 Set-up phase

During the set-up phase, sensor nodes aim to group together. First, each node computes its coefficient vector $\vec{Rc}$ using Eq. 4.1, and broadcasts it to its neighbors. When the neighbors receive $\vec{Rc}$, they will calculate the similarity using Eq. 4.3, and the information are stored in a neighbor table.

As shown in Fig. 4.4, we assume that node A is neighbor of nodes B, C and D. After broadcasting $\vec{Rc}$, every node checks the table and calculates the similarity. Tab. 4.1 shows the associated table of node A.

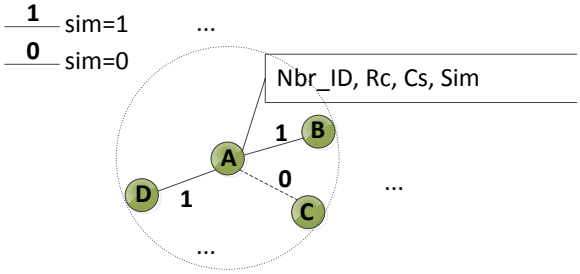


Figure 4.4: An example in set-up phase. The link labels correspond to the value of *sim* function.

Nbr_ID	Rc	Cs	Sim
B	$\vec{Rc}^{B}$=[$\cdots$,$Rc^{B}_{S_{eg}-2}$, $Rc^{B}_{S_{eg}-1}$, $Rc^{B}_{S_{eg}}$]	Cs($\vec{Rc}^{A}$, $\vec{Rc}^{B}$)	1
C	$\vec{Rc}^{C}$=[$\cdots$,$Rc^{C}_{S_{eg}-2}$, $Rc^{C}_{S_{eg}-1}$, $Rc^{C}_{S_{eg}}$]	Cs($\vec{Rc}^{A}$, $\vec{Rc}^{C}$)	0
D	$\vec{Rc}^{D}$=[$\cdots$,$Rc^{D}_{S_{eg}-2}$, $Rc^{D}_{S_{eg}-1)}$, $Rc^{D}_{S_{eg}}$]	Cs($\vec{Rc}^{A}$, $\vec{Rc}^{D}$)	1

Table 4.1: The neighbour table of node A.

When a node computes the similarity, it will notify to sink the results. Taking Tab. 4.1 as an example, node A confirms that nodes B and D are similar with itself, and it multicasts to the sink and nodes B and D that a new group is formed such as {A,B,D} (we use {} to denote group). Note that node B also generates a result such as {B,A}, and will send it to sink and node A. To avoid duplicated notifications, we use a timer T to control the waiting time before sending the notification message. According to the neighbor table, each node computes the number of nodes which have the similar evolution (i.e. $sim(\cdot,\cdot) = 1$ as defined in Eq. 4.3) with itself, assuming as f. Thus, for each node, the timer is set as $\frac{T}{f}$. As a result, the nodes owning more similar neighbors

will send notification faster than others. When a node receives a notification message, it considers that its own message is redundant, thereby cancelling its notification. If a node does not have similar neighbors (all sim values equal 0), it will do not generate notification message.

The pseudo-code for the set-up phase is shown in Alg. 1. We use several data to compute the regression coefficient, and these data duration is set-up phase duration, assuming as d. Considering segment length is S_ι, when node collects S_ι data, it will compute Rc (line 6), and send it to neighbours. As receiving others' Rc, node will compute the cosine similarity Cs (line 7) and $sim(\cdot,\cdot)$ (line 8). The sum of $sim(\cdot,\cdot)$ is used to know if a node has similar neighbours. If this condition is correct (line 10,11), node belongs to the same group which includes the nodes' ID; otherwise (line 13), node con not be a member of group. The timer $\frac{T}{f}$ is only used to avoid duplication, but there will be no impact on the result of group.

Algorithm 1 Pseudo-code in set-up phase

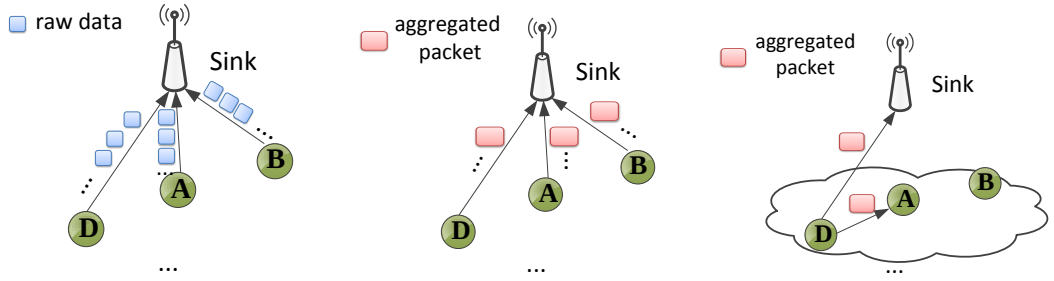
Require: raw data y_n; segment length S_ι; similarity threshold th_{cs}; neighbours' ID Nbr_ID; set-up phase duration d;
Ensure: group result G ;
1: initial S_ι, and $n = 1$;
2: **while** ($n \leq d$) **do**
3: collect $y_1, \cdots, y_n$;
4: set $f = 0$;
5: **if** (n mod S_ι=0) **then**
6: compute regression coefficient by Eq. 4.1; // send Rc to neighbours and receive others' Rc
7: compute Cs using Eq. 4.2;
8: compute $sim(\cdot,\cdot)$ using Eq. 4.3;
9: $f = \sum_{\forall Nbr\_ID} sim(\cdot,\cdot)$;
10: **if** $f > 0$ **then**
11: $G = \{Nbr\_ID | sim(\cdot, Nbr\_ID) = 1\}$; // send notification using timer $\frac{T}{f}$
12: **else**
13: $G = \varnothing$;
14: **end if**
15: **end if**
16: **end while**

4.4.2 Aggregation phase

In the aggregation phase of Simba, the group members share the same aggregated information. It means the nodes having similar evolutions use the same coefficients to aggregate data. For example, nodes A and B are in the same group, and we consider aggregation function is A-ARMA. The model coefficients (φ, ϑ in Eq. 2.11) of A-ARMA computed by node A are also used to predict node B's data. In each group, only the Group Leader (GL) processes and transmits aggregated packet. With the group information in set-up phase, sink can recover GL's data using aggregation function, and the data of other nodes can be retrieved easily. In order to guarantee the accuracy, we propose that each node in one group becomes GL in turn. We will demonstrate the aggregation progress and recovering progress in detail.

Aggregation progress

At the beginning of aggregation phase, sink will define the sequence of each node becoming leader, and this information is send to the first leader. The leader stores this information, and uses the sequence to assign the next leader. For example, considering the example of Fig. 4.4, node A (assuming it is the current GL) uses an sequence as A-B-D to assign the next leader, i.e. the next leader is node B, and B will also store this sequence.



(a) No aggregation (b) Original aggregation (c) Aggregation using Simba

Figure 4.5: Illustration of aggregation phase in Simba, comparing to situations of no aggregation and original aggregation.

Fig. 4.5 illustrates the comparisons of the data flow between no aggregation, original aggregation work and the aggregation phase in Simba. Without aggregation (Fig. 4.5(a)), each node directly sends its raw data to sink. In original aggregation work (Fig. 4.5(b)), each node uses an aggregation function to predict data step by step. When new data arrives, node compares the predicted data to the new arrived one: if the error is beyond pre-defined threshold th_{err}, node computes new aggregated

packet and transmits it to sink; otherwise, node waits for more data, and repeats this process (the details can be seen in Sec. 2.2.2). While in the aggregation phase of Simba, nodes in one group share the aggregated packet. Assuming nodes A, B and D belong to the same group as shown in Fig. 4.5(c), and node D is the current leader which aggregates information as the original aggregation process discussed above. When node D checks that the accuracy is beyond th_{err}, it sends new aggregated data to sink and the next leader (assuming node A in the example of Fig. 4.5(c)). Then node A will use the received aggregated packet to predict data and compare with new raw data, and then repeats the process as node D. Due to the similar evolution, nodes in one group share the aggregated information, and the group is represented by one node (i.e. GL). Meanwhile, each node in the same group becomes leader in turn, thus each node will check the aggregated packet as its raw data at each step, thereby guaranteeing the accuracy.

The main pseudo-code for aggregation phase is shown in Alg. 2. When a node receives the aggregated packet P, i.e. the coefficients of aggregation function, it will use the coefficients to predict data (line 2). And then compute the error (line 3) and check if this aggregated packet is acceptable or needs to be updated. If error is beyond the threshold, node transmit the packet to sink, and start a new aggregation process (line 8,9).

Algorithm 2 Pseudo-code in aggregation phase

Require: raw data y_n; error threshold th_{err}; aggregation function $\mathcal{F}$; prediction step S;

1: receive aggregated packet P;
2: predict data $\hat{y}_{n:n+S} = \mathcal{F}(P)$;
3: compute RMS error between $\hat{y}_{n:n+S}$ and $y_{n:n+S}$ by $e = \sqrt{\frac{\sum_1^S (\hat{y}_{n:n+S} - y_{n:n+S})^2}{S}}$
4: **if** $e \leq th_{err}$ **then**
5: $n = n + S$;
6: go to line 2, repeat;
7: **else**
8: transmit aggregated packet to sink
9: start a new aggregation process
10: **end if**

Recovering progress

During aggregation phase, sink is responsible for recovering the received data. It computes the data of the GL using the associated aggregation function. For the leader A,

sink recovers its data directly from the aggregation functions as:

$$\widehat{y}^A = \mathcal{F}(y^A) \tag{4.5}$$

where $\mathcal{F}$ denotes the aggregation function, $\widehat{y}^A$ states the recovered data of node A, and y^A is the aggregated data sent by node A. Supposing node B belongs to the same group of node A, and the difference between node A and node B is defined by:

$$D_{AB} = \widehat{y}^A - \widehat{y}^B \tag{4.6}$$

The initial value of D_{AB} is computed by the reports during set-up phase. Since node A and B have similar evolutions, thus sink can simply retrieve the data of B as:

$$\widehat{y}^B = \widehat{y}^A - D_{AB} \tag{4.7}$$

That is to say, if sink can recover the data of group leader (node A), it will compute the data of group members (node B).

Therefore, using Eq. 4.5, Eq. 4.6 and Eq. 4.7, sink can recover all the data from the group. Similar evolution maintains a relative stable difference (i.e. D_{AB}) between group members, thus sink can recover all the data accurately from the data of GL. Moreover, the cooperation of nodes in one group substantially reduces the number of packets transmissions thereby reducing energy consumption.

4.5 Performance evaluation

We first discuss the choice of similarity threshold th_{cs} that is mentioned in Sec. 4.4, then we present the accuracy issues and energy consumptions considering with and without Simba. In addition, comparative analysis are presented to show the benefits of using similar evolution. Based on the two datasets, we simulate the set-up phase using WSNet[105], an event-driven simulator for wireless networks, to show the results of group. Matlab is used to recover data and to compare the RMS error to raw data considering dataset $\mathcal{T}_h$.

4.5.1 Selection of similarity threshold th_{cs}

Similarity threshold th_{cs} is defined to check the similarity of data evolutions. The higher th_{cs} is, the more similar the data evolutions are, and the more nodes can be organized in one group. On one hand, a higher th_{cs} is desirable to maximize the similarity between evolutions to guarantee recovered accuracy. On the other, if it is

too high, less nodes will be grouped together, and the efficiency of aggregation will be not so powerful for Simba. Thus it is important to select a suitable similarity threshold.

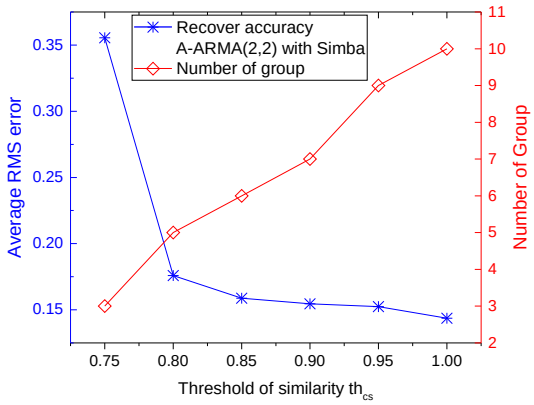
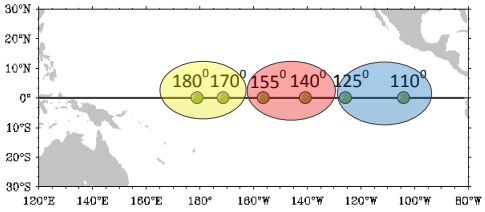


Figure 4.6: The influence of similarity threshold th_{cs} on recovered accuracy and group result, based on dataset $\mathcal{T}_h$ (error threshold $th_{err} = 0.03$).

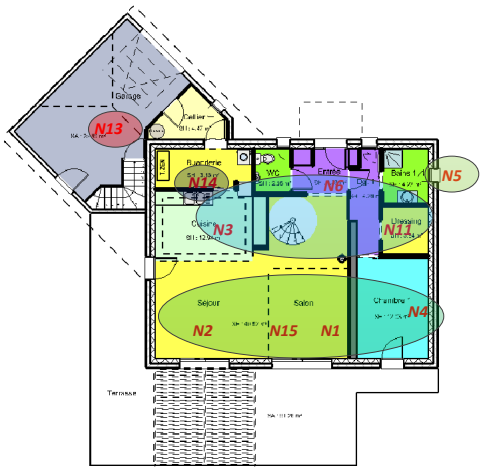
Considering that the initial number of groups is the number of nodes (6 for dataset $\mathcal{T}_o$ and 10 for $\mathcal{T}_h$ in Fig. 4.1). Thus if the number of groups decreases, it means more nodes are grouped. We use A-ARMA(2,2)[30] as the aggregation function with Simba to investigate number of groups and accuracy considering dataset $\mathcal{T}_h$. We illustrate the result in Fig. 4.6 with double Y-axes, where the left one corresponds to error (blue curve) and the right one corresponds to the number of groups (red curve). When similarity threshold th_{cs} is equal to 1, the group number is 10 (number of nodes), and each node forms an isolated group. Correspondingly, recovered accuracy is highest (RMS≤ 0.15) because each node only executes aggregation function based on its own data. As decreasing th_{cs}, more nodes meet th_{cs}, they group together, but the accuracy decreases. When $th_{cs} = 0.75$, we have only 3 groups in the network, while the accuracy is lowest. Thus, considering energy saving and accuracy, we choose 0.8 as the similarity threshold for dataset $\mathcal{T}_h$. The threshold for dataset $\mathcal{T}_o$ can be estimated using the same method, which is 0.85.

With the similarity threshold, we show the result of grouping in Fig. 4.7, where the grouped nodes are indicated by circles. For dataset $\mathcal{T}_o$ [101], we simulate that all the nodes are in a chain topology, and each node connects with its 1-hop neighbours. After set-up duration, there are 3 groups for this dataset, and we can see that the neighboring nodes are grouped together, shown in Fig. 4.7(a). Regarding $\mathcal{T}_h$, we simulate the nodes which are the same network topologies as defined in the project (see Appendix B). In order to collect sufficient number of data to have relative stable evolution, the duration of set-up phase is the time to collect 100 raw data. There are 5

groups formed for dataset $\mathcal{T}_h$ as shown in Fig. 4.7(b).



(a) Group result for $\mathcal{T}_o$.



(b) Group result for $\mathcal{T}_h$.

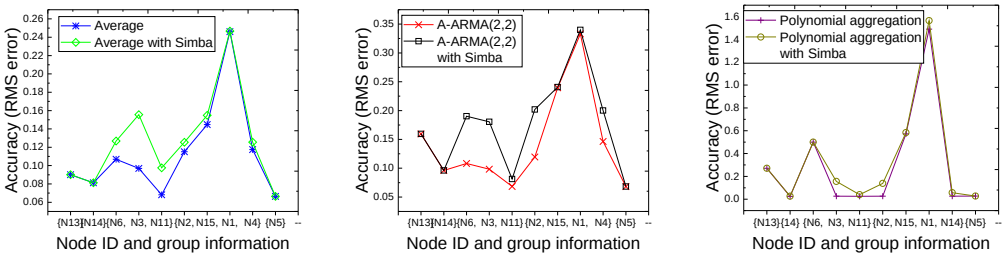
Figure 4.7: Results of grouping using dataset $\mathcal{T}_o$ and $\mathcal{T}_h$.

4.5.2 Accuracy and energy saving by Simba

An aggregation scheme should keep high recovered accuracy, save both energy and network capacity. Simba can be coupled with several aggregation functions in aggregation phase. Here we simulate three functions: A-ARMA, polynomial aggregation and average. Simba does not change the size of aggregated packet for the associated aggregate functions, thus we mainly investigate the accuracy, and the energy saving by Simba.

For the three aggregation functions, we compute the recovered accuracy of each node with Simba, comparing to the accuracy of original functions in Fig. 4.8 respectively. We can see that if a node does not belong to a group, it works on the original functions, thus there is no change for its accuracy, such as N13, N14 and N5. For nodes that are grouped together (e.g. {N6,N3,N11}), we note that Simba indeed introduces a little error. This is reasonable because at each step, only Group Leader represents the group to compute aggregation function on its own data, the others' data are recovered by similar evolution (Eq. 4.7). Tab. 4.2 shows the details of the accuracy: the accuracy decreases (RMS error increases), but in an acceptable range (decreasing by up to 22%).

Regarding the energy consumption, Fig. 4.9 shows the results for 10 nodes in situation of no aggregation, original functions (without Simba) and the situations using Simba. Firstly, we use the networking-oriented metrics (see Sec. 3.1) to evaluate the performances of Simba. Note that, Simba does not change the aggregated packet size



(a) Average (b) A-ARMA(2,2) (c) Polynomial aggregation

Figure 4.8: Recover accuracy with and without Simba (nodes ID in {} in x-axis are group members).

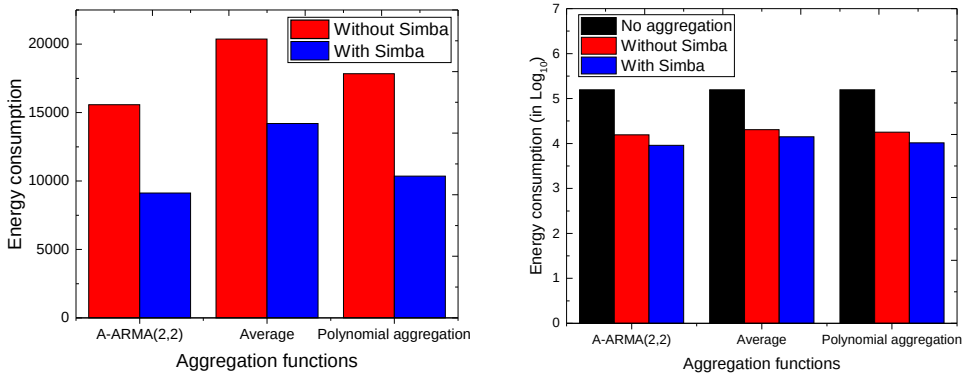
of the functions, thus the packet size coefficient of Simba is same with the original functions, e.g., $\lambda^{Simba}_{average} = \lambda_{average}$, $\lambda^{Simba}_{aarma} = \lambda_{aarma}$ and $\lambda^{Simba}_{poly} = \lambda_{poly}$. For the aggregation ratio, Tab. 4.2 lists the number of aggregated packets and the aggregation ratio ω respectively. We can see that A-ARMA can achieve an aggregation ratio as $\omega_{aarma} \approx 19\%$ (each node transmits about 1471 aggregated packets); average has aggregation ratio $\omega_{average} \approx 28\%$ and the ratio of polynomial aggregation is $\omega_{poly} \approx 21\%$. Simba reduces the number of aggregated packets for these functions because it organizes nodes to execute the functions in groups. For A-ARMA, the aggregation ratio using Simba is $\omega^{Simba}_{aarma} \approx 11.2\%$ (each node only transmits about 874 aggregated packets); average also gets a better aggregation ratio when coupled with Simba, that is $\omega^{Simba}_{average} \approx 19.6\%$; and the ratio of polynomial aggregation considering Simba is $\omega^{Simba}_{poly} \approx 14.7\%$. As we detailed in Sec. 3.1, smaller aggregation ratio means that the function saves more energy. Thus we can see that coupled with Simba, all the functions achieve a better energy saving even thought they sacrifice a little accuracy.

Comparing to the data collection in aggregation phase (7800 raw data for each node), the energy for set-up phase (duration for collecting 100 data) can be negligible. Assuming that energy for transmission is 1 and for reception is 1 in our simulation, we show the energy consumptions of different situations in Fig. 4.9. We can see that the whole energy consumption is reduced when Simba is coupled (see Fig. 4.9(a)). Whatever the functions are, Simba helps to save more energy, and at least saves 30% more than the original functions do. For A-ARMA(2,2), Simba achieves a 41% reduction on energy consumption. Comparing with the situation of no aggregation (a logarithmic scale), Simba can save energy around 94%, 91% and 95% for 3 aggregation function respectively (see Fig. 4.9(b)).

Table 4.2: Comparison between original functions and the situations coupled with Simba, using dataset $\mathcal{T}_h$.

#[1]raw data=7800			
aggregation functions	comparing entity	no Simba	using Simba
Average	recovered accuracy	0.144	0.176(↑)[2]
	# aggregated packet	2221	1529(↓)[3]
	ω [4]	28%	19.6%(↓)
A-ARMA	recovered accuracy	0.113	0.127(↑)
	# aggregated packet	1471	874(↓)
	ω	19%	11.2%(↓)
Polynomial	recovered accuracy	0.299	0.336(↑)
	# aggregated packet	1616	1146(↓)
	ω	21%	14.7% (↓)

[1] # represents the number of packets;
[2] ↑ denotes that the value increases comparing to the value in left;
[3] ↓ marks that the value decreases comparing to the value in left;
[4] ω denotes aggregation ratio.



(a) Energy comparison by different aggregation functions.

(b) Energy comparison in different situations (logarithmic scale).

Figure 4.9: Energy consumption comparison among no aggregation, original functions and using Simba.

4.5.3 Comparative analysis

Related works always pay attention to raw data, but few aggregation methods, to the best of our knowledge, focus on data evolution. Characteristic correlation [99] propose to cluster the nodes that own similar raw data, and we thus select it as the state-of-the-art benchmark for Simba. The nodes holding similar raw data form a virtual cluster in [99], and the average value is representative for the cluster. When sink receives the

representative value, it retrieves the value as all the cluster members' data.

To compare with Simba, we simulate characteristic correlation in Matlab using dataset $\mathcal{T}_h$, and nodes are deployed as same as Fig. 4.7(b). The categorizing range is actually the error threshold in Simba, referred from the literature [99]: we use 0.5^0C and 1^0C to test the recovered accuracy. In addition, as proposed in [99], the cluster needs to train several data in order to keep it more stable, thus the first 100 data of $\mathcal{T}_h$ are used for this purpose. Average is used as aggregation function for Simba and characteristic correlation. We calculate accuracy (RMS error between recovered data and raw data) for every 10 data, to show the performance of two aggregation methods.

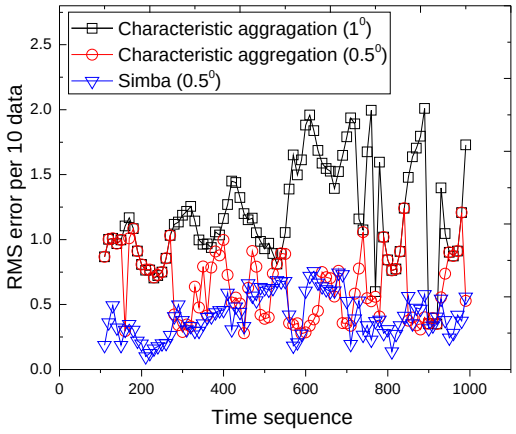


Figure 4.10: Comparison analysis between characteristic correlation and Simba, considering error threshold as 0.5^0C and 1^0C.

