PhD-FSTC-25-2010
The Faculty of Sciences,
Technology and Communication

UFR Mathématique
Informatique et Mécanique

DISSERTATION

Defense held on 28/09/2010 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG
EN INFORMATIQUE
ET
DOCTEUR DE L'UNIVERSITÉ PAUL VERLAINE – METZ
EN INFORMATIQUE

by

Julien SCHLEICH
Born on 8th May 1983 in Luxembourg (Luxembourg)

ROBUST DOMINATING SET BASED VIRTUAL BACKBONES FOR WIRELESS AD HOC NETWORKS

Dissertation defense committee

Dr Pham Dinh Tao, Chairman
Professor, INSA Rouen

Dr Stephen Rothkugel, Vice Chairman
Assistant Professor, Université de Luxembourg

Dr Serge Chaumette, Reviewer
Professor, Université de Bordeaux 1

Dr Frédéric Guinand, Reviewer
Professor, Université du Havre

Dr Pascal Bouvry, Dissertation Supervisor
Professor, Université du Luxembourg

Dr Lê Thi Hoài An, Dissertation Supervisor
Professor, Université Paul Verlaine – Metz, France

Abstract

Ad hoc networks are infrastructure-less spontaneous networks generally composed of wireless and mobile devices. From a practical point of view, ad hoc technologies offer solutions when infrastructure-based network are too costly, damaged or not suitable. Despite a wide panel of scenarios and the huge number of ad hoc capable devices currently in use, this technology is not widely used because of technical considerations mainly related to the lack of a global coordinator. In this thesis, we propose two different approaches to create virtual backbones in order to organize ad hoc networks. In a first time, we propose a centralized algorithm based on DC programming and DCA to solve the Min m-Vertex Dominating Set Problem and in a second part, we develop distributed and asynchronous algorithms, relying on 2-hop knowledge only, to build k-Vertex Connected m-Vertex Dominating Set-based Virtual Backbones. A global overview of the domain is provided through an extensive state-of-the-art and a hierarchical classification. The efficiency of both approaches is demonstrated with a wide panels of simulations, from randomly generated graphs to more realistic scenarios.

I would like to dedicate this thesis to my wife Caroline, my daughter Clémentine and my parents Marie-Astrid and Philippe.

Acknowledgements

First I would like to thanks my two co-advisors for their wise supervision that provided me with freedom and allowed me to take my own decisions. These three years have been formative and enriching.

I also would like to thank the team at the University of Luxembourg for their kindness and the things we shared, in particular everything that was not work-related.

I thank my family that allowed me to comfortably study at the University.

Finally I thank my friends that supported me during these three years even though I was annoying them with my work.

Contents

List of Figures x

List of Tables xi

Nomenclature xiii

List of Algorithms xiii

Definitions xvi

I Thesis 1

1 Introduction 3
- 1.1 Context . . . 4
- 1.2 Motivation . . . 5
- 1.3 Contributions . . . 5
- 1.4 Dissertation Outline . . . 6

2 Ad hoc networks 7
- 2.1 Presentation of ad hoc networks . . . 8
  - 2.1.1 Definitions . . . 8
  - 2.1.2 A comparison with real world communication . . . 9
  - 2.1.3 Characteristics and limitations . . . 10
    - 2.1.3.1 Wireless communication . . . 10
    - 2.1.3.2 Unstructured and/or time-varying network topology . . . 11
    - 2.1.3.3 Energy conservation . . . 11
    - 2.1.3.4 Resource-constrained computation . . . 12
    - 2.1.3.5 Scalability . . . 12
- 2.2 Use cases . . . 13
  - 2.2.1 No infrastructure . . . 13
  - 2.2.2 Damaged infrastructure . . . 13
  - 2.2.3 Cost of infrastructure . . . 13
- 2.3 Modeling ad hoc networks . . . 15
  - 2.3.1 Wireless channel . . . 15

2.3.2 Communication graph . . . 16
2.3.2.1 Static graphs . . . 16
2.3.2.2 Dynamic graphs . . . 18
2.3.3 Mobility model . . . 19
2.3.3.1 Synthetic models. . . . 20
2.3.3.2 Trace-based models. . . . 20
2.4 Recap of key points . . . 21

3 Topology management **23**
3.1 Motivation . . . 24
3.1.1 Scalability problems . . . 24
3.1.1.1 Density and available bandwidth . . . 24
3.1.1.2 Broadcast storm problem . . . 25
3.1.2 Two different approaches . . . 25
3.1.2.1 Topology control . . . 25
3.1.2.2 Virtual backbones . . . 27
3.2 Virtual backbones . . . 30
3.2.1 Definition . . . 30
3.2.2 Taxonomy . . . 31
3.2.2.1 Tree-based virtual backbone . . . 31
3.2.2.2 Cluster-based virtual backbone . . . 34
3.2.2.3 Dominating Set-based virtual backbone . . . 36
3.3 Metrics for virtual backbones . . . 37
3.3.1 Motivations and related criteria . . . 37
3.3.1.1 Motivations . . . 37
3.3.1.2 Quality criteria . . . 37
3.3.2 State-of-the-art . . . 39
3.3.2.1 Tree-based virtual backbones . . . 39
3.3.2.2 Cluster-based virtual backbones . . . 39
3.3.2.3 CDS-based virtual backbones . . . 40
3.3.3 Contributions for $CDS-$based virtual backbone . . . 40
3.3.3.1 Size . . . 41
3.3.3.2 Stability . . . 41
3.3.3.3 Representation . . . 42
3.4 Recap of key points . . . 44

4 Taxonomy for CDS-based virtual backbone **45**
4.1 Characteristics . . . 46
4.1.1 Approaches . . . 46
4.1.1.1 Exact . . . 46
4.1.1.2 Approximation . . . 47
4.1.1.3 Others . . . 47
4.1.2 Computation type . . . 47
4.1.2.1 Centralized . . . 47
4.1.2.2 Distributed / Decentralized . . . 47
4.1.3 Available data . . . 48

4.1.3.1 Global algorithms . . . 48
4.1.3.2 Localized algorithms . . . 48
4.1.4 Randomization . . . 49
4.1.4.1 Deterministic . . . 49
4.1.4.2 Stochastic . . . 49
4.1.5 Robustness considerations . . . 49
4.1.6 Synchronization requirements . . . 50
4.2 Taxonomy . . . 51
4.2.1 Centralized algorithms . . . 51
4.2.1.1 Exact algorithms. . . . 51
4.2.1.2 Approximation algorithms. . . . 51
4.2.1.3 Others algorithms. . . . 52
4.2.2 Distributed algorithms . . . 53
4.2.2.1 Self-pruning algorithms. . . . 54
4.2.2.2 Maximum Independent Set based algorithms. . . . 56
4.2.2.3 Multi-Point Relay algorithms. . . . 57
4.2.3 Robust CDS algorithms . . . 58
4.2.3.1 Increasing both domination and connectivity . . . 60
4.2.3.2 Increasing the domination . . . 61
4.2.3.3 Increasing the connectivity . . . 62
4.3 Recap of key points . . . 67

5 Dominating Set **69**
5.1 Definition . . . 70
5.1.1 Dominating set . . . 70
5.1.2 m-vertex domination property . . . 71
5.1.3 l-level domination property . . . 72
5.1.4 k-connectivity property . . . 72
5.2 State-of-the-art . . . 74
5.2.1 m,l-Dominating Sets . . . 74
5.2.1.1 Exact approaches . . . 74
5.2.1.2 Approximation and heuristic approaches . . . 74
5.2.1.3 Variants . . . 74
5.2.2 k,m,l-Connected Dominating Sets . . . 75
5.3 Centralized solution . . . 76
5.3.1 Motivation . . . 76
5.3.1.1 Motivation for the centralized approach . . . 76
5.3.1.2 Motivation for m,l-Dominating Set . . . 77
5.3.1.3 Motivation for using DC programming and DCA . . . 77
5.3.2 Difference of convex function techniques . . . 77
5.3.3 Mathematical model . . . 80
5.3.3.1 min DS model . . . 80
5.3.3.2 $m-$vertex domination constraint . . . 80
5.3.3.3 Generic scheme for $l-$level domination . . . 81
5.3.3.4 The DC formulation . . . 82

5.3.3.5 The DC Algorithm . . . 83
5.4 Distributed solutions . . . 86
5.4.1 Motivations . . . 86
5.4.2 Prerequisites . . . 87
5.4.3 Blackbone algorithm . . . 88
5.4.3.1 Vertex connectivity . . . 88
5.4.3.2 Blackbone subroutines . . . 90
5.4.3.3 Practical issues . . . 92
5.4.3.4 Theoretical study . . . 94
5.4.4 Blackbone2 algorithm . . . 95
5.4.4.1 Blackbone limitations . . . 95
5.4.4.2 k-vertex connectivity . . . 96
5.4.4.3 m-vertex domination . . . 98
5.4.4.4 Blackbone 2 main loop and sub-routines . . . 100
5.4.4.5 Optimizations . . . 102
5.4.4.6 Theoretical performances . . . 104
5.5 Recap of key points . . . 105

6 Experimentation **107**
6.1 Centralized approach . . . 108
6.1.1 Experimental setup . . . 108
6.1.2 Global methods are not scalable . . . 109
6.1.3 Finding good local solutions . . . 110
6.1.4 Influence of parameters . . . 111
6.1.4.1 The influence of graph average density . . . 111
6.1.4.2 The influence of the domination parameter . . . 113
6.1.5 Results analysis . . . 113
6.2 Distributed approach . . . 115
6.2.1 Experimental setup . . . 115
6.2.2 Blackbone 1 - static networks . . . 117
6.2.2.1 Experiment specific settings . . . 117
6.2.2.2 Performance comparison for k=1-vertex connected DS . . . 117
6.2.2.3 Performance comparison for k=2-vertex connected DS . . . 118
6.2.2.4 Results analysis . . . 119
6.2.3 Blackbone2 - static networks . . . 120
6.2.3.1 Experiment specific settings . . . 120
6.2.3.2 Performance comparison for k=2-vertex connected DS . . . 120
6.2.3.3 Performance comparison for k=1-vertex connected DS . . . 120
6.2.3.4 Impact of the vertex domination value . . . 122
6.2.3.5 Results analysis . . . 122
6.2.4 Blackbone 2 - dynamic networks . . . 124
6.2.4.1 Experiment specific settings . . . 124
6.2.4.2 Age and speed threshold do not work . . . 125
6.2.4.3 Performance comparison for each quality criterion . . . 125
6.2.4.4 Results summary . . . 130

6.2.5 Results analysis 131
6.3 Recap of key points 132

7 Conclusions and perspectives **133**
7.1 Conclusions 134
7.1.1 Centralized algorithm 134
7.1.2 Distributed algorithms in static networks 134
7.1.3 Distributed algorithms in dynamic networks 135
7.2 Perspectives 135

II Appendices **137**

A Scripts for DCA **139**
A.1 AMPL models 140
A.1.1 Integer programming model for min m-DS 140
A.1.2 Linear programming model for min m-DS 140
A.1.3 DCA model for min m-DS 141
A.2 Random graph generator implementation 142

B Simulation parameters for OMNeT++ **143**
B.1 omnetpp.ini 144
B.2 params.ini 145

References **147**

Publications **156**

Index **156**

List of Figures

2.1 Organization chart of the Audit & Sons company 11
2.2 A simple Unit Disk Graph with two network nodes. 18
2.3 In a Unit Disk Graph, a node can have a maximum of five independent neighbors. 18

3.1 Node u needs to send a packet to node v. To achieve its goal in a energy efficient fashion, u has to choose between direct and multi-hops communication. 26
3.2 A conflicting scenario due to the wireless shared-medium. Circles represents the radio coverage area with transmit power P. 27
3.3 Classical networks backbone and ad hoc network virtual backbone. 29
3.4 Topology control solution versus virtual backbone solution. 29
3.5 On the left the communication graph with edges values. 32
3.6 On the left, the maximum leaf spanning tree of a graph example. On the right the corresponding minimum connected dominating set. 34
3.7 On the same graph, a clustering solution and a dominating set. 35

4.1 Organization of the CDS-based virtual backbone taxonomy. 51
4.2 The nine blue vertices form a maximum independent set for the Generalized Petersen graph GP(12,4). 57

5.1 A solution for the five queens problem. All squares are either occupied by a queen or under the direct control of one or more queens. 70
5.2 Bold nodes represent the dominators. Figures 5.2a and 5.2b are two different solutions for the minimum dominating set problem. Figure 5.2b is also a solution for the minimum $2-$vertex dominating set problem. Figure 5.2c is an optimal solution for the $2-$level dominating set problem. 71
5.3 Considering node v point of view, all its 2-hop neighbors x, y can be reached by at least two different nodes a, b for x and b, c for y 89
5.4 On the left, the graph is $2-$vertex connected and as such there is no articulation point. On the right, node B is an articulation point as its removal would increase the number of connected components. 97

6.1 Results for graph density $\in [2, 15]$ 111
6.2 Influence of the average graph density on the dominating set size 112
6.3 Influence of the domination parameter 113
6.4 Size of the virtual backbones versus the total number of nodes 118
6.5 Results comparison for the biconnected virtual backbone 121

6.6 Results comparison for connected dominating set virtual backbones 122
6.7 Impact of the vertex-domination value 123
6.8 Results comparison with different age threshold values 127
6.9 Results comparison with different speed threshold values 127
6.10 Influence of the density on the backbone size 127
6.11 Influence of the speed on the backbone size 128
6.12 Influence of the density on the number of state changes 128
6.13 Influence of the speed on the number of state changes 128
6.14 Influence of the density $k = m = 1$ 129
6.15 Influence of the speed $k = m = 1$ 129
6.16 Influence of the forced stability threshold on the average number of state changes ($k = m = 2$) 129

List of Tables

2.1 Nominal power consumption of the CISCO IEEE 802.11 a/b/g wireless card. Power consumption is measured by the drained current, expressed in mA. . . 12

4.1 Summary of the main characteristics for the centralized algorithms. Δ is the maximum degree of a graph, n the number of nodes and H the harmonic function. 54
4.2 Summary of the main characteristics for the distributed algorithms. Δ is the maximum degree of a graph, n the number of nodes. 59
4.3 Summary of the main characteristics for the centralized algorithms. Δ is the maximum degree of a graph, n the number of nodes, m the number of edges and $Diam$ is the diameter of the graph. 63
4.4 Summary of the main characteristics for the distributed algorithms. Δ is the maximum degree of a graph, n the number of nodes, m the domination value and $Diam$ is the diameter of the graph. 66

6.1 Time (in seconds) to obtain solution with 100 to 300 nodes and density 10. . 110
6.2 Dominating set size (in %) for graphs from 100 to 300 nodes with density 10. 110
6.3 Dominating set size (in percentage) for graphs with 5000 nodes 112
6.4 Processing time for graphs with 5000 nodes 113
6.5 Period values for the simulations 116
6.6 Average density per number of node in the network 117
6.7 Computation time of the compared algorithms 121

List of Algorithms

1	First algorithm from Guha and Khuller	52
2	Second algorithm from Guha and Khuller	53
3	Butenko algorithm	53
4	Self-pruning distributed algorithm template for any node u	55
5	MIS-based algorithm template	57
6	CDSMIS algorithm	62
7	ICGA connectivity augmentation algorithm	64
8	Greedy algorithm for the $\min -DS$ problem	75
9	DCA to solve the $\min m - DS$ problem	84
10	DCA with restart mechanism	85
11	Blackbone main loop	92
12	Re-construction fix algorithm	94
13	findArt: an efficient algorithm for articulation point detection	98
14	m-domination checking algorithm	99
15	Pruning rule of the Blackbone 2 algorithm for the black node v	101
16	Construction rule of the Blackbone 2 algorithm for the white node v	101
17	Main algorithm of Blackbone2	102
18	Random graph generator	108
19	Greedy algorithm for the $\min -DS$ problem	109

List of definitions

1 Ad hoc - *Cambridge Dictionary of American English* . . . 8
2 Ad hoc network - *Synthetic definition* . . . 9
3 Network lifetime . . . 12
4 Scalability - *Bondi et al.* . . . 12
5 Network . . . 16
6 Range assignment . . . 17
7 Communication graph $G(N, E)$ . . . 17
8 Unit disk graph . . . 17
9 Dynamic network . . . 19
10 Dynamic communication graph . . . 19
11 Topology Control - *Santi* . . . 25
12 Virtual backbone - operational point-of-view . . . 31
13 Virtual backbone - graph theory definition . . . 31
14 Tree . . . 32
15 Spanning tree . . . 32
16 Clustering / Cluster analysis . . . 34
17 Dominating Set . . . 36
18 Connected Dominating Set . . . 36
19 Robustness - *Oxford Dictionary* . . . 38
20 Robustness - *Basagni* . . . 40
21 Exact algorithm . . . 46
22 Distributed or decentralized algorithm . . . 48
23 Deterministic algorithm . . . 49
24 Stochastic algorithm . . . 49
25 Piece - *Guha and Khuller* . . . 52
26 Weakly connected dominating set . . . 52
27 Effective degree - *Butenko et al.* . . . 53
28 Independent Set . . . 56
29 Free node - *Wu MPR* . . . 58
30 Dominating Set . . . 71
31 Domination number . . . 71
32 m-vertex dominating set - m-domination set . . . 72
33 Distance - *Graph Theory* . . . 72

34 l-level dominating set 72
35 m,l-Dominating Set 72
36 k-vertex connected graph or k-connected graph 73
37 k-edge connected graph 73
38 Cut-Vertex or Articulation Point 73
39 Separation set 73
40 k-Block 73
41 k-Leaf Block 73
42 k,m,l-Connected Dominating Set 73
43 Connected component of a graph $G = (V, E)$ 96
44 Articulation point - Cut vertex 96
45 Biconnected graph 97

Part I

Thesis

Chapter 1

Introduction

Contents

1.1 Context 4
1.2 Motivation 5
1.3 Contributions 5
1.4 Dissertation Outline 6

1.1 Context

Nowadays communicating devices are expected to be mobile and be able to access the Internet via high-speed data links. Cellphones, PDAs [1], netbooks [2] and more recently tablets [3] are very popular mobile devices with various communication capabilities such as 2G/3G cellular network, WiFi (b/g/n) or Bluetooth. Whatever the network interface is, communications are generally handled via an infrastructure-based network, i.e. a set of hierarchically-organized dedicated relay devices. Such an approach generally ensures a very good quality of service whatever the type of service (web browsing, audio/video streaming, large file transfer, ...) and its associated requirements (low end-to-end delay, constant bit rate, high speed transfers, ...).

However, by considering the mobile nature of current communications, infrastructure-based networks cannot be considered the perfect solution. Over the last fifteen years, another type of communication, namely ad hoc, has been extensively studied. In such an approach, end user terminals are the only existing network entities, i.e. no dedicated devices are helping relaying messages. As a consequence, this technology relies on the cooperation among network users to guaranty the delivery of messages in an organization-free structure.

From a theoretical efficiency point of view, ad hoc networks are more suitable to deal with today's ever changing network topologies. Indeed, as users are moving through time, this mobility induces major changes in the localization of the network density and thus the network load distribution is changing accordingly. As an example, let us consider business city centers, which are very dense places during working hours and quite sparse at night. In an infrastructure network, such places should be characterized by high capacity devices to deal with huge working hours telecommunication requirements. At night however, these dedicated devices are nearly useless. In ad hoc networks, such an efficiency problem does not exist as no dedicated devices are required: the network is created on purpose by the participating nodes and it disappears if it is not useful anymore.

From a more practical point of view, ad hoc technologies offer solutions when infrastructure-based network are too costly, damaged or not suitable. Mobile users always require wider network-covered areas. As fixed equipment such as cell towers are expensive, ad hoc technologies are a good complement approach to extend the actual covered area. Wide area sensing in remote or hostile places are another important use case for the ad hoc networking technologies. Many applications are possible, from fire detection in forest to underground petrol level or seismic activity monitoring. Ad hoc networks can also be deployed to complement or replace a damaged infrastructure network in case of natural disaster to help coordinating search & rescue teams.

Despite these promising scenarios and the huge number of ad hoc capable devices currently in use, this technology is not widely used because of technical considerations. Indeed, for the last fifteen years, research works have been conducted to solve problems related to the organization, energy efficiency or security issues that arise in such distributed environments.

In this dissertation, organization issues are tackled in particular. Many contributions have been proposed to cope with the lack of hierarchical structure by creating so-called virtual

[1] Personal Digital Assistant

[2] lightweight laptops, generally characterized by a small screen size

[3] multi-touch screen device bigger than a cellphone

backbone, i.e. a subset of network nodes and / or communication channels in charge of organizing the complete network.

1.2 Motivation

Ad hoc networks can not rely on dedicated network devices to relay or route messages from a source to a destination. As there is no global coordinator, nodes have to organize themselves to avoid scalability issues that may arise when the size of the network is growing. Virtual backbones, by selecting a subset of network nodes and / or communication channels, are a set of techniques mimicking the infrastructure in the classical network paradigm. Ad hoc virtual backbones however are not composed of dedicated devices but from regular network nodes which were selected to help relaying messages.

This dissertation focuses on dominating set based virtual backbones and their variants based on $k-$vertex-connectivity, $m-$vertex domination or $l-$level domination. We have chosen this Graph Theory formalism as it provides well defined notions that fits our requirements and can be adapted to various situations.

Many contributions based on the same notions have been proposed for the last fifteen years. Some of them are theoretical work and generally centralized approximation algorithms. As a consequence, applying them on real devices is not realistic as they do not scale well. Distributed solutions of many kinds based on various assumptions have also been studied. Some of them are designed to create robust virtual backbones, i.e. backbone resilient to node failure or mobility.

Although these solutions are providing insightful results they can still be enhanced according to some different views. First, centralized algorithms are generally based on simple heuristics or greedy mechanisms to create their solutions as finding the optimal solution for the considered problems is NP-Hard and thus non-tractable. The difference of convex function algorithm (DCA) is an innovative technique in non-convex programming a cutting-edge technique developed in the Operational Research domain and characterized by its high efficiency and a proved convergence. As a consequence, we decided to applied this method in order to find compact dominating set based virtual backbones. The proposed distributed algorithms are generally difficult to implement in real situations because either they rely on implicit synchronization of the network or they require some global information such as the total number of nodes to be functional. In this work we propose two distributed algorithms to create robust domination set based virtual backbones that only requires local information and are functional in an asynchronous context.

1.3 Contributions

The dissertation contains the following contributions:

- An extensive state-of-the-art on the $CDS-$based virtual backbone techniques. Centralized, distributed and robust algorithms are classified independently as their most important characteristics differ.

- Modeling of the $m-$vertex domination and the $l-$level domination constraints for the general minimum $m-$vertex $l-$level dominating set problem. These two contributions extend the original and well-known integer programming model of the minimum dominating set problem.

- A centralized DC Algorithm to solve the minimum $m-$vertex $l-$level dominating set problem (DCA). A restart mechanism is also proposed to increase the exploration of the solution space.
- Two distributed and localized algorithms, Blackbone 1 and 2, designed to create $k-$vertex connected $m-$vertex dominating set virtual backbones in an asynchronous and computer effective fashion.
- A set of quality metrics to analyze the behaviour of $CDS-$based virtual backbones algorithm in mobile ad hoc networks.

1.4 Dissertation Outline

The reminder of the dissertation is organized as follows:

Chapter 2 describes what ad hoc networks are and provides a classification of their use cases depending on the existence or the state of a network infrastructure. The modeling of such networks is introduced at the end of the chapter.

Chapter 3 presents topology management techniques such as topology control or virtual backbones. Differences between the two approaches are detailed and a taxonomy of the virtual backbone techniques is provided. A state-of-the-art on the metrics to quantify the quality of virtual backbones is introduced along with new measures designed for $CDS-$based virtual backbones in mobile ad hoc networks.

Chapter 4 provides a taxonomy of the $CDS-$based virtual backbone techniques. Centralized, distributed and robust contributions are explained, re-grouped and compared.

Chapter 5 introduces all the theoretical aspects of the proposed algorithms. A centralized scheme based on DC programming and DCA is developed along with its theoretical properties. Two distributed algorithms and some optimizations are detailed in the second part of this chapter.

Chapter 6 presents the results of experimental evaluation of the different proposed algorithms.

Chapter 7 summarizes the dissertation by presenting conclusions of the work. Next, future work and perspectives of the work are outlined.

Chapter 2

Ad hoc networks

Contents

2.1 Presentation of ad hoc networks **8**
- 2.1.1 Definitions 8
- 2.1.2 A comparison with real world communication 9
- 2.1.3 Characteristics and limitations 10

2.2 Use cases **13**
- 2.2.1 No infrastructure 13
- 2.2.2 Damaged infrastructure 13
- 2.2.3 Cost of infrastructure 13

2.3 Modeling ad hoc networks **15**
- 2.3.1 Wireless channel 15
- 2.3.2 Communication graph 16
- 2.3.3 Mobility model 19

2.4 Recap of key points **21**

This chapter introduces all the prerequisite notions and definitions concerning ad hoc networks and how graph theory can be used to model such networks.

In section 2.1, some insights are provided concerning the term *ad hoc* and what researchers mean when applying it to communication networks. A comparison with real world communication is presented in a second part as a way to illustrate the main differences between ad hoc and classical networking techniques. The last part of the ad hoc networks presentation summarizes the major characteristics of this class of networks and their associated limitations are detailed.

Section 2.2 proposes a classification of the most important use cases for ad hoc networking technologies depending on the status or the availability of an underlying infrastructure. Examples of concrete situations where ad hoc networks are already used are presented along with promising application ideas.

To conclude this chapter, section 2.3, presents different aspects concerning the modeling of ad hoc networks. In a first part, wireless channel characteristics are presented and commonly used models are introduced. The communication graph and its extension for dynamic contexts are then defined and extensively discussed. A classification of the popular mobility models concludes the modeling section.

2.1 Presentation of ad hoc networks

This section is dedicated to a simple yet essential question: “What are ad hoc networks?” In a first part we review the latin definition of the term *ad hoc* and what is meant when applying such a term to network communications. A comparison human interactions is presented in a second part as a way to illustrate the main differences between ad hoc and classical networking techniques. The last part of the ad hoc networks presentation summarizes the major characteristics of this class of networks and their associated limitations are detailed.

2.1.1 Definitions

Before considering what people generally include in the ad hoc networking domain, let us start with the etymological background. In the Cambridge Dictionary of American English, *ad hoc* is defined as an adverb or an adjective that means:

> **Definition 1** (Ad hoc - *Cambridge Dictionary of American English*).
> *for a particular purpose or need, especially for an immediate need.*

The term *ad hoc* has been used since the mid 16th century in the latin language and is nowadays used by scientists to qualify a particular class of networks.

Classical networks are characterized by a pre-existing infrastructure in charge of supporting basic services (e.g. routing) for connected machines. They are generally organized using hierarchical structures which are mostly fixed through time: reorganization of the hierarchy is rare. Almost everyone is using such a type of network as it is the most commonly used paradigm. For example, accessing the Google webpage on your home desktop computer is possible thanks to a series of intermediate relays (e.g. routers, gateway) which are dedicated to the routing of network packets from a source to a destination. With the help of this pre-existing infrastructure, very distant entities can communicate without even knowing the exact path used for the transmissions.

As an opposite paradigm, ad hoc networks are composed of different interconnected devices able to communicate without any preinstalled infrastructure. These networks are created for an *immediate need*, i.e. communication needs, and are only composed of the different available peripherals and usable communication channels between them.

A quite old and -from our point of view- incomplete definition of ad hoc network can be found in the IEEE802.11 specification (i.e. Wi-Fi). It states that *an ad hoc network is a network composed solely of stations within mutual communication range of each other via the wireless medium.*

This definition implies that a network can be seen as a complete graph in the sense that all pairs of nodes of the network can communicate with each other directly as they are in each other communication range. Moreover, as this definition can be found in the IEEE802.11 specification, it implies that communications are handled in a wireless manner. Stojmenovic and Wu [76] provide a definition which encompasses many important additional aspects of wireless ad hoc networking such as multi-hop communications or the intrinsic dynamic nature of the network.

Based on this last definition, distant nodes (i.e. that do not have any direct communication possibility) have to rely on cooperative intermediate neighbors to relay messages. These relay nodes are of course not dedicated to transmit information but are instead independent components of the network and as such they may also be the source or the destination in another communication session. To put it in a nutshell, all the peripherals in an ad hoc network have to behave like routers.

Obtaining such cooperative behaviors requires that all nodes organize themselves. Such process is generally referred as self-organization and can be found in the definition of Gerla [43], in which it emphasized that there is no pre-existing infrastructure in ad hoc network.

We now propose our own synthetic definition of *ad hoc network* that combines the major aspects of the previous definitions:

Definition 2 (Ad hoc network - *Synthetic definition*)**.**
An ad hoc network is a collection of interconnected wireless devices. Nodes can be mobile or static and they cannot rely on any pre-existing infrastructure to relay communication. As such, these networks are characterized by a self-organization process in order to support essential network services.

This synthetic and general definition encompasses a variety of more specific types of networks that may be found in the literature. As an example, the term *mobile ad hoc network* is generally used when the mobility aspect is the most important. Another important sub-class consider small and generally homogeneous networking components with sensing capacities and are referred as ad hoc sensor networks. In such networks mobility is assumed to be limited if any and generally the battery management aspect is one of the most important design issue.

2.1.2 A comparison with real world communication

In this section, we propose a comparison between telecommunication networks and a real company use case. Let Anne be a financial consultant of the Audit & and Sons company.

Anne needs a report to be translated into French before sending it to the client. This task should be taken care of by Caroline, a translator within the same company. Anne and Caroline are not in the same subdivision and as such, they depend on different managers, Bob and David respectively. The complete organization chart of the company is illustrated in Fig. 2.1.

Using classical network style communication would be for Anne to write a translation demand to Bob. Bob would have relayed her request to David, Caroline's manager. Finally, Caroline would have received a new translation assignment from David, and when it would have been completed the opposite path would have been used to send the translated report.

For the sake of the ad hoc counterpart, let us imagine that neither David nor Bob are available: an urgent project meeting is on. Using ad hoc networking style communication could have been done thanks to Eliot, Anne's colleague, who likes to be helpful even if he does not always know how. Eliot accepted to deliver Anne's request to Caroline because he wanted to be nice with her, however Eliot never heard of this Caroline. The call of duty made him go to the translation department to discuss with his old friend Fay, a colleague of Caroline, who does not bother transmitting the request.

As we can observe from this example, the main differences are:

- The communication path is not pre-established in the ad hoc network paradigm (Eliot and Fay had not agreement to transmit Anne's request to Caroline, or even from the financial to the translation department)
- Cooperation is required as no dedicated relay are available (If Eliot would have not been helpful, Anne could not have transmitted her request)
- In ad hoc networks, the components are likely to move (Eliot went to the translation department)
- Ad hoc networks are complementary to classical networks, as they permit communications when the infrastructure is not available (e.g. temporary malfunction of relay nodes, too costly to be built)

2.1.3 Characteristics and limitations

The intrinsic characteristics of ad hoc networks induce limitations that explain why it is such a popular research topic. Wireless communications, self-organization, energy conservation, resource-constrained computation or even scalability are very demanding characteristics that generate a wide panel of problems. Consequently, solutions should be proposed to overcome those limitations so that end users may benefit from the ad hoc possibilities.

Let us now describe in details what the main characteristics of the ad hoc networking paradigm are.

2.1.3.1 Wireless communication

Wireless communication channels are generally less reliable than their wired counterpart. Indeed, the quality of the communication channels can be affected by diverse factors such

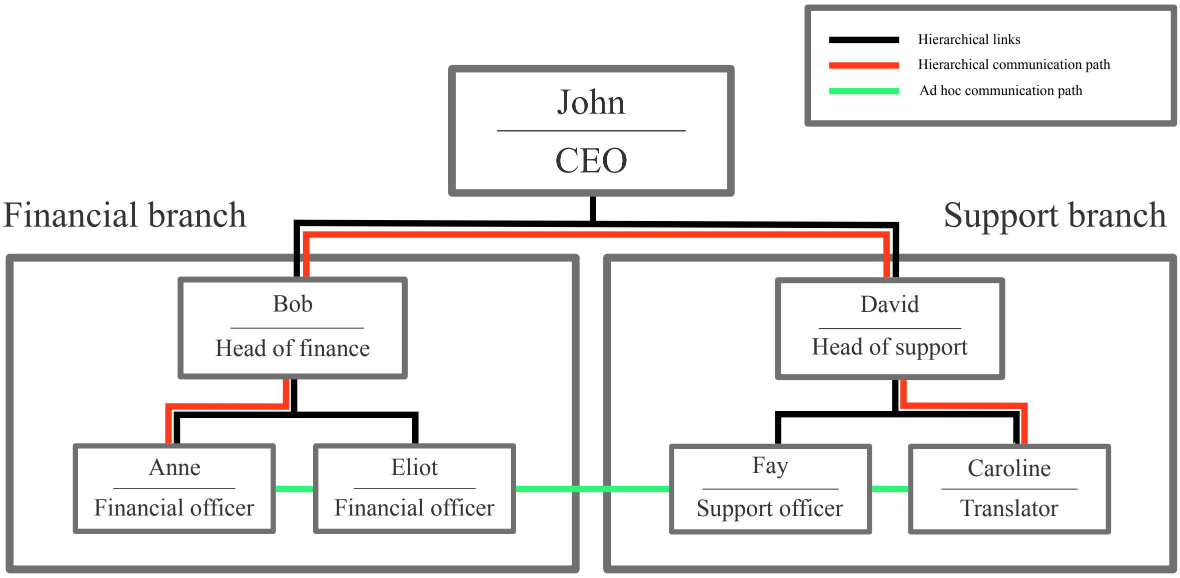


Figure 2.1: Organization chart of the Audit & Sons company

as atmospheric conditions, obstacles between the transmitter and the receiver or even interferences due to surrounding devices. Moreover, the most popular wireless techniques (e.g. Wi-Fi, Bluetooth) are broadcast-based, i.e. if a node A sends a message then all its direct neighbors shall receive it. This property induces two important facts. First, the communication medium is shared and as such, the density of nodes may greatly impact the available bandwidth as stated in [47]. Second, even for one-hop (direct) communications between two nodes A and B, security issues may exist as all nodes in the vicinity of nodes A or B can receive parts of the communication session. This confidentiality issue can be extended to the even more complex case of multi-hop communication that implies relay nodes. In that particular case, a new problem arises: how to be sure that a relayed message has not been altered?