Fig. 4.10 demonstrates the RMS error per 10 data for characteristic correlation and Simba: we can see Simba leads to a lower error than characteristic correlation. Moreover, with different categorizing range (0.5^0C and 1^0C), characteristic correlation shows a quite unstable RMS error. This is because characteristic correlation clusters nodes owning similar raw data, and when the raw data changes abruptly, the accuracy will be effected directly. For Simba, it uses evolution to group nodes together, the evolution is more stable than raw data. Besides, in characteristic correlation, sink retrieves the average values as all cluster members's data, it does not consider the difference between nodes. While for Simba, sink uses the mean difference to retrieve data, which can guarantee the accuracy further.

4.6 Conclusion

In this chapter, we propose *Sim*ilar-evolution *B*ased data *A*ggregation, which is a data-independent aggregation, benefitting from similar evolution. By investigation on real datasets, we analyse that: 1) abnormal data often appears even though the monitored variable is stationary (e.g. temperature), 2) several nodes show similar evolutions. Thus, comparing to the dynamic raw data, evolution is stable and lasting, which is more suitable to be used to organize nodes.

Simba works in two phase, the set-up phase and the aggregation phase. During set-up phase, nodes measure their data evolution and confirm similar neighbors. In aggregation phase, nodes in one group alternatively transmit the aggregated packets to sink. Each node only uses a few packets to transmit, and the sink can recover the whole data for a group. We can see that similar evolution gives nodes the opportunity of cooperating with each other when executing aggregation function. The experiment results show that Simba uses less aggregated packets to get high fidelity data, and Simba can save at least 30% energy than original aggregation functions do. Comparing to the situation without aggregation, Simba can save energy more than 91%.

5

Agnostic Aggregation in Wireless Sensor Networks

Contents

Different data property or network property requirements procure adaptive data aggregation functions. However, the related works are designed for specific data pattern or network topology. It means these functions often fail to recover data accurately

from non-anticipated data variations.For example, A-ARMA [30] proposed a prediction model with fixed parameters. Compressive sensing (Sec. 2.2.3) Compressive sensing is more suitable for dense networks, because there will more opportunities to select measurements. To avoid the impacts of data property on aggregation functions and improve the adaptability, we propose a property-independent aggregation function in this chapter: *Agnostic Aggregation* (A^2). It allows sensor nodes to self-tune aggregation parameters to adapt the new data property, with respect to efficiency and accuracy.

5.1 Introduction

We classified aggregation functions into two types: forecasting aggregation and compressing aggregation (see Sec. 2.2), and our research work is mainly focused on forecasting functions. Application of forecasting functions can further enhance energy saving by approximating sensing data at sink level under a user-defined accuracy requirement [106]. The process of such functions is: 1) prediction model is implemented at node level on real sensing data; 2) model parameters (instead of raw data) are sent to the sink which recovers data without communicating with the nodes; 3) model parameters are updated and transmitted each time when the prediction error exceeds a given error bound.

However, as mentioned previously, an associated drawback is that the prediction models are often designed with fixed parameters, which limits their adaptability. When facing non-anticipated data variations, such functions may not guarantee to recover data accurately.Thus our aim is to propose a property-independent aggregation function, which can adaptively adjust the model to fit the latest data to guarantee the accuracy.

Auto regression moving average (ARMA) model [107] is a well-used prediction model in time series studies, which is also frequently investigated in wireless sensor networks (see Sec. 2.2.2). Assuming a stationary property for the collected data, and assuming these data are represented by v_t, thus the data recovering process of ARMA model can be formulated as:

$$\mathcal{F}_{arma} \equiv \dot{v}_t = \varphi_0 + \varphi_1 \cdot v_{t-1} + \cdots + \varphi_p \cdot v_{t-p} + \vartheta_1 \cdot \varepsilon_{t-1} + \cdots + \vartheta_q \cdot \varepsilon_{t-q} \tag{5.1}$$

where $\varphi_0, \cdots, \varphi_p$ are the parameters for AR part, $\vartheta_1, \cdots, \vartheta_q$ are the parameters for MA part, p, q is the associated model order, and ε is the white noise. We can see that the order of p and q impact the number of parameters for an ARMA model, i.e. there are $p + q + 1$ parameters in the model.

Our proposition, Agnostic Aggregation (A^2), is based on A-ARMA model, which

can automatically adjust the order (p and q) to adapt the latest data, thereby guaranteeing the recovered accuracy. A-ARMA is an extension of ARMA model, dedicating to embedded system with low computation capacity through the use of sliding window technique [30]. In the following, we first introduce how to select the model order in Sec. 5.2, and then give in depth explanations of the proposed function A^2 in Sec. 5.3. Sec. 5.4 discusses the optimal setting for several system parameters and the comparative analysis of different models are presented in Sec. 5.5.

5.2 Model order adaptation

Since the performance of ARMA(p,q) model depends on the order of p and q , thus how to select, not necessarily the optimal, but a good order is the present question for A^2. Our objective is able to dynamically adapt the model when new data arrives: the new model should fit the latest data better. For example, we can consider the candidate models are ARMA(p_1, q_1), ARMA(p_2, q_2) and so on, what we need is one model with optimal (p, q) which make the model to be suitable for the data. To evaluate the different models, several criteria can be used such as: Akaike Information Criterion (AIC), AICc, Bayesian Information Criterion (BIC) [108, 109].

5.2.1 Akaike Information Criterion

The Akaike information criterion (AIC) [110] is a measure of the relative quality of statistical models for a given set of data. Given a collection of models for the data, AIC estimates the quality of each model, relative to the other models.

AIC uses the concept of information entropy [111] to offer a relative measure of the information loss for every approximating model, considering the principle of parsimony. AIC uses the Kullback-Leibler information [112] ("distance" between two models) and the maximum likelihood [61] (parameter estimation) to describe the trade-off between the bias and the variance (accuracy and complexity respectively). If the candidate model has few parameters, the model may not fit the data well (high bias) but have lower complexity (low variance); if a candidate model has a lot of parameters, the bias of the model may be low while the variance (complexity) is high. AIC can be formulated as:

$$AIC = \underbrace{-2 \cdot ln(L)}_{bias} + \underbrace{2 \cdot k}_{variance} \tag{5.2}$$

where L is the maximum likelihood function, and k is the dimension of the model (number of parameters). The first term of Eq. 5.2 demonstrates the bias and the second

term denotes the variance. The best model is the one which has the smallest value for AIC.

If the model errors are normally distributed, the Eq. 5.2 can be written as:

$$AIC = \underbrace{n \cdot ln(MSE)}_{bias} + \underbrace{2 \cdot k}_{variance} \tag{5.3}$$

where MSE stands for the mean squared error, and n is the number of observation. In our context of ARMA model, MSE is the mean squared error between predicted data and real data, n is the number of data, and k is the number of model parameters, i.e., $p + q + 1$.

AICc [113] is an extension of AIC, which is AIC with a correction for finite sample sizes. AICc is used in the case where the observation size is small relative to the model dimension. Relying on the rule of thumb, AICc can be used when $\frac{n}{k} < 40$ [108]. AICc can be computed by:

$$AICc = AIC + \frac{2 \cdot k \cdot (k+1)}{n - k - 1} \tag{5.4}$$

5.2.2 Bayesian Information Criterion

BIC [114] is similar to AIC, it is a criterion for model selection among a set of models, which is also based on the likelihood function. BIC is mainly used to choose the appropriate dimension of a model, such as the order for a polynomial regression. Similar as AIC, BIC also trades off between bias and variance, but with a little modification. BIC highlights that in large-sample limit, Eq. 5.3 will be a maximum likelihood estimator only. Thus, BIC modifies the second term of Eq. 5.3 as (assuming model errors are normally distributed):

$$BIC = n \cdot ln(MSE) + 2 \cdot k \cdot ln(n) \tag{5.5}$$

where the parameters have same meanings with Eq. 5.3.

Considering above definitions, AIC and BIC are both penalized-likelihood criteria. They are sometimes used for choosing the best order in regression and often used for comparing models, which ordinary statistical tests cannot do. AIC generally tries to find unknown model that has high dimensional reality, while BIC only choose the best model among a list of candidate models. AIC is good for making asymptotically equivalent to cross-validation, while BIC is good for consistent estimation. BIC assumes that the number of data has more impact on the model than AIC.

5.3 Agnostic Aggregation

Agnostic Aggregation (A^2) is a dynamic forecasting function, where sensor nodes locally update their own A-ARMA model parameters only when the predicted data do not fit the raw data, according to a given error threshold. A^2 uses model order adaptation methods (Sec. 5.2) to determine the appropriate order, which makes the new order more suitable for latest time series.

We introduce several system parameters in A^2, which includes:

- W: the size of the moving window,
- S: the step,
- W': the number of data to build the model,
- p_{max}: the maximum value of p',
- q_{max}: the maximum value of q',
- th_{err}: the error threshold.

As illustrated in Fig. 5.1, our proposition works as follow:

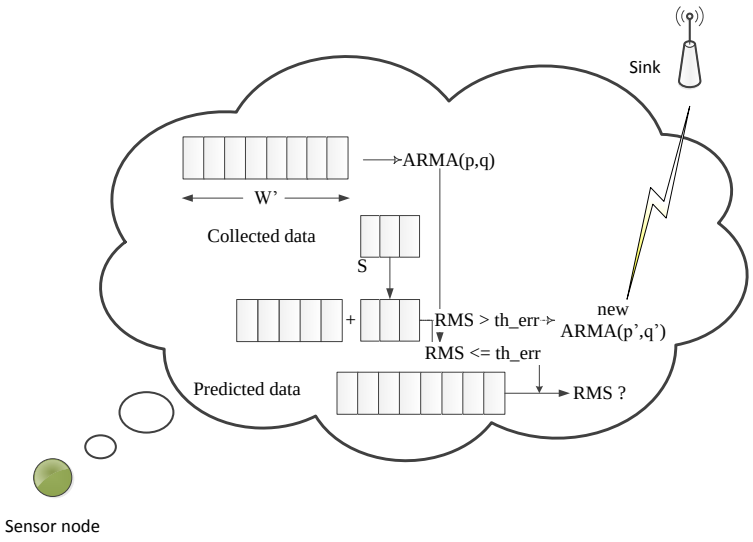


Figure 5.1: Illustration of agnostic aggregation.

1. Initially, each node builds an A-ARMA(p, q) model once it has collected W' data, where $p + q + 1$ parameters for ARMA model are computed (Eq. 5.1). Only the model parameters are communicated to the sink. The order (p, q) is just the initial order which predict more accurately.

2. The node measures the root mean square (RMS) error between S predicted data and S latest data. If the difference is within the given error threshold, th_{err}, then the node continues using its current ARMA model. In this case, there is no update send to the sink because the current model is sufficiently accurate.

3. If the difference is greater than th_{err}, then the node computes the new order p' and q' (by Eq. 5.3, Eq. 5.4 or Eq. 5.5) and re-computes the new parameters for the ARMA model on the most recent W' samples. Then the new parameters are send to sink.

In our function, considering the latest W' data, we build an adaptive model (when the error is greater than th_{err}) within new order. Generally, the adaptive model is more approaching the real situation because A^2 is based on the latest collected data. We choose (2,2) as our initial order because such order is suggested in the associated reference [30]. In this reference, the authors compared several different order, and showed that (2,2) can reduce the number of parameters and also keep the accuracy. By the way, (2,2) is just initialize the aggregation function. If this coefficient are not meet the accuracy, the model adaption (AIC/AICc/BIC) will evolve new order. So, we claim that basically, A^2 is incentive by this first order.

In terms of the computational complexity, the cost of the estimation procedure of A-ARMA models grows as p, q or the number of data grows. The complexity of executing an A-ARMA model parameter estimation process is demonstrated as $\mathcal{O}(m^3W')$ [30], where m is number of model parameters and W' is number of data to build the model. Firstly, the moving window technique reduces the complexity of the estimation procedure. Secondly, we find that A^2 actually leads to less parameters comparing to A-ARMA(2,2). Therefore, adaptive model order actually reduces the computational overhead.

To evaluate the performance of our proposition, we consider two real datasets $\mathcal{T}_m$ and $\mathcal{P}_s$ (see Appendix B and A), where $\mathcal{T}_m$ is temperature data in a museum (720 data), and $\mathcal{P}_s$ represents a collection of sea level pressure (11700 data).

5.4 System parameters selection for given datasets

In order to evaluate the performance of A^2, it is necessary to find the optimal system parameters. We consider three metrics as:

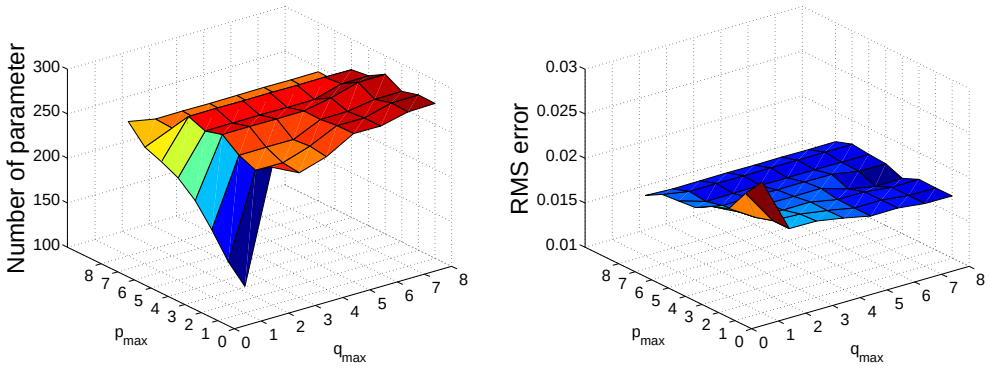
1) the number of transmission, which is linked to the number of model failure;

2) the recovered accuracy (RMS error): mean difference between the predicted data and the raw data;

3) the number of parameters: sum of $(p+q+1)$ for all transmissions.

The number of transmission actually is the number of model update. If current model cannot meet the error threshold, nodes need to update the model and transmit transmit a new aggregated packet, thereby leading a new transmission. The number of parameters denotes the size of aggregated packet. Note that the original packet size is the size of raw data, while the aggregated packet size is the size of parameters. All these results are achieved by numerical simulation using Matlab. Here, we consider AICc as the criterion due to the rule of thumb, i.e. the number of data is small relative to the number of model parameters.

Selection for p_{max} and q_{max}: First, we discuss the model order selection. With dataset $\mathcal{T}_m$, we simulated A^2 with p_{max}, q_{max} from 1 to 8, to evaluate the number of parameters and the accuracy. We plot the results on 3D figures (see Fig. 5.2). We can see that if p_{max} and q_{max} increase, then the number of parameters increase too (see Fig. 5.2(a)), even though they slightly decrease the RMS error (see Fig. 5.2(b)). From the whole view of Fig. 5.2, $p_{max} = q_{max} = 5$ is a good choice, which can keep the lower number of parameter and guarantee the accuracy.



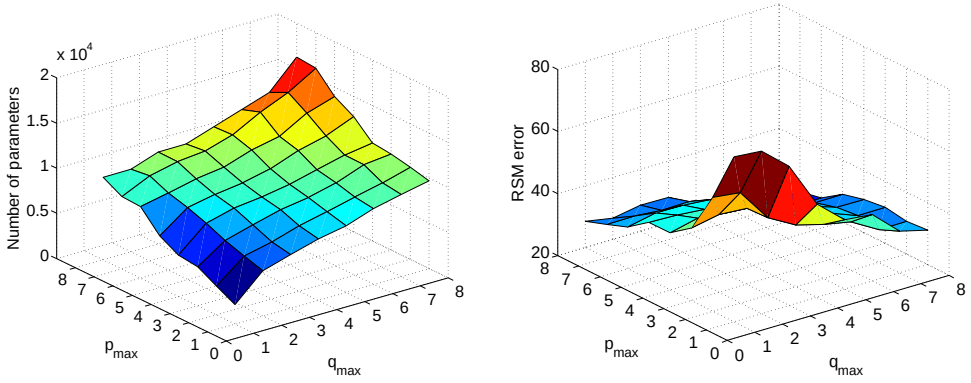
(a) $p_{max} - q_{max}$ effect on the number of parameters.

(b) $p_{max} - q_{max}$ effect on RMS.

Figure 5.2: The effect of p_{max} and q_{max} on dataset $\mathcal{T}_m$, where $W' = W = 20$, $S = 5$ and $th_{err} = 0.03$.

We also use dataset $\mathcal{P}_s$ to simulate A^2. As shown in Fig. 5.3, we can get similar results. Bigger p_{max} and q_{max} decrease the RMS error, but increase the number of parameters. In order to have a compromise, $p_{max} = q_{max} = 5$ can be considered also for dataset $\mathcal{P}_s$. Thus, without loss of generality, we limit the order upper bound to 5 ($p_{max} = q_{max} = 5$) in our following experiments.

Selection for W and W': Secondly, we investigate the values for W and W' considering the two datasets. We set different values of W and W' to investigate the effect

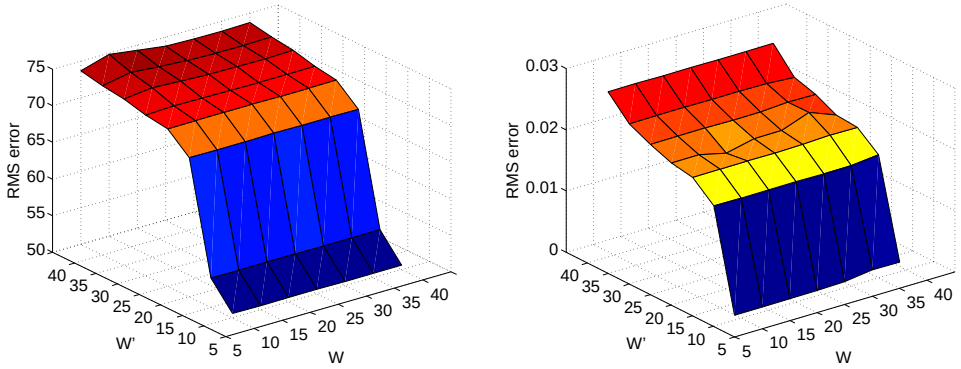


(a) $p_{max} - q_{max}$ effect on the number of parameters.

(b) $p_{max} - q_{max}$ effect on RMS.

Figure 5.3: The effect of p_{max} and q_{max} on dataset $\mathcal{P}_s$, where $W' = W = 20$, $S = 5$ and $th_{err} = 75$.

on the accuracy. We find that the number of data for building model (W') is not the bigger the better, see Fig. 5.4. When W' is bigger than 20, the RMS error increases dramatically. Thus $W' \leq 20$ is a good choice.



(a) For dataset $\mathcal{P}_s$, where $th_{err} = 75$.

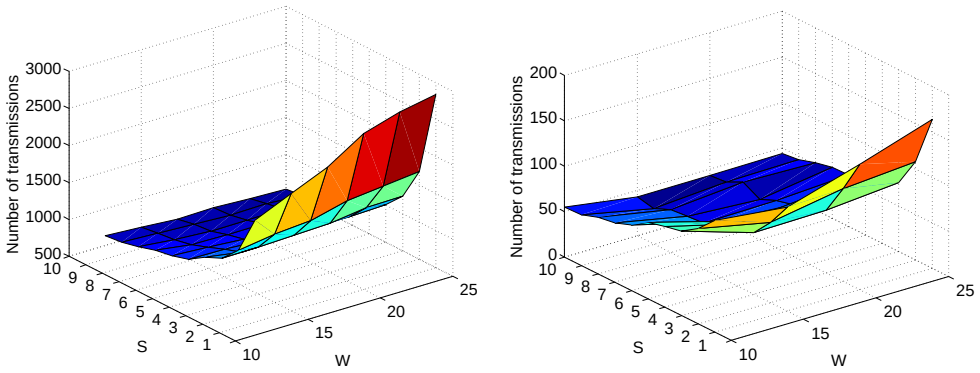
(b) For dataset $\mathcal{T}_m$, where $th_{err} = 0.03$.

Figure 5.4: The effect of W and W' on two datasets, where $p_{max} = q_{max} = 5$,and $S = 5$.

As defined above, W is the size of sliding window, and W' is the number of data to build model. If W' is bigger than W, a latency is introduced. That is because the node needs more data to build the model than it needs for the moving window, i.e., the node would collect more $W' - W$ data to be able to compute the new model. Thus to reduce the effect on accuracy, we consider $W' = W \leq 20$.

Selection for W and S: Finally, we investigate the impact of step S. We set $S \in [1, 10]$ and W varies from 10 to 25. As shown in Fig. 5.5, when S is greater than 5, the number of transmission is lower relatively, that is to say, increasing S can reduce the

number of transmissions. In fact, if S is large, there are less times to check error, so the number of transmission reduces correspondingly. But we need to point out that if S is too large, the cumulative error will increase. So in our context, $S \in [5, 10]$ is used.



(a) For dataset $\mathcal{P}_s$, where $th_{err} = 75$. (b) For dataset $\mathcal{T}_m$, where $th_{err} = 0.03$.

Figure 5.5: The effect of W and S on two datasets, where $p_{max} = q_{max} = 5$.

5.5 Performance evaluation

To evaluate A^2, we use real datasets to test the performance comparing to other aggregation functions firstly, and then, we construct a synthetic dataset with different data properties, to check whether A^2 can adapt the dynamic data variations.

5.5.1 Comparison using real datasets

Energy and accuracy issues

Considering dataset $\mathcal{T}_m$ with system parameters $W = W' = 20$, $S = 10$, $p_{max} = q_{max} = 5$ and $th_{err} = 0.03$, we compare the performance between A^2 and A-ARMA. We presented several criteria in Sec. 5.2, so we also show the details among different criteria in Tab. 5.1. We can see that AICc is more suitable for our context, which is consistent with the definition of AICc. Empirically, AICc is used in the case where the sample size is small, relative to the number of parameters ($\frac{n}{k} < 40$). In the scenario of $\mathcal{T}_m$, the data number is $n = W = 20$ and the number of parameters is $k = p+q+1 \approx 5$.

Meanwhile, networking-oriented metrics (aggregation ratio ω and packet size coefficient λ) are also computed and shown in Tab. 5.1. Aggregation ratio is the ratio between the number of aggregated packets and the number of generated packets without aggregation. Considering dataset $\mathcal{T}_m$, the number of generated packet is equal to

Table 5.1: The results for A-ARMA and A^2 on dataset $\mathcal{T}_m$.

#[1]raw data	720			
		Agnostic Aggregation(A^2)		
	A-ARMA	AIC	BIC	AICc
# transmissions	66	61	59	57
# parameters	330	301	256	215
ω^2	9.1%	8.4%	8.1%	7.9%
λ^3	5	4.9	4.3	3.8
recovered accuracy (RMS error)	0.0291	0.0259	0.0263	0.0182

[1] # represents the number of entity;
[2] ω denotes aggregation ratio;
[3] λ denotes packet size coefficient;.

the number of raw data, i.e. 720. The number of transmission in Tab. 5.1 is actually the number of aggregated packets. We note that A^2 holds smaller number of transmission, and then has smaller aggregation ratio comparing to A-ARMA. When AICc is used to select the model order, the aggregation ratio is lowest (7.9%), which saves more communication cost than A-ARMA. Regarding to the packet size coefficient, we assume that the packet size is proportional to the number of parameters inside the packet. Thus original packet's size is 1 (one raw data in one packet), and aggregated packet's size is the number of parameters. Under such assumption, we are able to calculate the average aggregated packet size as $\frac{number\ of\ parameter}{number\ of\ transmission}$, and we compute the packet size coefficient using definition 3.1.2. The results are given in Tab. 5.1. We note that A^2 achieves smaller packet size coefficient than A-ARMA, and the lowest coefficient is performed also using AICc. Thus in the following experiment, we only consider AICc as the criterion to select model order as its better performance.

We plot both the predicted data and raw data in Fig. 5.6, using the same parameters in Tab. 5.1. From Fig. 5.6 and Tab. 5.1, the accuracy of A^2 is better than A-ARMA: the predicted data by A^2 are more approaching the raw data.

Besides A-ARMA, we also compare A^2 with other aggregation functions. Polynomial aggregation [31] is a forecasting function using polynomial to predict data, and sink uses the polynomial parameters to recover data. FIX [115] aggregates fixed number of data, and we use average as the aggregation function (marked as FIX-average), to compare with A^2. Correspondingly, the number of the fixed data is the length of a window. We investigate accuracy and energy consumption among the three functions following.

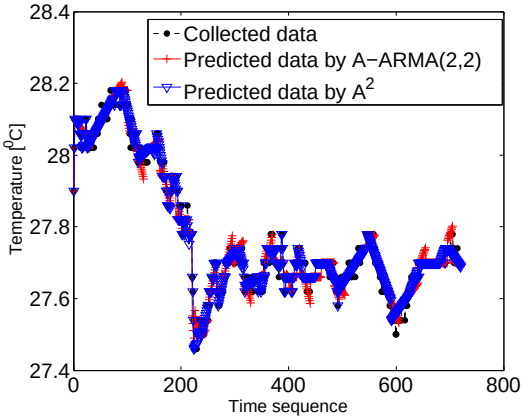
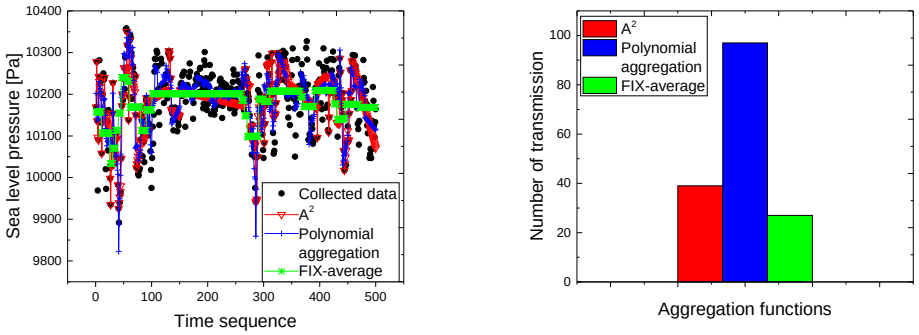


Figure 5.6: A^2 v.s. A-ARMA(2,2), the comparison between predicted data and raw data using dataset $\mathcal{T}_m$.

Fig. 5.7 shows the accuracy and number of transmission from the three aggregation functions considering dataset $\mathcal{P}_s$, where the data of sea level pressure are more dynamic than temperature data. We set window $W = 20$, step $S = 5$ and error threshold $th_{err} = 75$. In Fig. 5.7(a), RMS errors for A^2, Polynomial, FIX-average are 47.78, 50.25, 71.94 respectively. A^2 shows more fidelity than the other two aggregation functions, and FIX-average performs worst.



(a) Accuracy issues from the 3 aggregation functions.

(b) Number of transmission comparison among the 3 aggregation functions.

Figure 5.7: Comparisons between A^2, Polynomial aggregation and FIX-average using dataset $\mathcal{P}_s$.

Fig. 5.7(b) shows the number of transmissions from the three aggregation functions, where FIX-average keeps the lowest updates, and A^2 has a few more. This is because FIX-average uses a window as the number of fixed data: it means every 20

data will be aggregated directly. However, considering the recovered accuracy, FIX-average sacrifices too much accuracy to reduce the number of transmission. In terms of polynomial aggregation, it always trains new model when the accuracy cannot meet threshold, and does not consider data property, thus polynomial aggregation shows huge number of transmission than A^2 and FIX-average even though it can guarantee relatively high accuracy.

Introduced error

The environment for the nodes is unpredictable, which may introduce unforeseeable errors to the original sensing data, i.e., the abrupt fail of sensing components. Aggregation function should be robust to handle these situations. In addition, we are motivated to investigate the performance of our proposition under erroneous data.

The errors can be independent (each error affects only one sample) or consecutive (errors affect consecutive samples). We simulate several times the introduction of random errors in the data. After giving the results, we discuss the influence of the errors on the performance of A^2.

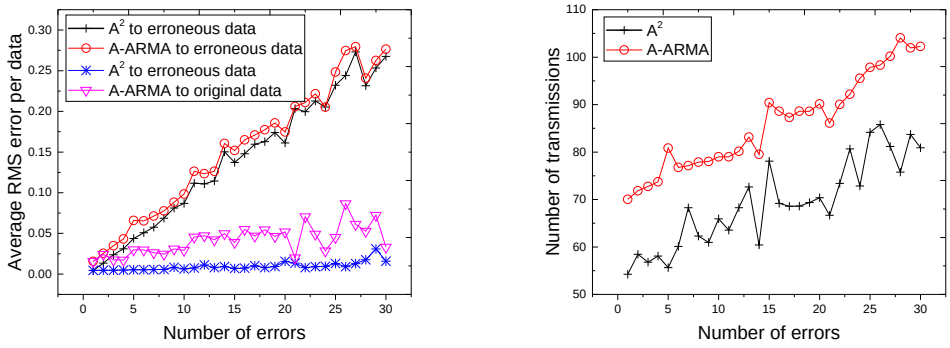
Independent errors

Fig. 5.8 compares the impact of independent errors on the accuracy and number of transmission of A-ARMA$(2, 2)$ and A^2. The accuracy of both models drops (i.e., the average RMS error increases) when more independent errors are introduced into the data (Fig. 5.8(a)). However, A^2 achieves better accuracy regardless to the erroneous and original data. In addition, the number of transmission for A^2 is lower than that for A-ARMA$(2, 2)$ (Fig. 5.8(b)). This highlights that when data are affected by errors, A^2 tends to correct these errors because of its optimal order, which makes the prediction values closer to the original data.

Similar as A-ARMA, A^2 also needs a correct model at the beginning to guarantee the accuracy of following steps, because errors at the beginning of a time series have a much higher impact on the estimation. If there are more independent errors introduced within the train process for building model, then the estimation of the model fails (the predicted values can not match the actual ones). So if at the beginning, the model is correct, the order adaptation is the most important step to improve the accuracy, that is why A^2 performs better than A-ARMA (fixed order).

Consecutive errors

For each analysis, errors of different length are applied while the total number of errors is limited to 30. For example, if the length of the error burst is five errors, then there are six error bursts in the dataset. As shown in Fig. 5.9(a), the accuracy of

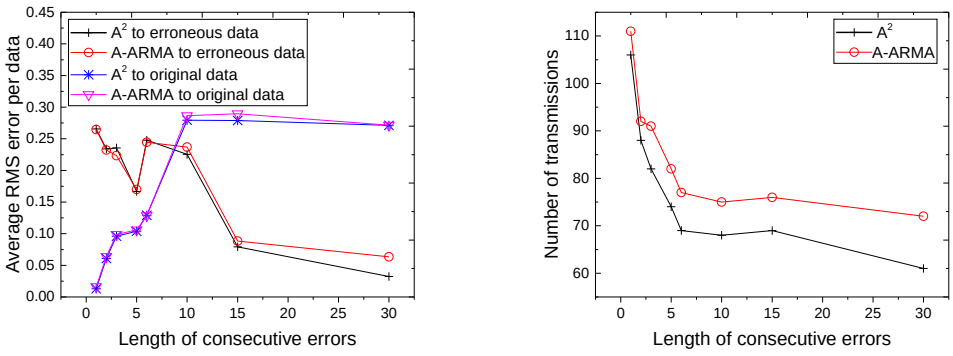


(a) Impact of independent errors on the accuracy.

(b) Impact of independent errors on number of transmission.

Figure 5.8: Impact of independent errors on A^2 and A-ARMA(2,2).

A-ARMA$(2, 2)$ and A^2 drops due to the consecutive error existing. When the consecutive errors are longer, the easier these errors are modelled by A^2 and A-ARMA$(2, 2)$, therefore the accuracy of erroneous data is improved.



(a) Impact of consecutive errors on the accuracy.

(b) Impact of consecutive errors on number of transmission.

Figure 5.9: Impact of consecutive errors on A^2 and A-ARMA(2,2).

The number of transmissions is shown in Fig. 5.9(b): as the length of a consecutive error increases, the number of transmissions reduces. This is because longer consecutive errors are tend to be considered as the normal data, and the model will fit these data. Meanwhile, we can see that A^2 always generates lower transmissions comparing to A-ARMA despite the length of consecutive errors.

A^2 and A-ARMA$(2, 2)$ exhibit similar characteristics, while obviously, A^2 guarantees the accuracy with less number of transmission. The accuracy of A^2 and A-ARMA

$(2, 2)$ are all impacted by the independent errors, but A^2 is more robust than A-ARMA (see Fig. 5.9(a)). The length of consecutive errors has a low impact on the accuracy. Then, no matter which type of error, A^2 is better and more efficient to limit the RMS error, i.e., A^2 holds the fewer additional model transmission than A-ARMA$(2, 2)$ does.

5.5.2 Comparison using synthetic dataset

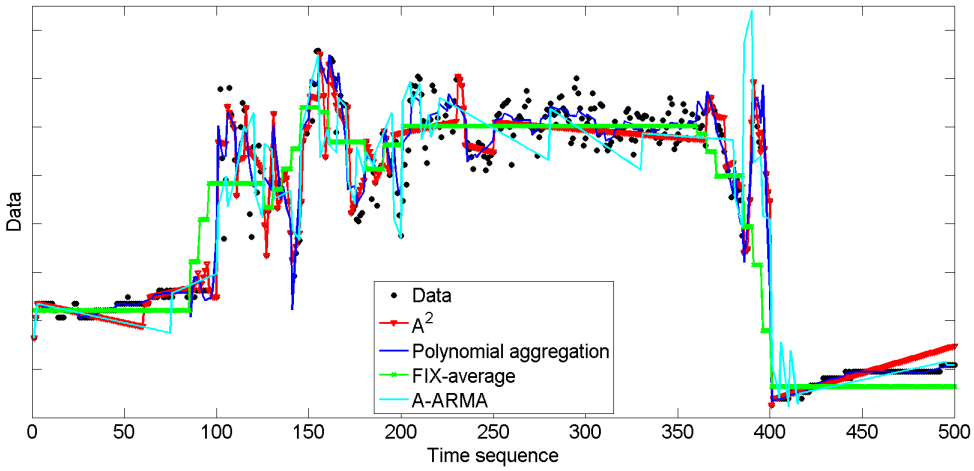
A dataset is often composed by the data with same property, e.g. $\mathcal{T}_m$ includes temperature data only, and $\mathcal{P}_s$ is the set of sea level pressure data. In order to show the capability of A^2 on self-tuning to fit the evolution of the data, we also consider a data sequence with different data properties We construct a dataset from the two datasets: the first 100 data from dataset $\mathcal{T}_m$, the following 300 data from $\mathcal{P}_s$ and the last 100 data from $\mathcal{T}_m$. We set $S = 5$, $th_{err} = 75$ and sliding window $W = 20$ to check the performance of the functions.

In Fig. 5.10(a), we plot the synthetic data and the predicted data by the four functions. Obviously, we can see that FIX-average and A-ARMA cannot fit the data when the property changes. Around sequence 100 and 400, FIX-average shows large deviation from real data. Regarding A-ARMA, during sequence 200 to 350, it can not predict data well due to the fixed order (ARMA(2,2)). The accuracy (RMS error) for FIX-average, A-ARMA, polynomial aggregation and A^2 is 79.63, 73.53, 37.97 and 37.29 respectively, where A^2 achieves the best accuracy, and polynomial aggregation also perform good accuracy. However, as the number of transmissions which is shown in Fig. 5.10(b), we find that polynomial aggregation uses almost 2 times more transmissions than A^2. In Fig. 5.10(b), FIX-average has the lowest number of transmissions, but it performs the worst accuracy (79.63). A-ARMA uses lower transmission but the accuracy is not acceptable (73.53). Under such synthetic data, other functions either increase transmission to guarantee accuracy (such as polynomial aggregation), or sacrifice accuracy to keep low transmission (such as FIX-average), while A^2 fits the different properties as soon as possible, and guarantees the accuracy under lower transmission.

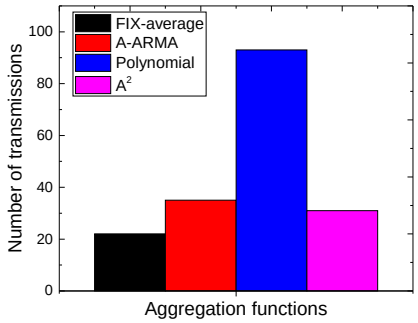
Comparing with A-ARMA, Polynomial aggregation, and FIX-average, A^2 has higher recovered accuracy and lower transmission, it can dynamically change model to adapt the data property to fit the data better than other functions.

5.6 Conclusion

In this chapter, we present a property-independent aggregation function, Agnostic Aggregation (A^2), which can self-tune the model as the change of data property. The



(a) Comparison between synthetic data and predicted data by the functions.



(b) Accuracy, number of transmission of the functions.

Figure 5.10: Performance of aggregation functions using synthetic dataset.

self-tuned mechanism makes the model more approaching the real data. The nodes locally train the model with dynamic order when the error crosses a given threshold. The experiment shows that A^2 performs better than other aggregation functions in accuracy and energy efficiency.

Moreover, considering real datasets, we point out that for good performance, the range of model order could be bounded by 5. The moving window should be equal to the number of data for building model. Comparing with other aggregation functions using either real datasets or synthetic dataset, A^2 achieves the best accuracy within lower transmission.

6

Classification of data aggregation functions using Markov Decision Processes

Contents

There are a lot of contributions about data aggregation functions in WSNs, but there is a lack of a formal unified framework that can classify them according to a given application and a target accuracy. Application here denotes the data, and the target accuracy is the requirement of accuracy constraint (accuracy constraint is the error between the raw data in the sensor side and the forecasting data at the sink level). We classify the aggregation functions in this chapter, from perspectives of performances evolution, data distribution and accuracy constraints. The performance is evaluated using the networking-oriented metrics (aggregation ratio ω and packet size coefficient λ, see Sec. 3.1), and we use Markov Decision Process to compute them. Data distribution is sampled from the real datasets. In this chapter, we are focusing only on forecasting aggregation only.

6.1 MDP model for aggregation functions

6.1.1 Brief introduction of MDP model

Markov decision process (MDP) [116, 117] is a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision agent. The decision agent is faced with the problem, and it is in charge of making decision. The goal of decision agent is to choose a sequence of actions which causes the system to perform optimally with respect to several predetermined criterion.

A MDP model is composed by 4 elements:

1) ***state space*** Ω, which is the set of all possible states of the system;

2) ***action set*** A, that is the action that can transfer from one state to another;

3) ***transition probabilities*** T, which is probability of one state to another state with a certain action;

4) ***rewards*** R, which represent the benefits obtained if agent chooses one of the action.

In Appendix C, we provide an overview of MDP fmodel in details and the computation methodology.

To the best of our knowledge, there is few literature related to classification of aggregation functions. A similar work uses game theory [118] to build model to do aggregation [119]. In this work, the authors deal with the conflicts between data obtained from different nodes. However, game theory is a solution for the situations having more decision agents, while in our context, we consider only one decision agent. The authors in [35] treat data aggregation as an *optimal stopping* problem, and they use MDP to determine when a node needs to do aggregation (but without consideration on accuracy). *Optimal stopping* is concerned with the problem of choosing a time to take a particular action, in order to maximise an expected reward. The objective of data aggregation is aggregating more data packet and to achieve better aggregation performance. Thus we believe that MDP framework is well adapted to evaluate aggregation functions when data arrives following a random process, which is typically the case in many WSNs application cases. Moreover, the MDP model can be extended to integrate other random raw data, such as the transmission opportunity at MAC layer, or packets losses.

6.1.2 System behaviour

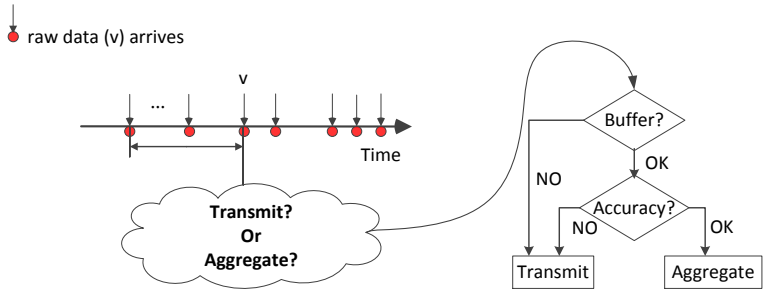


Figure 6.1: Node needs to make decision when a new data v arrives.

We firstly explain how aggregation works in real situation (as shown in Fig. 6.1). Because we model forecasting aggregation functions, so we consider data series generated by only one node: it means we deal with temporal correlation. From the perspective of aggregation, the node makes a decision when the new data v_i arrives. Considering constraints of buffer size and accuracy requirement,

- if buffer is not full and accuracy is under the given threshold, node aggregates v_i (it means nodes use aggregation function to aggregate v_i into the current aggregated packet);

- Otherwise, node transmits previous *aggregated packet* to sink, and new aggregation phase begins from v_i.

It is a standard *optimal stopping* problem, nodes want to have more data aggregated and achieve better aggregation performance.

Forecasting aggregation functions have a similar constraint, that is they all need accuracy constraint to determine if they need to transmit or not when a new data arrives. Markov Decision Process is a suitable framework to decide the appropriate action: transmit or aggregate, in order to achieve the optimality objective. Our optimality objective is to decrease the aggregation ratio ω. Based on this view, we describe our system description and MDP model in the following.

6.1.3 Model formulation

We assume that the accuracy constraint is fixed and equal to α and the data distribution is given for our model. Data distribution is composed by set $\mathbb{E}$ of possible raw data, v. The possible values of raw data is finite and its dimension is $\|\mathbb{E}\|$. The corresponding distribution probability $P(v)$ denotes the probability that data v arrives. Obviously, $\sum_{v\in\mathbb{E}} P(v) = 1$.

In forecasting aggregation functions, several functions use sliding window technique to achieve lower computation, like A-ARMA (see Sec. 2.2.2), others (like average) do not use it. Thus, we separate them into functions without sliding window and with sliding window. We firstly describe the model formulation for the functions without sliding window.

For functions without sliding window

The state space Ω is defined as the set of all possible states of the system. In our case, a state, denoted as x, includes all the buffered raw data where the buffered raw data is the set of all raw data received since the last transmission to the sink. At the beginning of aggregation phase, there is only one value in the buffer, so $x = (v_0)$ where $v_0 \in \mathbb{E}$, and this type of state is denoted as *initial state*. When a new data v_1 arrives, and the accuracy constraint α is meet, $x = (v_0, v_1)$. In real sensor application, sensor memory and computation are limited, it can not store all of the raw data if there is no transmission during a long time. Thus we assume that there is a buffer size limitation, which we denote Dim. Therefore, $1 \leq \|x\| \leq Dim,\ \forall x \in \Omega$. We can now describe the state space Ω as follow:

$$\Omega \equiv \{x = (v_0, \cdots, v_i) |\ 0 \leq i \leq Dim - 1, \forall v_i \in \mathbb{E}\} \tag{6.1}$$

The action set A When the system is in state x, and a new data v arrives, the decision agent should decide between two possible actions: 1) *aggregate*, means to aggregate the new value v with the previous data since the last transmission; or 2) *transmit*, means to stop current aggregation phase, to transmit aggregated packet to sink, and then, to start a new aggregation phase with v. *Transmit* is represented by action $a = 1$, whereas *aggregate* is represented by $a = 0$. Because there are $\|\mathbb{E}\|$ possible values, the set of possible actions in state x, denoted $A(x)$, is a binary vector of dimension $\|\mathbb{E}\|$. Formally,

$$A(x) = \{(a_0, \cdots, a_i) | i = \|\mathbb{E}\| - 1, a_j \in \{0, 1\}, 0 \leq j \leq i\} \tag{6.2}$$

Therefore, the action set A is the union of $A(x)$ for all $x \in \mathbb{E}$.

The transition probabilities T is highly related to the data distribution, which describes the transition probability from one state to another due to different raw data and actions. When the system is in the state $x = (v_0, \cdots, v_i)$, and a raw data v_{i+1} arrives, the agent chooses the corresponding action a. If accuracy between state x and state (x, v_{i+1}) is lower than the accuracy constraint α, then action a is determined by the MDP agent. Otherwise, the action should be *transmit*, $a = 1$. In this case, the system transits from state x to an initial state (v_{i+1}). Since the raw data v_{i+1} arrives with probability $P(v_{i+1})$, then the transition probability from state $x = (v_0, \cdots, v_i)$ to state y when a is chosen can be expressed as:

$$T_{x \to y}(v_{i+1}, a) = \begin{cases} (1 - a) \cdot P(v_{i+1}) & if\ y = (v_0, \cdots, v_{i+1}) \\ a \cdot P(v_{i+1}) & if\ y = (v_{i+1}) \end{cases}$$

The set of transition probability is given by:

$$T \equiv \{T_{x \to y}(v, a) | \forall (x, y) \in \Omega^2, \forall a \in \{0, 1\}, \forall v \in \mathbb{E}\} \tag{6.3}$$

The rewards R define the benefits earned by the system when the MDP agent choose the corresponding action. The purpose of aggregation is reducing traffic and saving energy, i.e., decreasing the number of transmitted packets. When node chooses *aggregate* mode (action $a = 0$), it saves one more packet. Therefore, we decide to reward the system by one unit only when the MDP agent chooses action *aggregate*. The reward perceived by the system when the MDP agent chooses action a that is associated to the raw data v when the system is in state x can expressed as:

$$R(x, v, a) = 1 - a$$

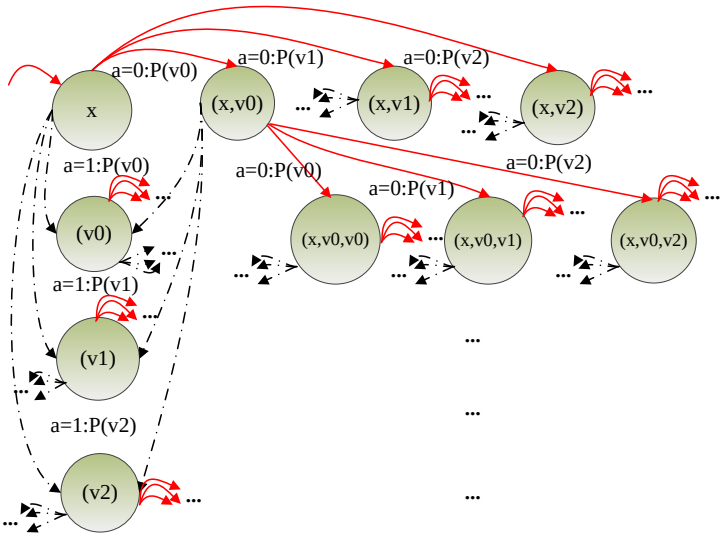


Figure 6.2: An example for MDP chain (for functions without sliding window), red lines denote reward is 1, and black dash-dot lines show reward is 0.