2.1.3.2 Unstructured and/or time-varying network topology

As ad hoc network are intrinsically infrastructureless, having a reliable routing protocol (i.e. a protocol in charge of delivering packets from a source to a destination) is a very difficult problem. Moreover, as the network topology may change through time (i.e. nodes may move, appear or disappear), obtaining up-to-date routing information is even more complicated. To cope with these problems, it is often assumed that there exists a self-organization process which helps the node to efficiently relay packets. Many different proposals have been studied in order to leverage the limitations induced by this characteristics: routing algorithms, topology control techniques, clustering techniques, spanning trees, connected dominating sets, . . .

2.1.3.3 Energy conservation

In ad hoc networks, the components of the network are generally portable device powered by a battery. As a consequence, one of the main design objective is energy efficiency in order to increase the network lifetime (see definition 3).

Definition 3 (Network lifetime)**.**
The network lifetime is the span from the deployment from the instant the network is considered nonfunctional. When a network should be considered nonfunctional is, however, application specific.

Table 2.1 shows a wireless transceiver requires much more power to transmit or receive packets than when being idle. Moreover, wireless transceivers can represent up to 50% of the total power used on portable devices. As such, the design of protocols for ad hoc networks should focus on message efficiency, i.e. the number of messages which is required by the protocol.

Technology	Power Idle (mA)	Power Tx (mA)	Power Rx (mA)
802.11 a	203	554	318
802.11 b	203	539	327
802.11 g	203	530	282

Table 2.1: Nominal power consumption of the CISCO IEEE 802.11 a/b/g wireless card. Power consumption is measured by the drained current, expressed in mA.

2.1.3.4 Resource-constrained computation

Ad hoc networks are generally composed of wireless peripherals with limited computation. As such, protocols and applications designed for these types have to attain efficiency requirements with the few available resources.

2.1.3.5 Scalability

The size of an ad hoc network (number of nodes, maximum number of hops between two nodes) is generally unpredictable and protocols designed for such networks should be able to cope with very different topologies. This requirement is even more difficult to obtain if we consider the energy conservation and the resource-constrained computation characteristics that we previously detailed.

As stated in definition 4, the scalability is the ability to handle growing amounts of work. From the ad hoc network point of view, two main criteria may induce heavier work load: the increase in the node density and the enlargement of the area covered by the network. The first implies more communication channels and thus even more requirements considering the bandwidth usage and sharing. A wider area induces more intermediary nodes to communicate and as a consequence, the average hop count is mechanically increased.

Definition 4 (Scalability - *Bondi et al.*)**.**
Scalability is a desirable property of a system, a network, or a process, which indicates its ability to either handle growing amounts of work in a graceful manner or to be readily enlarged [20].

2.2 Use cases

From our point of view, useful applications of the ad hoc networking technologies arise when classical infrastructure are not available or too costly to deploy. We propose a classification of interesting use cases structured around the presence or status of a prior infrastructure network.

2.2.1 No infrastructure

Ad hoc networks where investigated at first for military purposes. Indeed, battlefields are generally a wide, hostile, always moving area in which communications are vital in order to reduce human losses. Against this background, optimizing the coordination of moving units is an important issue that may be overcome with efficient communications. Ad hoc networks are particularly fitted for such demanding situations for two main reasons. First, it is very unlikely that a regular communication infrastructure is available before the arrival of the troops on the battlefield. The spontaneity and the adaptability of ad hoc networks are in this situation important characteristics to quickly deploy a usable communication network.

The second reason lies in the multi-hop capabilities of ad hoc networks. Indeed, this key feature helps overcoming the line-of-sight problems that generally occurs with regular radio telecommunications. As such, it increases the robustness of the communication network, one of the major design issues for military purposes.

2.2.2 Damaged infrastructure

Another important use case encompasses all the scenarios in which a previous infrastructure is damaged. This is the case of search & rescue operations in hostile environmental conditions such as earthquake, volcano eruption, and so on. In such harsh conditions, prior infrastructure may be partially or completely damaged which causes coordination difficulties (e.g. firefighters requesting a medical assistance). With current available technology, coordination rescuers' efforts when the fixed communication infrastructure is severely damaged is very difficult, even with walkie-talkies. Thus, one of the priorities in present-day disaster management is to reinstall the communication infrastructure by repairing the previous structures and by deploying temporary communication equipment. In that type of context, ad hoc networks are well suited to replace or complement the damaged infrastructure as it would save more time to effectively rescue people.

A particular class of ad hoc networks often referred as sensors networks may also be useful to avoid casualties on the rescue team side by gathering specific and detailed information about the environment. Practical examples for such examples embrace all kinds of sensing possibilities (e.g. temperature, pressure, gases, humidity. . .) and may prevent rescuers from being exposed to dangerous places of the environment.

2.2.3 Cost of infrastructure

Our last category of ad hoc network usage concerns communication requirements when building an infrastructure would be too costly. Ad hoc networks are a good solution to cover wide areas without having to rely on an infrastructure. Practical use cases are legion and can be roughly classified in two categories: zone monitoring and extension of an infrastructure network.

Zone monitoring is generally performed by sensor network, i.e. ad hoc networks composed of a high number of small nodes (from dozens to thousands) that are dedicated to simple tasks. Such nodes are often highly constrained by their computational capabilities and generally powered by a battery. These two characteristics are a general requirement to reduce the cost and increase the number of deployed units. Classical applications are numerous: seismic sensor networks, fire detection for vast forest zones, temperature sensors in oceans to increase the precision of weather predictions.

Ad hoc networks may also complement an existing infrastructure network by increasing the size of the covered area. This is particularly useful to broadcast local information to a lot of wireless nodes when infrastructure access points are sparsely covering a zone. Car networks are also an important field of research studied by universities and private companies. One direct application of such networks is increasing the security by detecting car accidents and propagating alert message in order to prevent the increase of casualties. Local traffic information is another promising application of such networks. Indeed, by coupling with GPS information, it is possible to detect traffic jam appearance (resp. disappearance) and transmit this information to other cars. For even more efficiency, information could be relayed by cars to a roadside network component connected to a global infrastructure in charge of aggregating all data and update the traffic information broadcast system.

2.3 Modeling ad hoc networks

Modelling wireless ad hoc networks is a vast domain that can be partitioned in different sub-domains such as the modelling of the wireless channel, the communication graph or the mobility model. In this section we provide a quick overview of these different domains along with their most important and well-known approaches.

2.3.1 Wireless channel

Nodes in ad hoc networks communicate through wireless transceivers which is the reason why the wireless channel model is an important part of the ad hoc network model. A radio channel between a transmitter unit u and a receiver unit v is established if and only if the power of the radio signal received by node v, P_r is above a sensitivity threshold β. The received power P_r depends on the power used by u to transmit P_t and on the path loss, which models the radio signal degradation with distance. Let $PL(u, v)$ be the path loss between units u and v.

$$P_r = \frac{P_t}{PL(u, v)} \tag{2.1}$$

Modeling the path loss in a realistic way is a very difficult task. The path loss model is one of the most important parameter in the simulator as it has a huge impact on both the realism of simulation and the simulation time. Depending on the path loss, simulators determine if two nodes can communicate, as such, results of a single simulation can differ greatly depending on the considered path loss model. Moreover, as the simulator is checking the availability of a usable communication channel, path loss calculations will be frequently repeated. As a matter of fact, if the path loss model requires heavy computations, the simulation time will increase accordingly. As more precise model are generally more complex too compute, a trade-off between realism and simulation has to be determined.

Path loss models have many physical phenomenon to take into account, such as:

- Reflection, when the radio signal hits the surface of large objects compared to its wavelength (e.g. the radio signal can be reflected by large buildings and wall)
- Diffraction, when the radio signal encounters very sharp edges
- Scattering, when several small objects are between the transmitter and the receiver (e.g. foliage, street signs)

Many contributions have been proposed through time, however some models are more popular than others. Here is a list of propagation models that are generally implemented in wireless network simulators:

- The free space propagation model [53], in which the received power falloff is proportional to the square of the distance. Line-of-sight has to be clear and only the direct path between the transmitter and the receiver is taken into account.
- The two-ray ground model [71] increases the realism by considering two transmission path: direct and via ground reflection. In this case the received power is proportional to the distance raised to the fourth power.

- The log-distance path model [71] is a generalization of the two previous approaches. It is based on a exponent α that depends on the environmental conditions.

2.3.2 Communication graph

A popular abstraction representing ad hoc network is referred to *communication graph*. Such graph defines the network topology, that is, the set of wireless links that the nodes can use to communicate with each other.

In a first time we present all the necessary notions and definitions to model a static network with static graphs and we then provide some insights about how to deal with mobile networks in the second part.

2.3.2.1 Static graphs

In order to define a communication graph in a static context, we have to define two mandatory components: a network and a range assignment. Such approach can be found in [73]. A definition of the general communication graph is then provided along with the presentation of a widely used sub-class of graphs: the Unit Disk Graphs.

Network.

A network is defined by a set of nodes N evolving in a bounded region R. For simplicity sake, it is generally assumed that R is a d-dimensional cube defined as follow: $R = [0, l]^d$ with l the side of the cube and $d = 1, 2, 3$. The position of the nodes N is provided by a location function $L : N \to R$ that maps every node $n \in N$ to its physical location $L(n)$.

We can now define a network M as follow:

Definition 5 (Network).
A network M is a couple $M = (N, L)$ composed of a set of nodes N and a location function L associating each element of $n \in N$ to its physical location

Range assignment.

From the graph theory point of view, we have defined a vertex set. We now introduce the set of directed edges between those vertices thanks to a range assignment RA. Definition 6 states that each node is characterized by its transmission range, i.e. the maximal distance that allows correct reception of the transmitted data. As the bounded region R is a d-dimensional space with $d = 1, 2, 3$, the sub-region associated to a particular node n and constrained by its transmission range r_n is defined as follow:

- If $d = 1$, the sub-region of R is a segment of length $2r_n$
- If $d = 2$, the sub-region of R is a circle of radius r_n and centered at $L(n)$
- If $d = 3$, the sub-region of R is a sphere of radius r_n and centered at $L(n)$

Definition 6 (Range assignment).
A range assignment RA for a network M is a function that assigns to every element $n \in N$ a value $RA(n) \in [0, r_{max}]$ representing its transmission range. Parameter r_{max} is called the maximum transmission range.

Note that transmission range values are generally obtained by considering a particular propagation model (see 2.3.1). The log-distance path model is generally chosen for realism and computation sake for simulations. In that case, any transmission range $r \in [0, r_{max}]$ is uniquely associated to with a transmit power $p \in [0, P_{max}]$ and thus the two notions can be interchangeably used.

We can now define the communication graph as follow:

Definition 7 (Communication graph $G(N, E)$).
A communication graph $G = (N, E)$ is a directed graph composed of a set of vertices, resp. directed edges, representing the communication nodes, resp. the available directional communication channel.
N is the set of nodes of a given network $M = (N, L)$. A directed edge $(u, v) \in E$ if and only if $RA(u) \geq \delta(L(u), L(v))$, with $\delta(L(u), L(v))$ the Euclidean distance between the location of nodes u and v.

Unit disk graph.

Definition 7 is characterized by directional links, and thus embeds the undirected case. It is however generally assumed that the graph is composed of undirected edges, as popular wireless technologies such as Wi-Fi require bidirectional wireless links. A very popular sub-class of communication graphs, the Unit Disk Graphs (UDG), have been introduced by Huson & Sen in [54] to such extend. In such graphs, an homogeneous and standardized range assignment is assumed, i.e. all wireless nodes are characterized by a transmission range of value 1. As a matter of fact, an edge between two nodes u and v exists if and only if the Euclidean distance between these two nodes is less or equal to 1. Figure 2.2 illustrates the notion of unit disk graph with a simple network composed of two nodes A and B represented by gray circles. The dashed circles represent the delimitation of their respective transmission range. The gray stroke connecting the two gray rounds stands for the available wireless communication channel as both nodes are in each other transmission range. Figure 2.3 provides a geometric demonstration concerning the maximum number of independent neighbors in a UDG. It is NP-hard to determine whether a graph can be represented as a unit disk graph [22]. However, many important and difficult graph optimization problems such as maximum independent set, graph coloring, and minimum dominating set can be approximated efficiently by using the geometric structure of these graphs. Definition 8 proposes a graph theory point of view of the Unit Disk Graphs.

Definition 8 (Unit disk graph).
A Unit Disk Graph $G = (N, E)$ is a graph formed from a collection of equal-radius circles, in which two circles are connected by an edge if one circle contains the center of the other circle.

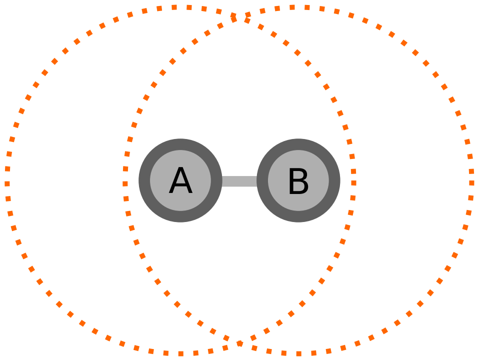


Figure 2.2: A simple Unit Disk Graph with two network nodes.

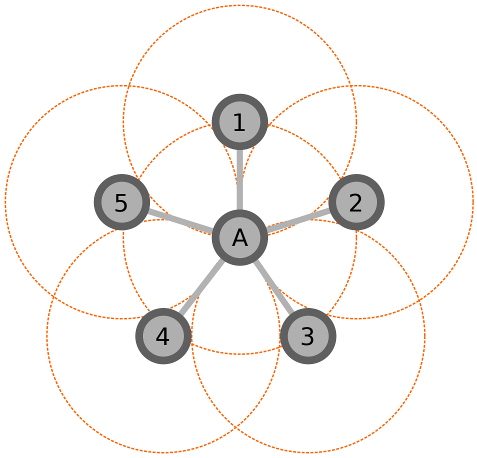


Figure 2.3: In a Unit Disk Graph, a node can have a maximum of five independent neighbors.

2.3.2.2 Dynamic graphs

Ad hoc networks use cases induces unsteadiness. Many factors may change the network topology through time:

- **Nodes mobility** induces that at some point, existing wireless channels are no longer available because the two devices are not anymore in each other transmission range. Obviously the opposite scenario may also happen: new communication channels may be created as nodes are getting closer to each other.

- **New node arrival** is a straightforward event when dealing with ad hoc networks as they are created spontaneously for some specific goals as underlies the term *ad hoc*.

- **Node disappearance** may happen as a consequence of mobility, the node moves out of the simulation area, or some power-related issue, lack of battery or device switched-off by the user.

Taking into account such changes in the communication graph model has been and still is a field of investigation. In a first time, we develop the necessary modifications to adapt the communication graph model presented in 2.3.2.1. The second part presents different proposition to handle dynamic graphs.

Adding the time dimension.

Nodes mobility can be easily handled by modifying the location function L. Indeed, the location can now be defined as $L_{dyn} : N \times T \rightarrow R$, and thus, for any node of $n \in N$ and any time $t \in T$, L_{dyn} provides coordinates in R representing the position of a node n at a time t.

To cope with node appearance or disappearance, the vertex set has to change through time. As such, we define N_t, the set of considered nodes at a time t. Definition 9 combines both aspect in the *dynamic network* notion.

Definition 9 (Dynamic network).
A dynamic network M_{dyn} is a sequence of static network at different time t. Moreover, M_t at a time t is a couple $M_t = (N_t, L_{dyn})$ composed of the set of vertices at time t and a location function L_{dyn} providing the position of any node $n \in N_t$ at any time t.

Moreover, topology control techniques are mainly based on modifying the transmission range of nodes through time in order to preserve some graph properties (e.g. connectivity, number of hops, ...). As a consequence, a dynamic range assignment is defined as: $RA_{dyn} : N_t \times T \rightarrow [0, r_{max}]$. A dynamic communication graph is defined as follow:

Definition 10 (Dynamic communication graph).
Given a network $M_t = (N_t, L_{dyn})$ and a dynamic range assignment RA_{dyn}, a dynamic communication graph is a sequence of static communication graphs for different values of time t. A static communication graph at time t is defined as $G_t = (N_t, E_t)$. An edge (u, v) at time t exists in E_t if and only if $RA_t(u) \geq \delta(L_{dyn}(u, t), L_{dyn}(v, t))$.

Other dynamic graphs models.

Dynamic graphs modelling has been a prolific research domain. In the thesis of Yoann Pigné [66], an extensive classification of such models is proposed.

2.3.3 Mobility model

Until now, all the dynamic graph models we introduced have had some dynamic capabilities, i.e. it is possible to add or remove vertices or edges. However, none of them explicitly detailed how the graph is actually changing. This sub-domain of ad hoc network modeling is generally referred as mobility modeling and consists in designing the movements patterns of the nodes in the simulation area. The mobility model is a major element of the simulation process as it greatly influences the realism of the simulation which is why many contributions have been proposed during the last decade. We propose a classification in two main classes: synthetic and trace-based models.

2.3.3.1 Synthetic models.

This class of models try to mimic real world mobility with very simple behaviours. The Random Waypoint model [77] is one of the most used mobility models as it is implemented in most network simulation and its application requires very few resources. In this model, each node chooses uniformly a destination at random and moves toward it along a straight line. Velocity of each node is also chosen uniformly at random in a predefined interval $[v_{min}, v_{max}]$. When a node arrives at destination, it remains stationary during a predefined pause period and then restart moving with the same patterns. This mobility model is well-known for its lack of realism closely related to the memory-free implementation of the nodes movement. To overcome this problem, the Gauss-Markov mobility model has been proposed in [23]. In this model, the velocity of a node at a time t depends on its previous speed at time $t-1$ and on an average speed. The Smooth Random Mobility Model [15] proposes realistic enhancement by adding an incremental change for velocity and direction. All these models are intended for free space environment which is why the Pathway Mobility Model [82] or the Obstacle Mobility Model [55] have been proposed. In the first one, the nodes are constrained to move on predefined path to arrive at a destination and in the second one polygonal obstacles are added to the simulation space.

2.3.3.2 Trace-based models.

Trace-based models are the exact opposite of the synthetic models as they are only replaying position or GPS traces in the simulation space. The main advantage of such approach is the realism as the data are collected on real devices (e.g. pedestrian, cars). Moreover, there is almost no required computation to use such model, which is an important criterion for the simulation of large networks. However, the main problem of this model lies in the small quantity of available traces and their representativeness. Indeed, the only way to simulate more scenarios is to collect more data, which is not always easy to obtain. Moreover, it is difficult to determine if all the possible cases have been tested, even if your trace database contains different scenarios.

2.4 Recap of key points

- An ad hoc network is a collection of interconnected wireless devices. Nodes can be mobile or static and they cannot rely on any pre-existing infrastructure to relay communication. As such, these networks are characterized by a self-organization process in order to support essential network services.
- Wireless communication channels are generally less reliable (obstacles, interferences) and offer less capacity (broadcast-based communications) than their wired counterpart.
- The lack of infrastructure implies that cooperation among nodes is mandatory to permit multi-hop communications.
- The mobility of the nodes may induce changes in the topology of the networks.
- Communication protocols design should take into account the scarce resources of the networks nodes (battery-based devices, few computational power) and of the communication channel (small number of protocol messages).
- Ad hoc network use cases embraces various situations in which classical network infrastructure are not existing, too costly to build or damaged.
- Modeling ad hoc networks is a wide domain composed of the wireless channel, the communication graph and the mobility models.

Chapter 3

Topology management

Contents

3.1 Motivation **24**
3.1.1 Scalability problems 24
3.1.2 Two different approaches 25
3.2 Virtual backbones **30**
3.2.1 Definition 30
3.2.2 Taxonomy 31
3.3 Metrics for virtual backbones **37**
3.3.1 Motivations and related criteria 37
3.3.2 State-of-the-art 39
3.3.3 Contributions for $CDS-$based virtual backbone 40
3.4 Recap of key points **44**

This chapter is dedicated to topology management techniques for ad hoc networks. In 3.1, the main problems induced by the usage of a shared medium and an always changing topology are presented. A presentation of the different topology management techniques (topology control and virtual backbones) is provided in a second time.

In section 3.2, we introduce a classification of the different virtual backbone techniques. In a first time we define what are virtual backbones and what are their most suitable characteristics. A classification of the most widely used techniques, i.e. trees, clusters and connected dominating sets, is developed in a second time and state of the art solutions are briefly presented.

To conclude, section 3.3 provides some insights concerning the quality measurement of virtual backbones. A review of the most used metrics for the different virtual backbone techniques is developed and some metrics are proposed for the connected dominating set approach.

3.1 Motivation

Topology management techniques have been extensively studied since the late 90s as they try to leverage the encountered scalability issues in ad hoc networks. In a first time we present the main problems induced by the usage of a shared medium and an ever changing topology. Then, we propose a classification of the techniques that have been developed so far to settle such issues.

3.1.1 Scalability problems

Mobile ad hoc networks face scalability problems (see 2.1.3.5 and definition 4) due to their wireless communications (details provided in 2.1.3.1) and their ever-changing topology (details in 2.1.3.2). In a first part, we provide details concerning the impact of the node density on the available bandwidth. These theoretical results have been obtained and published by Gupta *et al.* [47] in 2001. The second part introduces the broadcast storm problem, a network layer issue that has to be overcome to fully benefit of the ad hoc networking possibilities.

3.1.1.1 Density and available bandwidth

In wireless ad hoc networks, the shared medium and the lack of global coordinator imply that the node throughput declines rapidly to zero as the number of nodes in the network increases [47]. In their work, Gupta and Kumar homogeneously placed n wireless nodes in a unit disk area. All the nodes have the same transmission range and can transmit W bit per second. Under such assumptions, Gupta proved that the throughput $\lambda(n)$ obtainable by each node for a randomly chosen destination is equal to:

$$\lambda(n) = \Theta\left(\frac{W}{\sqrt{nlogn}}\right) \tag{3.1}$$

Obviously, in Gupta and Kumar experiment, the increase in the number of nodes in the fixed unit disk area also induces an increase in the average density, i.e. the average number of neighbor for any node. As a consequence, Gupta formalized the fact that increasing the density of the network quickly reduces the available bandwidth for each node because of the rapidly growing collision and interferences probability.

3.1.1.2 Broadcast storm problem

Gupta and Kumar proved formula 3.1 by abstracting problems due to the decentralized nature of ad hoc networks. In particular, packet routing and broadcasting issues. Broadcasting is a process designed to deliver a given piece of information to a wide area. It is frequently used to solve network layer problems. Mobile ad hoc networks, due to their rapidly changing topology, require even more broadcast packets for multiple usage such as sending alarm signals, routing table updates, . . .

One famous routing problem, previously existing in classical network architectures, arises when communication nodes endlessly broadcast a packet. This non-functional behavior happens when broadcast processes induce the flooding of the considered network which in turn cannot support the others communication session. This problem can be caused by:

- Cyclic routing paths if the broadcasted packets are never ignored or destroyed. The particular case has been overcome in classical networks with spanning tree based routing solution.

- In ad hoc networks, inefficient broadcasting scheme may lead to heavy contention. In such a scenario, many nodes may relay broadcast packets and thus creates redundant packets. This problem may arise even more often in high density networks as the available bandwidth per node is smaller (see 3.1.1.1).

3.1.2 Two different approaches

To leverage such scalability issues, many contributions have been proposed since the end of the 90s. From our point of view, all these propositions can be classified into two main categories: topology control methods and topology management techniques, also known as virtual backbones.

3.1.2.1 Topology control

Topology control methods aims at designing the network topology, i.e. changing the actual shape of the communication graph. Definition 11 proposed by Santi [73], clearly states that topology control is about tuning the range assignment in order to optimize either the energy consumption or the network capacity.

Definition 11 (Topology Control - *Santi*)**.**
Topology control is the art of coordinating nodes' decisions regarding their transmitting ranges, in order to generate a network with the desired properties (e.g. connectivity) while reducing node energy consumption and/or increasing network capacity.

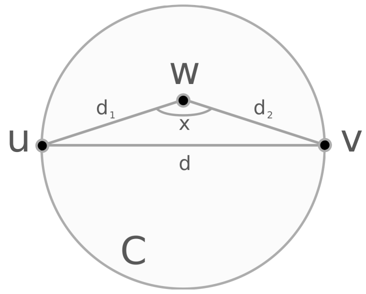


Figure 3.1: Node u needs to send a packet to node v. To achieve its goal in a energy efficient fashion, u has to choose between direct and multi-hops communication.

Energy consumption.

Most ad hoc network use cases are characterized by battery-powered devices. Moreover, wireless reception and transmission is one of the most energy-consuming process for devices such as sensors, mobile phones and PDAs. Optimizing the energy consumption is often referred as increasing the network lifetime, i.e. the time at which the network is not functional anymore. Figure 3.1 illustrates what kind of choices can be done to optimize energy usage. A node u can properly transmit packets to a node v if it emits at maximum power. A node w lying between u and v could be used as a relay for this communication session. In order to determine which of these two alternatives is better from an energy point of view, we need to choose a wireless channel model. A simple enough and yet comprehensive choice is the free space model. The minimum required power to send a message is then proportional to the square of the distance. From Figure 3.1, we can state that

$$d^2 = d_1^2 + d_2^2 - 2d_1^2 d_2^2 \cos(x) \tag{3.2}$$

Moreover, as $w \in C$, $\cos(x) \leq 0$. Thus $d^2 \geq d_1^2 + d_2^2$. It is now obvious that from an energy point of view, short range multi-hops communications are more efficient than the direct long range ones.

Network capacity.

Wireless networks are generally characterized by a shared-medium and as such, conflicting scenarios are more likely to appear. In Figure 3.2, node u is transmitting a packet to node v with a transmit power P. At the same time, node w is sending a packet for node z with the exact same transmit power P. As the distance between u and v, $d(u, v) > d(w, v)$, the interfering signal received by v (from w) has a higher power than the intended transmission from u and thus, the packet from u is corrupted. In order to solve such a problem, one may want to raise the power level used by u so that the interference level is not a problem anymore. However, increasing the power level of node u may lead to worsen others communication session that may happen in node u's surroundings. In that context, topology control techniques aim at increasing the network capacity by reducing the global amount of interferences.

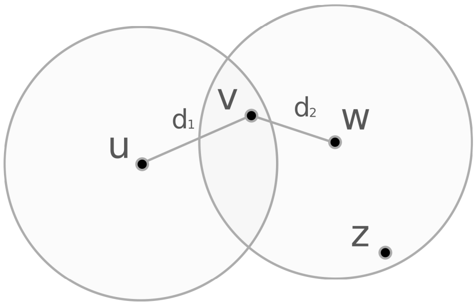


Figure 3.2: A conflicting scenario due to the wireless shared-medium. Circles represents the radio coverage area with transmit power P.

3.1.2.2 Virtual backbones

Topology control limitations.

In 3.1.2.1, we presented the two main objectives of topology control techniques: reducing energy consumption and increasing the network capacity by tweaking the range assignment and thus changing the communication graph. Although these methods do optimize some network characteristics, they may not solve all the existing problems. We agree that it is important to have energy efficient communications. However, as topology control methods tend to reduce the number of edges of the communication graph, the average number of hops for two nodes to communicate is naturally increasing. Such longer paths may not be suitable for quality of service (QoS) issues such as:

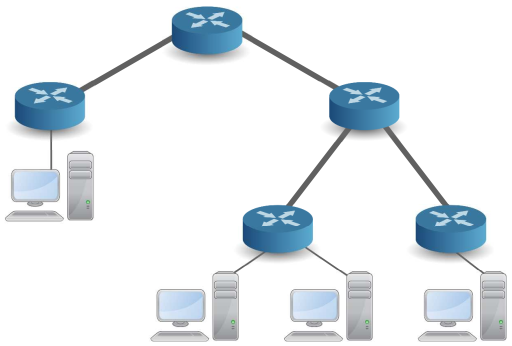
- Latency: the more intermediary nodes, the bigger the round-trip time, i.e. the time required to send a packet and receive an acknowledgement.
- Packet delivery success rate: as wireless nodes may be mobile, having long paths will induce a decrease in the percentage of correctly receive packets.

Reducing the number of edges like in topology control techniques is a practical solution to leverage the scalability issues that may arise from the routing protocols. It is however not practical to have an increase in the paths size. Virtual backbone methods propose to cope with such a drawback.

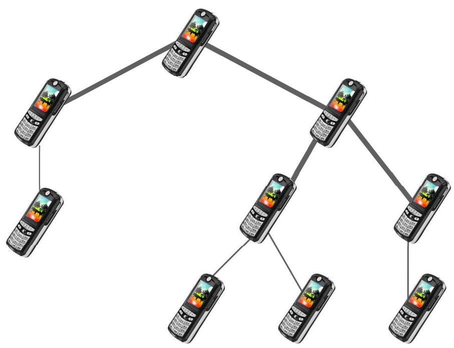
Another approach: virtual backbones.

Another approach to optimize ad hoc network communications is referred as virtual backbones. Its main difference with topology control techniques lies in the fact that the communication graph is not physically altered, i.e. the range assignment is not modified. A *virtual* structure is instead created to support the necessary network services and optimize the resource usage. That structure or backbone is said to be virtual as it is not the direct result of the physical dedicated network components like in the classical networks realm. Virtual backbones can be apprehended as a way to mimic the hierarchical organization scheme used in the regular networks in order to leverage the intrinsic scalability issues of the ad hoc networks. Figure 3.3 illustrates the virtual backbone approach as a direct transposition of the classical network hierarchical structure. On the left, a classical network organized thanks to an infrastructure composed of dedicated relays. On the right, an ad hoc network composed of cell phones. Thick lines represent the communication backbone for both illustrations.

The general purpose of virtual backbone methods is to select a subset of nodes and communication channels that will support some network services. The structure may evolve in order to adapt itself to the topology changes of the communication graph. Figure 3.4 illustrates a comparison between a topology control and a virtual backbone method. We can observe that if node A send a packet to node G, the path is longer with the TC method that the VB technique. Virtual backbone can be for example clusters, spanning trees or connected dominating sets. The two bold nodes of the VB method form here a minimum connected dominating set in charge of relaying packets for all the nodes of the network. Section 3.2 provides more details about these techniques.

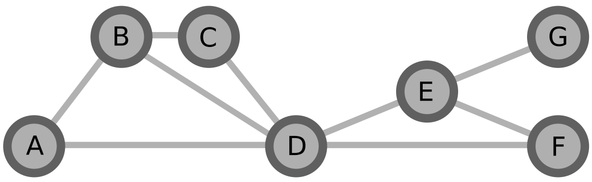


(a) Classical networks

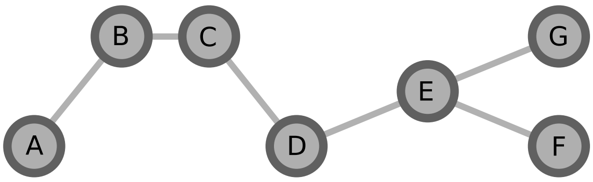


(b) Virtual backbone in an ad hoc network

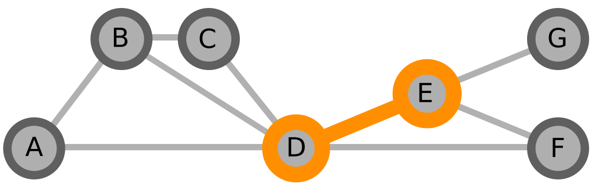
Figure 3.3: Classical networks backbone and ad hoc network virtual backbone.



(a) Original communication graph



(b) Topology control solution



(c) Virtual backbone solution

Figure 3.4: Topology control solution versus virtual backbone solution.

3.2 Virtual backbones

In this section we introduce a classification of the different virtual backbone techniques. In a first time we define what are virtual backbones and what are their main characteristics. A classification of the most widely used techniques, i.e. trees, clusters and connected dominating sets, is developed in a second part. Brief state-of-the-arts are provided and equivalency between each pair of techniques is discussed.

3.2.1 Definition

Ad hoc networks are intrinsically flat, i.e. there is no pre-existing hierarchy. In such a configuration, vital network services (e.g. routing, broadcasting) should be handled by all the nodes in a cooperative and organized manner. Classical networks rely on a hierarchical structure composed of dedicated equipments for relaying packets. In the literature, virtual backbones are referred as the direct transposition in the ad hoc realm of the classical networks well-organized structure.

Finding a definition of such structure is generally avoided by the authors. However, every one agrees on the suitable characteristics. In [61], Lin *et al.* proposed four problems that should be studied before designing a virtual backbone algorithm. Basagni [14] synthetically stated that a backbone should "*first and foremost be small. Additionally it should have other characteristics such as robustness to node failure low stretch, i.e. routes in the backbone should not be much longer than the shortest routes*". Let us discuss these three criteria:

- The set of nodes in the backbone should be the smallest possible for two main reasons. First, backbone nodes are in charge of relaying packets and thus are more likely to drain their battery. In that case, less backbone nodes means that less nodes are intensively used. The second reason is related to the purpose of creating such a structure. As virtual backbone are likely to be used by routing protocols, having less backbone nodes induce less protocol-related messages, such as routing table updates, and thus increase the available bandwidth for real communications.

- The backbone should be node-failure tolerant. This characteristic is important as loosing one node may result in a useless backbone. Many propositions have been made to increase the robustness of the structure: k-connectivity, i.e. having k independent paths between any pair of nodes, empirical criteria (and their combination) such as remaining battery level, low relative speed (stable surroundings), etc...

- A backbone should be characterized by a small stretch, i.e. the communication path using the virtual backbone should not be a lot longer than the shortest path in the communication graph. Indeed, a big stretch induce a substantial decrease in some quality of service (QoS) measures (round-trip time, percentage of successful delivery).