The set of rewards R is given by:

$$R \equiv \{R(x, v, a) | \forall x \in \Omega, \forall a \in A, \forall v \in \mathbb{E}\} \tag{6.4}$$

Fig. 6.2 shows a MDP chain for our model of functions without sliding window, assuming $\mathbb{E} = \{v_0, v_1, v_2\}$ is the possible values of raw data, and $P(v_0)$, $P(v_1)$ and $P(v_2)$ represent the probability of the three values respectively. The initial states are $(v_0), (v_1), (v_2)$, and we suppose that the system is now in state $x \in \Omega$. Action $a = 0$ represents aggregating with new raw data and $a = 1$ represents stopping the current aggregation and transmitting to the sink. On state x, when node decides to aggregate with the new data, state x can be transferred to $x \cup v_i$ with a transition probability corresponding to the arrival of the new raw data. With this action $a = 0$, MDP earns 1 reward, illustrated as red lines in Fig. 6.2. When the node decides to transmit, i.e., $a = 1$, the state x goes back to one of the initial states, from x to (v_i). In the latter case, the system is not rewarded as demonstrated by black dash-dot lines in Fig. 6.2.

For functions with sliding window

In terms of aggregation functions with sliding window, they compute parameters based the latest W raw data, where W is the the length of sliding window. Thus, instead of transmitting raw data, we consider that such aggregation functions only transmit the coefficients to sink. And the coefficients are used to predict the next data

by the function definitions. When a new data arrives, sensor node firstly compares the predicted data and the new one. If the error meets the accuracy constraint (α), the node keeps the same coefficients; otherwise, sensor node uses the last W raw data to compute new coefficients, and transmits them to sink. A node needs to save the coefficients in the whole aggregation processing, thus to model A-ARMA or polynomial aggregation, we also need to consider the sliding window technique.

A state, x, includes the coefficient part and the current window part (denoted as current W part), such as $(\underbrace{v_0, v_1, \cdots, v_{w-1}}_{coefficient\ part} | \underbrace{v_0, v_1, \cdots, v_{w-1}}_{current\ W\ part})$. The coefficient part is used to compute the current coefficients (the coefficients are calculated from the W data), the current W part is used to record the current W data. When a new data v_i arrives, state x firstly uses current coefficients to predict value $\dot{v}_i$, if the difference between v_i and $\dot{v}_i$ obeys accuracy constraint α, the coefficient part does not change, we update the current W part as $(v_0, v_1, \cdots, v_{w-1}|v_1, \cdots, v_{w-1}, v_i)$. If the difference violates α, the coefficient part and current W part all changes as $(v_1, \cdots, v_{w-1}, v_i|v_1, \cdots, v_{w-1}, v_i)$. The only difference with Sec. 6.1.3 is the state space. Action set A, transition probability T and the rewards R are the same with Eq. 6.2, Eq. 6.3 and Eq. 6.4 respectively.

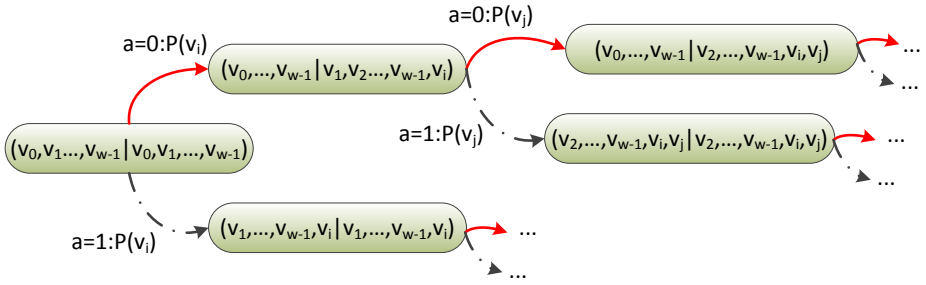


Figure 6.3: An example of model chain (for function with sliding window), red lines denotes action is 0, and black dash-dot lines demonstrate action is 1.

Fig. 6.3 shows a MDP chain for functions with sliding window. Supposing v_i has been aggregated, and now the state x is $(v_0, \cdots, v_{w-1}|v_1, v_2, \cdots, v_{w-1}, v_i)$. When v_j arrives, considering the accuracy constraint α, there are two possibilities: one is aggregated with v_j, another is using v_j to calculate the new coefficients and transmitting to sink. If accuracy is met, i.e., $a = 0 : P(v_j)$ in Fig. 6.3, v_j is added into the end of current W part, the coefficient part does not change. It means the current coefficients of sate x can meet the accuracy requirement, and the new state is $(v_0, \cdots, v_{w-1}|v_2, \cdots, v_{w-1}, v_i, v_j)$. If accuracy is not met, i.e., $a = 1 : P(v_j)$ in Fig. 6.3, the coefficients of state x cannot guarantee the accuracy, thus the state x needs to transfer to a new state as $(v_2, \cdots, v_{w-1}, v_i, v_j|v_2, \cdots, v_{w-1}, v_i, v_j)$. The new state uses new coefficient part to calculate coefficients and transmit.

For the coefficient part (W data) of a state, there are $\|E\|^{2*W}$ possibilities ($\|E\|$ is the possible values of raw data). As the increase of $\|E\|$ or W, the possibilities grow exponentially. Thus considering the computation complexity, we propose to sample several data as the state space, and we will detail sampling methodology in Sec. 6.2.

6.1.4 Networking-oriented metrics computation

Policy iteration or value iteration algorithms [117] are proposed to solve MDP and get optimal policy d^* for each state. The optimal policy in our context is a policy which is able to aggregate more data, i.e. obtaining a smaller aggregation ratio. Our objective is classifying aggregation functions according to performance, the data distribution and the accuracy constraint. For the performance, we use the networking-oriented metrics proposed in Sec. 3.1: *Aggregation ratio*, and *Packet size coefficient*. Remember that *Aggregation ratio* (ω) is the ratio of number of aggregated packets to number of generated packets, and *Packet size coefficient* (λ) is the ratio of size of *Aggregated packet* to size of original packets that have been aggregated.

With the optimal policy d^*, MDP model changes to a Markov chain, as shown in Fig. 6.4. Note that several states are not reachable, because the optimal policy d^* or accuracy constraint remove several transferring. For example, in Fig. 6.4, when v_2 arrives, assuming the accuracy between state x and state (x, v_2) exceeds the threshold, thus x cannot transfer to (x, v_2), and it only goes to initial state (v_2) with $a = 1 : P(v_2)$. Thus (x, v_2) is unreachable state, and it is removed in the associated Markov chain.

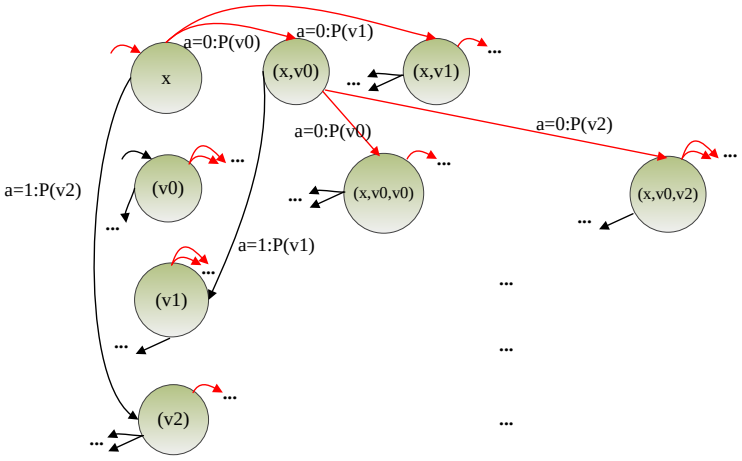


Figure 6.4: The corresponding Markov Chain (without sliding window).

Here, we use Ω' to denotes the present state space, i.e., Ω' includes all the reachable states from Ω. Note that $\Omega' \subseteq \Omega$, and all the states in Ω' can go to other states or go

from other states. And we mark the associated transition probability as T'. Therefore, we build a discrete-time Markov chain (DTMC) based on Ω', T' is a subset of T from MDP, which can be formulated as: $T' \equiv \{T'_{x\to y}(v) | \forall (x,y) \in \Omega'^2, \forall v \in \mathbb{E}\}$, where:

$$T'_{x\to y}(v) = \begin{cases} (1-a^*)\cdot P(v) & if\ y = x \cup v \\ a^* \cdot P(v) & if\ y = v \end{cases} \tag{6.5}$$

a^* is the MDP optimal action associated to state x and raw data v.

Considering this DTMC, we can compute the steady state [120], denoted as π. [121, 122] propose numerical solutions for computing steady state, either direct technique or iterative numerical methods can be used. We calculate steady state of our DTMC using Gaussian Seidel iteration [123].

We formulate networking-oriented metrics with steady state π. Firstly, *Aggregation ratio* ω, which denotes the ratio between packet transmitted and packet generated, it can be expressed as:

$$\omega = \frac{\sum\limits_{x\in\Omega', v\in\mathbb{E}} \pi(x)\cdot a^* \cdot P(v)}{\sum\limits_{x\in\Omega', v\in\mathbb{E}} \pi(x)\|x\| \cdot a^* \cdot P(v)} \tag{6.6}$$

In this formulation, $\pi(x)$ denotes the stationary probability of state x; $\|x\|$ is the dimension of state x, which shows the number of raw data buffered in state x. Packet transmitting is formulated by action $a^* = 1$, for state x, when raw data is v, the probability of transmitting is $a^* \cdot P(v)$.

Secondly, *Packet size coefficient* λ, which describes the ratio between the size of the aggregated packet compared to the original one, assuming the original packet size is a constant, p_{size}, is given by Eq. 6.7:

$$\lambda = \frac{\sum\limits_{x\in\Omega', v\in\mathbb{E}} \pi(x)\cdot a^* \cdot P(v) \cdot \frac{p'_{size}}{p_{size}\|x\|}}{\sum\limits_{x\in\Omega', v\in\mathbb{E}} \pi(x)\cdot a^* \cdot P(v)} \tag{6.7}$$

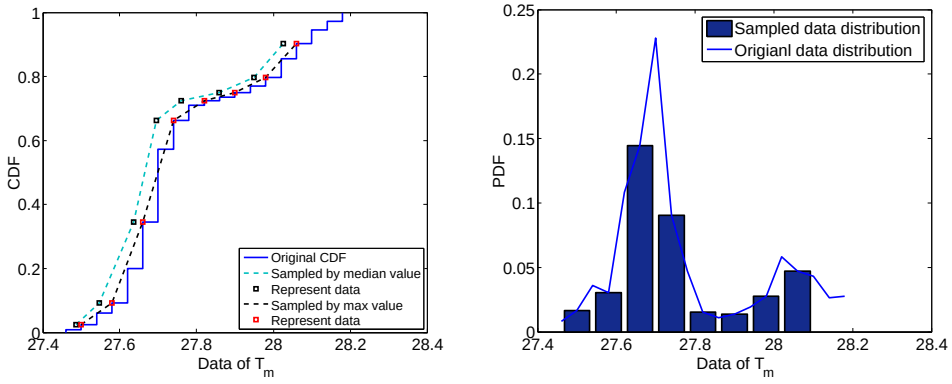
Similar with ω, $a^* \cdot P(v)$ shows the probability of transmitting when raw data is v, $\frac{p'_{size}}{p_{size}\|x\|}$ describe the ratio between aggregated packet size and original sizes. We define the range of λ is no smaller than 1 (see in Definition 3.1.2), thus if an aggregation function does not change packet size, the associated packet size coefficient is equal to 1 (e.g. average). While for A-ARMA and polynomial aggregation, they use coefficients to predict or estimate data, thus the size of the coefficients (content of aggregated packets) is different to size of original packets. Therefore, to these two aggregation functions, we need to compute their packet size coefficients.

6.2 Data distribution sampling

Considering the MDP modelling, we need the data distribution as the input to compute the networking-oriented metrics. Thus, we propose to sample data from real datasets. Due to computation limitation as we mentioned in the end of Sec. 6.1.3), we can use only few values. In our case, we study 8 values (see Sec. 6.3.1). We discuss in this section how we can choose few representative values from real dataset. First, we separate all the different data into 8 uniform intervals. Assuming an interval $i \in [1, 8]$ includes values such as $[v_1^i, v_{n_i}^i]$, n_i is number of values in the interval, and the values from v_1^i to $v_{n_i}^i$ are in ascending order. Second, we compute the PDF (Probability Density Function) for each interval as the data distribution. Finally, we choose one value from each interval to represent the interval. We consider two values as representatives, one is the max value v_{max}^i, another is median value v_{med}^i. It is obvious that $v_{max}^i = v_{n_i}^i$ due to the ascending order, and v_{med}^i can be computed as:

$$v_{med}^i = \begin{cases} v_{(n_i+1)/2}^i & \text{if } n_i \text{ is odd} \\ \frac{v_{n_i/2}^i + v_{(n_i+1)/2}^i}{2} & \text{if } n_i \text{ is even} \end{cases}$$

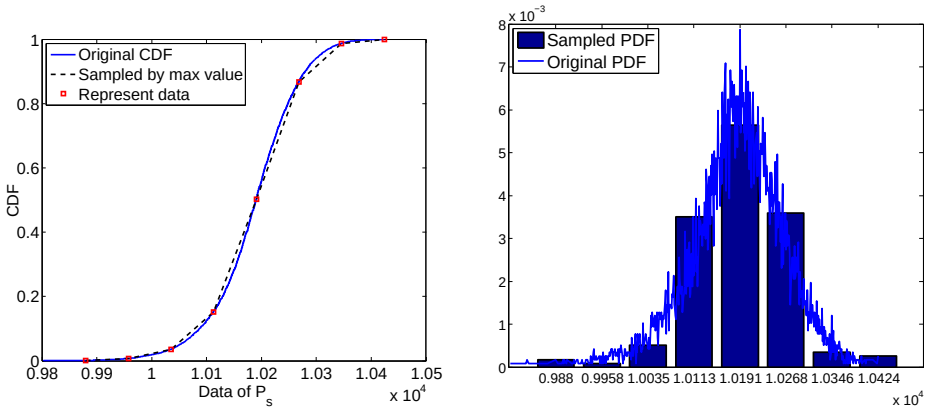
As shown in Fig. 6.5(a), we plot max value v_{max}^i and median value v_{med}^i respectively, and we find that max value of each area can fit the original CDF better, thus we choose max value of each interval as the represent data. Fig. 6.5(b) is the sampled PDF use max data.



(a) CDF of $\mathcal{T}_m$. (b) PDF of $\mathcal{T}_m$.

Figure 6.5: CDF and PDF of dataset $\mathcal{T}_m$.

Similarly, consider different dataset, we use same sampling method. For dataset $\mathcal{P}_s$, the sampled CDF and PDF are shown in Fig. 6.6.



(a) CDF of $\mathcal{P}_s$. (b) PDF of $\mathcal{P}_s$.

Figure 6.6: CDF and PDF of dataset $\mathcal{P}_s$.

6.3 Performance evaluation

6.3.1 Computation complexity

Our model faces a problem of the size of state space $\|\Omega\|$, because the state is a sequence of raw data, even though we assume a limited buffer size (Dim). When Dim increases, the dimension increases exponentially. Thus, the space complexity of our MDP is a crucial question we faced with. As mentioned in Sec. 6.2, we study 8 possible data, i.e. $\|\mathbb{E}\| = 8$. The impact of Dim on size of sate space $\|\Omega\|$ is shown in Fig. 6.7. We note that when Dim is bigger than 8, size of state space $\|\Omega\|$ becomes bigger than hundred millions. When the size of state space increases, elapsed time for solving MDP also increases exponentially, and if Dim is too large, we cannot solve it due to memory limitation. Thus in our following experiments, we consider $Dim = 8$. We also shows that $Dim = 8$ is an appropriate choice to analyse aggregation functions in Sec. 6.3.4.

6.3.2 Simulation methodology

In this section, we discuss the simulation methodology that we follow to validate our model against simulations. We use data distributions illustrated in Fig. 6.5 to randomly generate 20 data sequences, and each sequence has 500 data. The simulation results that we will present are an average of 20 simulations with a 95% confidence interval. For this data distribution, we consider a root mean square (RMS) error accuracy constraint α from 0.01 to 0.1. We detail simulation of different aggregation functions as following:

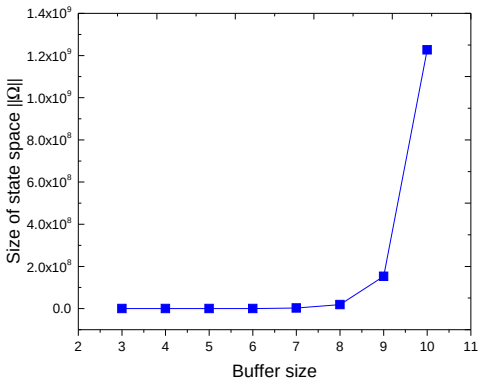


Figure 6.7: Size of state space $\|\Omega\|$ increases exponentially with buffer size limitation Dim.

- Average: Node does not transmit if buffer is not saturated and accuracy meets the constraint. Otherwise, node needs to transmit the latest average value. In simulation, when a new raw data arrives, the node compares the RMS error between current average value and the new raw data: if accuracy satisfies α and buffer is not overflowed, it keeps the *aggregated packet* and wait for the next value. If buffer is overflowed, whatever accuracy is met or not, it needs to transmit.

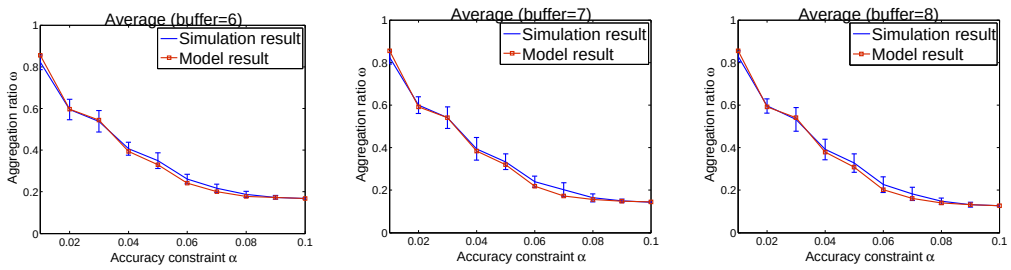
- A-ARMA(2,2): A-ARMA model needs to train values to generate forecasting coefficients, and the coefficients are composed of *aggregated packet* for A-ARMA. We assume that the size of sliding window is equal to the buffer size. Because A-ARMA(2,2) generates 5 coefficients [30], thus the minimal buffer size starts from 5.

- Polynomial aggregation: [31] shows that order 4 is usually enough to approximate the original time series according to the error threshold. Therefore, maximum order of polynomial in our simulation is 4, correspondingly, the maximum parameters generated by polynomial function is 5, same as A-ARMA. Moreover, we also consider a sliding window for polynomial aggregation.

6.3.3 Model validation

In this section, we show the theoretical results and simulation results in the same figure, to validate our model. Meanwhile, we investigate the impact of buffer size on aggregation ratio.

For the function of Average, the packet size remains constant, so we only calculate *Aggregation ratio* ω. We simulate the situation with buffer varying from 2 to 8, and accuracy constraint from 0.01 to 0.1. The data sequences are generated using the same distribution as the dataset $\mathcal{T}_m$ (Fig. 6.5). Fig. 6.8 shows both the theoretical and simulation results, considering a buffer size of 6, 7 and 8. From Fig. 6.8(a), when accuracy constraint $\alpha = 0.09$ and buffer is 6, ω is around 0.2. While in Fig. 6.8(c), ω is around 0.1 with buffer 8. Remember that the lower aggregation ratio is, the more energy saves. Thus buffer size affects ω to some extent, with bigger buffer, node can achieve lower ω. But, from Fig. 6.8(b) and (c), we can see that there is no difference with buffer 7 and buffer 8. It means bigger buffer is not always better, when buffer is enough big, there is no significative change for the aggregation ratio.

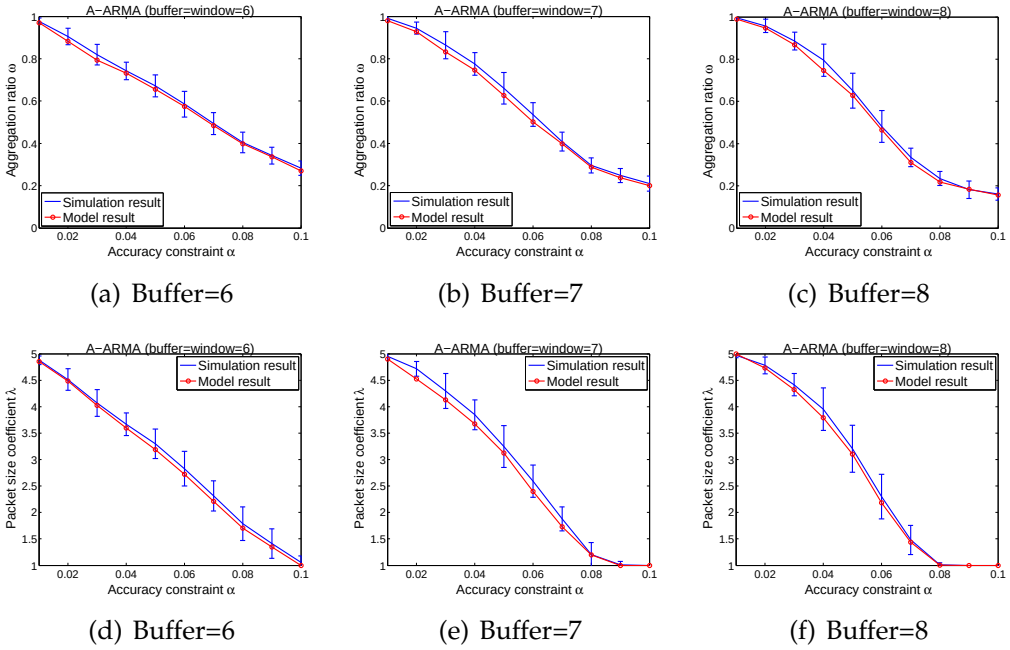


(a) Buffer=6 (b) Buffer=7 (c) Buffer=8

Figure 6.8: Validation of MDP model, aggregation ratio ω considering aggregation function as Average (simulation result is show in 95% confidence interval).

Considering A-ARMA, the buffer limitation also varies from 6 to 8, and accuracy constraint is from 0.01 to 0.1. Fig. 6.9 presents the results of aggregation ratio for A-ARMA with buffer 6, 7 and 8. Whatever is the buffer size, the theoretical result fits simulation well. Similar as Average, different buffer size shows different range of ω, but the difference between buffer 7 and buffer 8 is not so obvious (when $\alpha \geq 0.09$). It also shows that big buffer is not always better, when buffer is enough big, there is no change for aggregation ratio. We also plot the packet size coefficient of A-ARMA in Fig. 6.9(d)(e)(f). Assuming the size of packet is proportional to the number of coefficients in the packet. Thus for A-ARMA(2,2), the maximum number of coefficients in an aggregated packet is 5, and the maximum λ is considered as 5. When accuracy constraint is strict (small α as 0.01), A-ARMA proposes a bigger aggregated packet to transmit; as the constraint releases, λ decreases because one aggregated packet can represent more original packets.

For polynomial aggregation, we consider the same settings as for A-ARMA. The networking-oriented metrics (ω and λ) are shown in Fig. 6.10. We can see with different buffer, the results of our theoretical model fit the simulation well. Similar as above



(a) Buffer=6 (b) Buffer=7 (c) Buffer=8

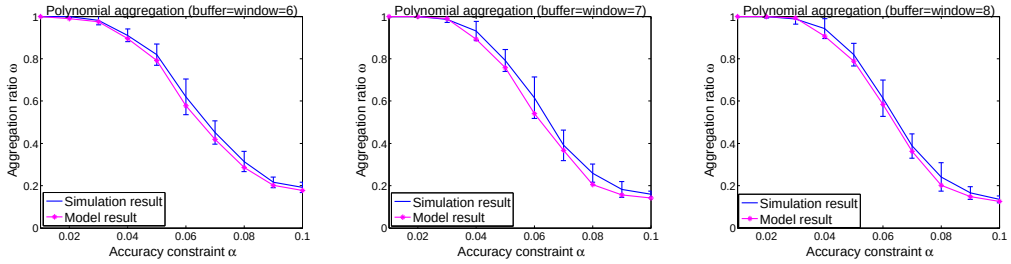
(d) Buffer=6 (e) Buffer=7 (f) Buffer=8

Figure 6.9: Validation of MDP model, networking-oriented metrics, ω and λ considering aggregation function as A-ARMA (simulation result is show with 95% confidence interval).

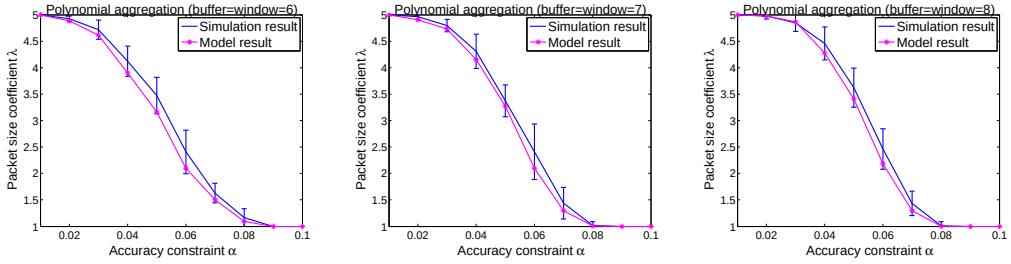
functions, different buffer size shows different range of ω, but the difference between buffer 7 and buffer 8 is not so obvious. It also proves our above idea: when buffer is enough big, there is no impact on aggregation ratio. For the polynomial, the upper bound of order is 4, thus the maximum number of coefficients is 5 (see Eq. 2.13), and the maximum λ is 5 (same as A-ARMA). Because A-ARMA and polynomial aggregation all use regression method to predict data, thus the shape of networking-oriented metrics are similar with each other.

6.3.4 Buffer size discussion

As discussed in our model description, when buffer limitation Dim increases, $\|\Omega\|$ increases exponentially (Fig. 6.7). Therefore, in the model, we set Dim to 8. Using average aggregation function, we generate 20 group data sequences with length as 500 from data distribution, and we consider the buffer size from 2 to 20, setting an accuracy constraint $\alpha = 0.06$. Plotting average of 20 experiments with a confidence interval 95% in Fig. 6.11, we can see that as the buffer increases, the decreasing trend of aggregation ratio is slowing down. When buffer size is bigger than 8, ω remains quite stable, i.e., the value of ω gradually converge with the buffer size. This is the same situation with A-ARMA and polynomial aggregation: from Fig. 6.9(b)(c) and



(a) Buffer=6 (b) Buffer=7 (c) Buffer=8



(d) Buffer=6 (e) Buffer=7 (f) Buffer=8

Figure 6.10: Validation of MDP model, networking-oriented metrics ω and λ considering aggregation function as polynomial aggregation (simulation result is show with 95% confidence interval).

Fig. 6.10(b)(c), ω almost shows the same value with buffer 7 and buffer 8. Thus, it is not necessary to test all the possible buffer size: buffer of 8 is an appropriate choice for analyzing aggregation functions.

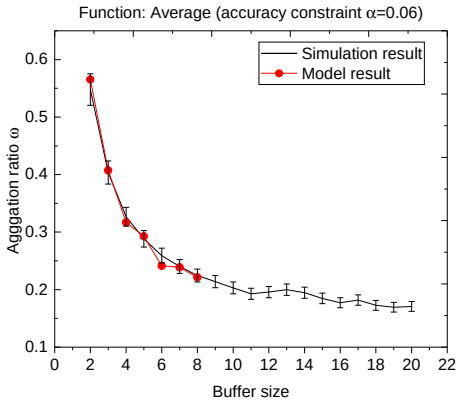


Figure 6.11: Considering function as Average, simulation with buffer size from 2 to 20, accuracy constraint is 0.06.

6.4 Classification of aggregation functions

In this section and based on our MDP model, we explore the effect of data distribution on aggregation functions performances.

6.4.1 How to characterize a data distribution

To investigate the impact of data distribution on performance of aggregation functions, we firstly study how to characterize a data distribution. The simplest way is to find an appropriate metric to describe the data distribution. For this purpose, we generate 16 random distributions with the same sampled data values from dataset $\mathcal{P}_s$. These 16 data distributions are illustrated in the Appendix D.

We list four different data distributions in the first column of Tab. 6.1, and also show the associated performance of aggregation functions in the second column (all the results can be seen in Appendix D). These performances are shown by $\omega \cdot \lambda$ [1] and are computed using our MDP model. We set the buffer size to 8 and vary the accuracy constraint from 0 to 190 (standard deviation of the sampled values from dataset $\mathcal{P}_s$).

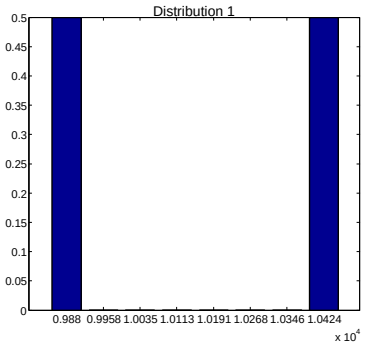
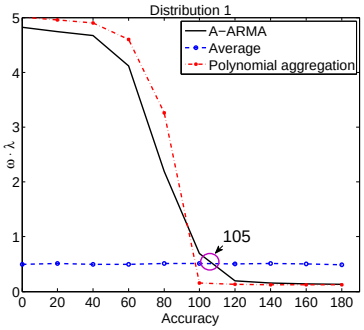
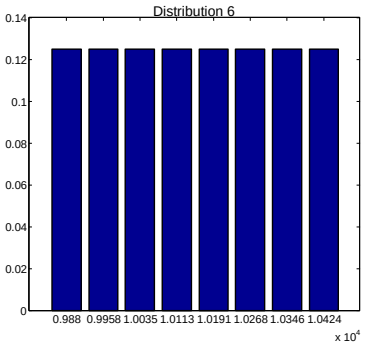
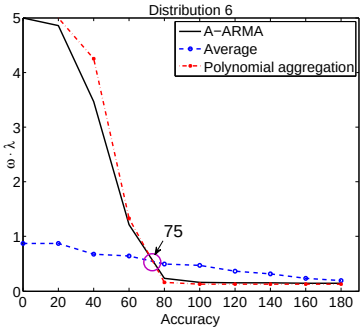
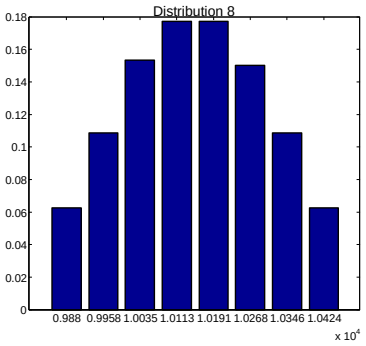
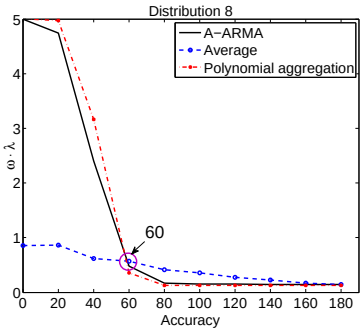
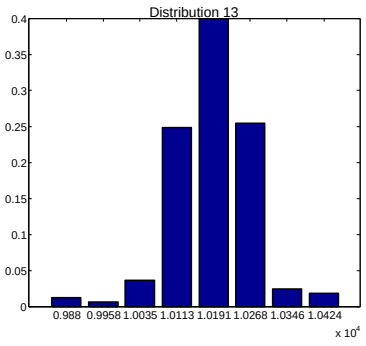
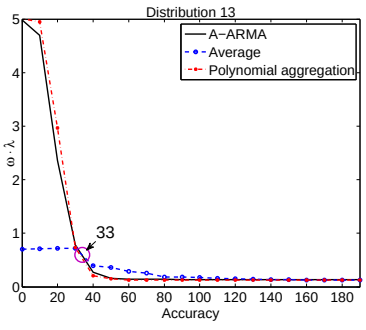
Better performances of aggregation functions are associated to smaller values of $\omega \cdot \lambda$. From Tab. 6.1, note that the performances of the three aggregation functions improve for larger values of accuracy. This is because when accuracy requirement is relaxed, the functions can aggregate more data into one packet. We can also see that for the three aggregation functions $\omega \cdot \lambda$ is a monotonic decreasing function of accuracy.

For strict constraints on accuracy, as seen in Tab. 6.1, average performs better than A-ARMA and polynomial aggregation. But above a certain accuracy threshold, A-ARMA and polynomial aggregation achieve better performances. This threshold is illustrated by purple circles in figures of second column of Tab. 6.1. We define the *crossed accuracy* as the accuracy threshold where A-ARMA performs better than average.

Data distributions that are shown in Tab. 6.1 are sorted in decreasing order of the *crossed accuracy*. From the *crossed accuracy* is around 105 of data distribution 1 to *crossed accuracy* is about 33 of data distribution 13. Similarly, the 16 data distributions in Appendix D are also sorted by the decreasing order of the *crossed accuracy*. The performances of aggregation functions (second column of Tab. 6.1) show that they are strongly related with the data distributions. In the following, we analyse several metrics, to characterize the data distributions that can capture the performances changes of the aggregation functions.

[1] We mentioned in Sec. 3.4 that $\omega \cdot \lambda$ can be used to express the performance of an aggregation function, the smaller value of $\omega \cdot \lambda$ is, the more energy and network capacity can be saved.

Table 6.1: Data distributions and performance of aggregation functions.

Data distribution	Function performance	Metrics
		$E(x)$ 10152 $V(x)$ 73984 $H(x)$ 1 d_{Euc} 0 d_{KL} 0 d_{Js} 0
		$E(x)$ 10151.88 $V(x)$ 31678.36 $H(x)$ 3 d_{Euc} 0.61 d_{KL} 2 d_{Js} 0.55
		$E(x)$ 10151.48 $V(x)$ 22074.76 $H(x)$ 2.91 d_{Euc} 0.72 d_{KL} 3 d_{Js} 0.72
		$E(x)$ 10188.36 $V(x)$ 6969.84 $H(x)$ 2.06 d_{Euc} 0.87 d_{KL} 7.13 d_{Js} 0.95

The first metric that we study is the Shannon entropy [124], which is a metric to measure the disorder or the uncertainty of random variables. The entropy of a discrete random variable X can be written as:

$$H(X) = -\sum_{i=1}^{n} P(i) \cdot log_2 P(i) \tag{6.8}$$

where n is the number of different values, and $P(i)$ is the associated probability of value i. Note that Entropy takes into account only the probability. The values of the random variable X does not appear in the expression.

In Fig. 6.12(a), we plot the *crossed accuracy* values (y axis on the right) associated to the 16 data distributions. As explained before, these distributions are sorted in decreasing order of the *crossed accuracy*, which is illustrated by the blue decreasing curve. In Fig. 6.12(a), we also plot in red the Entropy associated with the 16 data distributions. The quasi-concave shape of entropy does not follow the monotonic decreasing tendency of the *crossed accuracy*. Distribution 6 (uniform distribution) has the biggest entropy value 3, while *crossed accuracy* (75) is not the highest or lowest one. Entropy does not follow the *crossed accuracy* decreasing tendency because it does not integrate into its expression the values of the data, to which the accuracy is directly related. Therefore, we investigate others statistic metrics that are related with values. Precisely, the distribution moments.

We first start with moment of order one, namely: the expected value (mean value). In statistics, expected value measures central tendency of a probability distribution. Expected value of a random variable X can be calculated as:

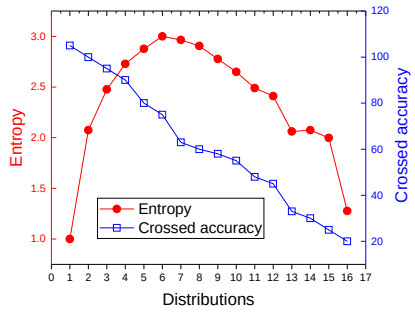
$$E(X) = \sum_{i=1}^{n} i \cdot P(i) \tag{6.9}$$

Similarly, we plot in Fig. 6.12(b) the expected value in red (y axis on the left) associated to these distributions. We can see that expected value is not monotonic function and it does not fit the decreasing tendency of the *crossed accuracy* curve.

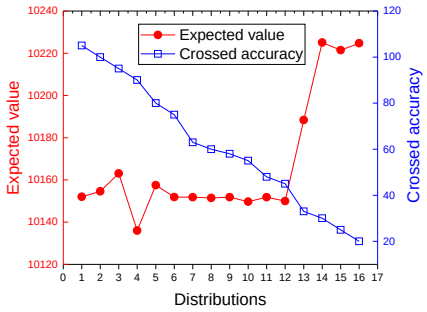
We then investigate moment of order two, namely: variance. Variance is used to measure how far a set of values is spread out. It can be formulated as:

$$V(X) = E(X^2) - [E(X)]^2 \tag{6.10}$$

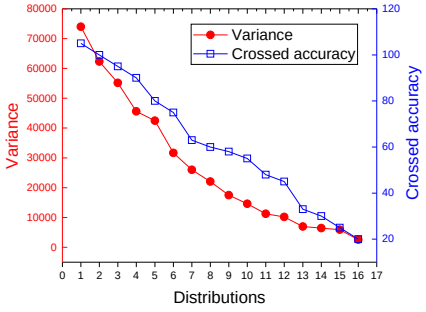
where $E(X)$ is the expected value of a random variable X. Variance is always non-negative: a small variance indicates that the data points tend to be very close to the expected value and hence to each other, while a high variance indicates that the data points are very spread out around the expected value and from each other. In



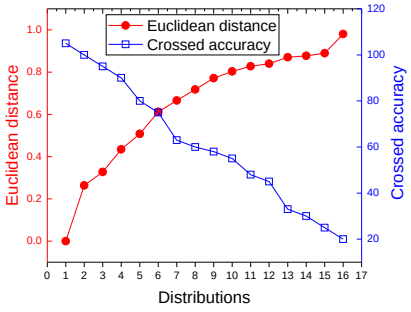
(a) Entropy changes with performance



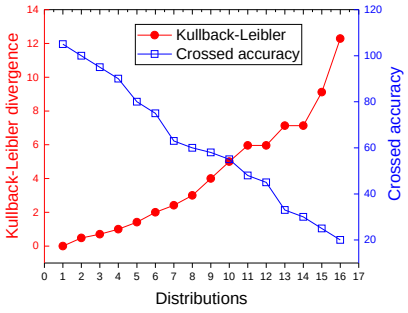
(b) Expected value changes with performance



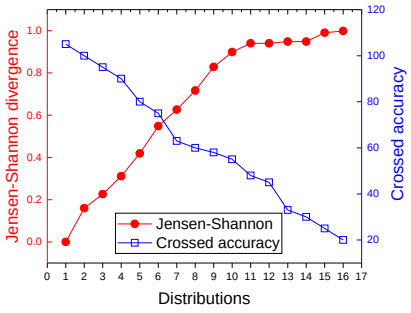
(c) Variance changes with performance



(d) Euclidean distance changes with performance



(e) Kullback-Leibler divergence changes with performance



(f) Jensen-Shannon divergence changes with performance

Figure 6.12: Metrics changes with performance of aggregation functions.

Fig. 6.12(c), we plot the variance associated with 16 data distribution. We can see that variance is a monotonic decreasing function, and its decreasing tendency follows quite well the *crossed accuracy* blue curve.

We have also investigated moments of order three and four (i.e. skewness and kurtosis), but the results that we obtained show that these metrics do not follow the shape

of *crossed accuracy* tendency. Thus among the distribution moments, variance seems to be the most adapted metric to capture the performances changes of the aggregation functions.

As illustrated in Tab. 6.1, the worst performances of A-ARMA and polynomial aggregation are obtained by data distribution 1, while the best performances are obtained by data distribution 16 (see Appendix D). Motivated by this observation, we investigate metrics that allow to compare data distribution to data distribution 1 (respectively to data distribution 16). There are two approaches to measure the difference between distribution P and distribution Q: considering the distribution as vector or as probability. Since each histogram is assumed to be independent from other histogram, a distribution can be considered as a vector. Thus, geometrical distances is used to compare two distributions. This is likely to measuring the overlapping between two data distributions as the distance. In terms of geometrical distances, we firstly study Euclidean distance, which is expressed as:

$$d_{Euc} = \sqrt{\sum_{i=1}^{n} |P(i) - Q(i)|^2} \tag{6.11}$$

In Fig. 6.12(d) we plot the Euclidean distance between data distribution 1 and other distributions. The curve in red show that it is a monotonic increasing function. Although the Euclidean distance and *crossed accuracy* are monotonic functions, they do not have the same slope. In particular, we can see from distribution 1 to 2, that the Euclidean distance has a sharp increase where *crossed accuracy* keeps slower decrease.

Euclidean distance treats data distributions as vectors, while there are several metrics using Shannon entropy, they consider data distribution as probability, like Kullback-Leibler divergence [112]:

$$d_{KL} = \sum_{i=1}^{n} P(i) log \frac{P(i)}{Q(i)} \tag{6.12}$$

which is a way to compute the relative entropy. Specifically, the Kullback–Leibler divergence of Q from P, is a measure of the information gained from the prior probability distribution P to the posterior probability distribution Q. More exactly, it is the amount of information that is lost when Q is used to approximate P. In Fig. 6.12(e), we plot Kullback-Leibler divergence and *crossed accuracy* in double y-axes. We can see that the Kullback-Leibler divergence is a monotonic increasing function. However from distribution 14 to 16, it has a sharp increase while *crossed accuracy* shows slower tendency.

Jensen-Shannon divergence [125] is based on the Kullback-Leibler divergence, with

notable differences, including a fact that it is always a finite value. It can be expressed as:

$$d_{Js} = \frac{1}{2}[\sum_{i=1}^{n} P(i)log\frac{2P(i)}{P(i)+Q(i)} + \sum_{i=1}^{n} Q(i)log\frac{2Q(i)}{P(i)+Q(i)}] \tag{6.13}$$

which is bounded by 1 (using the base 2 logarithm). Jensen-Shannon divergence is introduced as a criterion for the synthesis of random distributions. The results of Jensen-Shannon divergence are shown in Fig. 6.12(f). Again, this metric is monotonic function with increasing tendency. It captures the *crossed accuracy* change from distribution 1 to 12, but after that, it shows convergence tendency, while *crossed accuracy* keeps to decrease.

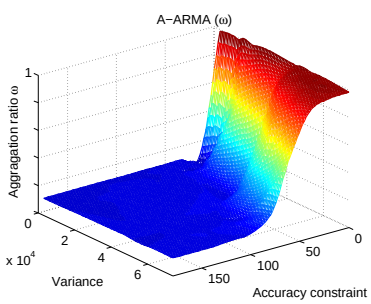
Since we list the *crossed accuracy* in descending order, the metric of Entropy and Expected value (Fig. 6.12(a)(b)) cannot be chosen due to their non-monotonic tendency. Although Euclidean distance, Kullback-Leibler divergence and Jensen-Shannon divergence are monotonic functions similar to *crossed accuracy*, they do not always capture the slope as variance does. Therefore, Variance appears to be the most appropriate metric to characterize data distribution.

6.4.2 Aggregation functions classification

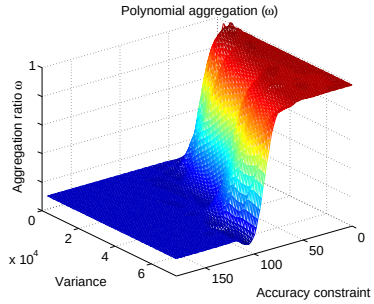
In Fig. 6.13, we plot the aggregation ratio ω for the three aggregation functions as a function of variance for the 16 data distributions associated to dataset $\mathcal{P}_s$ (the variance from 0 to 73984). Similarly, we plot in Fig. 6.14 the performances ($\omega \cdot \lambda$) for A-ARMA and polynomial aggregation. We omit the representation of average since packet size coefficient is always equals to 1.

For our networking-oriented metrics, better performance of aggregation functions are expressed by smaller value of them. Thus in Fig. 6.13 and Fig. 6.14 we see that when accuracy constraint relaxes or variance decreases, the aggregation functions performs better. It is reasonable because relaxing accuracy constraints means that more data can be aggregated together. Consequently, the number of aggregated packet is reduced and thus the value of aggregation ratio is minimized (Fig. 6.13). While when accuracy constraint is strict (approaching 0), only average can get benefits when the consecutive two data are same, A-ARMA and polynomial aggregation performs worse at this accuracy constraint. As accuracy constraint relaxes, A-ARMA and polynomial can aggregate more data using their predicting model. Take A-ARMA as an example, when it reaches the *crossed accuracy*, it performs better than Average.

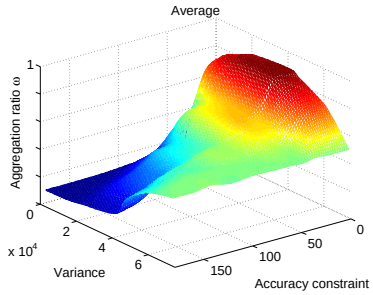
On the other hand, when variance is higher, which means that the data distribution is dispersed (like the distribution 1), average can keep around 50% aggregation



(a) A-ARMA

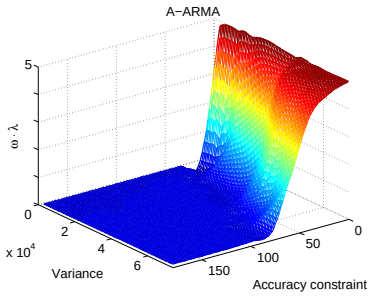


(b) Polynomial aggregation

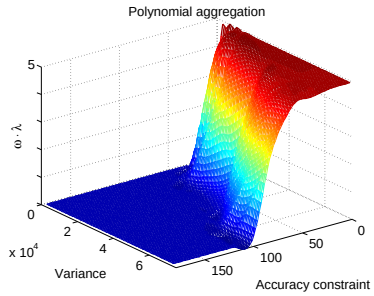


(c) Average

Figure 6.13: Aggregation ratio ω of aggregation functions, corresponds with variance and accuracy constraints.



(a) A-ARMA



(b) Polynomial aggregation

Figure 6.14: Performances of aggregation functions, correspond with variance and accuracy constraints.

ratio. As variance decreases, A-ARMA and polynomial aggregation gradually performs better than average.