Many virtual backbone contributions can be found for ad hoc networks since the mid 90s. We propose to classify these techniques in three main categories, depending on the type of the resulting structure: spanning trees, clusters and connected dominating sets. As the background for these types of methods are different, their main objective may also differs. Indeed, spanning trees methods generally rely on the weight of edges in order to minimize the total weight of the structure. Clustering algorithms tends to elect clusterheads, i.e.

nodes best fitted to represent their surroundings (generally done by aggregating empirical criteria into a weight function). These methods are often compared by the number of elected clusterheads and their maintenance requirements, i.e. how often clusterheads should be updated in order to cope with the network topology changes. Connected dominating set approaches mainly consider the size of the backbone, but from two different points-of-view: analytical and experimental. Indeed, as these methods are directly inspired by the graph theory methodology, the algorithms are often compared considering their theoretical results by restraining the analysis to a particular class of graph: the unit disk graph (see 2.3.2.1).

As we have now delimited the domain that encompasses what we consider as the virtual backbone techniques, we propose two definitions with different points-of-view. The first definition summarize what is a virtual backbone and what characteristics are expected:

Definition 12 (VIRTUAL BACKBONE - OPERATIONAL POINT-OF-VIEW).
A virtual backbone is a subset of nodes and / or communication links in charge of providing services to the complete network.

The second definition formalizes what is a virtual backbone from the graph theory point-of-view:

Definition 13 (VIRTUAL BACKBONE - GRAPH THEORY DEFINITION).
A virtual backbone is a subgraph $G_{vb} = (V_{vb}, E_{vb})$ of the communication graph $G = (V, E)$ with V the set of wireless nodes and E the available communication channels. V_{vb} is the backbone nodes set and E_{vb} are the set of communication channel with both ends in V_{vb}.

This definition is wide enough to include all the previously introduced possibilities:

- If the solution is a connected dominating set $S \subset V$, $V_{vb} = S$ and $E_{vb} \neq \emptyset$. Moreover G_{vb} should be connected and $\forall v \in V \setminus V_{vb},\ \exists w \in V_{vb}/(v, w) \in E$.
- If the solution is a spanning tree, $V_{vb} = V$ and E_{vb} contains a selected set of edges. Moreover, G_{vb} is connected and contains no cycle.
- If this is the solution of a clustering method, V_{vb} represents the elected cluster-heads. E_{vb} can either be empty or contains the edges from a cluster-slave to its cluster-head. This definition induces that clusters are all disjoint.

3.2.2 Taxonomy

As previously explained in 3.2.1, we propose an overview of the virtual backbone realm by detailing three different techniques: spanning trees, clusters and connected dominating set. For each of them, we will explain their underlying ideas and concepts. Major contributions to each sub-domain are commented and the equivalency of each method is discussed.

3.2.2.1 Tree-based virtual backbone

In a first time we provide all the necessary definitions to properly explain what are tree-based virtual backbones. A short state-of-the-art about this type of structure is then developed and equivalencies with the other methods are tackled in the last part.

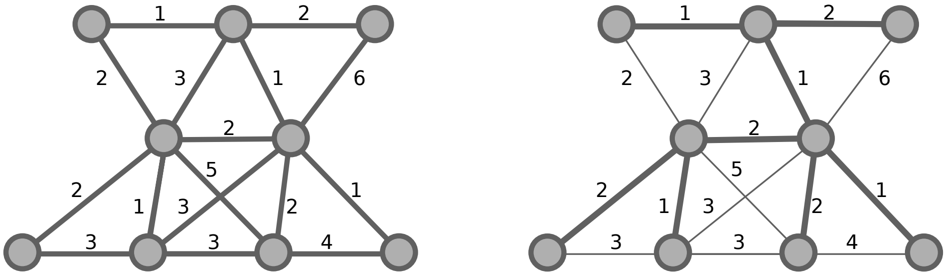


Figure 3.5: On the left the communication graph with edges values.

Definitions.

By looking at a communication graph, we can notice that only a subset of edges are required to cover all the nodes, i.e. some edges could be suppressed and the graph would still remain connected. Based on this idea, trees and more precisely spanning trees have been studied for ad hoc networks to support network services such as routing, multicast, broadcast or even security issues [67]. Definition 14 states that in graph theory, a tree is a connected structure without cycles. The definition of spanning trees (see definition 15) provides an additional characteristic to the structure: the coverage, i.e. the spanning tree T_G of a connected graph G is a connected cycle-free structure covering the whole vertex set of G. If G is not connected, i.e. G is composed of a set of connected components, then the cycle-free connected and covering structure is called spanning forest, i.e. one spanning tree per connected component of G.

Definition 14 (Tree).
A tree is an undirected simple graph T that satisfies the two following conditions: T is connected and has no cycle.

Definition 15 (Spanning tree).
A spanning tree T_G of a connected graph G can be defined as a maximal set of edges of G that contains no cycle, or as a minimal set of edges that connect all vertices of G.

Obtaining a spanning tree structure is really advantageous in ad hoc networks as it greatly reduces the routing protocol complexity. Indeed, as only a subset of edges are in the spanning tree, the routing possibilities are less numerous. Spanning tree techniques generally rely on some weights to compute the most suitable edges. Such weights can be placed on nodes and / or nodes depending on which scenario is considered. In such a context, the problem is generally renamed as minimum weight spanning tree and thus consists in finding a spanning tree whose cumulated weight is minimal. Figure 3.5 illustrates a possible solution for the minimum weight spanning tree for a graph with weighted edges. In this example, these values are arbitrary distributed and thus are not reflecting some pre-calculated quality.

However, if creating and / or maintaining a tree is not too difficult when the algorithm deal with the complete graph, it is much more complicated in a decentralized manner with local information. Ad hoc networks are indeed decentralized entities, i.e. there is no global coordinator. As such, nodes can take decisions based on their own needs and the state of their surroundings. Gathering the state of the complete network in one single node is not scalable as the nodes may be numerous and the topology may be constantly changing. This is the reason why localized algorithms, i.e. algorithms that rely only on local information, are preferred for ad hoc networks. Having a decentralized and localized algorithm is more suitable for scalability issues, however, in such context, guarantying the absence of cycle in the global communication graph is much more difficult.

State-of-the-art.

The first algorithm for finding a minimum spanning tree was developed by Czech scientist Otakar Borůvka in 1926 [64]. Its purpose was an efficient electrical coverage of Moravia, an historical region in Central Europe in the east of the Czech Republic. There are now two algorithms commonly used, Prim's algorithm and Kruskal's algorithm [69, 57]. All three are greedy algorithms that run in polynomial time. Another well-known algorithm to solve the same problem has been proposed by Tarjan [28] and is based on a depth-first search approach. Many parallel algorithms [48, 13] were also proposed to efficiently solve this problem.

Two main variants can be found in the literature: minimum weight spanning tree and maximum leaf spanning trees. The first problem considers that either nodes or edges (or both) are characterized by weights values. The objective is then to compute a valid spanning tree with the smallest aggregated weight, i.e. the sum of the spanning tree edges and / or nodes weight values. The second problem is dealing with finding a spanning tree with a maximum number of leaves.

The maximum leaf spanning tree problem requires to find the spanning trees containing the maximum number of leaves, i.e. vertices connected to only one vertex in the tree. Exact approaches to optimally solve this problem have been proposed in [37, 39].

The most common distributed algorithm is the Spanning Tree Protocol (RFC 1493), used by OSI link layer devices to create a spanning tree using the existing links as the source graph in order to avoid broadcast storms in classical networks.

An approach to guaranty cycle-free structures has been proposed by Casteigts *et al.* [26] and extensively developed since then by Piyatumrong *et al.* in [67, 68]. The main idea is to build a spanning tree by iteratively merging smaller trees. The cycle-free property is obtained thanks to the usage of a token: each tree possess a unique token which is regularly transferred from one node of the tree to another. Two trees are allowed to merge if and only if their respective token are located on neighbor nodes, i.e. the node a of tree T_a and the node b of tree T_b are neighbors and both have the token of their tree.

Equivalence.

The maximum leaf spanning tree problem is equivalent to finding the minimum connected dominating set [39]. Moreover, this problem has been proved to be NP-complete by a reduction from the dominating set problem [40]. Figure 3.6 illustrates this equivalence: to obtain the minimum dominating set of a graph, put all non-leaf nodes in the dominating set.

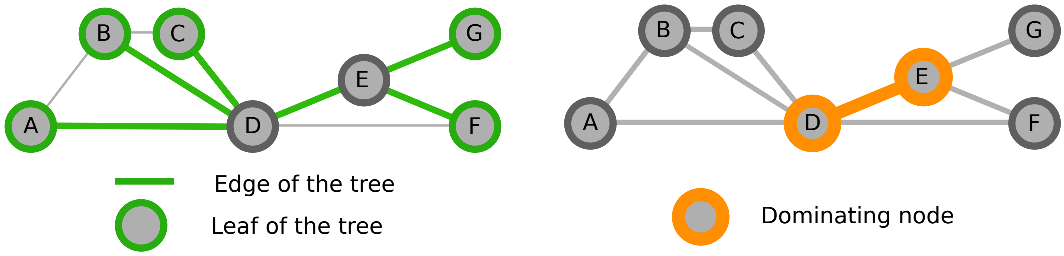


Figure 3.6: On the left, the maximum leaf spanning tree of a graph example. On the right the corresponding minimum connected dominating set.

3.2.2.2 Cluster-based virtual backbone

Clustering related notions are presented in the first paragraph and a state-of-the-art is provided in a second time. Equivalencies between the different presented techniques are then detailed at the end of this part.

Definition.

Clustering techniques for ad hoc networks are inspired by their data-mining counterpart. Definition 16 gives a general template for all clustering methods. Some methods, such as *k-mean* [63] or the *agglomerative hierarchical cluster* are well-known techniques of this domain. The ad hoc clustering techniques aim at regrouping nodes into clusters. A cluster is represented by one specific node of the cluster, the cluster-head, that is generally chosen as the most suited to help the other members of the cluster, the cluster-slaves. The objective of such a process is to create a partition of the whole communication graph such that all the nodes are participating to one cluster.

Definition 16 (CLUSTERING / CLUSTER ANALYSIS).
Clustering consists in the assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are similar in some sense.

State-of-the-art.

In statistics and machine learning, k-means clustering is a method of cluster analysis which aims at partitioning n observations into k clusters in which each observation belongs to the cluster with the nearest mean. This method requires a preset value for the number of clusters k, and thus the number of clusters has to be known a priori. This constraint is not well-adapted to the ever changing ad hoc networks as the global topology is rarely known by a global coordinator.

Hierarchical clustering creates a hierarchy of clusters which may be represented in a tree structure called a dendrogram. The root of the tree consists in a single cluster containing all observations, and the leaves correspond to individual observations. Algorithms for hierarchical clustering are generally either agglomerative, in which one starts at the leaves and successively merges clusters together; or divisive, in which one starts at the root and recursively splits the clusters. This technique allows an observer to vary the granularity,

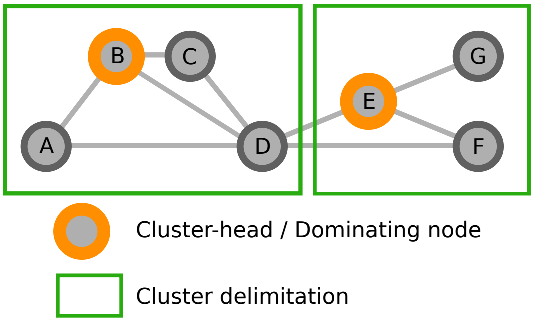


Figure 3.7: On the same graph, a clustering solution and a dominating set.

i.e. depending on the height in the tree, the number of clusters vary. However, such global method is not suitable in decentralized contexts.

In [62], a centralized cluster-head election algorithm is presented, where the base station assigns the cluster-head roles based on the energy level and the geographical position of the nodes. [46] provides a fuzzy logic based centralized algorithm. Nodes are selected as cluster-heads by a central coordinator based on their distances to each other, energy level, and the concentration of the nodes in the region.

Early approaches such as [78, 35] used an unique identifier (ID) per node that was used to compute its priority. All nodes locally exchanged these priority information within their two-hops neighborhood and then designated the cluster-head as the highest priority node. Chatterjee *et al.* [27] proposed a well-known clustering algorithm called *Weighted Clustering Algorithm* (WCA). WCA creates one-hop clusters with one cluster-head. A heuristic weight function combining the distances between the neighbors, the number of neighbors, the speed of the neighboring nodes and the remaining battery power of a node is proposed. Nodes are assumed to be provided with geographical information or relative distances of one node and its surroundings. Brust *et al.* proposed WACA in [24], an algorithm whose weight function relies solely on locally available information. The main concern of the authors in this work is to reduce the network communication overhead during the clustering process. As each node elects one neighbor as cluster-head in its direct neighborhood, their solution may lead to chains of cluster-head, i.e. a node a may elect b as cluster-head and b may elect c (a and c are not neighbors).

Equivalence.

One-hop clustering algorithms creates structures in which cluster-slave have at least one cluster-head in their direct neighborhood (one-hop). Such characteristic correspond exactly to the coverage property of a dominating set. If no weight values are considered, finding the minimum number of clusters (and cluster-heads) is equivalent to finding the minimum dominating set of the communication graph. Figure 3.7 provides an illustration of this equivalence.

3.2.2.3 Dominating Set-based virtual backbone

The last category of approaches to create virtual backbone aims at creating a structure with the characteristics of a connected dominating set. This notion is a connected variation of the dominating set, i.e. a set of nodes covering a whole graph. Definitions 30 and 18 respectively define the concepts of dominating set and connected dominating set.

Creating such a backbone is suitable as all devices may either be in the backbone or have at least a one-hop neighbor in the backbone. Moreover, the sub-graph induced by the components of the backbone is connected, i.e. a path only composed of backbone nodes exists between any pair of backbone nodes.

Many contributions have been proposed through time and comparing them is not an easy task. Against this background, a taxonomy is proposed in chapter 4.

Definition 17 (Dominating Set).
In graph theory, a dominating set for $G = (V, E)$ is a subset $V' \subseteq V$ such that for all $u \in V - V'$ there is at least a $v \in V'$ for which $(u, v) \in E$.

Definition 18 (Connected Dominating Set).
In graph theory, a connected dominating set for $G = (V, E)$ is a subset $V' \subseteq V$ such that for all $u \in V - V'$ there is at least a $v \in V'$ for which $(u, v) \in E$ and the sub-graph induced by V', $G[V']$ has only one connected component.

3.3 Metrics for virtual backbones

From our point of view, either algorithms or their results can be compared. As far as algorithms are concerned, they can be differentiated by their nature (centralized, distributed), their time and message complexity or even their approximation ratio if it is relevant. Chapter 4 provides an exhaustive taxonomy of the $CDS-$based virtual backbone algorithms and thus their main characteristics are detailed. In this section, we focus on how to determine the quality of the virtual backbones themselves, thanks to quality measures or metrics. We start by presenting the motivation of such an approach and we provide some important characteristics for virtual backbones. In the two following parts, we introduce some of the most used metrics for both tree-based and cluster-based virtual backbones. The last part first provides an overview of the quality metrics for $CDS-$virtual backbones and then proposes a set of specific measures to quantify quality in mobile environments.

3.3.1 Motivations and related criteria

In a first time we explain why comparing virtual backbones is important and what are their suitable characteristics. The quality criteria for such structures are then classified into three main categories.

3.3.1.1 Motivations

Virtual backbone is a solution to leverage the inherent scalability issues of ad hoc networks. As different algorithms means different virtual backbones, it is important for comparison purpose, to quantify the quality of a given virtual backbone. From the purpose of creating such structures, we can deduce the two following facts about quality:

- A good virtual backbone should have a more positive impact on the underlying network than a less suitable one. By defining the nature of the positive impact it is then possible to derive quality measures.
- The overhead required to establish and maintain these virtual backbones should be minimized. From this statement we deduce that the structure should be robust and stable in order to reduce the maintenance processes.

3.3.1.2 Quality criteria

From these two straightforward statements, we propose the following categories regrouping a wide panel of quality measures if not all of them.

Size or weight.

The cardinality or the total weight of a virtual backbone is one of the most popular way to compare different solutions as it is generally assumed that more compact or light structures induce less energy consumption and better bandwidth usage. This category is the most commonly used to determine the quality of $CDS-$based and $tree-$based virtual backbones. In the former, cardinality of the dominating set is generally considered and in the latter, weight values of edges or vertices are taken into account.

Robustness and stability.

We intuitively know that a robust and stable solution is more suitable. In the case of virtual backbones for mobile ad hoc networks, these structures are generally intended to be used by higher-level protocols such as routing and broadcasting. In such a context, having a robust and stable structure is a suitable feature to reduce the emission of protocol maintenance messages.

Let us first define what robustness means. Robustness is defined in the Oxford Dictionary as the ability to withstand or overcome adverse conditions (see definition 19). Based on this definition, many metrics can quantify a robustness aspect depending on the designer point-of-view. However, for any robustness-related measure, one important notion has to be defined. Indeed, in order to quantify how much robust is a virtual backbone, adverse conditions should also be defined and be measurable. This idea has been developed by Siegel *et al.* in [3]. Indeed, to quantify the robustness of a system, Siegel proposed a methodology based on the three following questions:

1. What behavior of the system makes it robust?
2. What uncertainties is the system robust against?
3. Quantitatively, exactly how robust is the system?

Adverse conditions may be from different nature in mobile ad hoc networks. Topology changes for example are the most recurrent type of event that may happen. In such cases, the robustness of a virtual backbone can be defined as its capacity to withstand topology changes due to mobility or node failure.

Another adverse condition is related to the distributed and concurrent aspect of these networks. Indeed, as far as decentralized algorithms are concerned, network nodes have to gather information to take local decisions, i.e. to be a backbone or a normal node. These pieces of information are most of the time obtained with beacon packets or specific control messages. However, due to topology changes and radio related problems, the local information of some nodes may be inconsistent (e.g. some neighbors may not be reachable anymore) or incomplete. As local decisions are based on these pieces of information, wrong decisions may be taken by some nodes, and the set of selected nodes may not be a CDS during some time, i.e. some nodes may not be covered or the backbone is not connected. As a consequence, we think that a robust virtual backbone algorithm should be able to withstand these kind of adverse situations and that its ability should be quantified using specific quality metrics.

Definition 19 (Robustness - *Oxford Dictionary*).
The ability to withstand or overcome adverse conditions.

Quality of representation.

One of the often overlooked quality criterion for virtual backbones is the quantification of how well the structure represents the network. Although this type of measurement is commonly used for cluster-based techniques, it is generally not considered for tree-based and $CDS-$based approaches. This category may encompass many different aspects, such as:

- The access to the backbone, i.e. all non-backbone nodes should have a way to access the virtual backbone.

- All pairs of nodes in the same connected components should be able to communicate, i.e. if communication is possible via the communication graph it should also be possible by using the virtual backbone.

- The virtual backbone should be characterized by a small stretch, i.e. for any pair of network nodes, communicating via the backbone should not induce much longer paths than the shortest available paths of the communication graph. As previously stated, this characteristic is suitable in order to avoid substantial decreases in some quality of service (QoS) measures (round-trip time, percentage of successful delivery).

3.3.2 State-of-the-art

We have seen that the quality of a virtual backbone is a highly versatile notion. A brief state-of-the-art for each type of virtual backbone is now provided.

3.3.2.1 Tree-based virtual backbones

Spanning tree algorithms generally try to minimize the total weight of the generated tree. Weights can be on the nodes and/or on the edges of the communication graph. These weights can represent many networks characteristics, e.g. wireless channel quality (based on signal to noise ratio or percentage of successful communication), trust values [67] or even the volatility of the neighborhood [66].

3.3.2.2 Cluster-based virtual backbones

Representation.

Clustering techniques are designed to re-group nodes based on their similarity. As a consequence, depending on the considered similarity measure, very different results may be found. Some measures are however widely used for comparison purposes. The modularity measure Q, introduced by Newman and Girvan in [65] is one of the first metric used to evaluate the quality of a partition. It measures how the generated set of clusters match the community structure exhibited by the network topology. Whenever a natural or pre-defined partition exists for a network, many other metrics have been developed to estimate a normalized distance between the given and the computed clusters. In [33], Danon *et al.* introduced the normalized mutual information (NMI) measure based on information theory. This normalized metric evolves between 0 when computed (C) and pre-determined (C') partitions are independent, and 1 when they are identical.

Stability.

Clustering techniques for ad hoc networks are generally considering the stability of their clusters. Indeed, as clustering techniques generally re-group nodes that are more alike, the clusters have to evolve if the characteristics of the nodes are changing. The frequency of cluster-head changes in particular is regularly considered a good indicator of the clustering mechanism. In that case, the less changes the better.

3.3.2.3 CDS-based virtual backbones

In the majority of the contributions, few indicators are used to measure the quality of the generated virtual backbone. It is indeed more common to provide information concerning the characteristics of the provided algorithm, such as computation, time or message complexity. In a sense, approximation factors, generally derived from the characteristics of a specific and well-defined context (e.g. Unit Disk Graph) can be considered as a quality metric as it provides theoretical values concerning the size of the generated output.

However, even if such theoretical results are insightful, it does not help measuring the quality of the generated virtual backbone obtained by processing the algorithm. To such extend, for most proposed solutions, simulations are done as a way to empirically demonstrate that the algorithm behave correctly. In these papers [90, 90, 31, 58], only the size of the backbones are taken into account as a way to directly compare solutions generated from different algorithms.

Basagni *et al.* in [14] compared the robustness of different $CDS-$based virtual backbone algorithms as the maximum number of node failure until the backbone is considered broken, i.e. the backbone is disconnected or some non-backbone nodes are not covered anymore. As a consequence, definition 20 provides an indirect measure on how long the network will be operational before requiring a backbone re-computation or maintenance.

Definition 20 (Robustness - *Basagni*)**.**
Robustness is quantified by the number of backbone nodes whose removal (because of failure or energy depletion), causes backbone disconnection or the uncovering of ordinary nodes.

As previously mentioned, very few works have tackled the effects of mobile environment which brings new challenges in terms of algorithms robustness and performance metrics. Indeed the vast majority of the literature only considers static networks. As an exception, some interesting conclusions can be found in [1] in which two different approaches to construct CDS are compared with respect to some communication protocol metrics such as collision rate, coverage percentage of broadcast packets or bandwidth usage. These types of metrics are very important when real implementation is considered, however results are only sound with the selected communication protocols and may vary a lot with different configurations. Interesting work about metrics in dynamic graphs can also be found in Yoann Pigné's Ph.D. thesis [66]. These propositions are good indicators of the dynamicity of a graph but are not designed for quantifying the quality of a virtual backbones.

3.3.3 Contributions for $CDS-$based virtual backbone

In this subsection, we propose four quality measures relative to the quality of a CDS backbone through time. Some of them are well-known such as the cardinality and some others are original. These measures are re-grouped as a tool box to quantify raw performances of a given algorithm. However, aggregating these measures with different weight values to compare the performances of some specific algorithms is not in the scope of this work.

We think that this contribution is useful as very few works have been considering the problem of quantifying CDS solutions. Moreover, dynamic quality aspects of the structure are generally completely avoided as static networks simulations represent the wide majority of the empirical assessments.

Let us consider $VB(G_d)$ the virtual backbone built on top of the dynamic graph G_d. Let us consider that the simulation time t is discretized and composed of T steps, $t_i \in \{t_1, \ldots, t_T\}$. As a consequence, $VB_t(G_d)$ represents the actual virtual backbone at an instant t.

3.3.3.1 Size

As previously mentioned, the size of the CDS has a major impact on the quantity of messages and thus on the available bandwidth for real communication. In a static context, we just have to count the number of nodes in the backbone. However, such approach is designed for static instances of graph. A simple idea to integrate the dynamicity is to save a series of size measures on a fixed frequency and to compute the average value. However this may not detect an algorithm that offers really compact and near optimal structures at some time of the simulation and very large ones at some other. Therefore we propose to use a more complete statistical data set (e.g. mean, median, min, max, standard deviation) to have a more detailed view of your algorithm behavior concerning the backbone size. To obtain a network-size independent set of measures, the percentage of backbone nodes can be considered instead of the size of the backbone.

Measure 1 (CARDINALITY).
Mean, median, minimal, maximal and standard deviation of the series of values : $\{VB_t(G_d)\}_{t\in\{t_1,\ldots,t_T\}}$.
Mean:

$$\overline{VB_t(G_d)} = \frac{\sum_{t=1,\ldots,t_T} |VB_t(G_d)|}{t_T}$$

Standard deviation:

$$\sigma_{VB_t(G_d)} = \sqrt{\frac{1}{t_T}\sum_{t=1}^{t_T}\left(VB_t(G_d) - \overline{VB_t(G_d)}\right)}$$

3.3.3.2 Stability

The stability of the backbone is generally an antagonist notion of the size. It is indeed straightforward that having the smallest possible backbone requires frequent maintenance to adapt quickly to the topology changes. Some may however argue that it also depends on how well nodes are chosen, i.e. two compact solutions may not have the same maintenance requirements if more stable nodes are selected for the backbone.

A stable backbone is suitable for higher-level services such as broadcast or routing, because it reduces their needs of control messages. Indeed, in a stable backbone, updating the routing paths can be done less often. A simple idea to compare algorithms requirements is to count the number of nodes going in and out of the backbone during the simulation time. The details are presented in measure 2. This measure is really simple but it does not quantify the stability of the backbone versus the stability of the network, i.e. the values of this measure highly depends on the simulated scenario. This type of measurements is well-known in the ad hoc clustering domain in which finding the more suitable nodes as representatives (cluster-heads) is a major objective.

Measure 2 (Number of state changes).
Let C_{t_i} the set of nodes that change their states at time t_i. Counting the number of changes during the simulation is:

$$statechange = \sum_{\{t_i\}_{t_1,\dots,t_T}} |C_{t_i}|$$

For a network size independent measure, we can average the number of changes by the number of nodes.

This work does not take into account the protocols based on top of the backbone. However, it can be envisaged to create a protocol-specific measure that would count the overhead induced by the backbone (size and stability) compared to an optimal solution.

3.3.3.3 Representation

Mobility induces inconsistencies between local information gathered by all nodes and the communication graph. As a consequence to this inherent characteristic, wrong concurrent and distributed decisions may lead to:

- virtual backbones not representing well the connected component of the communication graph.
- isolated nodes, i.e. non-dominated non-backbone nodes.

In both cases, some couples of nodes may not be able to communicate using the virtual backbone despite an existing route in the communication graph. In order to quantify these problems we propose two straightforward measures, 3 and 4. Measure 3 keeps track of the number of connected component in the communication graph and in the graph induced by the virtual backbones. The computed ratio can have different value ranges which represent different cases:

- $ratio = 1$, the generated backbones are likely to fit the connected components of the communication graph.
- $ratio < 1$, as more backbones than connected components exist, some nodes inside the same connected component may not be able to communicate via the backbone.
- $ratio > 1$, this case may happen if the backbone is not adapting quickly enough to the topology changes.

Measure 3 (Connected component ratio)**.**
Ratio between the number of connected component, CC, of the graph G_d and the number of connected components of the backbone nodes induced subgraph, CC_{VB} at time t.

$$cc\_ratio_t = \frac{|CC|}{|CC_{VB}|}$$

Mean, median, minimal, maximal and standard deviation of the series of values cc_ratio_t can be used to obtain a summary of this measure through time.

Measure 4 keeps track of the number of isolated nodes, i.e. non-backbone nodes with no backbone node in its direct neighborhood.

Measure 4 (Isolated nodes)**.**
Counter of the number of isolated nodes, i.e. non-backbone node not covered by the backbone node at time t. $N_1^B(v,t)$ represents the one-hop backbone neighbors of node v at time t.

$$nb\_isolated_t = \Big| \{v \in V \ / \ N_1^B(v,t) = \emptyset\} \Big|$$

Mean, median, minimal, maximal and standard deviation of the series of values $nb\_isolated_t$ can be used to obtain a summary of this measure through time.

3.4 Recap of key points

- Topology management techniques are designed to leverage the encountered scalability issues in ad hoc networks, such as the available bandwidth per node or the broadcast storm problem.
- Topology managements techniques can be classified into two categories: topology control and virtual backbones
- Topology control methods generally aim at designing the network topology in order to optimize some criterion (energy consumption, network capacity,...). These techniques fulfill their objectives by changing the range assignment, i.e. modifying the transmitting power for each node.
- Virtual backbones do not modify the underlying communication graph. Instead, a subset of network nodes and/or communication channels are selected to relay the network traffic.
- Three main types of virtual backbones exist: trees, clusters and connected dominating sets.
- Four quality metrics for $CDS-$based virtual backbone have been proposed as a toolkit to compare algorithms for mobile ad hoc networks.

Chapter 4

Taxonomy for CDS-based virtual backbone

Contents

4.1 Characteristics **46**
4.1.1 Approaches 46
4.1.2 Computation type 47
4.1.3 Available data 48
4.1.4 Randomization 49
4.1.5 Robustness considerations 49
4.1.6 Synchronization requirements 50
4.2 Taxonomy **51**
4.2.1 Centralized algorithms 51
4.2.2 Distributed algorithms 53
4.2.3 Robust CDS algorithms 58
4.3 Recap of key points **67**

Algorithms to create connected dominating set based virtual backbones are numerous. This profusion of solutions is partly, if not mainly, due to the direct application to the ad hoc networks realm. In order to provide a clear overview of all these contributions, we start by presenting the main characteristics of these algorithms in section 4.1.

Section 4.2 contain classifications for three main categories: centralized, distributed and robust CDS algorithms. For each of them, subcategories along with the presentation of their most representative algorithms are detailed.

4.1 Characteristics

In this section we present some salient characteristics concerning the algorithms in general (type of result, computation type, available data or randomization) and some others which are specific to the virtual backbones for ad hoc networks (robustness consideration and synchronization requirements).

4.1.1 Approaches

Three different approaches exist depending on the quality of the solution:

- Exact algorithms, that require a tremendous amount of computation but always provide an optimal solution.
- Approximation algorithm, that are generally more computer efficient and guaranty the quality of their solution via their approximation ratio.
- All the other algorithms, that do not guaranty anything but provide reasonably good solution in a reasonably computer efficient fashion.

4.1.1.1 Exact

The Minimum Connected Dominating Set (MCDS) in general graphs was studied in [40], in which a reduction from the Set Cover Problem [40] to the MCDS was shown. This result implies that this problem is NP-Hard, and as a consequence, no time-efficient algorithm can be found to optimally solve the MCDS problem. Definition 21 states that an algorithm is exact if the computed solution S for any input graph I is always valid and optimal. Exact algorithms are often compared to each other via their time or space complexity, i.e. a formal way to represent the time or space required by an algorithm to process a given input in the worst case. Moreover, some contribution are studying special classes of graphs only and thus cannot be applied mainstream. Focusing on a particular class of graphs is a common techniques used to benefit from the specific characteristics of such class (e.g. in UDG, any node has a maximum of five independent neighbors).

This type of algorithm are very important as they provide optimal solution and thus permit to have a better idea of the empirical performances of the approximation or heuristic algorithms.

Definition 21 (Exact algorithm).
An algorithm A is exact if for any given input I, a valid and optimal solution S is obtained by applying algorithm A

4.1.1.2 Approximation

In computer science and operational research, approximation algorithms aim to find approximate solutions to optimization problems. Approximation algorithms are often associated with NP-hard problems; since it is unlikely that there can ever be efficient polynomial time exact algorithms solving NP-hard problems, one settles for polynomial time sub-optimal solutions.

Approximation algorithms are mainly characterized by their result guaranty: the approximation ratio or factor. Let us consider $OPT(I)$ the optimal solution of the MCDS problem for the input I. A $\rho-$approximation algorithm, for some $\rho > 1$, is an algorithm that produces solutions whose value is at most $\rho \cdot OPT(I)$.

This types of algorithms generally provide good quality solution in polynomial time, which is much more suitable in practice. Moreover, their performance guaranty is an important asset for demanding applications.

4.1.1.3 Others

This last category is composed of pragmatic algorithms which usually find reasonably good solutions reasonably fast. They are generally able to produce acceptable solutions in many practical scenarios but provide no formal proof of their correctness. Alternatively, they may be correct, but may not be proven to produce a bounded solution, or to use reasonable resources. This type of algorithm is typically used when there is no known method to find an optimal solution, under the given constraints (of time, space etc.) or at all.

As detailed previously in chapter 2, ad hoc networks are generally characterized by a very constraint environment. As a matter of fact, in this context, this type of algorithm is particularly suitable as it generally requires less computational capacities than the approximation scheme and the exact algorithms.

4.1.2 Computation type

All the contribution we are reviewing can be partitioned into two main classes: centralized and distributed algorithms.

4.1.2.1 Centralized

A centralized algorithm is characterized by its only one decision entity, i.e. one global coordinator is in charge of computing a solution S from the input I.

As ad hoc networks are intrinsically distributed entities, centralized algorithms are generally not suitable. Some may argue that they can be implemented as far as the network size is not too big. Indeed, the bigger the number of nodes in the network, the more communication is required to gather the whole communication graph in one specific node. Nevertheless, these algorithms are useful as they generally provide better quality solutions which can be used to compare the performances of distributed solutions.

4.1.2.2 Distributed / Decentralized

As stated in definition 22, a distributed algorithm is an algorithm designed to be executed on separated entities. These entities are partly or fully connected to each other and concurrently execute the algorithm with limited information about what the other entities are doing.

One of the major challenges in developing and implementing distributed algorithms is successfully coordinating the behavior of the independent entities as some of them may fail and the communication between them may be unreliable.

Definition 22 (Distributed or decentralized algorithm).
A distributed algorithm is an algorithm designed to run on computer hardware constructed from interconnected processors. Distributed algorithms are used in a host of application areas of distributed computing, such as telecommunications, scientific computing, distributed information processing, and real-time process control. Standard problems solved by distributed algorithms include leader election, consensus, distributed search, spanning tree generation, mutual exclusion, and resource allocation.

4.1.3 Available data

Another important criterion to classify the algorithm is related to the amount of information to which the algorithm has access to compute its solution. Two main categories can be roughly defined: global and localized.