We can also see in Fig. 6.13 and Fig. 6.14 that A-ARMA and polynomial aggregation show similar performances although they use different regression methods. A-ARMA performs better than polynomial when accuracy constraint is relatively smaller

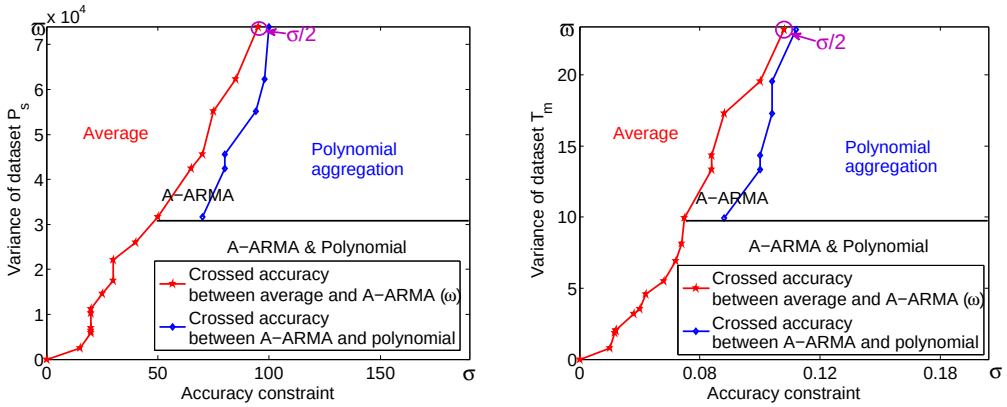
(accuracy constraint from 0 to 100 in Fig. 6.14(a)(b)). When accuracy constraint is bigger than 100, polynomial aggregation and A-ARMA show similar performances.

6.4.3 General recommendation

We want to propose a general guideline to allow researchers or engineer to choose an appropriate aggregation functions when data distribution and accuracy requirement are defined. Thus we intend to draw a map to indicate when an aggregation function performs better than the others for some data distributions and for a certain accuracy constraint. To classify the aggregation functions, we propose to identify the areas where each aggregation function performs better than the other ones, and the *crossed accuracy* is used to identify the border of each area. Firstly, we seek to identify the area for average. Note that the packet size coefficient λ for average is always equal to 1, thus we only consider to use the aggregation ratio ω to identify the area for average. The *crossed accuracy*s between A-ARMA and average on aggregation ratio for 16 data distributions are used to denote the area of average. Secondly, we need to identify the border between A-ARMA and polynomial aggregation. Considering the performance of functions shown in Tab. 6.1, we use the *crossed accuracy* between A-ARMA and polynomial aggregation to show the border of them. We can see in second column of Tab. 6.1 (all the complete results can be seen in Appendix D) that as the variances of data distributions decrease, A-ARMA and polynomial aggregation performs similar with each other, and their performance lines cross on the line of average. In such situation, we consider that A-ARMA and polynomial aggregation show similar performances, and their areas overlap.

In Fig. 6.15(a), we plot the variance as a function of crossed accuracy of average v.s. A-ARMA (red line) and A-ARMA v.s. polynomial aggregation (blue line) for the 16 data distributions using dataset $\mathcal{P}_s$. The black line denotes that the areas of A-ARMA and polynomial aggregation are overlapped, i.e. when variance is under this line, A-ARMA and polynomial aggregation performs similar. We also plot the same function in Fig. 6.15(b) for another dataset $\mathcal{T}_m$. For the two datasets, we vary the accuracy constraints from 0 to the standard deviation of sampled data (marked as σ). Their variances are calculated by Eq. 6.10 for different data distributions, and vary from 0 to maximum variance (denoted by ϖ).

As detailed in Appendix A and Appendix B, these two datasets have different data value and data property. Dataset $\mathcal{P}_s$ is a collection of sea level pressure, and $\mathcal{T}_m$ is a collection of indoor temperature. While comparing Fig. 6.15(a)(b), we see three same points: ①, the border between A-ARMA and average in these two figures are all around $\frac{\sigma}{2}$ (indicated by purple circle in these figures); ②, even though the border of



(a) Using dataset $\mathcal{P}_s$ (b) Using dataset $\mathcal{T}_m$

Figure 6.15: Classification of aggregation functions in real datasets.

average has different shape (red lines in the figures), the schematic range is the same, i.e. accuracy is from 0 to $\frac{\sigma}{2}$, and variance is from 0 to ϖ; ③, begin from data distribution 6 (uniform distribution in Appendix D), the performance lines of A-ARMA and polynomial aggregation cross on the line of average, i.e. starting from uniform distribution, these two functions perform similar.

That is to say, even though the data value, the accuracy constraints and variance are totally different, the classification lines between these function are quite similar. Motivated by this observation, we propose our general classification of aggregation functions.

We draw a schematic map to guild how to choose aggregation functions when the data distribution and accuracy constraint are presented. As shown in Fig. 6.16, x-axis denotes the accuracy from 0 to σ, and y-axis marks the variance from 0 to maximum variance ϖ. We plot a schematic border for average (red line in Fig. 6.16), where accuracy is from 0 to $\frac{\sigma}{2}$, and variance is from 0 to ϖ. When the variance is in ϖ, the accuracy border between A-ARMA and polynomial aggregation is a little bigger than $\frac{\sigma}{2}$. Similar as Fig. 6.15, the areas of A-ARMA and polynomial aggregation overlap when variance is relatively small. The areas in Fig. 6.16 illustrate the performance of aggregation functions, i.e. in certain area, the associated function save more energy and network capacity than others. From our general classification map, when variance is high (data distribution is dispersed), and accuracy constraints is relatively strict, it is better to use average to aggregate data. When accuracy requirement is a little bigger than $\frac{\sigma}{2}$, A-ARMA can achieve better performance; when variance is lower, A-ARMA or polynomial aggregation all save considerable energy and network capacity.

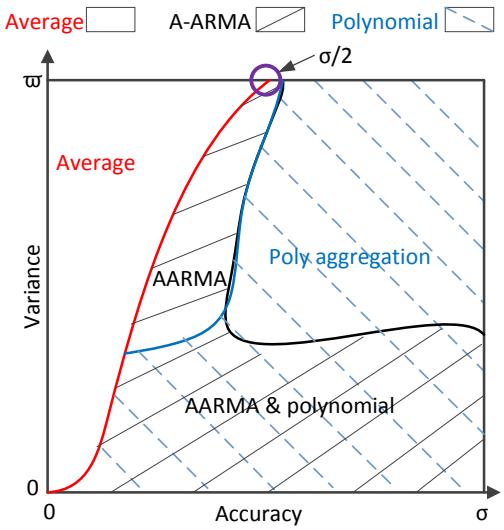


Figure 6.16: General classification of aggregation functions.

6.5 Conclusion

In this chapter, we propose a framework to classify forecasting aggregation functions. From perspectives of application and accuracy requirements, we evaluate the performances of the functions, and propose a general classification among them. The performance is evaluate by the networking-oriented metrics (aggregation ratio and packet size coefficient), and we use Markov decision processes to compute their optimal setting: when it is optimal to aggregate and when it is optimal to transmit in the viewpoint of aggregation ratio. We sample data distribution from real datasets and verify the accuracy of our model. To study the impact of data distribution on performances of aggregation functions , we compare several metrics, and finally select variance to characterize the different data distribution. We test our classification method on different datasets and get similar results. We finally propose a general classification of aggregation functions, which depends only on the variance of the data distribution and the target accuracy. Such classification is able to show the optimal aggregation function according to the data distribution and accuracy requirement.

7

Conclusion and future perspective

Contents

In this Chapter, we conclude this manuscript with a summary of our contributions and discuss the directions for future researches.

7.1 Conclusions

In wireless sensor networks, energy-saving is a key issue for all the researchers. Many MAC and routing protocols are proposed to saving energy by duty-cycle mechanism or transferring packets through a shorter path. However, MAC and routing protocols only focus on how to effectively transfer a packet, not on whether this packet is needed to the sink or not. The fundamental way to save energy is reducing number of packets. Data aggregation is a data gathering processing which can reduce redundant and/or correlated packets. Under a data aggregation mechanism, nodes can process the correlated information into a digest, and only sends this digest to sink instead of raw data. In this manuscript, considering the limitations of state-of-the-art works, we propose data-independent aggregation an property-independent aggregation functions. In the

meanwhile, we evaluate performances of aggregation functions and classify them.

Networking-oriented metrics are proposed in Chapter 3 to evaluate the impact of aggregation functions at networking level. We claim that the objectives of data aggregation are saving energy and network capacity, correspondingly, we propose two new metrics: *aggregation ratio* and *packet size coefficient*. Aggregation ratio can denote the ability of saving energy, and packet size coefficient is used to describe the network capacity with data aggregation.

In Chapter 4, a new data-independent aggregation mechanism is proposed, Similar-evolution Based data Aggregation (denoted as Simba). Many aggregation mechanisms are based on raw data, while from observation of real datasets, we highlight that some nodes show similar evolution instead of similar data. Based on data evolution, Simba groups the nodes having similar evolution together, and makes the nodes in the same group to execute aggregation functions alternatively. The design of Simba avoids the sensitivity of raw data, guarantees the recover fidelity, and also saves more energy.

Without considering the data property, we propose an aggregation function: Agnostic Aggregation (A^2) in Chapter 5. A^2 can go against non-anticipated data variations of dynamic scenarios. More specifically, A^2 allows sensor nodes to dynamically adapt the aggregation function to fix the latest data. As far as we know, A^2 is the first aggregation function which is able to self-adapt to the applications.

A classification framework for forecasting aggregation functions is provided in Chapter 6, this framework evaluate aggregation functions from three perspectives: performance, data property and accuracy constraints. Firstly, we build a stochastic model (MDP model) to compute networking-oriented metrics for different aggregation functions; secondly, we study the impact of different datasets on performance of aggregation functions; and then, we select an appropriate metric (i.e. variance) to characterize the data distribution; finally, we classify the functions according to data distribution and the target accuracy.

7.2 Future perspectives

This thesis focuses on aggregation function in wireless sensor networks, which can be extended to some areas:

1 **Temporal-spatial data aggregation.** In this manuscript, we put our main concern on forecasting data aggregation functions, which are based on temporal correlation. Recently, more and more works show that data aggregation can achieve better performance by combining temporal and spatial correlations [126].

Thus as an extension, we will further our investigate on temporal-spatial data aggregation. Considering the spatial locations, neighboring nodes will send data to a pre-defined leader, and as time goes, this leader will aggregate temporal-spatial data. While in our current work (in chapter 6), we have a limitation on buffer size (8 data), such condition is a strong limitation for spatial data. It means, for example, there are only 8 nodes can share one leader or 2 nodes can only send four continues data to the same leader. Thus, how to design better temporal-spatial aggregation function, and how to evaluate the performance of temporal-spatial data aggregation functions are also challenges for us.

2 **Packet loss in data aggregation**. In order to achieve data aggregation, we all assume that either the raw data or aggregated data can be transmitted successfully to the sink. However, in real wireless communication, packet loss is a common phenomenon. Investigating the works that related to data aggregation, few works consider such situation. If an aggregated packet is lost, how a sensor node deal with: whether re-transmit until success or just drop the packet. According to the viewpoints of energy and network capacity, dropping is a good choice; while from the view of data aggregation, sink loses many information and may not recover data accurately. Thus as another extension, we are motivated to study the packet loss in data aggregation, and to investigate the impact on energy and accuracy.

3 **Secure data aggregation.** Due to hostile environments and unique properties of wireless sensor networks, it is a challenging task to protect sensitive information transmitted by wireless sensor networks. In addition, wireless sensor networks have security problems that traditional networks do not face. Therefore, security is an important issue for wireless sensor networks and there are many security considerations that should be investigated. As the majority of wireless sensor network applications require a certain level of security, it is not possible to sacrifice security for data aggregation. In addition, there is a strong conflict between security and data aggregation protocols. Security protocols require sensor nodes to encrypt and authenticate any sensed data prior to its transmission and prefer data to be decrypted by the base station[127, 128]. On the other hand, data aggregation protocols prefer raw data to implement data aggregation at every intermediate node so that energy efficiency is maximized. Moreover, data aggregation results in alterations in sensor data and therefore it is a challenging task to provide source and data authentication along with data aggregation.

4 **Data aggregation in VANETs.** Vehicular Ad hoc Networks (VANETs) have been regarded as an emerging and promising field in both industry and academia. It has potential to improve the efficiency and safety of future highway systems. Different with WSNs, VANETs have no significant power constraints, unlike sensor networks where limited battery life is a major concern; but bandwidth utilization is an important issue for VANET communications. Thus data aggregation needs to be discuss under VANETs. One challenge in data aggregation of VANETs is to ensure that reports from different sources can be delivered to the same node at the same time so that they can be merged together. Unlike sensor networks, there are aggregation structure can be researched, rapid changes of topologies of VANETs can not use any of the structured aggregation protocols. Recently, several VANET projects use periodical broadcasting for data aggregation[129, 130], but content exchange based on periodical broadcasting may not be an efficient way in terms of communication overhead, and what is an optimal broadcast interval is still an unsolved issue.

Bibliography

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

[2] I. F. Akyildiz and M. C. Vuran, *Wireless sensor networks*. John Wiley & Sons, 2010, vol. 4.

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

[4] P. Jesus, C. Baquero, and P. Almeida, "A survey of distributed data aggregation algorithms," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 381–404, 2015.

[5] L. Yong-Min, W. Shu-Ci, and N. Xiao-Hong, "The architecture and characteristics of wireless sensor network," in *IEEE International Conference on Computer Technology and Development*, Kota Kinabalu, Malaysia, Nov 2009.

[6] M. Z. A. Bhuiyan, G. Wang, J. Cao, and J. Wu, "Deploying wireless sensor networks with fault-tolerance for structural health monitoring," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 382–395, 2015.

[7] T. J. Dishongh, M. McGrath, and B. Kuris, *Wireless sensor networks for healthcare applications*. Artech House, 2014.

[8] S. Li, J. Cui, and Z. Li, "Wireless sensor network for precise agriculture monitoring," in *IEEE International Conference on Intelligent Computation Technology and Automation*, Shenzhen, China, Mar 2011.

[9] M. H. Anisi, G. Abdul-Salaam, and A. H. Abdullah, "A survey of wireless sensor network approaches and their energy consumption for monitoring farm fields in precision agriculture," *Precision Agriculture*, vol. 16, no. 2, pp. 216–238, 2015.

[10] W. Na, N.-N. Dao, and S. Cho, "Reliable multicasting service for densely deployed military sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2015, 2015.

[11] L. M. Oliveira and J. J. Rodrigues, "Wireless sensor networks: a survey on environmental monitoring," *Journal of communications*, vol. 6, no. 2, pp. 143–151, 2011.

[12] J. Paek, J. Hicks, S. Coe, and R. Govindan, "Image-based environmental monitoring sensor application using an embedded wireless sensor network," *MDPI Sensors*, vol. 14, no. 9, pp. 15 981–16 002, 2014.

[13] M. V. Ramesh, "Real-time wireless sensor network for landslide detection," in *IARIA SENSORCOMM*, Athens/Glyfada, Greece, June 2009.

[14] R. Tan, G. Xing, J. Chen, W.-Z. Song, and R. Huang, "Quality-driven volcanic earthquake detection using wireless sensor networks," in *IEEE RTSS*, San Diego, USA, Nov 2010.

[15] S. K. Roy, A. Roy, S. Misra, N. S. Raghuwanshi, and M. S. Obaidat, "Aid: A prototype for agricultural intrusion detection using wireless sensor network," in *IEEE ICC*, Cape Town, South Africa, 2015.

[16] V. Potdar, A. Sharif, and E. Chang, "Wireless sensor networks: A survey," in *IEEE International Conference on Advanced Information Networking and Applications Workshops*, Bradford, United Kingdom, May 2009.

[17] A. Tripathi, S. Gupta, and B. Chourasiya, "Survey on data aggregation techniques for wireless sensor networks," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 3, no. 7, pp. 7366–7371, 2014.

[18] E. Hamida, P. Borgnat, H. Esaki, P. Abry, E. Fleury *et al.*, "Live e! sensor network: Correlations in time and space," in *XXIIe Colloque GRETSI-Traitement du Signal et des Images*, Dijon, France, Sep 2009.

[19] M. C. Vuran, Ö. B. Akan, and I. F. Akyildiz, "Spatio-temporal correlation: theory and applications for wireless sensor networks," *Elsevier Computer Networks*, vol. 45, no. 3, pp. 245–259, 2004.

[20] L. A. Villas, A. Boukerche, H. A. De Oliveira, R. B. De Araujo, and A. A. Loureiro, "A spatial correlation aware algorithm to perform efficient data collection in wireless sensor networks," *Elsevier Ad Hoc Networks*, vol. 12, pp. 69–85, 2014.

[21] B. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *IEEE International Conference on Distributed Computing Systems Workshops*, Vienna, Austria, Jul 2002.

[22] K. Maraiya, K. Kant, and N. Gupta, "Wireless sensor network: a review on data aggregation," *International Journal of Scientific & Engineering Research*, vol. 2, no. 4, pp. 1–6, 2011.

[23] J. Cui and F. Valois, "Data aggregation in wireless sensor networks: Compressing or forecasting?" in *IEEE WCNC*, Istanbul, Turkey, Apr. 2014.

[24] H. Li, C. Wu, Q.-S. Hua, and F. C. Lau, "Latency-minimizing data aggregation in wireless sensor networks under physical interference model," *Ad Hoc Networks*, vol. 12, pp. 52–68, 2014.

[25] M. Kumar and K. Dutta, "A survey of security concerns in various data aggregation techniques in wireless sensor networks," in *Intelligent Computing, Communication and Devices*. Springer, 2015, vol. 309, no. 1, pp. 1–15.

[26] T. Jung, X. Mao, X.-Y. Li, S.-J. Tang, W. Gong, and L. Zhang, "Privacy-preserving data aggregation without secure channel: Multivariate polynomial evaluation," in *IEEE INFOCOM*. Turin, Italy: IEEE, Apr 2013.

[27] Z. Erkin, J. R. Troncoso-Pastoriza, R. L. Lagendijk, and F. Perez-Gonzalez, "Privacy-preserving data aggregation in smart metering systems: An overview," *IEEE Signal Processing Magazine*, vol. 30, no. 2, pp. 75–86, 2013.

[28] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2–16, 2003.

[29] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[30] J. Lu, F. Valois, M. Dohler, and M.-Y. Wu, "Optimized data aggregation in wsns using adaptive arma," in *IARIA SENSORCOMM*, Venice,Italy, July 2010.

[31] E. Ben Hamida, H. Ochiai, H. Esaki, P. Borgnat, P. Abry, and E. Fleury, "Measurement analysis of the live e! sensor network: Spatial-temporal correlations and data aggregation," in *IEEE/IPSJ SAINT*, Seattle, USA, July 2009.

[32] C. Liu, K. Wu, and M. Tsao, "Energy efficient information collection with the arima model in wireless sensor networks," in *IEEE GLOBECOM*, St. Louis, MO, Nov 2005.

[33] J. Wang, S. Tang, B. Yin, and X.-Y. Li, "Data gathering in wireless sensor networks through intelligent compressive sensing," in *IEEE INFOCOM*, Orlando, Florida USA, March 2012.

[34] M. Bagaa, Y. Challal, A. Ksentini, A. Derhab, and N. Badache, "Data aggregation scheduling algorithms in wireless sensor networks: Solutions and challenges," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1339–1368, 2014.

[35] Z. Ye, A. Abouzeid, and J. Ai, "Optimal stochastic policies for distributed data aggregation in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 5, pp. 1494–1507, Oct 2009.

[36] B.-H. Liu and J.-Y. Jhang, "Efficient distributed data scheduling algorithm for data aggregation in wireless sensor networks," *Elsevier Computer Networks*, vol. 65, pp. 73–83, 2014.

[37] M. Bagaa, M. Younis, D. Djenouri, A. Derhab, and N. Badache, "Distributed low-latency data aggregation scheduling in wireless sensor networks," *ACM Transaction on Sensor Networks*, vol. 11, no. 3, pp. 49:1–49:36, Apr. 2015.

[38] G. Prabhu and K. Dhamotharan, "A survey on secure data aggregation scheme for wireless sensor networks," *International Journal of Science, Engineering and Technology Research*, vol. 3, no. 2, pp. 329–335, 2014.

[39] N. Kumar, N. Chilamkurti, and J. J. Rodrigues, "Learning automata-based opportunistic data aggregation and forwarding scheme for alert generation in vehicular ad hoc networks," *Computer Communications*, vol. 39, pp. 22–32, 2014.

[40] 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, 2007.

[41] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: A tiny aggregation service for ad-hoc sensor networks," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 131–146, 2002.

[42] N.-C. Wang, Y.-F. Huang, J.-S. Chen, and P.-C. Yeh, "Energy-aware data aggregation for grid-based wireless sensor networks with a mobile sink," *Wireless Personal Communications*, vol. 43, no. 4, pp. 1539–1551, 2007.

[43] T.-W. Kuo and M.-J. Tsai, "On the construction of data aggregation tree with minimum energy cost in wireless sensor networks: Np-completeness and approximation algorithms," in *IEEE INFOCOM*. Orlando, USA: IEEE, Mar 2012.

[44] M. Shan, G. Chen, D. Luo, X. Zhu, and X. Wu, "Building maximum lifetime shortest path data aggregation trees in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 11, no. 1, pp. 11:1–11:24, Jul. 2014.

[45] Y. Xun-Xin and Z. Rui-Hua, "An energy-efficient mobile sink routing algorithm for wireless sensor networks," in *IEEE WiCOM*. Wuhan, China: IEEE, Sep 2011.

[46] 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.

[47] 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.

[48] A. Sinha and D. K. Lobiyal, "Performance evaluation of data aggregation for cluster-based wireless sensor network," *Human-Centric Computing and Information Sciences*, vol. 3, no. 1, pp. 1–17, 2013.

[49] J.-H. Shin, J. Kim, K. Park, and D. Park, "Railroad: virtual infrastructure for data dissemination in wireless sensor networks," in *ACM PE-WASUN*, Montreal, Canada, October 2005.

[50] C. Tunca, S. Isik, M. Y. Donmez, and C. Ersoy, "Ring routing: An energy-efficient routing protocol for wireless sensor networks with a mobile sink," *IEEE Transactions on Mobile Computing*, vol. 14, no. 9, pp. 1947–1960, 2015.

[51] J.-L. Lu and F. Valois, "On the data dissemination in wsns," in *IEEE WiMOB*, NY, USA, October 2007.

[52] X. Xu, W. Liang, and T. Wark, "Data quality maximization in sensor networks with a mobile sink," in *IEEE DCOSS*, Barcelona, Spain, June 2011.

[53] I. Solis and K. Obraczka, "Isolines: efficient spatio-temporal data aggregation in sensor networks," *Wireless Communications and Mobile Computing*, vol. 9, no. 3, pp. 357–367, 2009.

[54] R. Guocan and D. Guowei, "An improved isoline based data aggregation scheme in wireless sensor networks," *Elsevier Procedia Engineering*, vol. 23, pp. 326–332, 2011.

[55] A. Khelil, "A comparative effectiveness evaluation of map construction protocols in wireless sensor networks," *IEEE Systems Journal*, vol. 8, no. 3, pp. 708–716, Sept 2014.

[56] K.-W. Fan, S. Liu, and P. Sinha, "Structure-free data aggregation in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 8, pp. 929–942, 2007.

[57] H. Yousefi, M. H. Yeganeh, N. Alinaghipour, and A. Movaghar, "Structure-free real-time data aggregation in wireless sensor networks," *Elsevier Computer Communications*, vol. 35, no. 9, pp. 1132–1140, 2012.

[58] C.-M. Chao and T.-Y. Hsiao, "Design of structure-free and energy-balanced data aggregation in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 37, pp. 229–239, 2014.

[59] K.-W. Fan, S. Liu, and P. Sinha, "On the potential of structure-free data aggregation in sensor networks." in *IEEE INFOCOM*, Barcelona, Spain, Mar 2006.

[60] J. D. Hamilton, *Time series analysis*. Cambridge Univ Press, 1994, vol. 2.