4.1.3.1 Global algorithms

Algorithms benefiting from global information have access to the complete communication graph to take their decision. If we consider all the works that are reviewed in this thesis, we can conclude that all the centralized algorithm are using global information. However, some distributed algorithms are also benefiting from the global information.

$$\text{Centralized algorithm} \subset \text{Algorithm using global information} \tag{4.1}$$
$$\text{Algorithm using global information} \not\subset \text{Centralized algorithm} \tag{4.2}$$

4.1.3.2 Localized algorithms

Localized algorithms are relying on local information to take decisions. As such, all the localized algorithm are obviously also distributed, i.e. the different entities computing the solution have only access to a part of the global information. The scope limit of this local information is generally referred as the hop count, i.e. the distance from the considered entity to its farthest known neighbor.

As ad hoc networks are characterized by the lack of global coordinator, this type of algorithm is particularly suitable for real-implementation. However, having less information generally induces poorer quality for the solution.

$$\text{Localized algorithm} \subset \text{Distributed algorithms} \tag{4.3}$$
$$\text{Distributed algorithms} \not\subset \text{Localized algorithm} \tag{4.4}$$

4.1.4 Randomization

Whether an algorithm always provides the same solution for the same input or not, it can be classified in two different categories: deterministic or stochastic.

4.1.4.1 Deterministic

As stated in definition 23, a deterministic algorithm always produces the same solution for a given input. The majority of the reviewed contribution are deterministic algorithms.

Definition 23 (Deterministic algorithm)**.**
In computer science, a deterministic algorithm is an algorithm which, in informal terms, behaves predictably. Given a particular input, it will always produce the same output, and the underlying machine will always pass through the same sequence of states.

4.1.4.2 Stochastic

Some proposed algorithms to build CDS-based VB are stochastic, which means that at some moment of the algorithm a decision is based on a randomly generated number (see definition 24. Such approaches do not provide strong guaranty concerning the provided solution, however they tend to be lightweight processes. Moreover, the authors empirically measured via simulations the probability of obtaining a valid solution depending on some environmental factors such as node density or the velocity of the nodes.

Definition 24 (Stochastic algorithm)**.**
A stochastic or probabilistic algorithm is an algorithm which employs a degree of randomness as part of its logic. The algorithm typically uses uniformly random bits as an auxiliary input to guide its behavior, in the hope of achieving good performance in the "average case" over all possible choices of random bits. Formally, the algorithm's performance will be a random variable determined by the random bits; thus either the running time, or the output (or both) are random variables.

4.1.5 Robustness considerations

Many contributions have been focusing on creating connected dominating sets in ad hoc networks. However, these networks are characterized by node failure and / or mobility. As a consequence, resilient solution have been proposed: increasing the domination (or vertex domination) and the connectivity.

The $m-$domination (or $m-$vertex domination) is a property stating that non-backbone nodes should have at least m backbone nodes in their direct neighborhood and thus help reducing the number of non-connected nodes, i.e. nodes with no access to the virtual-backbone. The $k-$connectivity constraint increases the reliability of the backbone itself as k independent paths have to exist between any pair of backbone node, i.e. $k-1$ backbone nodes can fail and the backbone will remain connected.

A plethora of solutions have been proposed but not all of them create $k-$connected $m-$dominating sets for general values k and m. Some contributions are restricted to special cases such as $k = 1$ and $m = 2$ or $k \leq m$.

4.1.6 Synchronization requirements

Last but no least, there is a criterion that concerns what we call *synchronization requirements*. This criterion encompasses all the assumed assumptions required by the algorithms to perform, i.e. the synchronization required to simultaneously process different algorithms steps. It is important, from our point of view to classify in different classes the algorithms requiring synchronization and those who do not rely on such a characteristic.

4.2 Taxonomy

In this section we provide a systematic state-of-the-art for three different classes of algorithms. Centralized algorithms to create CDS are classified into three subcategories (exact, approximation and heuristics). Distributed algorithms are in a second time presented and regrouped into three subcategories: self-pruning, MIS-based and Multi-Point Relay approaches. Robust approaches are then detailed in the last part. A classification based on their technique to achieve the desired robustness property is proposed. Figure 4.1 illustrates the organization of the proposed taxonomy.

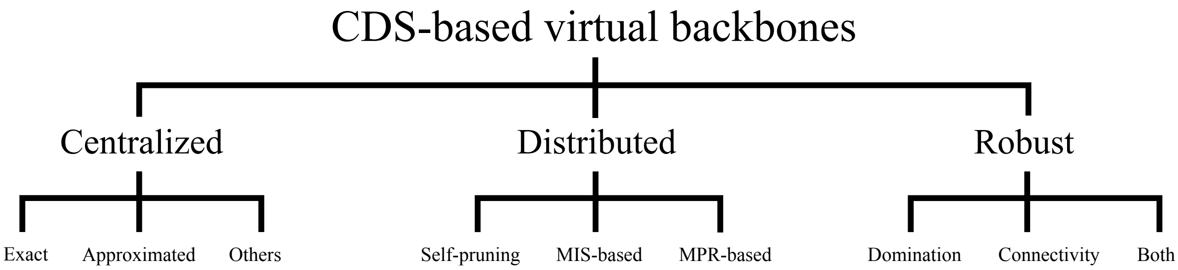


Figure 4.1: Organization of the CDS-based virtual backbone taxonomy.

4.2.1 Centralized algorithms

Centralized algorithms can be partitioned into three main classes: exact, approximation and all those who do not fit in the previous categories. The following algorithms are benefiting from the complete communication graph (global algorithm) and one entity is computing the solution.

4.2.1.1 Exact algorithms.

During the writing of this thesis, the best exact algorithm in terms of time-complexity is proposed by Liedloff *et al.* [37]. This work proposed an algorithm with running time $O(1.8966^n)$ based on a shrewd branching technique. The analysis of the time-complexity has been analyzed using the Measure-and-Conquer technique. Table 4.1 summarizes the main characteristics of all the considered centralized contributions. Another interesting contribution has been developed by Tetsuya Fujie in [39]. Although the author is proposing an exact algorithm to solve the Maximum Leaf Spanning Tree, it does not provide any analysis concerning the time-complexity. However, an empirical study based on the performances of the algorithms for randomly generated graphs is provided. This insightful experience enlightens the fact that the time required to solve an instance of graph is strongly correlated with the density of the graph.

4.2.1.2 Approximation algorithms.

Guha and Khuller [44] proposed two well-know approximation algorithms for the MCDS problem. The first one is characterized by a greedy behavior defined in algorithm 1. This simple yet efficient algorithm has an approximation ratio of $2 \cdot (1 + H(\Delta)) \cdot |OPT|$ with H the harmonic function. The four main steps of this algorithm are detailed in algorithm 1.

Algorithm 1: First algorithm from Guha and Khuller

```
Input: Graph G = (V, E)
Output: The set of black nodes
Mark all nodes white;
Mark the node with the most white neighbors black;
Mark its neighbors gray;
while Some white nodes remain do
    Find the gray node u
    or
    a couple of node (u, v)
    (one gray and on of its white neighbor)
    with the most white neighbors;
    Mark u or (u, v) black;
    Mark their neighbors gray;
```

The second algorithm of Guha and Khuller [44] requires the definition of a *piece* (see definition 25. In a first time the algorithm creates a Weakly Connected Dominating Set (see definition 26). The second and last phase of the algorithm adds some nodes in order to obtain a valid CDS. The approximation ration of this algorithm is $3\ln(\Delta) \cdot |OPT|$. The three steps are presented in algorithm 2

Definition 25 (Piece - *Guha and Khuller*).
A piece *is either a black connected component or a white node.*

Definition 26 (Weakly connected dominating set).
In a graph $G = (V, E)$, the sub-graph weakly induced by S ($S \subset V$) is the graph $S_w = (N[S], E \cap (N[S] \times S))$ where $N[S]$ includes the nodes in S and all of their one-hop neighbors. The edges of S_w are all edges of G that have at least one end point in S.
The subset S is a weakly connected dominating set if S is a dominating set and S_w is connected.

4.2.1.3 Others algorithms.

Butenko *et al.* proposed a pruning-based heuristic, i.e. the initial CDS C is large and useless nodes are iteratively removed. At first, the algorithm marks all nodes to white and initialize C as the whole vertex set of the input graph. Then, it considers a white node $x \in C$ with minimum effective degree. If removing x from C makes the induced graph of C disconnected, then the algorithm retains x and colors it black. Otherwise it removes x from C. At the same time, if x does not have a black neighbor in C, its neighbor with maximum effective

Algorithm 2: Second algorithm from Guha and Khuller

Input: Graph $G = (V, E)$
Output: The set of black nodes
All node are marked *white*;
while *Some white node remain* **do**
  Find the node that maximize the reduction of the number of pieces;
  Mark this node *black*;
  Mark its neighbors *gray*;
Find a stainer tree connecting all *black* nodes
by coloring chains of two *gray* nodes *black*;

degree in C is colored black. This procedure is repeated until there is no white node left in C. The black nodes form the CDS. This procedure is presented in algorithm 3.

Definition 27 (Effective degree - *Butenko et al.*)**.**
The effective degree of a node is its number of white neighbors in the CDS C.

Algorithm 3: Butenko algorithm

Input: Graph $G = (V, E)$
Output: The set of black nodes
All node are marked *white*;
The set C contain V;
Find $x \in C$ with minimum effective degree;
if $C \setminus x$ *is not connected* **then**
  Mark x black;
else
  Remove x from C;
if x *has no black neighbor in* C **then**
  Mark its neighbor with maximum effective degree in C black;

4.2.2 Distributed algorithms

Many different distributed algorithms with many different characteristics have been proposed. Although they can be partitioned into approximation and heuristics algorithms, we think this type of classification would not be very useful to get a clear view. Instead we propose to present the main different schemes:

- Self-pruning algorithms. These algorithms are composed of two main steps, a *marking process* that creates a *CDS* by adding numerous nodes and a self-pruning phase that removes useless nodes by applying a set of *pruning rules*. These two steps are computed locally by each node by considering their local graph.

Name	Type	Approx. ratio	Time complexity
Liedloff [37]	Exact	-	$O(1.8966^n)$
Fujie [39]	Exact	-	-
Guha1 [44]	Approximation	$2 \cdot (1 + H(\Delta))$	-
Guha2 [44]	Approximation	$3 \ln(\Delta)$	-
Butenko [25]	Other	-	$O(\|V\| \cdot \|E\|)$

Table 4.1: Summary of the main characteristics for the centralized algorithms. Δ is the maximum degree of a graph, n the number of nodes and H the harmonic function.

- Maximum Independent Set based algorithms. These algorithms first build a maximum independent set which is also a dominating set. The second phase consists in adding a minimal set of nodes to connect the chosen set of nodes.
- Multi-Point Relay based algorithms. The main idea of multi-point relay techniques is for any node u to select a subset of one-hop relay nodes to cover all the two-hop nodes.

4.2.2.1 Self-pruning algorithms.

As stated previously, a self-pruning algorithm is characterized by the two following steps: it adds nodes to the CDS with a generally trivial *marking rule*. This first rule creates a valid CDS. In a second step, the algorithm removes useless nodes from the CDS with a set of *pruning rule(s)*. Algorithm 4 provides a general template for this class of algorithms.

One of the most inspirational pruning-based algorithm is due to Wu and Li [58]. In this algorithm, all nodes benefit from a two-hops knowledge in order to take their local decision. Such topological information is gathered via periodical packets, sent by each node and containing their list of direct neighbors. The marking process (see rule 1) is quite straightforward: every node marks itself in the CDS if it has two unconnected neighbors. Note that this procedure is not working for complete graph (or clique), but the authors argued that this special case can be easily treated separately. This first step creates a valid CDS but adds many redundant nodes. Two pruning rules are then proposed to remove some useless nodes. Rule 1 (see rule 2) allows a node u to remove itself from the backbone if one of its direct neighbor, say v, covers all the neighbors of u and has an higher identifier. Rule 2 (see rule 3) extends rule 1 by considering a pair of covering neighbors with higher identifier.

Rule 1 (Marking rule - *Wu and Li*).
Initially, unmark each node. Each node u exchanges its neighbor set $N[u]$ with all its neighbors. u marks itself if there exist vertices v and w such that $(w, u) \in E$ and $(u, v) \in E$, but $(w, v) \notin E$.

Rule 2 (RULE 1 - *Wu and Li*).
Consider two marked vertices u and v, i.e. u and v have been selected via the marking rule. *If $N[u] \subset N[v]$ and $id(u) < id(v)$, unmark u, i.e., u is not required in the CDS.*

Rule 3 (RULE 2 - *Wu and Li*)**.**
Assume v and w are two marked neighbors of marked vertex u. If $N(u) \subset N(v) \cup N(w)$ and $id(u) = minid(u), id(v), id(w)$, then unmark u.

Two major enhancements have been proposed for the original algorithm. Stojmenovic *et al.* [75] proposed to change the identifier-based tie break of the original algorithm by a degree-based tie break, i.e. nodes with a higher number of neighbors will be more likely to stay in the backbone. This small change permits to obtain considerable smaller backbones with a better distribution of backbone nodes in the simulation space. Indeed, the first algorithm elected was more efficient on the border and nodes in the center of the simulation space were stuck in the CDS. In [31], Dai *et al.* proposed a generalization of the two previous rules (see rule 4), referred as rule k. In this last evolution, not only one or two nodes are considered to determine whether a particular node should leave the CDS.

The time complexity of the original version of the algorithm is $O(\Delta^3)$ because a node compares its neighbors set with $\Delta(\Delta - 1)$ pair of marked neighbors in the worst case (rule 2). In the version of Dai *et al.*, the time complexity is reduced to $O(\Delta^2)$ as the decomposition of a graph into strongly connected component requires $O(|V| + |E|)$ and $|E| = |V| = \Delta^2$ in the worst case. The message complexity of all the versions is reasonable: each node sends two messages. The first message is a classical beacon packet in order for each node to advertise itself. As a consequence of the first round of message, all nodes are aware of their direct neighbors. In the second round of message, every node sends its neighbors' list and thus all nodes can compute their local graph with their two-hops knowledge. Table 4.2 presents the time and message complexity for all the reviewed contributions.

Rule 4 (RULE K - *Wu and Li*)**.**
Assume $V' = \{v_1, v_2, \ldots, v_k\}$ is the vertex set of a strongly connected subgraph composed of marked nodes. If $N[u] - V' \subset N[V']$ and $id(u) = min\{id(v_1, \ldots, id(v_k)\}$, unmark u.

Algorithm 4: Self-pruning distributed algorithm template for any node u

```
Input: Local graph G_u = (V, E)
Output: The value of mark
mark = false;
// Marking process
if u is required as part of the initial CDS then
    mark = true;
// Pruning rule(s)
for All pruning rules do
    if The current pruning rule excludes node u then
        mark = false;
        break;
return mark;
```

4.2.2.2 Maximum Independent Set based algorithms.

The general approach of this category of algorithms is based on a classical problem from the graph theory: the *Maximum Independent Set Problem*. An independent set (see definition 28 that is not the subset of another independent set is called maximal (see figure 4.2 for an example). Such sets are dominating sets. Finding such a set of nodes is the first step of this category. It will be referred as the *MIS construction phase*. The second phase consists in finding a set of connectors, preferably a small one, to connect the MIS. This second will be referred as the *connecting phase*. Algorithm 5 provides a general template for these contributions.

> **Definition 28** (Independent Set).
> *An independent set or stable set is a set of vertices in a graph, no two of which are adjacent. That is, it is a set I of vertices such that for every two vertices in I, there is no edge connecting the two. Equivalently, each edge in the graph has at most one endpoint in I. The size of an independent set is the number of vertices it contains.*

Creating an MIS is either single-leader or multiple-leader based as stated in [19]. A single-leader process requires a leader-election algorithm, which takes $O(n \log n)$ messages. In the multiple-leader case, nodes with maximum degree or identifier (id.) among all neighbors can serve as leaders and thus reduce the message complexity. An MIS-based solution takes advantage of the Unit Disk Graph characteristics to provide an approximation ratio. Indeed, in UDG, a node can have a maximum of five independent neighbors, and thus, in the worst scenario a MIS can be five time bigger than the optimal solution.

Alzoubi *et al.* [17] proposed a solution requiring the distributed election of a leader [29] in order to construct a rooted spanning tree. Based on this structure, an iterative labelling strategy is used to classify the nodes in the tree to be either black (dominator) or gray (dominatee), based on their ranks. The rank of a node is the ordered pair of its level (number of hops to the root of the spanning tree) and its id. The labelling process begins from the root node and finishes at the leaves. The node with the lowest rank marks itself black and broadcast a DOMINATOR message. The marking process then continues according to the following rules:

- If the first message received by a node is a DOMINATOR message, it marks itself gray and broadcasts a DOMINATEE message.
- If a node received DOMINATEE messages from all its lower rank neighbors, it marks itself black and sends a DOMINATOR message.

The *connecting phase* is also initiated by the root when the *MIS construction phase* is finished. First the root joins the CDS and broadcasts an INVITE message which is relayed to all two-hops neighbors out of the current CDS. When a black node receives the INVITE message for the first time, it joins the CDS together with the gray nodes that relayed the message. This black node then broadcasts a JOIN message. This process finishes when all the black nodes are in the CDS. This algorithm has an approximation ratio of $8opt + 1$, a time complexity of $O(n)$ and a message complexity of $O(n \log n)$ due to the leader election mechanism.

In the multi-leader version [4], all nodes are initially candidates. Whenever the id of a node becomes the smallest among all of its one-hop candidate neighbors, it will change its

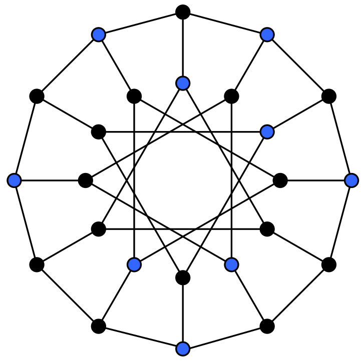


Figure 4.2: The nine blue vertices form a maximum independent set for the Generalized Petersen graph GP(12,4).

status to dominator. Consequently, its candidate neighbors become dominatee. After all the nodes change status, the *connecting phase* consists for each dominator node to identify a path of at most three hops to another dominator with larger identifier. The dominatees on this path are added to the final CDS.

Algorithm 5: MIS-based algorithm template

Input: $G = (V, E)$

Output: The CDS

```
1 // Maximum Independent Set
2 create a maximum independent set I;
3 // Connecting the dominating set
4 determine a small set of connectors C;
5 CDS = I ∪ C
```

4.2.2.3 Multi-Point Relay algorithms.

The main idea of multi-point relay techniques is for any node u to select a subset of one-hop relay nodes to cover all the two-hop nodes. As stated in [70], finding a multi-point relay set with minimum size is NP-Complete.

In [2], Adjih *et al.* proposed a localized heuristic composed of two main phase: the *MPR phase* and a *connecting and pruning phase*. The *MPR phase* is an iterative process composed of two simple rules (see rules 5 and 6). Each node u creates its own multi-point relay set $MPR(u)$ by applying these two rules. Rule 2 is iterated until no uncovered two-hop nodes remain. The second phase is composed of two simple rules (see rules 7 and 8) generating the CDS based on the unique *id* of each node and the $MPRs$ of their neighbors. The correctness of this algorithm is proved in [2] but no performance analysis is available.

Rule 5 (MPR rule 1 - *Adjih et al.*).
v is added to $MPR(u)$ if v has an exclusive access to some two-hop nodes, i.e. v is the only node capable of relaying packets for those nodes.

Rule 6 (MPR RULE 2 - *Adjih et al.*).
v is added to $MPR(u)$ if v is the node covering the most uncovered two-hops nodes.

Rule 7 (CONNECTING RULE 1 - *Adjih et al.*).
u enters the CDS if it has the smallest id amongst its neighbors.

Rule 8 (CONNECTING RULE 2 - *Adjih et al.*).
u enters the CDS if it is a multi-point relay of its smallest id neighbor.

In [86], Wu proposed two enhancements to the original algorithm. First, an additional MPR rule is proposed. This additional rule is executed before the original MPR ones. It consists in adding all the free nodes (see rule 29 of u to $MPR(u)$. The second enhancement of Wu lies in the modification of the first connecting rule (see rule 9): a second condition is added and thus reduces the total number of CDS nodes. This additional condition reminds us the marking process of [58] and thus suffers from the same restriction: complete graphs have to be taken care of as a special case.

Definition 29 (FREE NODE - *Wu MPR*).
v is a free node of u if v is a one-hop neighbor of u and u is not the smallest id neighbor of v.

Rule 9 (CONNECTING RULE 1 - *Wu*).
u enters the CDS if it has the smallest id amongst its neighbors and u has two unconnected neighbors.

4.2.3 Robust CDS algorithms

Recently, many different algorithms have been proposed to create more robust structures. Centralized and distributed algorithms have been designed to either increase the domination or the connectivity or even both. We chose to give an overview of the different techniques that have been developed to increase the previously mentioned properties.

Our first algorithm category groups the solution that increases the domination and the connectivity at the same time. These methods are characterized by unique computational step and the fact that they cannot create structure with different value for m and k.

The second part of this classification is dedicated to methods to increase the domination of a subset of nodes. These techniques are generally the first step of some algorithm to create a $k, m-$CDS: first create a $1, m-$CDS and then augment the connectivity.

The last category regroups all the specific techniques we found to increase the connectivity of a given subset of nodes. Various methods are proposed, some of them require more computational power.

At the end of this subsection, two tables provide important details about the reviewed algorithms. Table 4.4 is dedicated to the centralized solutions and table 4.3 details the distributed algorithms.

Name	**Type**	**Approximation ratio**	**Time complexity**	**Message complexity**
WuLi [58]	Self-pruning	$O(n)$	$O(\Delta^3)$	$O(1)$
Stojmenovic [75]	Self-pruning	$O(n)$	$O(\Delta^3)$	$O(1)$
DaiWu [31]	Self-pruning	$O(n)$	$O(\Delta^2)$	$O(1)$
AlzoubiSingle [17]	MIS	$8opt + 1$ in UDG	$O(n)$	$O(n \log n)$
AlzoubiMulti [4]	MIS	$192opt + 48$ in UDG	-	$O(n)$
Adjih MPR [2]	MPR	-	$O(\Delta^2)$	-
Wu MPR [86]	MPR	-	$O(\Delta^2)$	-

Table 4.2: Summary of the main characteristics for the distributed algorithms. Δ is the maximum degree of a graph, n the number of nodes.

4.2.3.1 Increasing both domination and connectivity

Some authors provided solutions to increase both the domination and the connectivity of a subset of nodes. These methods share one common characteristic: they cannot set a different value for the domination and the connectivity. We classified these approaches into two main categories: stochastic algorithms and a deterministic approach using an generalized version of the coverage property.

Stochastic.

Two stochastic algorithms, *k-Gossip* and *k-Grid* have been proposed by Wu *et al.* in [32]. In the scope of this taxonomy, an algorithm is said to be stochastic if in perfect conditions, the validity of the solution is not ensured for any run of the algorithm. Perfect conditions consider that message rounds occur without any collision and that the exchanged information is correct and up-to-date.

k-Gossip is a simple extension of an exiting stochastic algorithm [52], where each node becomes a backbone node with a given probability p_k. This algorithm has very low overhead, as it only requires to compute a random number and compare it to the probability p_k. Moreover, no message are exchanged during the process. Table 4.4 summarizes the main characteristics of all the considered distributed solutions for robust CDS.

The *k-Grid* algorithm is inspired by a stochastic topology control scheme [16]; it reduces the $k = m - CDS$ size via selecting B_k backbone nodes within the neighborhood of each node. This protocol incurs a slight overhead of neighborhood density estimation which only requires one round of message and can be computed in $O(\Delta)$ in the worst case.

These two algorithms require very few messages and computation power which are two important properties for mobile ad hoc networks. As *k-Gossip* and *k-Grid* [32] creates $k = mCDS$, their approach to increase the connectivity is embedded in their approach to increase the domination. Moreover, only one computation step is required to obtain a solution. Such characteristic is highly suitable as no synchronization is required. However, good p_k and B_k values highly depends on the network size and density, two global information that are difficult to gather or estimate in a distributed context.

k-coverage property.

Along with the two stochastic approaches, Wu *et al.* [32] proposed a deterministic algorithm based on the *k-coverage condition* (see definition 10). In the original coverage condition [87], a node can be removed from a CDS if all its neighbors are interconnected via a replacement path. In the *k-coverage condition*, the criterion is more strict: if a node is to be removed from a $k = m - CDS$, all its neighbors must be k-connected with each other via higher priority nodes.

This algorithm requires two rounds of messages so that all nodes gather their two-hop topology graph. Only one computation step is required to obtain a valid solution in a deterministic way. However, this algorithm suffers from a high computational cost: $O(k\Delta^4)$ and may not be suitable for high density networks.

Rule 10 (k-Coverage condition - *Wu et al*).
Node v has a non-backbone status if for any two neighbors u and w, k node disjoint replacement paths exist that connect u and w via several intermediate nodes (if any) with higher IDs than v.

4.2.3.2 Increasing the domination

In the literature, some techniques have been developed to specifically increase the domination of a subset of nodes [90, 89]. We classified these methods into two main categories: greedy approaches and multiple maximum independent set methods.

Greedy.

In [90], Wu *et al.* proposed CGA, a centralized algorithm based on a simple greedy scheme to create $k, m-$CDS for any value of k and m. The CGA algorithm consists in three consecutive steps, all of them are greedy-based: creating a $m-$dominating set, augments the set until it is $k-$connected, prune some useless nodes if any.

In order to obtain a $m-$dominating set, all the nodes are sorted in decreasing order based on tuple $(N_i,\ e_i,\ ID_i)$, with N_i the number of neighbor of node i, e_i the remaining energy of node i and ID_i the identifier of node i. A $m-$dominating set C is then constructed by repeatedly adding nodes from the sorted list until the set is $m-$dominating.

Wu *et al.* proposed $ICGA$ in [89] to solve some issues related to CGA, such as the guaranty of obtaining a valid solution and an approximation ration (see equation 4.5). In $ICGA$, a $1, 1-$CDS is created and all its nodes are added to the dominating set C. Then, while there exists a non-backbrone node with less than m dominators in its direct neighborhood, the following greedy procedure is executed. First we collect all the non-backbone nodes that are not $m-$dominated in a set P. Then we find the non-backbone node v dominating the most nodes in P and we add it to C.

$$f = \begin{cases} k \le 6 & \begin{cases} 5k + \frac{5}{m} + 5H_{k-1} & m \le 5 \\ 7k & m \ge 6 \end{cases} \\ k \ge 7 & \begin{cases} 7k - 7 & m \le 5 \\ 7k & m \ge 6. \end{cases} \end{cases} \tag{4.5}$$

Multiple MIS.

Another popular technique to build a $m-$dominating set consists in adding m different maximum independent set. The general template is presented in the $CDSMIS$ [81] algorithm 6: at each iteration, a different MIS is generated and remove from the original graph. The union of these different MIS is the $m-$dominating set.

Many algorithms, centralized or distributed uses this approach. In [81], Thai *et al.* proposed $CDSMIS$, a centralized algorithm with an approximation ratio of $m + 7 + \ln 5$. In [90], Wu *et al.* presented DDA, a distributed algorithm to build $k, m-$CDS. DDA first builds a $1, m-$CDS with a distributed but similar approach than *CDSMIS*. The second phase adds nodes until the connectivity is sufficient. Wu *et al.* proposed $MDSA$ in [89], a distributed algorithm to build $1, m-$CDS with an approximation ratio (see equation 4.6) that depends on the value of m. More details are provided about $MDSA$ in table 4.4.

Algorithm 6: CDSMIS algorithm

Input: An $m-$connected $G = (V, E)$

Output: A $1, m-$CDS C of G

```
Construct a 1−CDS C in the given network where C = I_1 ∪ B;
Remove I_1 from the graph  for i = 2 to m do
    Construct an MIS I_i in G − (I_1 ∪ I_2 ∪ · · · ∪ I_{i−1});
    C = C ∪ I_i;
Return C where C is the 1, m−CDS
```

$$f = \begin{cases} 5 + \frac{5}{m} & \text{for } m \leq 5 \\ 7 & \text{for } m \geq 6. \end{cases} \tag{4.6}$$

4.2.3.3 Increasing the connectivity

Identically to increasing the domination, specific techniques to augment the connectivity of a subset of nodes have been developed [84, 90, 89]. In this part we classified these methods into four main categories: greedy, iterative increase of the connectivity, iterative increase of the hop count and finally common backbone nodes negotiation.

Greedy.

In [84], Wang *et al.* proposed a centralized solution to create $2-$connected dominating sets. The main contribution in this work is a greedy algorithm to increase the connectivity of a given CDS C. A main loop is iterated until the structure is $2-$connected. Such test can be done in linear time with a well-known algorithm that computes the different $2-$connected components also known as blocks. If more than one biconnected component remain, a path is chosen based on these criteria:

- The path can connect a leaf block in C to another portion of C. A leaf block is a block with only one leaf neighbor.
- The path does not contain any nodes in C except the two endpoints.

When such a path is found, its intermediate nodes are added to C and the loop condition is tested.

As for the $m-$domination, CGA [90] creates a $k-$connected set by adding sorted nodes to the set in a greedy way. Indeed in order to obtain a $k-$connected set, all the nodes are sorted in decreasing order based on tuple (N_i, e_i, ID_i), with N_i the number of neighbor of node i, e_i the remaining energy of node i and ID_i the identifier of node i. The first step of the CGA algorithm is to create a $m-$dominating set C. The $k-$connectivity is obtained in a second time by repeatedly adding nodes from the sorted list to C and checking if C is $k-$connected. However, the CGA algorithm cannot be applied successfully on a general graph. The input graph should indeed be $\max(k, m)-$connected to ensure a valid solution.

Name	k cases	m cases	Approximation ratio	Time complexity	Message complexity
CDSA [84]	$k = 2$	$m = 1$	constant 64	$O(n^3)$	-
CGA [90]	any value	any value	-	$O(m \cdot n^{3.5})$	-
MDSA [89]	$k = 1$	any value	see 4.6	$O(m \cdot Diam)$	$O(m(\Delta + 1)n)$
ICGA [89]	any value	any value	see 4.5	$O(m \cdot n^{3.5})$	-
CDSMIS [81]	$k = 1$	any value	$m + 7 + \ln 5$	-	-
CDSAN [81]	any value	$m = k$	$(k + 7 + \ln 5)(2k - 1)$	-	-
CDSMIS+CDSAN [81]	any value	any value	$(2k + 9)(k + 6 + \ln 5)$	-	-

Table 4.3: Summary of the main characteristics for the centralized algorithms. Δ is the maximum degree of a graph, n the number of nodes, m the number of edges and $Diam$ is the diameter of the graph.

Iterative increase of the connectivity.

In [89], Wu and Li proposed an iterative scheme to build a $k, m - CDS$ from a $1, m - CDS$. Algorithm 7 is solely based on a lemma stating that adding to a $k-$connected set G a connected set F which can k dominate G, then the union of both set is a $k+1-$connected set.

Lemma 1 (ICGA MAIN LEMMA - *Wu and Li*)**.**
Given a k vertex-connected graph G and a connected set F which can k dominate G, the graph G' composed by $G \cup F$ is $k+1$ vertex-connected.

Algorithm 7: ICGA connectivity augmentation algorithm

```
Input: An 1,m − CDS C_1m of F
Output: A k,m−CDS C_km of G
for i=1 to k-1 do
    if C_im is (i+1)-connected then
        C_(i+1)m ← C_im;
    else
        Find a connected set F_i with a small size which can i dominate C_im;
        C_(i+1)m ← C_im ∪ F_i;
Return C_km;
```

Iterative increase of hop count.

In [90], Wu *et al.* proposed a distributed algorithm named *DDA* that first creates a $m-$dominating set and then augments it until it is $k-$connected.

In order to obtain a $k - connected$ set, the algorithm first elects a leader to be in charge of the following procedure. The chosen node u creates a $k-$connected structure C_2 based on the previously build $m-$dominating DS set in its two-hop neighborhood. If this process is successful, three-hop information is gathered and u tries to create a $k-$connected set based on the C_2 and DS in its three-hop neighborhood. The neighborhood is iteratively extended to the entire network if all the sub-process are successful, i.e. a valid solution can be found. If no solution can be found during one round, a new leader v is chosen and the whole process restarts.

This algorithm suffers from high message and time complexities. This information is summarized in table 4.4.

Common backbone nodes negociation.

In [89], Wu and Li proposed to reduce the message complexity of *DDA* with their algorithm they named *LDA*. The major difference lies in the locality of required knowledge: only two-hop information is necessary to compute a valid solution. *LDA* is composed of three steps to obtain a $k, m-$CDS virtual backbone.

1. Building a $1, m-$CDS $C_{1,m}$ by using the $MDSA$ [89] algorithm. At first, a $1, 1-$CDS $C_{1,1}$ is build and the domination is increased iteratively until m is attained.

2. Negotiation of a common backbone node set S_{cbn} with size $|S_{cbn}| \geq k$ for each node in $C_{1,1}$. This process requires the election of a leader and every node from the $C_{1,1}$ iteratively computes its S_{cbn} with its neighbors from $C_{1,1}$.

3. The last step is the actual building a local $k-$connected set from S_{cbn} and the $C_{1,m}$. This process is done by the nodes in $C_{1,1}$. At the end non-backbone nodes can be greedily added to the final structure if required.

Name	k cases	m cases	Type	Approximation ratio	Time complexity	Message complexity
k-Gossip [32]	any value	$m = k$	other	-	$O(1)$	no msg
k-Grid [32]	any value	$m = k$	other	-	$O(\Delta)$	1 round
k-Coverage [32]	any value	$m = k$	other	-	$O(k\Delta^4)$	2 rounds
DDA [90]	any value	any value	approximation	$\frac{5}{m}(k^2+1)(m+42)$	$O(m\Delta + Diam)$	$O(n\Delta^2)$
MDSA [89]	$k = 1$	any value	approximation	see equation 4.6	$O(m \cdot Diam)$	$O(m(\Delta+1)n)$
LDA [89]	any value	any value	approximation	$\max(\frac{5}{m}, 1) \cdot 2\Delta$	$O((m+\Delta) \cdot Diam)$	$O(n\Delta)$

Table 4.4: Summary of the main characteristics for the distributed algorithms. Δ is the maximum degree of a graph, n the number of nodes, m the domination value and $Diam$ is the diameter of the graph.