[61] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in *Selected Papers of Hirotugu Akaike*. Springer, 1998, pp. 199–213.

[62] G. Schwarz, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[63] G. P. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, 2003.

[64] K. Sayood, *Introduction to data compression*. Newnes, 2012.

[65] M. A. Razzaque, C. Bleakley, and S. Dobson, "Compression in wireless sensor networks: A survey and comparative evaluation," *ACM Transaction on Sensor and Networks*, vol. 10, no. 1, pp. 5:1–5:44, 2013.

[66] N. Kimura and S. Latifi, "A survey on data compression in wireless sensor networks," in *IEEE International Conference on Information Technology: Coding and Computing*, Las Vegas, USA, Apr 2005.

[67] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, and K. Araki, "Practical data compression in wireless sensor networks: A survey," *Journal of Network and Computer Applications*, vol. 35, no. 1, pp. 37–59, 2012.

[68] D. Petrovic, R. C. Shah, K. Ramchandran, and J. Rabaey, "Data funneling: Routing with aggregation and compression for wireless sensor networks," in *IEEE International Workshop on Sensor Network Protocols and Applications*, Anchorage, USA, May 2003.

[69] T. Arici, B. Gedik, Y. Altunbasak, and L. Liu, "Pinco: A pipelined in-network compression scheme for data collection in wireless sensor networks," in *IEEE ICCCN*, Dallas, USA, Oct 2003.

[70] E. Candes and M. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[71] R. G. Baraniuk, "Compressive sensing," *IEEE signal processing magazine*, vol. 24, no. 4, pp. 118–124, 2007.

[72] J. Haupt, W. Bajwa, M. Rabbat, and R. Nowak, "Compressed sensing for networked data," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 92 –101, 2008.

[73] E. J. Candes and J. K. Romberg, "Signal recovery from random projections," in *Electronic Imaging*, vol. 5674, 2005, pp. 76–86.

[74] C. T. Chou, R. Rana, and W. Hu, "Energy efficient information collection in wireless sensor networks using adaptive compressive sensing," in *IEEE LCN*, Swissôtel, Zürich, Oct 2009.

[75] M. Laifenfeld and I. Bilik, "Distributed compressive sensing and communications in wireless sensor networks," in *IEEE Convention of Electrical Electronics Engineers in Israel*, Eilat, Israel, Nov 2012.

[76] C. Luo, F. Wu, J. Sun, and C. Chen, "Compressive data gathering for large-scale wireless sensor networks," in *ACM MobiCom*, Beijing, China, Sep 2009.

[77] W. Chen and I. Wassell, "Energy-efficient signal acquisition in wireless sensor networks: a compressive sensing framework," *IET Wireless Sensor Systems*, vol. 2, no. 1, pp. 1–8, 2012.

[78] G. Quer, D. Zordan, R. Masiero, M. Zorzi, and M. Rossi, "Wsn-control: Signal reconstruction through compressive sensing in wireless sensor networks," in *IEEE LCN*, Denver, Colorado, U.S.A., Oct 2010.

[79] R. Masiero, G. Quer, D. Munaretto, M. Rossi, J. Widmer, and M. Zorzi, "Data acquisition through joint compressive sensing and principal component analysis," in *IEEE GLOBECOM*, Honolulu, USA, Nov 2009.

[80] L. Xiang, J. Luo, and C. Rosenberg, "Compressed data aggregation: Energy-efficient and high-fidelity data collection," *IEEE/ACM Transactions on Networking*, vol. 21, no. 6, Dec 2013.

[81] J. Luo, L. Xiang, and C. Rosenberg, "Does compressed sensing improve the throughput of wireless sensor networks?" in *IEEE ICC*, Cape Town, South Africa, may 2010.

[82] H. Zheng, S. Xiao, X. Wang, and X. Tian, "On the capacity and delay of data gathering with compressive sensing in wireless sensor networks," in *IEEE GLOBECOM*, Houston, USA, Dec 2011.

[83] N. Pantazis, S. A. Nikolidakis, and D. D. Vergados, "Energy-efficient routing protocols in wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 551–591, 2013.

[84] A. Bachir, M. Dohler, T. Watteyne, and K. K. Leung, "Mac essentials for wireless sensor networks," *IEEE Communications Surveys Tutorials*, vol. 12, no. 2, pp. 222–248, 2010.

[85] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Elsevier Ad hoc networks*, vol. 3, no. 3, pp. 325–349, 2005.

[86] N. Pantazis, S. A. Nikolidakis, and D. D. Vergados, "Energy-efficient routing protocols in wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 551–591, 2013.

[87] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*. Wiley, 2007.

[88] R. D. Komguem, I. Amadou, G. Chelius, and F. Valois, "Routing protocols: When to use it in terms of energy?" in *IEEE WCNC*, Paris, France, April 2012.

[89] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for ad hoc networks," in *IEEE INMIC*, Pakistan, Dec 2001.

[90] S. Mahfoudh and P. Minet, "Energy-aware routing in wireless ad hoc and sensor networks," in *ACM IWCMC*, Caen, France, Jun 2010.

[91] B. Karp and H.-T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *ACM MobiCom*, Boston,USA, Aug 2000.

[92] F. Ye, G. Zhong, S. Lu, and L. Zhang, "Gradient broadcast: A robust data delivery protocol for large scale sensor networks," *Wireless Networks*, vol. 11, no. 3, pp. 285–298, 2005.

[93] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 1, pp. 14–25, 2002.

[94] I. Mabrouki, X. Lagrange, and G. Froc, "Random walk based routing protocol for wireless sensor networks," in *ICST Proceedings of international conference on Performance evaluation methodologies and tools*, Nante, France, Oct 2007.

[95] P. Huang, L. Xiao, S. Soltani, M. W. Mutka, and N. Xi, "The evolution of mac protocols in wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 101–120, 2013.

[96] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *ACM SenSys*, Baltimore, MD, USA, 2004.

[97] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks," in *ACM SenSys*, Boulder, Colorado, USA, 2006.

[98] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, 2004.

[99] H. Li, V. Pandit, A. Knox, and D. P. Agrawal, "A novel characteristic correlation approach for aggregating data in wireless sensor networks," in *IEEE WoWMoM*, Madrid, Spain, Jun 2013.

[100] S. Imon, A. Khan, and S. Das, "Effect: An energy efficient framework for data compression in tree-based wireless sensor networks," in *IEEE WoWMoM*, Sydney, Australia, June 2014.

[101] tropical atomosphere ocean project, "Temperature data." [Online]. Available: http://www.pmel.noaa.gov/tao/data_deliv/deliv.html

[102] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis*. John Wiley & Sons, 2012.

[103] G. C. Zeitler, A. C. Singer, and S. S. Kozat, "Universal piecewise linear regression of individual sequences: Lower bound," in *IEEE ICASSP*, Honolulu, USA, April 2007.

[104] A. Madylova and S. G. Oguducu, "A taxonomy based semantic similarity of documents using the cosine measure," in *IEEE ISCIS*, North Cyprus, Sept 2009.

[105] F. Yang and I. Augé-Blum, "Constructing virtual coordinate for routing in wireless sensor networks under unreliable links," in *ACM IWCMC*, Leipzig, Germany, Jun 2009.

[106] Y.-A. Le Borgne, S. Santini, and G. Bontempi, "Adaptive model selection for time series prediction in wireless sensor networks," *Signal Processing*, vol. 87, no. 12, pp. 3010–3020, 2007.

[107] B. Choi, *ARMA model identification*. Springer Science & Business Media, 2012.

[108] W. Penny, "Comparing dynamic causal models using aic, bic and free energy," *Elsevier Neuroimage*, vol. 59, no. 1, pp. 319–330, 2012.

[109] G. Claeskens, N. L. Hjort *et al.*, *Model selection and model averaging*. Cambridge University Press Cambridge, 2008.

[110] H. Akaike, "Factor analysis and AIC," *Psychometrika*, vol. 52, no. 3, pp. 317–332, 1987.

[111] R. U. Ayres, "Information, entropy, and progress: a new evolutionary paradigm," *New York: American Institute of Physics*, vol. 1, 1994.

[112] S. Kullback, "Letter to the editor: the kullback-leibler distance," *American Statistician*, vol. 41, no. 4, pp. 340–341, 1987.

[113] K. P. Burnham and D. R. Anderson, *Model selection and multimodel inference: a practical information-theoretic approach*. Springer Science & Business Media, 2003.

[114] H. Bhat and N. Kumar, "On the derivation of the bayesian information criterion," *School of Natural Sciences, University of California*, 2010.

[115] T. He, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Aida: Adaptive application-independent data aggregation in wireless sensor networks," *ACM Transactions on Embedded Computing Systems*, vol. 3, no. 2, pp. 426–457, 2004.

[116] R. Bellman, "A markovian decision process," *Indiana University Mathematics Journal*, vol. 6, pp. 679–684, 1957.

[117] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2009.

[118] A. M. Colman, *Game theory and experimental games: The study of strategic interaction*. Elsevier, 2014.

[119] J. Lin, N. Xiong, A. V. Vasilakos, G. Chen, and W. Guo, "Evolutionary game-based data aggregation model for wireless sensor networks," *IET Communications*, vol. 5, no. 12, pp. 1691–1697, 2011.

[120] J. R. Norris, *Markov chains*. Cambridge university press, 1998.

[121] W. J. Stewart, *Introduction to the numerical solution of Markov chains*. Princeton University Press Princeton, 1994.

[122] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, *Queueing networks and markov chains*. John Wiley & Sons, Inc, 2000.

[123] H. J. Kushner, *Introduction to stochastic control*. Holt, Rinehart and Winston New York, 1971.

[124] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[125] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 145–151, 1991.

[126] C. Liu and G. Cao, "Spatial-temporal coverage optimization in wireless sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 10, no. 4, pp. 465–478, 2011.

[127] C.-X. Liu, Y. Liu, Z.-J. Zhang, and Z.-Y. Cheng, "High energy-efficient and privacy-preserving secure data aggregation for wireless sensor networks," *International Journal of Communication Systems*, vol. 26, no. 3, pp. 380–394, 2013.

[128] S. Roy, M. Conti, S. Setia, and S. Jajodia, "Secure data aggregation in wireless sensor networks: Filtering out the attacker's impact," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 4, pp. 681–694, 2014.

[129] B. Yu, C.-Z. Xu, and M. Guo, "Adaptive forwarding delay control for vanet data aggregation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 1, pp. 11–18, 2012.

[130] J. Jiru, L. Bremer, and K. Graffi, "Data aggregation in vanets a generalized framework for channel load adaptive schemes," in *IEEE LCN*. Edmonton, Canada: IEEE, Mar 2014.

[131] R. Bellman, "Dynamic programming princeton university press," *Princeton, NJ*, 1957.

[132] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

List of publications

International Conference Papers

[1] **J. Cui** and F. Valois, "Simba: Similar-evolution Based Aggregation in Wireless Sensor Networks", *IEEE/IFTP Wireless Days*, Toulouse, France, Mar. 2016.

[2] **J. Cui**, O. Lalami, J. Lu and F. Valois, "A^2: Agnostic Aggregation in Wireless Sensor Networks", *IEEE CCNC*, Las Vegas, USA, Jan. 2016.

[3] **J. Cui**, K. Boussetta and F. Valois, "Performance Evaluation of Data Aggregation Functions using Markov Decision Processes", *ACM PEWASUN*, Cancun, Mexico, Nov. 2015.

[4] **J. Cui**, and F. Valois, "Data aggregation in Wireless Sensor Networks: Compressing or Forecasting?" *IEEE WCNC*, Istanbul, Turkey, Apr. 2014.

Research Reports

[5] **J. Cui**, and F. Valois, "Data aggregation in Wireless Sensor Networks: Compressing or Forecasting?" *INRIA Research Report, RR-8362*, Sep. 2013.

Appendices

A

Dataset–$\mathcal{T}_o$ and $\mathcal{P}_s$

Dataset $\mathcal{T}_o$ is a collection of temperature from ocean surface in Tropical Atmosphere Ocean Project, and dataset $\mathcal{P}_s$ is data of sea level pressure from the same project. This project monitors real-time data from ocean buoys for improved detection, and these data are used to understand and predict $EINiño$ and $LaNiña$. The project array consists of approximately 70 moorings in the Tropical Pacific Ocean, telemetering oceanographic and meteorological data to shore in real-time via the Argos satellite system. The array is a major component of the $EINiño$/Southern Oscillation (ENSO) Observing System, the Global Climate Observing System (GCOS) and the Global Ocean Observing System (GOOS). Support is provided primarily by the United States (National Oceanic and Atmospheric Administration) and Japan (Japan Agency for Marine-earth Science and Technology).

The original data includes sea surface temperature, sea level pressure, shortwave radiation etc. Dataset $\mathcal{T}_o$ uses data of sea surface temperature, and dataset $\mathcal{P}_s$ uses data of sea level pressure of one node. The details of the project can be found in [101], and all of the data can be downloaded from this website.

Here, we show data example of dataset $\mathcal{T}_o$ in table A.1. In this dataset, each node has 167 data, data range from 19.76^0C to 30.58^0C, there are totally 584 different values. We plot the whole data in Fig. A.1.

In terms of dataset $\mathcal{P}_s$, we show the data example in table A.2. In this dataset, the node has 11700 data, data range from 9803 to 10424, there are totally 489 different

Table A.1: Data example of dataset $\mathcal{T}_o$

Node	110W	125W	140W	155W	170W	180W
Temperature [^0C]	27.74	27.91	28.07	28.12	28.44	28.92
	26.83	27.45	28.09	28.79	28.86	29.51
	23.62	24.69	25.78	28.11	29.05	29.96
	...	...	...	...	...	...

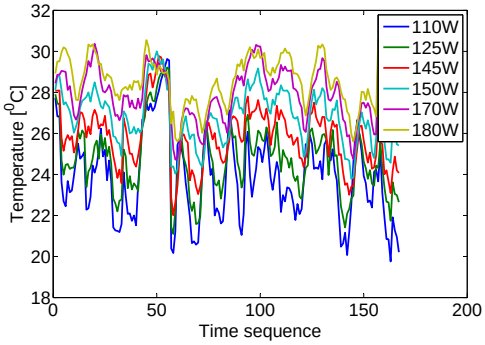


Figure A.1: Dataset $\mathcal{T}_o$.

values. The whole dataset can be seen in Fig A.2.

Table A.2: Data example of dataset $\mathcal{P}_s$

Sea Level Pressure [Ph]	10169
	10278
	10136
	9969
	...

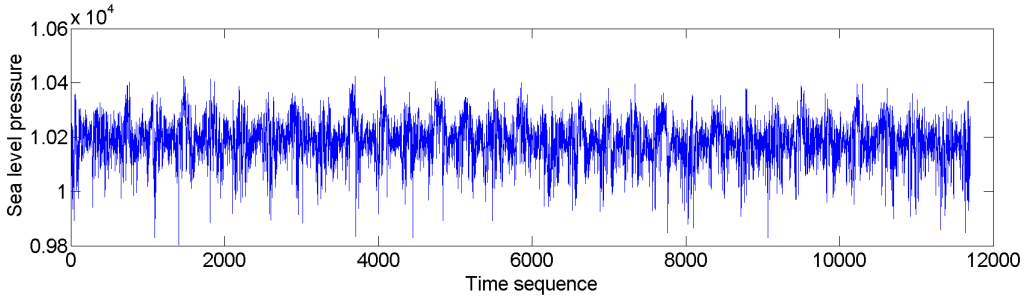


Figure A.2: Dataset $\mathcal{P}_s$.

B

Dataset–$\mathcal{T}_h$ and $\mathcal{T}_m$

Dataset $\mathcal{T}_h$ comes from a custom project, INSA BQR Arbre project. This project is interested in scientific, technical and societal instrumentation of buildings by networks of sensors/actuators. By extension, we can consider the instrumentation of the city from the perspective of a continuum physical world - digital world, a scan of the living space. This project places temperature sensor nodes in an apartment, to monitor the evolution of temperatures during the day or over longer periods long and comparing the differences obtained by the coins. The sensors are placed on legs at a height 1.50m so as not to have temperature variations due to different heights. The sensors directly exposed to the sun were protected by screens to avoid their overheating. The main room of the house is consisting of a part on the ground floor and part in mezzanine and its surface is particularly high ($60m^2$ on the ground floor and mezzanine $24m^2$). Of these 10 sensors are placed at first floor, and 6 on the seconde floor. We use the temperature reading of sensors in the first floor, node deployment is shown in Fig. 4.7(b). The details of this project can be found in http://www.citi-lab.fr/project/projets-internes. And all the data can be got from UrbaNet team of CITI lab in INSA de Lyon. Dataset $\mathcal{T}_m$ is a collection of temperature from a museum, we cite this dataset from [30].

Here, we show several data examples of dataset $\mathcal{T}_h$ in table B.1. In this dataset, each node has 7900 data, data ranges from -0.5^0C to 25.2^0C, there are totally 250 different values. The whole dataset can be seen in Fig. 4.1.

Table B.1: Data example of dataset $\mathcal{T}_o$

Node	N13	N14	N6	N3	N2	N15	N1	N4	N11	N5
Temperature [^0C]	5.1	16.5	17.6	17.5	18.8	18.6	18.5	18.2	18.9	16.6
	5	16.8	17.6	17.6	17.8	17.8	17.7	17.6	17.6	16.7
	5	16.7	17.6	17.6	17.4	17.5	17.6	17.3	17.1	16.6
	...	...	...	...	...	...	...	...	...	...

In terms of dataset $\mathcal{T}_m$, we show the data example in table B.2. In this dataset, the node has 720 data, data ranges from 27.46^0C to 28.18^0C, there are totally 19 different values. The dataset can be seen in Fig. B.1.

Table B.2: Data example of dataset $\mathcal{T}_m$

Temperature[^0C]	27.90
	28.02
	28.1
	28.1
	...

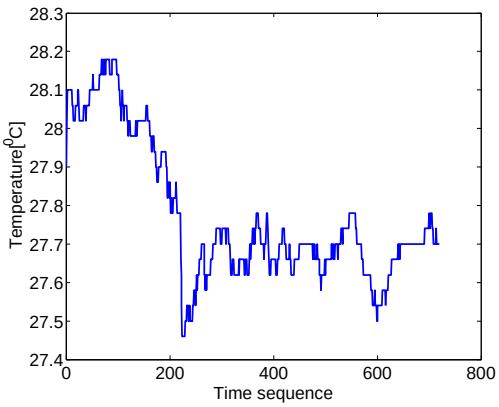


Figure B.1: Dataset $\mathcal{T}_m$.

C

Markov Decision Process

C.1 Introduction

Markov Decision Process (MDP) [116, 117] is a framework which uses a decision agent to choose an appropriate action in order to allow a stochastic system to achieve its optimality objective. For example, suppose the video game illustrated in Fig. C.1, where a cat wants to play with a mouse. The cat movements are controlled by a player, while the mouse movements are randomly generated by the computer. We assume that at each period of time, the player moves the cat one step among four possible directions: up, down, left or right. Immediately after, the mouse moves randomly and without knowing the cat's position. The aim of the player is to position the cat as nearest as possible to the mouse. When the cat catches the mouse, it releases it again and continues the game. At each period of time, the player observes the position of the mouse, and has to decide where to move the cat so that it can stay or it can get as close as possible to the mouse. As we will detail later, the sequence of movements that the player should decide can be obtained using the MDP framework.

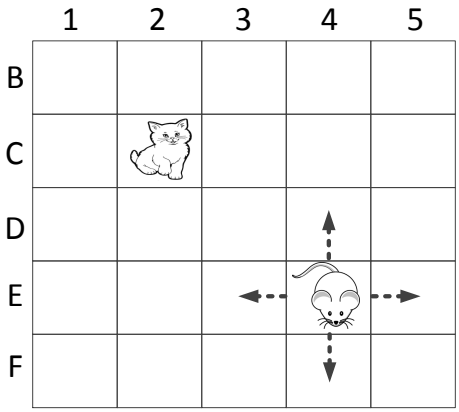


Figure C.1: An example of MDP problem.

C.2 MDP description

As shown in Fig. C.2, in MDP framework, decisions are made at points of time refereed to as *decision epochs*. If the set of decision epochs is discrete, then we consider the problem as discrete time. Otherwise, when the decisions can occur at any time, then the problem can be modeled as continuous time MDP. In the work presented in Chapter 6, our problem is a discrete time one.

If the number of decision epochs is finite, then the problem is a *finite horizon* problem; otherwise, it will be an *infinite horizon* problem [117]. For instance, if we assume that the cat must make a decision to minimise the distance with the mouse before a fixed number of steps, then this type of problem is a finite horizon one, and the number of steps is the horizon length. Otherwise, if we assume that the cat plays with the mouse infinitely and it release it each time it catches it, with the aim to remain as close as possible to the mouse all over the game duration, then this problem in Fig. C.1 has to be modeled as an infinite horizon problem. In the work presented in Chapter 6, we

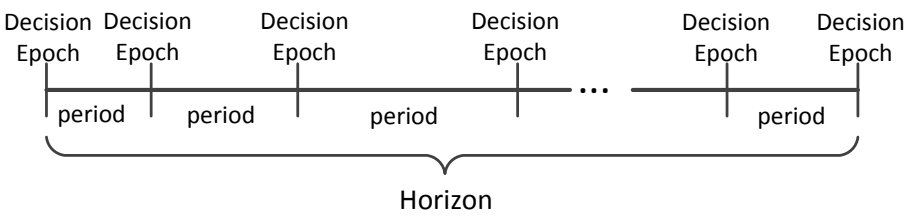


Figure C.2: Decision epochs, period and horizon.

focus on an infinite horizon problem.

A Markov decision process is described by 4 elements: a set of states, a set of actions, a set of transition probabilities and a set of rewards. We will describe these

elements in detail below.

A state is used to describe a possible system status. We denote the set of all possible states (or state space) by Ω. In the example of Fig. C.1, a state x is given by (C, M), where C is the position of cat and M is the position of mouse in the grid $\mathcal{G}$. The state space Ω is composed by all possible pairs of cat and mouse positions:

$$\Omega \equiv \{\forall x = (C, M) | \forall (C, M) \in \mathcal{G}^2\} \tag{C.1}$$

At each epoch, the decision agent (the player) chooses an **action**, which causes the system to transit to a new state. In Fig. C.1, the set A of possible actions is composed of the 4 moving directions: up, down, left and right, which we will refer to a_0, a_1, a_2, a_3, respectively. Notice that there are some states where only a subset of the 4 directions is possible. For example when the cat is in the corner, it can only moves to 2 directions. Therefore, we define $A(x)$ as the set of allowed actions associated to state x:

$$A(x) \subseteq \{a_0, a_1, a_2, a_3\}$$

and we can then express the set of actions as:

$$A \equiv \{A(x) | \forall x \in \Omega\} \tag{C.2}$$

When the player chooses the direction, the system state changes because of the new cat position. Moreover, the system state changes also because of the mouse random movement. We assume that the mouse movement follows a Markovian process. In other words, the random mouse movements depend only on its current position, and not on its past trajectory. We denote $T_{x \to y}(a)$ as the transition probability from state x to state y when the decision agent chooses action $a \in A(x)$. We can express the set of transition probabilities of the MDP as follow:

$$T \equiv \{T_{x \to y}(a) | \forall (x, y) \in \Omega^2, \forall a \in A(x)\} \tag{C.3}$$

R is the set of rewards earned by the system each time the decision agent (the player) makes a choice. As for the transition probabilities, the reward perceived by the system when it transit to state x depends on the action a chosen by the decision agent. We denote $R(x, a)$ this reward quantity. Since in Fig. C.1 the objective of the play is to minimize the cat distance to the mouse at each epoch, then the reward $R(x, a)$ of Fig. C.1 can be expressed as:

$$R(x, a) = \frac{1}{D(x)}$$

where $D(x)$ is the Euclidean distance between the cat and the mouse associated to state x. We can then express the set of rewards as follow:

$$R \equiv \{R(x,a)|\forall x \in \Omega, \forall a \in A\} \tag{C.4}$$

Notice that the Markov property is obtained because the transition probabilities, reward functions and actions chosen by the decision agent in that state depend only the current state of the system.