4.3 Recap of key points

- We propose an exhaustive and detailed taxonomy for $CDS-$based virtual backbones. For each categories and subcategories, a detailed state-of-the-art is provided for the sake of clarity.
- We presented and motivated six major characteristics to classify the algorithms.
- We proposed three different top-level classifications for centralized, distributed and robust CDS algorithms.
- In the robust CDS hierarchy, we proposed a new way to classify these algorithms based on their way to obtain the desired robustness criterion.
- Recapitulating tables are provided in order to quickly compare all the considered algorithms.

Chapter 5

Dominating Set

Contents

5.1 Definition **70**
5.1.1 Dominating set 70
5.1.2 m-vertex domination property 71
5.1.3 l-level domination property 72
5.1.4 k-connectivity property 72
5.2 State-of-the-art **74**
5.2.1 m,l-Dominating Sets 74
5.2.2 k,m,l-Connected Dominating Sets 75
5.3 Centralized solution **76**
5.3.1 Motivation 76
5.3.2 Difference of convex function techniques 77
5.3.3 Mathematical model 80
5.4 Distributed solutions **86**
5.4.1 Motivations 86
5.4.2 Prerequisites 87
5.4.3 Blackbone algorithm 88
5.4.4 Blackbone2 algorithm 95
5.5 Recap of key points **105**

As stated in [18], the first instances of a dominating set problem arose in the 1850's, well before the advent of wireless networks. The objective of the n queens problem is to find the minimum number of queens that can be placed on a chessboard such that all squares are either attacked or occupied by a queen. This problem was formulated as a dominating set of a graph $G = (V, E)$, with the vertices corresponding to squares on the chessboard, and $(u, v) \in E$ if and only if a queen can move from the square corresponding to u to the square corresponding to v. Fig. 5.1 is a solution of this historical domination problem.

In this chapter we present our algorithmic contributions along with a new optimization problem, the minimum m-level dominating set problem. The first section (see 5.1) provides all the necessary definitions concerning the dominating set problem and its variants. In the second section a state-of-the-art summarizes what has been proposed in the literature to solve this problems (see 5.2). Section 5.3 details our centralized algorithm based on the difference of convex function algorithm. Finally, in section 5.4 two distributed algorithms and some variants are presented.

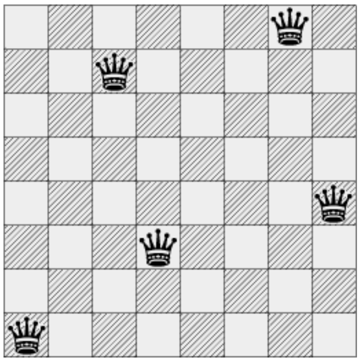


Figure 5.1: A solution for the five queens problem. All squares are either occupied by a queen or under the direct control of one or more queens.

5.1 Definition

The min *m,l-dominating set* problem is a generalization of the original min *dominating set* problem. For the sake of clarity, we first provide a definition of the original problem along with some additional well-known properties. Two additional properties, the $m-$vertex domination and the $l-$level domination, are defined in a second time to complete the definition of the general min *m,l-dominating set* problem.

5.1.1 Dominating set

A dominating set $V' \subseteq V$ of a graph $G = (V, E)$ is a subset of nodes dominating the whole graph, i.e. all nodes not in the dominating set V' have at least one adjacent vertex or neighbor in the set V. Vertices in V' are also referred as dominators or dominating nodes and dominated vertices are referred as dominatees or dominated nodes. See definition 30 for a formal formulation of this concept.

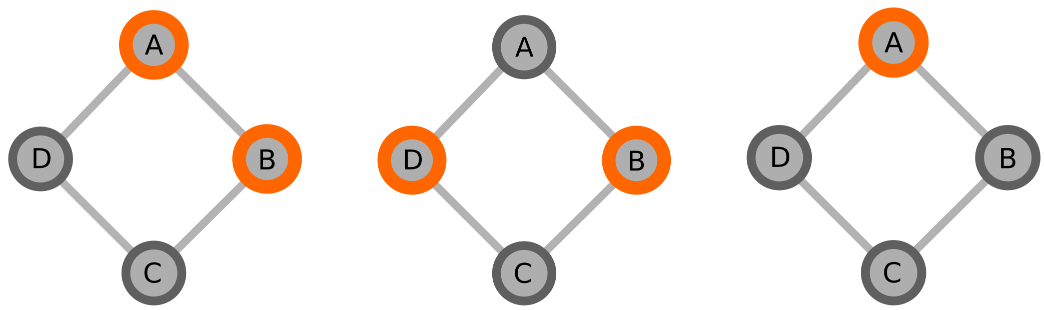
Definition 30 (Dominating Set).
In graph theory, a dominating set for $G = (V, E)$ is a subset $V' \subseteq V$ such that for all $u \in V - V'$ there is at least a $v \in V'$ for which $(u, v) \in E$.

Let us define the domination number $\gamma(G)$ as the number of vertices in the smallest dominating sets for a graph $G = (V, E)$ (see definition 31). The dominating set problem is a classical NP-complete decision problem [41] which is testing $\gamma(G) \leq K$ for a given graph G and input K. The related optimization problem consists in finding the value of $\gamma(G)$. The solution of the min Dominating Set problem, i.e. the set of dominating nodes, may not be unique as illustrated in Figure 5.2.

Results concerning the bounds of $\gamma(G)$ can be found in [50]. Let G be a graph with $n \geq 1$ vertices and let Δ be the maximum degree of the graph. The following bounds on $\gamma(G)$ are known:

- One vertex can dominate at most Δ other vertices; therefore $\gamma(G) \geq n/(1 + \Delta)$.
- The set of all vertices is a dominating set in any graph; therefore $\gamma(G) \leq n$.
- If there are no isolated vertices in G, then there are two disjoint dominating sets in G. Therefore in any graph without isolated vertices it holds that $\gamma(G) \leq n/2$.

Definition 31 (Domination number).
The domination number $\gamma(G)$ is the number of vertices in the smallest dominating sets for a graph $G = (V, E)$.



(a) Dominating set {A,B} (b) 2-vertex dominating set {B,D} (c) 2-level dominating set {A}

Figure 5.2: Bold nodes represent the dominators. Figures 5.2a and 5.2b are two different solutions for the minimum dominating set problem. Figure 5.2b is also a solution for the minimum 2−vertex dominating set problem. Figure 5.2c is an optimal solution for the 2−level dominating set problem.

5.1.2 m-vertex domination property

The notion of $m-$vertex domination has been introduced by Harary *et al.* [49] with the so called double domination ($m = 2$) and then Liao *et al.* extended it to a general value in [59] (see definition 32). Fig. 5.2a and 5.2b provide an optimal solution for both the $m-$vertex dominating set with $m = 1, 2$ for the same instance of graph. Shang *et al.* in [74] referred to this property as the $m-$tuple domination.

Definition 32 (M-VERTEX DOMINATING SET - M-DOMINATION SET).
In graph theory, a subset of nodes D is a $m-$dominating set or a $m-$vertex dominating set of a graph $G = (V, E)$ if and only if every vertex not in D are joined to at least m members of D by some edges.

5.1.3 l-level domination property

The $l-$level domination extends the original domination constraint as follow: it is considered that a node u is dominated by a dominator v if the graph distance (see definition 33) between u and v is at most l. A minimum $l-$level dominating set (see definition 34) with $l = 2$ is illustrated by Figure 5.2c.

Definition 33 (DISTANCE - *Graph Theory*).
The distance between two vertices in a graph is the number of edges in a shortest path connecting them. If there is no path connecting the two vertices, i.e., if they belong to different connected components, then conventionally the distance is defined as infinite.

Definition 34 (L-LEVEL DOMINATING SET).
In graph theory, a $l-$level dominating set for $G = (V, E)$ is a subset $V' \subseteq V$ such that for all $u \in V - V'$ there is at least a $v \in V'$ for which $d_G(u, v) \leq l$.

The general problem can now be easily defined: the minimum $m, l-$ dominating set problem consists in finding the minimum size of $m-$vertex $l-$level dominating set.

Definition 35 (M,L-DOMINATING SET).
A $m-$vertex $l-$level dominating set for $G = (V, E)$ is a subset $V' \subseteq V$ such that for all $u \in V - V'$ there is a subset $V'' \in V'$ such as $|V''| \geq m$ and $\forall v \in V'' : d_G(u, v) \leq l$.

5.1.4 k-connectivity property

In graph theory, two different $k-$connectivity notions can be found. If edges are concerned, it is named $k-$edge connectivity (see definition 37), and if vertices are concerned, it is called $k-$vertex connectivity (see definition of a $k-$vertex connected graph 36). In both cases, these notions define how much edges or vertices can be removed from a connected component until it becomes not connected.

From an ad hoc network point of view, having a $k-$vertex connected virtual backbone is particularly suitable as any $k - 1$ backbone nodes may fail until the backbone is broken. In graph theory, this set of failing nodes is called a separation set (see definition 39). In the specific case $k = 2$, the separation set contains only one node and is often referred as cut vertex or articulation point (see definition 38).

In order to check if a subset of nodes is $k-$vertex connected, some algorithms compute its different *k-blocks* (see definition 40) which are the maximal $k-$vertex connected subgraphs. The subset is $k-$vertex connected if it contains only one $k-$block. Otherwise, some authors trying to create a $k-$vertex connected set are detecting specific $k-$blocks called *k-leaf block* (see definition 41) in order to merge them to another regular block [84].

To conclude this definition section, we add the $k-$vertex connectivity to the $m, l-$DS problem and thus we obtain the $k, m, l-$CDS variant (see definition 42).

Definition 36 (k-vertex connected graph or k-connected graph).
A graph G is k-vertex connected if it is connected and removing any $k-1$ nodes from G will not partition G, i.e., G is still connected.

Definition 37 (k-edge connected graph).
Let $G = (V, E)$ be an arbitrary graph. If $G' = (E \setminus X, V)$ is connected for all $X \subseteq E$ where $|X| < k$, then G is $k-$edge connected.

Definition 38 (Cut-Vertex or Articulation Point).
A cut-vertex or articulation point of a graph $G = (V, E)$ is a vertex $v \in V$, such that $G - v$ has more than one connected component.

Definition 39 (Separation set).
A separation set of a graph $G = (V, E)$ is a set $S \subset V$, such that $G - S$ has more than one connected component. In the special case $|S| = 1$, S is a cut vertex.

Definition 40 (k-Block).
A k-block of a graph is a maximal k-connected subgraph of G that has no separating set. If G itself is k-connected and has no separating set, then G is a k-block.

Definition 41 (k-Leaf Block).
A k-leaf block is a k-block with only one separating set with size of $(k-1)$.

Definition 42 (k,m,l-Connected Dominating Set).
A $k, m, l - CDS$ is a $k-$vertex connected $m-$vertex and $l-$level dominating set.

5.2 State-of-the-art

In this section we provide a state-of-the-art for both $m, l-$DS and $k, m, l-$CDS techniques.

5.2.1 m,l-Dominating Sets

As stated in [51], the domination problem has been studied from the 1950s onwards, but the rate of research on domination significantly increased in the mid-1970s. During the last decade some work has been done in decentralized approaches. These contributions are mainly due to the increasing popularity of ad hoc network research.

As the publications concerning this problem are numerous we propose a classification: exact approaches, approximation and heuristic algorithms and close variants studies.

5.2.1.1 Exact approaches

Many works have been trying to lower the worst case running time. Different classes of graphs have been studied in order to determine the characteristics of the problem $\min DS$. Liedloff proposed in [60] an algorithm to solve the dominating set problem on bipartite graphs in time $O^*(2^{n/2})$. In [38], Fomin *et al.* provided algorithms for split graphs, bipartite graphs and graphs of maximum degree three. Finally in [42], Gaspers *et al.* provided algorithms for chordal graphs, circle graphs and dense graphs. Concerning the general case, the best algorithm has been proposed by Rooij and Bodlaender [83] and its worst case running time is $O(1.5063^n)$. Exact approaches always provide the optimal solutions but the dimension of the problems is not exceeding 100 nodes.

5.2.1.2 Approximation and heuristic approaches

The straightforward approximation algorithm shows a greedy behavior: at each step, the algorithm picks the node that covers the greatest number of uncovered nodes. The corresponding pseudo-code can be found in algorithm 19. This solution is generally referred to as *Greedy-Set-Cover* and its approximation ratio is $\ln(|V|) + 1$. Since then, solutions with constant factor approximation have been proposed for specific classes of graphs. In [30], Dai and Yu proposed a $5 + \epsilon$-approximation algorithm for unit disk graph, a class of graphs used for wireless network modeling. A completely different approach has been developed by Khamis *et al.* in [56] as they designed a randomized algorithm. However their algorithm can only be applied for graphs of maximum degree five.

5.2.1.3 Variants

Many variants have been studied since the mid-1970s and lots of different notions have been proposed such as extended dominating set [88], rainbow domination [21] or even global domination [72]. However one main variant has been extensively studied: connected DS [31, 85] as they can be directly applied to overcome some of the ad hoc networks main problems. Independent dominating sets are also often used as a way to designate clusterhead, i.e. nodes in charge of a subset of nodes. Connected dominating sets are a straightforward way of creating virtual backbones in ad hoc networks and as such, many approximation or heuristic solutions

Algorithm 8: Greedy algorithm for the min $-DS$ problem

Input: Graph $G = (V, E)$

Output: The solution x

```
//Initialization
```
$x \leftarrow (0, \dots, 0)$;

$unCovered \leftarrow E$;

```
//Main loop
```
while $unCovered \neq \emptyset$ **do**

  getMostCoveringNode(curNode);

  $x[curNode] \leftarrow 1$;

  removeCoveredNodes(unCovered, curNode);

return x;

can be found in the literature. An extensive comparison of popular connected dominating set algorithms can be found in [14].

5.2.2 k,m,l-Connected Dominating Sets

In chapter 4, a detailed taxonomy is provided concerning $k, m - CDS$ techniques. Although we introduced the $l-$level domination property to define the general problem, to the best of our knowledge no work has been investigated to obtain structures with $l > 1$.

In a first time (see sections 4.2.1 and 4.2.2), algorithms for the case $k = m = 1$ are presented. Different centralized methods (see section 4.2.1) are introduced: exact (and thus exponential), approximation techniques and heuristics. For $k = m = 1$, interesting distributed algorithms have been then classified in section 4.2.2 depending on their scheme: self-pruning, maximum independent set or multi-point relay.

Robust solutions ($k > 1$ or $m > 1$) are then presented in section 4.2.3. Here again, solutions of the literature have been classified based on their approach: increasing of both domination and connectivity at the same time or specific technique for either the domination or the connectivity.

5.3 Centralized solution

The following section presents one of our major contributions: a centralized algorithm to solve the min $m-$vertex dominating set problem. In a first time we explain our different motivations for designing such an algorithm. The second part provides a short presentation of the difference of convex function techniques (DC) and the DC algorithm (DCA). The mathematical models of the problem considered here and its variants are then detailed extensively along with the centralized algorithm to solve it.

5.3.1 Motivation

Finding such a subset has many applications for the class of real world problems dealing with coverage or secure coverage (for $m \geq 2$). In particular, some approximation and heuristic solutions have been proposed [30, 45, 56] to assess a variety of telecommunication problems, in both infrastructure-based and ad hoc networks. A classical example for the former type of network would be to choose a set of locations to install relay antennas. In ad hoc networks, creating a dominating set is a way to organize the network and is generally used as a first step for generating a connected dominating set [45].

5.3.1.1 Motivation for the centralized approach

Ad hoc networks are intrinsically decentralized entities and as such it is an euphemism to say that applying centralized algorithm is not suitable. Although it may not be a good solution in practice, they generally offer better results as they benefit from the complete graph to decide whether a node should or should not be in the result set or not. As the min dominating set problem and its variants are $NP-$hard problems for most interesting graph classes, it is generally not possible to compare results from a distributed algorithm with the optimal solution if the size of the network is too big. Remaining solutions to assess the quality of a distributed algorithm are:

- Comparing its performances with a centralized algorithm (approximation or heuristic) that is known to provide near-optimal solutions. By doing so, we can assess with confidence if an algorithm behave well or poorly.

- Comparing its performances with other well-known distributed algorithms. This solution may not be sufficient as it is unlikely to provide insights about how far from the optimal solution is the solution of the tested algorithm. Indeed, if our algorithm, say A, is compared to another distributed algorithm B and if A provides solutions 1% smaller than solutions from B, we cannot deduce anything but A is better than B. Maybe 1% is a huge improvement but it could also be negligible.

- Only consider the theoretical performances of the algorithm, i.e. approximation ratio, time complexity or even message complexity. In this case, all algorithms can be easily compared however even if the approximation ratio of an algorithm A, $approx_A$ is smaller than $approx_B$, it does not mean that algorithm A will produce smaller solutions in simulations or in real implementations.

From our point of view, designing centralized algorithm for ad hoc networks is useful for benchmarking purpose with distributed solutions and as a first step. Indeed, in the literature, many authors designed distributed version of their centralized algorithm.

Although this thesis is dealing with virtual backbones in ad hoc networks, minimum dominating set can be used to model many different problems for which centralized approaches are suited or recommended. In Operational Research, this is the case for a class of problems dealing with coverage property, e.g. choosing where to build radio tower to cover a huge area.

5.3.1.2 Motivation for m,l-Dominating Set

As stated previously, dominating sets are very versatile in the MANET context and can be used to model many different situations. The general problem benefits from both $m-$vertex and $l-$level domination constraints and thus is even more adaptable. $m-$vertex domination is an essential property when dealing with fault-tolerance and robustness. It is indeed generally suitable not to consider minimum coverage in real-life scenarios as the quality of such solutions may be greatly diminished by some unexpected failure problem. $l-$level domination permits to model a more flexible concept of domination. This property can be particularly well combined with the a $m-$vertex domination that may be too constrained.

5.3.1.3 Motivation for using DC programming and DCA

Many contributions have been proposed through the last decades to solve the minimum dominating set problem. Although these solutions are providing insightful results some not yet used techniques may prove to be efficient. Centralized algorithms are generally based on simple heuristics or greedy mechanisms to create their solutions as finding the optimal solution for the considered problems is NP-Hard and thus non-tractable. DC programming and DCA is an innovative approach in nonconvex programming and the Operational Research domain. It is characterized by its high efficiency and a proved convergence. This technique permits to address non-convex optimization problems via a decomposition of the objective function into two convex functions and to solve it via an iterative DC algorithm. Although this method is designed for continuous solution space, results can be constrained to boolean values using the exact penalty technique for which many encouraging results have been obtained [6, 7, 10, 12]. As a consequence, we decided to apply this method in order to find compact dominating set based virtual backbones.

5.3.2 Difference of convex function techniques

Main idea.

DC programming and DCA introduced by Pham Dinh Tao in 1985 and extensively developed by Le Thi Hoai An and Pham Dinh Tao since 1994 (see [5, 8, 9, 11, 79, 80] and references therein) is an efficient approach for nonconvex continuous optimization. They address a general DC program of the form:

$$\alpha = \inf\{\mathcal{F}(x) := g(x) - h(x) \ : \ x \in \mathbb{R}^n\} \ (P_{dc}) \tag{5.1}$$

where g and h are lower semi-continuous proper convex functions on $\mathbb{R}^n$. Such a function $\mathcal{F}$ is called DC function, and $g-h$, DC decomposition of $\mathcal{F}$ while the convex functions g and h are DC components of $\mathcal{F}$. If g or h are polyhedral convex functions then (P_{dc}) is called a polyhedral DC program. It should be noted that a constrained DC program whose feasible set C is convex can always be transformed into an unconstrained DC program by adding the indicator function χ_C of C to the first DC component g:

$$\chi_C(x) := \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{otherwise.} \end{cases}$$

Let $g^*(y)$ defined by:

$$g^*(y) = \sup\{\langle x, y\rangle - g(x) \; : \; x \in \mathbb{R}^n\}$$

be the conjugate function of g. Then the so-called dual program of (P_{dc}) is defined by:

$$\alpha_D = \inf\{h^*(y) - g^*(y) : y \in Y\}. \quad (D_{dc})$$

One can prove that $\alpha = \alpha_D$ and there is a perfect symmetry between primal and dual DC programs: the dual of (D_{dc}) is exactly (P_{dc}).

Relaxing optimal globality.

The complexity of DC programs resides, of course, in the lack of practical optimal globality conditions. We developed instead the following necessary local optimality conditions for DC programs in their primal part, by symmetry their dual part is trivial (see [5, 8, 9, 79, 80], and references therein):

$$\partial h(x^*) \cap \partial g(x^*) \neq \emptyset \tag{5.2}$$

(such a point x^* is called *critical point* of $g-h$ or for (P_{dc})), and

$$\emptyset \neq \partial h(x^*) \subset \partial g(x^*). \tag{5.3}$$

The condition (5.3) is also sufficient for many classes of DC programs. In particular it is sufficient for the next cases quite often encountered in practice:

1. In polyhedral DC programs with h being a polyhedral convex function (see [5, 79] and references therein). In this case, if h is differentiable at a critical point x^*, then x^* is actually a local minimizer for (P_{dc});

2. In case the function $\mathcal{F}$ is locally convex at x^* ([9]).

The transportation of global solutions between (P_{dc}) and (D_{dc}) is expressed by:

Property 1.

$$[\cup_{y^*\in\mathcal{D}} \partial g^*(y^*)] \subset \mathcal{P}, \;\; [\cup_{x^*\in\mathcal{P}} \partial h(x^*)] \subset \mathcal{D} \tag{5.4}$$

where $\mathcal{P}$ and $\mathcal{D}$ denote the solution sets of (P_{dc}) and (D_{dc}) respectively.

Under technical conditions, this transportation holds also for local solutions of (P_{dc}) and (D_{dc}). For example (see [9, 79] for more information):

Property 2.

Let x^* be a local solution to (P_{dc}) and let $y^* \in \partial h(x^*)$. If g^* is differentiable at y^* then y^* is a local solution to (D_{dc}). Similarly, let y^* be a local solution to (D_{dc}) and let $x^* \in \partial g^*(y^*)$. If h is differentiable at x^* then x^* is a local solution to (P_{dc}).

Based on local optimality conditions and duality in DCA, the idea of DCA is quite simple: each iteration k one linearizes the concave part $-h$ and then solve the resulting convex program. More precisely, DCA consists of computing at each iteration k:

$$\begin{aligned} y^k &\in \partial h(x^k) \\ x^{k+1} &\in \arg\min_{x \in \mathbb{R}^n} \{g(x) - h(x^k) - \langle x - x^k, y^k \rangle\}. \ (P_k) \end{aligned}$$

First of all, it is interesting to mention that our works involve the convex DC components g and h but not the DC function $\mathcal{F}$ itself. Moreover, a DC function $\mathcal{F}$ *has infinitely many DC decompositions which have crucial impacts on the qualities* (speed of convergence, robustness, efficiency, globality of computed solutions,...) of DCA. For a given DC program, the choice of *optimal* DC decompositions is still open. Of course, this depends strongly on the very specific structure of the problem being considered. In order to tackle the large scale setting, one tries in practice to choose g and h such that sequences $\{x^k\}$ and $\{y^k\}$ can be easily calculated, i.e. either they are in explicit form or their computations are inexpensive.

Convergence properties.

Convergence properties of the DCA and its theoretical basis are described in [5, 8, 9, 79, 80]. However, it is worthwhile to report the following properties that are useful in the next paragraphs (for the sake of simplicity, we omit here the dual part of these properties):

- DCA is a descent method *without line search*, say the sequence $\{g(x^k) - h(x^k)\}$ is decreasing.
- If $g(x^{k+1}) - h(x^{k+1}) = g(x^k) - h(x^k)$ then x^k is a critical point of $g - h$. In this case, DCA terminates at k^{th} iteration.
- If the optimal value α of problem (P_{dc}) is finite and the infinite sequence $\{x^k\}$ is bounded, then every limit point $\widetilde{x}$ of this sequence is a critical point of $g - h$.
- DCA has a linear convergence for general DC programs. Especially, for polyhedral DC programs the sequences $\{x^k\}$ contains finitely many elements and the algorithm convergences to a solution in a finite number of iterations.

For a complete study of DC programming and DCA the reader is referred to [5, 8, 9, 79, 80] and references therein. The solution of a nonconvex program by DCA must be composed of two stages:

- the search of an *appropriate* DC decomposition;
- the search of a *good* initial point.

5.3.3 Mathematical model

In this subsection we present some mathematical models for the $\min -DS$ problem and its variants. In 5.3.3.1 we provide the well-known integer programming model for the min-DS problem. A straightforward extension of the original model is proposed in 5.3.3.2 to take into account the $m-$vertex domination. A generic scheme to design a model for a particular value of $l-$level domination concludes this subsection in 5.3.3.3.

5.3.3.1 min DS model

Let $G = (V, E)$ be the input graph with $|V| = n$. The edge set E is modeled by an adjacency matrix $A = (a_{i,j})_{i=1\dots n, j=1\dots n}$. $a_{i,j}$ values are defined as follow:

$$a_{i,j} = \begin{cases} 0 & \text{if no edge exists between nodes } i \text{ and } j \\ 1 & \text{if an edge exists between nodes } i \text{ and } j \\ 1 & \text{if } i = j. \end{cases}$$

Notice that $a_{i,j} = a_{j,i}$ as we are considering non-oriented graphs only. Moreover, in the case $i = j$, it is assumed that $a_{i,j} = 1$ as it helps creating a more elegant model. The main objective of the problem is to select a subset of nodes with minimum cardinality. To such extend, let $x = (x_i)_{i=1\dots n}$ be the dominating set and our unique variable in this model. Each component of this variable represents the state of a particular vertex of the graph and is coded as follow:

$$x_i = \begin{cases} 0 & \text{if } x_i \text{ is not in the dominating set} \\ 1 & \text{if } x_i \text{ is in the dominating set.} \end{cases}$$

The equation 5.8 is the objective function of the problem as we want to minimize the cardinality of the dominating set. The variable x represents an optimal and valid solution, as the solution set may not be a singleton. The equation 5.9 takes care of the domination constraints. Indeed if a node i is a dominator ($x_i = 1$), as $a_{i,i} = 1$, the sum returns at least 1 and the constraint is not violated. For the dominated node case ($x_i = 0$), there has to be at least one direct neighbor j of i ($a_{i,j} = 1$) which is in the dominating set ($x_j = 1$). Notice that both cases can exist for some node, i.e. they can be in the dominating set and have one or more dominators in their direct neighborhood.

$$Obj \quad \min \textstyle\sum_{i=1}^{n} x_i \tag{5.5}$$

$$St \quad \forall i \in 1 \dots n \quad \sum_{j=1}^{n} a_{i,j}\, x_j \geq 1 \tag{5.6}$$

$$x \in \{0;1\}^n. \tag{5.7}$$

5.3.3.2 $m-$vertex domination constraint

Obtaining a $m-$vertex dominating set can be modeled by slightly modifying the domination constraint. Indeed, on the right side of equation 5.9 we now take into account the fact that a dominator node does not need to be dominated to ensure a valid solution.

$$\min \sum_{i=1}^{n} x_i \tag{5.8}$$

$$s.t. \quad \sum_{j=1}^{n} a_{i,j}\ x_j \geq m(1 - x_i), \forall i = 1 \dots n \tag{5.9}$$

$$x \in \{0; 1\}^n. \tag{5.10}$$

5.3.3.3 Generic scheme for $l-$level domination

To the best of our knowledge, no model for the min $l-$level dominating set problem exists. In this part we propose a generic scheme to create a model for any value of l.

Case l=2.

In this problem any node i can be into three non-exclusive situations. First it can be a dominator and thus $x_i = 1$. As in the previous model, node i can have a dominator j in its direct neighborhood ($a_{i,j} = 1$ and x_j). The last case for $l = 2$ is when node i has a dominator k and $d_G(i, k) = 2$, i.e. $\exists j, k/a_{i,j} = 1,\ a_{j,k} = 1,\ x_k = 1$.

The model for the min $2-$level dominating set is then as follow:

$$\begin{array}{ll} Obj & \min \sum_{i=1}^{n} x_i \\ St & \forall i \in 1 \dots n \quad \sum_{j=1}^{n} a_{i,j}\ x_j + \sum_{j=1}^{n} \sum_{k>j}^{n} a_{i,j} a_{j,k} x_k \geq 1 \\ & x \in \{0; 1\}^n. \end{array}$$

An interesting characteristic of this model is that the constraints remain linear.

General case l.

From the previous case we can derive a general scheme that permits us to create a model for any value of l. However, in practical scenarios the following constraints may not be suitable for large values of l. The following equation represent the constraint for a given value l. Such a constraint has to be satisfied for any i, i.e. a dominator is at a maximum distance l of node i.

$$\begin{aligned} \forall i \in 1 \dots n \quad & \sum_{j=1}^{n} a_{i,j}\ x_j \quad + \\ & \sum_{j=1}^{n} \sum_{k>j}^{n} a_{i,j} a_{j,k} x_k \quad + \\ & \dots \quad + \\ & \sum_{j=1}^{n} \sum_{k>j}^{n} \dots \sum_{z>y}^{n} a_{i,j} a_{j,k} \dots a_{y,z} x_z \geq 1. \end{aligned}$$

5.3.3.4 The DC formulation

In this part we explain how to apply the difference of convex function technique to our mathematical model. The first step of the methodology is to transform the genuine integer programming model into a DC decomposition. Both DC components g and h have to be convex functions.

Let us rewrite the original problem in a more compact form (see equation 5.11):

$$\alpha = \min \left\{ \langle e, x \rangle : Ax \geq 1, x \in \{0;1\}^n \right\}, \tag{5.11}$$

where e is the vector of ones in $\mathbb{R}^n$.

The DC technique works in continuous space only and our problem is characterized by discrete variables. This problem is overcome by using the exact penalty, a methodology to constraint some chosen variables into discrete values. The details concerning this method can be found in [11].

As mentioned before, the DC algorithm is working in continuous space. As a matter of fact our variable is now defined as: $x = (x_i)_{i=1\ldots n}$ and $\forall i = 1 \ldots n, x_i \in [0;1]$. As we still want to obtain a $0-1$ solution we use the exact penalty technique to avoid non-integer solutions. To such extend, let us consider the function p and the bounded polyhedral convex set K defined by:

$$p(x) = \sum_{i=1}^{n} x_i(1-x_i), \quad \text{and}$$
$$K = \{x \in \mathbb{R}^n : Ax \geq 1, x \in [0,1]^n\}.$$

Notice that p is equal to zero if and only if all x_i are either 0 or 1. Moreover, p is finite and concave on $\mathbb{R}^n$ and non-negative on K. By using p and K we obtain the following equality:

$$\Big\{ x \in \mathbb{R}^n : Ax \geq 1,\ x \in \{0,1\}^n \Big\} = \Big\{ x \in K : p(x) \leq 0 \Big\}.$$

Hence, problem (5.11) can be rewritten as:

$$\alpha = \min\{\langle e, x\rangle : x \in K,\ p(x) \leq 0\}.$$

In [11] the following theorem can be found:

Theorem 1 (Exact Penalty).
Let K be a nonempty bounded polyhedral convex set, f be a finite concave function on K. Then there exists $t_0 \geq 0$ such that for all $t > t_0$ the following problems have the same optimal value and the same solution set:

$$\begin{aligned} (P) \quad & \alpha && = \inf\{f(x) : x \in K, p(x) \leq 0\} \\ (P_t) \quad & \alpha(t) && = \inf\{f(x) + tp(x) : x \in K\}. \end{aligned}$$

According to theorem 1, for a sufficiently large number t, the problem (5.11) is equivalent to

$$\begin{aligned} & \alpha(t) && = \min\{\mathcal{F}(x) := \langle e, x\rangle + tp(x) : x \in K\} \\ \Leftrightarrow \quad & \alpha(t) && = \min\{\mathcal{F}(x) := \langle (t+1)e, x\rangle - t\|x\|^2 : x \in K\}. \end{aligned} \tag{5.12}$$

The last problem is a concave quadratic program and we adopt the following DC decomposition of $\mathcal{F}$:

$$\alpha = \inf\{\mathcal{F}(x) := g(x) - h(x) \ : \ x \in \mathbb{R}^n\},$$

with

$$\begin{aligned} g(x) &= \chi_K(x), \\ h(x) &= -\langle (t+1)e, x\rangle + t\|x\|^2. \end{aligned}$$

Both functions g and h are convex which implies that (5.12) is a DC program. More precisely (5.12) is a polyhedral DC program since g is a polyhedral convex function.

5.3.3.5 The DC Algorithm

Now we create the DC algorithm applied on the previous DC program. As previously explained in section 5.3.2, DCA consists in computing at each iteration k:

$$\begin{aligned} &y^k \in \partial h(x^k) \\ &x^{k+1} \in \arg\min_{x\in K}\{g(x) - h(x^k) - \langle x - x^k, y^k\rangle\}. \end{aligned}$$

By the very definition of h, we have

$$\nabla h(x^k) = -(t+1)e + 2tx^k. \tag{5.13}$$

Finding x^{k+1} consists in solving the following convex program after removing useless constants:

$$\begin{aligned} & x^{k+1} \in \arg \ \min_{x\in K}\{g(x) - \langle x, y^k\rangle\} \\ \Leftrightarrow \ & x^{k+1} \in \arg \ \min\{\langle (t+1)e - 2tx^k, x\rangle : x \in K\} \end{aligned}$$

Algorithm 9 describes the DCA for solving problem (5.12).
The convergence of this DCA is proven by the following theorem:

Theorem 2 (Convergence properties of DCA)**.**

(i) DCA generates a sequence $\{x^k\}$ contained in the vertex set of K (denoted $V(K)$) such that the sequence $\{\mathcal{F}(x^k) := \langle e, x^k\rangle + tp(x^k)\}$ is decreasing.

(ii) The sequence $\{x^k\}$ converges to a critical point $x^* \in V(K)$ of (5.12) after a finite number of iterations.

(iii) Moreover, the point x^* is almost always a local minimizer of Problem (5.12).

(iv) For a sufficiently large number t , if at an iteration r we have $x^r \in \{0,1\}^n$, then $x^k \in \{0,1\}^n$ for all $k \geq r$.

Algorithm 9: DCA to solve the $\min m - DS$ problem

Input: Graph $G = (V, E)$

Output: The solution x^k

// Initialisation

$k \leftarrow 0$;

Compute x^0 by solving the relaxed problem;

// Main loop

end $\leftarrow$ false;

repeat

  $y^k \leftarrow 2x^k t$;

  $x^{k+1} \leftarrow \arg\min\{\langle (t+1)e - 2tx^k, x\rangle : x \in K\} \quad (PL)$;

  if $\|x^{k+1} - x^k\|^2 = 0$ **then**

    end $\leftarrow$ true;

  $k \leftarrow k + 1$;

until *end* ;

return x^k;

Proof. (i) and (ii) are direct consequences of the convergence properties of a polyhedral DC program. Only (iii) and (iv) need a proof.

We first note that since g is a polyhedral convex function, so is g^*. It follows that y^*, the limit point of the sequence $\{y^k\}$, is almost always a local solution to the dual DC program of (5.12). Using Property 2 and taking into account the fact that h is differentiable everywhere, we conclude that x^* is almost always a local solution to Problem (5.12).

Let now

$$t > t_1 := \max\left\{\frac{e^T x - \eta}{\theta} : x \in V(K),\ p(x) \leq 0\right\},$$

where $\eta := \min\{e^T x : x \in V(K)\}$ and $\theta := \min\{p(x) : x \in V(K)\}$.
Let $\{x^k)\} \subset V(K)$ $(k \geq 1)$ be generated by DCA. If $V(K) \subset \{0,1\}^n$, then the assertion is trivial. Otherwise, let $x^r \in \{0,1\}^n$ and $x^{r+1} \in V(K)$ be an optimal solution of the linear program (PL). Then from (i) of this theorem we have

$$e^T x^{r+1} + tp(x^{r+1}) \leq e^T x^r + tp(x^r).$$

Since $p(x^r) = 0$, it follows

$$tp(x^{r+1}) \leq e^T x^r - e^T x^{r+1} \leq e^T x^r - \eta.$$

If $p(x^{r+1}) > 0$, then

$$t \leq \frac{e^T x^r - e^T x^{r+1}}{p(x^{r+1})} \leq \frac{e^T x^r - \eta}{\theta} \leq t_1$$

which contradicts the fact that $t > t_1$.
The proof is then complete.

In order to increase the exploration of the solution space and thus obtain better solutions, we propose a restarting mechanism. When the previous DC Algorithm converges to a non-integer solution, the restarting mechanism relaunch the same procedure from another starting point. Algorithm 10 provides the details of the implementation.

Algorithm 10: DCA with restart mechanism

Input: Graph $G = (V, E)$
Output: The solution x^k

`// Initialisation`
$k \leftarrow 0$;
Compute x^0 by solving the relaxed problem;
`// Main loop`
end $\leftarrow$ `false`;
repeat
    $y^k \leftarrow 2x^k t$;
    $x^{k+1} \leftarrow \arg\min\{\langle (t+1)e - 2tx^k, x\rangle : x \in K\}$;
    if $\|x^{k+1} - x^k\|^2 = 0$ **then**
        if x^{k+1} *is not a* $0-1$ *solution* **then**
            Fix non $0-1$ components to 1;
        else
            end $\leftarrow$ `true`;
    $k \leftarrow k + 1$;
until *end* ;
return x^k;

5.4 Distributed solutions

In this section we propose two distributed algorithms to create $CDS-$based virtual backbones. In a first time we present the different aspects that motivates our work. Prerequisites notions are introduced in a second part followed by the two proposed algorithms, namely Blackbone1 and Blackbone2.

5.4.1 Motivations

In this part we provide the motivations related to the design characteristics of the two distributed algorithms that we proposed. In a first time we briefly summarize why decentralization is critical. In the next paragraphs we provide the necessary explanations about our design choices and our hypothesis concerning the network.

Motivation for decentralization.

In ad hoc networks, applying a centralized algorithm means gathering the complete network topology into one network node so that it could compute a virtual backbone for the whole network and then broadcast its result. Such a scheme is not adapted to ad hoc networks for the following scalability reasons. First, this specific node has to be chosen or elected somehow. This could be envisaged for static and small size networks as the complete topology graph can be handled but this process does not scale when the number of network nodes increases or the nodes move. The second problem is also about the scalability of the computing problem. Ad hoc network nodes are generally characterized by low computation capacities and thus applying an algorithm on a large topology graph may not be tractable for a particular node. Indeed, applying the algorithm may require too much resources for a single node or the required time to compute a solution may not be adapted to a changing topology. The third and last scalability issues is about mobility implications. As ad hoc networks are generally composed of moving battery-based nodes, the chosen node is unlikely to gather a valid topology graph at a time t to compute a corresponding solution. If the computing node managed to gather a valid topology at time t, it is also unlikely that the solution would be broadcasted on time, i.e. at the time the solution reaches the nodes, the topology may have change already.

Distributed algorithms are suitable for ad hoc network as they are intrinsically distributed entities. No specific node has to be elected to apply an algorithm as all nodes are participating. Moreover nodes are generally computing a solution based on local information only which is obviously smaller than the complete topology graph. Finally gathering up-to-date local topology data is more scalable and thus the distributed solution is more likely to produce up-to-date solution.

Motivation for localization.

For our two distributed contributions, we chose to rely on two-hop information only, i.e. each node knows its surroundings until its neighbors of neighbors. This information will be exchanged via periodical packets containing the identifier of the sender and two lists: its neighbors identifier and state (backbone or non-backbone node). No global information such as average node density or the number of nodes is required which is an important point when real implementations are targeted. This type of information is required in [32] and thus reduces the practical usage of *k-Gossip* and *k-Grid* algorithms to a priori known networks.

Motivation for asynchronicity.

Most of the proposed distributed algorithms rely on an implicit synchronization of the network as they are composed of steps. It is generally assumed for such contributions that the algorithm is started on every nodes and that step k begins for all nodes when all of them finished step $k-1$. Such constraints are not scalable as it induces huge quantity of message exchanges to synchronize all nodes.

Asynchronous protocols are more suitable as they do not rely on synchronization and thus can take decisions without waiting for other nodes. Such characteristic is essential when dealing with spontaneous networks in which the participating network nodes are always moving and/or changing, i.e. new nodes enter the network and some others leave it.

Motivation for fault-tolerance.

Fault-tolerance is an important characteristic for virtual backbones as their main objective is to maintain a structure on top of a mobile network. In clustering virtual backbones, selecting the best nodes based on some quality measures is a well-established technique. Those measures generally aggregate various pieces of information such as remaining battery level, number of neighbors or relative mobility to determine which nodes are more suitable for the backbone. This approach relies on common sense but cannot guaranty the efficiency of the choices, i.e. some nodes may fulfill all the desired characteristics and yet fail right after it has been chosen.

As an opposite philosophy, in our work we chose to increase the vertex domination and the vertex-connectivity of the generated virtual backbones to make it more robust. These two features, when correctly applied, ensure that the failure of some backbone nodes leaves the structure in a valid state, i.e. connected and dominating.

5.4.2 Prerequisites

In the remaining of this chapter, two distributed and localized algorithms are detailed. In such a context, network nodes benefit from a limited and egocentric knowledge of their own environment in general and of their neighbors in particular. Let us assume that v is the considered network node, i.e. the node in the center of the local graph $G(v) = (V(v), E(v))$. $V(v)$ is the vertex set containing all the nodes known by v including itself and $E(v)$ are the available wireless communication channels between two network nodes of $V(v)$.

Our two distributed algorithm only require two-hop knowledge to compute a solution, i.e. vertices $w \in V(v)$ are either one-hop or direct neighbors, two-hop neighbors or neighbors of neighbors, or v itself. Let $N_1(v)$ be the one-hop neighborhood of the network node v and $N_2(v)$ its 2-hop neighborhood. Let $N_{1\cup 2}(v) = N_1(v) \cup N_2(v) = V(v)$ be the complete neighborhood of node v. Let $CN(x,y)$ be the common neighbors of two different network nodes x and y, i.e. $CN(x,y) = \Big\{w \in V(x)/(x,w) \in E(x) \wedge (y,w) \in E(y)\Big\}$.

In our two distributed algorithms, two colors are used to represent the state of a given network node. Black means "backbone member" and white "not backbone member". Black nodes regularly check if they are still needed to guarantee the connectivity or the vertex-domination of the backbone, and white nodes periodically check if they are required to fulfil the connectivity or the vertex-domination conditions. Let $N_1^B(v)$ be the black or backbone nodes in the 1-hop neighborhood of node v. Similarly, let $N_2^B(v)$ be the black nodes in the 2-hop neighborhood of node v and $N_{1\cup 2}^B(v) = N_1^B(v) \cup N_2^B(v)$ be the complete black neighborhood of node v. Finally, let $CN^B(x, y)$ be the common black neighbors of x and y.

5.4.3 Blackbone algorithm

Blackbone is a distributed localized algorithm which is generating $k-$vertex connected dominating sets only. From the general problem point-of-view, Blackbone is only considering the case with one-vertex and one-level domination. Increasing the vertex domination is successfully applied in the second version of the algorithm, Blackbone 2, described in 5.4.4. The $l-$level domination constraint has not yet been envisaged for our distributed contributions as it may require to increase the scope of the local graph, i.e. increasing the number of hops and thus increase the message complexity.

The presentation of Blackbone is divided in four parts. First, the so-called *robustness property*, responsible for the control of the vertex-connectivity is introduced and its validity is proven. Based on the previous property and some observations, four subroutines and the main loop of the algorithm are detailed in the second part. A theoretical study of the algorithm (validity of the solution, time and message complexity and approximation factor) is then provided in a third time and practical issues related to the concurrent nature of wireless simulations are introduced along with their potential solutions.

5.4.3.1 Vertex connectivity

Robustness property.

In order to increase the vertex connectivity of the generated virtual backbone, the so-called robustness property is introduced. The underlying idea is simple: for a given node v, we would like to ensure that each two-hop node can be reached by at least k distinct direct (one-hop) neighbors. As we can see in Figure 5.3, x and y are two-hop neighbors of v. v can send messages to x if either a or b act as relay node. Similarly, y can be accessed via either b or c. Such a robust configuration is what we want to have thanks to the *robustness property*. Indeed, it induces that even if one direct neighbor disappears, e.g. a, v can still reach all its 2-hop neighbors. The robustness property is formalized in definition 1.

Property 1 (ROBUSTNESS PROPERTY - *Blackbone*).
For a network node v and a predefined connectivity value k, the local graph $G(v)$ contain a robust $k-$connected virtual backbone if:

$$\forall x \in N_2^B(v)\ ,\quad \left| CN^B(v, x) \right| \geq k$$

From this local property, we would like to obtain a global or network-wide one. To such extend we propose lemma 2 which is simply stating that if all nodes of the network are

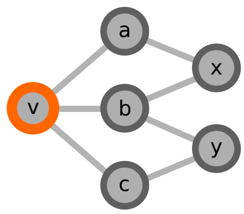


Figure 5.3: Considering node v point of view, all its 2-hop neighbors x, y can be reached by at least two different nodes a, b for x and b, c for y

characterized by a robust local graph in the sense of definition 1, then the complete network graph is supported by a robust virtual backbone.

Lemma 2 (Global robustness).
Let $G = (V, E)$ be the complete communication graph and V^B the backbone nodes. If all the nodes of the backbone verify the robustness property for a connectivity value k, then between any two nodes $x, y \in V^B$ there are at least k independent or node-disjoint paths.

Let us prove lemma 2 by recurrence. Let us assume that in graph $G = (V, E)$ all nodes verify the robustness property for a connectivity value $\geq k$. Let V^B be the set of black vertices of G. Let us prove that for any pair of nodes $(x, y) \in V^B$ k node-disjoint paths only composed of black nodes can be found to link them.

Initial case: x and y are both black and 2-hop neighbors.

As x respects the robustness property and y is a two-hop neighbor of x, we have:

$$\exists S \subset N_1^B(x) \,/\, |S| \geq k \wedge \Big(\forall u \in S, u \in CN^B(x, y)\Big).$$

As a consequence, between x and y they are at least k independent or node-disjoint paths: $\forall u \in S,\ (x, u, y)$ is a path composed of black nodes only.

Recurrence: Assume n and prove $n + 1$.

First, let us assume that for every pair of black nodes x, y such as x and y are less or equal to $n-$hop neighbors, there are k independent paths $P_1(x, y), P_2(x, y), ..., P_k(x, y)$ to link them. Let $L = \{l_1, l_2, ..., l_k\}$ be the last node of $P_1(x, y), P_2(x, y), ..., P_k(x, y)$ and let the black node d be a direct neighbor of y, i.e. $w \in N_1^B(y)$. Two cases may happen:

- $\exists\ l \in L \ / \ w \in N_1(l)$. In such a scenario w is neighbor with one of the last nodes of the set L. As a consequence, $d_G(x, w) \leq d_G(x, y)$. For the recurrence sake, we assumed that two black nodes at a distance less or equal to $n-$hops can be connected via at least k independent paths. The previous assumption directly induces that x and w are connected via k independent paths.

- $\forall\ l \in L\ /\ w \notin N_1(l)$. In this case, w is a two hop black neighbor of all the nodes in L. This property directly induces that w can be a $(n+1)-$hop neighbor of x or less. The recurrence takes care of the case where w is a $n-$hop neighbor of x or less. The robustness property tells us that k independent paths exist between $l \in L$ and d. From this we deduce the following:

$$
\begin{aligned}
\exists\ r_{1,1}, \dots, r_{1,k} \quad / \quad & (r_{1,1}, \dots, r_{1,k} \in CN^B(l_1, w)) \\
\exists\ r_{2,1}, \dots, r_{2,k} \quad / \quad & (r_{2,1}, \dots, r_{2,k} \in CN^B(l_2, w)) \\
& \dots \\
\exists\ r_{k,1}, \dots, r_{k,k} \quad / \quad & (r_{k,1}, \dots, r_{k,k} \in CN^B(l_k, w)).
\end{aligned}
$$

The black nodes $r_{1..k,1..k}$ are relay nodes between $l_{1..k}$ and w. Additionally, we assume that $r_{i,1} \neq r_{i,2} \neq \dots \neq r_{i,k} \forall i \in \{1..k\}$. In the worst case scenario in the sense of paths diversity, we may have $r_{1,i} = r_{2,i} = \dots = r_{k,i}\ \forall i \in \{1..k\}$. However, even in that case k independent paths $P_1(x,w), P_2(x,w), \dots, P_k(x,w)$ exist between x and w. Indeed, in such a case, $(x, \dots, l_1, r_{1,1}, w), (x, ..., b, r_{2,2}, w), \dots (x, \dots, l_k, r_{k,k}, w)$ are independent paths. $\square$

5.4.3.2 Blackbone subroutines

Based on this proof, we developed four simple rules to create k-vertex connected dominating set. Two rules are designed to add nodes to the backbone (*re-initialization* and *construction*), and the last two rules permit to remove useless nodes (*normal pruning rule* and *clique pruning rule*. A main loop is aggregating these subroutines to obtain the final Blackbone algorithm.

Re-initialization rule.

The initial state of a node is black, i.e. we consider the backbone state as the default setting. As a consequence, when a white node is isolated (no neighbors at all whatever their color) due to mobility or node failure, it has to go back to the initial black state. This simple rule guarantees the coherence of the structure: an isolated node is considered a one-node backbone. The re-initialization rule can be formalized as follow:

Rule 11 (Re-initialization rule - *Blackbone*)**.**
A node v creates a one node backbone by changing its color to black if and only if $|N_1(v)| = 0$.

Construction rule.

The construction rule is based on the robustness property. Its objective is to check if a node is required to achieve the $k-$vertex connectivity in its local graph. In Blackbone, one of our main design property was to consider that a particular network node should change its state based only on its local knowledge, i.e. a node should not force one of its neighbor to enter the virtual backbone. As such, we need to change the considered point-of-view in the robustness property: a specific node v checks if it can help connecting two of its direct black neighbors. To such extend, let us consider S_v, the set of unconnected one-hop black neighbors of a node v defined as follow:

$$S_v = \Big\{(x, y) \in N_1^B(v) \ / \ \Big(x \notin N_1^B(y) \wedge y \notin N_1^B(x)\Big)\Big\}.$$

For a given couple of nodes $(x, y) \in S_v$, we compute the set of their common black neighbors, i.e. the set of black nodes that may relay messages from x to y:

$$CN^B(x, y) = N_{1\cup 2}^B(x) \cap N_{1\cup 2}^B(y).$$

The considered node v has to be part of the backbone if there exists at least one couple of neighbors $(x, y) \in S_v$, which are connected by less than k black nodes. From this statement we derive the construction rule (see rule 12).

Rule 12 (Construction rule - *Blackbone*)**.**
A node v should change its color to black and thus be part of the backbone if and only if the following proposition is true:

$$\exists (x, y) \in S_v \ / \Big(|CN^B(x, y)| \leq k - 1\Big).$$

Normal pruning rule.

The normal pruning rule is the reverse operation of the construction rule. Indeed, it checks if a node v can get out of the backbone by detecting that every couple of node $(x, y) \in S$ has enough common black neighbors. Obviously, in such a case, node v is not needed anymore as it creates redundancy. As a consequence, node v can change its color to white. Rule 13 summarizes this subroutine in a more formal way.

Rule 13 (Normal pruning rule - *Blackbone*)**.**
A node v should go out of the backbone and thus changing its color to white if and only if the following proposition is true:

$$\forall (x, y) \in S_v \ / \Big(|CN^B(x, y)| > k\Big).$$

Clique pruning rule.

The normal pruning rule focuses on couples of unconnected neighbors only. If all the neighbors are connected, the previous rule cannot be applied. This particular case is called clique, i.e. a graph in which every vertex is connected to every other vertex. In this case, only one node is needed to create a CDS and maximum two nodes for 2-CDS. In a more formal way, a node v gets out of the backbone if and only if:

$$\forall (x, y) \in N_1^B(v), x \in N_1^B(y) \wedge y \in N_1^B(x).$$

Main loop.

The main loop, presented in Algorithm 11 is simply aggregating the four subroutines and apply them depending on the current state of the considered node. Indeed, if a node u is black, the main loop checks if u is still required to fulfil the desired robustness property. As a consequence, when a node is black, the algorithm will process the normal and the clique pruning rule. If one of these two rules correspond to the current situation of node u then its color is changed for white, otherwise the node u stays in the backbone until the next time step. Similarly, if u is white, the main loop first it checks if it is alone. If u has neighbors, the algorithm checks if it can connect two unconnected backbones, i.e. if node u can help increasing the vertex-connectivity of the virtual backbone. If u can be useful, the main loop finally checks if it can reinforce the backbone. Algorithm 11 is executed independently on every node. Each network node possesses its own timer to repeat the main loop until the simulation reaches its time limit. As a consequence, depending on when the nodes are switched on or activated, they may not process the main loop at the same time. This characteristic is particularly suitable as synchronization is difficult to obtain in ad hoc networks.

Algorithm 11: Blackbone main loop

```
for every time step do
    if node is white then
        if reinitialization() or construction() then
            change color to black;
    else
        if normalPruningRule() or cliquePruningRule() then
            change color to white;
```

5.4.3.3 Practical issues

The main design consideration of the Blackbone algorithm is its applicability in realistic simulated environment, i.e. the algorithm should be able to work in an asynchronous and concurrent manner. As a consequence, during the first simulations we noticed that nodes can take simultaneous decisions that induce an temporary invalid state (e.g. some neighbors nodes may decide to leave the backbone simultaneously which may leave some other nodes uncovered). In order to face these practical issues some implementation features are proposed.

Pruning rules.

The idea of pruning rules has been first introduced by Wu *et al.* in [58] and is about removing useless nodes from the backbone. As nodes may have incomplete or out-of-date information about their neighborhood some rules are required to guaranty a valid solution. The first solution which is generally proposed in the literature is to add a constraint, based on an unique identifier, to the pruning. This heuristics avoid multiple nodes to go out at a given time but induces really bad solution for some specific graphs, i.e. chains of dependence may inhibit nodes to change their status. In our work, we propose another approach: we relax the previously described constraint (nodes are free to get out of the backbone at the same time without considering identifiers) and we add a re-construction fix mechanism to ensure the convergence to a valid and more compact solution. The underlying idea behind this heuristic is that it may increase the exploration of the solution space as at some time the backbone may be inconsistent, which is not possible with identifier-based constraints. We think that such an approach can enhance the quality of the final solution as it induces more exploration capabilities in the solution space.

Identifier-based decisions.

This simple heuristic greatly reduces inconsistency risks by adding a simple constraint to the pruning rule. A node v may go out of the backbone if the robustness condition is met and if it has the lowest (or highest) unique identifier in its one-hop neighborhood. In a more formal way, a node v can go out of the backbone if and only if:

$$\forall(x,y) \in S_v \big/ \big(|CN^B(x,y)| > 1\big) \wedge \forall u \in N_1^B(v) / id(v) < id(u).$$

This approach can although be used for the clique pruning rule, i.e. if a node detects that it is in a complete sub-graph, it goes out if and only if it has the lowest unique identifier in its 1-hop black neighborhood. If the clique contains multiple nodes, this process will be iterated at every time step until no nodes may have the possibility to go out. As we rely on two-hop knowledge, we have access to the list of neighbors of the nodes that are potentially in the clique. This information can be used to detect if these nodes consider themselves in a clique by comparing their list of neighbors to the list of neighbors of the considered node, say v. The heuristic consists in computing the set of nodes $Clique(v)$ which considered themselves in a clique and compare identifier of these nodes. In a more formal way:

$$Clique(v) = \Big\{ u \in N_1^B(v) \Big/ \big(N_1^B(u) \setminus v \subset N_1^B(v) \setminus u \big) \Big\}.$$

Then, a node v in a clique can go out of the backbone if and only if:

$$\forall u \in Clique(v),\ id(v) < id(u).$$

Always go out heuristic.

The second heuristic proposed in this work allows a node to go out of the backbone every time the robustness condition is achieved, without considering synchronization problems. This implies inconsistencies during execution and requires a re-construction mechanism to ensure a valid solution. The idea behind this heuristic is that relaxing constraints may create opportunities to achieve better solution that would have never existed with strict conditions. This heuristic has been tested for the normal pruning rule and the clique pruning rule.

Re-construction fix.

This rule can be added to the re-construction rule in order to fix some problems due to simultaneous changes. Problems may happen in the structure when a node quits the backbone at the same time than some of its neighbors. If none of its 1-hop neighbor is in the structure we consider two cases in order to avoid too many re-constructions during next round. This algorithm is presented in Algorithm 12. The use of the *always go out* heuristics requires the re-construction fix in order to converge to a valid solution because of the inconsistencies that may be created.

Algorithm 12: Re-construction fix algorithm

if *all neighbors are white* **then**
  if *I am the highest id in my 1 hop neighborhood* **then**
    state = BLACK;
else
  if *my white neighbors are not covered* **then**
    state = BLACK;

5.4.3.4 Theoretical study

In this part a theoretical study of the *Blackbone* algorithm is provided. The validity of the generated solutions is proved in a first time. Message and time complexity are then discussed and the approximation ratio for the identifier-based version is presented.

Valid $k-$vertex connected dominating set solution.

We previously stated in 5.4.3.1 that the algorithm creates $k-$vertex connected structures when the communication graph permits it. As a matter of fact, in order to obtain a valid solution we have to prove that the structure is a dominating set. To such extend, we consider the following cases in which the considered node is white:

- Node v has no neighbor, whatever their color. In this first case, the *re-initialization* rule is triggered in Blackbone. The lonely node becomes black and thus forms a one-node backbone that is dominating itself. In such a trivial case we do have a connected dominating set composed of one node and obviously the $k-$vertex connectivity property is not applicable.

- Node v has some neighbors but all of them are white. Such case may happen due to simultaneous status changes. Algorithm 12 states that in that specific case, to ensure the domination, only the node with the highest identifier in its direct neighborhood may enter the backbone.

- Node v has some uncovered one-hop neighbors. Once again, algorithm 12 fixes this issue by letting node v enter the backbone and thus all previously uncovered nodes are now dominated.

The two last cases rely on algorithm 12 as the *construction* rule is only taking into account the connectivity of black neighbors. With always up-to-date information and non-simultaneous status changing, the four main sub-routines would have been sufficient to guaranty a valid solution. As all possible cases have been taken into account, we can conclude that Blackbone creates $k-$vertex connected dominating set when the connectivity requirements can be fulfilled.

Message complexity.

Blackbone requires network nodes to periodically exchange their list of neighbors along with their status. Such information could be encapsulated into beacons, i.e. small periodical messages to reveal its presence in wireless networks. As Blackbone is not characterized by a finite number of message rounds, it is impossible to provide a message complexity, we can only states that at every considered step, each node sends one message. However, the size of the exchange message can be defined. Indeed, let us consider Δ as the maximal node degree. All network nodes have to exchange their list of neighbors along with their status. As a consequence, the message size required by *Blackbone* is $\mathcal{O}(\Delta)$.

Time complexity.

The time complexity of the *Blackbone* algorithm is equivalent to the time complexity of the *construction* rule as it is the more demanding sub-routine. Indeed, for every couple of black neighbors (x, y) the common black nodes set $CN^B(x, y)$ has to be computed to check if its cardinality is greater or equal to k, the vertex connectivity value. In the worst case, the number of couples is $\Delta(\Delta - 1)$. Moreover with two sorted list of neighbors, obtaining the common black nodes list corresponds to computing the intersection of these two lists and thus it requires $\mathcal{O}(\Delta)$. As a consequence, *Blackbone* is characterized by a time complexity of $\mathcal{O}(\Delta^3)$.

Approximation factor.

If we consider the identifier-based decisions for both normal and clique pruning rule, the worst case scenario is the classical star graph, i.e. a node in the center and an infinite number of independent neighbors. If the node in the center has the lowest identifier then all the independent neighbors have to stay because they are in a clique (composed of themself and the center node) but they do not have the lowest identifier. The solution provided by *Blackbone* is in this case the worst possible and thus the approximation ratio of the algorithm is $\mathcal{O}(n)$. It is also obvious that it remains unchanged if we restrict the analysis to Unit Disk Graphs. Hopefully, the version with the *always go out* heuristics and the construction fix algorithm provides better practical results even if they can not be theoretically proved.

5.4.4 Blackbone2 algorithm

The following part presents our second distributed algorithm. Although the name is almost identical some major differences characterize this contribution. In a first time, we explain the limitation of the first Blackbone algorithm. The solutions to obtain k-vertex connectivity and m-vertex domination are presented respectively in the second and the third part. The required sub-routines and the main loop of the Blackbone2 algorithm are then introduced followed by some optimizations. Finally, the theoretical performances are detailed.

5.4.4.1 Blackbone limitations

The first version of the *Blackbone* algorithm provides a naive yet efficient method to find compact $k-$vertex connected dominating set virtual backbones. The *robustness* property that is introduced to deal with the desired value of connectivity is a simple notion but suffers from an important drawback: for every pair of unconnected black nodes, the only considered node-disjoint paths contain one relay node. Such a constraint induces large dominating sets as nodes *in the middle* are more likely to be forced to stay in the backbone even if they may not be useful. Indeed, paths with two or more relay nodes can be considered as long as they are node-disjoint. *Blackbone 2* has been designed to take into account this problem. However, the algorithm is now restrained to special cases of the vertex connectivity: only connected and bi-connected structures can be created. This new vertex connectivity checking is detailed in 5.4.4.2

The second improvement of *Blackbone 2* is to propose to deal with the $m-$vertex domination property. Any value of m can be addressed. It is however important to notice that a node with less than m neighbors is forced to enter the backbone. The details concerning the vertex domination checking sub-routine can be found in 5.4.4.3.

5.4.4.2 k-vertex connectivity

This part presents the main idea of the algorithm to check whether the structure is connected enough. For *Blackbone 2*, we chose to restrain ourselves to the connected and bi-connected case ($k-$vertex connectivity with $k = \{1, 2\}$) in order to propose a very efficient algorithm considering the time complexity.

Definitions.

The main idea of the *Blackbone 2* algorithm is to create connected or bi-connected virtual backbone by using time-efficient and well-known algorithms from the Graph Theory. Checking if a graph $G = (V, E)$ is connected can be done by using the Depth-first search algorithm. Indeed, a direct application of this algorithm is to compute the connected components (see definition 43) of a given graph. As a consequence, if only one connected component can be found in a graph G, it induces that G is connected.

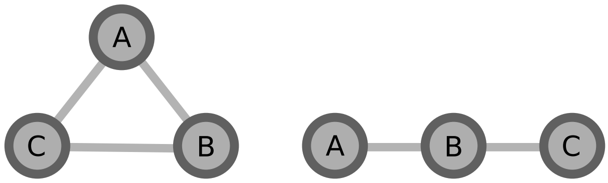
Definition 43 (Connected component of a graph $G = (V, E)$).
A connected component of an undirected graph $G = (V, E)$ is a subgraph in which any two vertices are connected to each other by paths, and to which no more vertices or edges from G can be added while preserving its connectivity.

To check if a graph $G = (V, E)$ is bi-connected, a more elaborated version of the Depth-first search algorithm can be used. In this version the objective is to detect cut vertices, also known as articulation points, in the considered graph. Definition 44 states that if a cut vertex is removed from the graph, it induces that the number of connected component increases. As a result, in a graph with no cut vertex, removing one node does not increase the number of connected component, i.e. the graph remains connected. Definition 45 is directly induced by the previous conclusion: a biconnected graph is a connected graph with no cut vertex.

Definition 44 (Articulation point - Cut vertex).
An articulation point or cut vertex is a vertex that if removed (along with all edges incident with it) produces a graph with more connected components than the original graph.

Definition 45 (Biconnected graph).
A biconnected graph is a connected graph with no cut vertex.

Figure 5.4a is a classical example for simple biconnected graph. Indeed, if one node is removed, the remaining graph will still be connected. On the contrary, Figure 5.4b shows a graph with one articulation point: node B.



(a) No cut vertex (b) One cut vertex

Figure 5.4: On the left, the graph is 2−vertex connected and as such there is no articulation point. On the right, node B is an articulation point as its removal would increase the number of connected components.

Cut-vertices detection algorithm.

An efficient algorithm to compute the cut-vertices has a time complexity of $\mathcal{O}(|V| + |E|)$, i.e. the same that the Depth-first search algorithm. In order to do the processing, some extra data is required for every node. Let v be a node of $G = (V, E)$.

- Num(v): the visit number obtained from a depth-first search (from any node of the graph).
- Low(v): lowest-numbered vertex reachable from v using 0 or more spanning tree edges and then at most one back edge.

As a consequence, Low(v) is the minimum value among the three following cases:

- Num(v);
- Lowest Num(w) among all back edges (v,w);
- Lowest Low(w) among all tree edges (v,w).

Num(v) and Low(v) can be computed in time $\mathcal{O}(|V| + |E|)$ once again with the Depth-first search algorithm. Algorithm 13 detects all the articulation points for a given input graph. As we only want to determine if the graph contain one articulation point, we can stop the computation at the exact moment one of them is detected. This straightforward optimization reduces computation in simulations but obviously does not change the theoretical computation time.

Algorithm 13: findArt: an efficient algorithm for articulation point detection

```
Input: Vertex v
Visited(v) = true;
Low(v) = Num(v) = counter++;
foreach w adjacent to v do
    if not(Visited(w)) then
        Parent(w) = v;
        findArt(w);
        if Low(w) >= Num(v) then
            v is an articulation point
        Low(v) = min(Low(v), Low(w));
    else
        if Parent(v) != w then
            Low(v) = min(Low(v),Num(w));
```

From a local to a global property.

Let $G(v) = (V(v), E(v))$ be the local graph of the node v. This graph represents what v knows about its neighborhood. *Blackbone 2* only requires two-hop neighborhood to take a decision, that is why $V(v) = N_{1\cup 2}(v)$. $E(v)$ contains a restricted set of edges defined by:

$$E(v) = \Big\{(u, w) \in E/u \in N_1(v) \wedge w \in N_{1\cup 2}(v)\Big\}.$$

Let $G^B(v) = (V^B(v), E^B(v))$ be the local graph of the node v induced by black nodes only. $V^B = N^B_{1\cup 2}(v)$ and $E^B(v)$ contains a restricted set of edges defined by:

$$E^B(v) = \Big\{(u, w) \in E/u \in N^B_1(v) \wedge w \in N^B_{1\cup 2}(v)\Big\}.$$

Let $G^B = (V^B, E^B)$ be the global black graph obtained by the union of all local black graph.

$$G^B = \cup_{v\in V^B} G^B(v).$$

Property 2 (Global biconnectivity)**.**
If $\forall v \in V$, $G^B(v)$ is biconnected, then G^B is biconnected.

5.4.4.3 m-vertex domination

Having two node-disjoint paths between any pair of nodes in a virtual backbone increases its resilience to backbone node failure or wireless channel disappearing due to node mobility. However, it does not improve the fault-tolerance for non backbone nodes. The *Blackbone 2* algorithm proposes an simple and efficient mechanism to ensure the $m-$vertex domination of the virtual backbone. Moreover, the value of m can be set to any integer value greater than zero without any time efficiency impact.

m-vertex domination property.

From the definition of the $m - vertex$ domination constraint (see definition 32) we can conclude that a node v only needs to check how many backbone neighbors are in its one-hop neighborhood to determine if it should enter the virtual backbone. Moreover, for a given non-backbone node V, even if it is enough dominated, v should check if it can help dominating its one-hop non-backbone neighbors. In a more formal way, for a given node v, the following properties have to be checked:

$$|N_1^B(v)| \geq m \tag{5.14}$$

$$\forall w_i \in N_1^W(v),\ \exists z_1, ...z_m \in N_{1\cup 2}^B(v) \Big/ (z_1, w_i), ..., (z_m, w_i) \in E(v). \tag{5.15}$$

m-vertex domination checking algorithm.

The proposed algorithm, is the direct translation of definition 32. Algorithm 14 first checks if a given node v is enough dominated by its one-hop black neighbors. If v has enough black neighbors it checks if it can help dominating its one-hop white neighbors. With an appropriate data structure, this algorithm is processed in $\mathcal{O}(|\Delta|)$ and thus can be deployed even on very low computation power devices.