The main problem of the player is to determine the sequence of optimal actions (movements) that it has to choose in each state of the system in order to achieve the optimality objective: keeps the cat as near as the mouse all over the game duration. We will express formally the above optimal objective in the next section.

In MDP framework, the sequence of actions is called control policy (d). The set of all possible policies is denoted as Π. An optimal control policy $d^* \in \Pi$ is the sequence of optimal action for each states and the associated event. We will describe in Sec. C.4 an algorithm which allows to compute the optimal control policy.

C.3 Optimality objective: the average reward

In MDP infinite horizon problems, a classical objective is to maximize the average reward of a system (respectively minimize the average cost). To express this quantity, one need to use the following *Bellman equation* [131]:

$$V_t^d(x) = R(x, d(x)) + \sum_{y \in \Omega} T_{x \to y}(d(x)) \cdot V_{t-1}^d(y) \tag{C.5}$$

Here $V_t^d(x)$ is the total reward perceived by the system after t decision epochs, assuming that the decision agent apply a control policy $d \in \Pi$. $R(x, d(x))$ is the reward eared in state x, $T_{x \to y}(d(x))$ is the probability from state x to y under action $d(x)$, and $V_{t-1}^d(y)$ is the value of state y at decision epoch $t-1$.

Let $g^x(d)$ be the average reward of the system, if the initial state of the MDP is $x \in \Omega$ and the decision agent apply a control policy $d \in \Pi$. Then $g^x(d)$ can be expressed as:

$$g^x(d) = \lim_{t \to +\infty} \frac{V_t^d(x)}{t}$$

If Ω contains at least one recurrent state [120], then the average reward of the system is not related with the initial state x, i.e.,:

$$g(d) = g^x(d) = \lim_{t \to +\infty} \frac{V_t^d(x)}{t} \tag{C.6}$$

Thus, the optimal control policy d^* which maximise the average reward is given by the following expression:

$$\forall x \in \Omega, d^* = \arg\max_{d\in\Pi} g(d) = \arg\max_{d\in\Pi} \lim_{t\to+\infty} \frac{V_t^d(x)}{t} \tag{C.7}$$

The optimal control policy can be computed using different well-known algorithms, such as linear programming [117], dynamic programming [132]: policy iteration and value iteration. In the next section, we will detail value iteration algorithm that we used in Chapter 6.

C.4 Value iteration

Value iteration algorithm computes recursively Bellman equation. Precisely, it computes at iteration n the value function $V_n(x)$ associated to each state $x \in \Omega$ as follow:

$$V_n(x) = \max_{a\in A(x)} R(x,a) + \sum_{y\in\Omega} T_{x\to y}(a) \cdot V_{n-1}(y) \tag{C.8}$$

This quantity can be interpreted as the maximal total expected reward with n periods left to the time horizon when the current state is x and a terminal reward of $v_0(y)$ is incurred when the system ends up at state y. The chosen policy d^n after the n^{th} iteration of the algorithm is characterized by the following relations:

$$\forall x \in \Omega, d^n(x) = \arg\max_{a\in A(x)} R(x,a) + \sum_{y\in\Omega} T_{x\to y}(a) \cdot V_{n-1}(y)$$

It has been proven [117] that when $n \to \infty$, $|V_n(x) - V_{n-1}(x)|$ converge to the average gain $g(d^*)$ (Eq. C.6). Thus, the algorithm converges when $\exists n$ such that $\forall (x,y) \in \Omega^2$,

$$|V_n(x) - V_{n-1}(x)| \simeq |V_n(y) - V_{n-1}(y)| \simeq g(d^*)$$

thus a stopping condition at the n^{th} iteration is met when $0 \leq M_n - m_n \leq \varepsilon m_n$, where

$$M_n \equiv \max_{x\in\Omega} |V_n(x) - V_{n-1}(x)|,\ m_n \equiv \min_{x\in\Omega} |V_n(x) - V_{n-1}(x)| \tag{C.9}$$

The algorithm starts with any arbitrarily chosen function $V_0(x)$, $\forall x \in \Omega$, a necessary condition to guarantee the convergence of the algorithm is to satisfy the following initialization rule:

$$\forall x \in \Omega, 0 \leq V_0(x) \leq \max_{a\in A(x)} \{R(x, a(x))\}$$

The pseudo code of value iteration algorithm 3 is illustrated below. In step 3 of the algorithm 3, the convergence test can be achieved because: if $\forall n \geq 1$ and $\forall x \in \Omega$, $V_n(x)$ is defined by Eq. C.8 and M_n, m_n are the quantities defined by Eq. C.9, then the series $\{m_n | n \geq 1\}$ and $\{M_n | n \geq 1\}$ are respectively an increasing and decreasing real sequences, thus $\exists n \geq 1$, such that $m_n \leq g(d^*) \leq M_n$.

Algorithm 3 Value_iteration

Require:
1: State space, Ω;
2: Action set, A;
3: Transition probability, T;
4: Reward, R;
Ensure: optimal policy d^*
5: **STEP 1.** % initialization
6: **for** all $x \in \Omega$ **do**
7: choose $V_0(x)$ such that $0 \leq V_0(x) \leq \max_{a \in A(x)} \{R(x, a(x))\}$;
8: **end for**
9: n=1;
10: $\varepsilon \simeq 0$
11: **STEP 2.** %Relative values and the optimal actions updates
12: **for** all $x \in \Omega$ **do**

$$V_n(x) = \max_{a \in A(x)} \{R(x, a) + \sum_{y \in \Omega} T_{x \to y}(a) V_{n-1}(y)\}$$

$$d^*(x) = \arg \max_{a \in A(x)} \{R(x, a) + \sum_{y \in \Omega} T_{x \to y}(a) V_{n-1}(y)\}$$

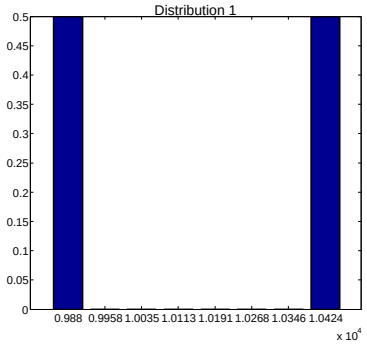
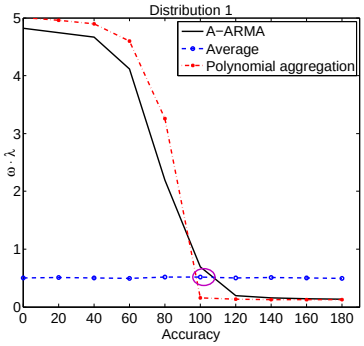
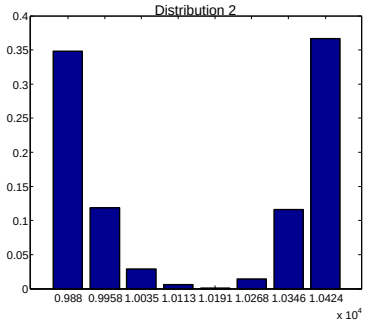
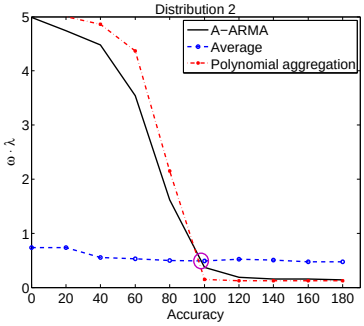
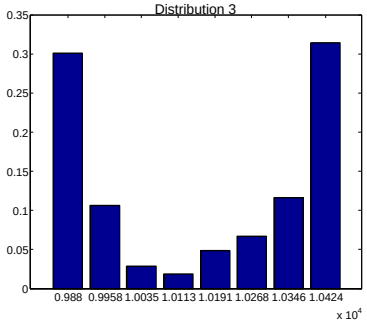
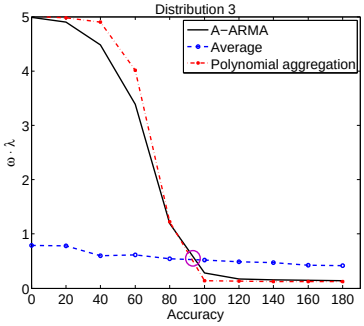
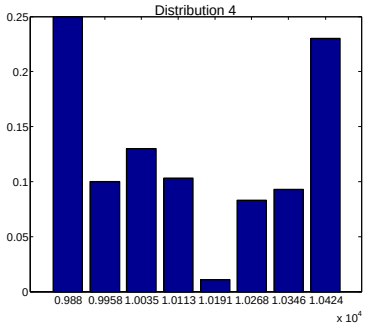
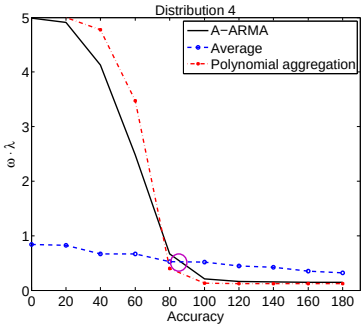
13: **end for**
14: **STEP 3.** % Convergence test
15: $m_n = \min_{x \in \Omega} |V_n(x) - V_{n-1}(x)|$
16: $M_n = \max_{x \in \Omega} |V_n(x) - V_{n-1}(x)|$
17: **if** $0 \leq |M_n - m_n| \leq \varepsilon m_n$ **then**
18: **return** d^*
19: **else**
20: $n = n + 1$
21: Goto **STEP 2**
22: **end if**

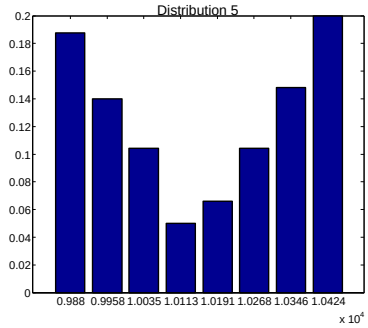
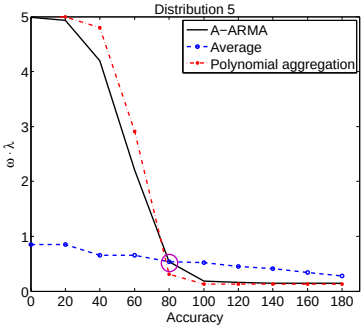
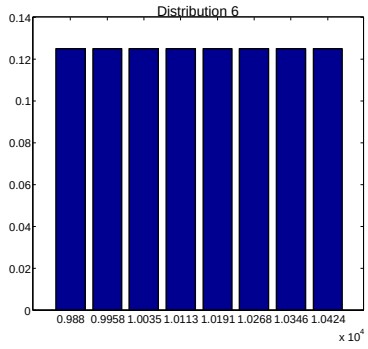
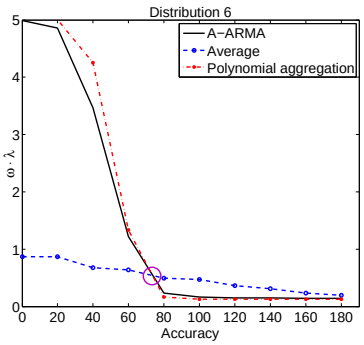
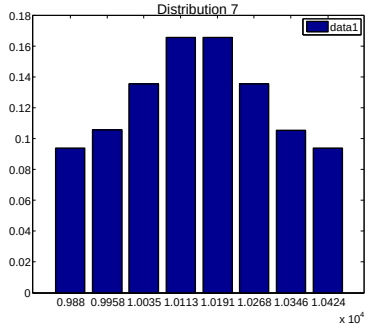
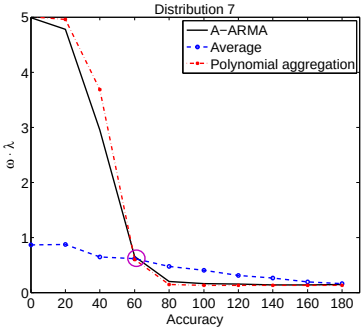
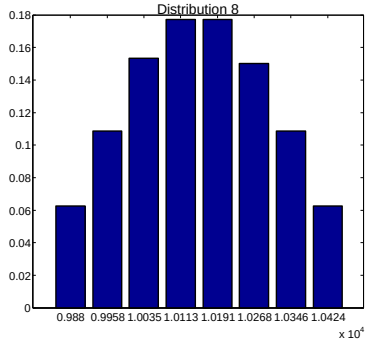
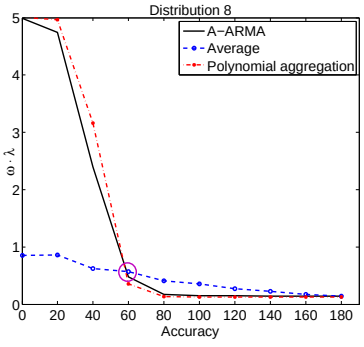
D

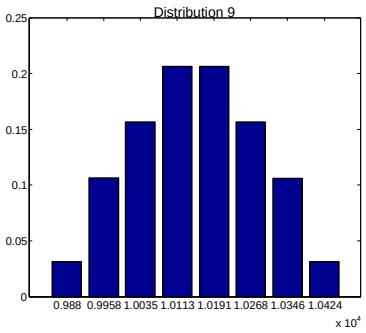
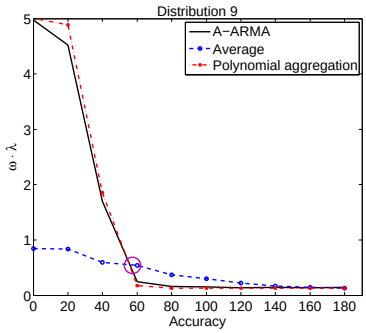
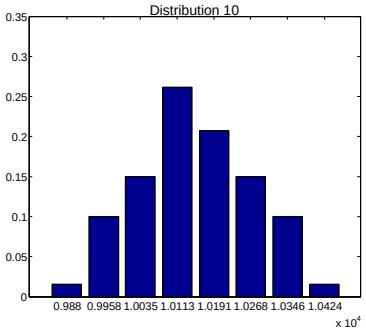
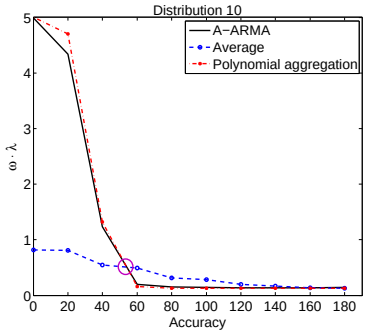
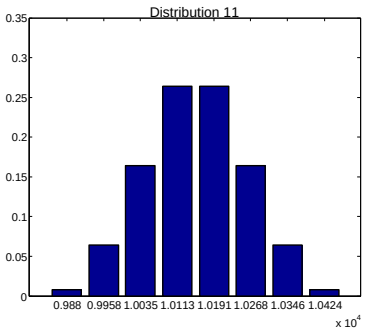
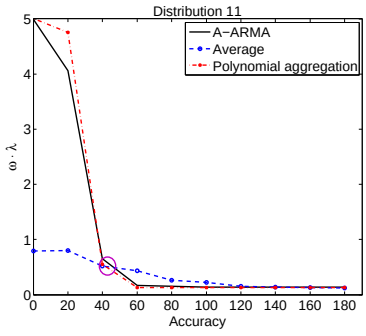
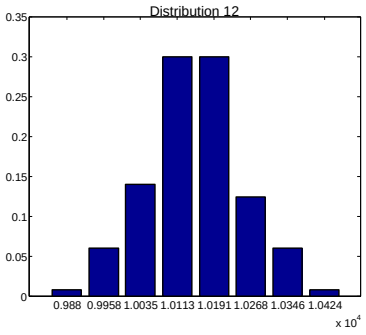
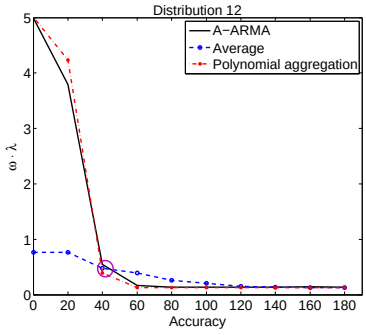
Data distribution and aggregation functions performances

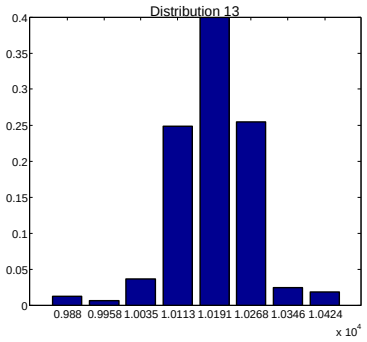
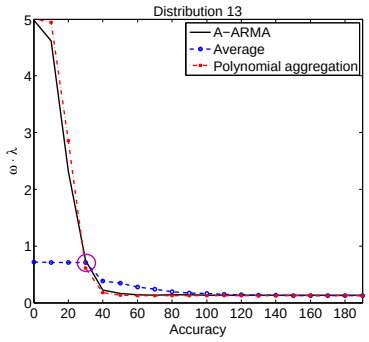
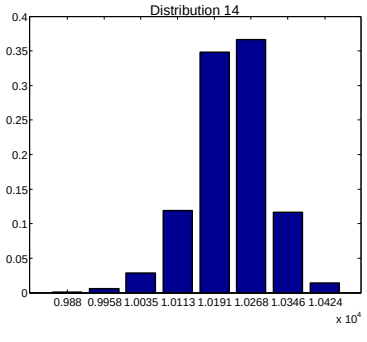
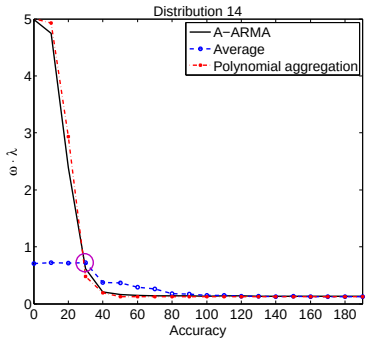
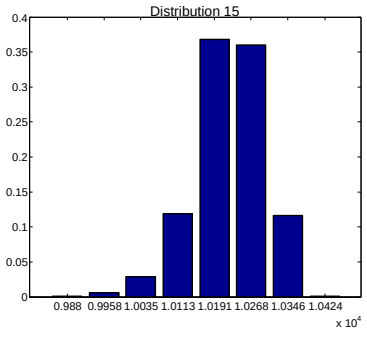
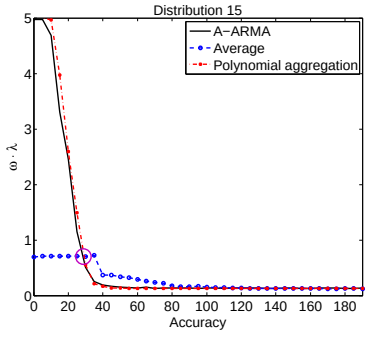
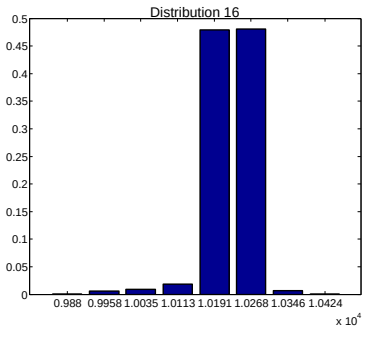
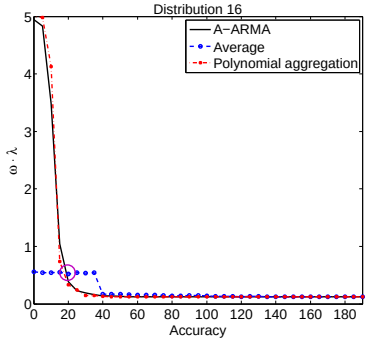
This appendix shows all the 16 data distributions and the corresponding aggregation functions performances (using data set $\mathcal{P}_s$). The metrics in the third column are calculated by Eq. 6.8,6.9,6.10,6.11,6.12,6.13 respectively, where E is expected value, V is variance, H is entropy, d_{Euc} is Euclidean distance, d_{KL} is Kullback-Leibler divergence, and d_{Js} is Jensen-shannon divergence (d_{Euc}, d_{KL} and d_{Js} are referred to distribution 1). The purple circles in the figures in second column denote *cross accuracy*, where *crossed accuracy* is the accuracy that A-ARMA performs better than Average (intersection of the performances lines of A-ARMA and Average). The sequence of the distributions are sorted as the *cross accuracy* decreasing. When accuracy requirement is bigger than this *cross accuracy*, A-ARMA save more energy and network capacity than average. The sampled data values are 9880,9958,10035,10113,10191,10268,10346,10424 respectively.

Table D.1: Data distributions and aggregation function performances for dataset $\mathcal{P}_s$.

	Data distribution	Function performance	Metrics
1			• E 10152 • V 73984 • H 1 • d_{Euc} 0 • d_{KL} 0 • d_{Js} 0
2			• E 10154.58 • V 62338.41 • H 2.07 • d_{Euc} 0.26 • d_{KL} 0.48 • d_{Js} 0.16
3			• E 10163.1 • V 55141.12 • H 2.48 • d_{Euc} 0.33 • d_{KL} 0.7 • d_{Js} 0.23
4			• E 10136.03 • V 45590.9 • H 2.48 • d_{Euc} 0.43 • d_{KL} 1 • d_{Js} 0.31

	Data distribution	Function performance	Metrics
5			• E 10157.49 • V 42468.52 • H 2.88 • d_{Euc} 0.51 • d_{KL} 1.42 • d_{Js} 0.42
6			• E 10151.88 • V 31678.36 • H 3 • d_{Euc} 0.61 • d_{KL} 2 • d_{Js} 0.55
7			• E 10151.85 • V 25987.99 • H 2.96 • d_{Euc} 0.67 • d_{KL} 2.42 • d_{Js} 0.63
8			• E 10151.48 • V 22074.76 • H 2.91 • d_{Euc} 0.72 • d_{KL} 3 • d_{Js} 0.72

	Data distribution	Function performance	Metrics
9			• E 10151.79 • V 17484.4 • H 2.78 • d_{Euc} 0.80 • d_{KL} 4 • d_{Js} 0.83
10			• E 10149.74 • V 14618.82 • H 2.65 • d_{Euc} 0.80 • d_{KL} 5 • d_{Js} 0.90
11			• E 10151.84 • V 11255.99 • H 2.49 • d_{Euc} 0.83 • d_{KL} 5.97 • d_{Js} 0.94
12			• E 10150 • V 10193.68 • H 2.41 • d_{Euc} 0.84 • d_{KL} 5.97 • d_{Js} 0.94

	Data distribution	Function performance	Metrics
13			• E 10188.36 • V 6969.84 • H 2.06 • d_{Euc} 0.87 • d_{KL} 7.13 • d_{Js} 0.95
14			• E 10225.1 • V 6412.53 • H 2.07 • d_{Euc} 0.88 • d_{KL} 7.13 • d_{Js} 0.95
15			• E 10221.5 • V 5888.2 • H 2 • d_{Euc} 0.89 • d_{KL} 9.11 • d_{Js} 1.00
16			• E 10224.74 • V 2537.99 • H 1.28 • d_{Euc} 0.98 • d_{KL} 12.29 • d_{Js} 1.00