Algorithm 14: m-domination checking algorithm

Input: m, the domination parameter

Output: *true* if the subgraph is m-vertex dominating, *false* otherwise

dominating = true;
if $|N_1^B(v)| < m$ **then**
    dominating = false;
    break;
for *each* $w \in N_1^W(v)$ **do**
    if $|N_1^B(w)| >= m$ **then**
        dominating = false;
        break;
return dominating;

From a local to a global property.

It is straightforward that if the local graph of all the network nodes contains a $m-$vertex dominating set, then the complete communication graph also has a $m-$vertex dominating set, i.e. the backbone nodes. Property 3 proposes the same statement in a more formal way.

Property 3 (Global domination)**.**
Let $DS(G)$ be a dominating set of a graph $G = (V, E)$. If for every node $v \in V$, $G(v)$ is m-vertex dominated by $DS_{G(v)}$, then the $\cup_{v \in V}(DS_{G(v)})$ is a m-dominating set of G.

5.4.4.4 Blackbone 2 main loop and sub-routines

The *Blackbone 2* algorithm is based on two completely symmetric rules, but only one will be executed depending on the current node state, i.e. its current color. Indeed, when a node v is outside the backbone a construction rule will be triggered to determine if v can stay outside the backbone or has to be part of it. Similarly, backbone nodes regularly check with a pruning rule if they are still required to fulfill the backbone properties. Both rules are using the previously presented checking algorithms for the $k-$vertex connectivity or the $m-$vertex domination.

Subgraph for biconnectivity checking.

For the connectivity property we do not consider the complete local graph $G(v)$, but a sub-graph induced by the black nodes set, $N^B_{1\cup 2}(v)$, as we do not require non-backbone nodes to check the connectivity of the virtual backbone. More precisely we only consider the one-hop black nodes and the useful two-hop black nodes, i.e. black nodes that help for the bi-connectivity. As a consequence, two-hop black nodes with less than two black neighbors will be ignored from the computation. Indeed, even if node v enters the backbone it will not change the fact that the sub-graph is not bi-connected. Moreover the considered subgraph does not contain the node v. Let $V_{ul}(v)$ be the set of useless 2-hop black nodes:

$$V_{ul}(v) = \{u \in N^B_2(v) \,/\, |N^B_1(u)| < 2\}.$$

In a more formal way, the set of considered nodes $V_c(v)$ is:

$$V_c(v) = N^B_{1\cup 2}(v) \setminus V_{ul}(v).$$

Let $G_c(v)$ be the graph induced by the set of nodes $V_c(v)$.

Pruning rule.

The pruning rule checks for a given node v if the sub-graph induced by the black nodes is a $m-$vertex dominating set of its local graph $G(v)$. Additionally, the rule checks if the connectivity of the local graph fulfils the desired requirement. If a $k=1$-vertex connected structure is wanted, the number of connected components of the sub-graph induced by the black nodes is computed. Otherwise, for a bi-connected structure, the bi-connectivity of the sub-graph $G_c(v)$ is tested using algorithm 13. Algorithm 15 shows that when the domination and the connectivity constraints are satisfied, the considered node v can get out of the backbone.

Construction rule.

The construction rule is the complete opposite of the pruning rule. A node v is required to enter the virtual backbone if v or its one-hop white neighbors are not $m-$vertex dominated or if the considered sub-graph is not connected enough. For $k = 1-$vertex connectivity, $G^B(v)$ is tested and for $k = 2-$vertex connectivity, $G_c(v)$ is checked for cut vertices. Algorithm 16 summarizes the construction rule.

Algorithm 15: Pruning rule of the Blackbone 2 algorithm for the black node v

Input: $k \in \{1, 2\}$, the connectivity
$m \in \mathbb{N}^{+}$, the domination parameter
Output: *true* if the node can get out of the backbone
false if it is still required

```
changeState = false;
if N^B_{1∪2}(v) is a m-dominating set of G(v) then
    if k = 1 and G^B(v) is connected then
        changeState = true;
    if k = 2 and G_c(v) is biconnected then
        changeState = true;
return changeState;
```

Algorithm 16: Construction rule of the Blackbone 2 algorithm for the white node v

Input: $k \in \{1, 2\}$, the connectivity
$m \in \mathbb{N}^{+}$, the domination parameter
Output: *true* if the node can get out of the backbone
false if it is still required

```
changeState = true;
if N^B_{1∪2}(v) is a m-dominating set of G(v) then
    if k = 1 and G^B(v) is connected then
        changeState = false;
    if k = 2 and G_c(v) is biconnected then
        changeState = false;
return changeState;
```

Main loop.

The main algorithm repeats the same operation during the whole simulation and for every network node. Depending on the color of the considered node, the main loop selects the rule to be applied and changes the color when it is required. The main loop is detailed in algorithm 17.

Algorithm 17: Main algorithm of Blackbone2

```
Input: k ∈ {1, 2}, the connectivity
m ∈ ℕ+, the domination parameter
for every time step do
    if BLACK and pruningRule(k, m) then
        color = WHITE;
    if WHITE and constructionRule(k, m) then
        color = BLACK;
```

5.4.4.5 Optimizations

In this section we propose some optimization to the genuine algorithm. In a first time, to avoid useless changes of status, we propose two specific cases for which the status of a node can be determined without using either the construction or the pruning rule. In a second time, as the *Blackbone 2* takes its decisions based on the local sub-graphs, we propose two approaches to avoid useless changes of status due to mobility. The underlying idea is for a given node v to ignore neighbors that may soon disappear from its vicinity. Two different methods to detect such nodes are proposed depending on the availability of position data. The third type of optimization is also introduced to increase the stability of the generated virtual backbones. The main idea is to force some nodes to stay in the backbone as they are assumed to be useful sooner or later if not immediately.

Fixing some nodes.

The first basic optimization is to fix nodes that have absolutely no possibility to change their state at the current simulation time. For these specific situations it is useless to process the domination and connectivity checking. Here is a non exhaustive list of special cases for which an optimal solution does not require any complicated computation:

- Lonely nodes: if a node has no neighbor it can go to non-backbone state and stays in it until new neighbors appear.

- Complete graphs or clique: if the local graph is a complete graph, then no backbone nodes is required as no multi-hop communication are available.

These straightforward optimization are added to all the following *Blackbone 2* variants.

Reducing the considered subgraph.

This second optimization type is designed to increase the stability measures. The main idea is that it may not be efficient in terms of maintenance if the nodes adapt themselves to all the topology changes. In fact, nodes should only consider useful neighbors to check the domination or the connectivity. As less nodes are considered we assume that it induces less possibilities for the nodes to change their status.

We propose two straightforward and yet different heuristics to filter useless nodes:

- the age of a node. Let us define the age of a neighbor w for a node v as the cumulated time since node w entered the neighborhood of node u. Such an information can be gathered by simply incrementing a variable for each neighbor w since it appeared in the one-hop or two-hop neighborhood. The *Blackbone 2* algorithm using this method will be referred as the age threshold variant.
- the relative speed of a node. The idea is to compute the relative speed of any neighbor w of a node v. Such information can be computed via GPS data and as a consequence the remaining time until a particular neighbor w goes out of the neighborhood can be estimated. The *Blackbone 2* algorithm using this method will be referred as the relative speed threshold variant.

The main difference between these two heuristic choices is that relative speed may be more precise but requires additional information and thus increases the bandwidth requirements for the algorithm. Indeed, the *Blackbone 2* algorithm gathers data via specific messages containing a list of neighbors and their useful details. For a given node, computing the age of its neighbors can be done by counting the number of messages received from them. However no geographical data can be extracted from the original message format. To overcome this problem we propose two solutions:

- if nodes are equipped with a GPS chip or any location based device, the corresponding geographical data can be embedded into the exchanged messages. This possibility provides precise positioning data but require additional hardware capabilities. Moreover, relying on such data does not take into account obstacles that may prevent good quality communication between geographically close pair of nodes.
- the nodes may estimate the distance to their neighbors thanks to the signal strength evaluated while receiving the exchanged messages. Such an approach can only provide estimated distances between pairs of nodes but does not require additional hardware capabilities. Measurements based on the quality of the wireless channel may also be more realistic concerning the presence of obstacle between some pairs of nodes.

Fixing the state of some nodes.

The major problem of the *Blackbone 2* algorithm comes from the unbounded number of computation steps until convergence, i.e. the nodes may be changing states for some time until they are all fixed. This is due to some sort of domino effect: a node that changes its state may create a chain of reactions. In order to reduce these changes of state we propose two heuristic choices:

- If a node is in the backbone and it has the highest number of neighbors amongst its neighbors then it stays in the backbone. This heuristic choice gives more importance to higher degree nodes as more nodes may rely on them. The version of the *Blackbone 2* algorithm using this method will be referred as the high degree variant.

- If a node is in the backbone, it waits a specific amount of time before considering the possibility of leaving it. This second heuristic choice reduces the *blinking* problem, i.e. nodes in the same part of the network that keep changing state. This problem is mainly due to the intrinsic asynchronous nature of ad hoc networks: a node v may take a different solution if it had received a packet from its neighbor w that just changed its state. The version of the *Blackbone 2* algorithm using this approach will be referred as the forced stability variant.

5.4.4.6 Theoretical performances

In this section we briefly discuss the theoretical performances of the Blackbone2 algorithm in terms of time and message complexity.

Time complexity.

Let Δ be the maximum node degree in a communication graph $G = (V, E)$. Let us consider a particular node v. In the worst case scenario, $|V(v)| = \Delta^2$ and $|E(v)| = \Delta^2$. If $|V(v)| = \Delta^2$, it means that v has Δ neighbors and all its neighbors also have Δ neighbors. If $|E(v)| = \Delta^2$, it means that v has Δ neighbors and they form a complete graph, i.e. all pair of nodes are connected. Based on the previous assumption, checking the existence of an articulation point requires $\mathcal{O}(\Delta^2)$. As the domination checking algorithm only requires $\mathcal{O}(\Delta)$ and the two rules (pruning and construction) have a complexity of $\mathcal{O}(\Delta^2)$, we can conclude that the *Blackbone 2* algorithm has a time complexity of $\mathcal{O}(\Delta^2)$.

Message complexity.

The *Blackbone2* algorithm requires network nodes to periodically exchange their list of neighbors along with their status. Such information could be encapsulated into beacons, i.e. small periodical message to reveal its presence in wireless networks. As Blackbone is not characterized by a finite number of message rounds, it is impossible to provide a message complexity, we can only state that at every considered time step, each node sends one message. However, the size of the exchange message can be defined. Indeed, let us consider Δ as the maximal node degree. All network nodes have to exchange their list of neighbors along with their status. As a consequence, the message size required by *Blackbone2* is $\mathcal{O}(\Delta)$.

5.5 Recap of key points

- We proposed a mathematical model for the minimum $m-$vertex dominating set problem.
- A generic scheme for the $l-$level domination property is proposed and can be adapted to any integer value bigger than one.
- A DC Algorithm has been developed to solve the minimum dominating set problem along with its variants. An iterative scheme has been proposed to optimize the exploration of the solution space and always provide a valid solution.
- Two distributed and localized algorithms, Blackbone and Blackbone2 have been proposed to create special cases of the general minimum $k-$vertex connected $m-$vertex dominating set problem.
- Blackbone is mainly based on the robustness property introduced in this chapter which is checking all pairs of unconnected neighbors for connecting routes. Any value of $k-$vertex connectivity can be set but the algorithm only considers the one-vertex domination value.
- Blackbone2 extends the size of the connecting routes and thus permits to reduce the number of backbone nodes. The required computation time is also reduced compared to the first Blackbone algorithm. Although only the connected and biconnected cases are encompassed, the $m-$vertex domination can bet set to any value. As a consequence, the first Blackbone algorithm can still be useful when a $k-$vertex connectivity with $k > 2$ is required.
- Both algorithms differ from the literature as they do not rely on an implicit synchronization of the network and they require an indefinite number of steps to converge. The latter characteristic allows the algorithm to explore the solution space with more freedom and thus find more compact virtual backbones.

Chapter 6

Experimentation

Contents

6.1 Centralized approach . . . 108
6.1.1 Experimental setup . . . 108
6.1.2 Global methods are not scalable . . . 109
6.1.3 Finding good local solutions . . . 110
6.1.4 Influence of parameters . . . 111
6.1.5 Results analysis . . . 113
6.2 Distributed approach . . . 115
6.2.1 Experimental setup . . . 115
6.2.2 Blackbone 1 - static networks . . . 117
6.2.3 Blackbone2 - static networks . . . 120
6.2.4 Blackbone 2 - dynamic networks . . . 124
6.2.5 Results analysis . . . 131
6.3 Recap of key points . . . 132

In this chapter we present the simulations we performed to empirically test the validity and the efficiency of the solutions proposed in chapter 5. In a first time we detail and analyze the results obtained by our centralized algorithm based on DC programming and DCA (see section 6.1). The two distributed algorithms, Blackbone and Blackbone2 are tested in section 6.2. A first series of simulations results are presented for Blackbone with static networks (see 6.2.2). The results for Blackbone2 are finally provided for static (see 6.2.3) and dynamic environments (see 6.2.4).

6.1 Centralized approach

In this section we first show that the NP-completeness of the domination problems only permits us to optimally solve small instances of graphs. Then we summarize the results obtained with the classical heuristic presented by algorithm 19 and the two 0-1 programming approaches: DCA and CPLEX with a time limit. The last part empirically studies the effects on the size of the solutions and computation time when changing some problem parameters.

6.1.1 Experimental setup

Random graph generation.

In order to test our methods we needed to randomly generates many instances of graphs. To such extend we used GraphStream [34] to create graphs with the desired properties. As the nodes are fixed and known a priori, we only have to randomly create edges such as we obtain the wanted average density. Algorithm 18 is used to create particular instances of graphs that correspond to the $G(n, M)$ model proposed by Erdős and Rényi [36], in which a graph is chosen uniformly at random from the collection of all graphs which have n nodes and M edges. In our case, M is the total number of edges such as the average local density, i.e. the number of neighbors, is equal to a density value d. As such, $M = n * d$.

Algorithm 18: Random graph generator

Input: n, the number of nodes
Input: d, the average density
Output: $G = (V, E)$, the graph

$M \leftarrow n * d$;
$nbEdges \leftarrow 0$;
while $nbEdges \leq M$ **do**
  $n1 \leftarrow rand(0, n-1)$;
  $n2 \leftarrow rand(0, n-1)$;
  if $(n1 \neq n2)$ *and* $(E[n1, n2] \neq 1)$ **then**
    $E[n1, n2] \leftarrow 1$;
    $E[n2, n1] \leftarrow 1$;
    $nbEdges+ = 2$;
return $G = (V, E)$;

Compared algorithms.

We compare our DC algorithm with the reference commercial solver for $0-1$ problems, i.e. CPLEX from ILOG company. In the first experiences we empirically confirm that the minimum dominating set is quickly intractable when the size of the problem increases. For the other tests, CPLEX will be given a fixed amount of time to compute its best solution. Without any time constraint, the ILOG solver will compute the optimal solution. This method will be referred as $CPLEX_{opt}$. This variant with a time constraint will simply be referred as $CPLEX$. As an upper bound, we also included the straightforward approximation algorithm based on a greedy behavior: at each step, the algorithm picks up the node that covers the greatest number of uncovered nodes. The corresponding pseudo-code can be found in algorithm 19. This solution is generally referred to as *Greedy-Set-Cover* and its approximation ratio is $\ln(|V|)+1$. This algorithm will be referred as *Heuristic* in the following tests.

Algorithm 19: Greedy algorithm for the $\min -DS$ problem

Input: Graph $G = (V, E)$
Output: The solution x

```
//Initialization
x ← (0, . . . , 0);
unCovered ← E;
//Main loop
while unCovered ≠ ∅ do
    getMostCoveringNode(curNode);
    x[curNode] ← 1;
    removeCoveredNodes(unCovered, curNode);
return x;
```

6.1.2 Global methods are not scalable

In this first series of tests, we empirically prove that optimally solving the minimum dominating set problem requires an exponentially increasing computation time with respect to the increase of the problem size. In Table 6.1 we observe that the computation time of CPLEX increases very quickly with the size of the node set. Although this approach guarantees an optimal solution, it is not scalable for bigger problems (a few hundreds of nodes).

The solutions proposed by our DC Algorithm are not always optimal. Nevertheless, they are valid (the solution is a dominating set) and the computation is characterized by a very slow increases. Indeed, we can observe that only two seconds are needed to compute problems with hundred nodes and that only one more second is required to solve problems with three hundred nodes.

The greedy algorithm obtains pretty bad solutions, i.e. the cardinality of the dominating set is a lot bigger than those of CPLEX or DCA. However, for these small instances of graph, the computation time is almost negligible. It should also be noticed that the increase in

Number of vertices	100	150	200	250	300
$t_{CPLEX_{opt}}$	0.6	1.4	44.8	459.8	3835.8
t_{DCA}	2	2.2	2.8	2.8	3
$t_{Heuristic}$	0.1	0.1	0.3	0.5	0.6

Table 6.1: Time (in seconds) to obtain solution with 100 to 300 nodes and density 10.

Number of vertices	100	150	200	250	300
$\%_{CPLEX_{opt}}$	12.2	12	12.3	12.16	12
$\%_{DCA}$	13	13.3	14.4	14.16	13.6
$\%_{Heuristic}$	18	17.3	17.5	17.4	17.2

Table 6.2: Dominating set size (in %) for graphs from 100 to 300 nodes with density 10.

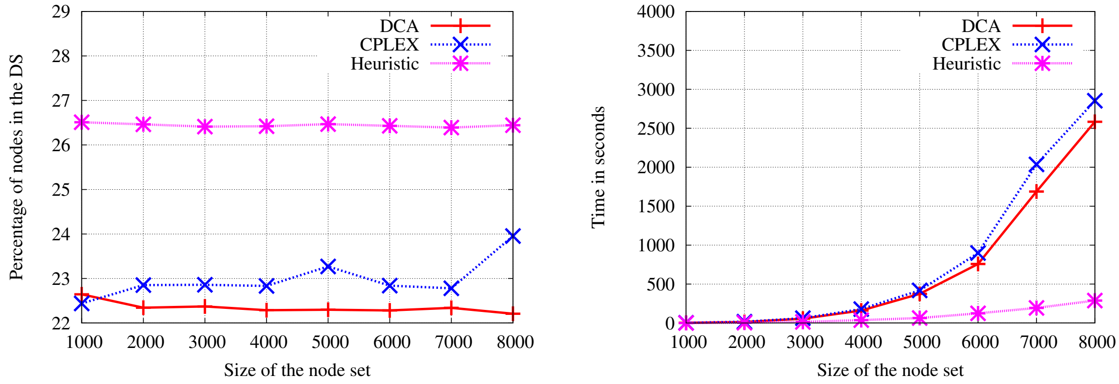
the computation time of DCA is slower (+50%) than in the case of the greedy algorithm (+500%).

The results concerning the size of the solutions are shown in Table 6.2. The average difference between the optimal solution and DCA is equal to 1.56% of the total node set. Even if the numbers may look small, such difference is non negligible. However, as we previously observed in Table 6.1, the computation time overhead is really important and increases quickly with the size of the problem: 3 seconds for solving 300 nodes instances with DCA and 3835 seconds for CPLEX. At the extreme opposite, the greedy algorithm finds really poor solutions ($5, 35\%$ bigger than CPLEX) but its computation time is almost equal to zero for such small instances. In such a context, DCA provides the most advantageous trade-off as it produces almost optimal solution in a reasonable time.

6.1.3 Finding good local solutions

As stated previously, solving big instances of graphs (more than 1000 nodes) seems out-of-reach for global approaches. As such, we propose to compare local optimal solutions that may be obtained in a decent computation time. In order to fairly compare the two approaches (DCA and CPLEX), we impose a time limit to the CPLEX solver. In such conditions, the software of ILOG will return the best solution found before the time limit. This time limit is set to the amount of time used by the DCA for each instance of graph. For some tests, CPLEX did not find a valid solution, i.e. a vector X composed of zeros and ones, at the time limit. In these cases, the solver simply continues its computations until an integer solution is found. As such, CPLEX has at least the same amount of time than DCA to compute a solution.

In Fig. 6.1a we observe that CPLEX provides better solutions until the node set increases to 1000. From that size the DCA always produces more compact dominating sets. Moreover



(a) Percentage of nodes

(b) Computation time

Figure 6.1: Results for graph density $\in [2, 15]$

Fig. 6.1b shows that the DCA requires less computation time than CPLEX. Although CPLEX is constrained with a time limit, in some cases the solver has not found any valid solution yet. In such a situation, CPLEX continues its computation until a valid solution is discovered. The greedy algorithm gives us a rough idea of the solutions we can obtain without using sophisticated methods: the quality is poor but the required computation time is a lot lower. Moreover, the growth of the computation time is slower with the greedy approach than the CPLEX and DCA. It can however be noticed that the greedy algorithm requires much more time when the graph density is low: more nodes are required to dominate the graph and thus more loop iteration have to be processed.

6.1.4 Influence of parameters

In this subsection we empirically study the influence of different parameters on two major criteria: the size of the solution and the computation time required to find it. In a first time we show the impact of the graph density on the behavior of the different algorithms and in a second time, we make the vertex domination value vary.

6.1.4.1 The influence of graph average density

In Fig. 6.2 we observe that the percentage of nodes in the dominating set is rapidly decreasing when the average density is increasing. This is a natural observation directly implied by the fact that in more dense graphs, an average node has more neighbors and thus covers more nodes.

Except for our low density tests, DCA provides more compact dominating sets that tends to the optimal solutions. When the density increases, using CPLEX results are equivalent to those of the greedy heuristic. As such, we conclude that DCA is more robust than CPLEX to the variations of density.

As we did not compute the exact solutions for such big problems, we can only estimate what the optimal value could be. As mentioned in algorithm 18, we randomly generated

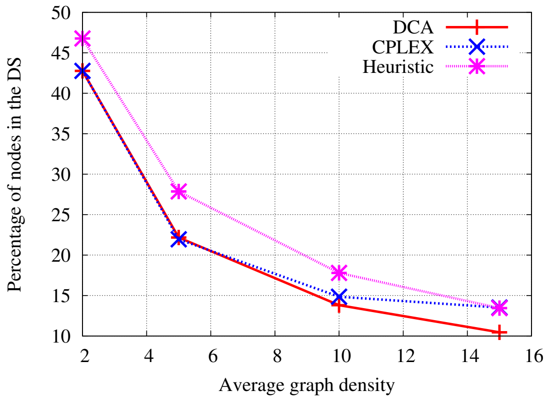


Figure 6.2: Influence of the average graph density on the dominating set size

avg_dens	**2**	**5**	**10**	**15**
$\%_{CPLEX}$	42	21.94	14.85	13.52
$\%_{DCA}$	42	22.16	13.8	10.45
$\%_{Greedy}$	46.77	27.86	17.79	13.44
$\%_{avg\_opt}$	33.3	16.6	9.09	6.25

Table 6.3: Dominating set size (in percentage) for graphs with 5000 nodes

M edges. A good approximation to compare such graphs would be to consider the case of uniform distribution of edges, i.e. all nodes have the same number of neighbors. In that case we can directly compute the optimal solution with formula (6.1). As the edges are supposedly fairly distributed, we divide the total percentage of nodes by the average density (the covered nodes) plus one (the dominating node).

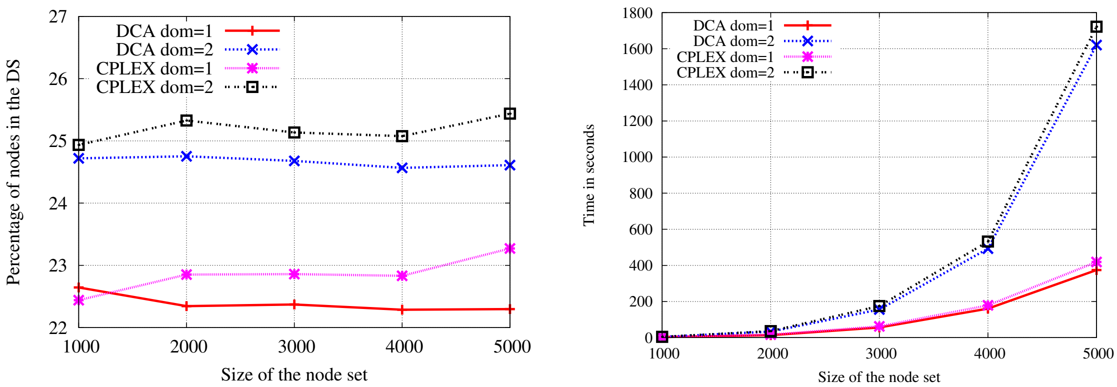
$$avg\_opt = \frac{100}{avg\_dens + 1}. \tag{6.1}$$

Table 6.3 presents the complete results for DCA, CPLEX, the greedy algorithm and the avg_opt value for the 5000 nodes tests. We observe that CPLEX and DCA obtain almost the same results for low density graphs, with a slight advantage for CPLEX. However, DCA obtains significantly better results when the density increases. The greedy algorithm obtains poor results independently of the density, as its solutions are $4,7\%$ higher than DCA.

In Table 6.4 we observe that the average density greatly influences the time required to find a solution: the denser the graph is, the more time is required to solve the problem for DCA and CPLEX. This observation is completely different for the heuristic as choosing a node may remove more uncovered nodes in dense graphs.

avg_dens	**2**	**5**	**10**	**15**
t_{CPLEX}	15	46	432.8	1182.4
t_{DCA}	**14.8**	**31**	376.4	1075
t_{Greedy}	113.6	66	**43.8**	**34**

Table 6.4: Processing time for graphs with 5000 nodes



(a) Percentage of nodes (b) Computation time

Figure 6.3: Influence of the domination parameter

6.1.4.2 The influence of the domination parameter

In Fig. 6.3a we clearly observe the size overhead induced by the increase of the vertex domination value. Indeed, obtaining a $m = 2-$vertex dominating set does not require much more dominating nodes than for a $m = 1-$vertex dominating set. The behavior of both resolution methods seems however completely independent of the domination value: DCA produces more compact dominating sets than CPLEX except for dom=1 and size=1000.

In Fig 6.3b, we observe that the domination parameter greatly influences the computation time. However, even if computing 2−vertex dominating solutions is indeed longer, it does not affect the small time difference between both methods: DCA and CPLEX. Indeed, whatever the vertex domination value is, CPLEX requires a little bit more computation time to provide a valid solution.

6.1.5 Results analysis

The minimum dominating set is a NP-Hard problem and thus cannot be solved in a reasonable amount of time when the size of the problem increases. In 6.1.2 we illustrated this fact by regularly increasing the size of the problem and observing the required computation time to optimally solve the problem.

As a consequence, nearly-optimal methods characterized by an efficient computation time are required. To such extend, a DC Algorithm has been proposed and compared with the classical greedy approximation algorithm. In 6.1.2 we observed that DCA is a good trade-off between the quality of its solutions and its time requirements. Indeed, for small instances of graphs, the proposed DC Algorithm obtains nearly optimal results and its computation time increases very slowly compared to the size of the problem. Although the greedy algorithm requires even less computation time, the generated solutions are not close to the optimal values.

In 6.1.3, big instances of graphs have been tested and a time limit has been set for CPLEX. In such a configuration, the DC Algorithm obtained more compact solutions in less time than the ILOG solver. Although this results are very encouraging, it should be noticed that the greedy approach still requires less computation time when considering the complete series of tests.

In 6.1.4.1 it can be observed that the computation time requirements are highly correlated to the average graph density. Indeed, when the density is low, the greedy algorithm obviously needs more iterations to obtain a valid solution. As a consequence, it requires more time than CPLEX or DCA to computes bigger dominating set.

In 6.1.4.1 it can also be noticed that DCA is more robust than CPLEX when the average graph density increases. Indeed, in the high density case, CPLEX and the greedy approach generates almost the same quality of solution while DCA is still obtaining very compact dominating sets.

Although the increase of the vertex domination value has a major impact on the size of the solutions and the required computation time, it does not change the advantage of DCA on CPLEX. These results can be observed in 6.1.4.2 and proved once again that DCA is a robust solution to solve the considered problem.

6.2 Distributed approach

In this section we present the results of the two versions of Blackbone algorithm. In a first time we provide all the details concerning the experimental setup. The two following parts present the results of both versions of Blackbone with static networks. Finally, results obtained in mobile environments are detailed.

6.2.1 Experimental setup

Simulator and framework.

Even though most of the contributions have been focused on highlighting their theoretical characteristics, we think that an algorithm for ad hoc networks should be first and foremost be empirically tested via implementations on real devices or simulations. Indeed, while designing an algorithm, many assumptions are made on its execution environment for the sake of simplicity and as a consequence, a validation in more realistic conditions is necessary.

In our case, we chose to simulate our algorithms. It can be argued that simulated results are less precise than tests on real devices as simulators cannot represent all the details of an environment and generally rely on simple models to represent the wireless channel or the mobility of the network nodes. However, simulations also have decisive advantages for the validation process, such as the possibility to test the algorithm with hundreds of nodes or to tests thousands of different situations.

The simulator used for all the experimentation is *OMNeT++* version 3.3p1 with the Mobility Framework version 2.0p3. We chose this simulator for its elegant and modular conception which helps developing and integrating your own protocols. Moreover we selected the Mobility Framework, an independent project that provides many already-made components, such as mobility management or IEEE802.11b support.

Algorithms basic settings.

The Blackbone algorithms, whatever their version are based on periodical events such as:

- the beaconing period T_B, which is the amount of time between two successive messages containing information about the neighborhood are sent. This algorithm parameter directly impacts the bandwidth usage as more frequent messages induce less bandwidth for real communication. However, setting a long beaconing period also directly impact on the freshness of the gathered information, i.e. the longer the period is, the higher the probability for a node to have not up-to-date information about its neighborhood.
- The checking period T_{Ch} that represents the amount of time between two Blackbone iteration, i.e. how often does a network node checks its states. This parameter impacts the computational overhead of the algorithm. Indeed, if the period is small the network nodes will compute their version of Blackbone more frequently and that may drain the batteries quicker. However this parameter also influence the quality of representation of the virtual backbone. Indeed, if the nodes often check their state, they may adapt quicker to topology change. We can notice that the checking period is somehow related to the beaconing period as it is not useful to check for a status update if no new topological information has been gathered.

- Forgetting or removing periods for one and two-hops neighbors, T_{Rm1} and T_{Rm2}. As the network topology may change through the simulation time, network nodes should be able to remove old neighbors that are not in their vicinity anymore. Such updates of the local neighborhood are generally accessed via a global entity provided by the simulator. For the sake of realism, in our simulation our network nodes will take their decisions based on exchanged messages only. If no messages coming from a particular neighbor is received during a period T_{Rm1}, this neighbor can be deleted from the local graph. Similarly, a two-hop neighbor which is not appearing in one-hop neighbors' lists for a period T_{Rm2} will be removed from the considered neighborhood.

After some preliminary tests, we decided to set the value of T_B to 500ms. All the other periods, including the statistical measures period used to gather global information of the network are based on the value of T_B. A summary of these values is shown on table 6.5. T_{Ch} is set to the value of T_B and thus allows the tested algorithms to adapt very quickly to topology changes. Due to the constraints of the wireless channels such as collision or interferences, we set the two removing periods T_{Rm1} and T_{Rm2} to four times the beaconing period. Preliminary tests permit to determine that these values are high enough to avoid removing nodes that may not have been able to transmit due to some wireless disturbances.

Period	**Value**
T_B	500 ms
T_{Ch}	500 ms
T_{Rm1}	2000 ms
T_{Rm2}	2000 ms

Table 6.5: Period values for the simulations

Shared simulation settings.

Ad hoc networks are composed of autonomous network nodes and as such, having all devices sending messages at the same time is very unlikely in real scenarios. However, by default a simulator will process all the nodes as soon as it can. As a consequence, in order to initialize the simulation, all the nodes will start their beaconing process at a time defined by a random number. Additionally, this randomness reduces collisions at the beginning of the simulation. The OMNeT++ default random number generator (Mersenne Twister RNG by M. Matsumoto and T. Nishimura) has been used to distribute the nodes in the simulation space. The seed of the simulation is equal to the number of its run number.

6.2.2 Blackbone 1 - static networks

In this part we present the results concerning the first version of Blackbone. Our algorithm is compared to WuLi(*) [58] and the $k-$coverage condition [31] from Dai *et al.* for the one-vertex connected version of the algorithm. Additionally, Blackbone is compared to the $k-$coverage condition for the $k = 2-$vertex connected virtual backbones. It should be noted that the terms *DaiWu* and *k-coverage condition* will refer to the same algorithm.

6.2.2.1 Experiment specific settings

The simulation space is a 100-meter square. The transmission range has been fixed to 25 meters. These experiments have been repeated for different numbers of nodes (20, 30, 40 up to 140), which permits to deal with a wide range of average node degree or density [2, 22]. In table 6.6 a summary of the mean densities by number of nodes in the network is shown.

Number of nodes	Average density
20	2,87
30	4.52
40	6.18
50	7.73
60	6.37
70	10.93
80	12.47
90	14.06
100	15.6
120	18.68
140	21.76

Table 6.6: Average density per number of node in the network

6.2.2.2 Performance comparison for k=1-vertex connected DS

WuLi(*) algorithm is composed of two steps, a marking process and a pruning rule. The marking process adds a node v to the virtual backbone if and only if v has at least two unconnected neighbors. The pruning rule of WuLi(*) (see definition 4) considers every subset of nodes that may cover the set of backbone nodes of a particular node v to detect if v can step out of the CDS. We chose to compare our algorithm with WuLi(*) as it is regularly used by other authors to compare their own solution.

The $k-$coverage condition with $k = 1$ is however more adapted for a comparison with our algorithm. Indeed, it is not composed of different synchronized steps such as WuLi(*). As a consequence, our algorithm is also compared to the $k-$coverage condition.

Figure 6.4a shows that Blackbone produces more compact virtual backbones than the two other algorithms when the average node degree is below 14; this density is achieved with 90 nodes in our network. When the density increases, the k-Coverage algorithm obtains the best results. WuLi(*) algorithm obtains better results than Blackbone only when the number of nodes in the network is higher than 120.

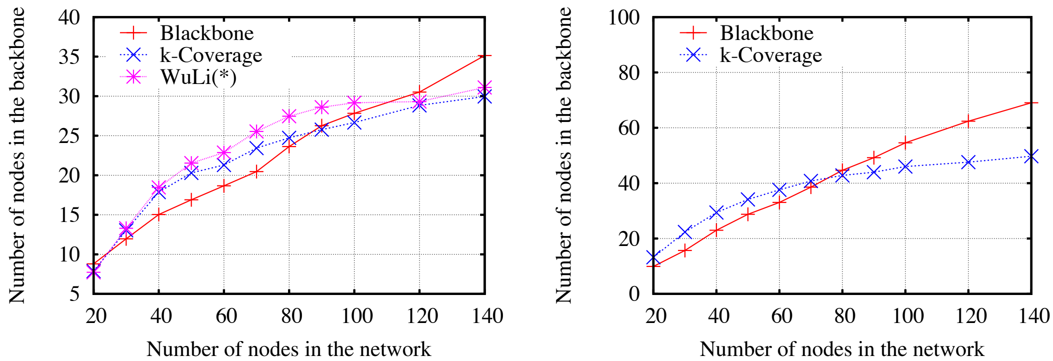
6.2.2.3 Performance comparison for k=2-vertex connected DS

The main goal of Blackbone was to propose a fully distributed $k-$vertex connected dominating set algorithm and to compare its performances to the state-of-the-art solutions in the literature. However, as far as we know, no equivalent algorithm has been proposed. Indeed, our results cannot be fairly compared to the solutions we found in the literature because of important differences.

First, a lot of algorithms are composed of distinct phases, (e.g. DDA [90] or LDA [89]) and thus require the ad hoc networks to be synchronized. Moreover LDA [89] is based on CDS-BD-D which requires a node to be the root of the network. Second, some algorithms may have access to some piece of information that is very difficult to gather in a decentralized way, such as network size and node density for the two first approaches in [31], or even constant maximal node degree for LDA [89].

The most similar algorithm existing in the literature is the deterministic approach of Dai *et al.*, although it only works for the case $k = m$. This algorithm uses the k-coverage condition [31] that checks if there are k node-disjoint paths between every pair of neighbors. Such a computation is done thanks to a variation of the Edmonds and Karp maximum flow algorithm. We have compared our results with this approach, even if there is one major difference: the Blackbone algorithm does not guaranty a $2-$vertex dominating set.

As we can see in Figure 6.4b, our algorithm provides better results for networks with an average node degree lower than 12; This density is achieved with 80 nodes in the network. k-Coverage seems to be less sensitive to the network density but its computation time is higher : $\mathcal{O}(\Delta^3) < \mathcal{O}(k\Delta^4)$. This high computation cost may be an important factor when dealing with mobile environment, where information may be incomplete and available computation time is short. Moreover, as the computation time is a mathematical function based on the maximum node degree, the $k-$coverage condition may require a lot more computation power than Blackbone in dense situations.



(a) k=1-vertex connected DS (b) k=2-vertex connected DS

Figure 6.4: Size of the virtual backbones versus the total number of nodes

6.2.2.4 Results analysis

Blackbone obtain very promising results provided the network density is not too high. Moreover, these results do not rely on the synchronization of the network contrary to WuLi(*). Poor results in high density situations are most probably due to the robustness property that may be too constraining and thus adds too many nodes into the backbone. Blackbone2 solves this problem in the next simulations.

6.2.3 Blackbone2 - static networks

Blackbone2 is a direct evolution of the first Blackbone algorithm that corrects the high density issues due to the construction rule (see rule 12). Blackbone2 is limited to connected and biconnected structures but permits to increase the vertex domination value. We first compare Blackbone2 results for biconnected backbones with the first version of Blackbone and the $k-$coverage condition with $k = 2$. Connected dominating sets results are then additionally compared with WuLi(*).

6.2.3.1 Experiment specific settings

The simulation space is a 100 meters square. The transmission range has been fixed to 25 meters. These experiments have been repeated for different numbers of nodes (20, 30, 40 up to 140), which permits to deal with a wide range of average node degree or density [2, 22]. In table 6.6 a summary of the mean densities by number of nodes in the network is shown.

6.2.3.2 Performance comparison for k=2-vertex connected DS

From our point of view, considering the characteristics of the proposed algorithms from the literature, fair comparisons in the biconnected case can be made by comparing:

- Blackbone with Blackbone2 for $2-$vertex connected $1-$vertex dominating sets. In Figure 6.5a, we can observe that Blackbone produces compact solutions only when the network density is not too high. Such a problem is overcome with Blackbone2 which always generates more compact solutions.

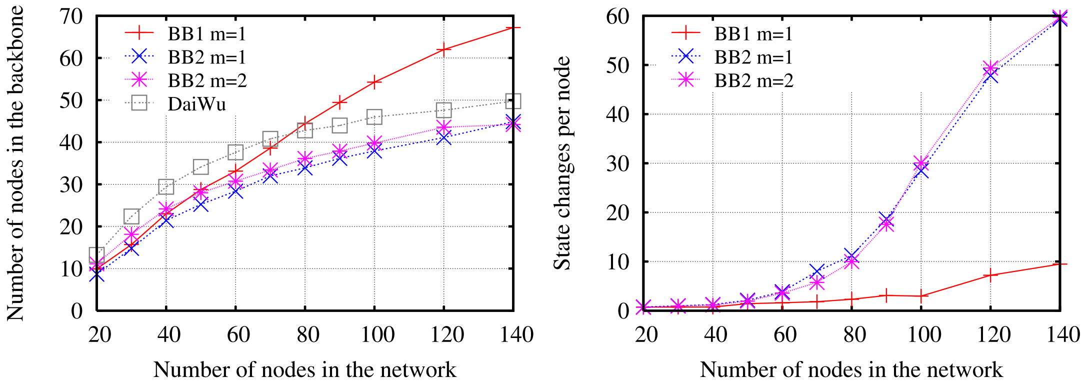
- Blackbone2 with the $k-$coverage condition introduced by DaiWu for $2-$vertex connected $2-$vertex dominating sets. In Figure 6.5a, we observe that Blackbone2 provides more compact virtual backbone than the coverage condition of Dai *et al.* whatever the network density.

In both cases we can observe that Blackbone2 gives better results for all the simulated networks. Moreover, Blackbone2 can be computed in $\mathcal{O}(\Delta^2)$ when the first version of Blackbone required $\mathcal{O}(\Delta^3)$. Moreover, the k-coverage condition can be computed in $\mathcal{O}(k\Delta^4)$. This high computation cost may be an important factor when dealing with mobile environments, where information may be incomplete and available computation time is short. All computation times can be found in table 6.7.

Although the computation time and the generated solution are much better, Blackbone2 suffers from high convergence delays. This is probably due to a more constrained construction mechanism in the first Blackbone algorithm that induces less freedom in the exploration of the solution space. As a consequence we can conclude that in Blackbone2 there are more possibilities to achieve a better solution, but the exploration process takes more time. These results can be seen on Figure 6.5b.

6.2.3.3 Performance comparison for k=1-vertex connected DS

For the one-vertex connected case, we propose to compare Blackbone2 performances with Blackbone, WuLi(*) and the $k = 1-$coverage condition.



(a) Size of the virtual backbone (b) State changes before convergence

Figure 6.5: Results comparison for the biconnected virtual backbone

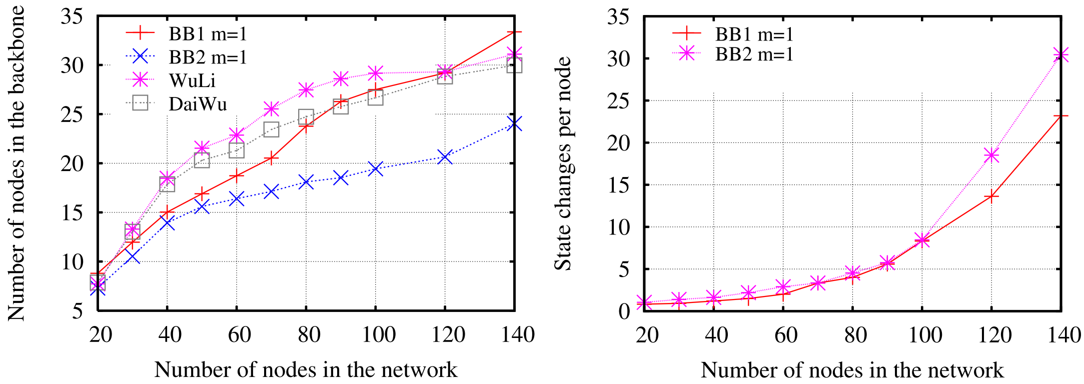
Algorithm	**Computation time**
WuLi(*)	$\mathcal{O}(\Delta^2)$
Blackbone2	$\mathcal{O}(\Delta^2)$
Blackbone	$\mathcal{O}(\Delta^3)$
Coverage Condition	$\mathcal{O}(\Delta^3)$
k Coverage Condition	$\mathcal{O}(k\Delta^4)$

Table 6.7: Computation time of the compared algorithms

In Figure 6.6a we can observe that Blackbone2 creates more compact virtual backbone whatever the size of the network. Here again the first version of Blackbone obtains good results until the network density becomes to high. Indeed, when there are over 80 nodes, DaiWu's algorithm ($k-$coverage condition) obtains better results than Blackbone. In the very high scenarios, even WuLi(*) creates more compact connected dominating set than Blackbone.

Blackbone2 surpasses Blackbone concerning the size of its generated virtual backbone and also requires less computation time $O(\Delta^2) < O(\Delta^3$. DaiWu algorithm obtains good results but suffers from a very high computation cost $O(k\Delta^4)$. All the theoretical computation times can be found in table 6.7.

In Figure 6.6b, the first version of Blackbone obtains better results concerning the required number of state changes until convergence. This characteristic is particularly suitable in mobile environments in which topology changes induce recurrent backbone maintenance. Blackbone2 improvements are proposed in 6.2.4 to increase the speed of convergence in mobile environments.



(a) Size of the virtual backbone

(b) State changes before convergence

Figure 6.6: Results comparison for connected dominating set virtual backbones

6.2.3.4 Impact of the vertex domination value

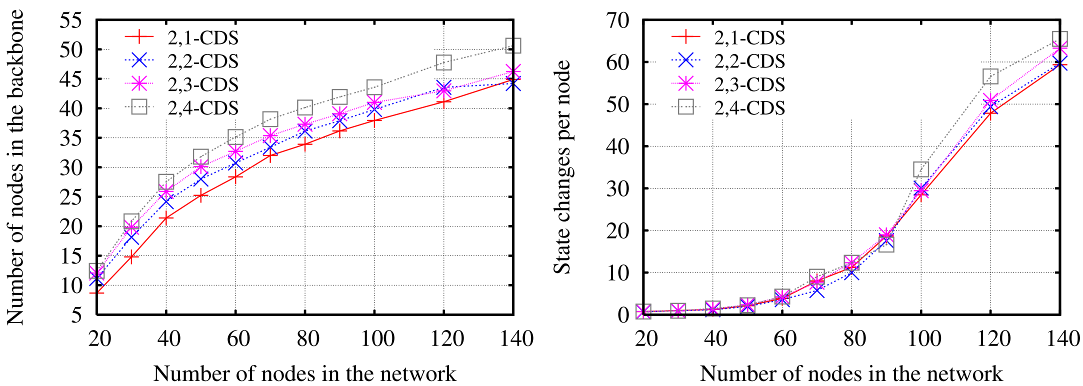
In this part we present the impact of the vertex domination value on the results of the Blackbone2 algorithm.

In Figure 6.7a, we can observe that the Blackbone2 algorithm require few additional backbone nodes when increasing the vertex domination. Moreover, the impact of the domination parameter is negligible in very dense networks for $m \in \{1, 2, 3\}$. Indeed, when there are 140 nodes in the simulation space, the results are almost the same for the three first values of m. This is most probably due to the increasing density, a single backbone node will cover more nodes when the density is higher. These results prove that increasing the robustness of the backbone by increasing the vertex domination does not impact too much on the compactness of the virtual backbones generated by Blackbone2.

Figure 6.7b provides details concerning the required quantity of state changes before converging to a valid solution. In low density networks, increasing the vertex domination has a negligible impact. However, we can see that the difference is more important when the number of the nodes increases. As previously stated, the main drawback of the Blackbone2 algorithm lies in number of state changes before the convergence. Moreover, this necessary quantity is directly correlated to the network density. This observation can be explained by the fact that higher density means more possibilities for some *domino effect*: if a node changes its state, some other nodes may also change.

6.2.3.5 Results analysis

Blackbone2 obtains very encouraging results in static networks as it surpass all the other compared algorithms. Its time efficient design allows him to be deployed on real devices, even if the convergence of the algorithm is its major drawback. In the next simulations, we provide an exhaustive study of Blackbone2's performances in mobile environments.



(a) Size of the virtual backbone

(b) State changes before convergence

Figure 6.7: Impact of the vertex-domination value

6.2.4 Blackbone 2 - dynamic networks

In this part we propose a detailed study of the Blackbone2 algorithm in mobile environment with respect to the different quality measures introduced in 3.3. Some optimizations are also proposed to enhance the performances of the genuine Blackbone2 algorithm.

6.2.4.1 Experiment specific settings

Number of runs.

These experiments have been repeated for a different numbers of nodes (20, 30, 40 up to 100), to deal with a wide range of average node degree or density $[0, 10]$.

For each network size and for each speed value we run 30 different simulations in order to compute average values. The run number is also used as a seed for the random number generator used by $OMNeT++$ to assign the initial position of the nodes and produce their movements.

Simulation space.

The simulation space is a 215x125-meter rectangle and the transmission range has been fixed to 25 meters. These arbitrary values have been chosen for they were well-suited to the screen and the OMNeT++ graphical user interface.

Mobility settings.

We want to test the algorithms in high mobility conditions to check if Blackbone2 can realistically be implemented in real moving devices. We chose to use the random waypoint mobility model to generate the network nodes movements. We are aware that this model does not mimic any realistic behavior. However, we think that this artificial mobility model may represent the worst case in terms of stability for the communication graph because of its randomness and thus it is a particularly challenging benchmark. In a particular simulation sim, all nodes of the network have the same speed $s_{sim} \in [0:7]$ meters per seconds.

Algorithm specific settings.

The different versions of the Blackbone2 algorithm presented in 5.4.4.4 and 5.4.4.5 have also been tested with different parameter values. The heuristics that reduces the considered subset of nodes by selecting *old enough* neighbors has been tested for different age threshold values equal to $Th_a = x * Th_B$ (see table 6.5) with $x \in [0:10]$. The case $Th_a = 0$ corresponds to the regular algorithm.

The second reduced subgraph optimization considers neighbors with a *small enough* relative speed. Different speed threshold, $Th_s \in [0:10]$ have also been tested. The regular algorithm corresponds to $Th = \infty$.

The last optimization that requires an additional parameter is the *forced stability* version. The period for which a backbone node has to stay before considering leaving it is also based on the beacon period: $Th_{stab} = x * Th_B$ with $x \in [0:7]$.

Considered graph for statistic measurements.

The communication graph is an abstract notion that is regularly used in the literature to compare solutions of a proposed algorithm with the so-called optimal solution. In this graph, the vertices represent the nodes of the network and an edge exists if and only if two nodes are in each others communication range.

Communication graph is an abstract notion because in real ad hoc networks, nodes discover themselves by sending small packets called beacons, and thus, even if an edge exists between two nodes in the communication graph, these nodes may not know each other in practice. This observation is particularly true in mobile contexts where local information may be incomplete or partially wrong. As we are dealing with algorithms that are designed to create a backbone on top of a *given* graph, it would be unfair to compare results with the communication graph, as it may differ from the merging of all the local graphs on real devices or in a simulator.

For this reason, we propose not to use the communication graph to compute the measures. Instead we will consider the graph $G_m = (V_m, E_m)$ obtained by merging all the local information. The vertices set, V_m contain all the nodes of the network and an edge between two nodes u and v exists in E_m if and only if v is in the direct neighbors' list of node u and u is in the direct neighbors' list of node v. As a consequence, E_m can be defined as follow:

$$E_m = \left\{(u, v) \;/\; u \in N_1(v) \;\wedge\; v \in N_1(u)\right\}$$

6.2.4.2 Age and speed threshold do not work

The idea behind the usage of threshold concerning the age of the relative speed between nodes is quite straightforward. Indeed, nodes should preferably consider stable neighbors as they are more likely to help achieving successful communication sessions. Having a higher threshold was not supposed to reduce the backbone size but it was expected to increase the stability as the backbone should have been less sensitive to short term topology changes, i.e. new neighbors quickly disappearing.

However, it seems that having such threshold is not helping, at least with the random mobility model. Indeed, in Fig. 6.8a we can observe that the age threshold does not change the size of the given solutions. However, in Fig. 6.8b the state changes are worse if the threshold is increased. This seems to be even worse with low-density networks for which the stability difference is maximal.

Concerning the speed threshold, the results are even worse. Figure 6.9a shows that the relative speed threshold induces a small increase in the size of the backbone. Moreover, the stability is also not increased. Indeed, Fig. 6.9b shows that the genuine version is a lot better than the relative speed variant.

These results are most probably induced by the randomness nature of the chosen mobility model. However, we think that using this model is a good indicator as it can be considered the worst possible scenario.

6.2.4.3 Performance comparison for each quality criterion

In this subsection we compare the performances of the different *Blackbone 2* variants and the Dai and Wu algorithm ($k-$coverage condition) with respect to the previously presented quality measures. We have chosen this algorithm as it is the only one meeting our requirements. First, the algorithm should be distributed and it should rely on local information only. Second, it should be composed of one computational step as multiple-step algorithms rely on synchronicity which is not suitable in an asynchronous environment.

Size.

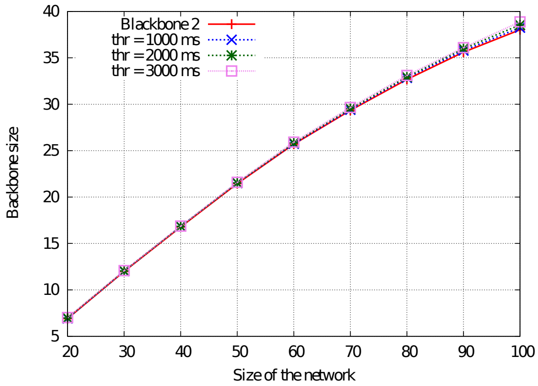
In Fig. 6.10a and 6.10b we can observe the percentage of nodes used by the different algorithms versus the increasing size of the network. The genuine version of the algorithm and the high degree variant show the most compact backbones independently of the required robustness: $k = m = \{1, 2\}$. The forced stability variant gives also very encouraging results when the robustness is set to 2, i.e. two-vertex connected and two-vertex dominating structures. Dai and Wu obtains better performances than the forced stability variant in one particular case: low robustness ($k = m = 1$) and dense networks.

In Fig. 6.11a and 6.11b the backbone size performance is displayed versus the speed of the nodes. We observe the same kind of results than previously: best results are achieved by the genuine algorithm and the high degree variant. An additional interesting information is that these two algorithms are characterized by their steadiness: the size of the solutions remain stable. In the low robustness backbones, the Dai and Wu algorithm achieves better results when the mobility is higher. The forced stability variant however, show a inclination to build less compact solution when the nodes move quickly. This problem is almost overcome when dealing with more robust backbones.

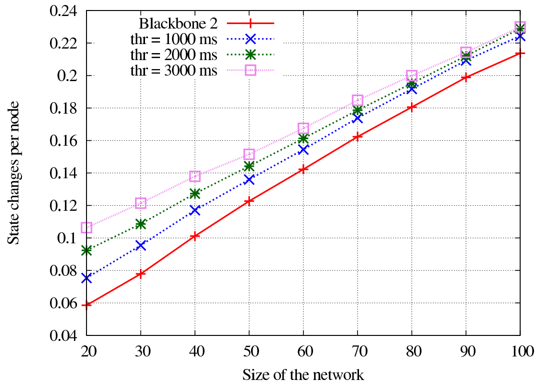
Stability.

In Fig. 6.12a and 6.12b we can observe that the algorithm creating the most stable backbone is Dai and Wu: the number of state changes during the simulation time is a lot less important compared to the different version of Blackbone2. This important difference is due to one main reason: in the Dai and Wu algorithm, a node checks a criterion based on its complete neighborhood when the Blackbone2 algorithm consider two different sets of neighbors. Indeed, in the Blackbone2 algorithm, a node checks if its non-backbone neighbors are covered and if its backbone neighbors are enough connected. These two sets are of course more frequently likely to change during the simulation time than the complete set of neighbors and thus the Blackbone2 algorithm will induce more state changes in order to adapt to these updates. That reason explains the better performances of the two variants: high degree and forced stability. In these two cases, some nodes are temporarily fixed into a previous state based on a particular condition.

The forced stability variant obtains the best results if we consider Blackbone2 variants only. However, its performances mainly depend on the value of its parameter as we can see in Fig. 6.16. Figures 6.13a and 6.13b show that the state changes are also dependent of the speed of the nodes. However the Dai and Wu algorithm is characterized by a slow and almost linear increasing when the Blackbone variants suffer from a high rise as soon as the mobility appears.

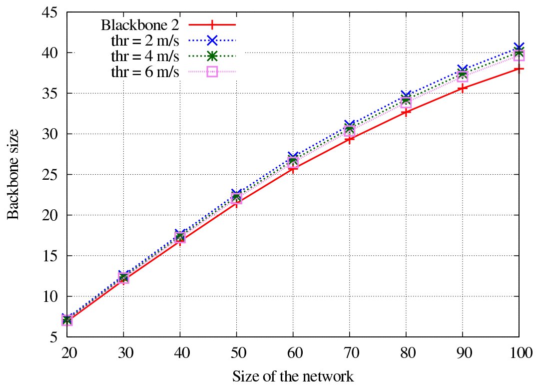


(a) Size of the virtual backbone

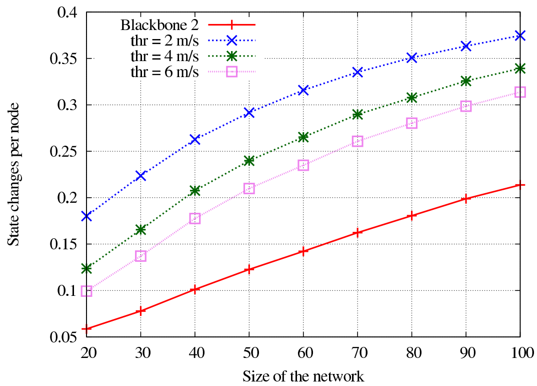


(b) State changes before convergence

Figure 6.8: Results comparison with different age threshold values

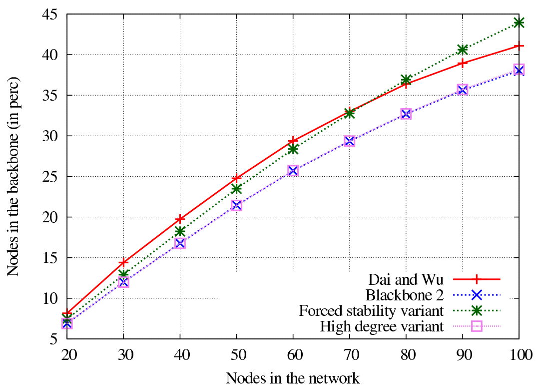


(a) Size of the virtual backbone

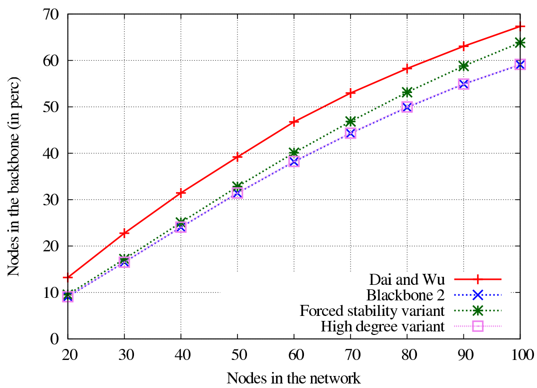


(b) State changes before convergence

Figure 6.9: Results comparison with different speed threshold values

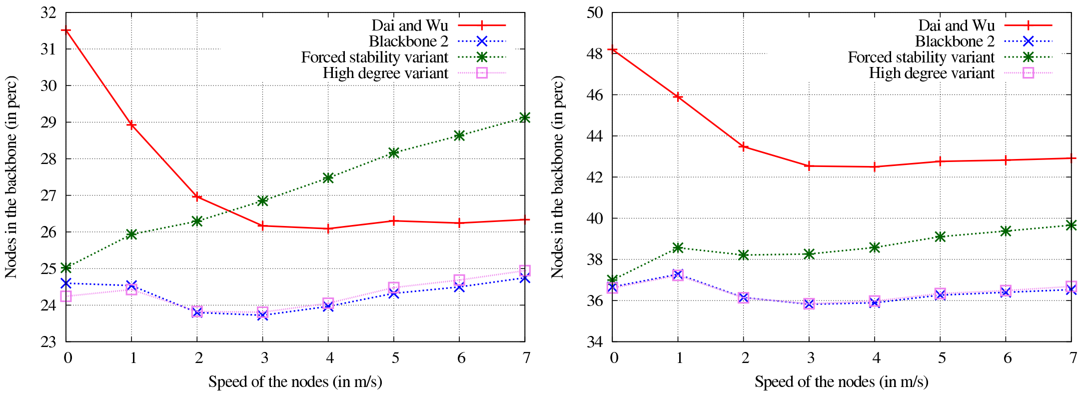


(a) $k = m = 1$



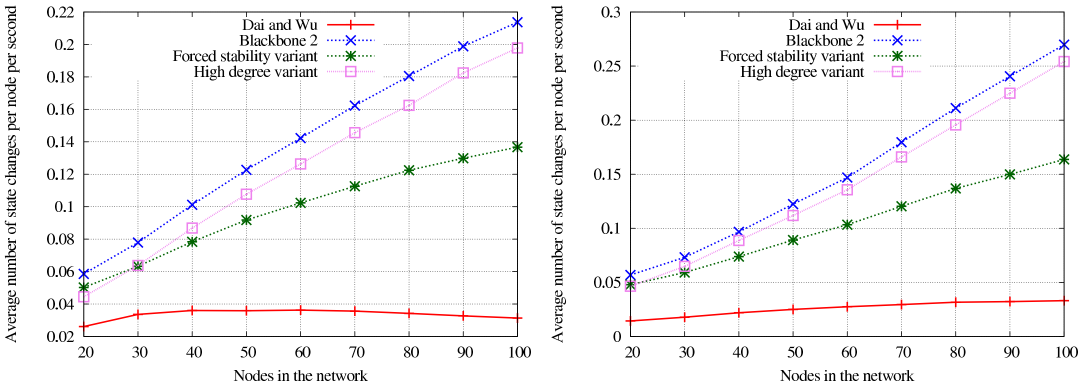
(b) $k = m = 2$

Figure 6.10: Influence of the density on the backbone size



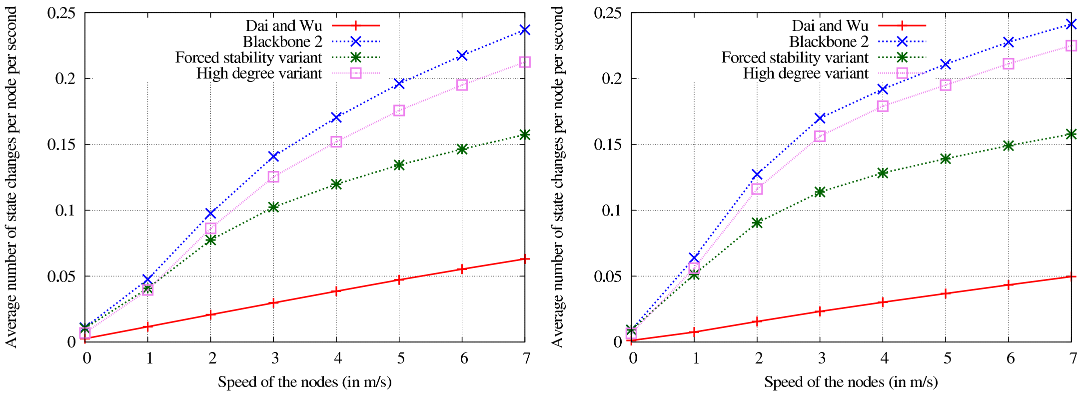
(a) $k = m = 1$ (b) $k = m = 2$

Figure 6.11: Influence of the speed on the backbone size



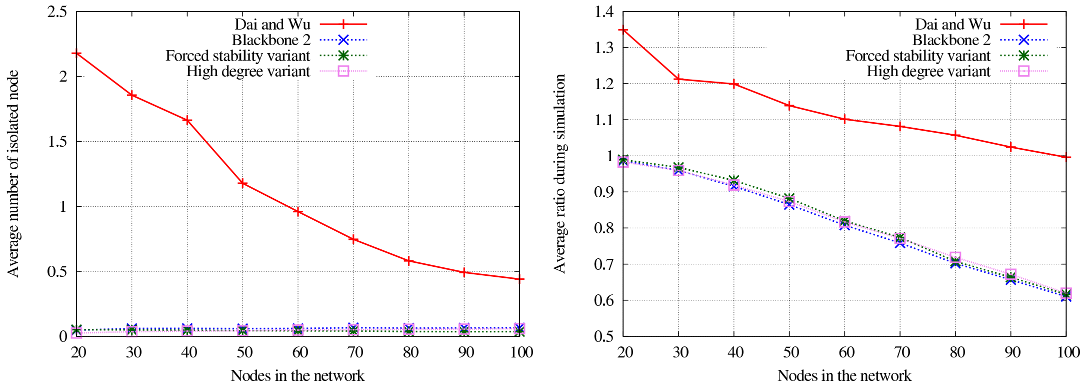
(a) $k = m = 1$ (b) $k = m = 2$

Figure 6.12: Influence of the density on the number of state changes



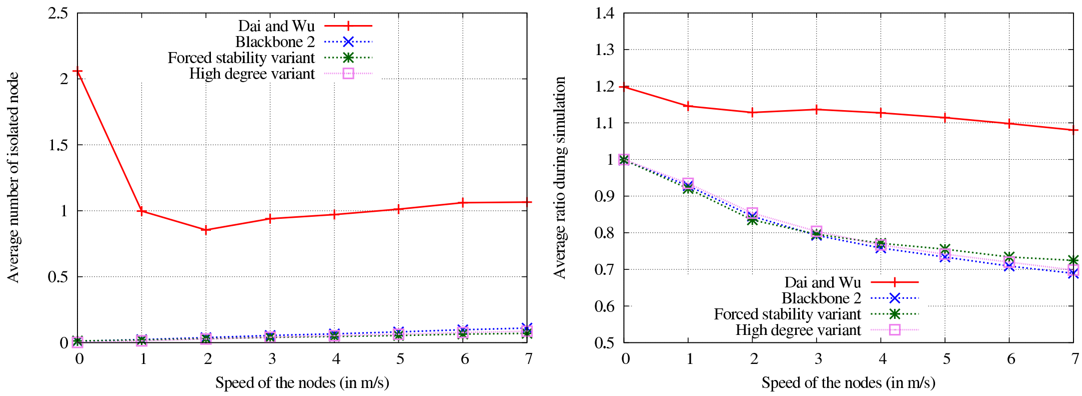
(a) $k = m = 1$ (b) $k = m = 2$

Figure 6.13: Influence of the speed on the number of state changes



(a) Average number of isolated nodes (b) Connected component ratio

Figure 6.14: Influence of the density $k = m = 1$



(a) Average number of isolated nodes (b) Connected component ratio

Figure 6.15: Influence of the speed $k = m = 1$

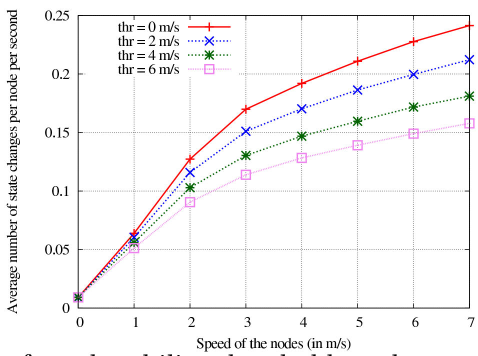


Figure 6.16: Influence of the forced stability threshold on the average number of state changes ($k = m = 2$)

Availability.

The availability criterion is composed of two different measures: the connected component ratio and the number of isolated nodes. These two measures are designed to quantify the quality of service proposed by the backbones. Our results also show that these two measures are antagonist in most of the cases: a worse cover of the non-backbone nodes may induce a better ratio. Let us take the example of a simple connected component composed of two connected nodes u and v. If these two nodes are not in the backbone, then the total number of connected components CC is increased by one and the number of connected component induced by backbone nodes, CC_{VB}, remains the same.

For the results concerning the isolated nodes, we propose two figures, 6.14a and 6.15a, that summarize the performances in the low robustness case. The results are almost identical in the more robust case. From these two figures we can conclude that the Blackbone2 variants obtain significantly better results than the Dai and Wu algorithm. Indeed, the Blackbone2 variants remain stable whatever the size of the network or the speed of the nodes. Dai and Wu algorithm on the contrary, shows very fluctuant results depending on the simulation context. This is particularly surprising considering the bigger size of its generated backbones. As a consequence, we suppose that Blackbone2 variants are able to pick up nodes in a way that reduces isolated nodes even with less backbone nodes.

However, the results are completely different concerning the connected components ratio. In Fig. 6.14b and 6.15b we can see that the Blackbone2 variants obtains a decreasing ratio when the density or the speed increases. These results can be explained by the increasing number of state changes required when these two parameters are increased. The Dai and Wu algorithm shows very good results independently of the context: the ratio stays above 1. These good results can however be partially explained by the poor results concerning the isolated nodes. Moreover, values above 1 can be considered a lack of adaptability as the algorithm may not have adapted itself quick enough to the topology changes. This low-adaptability characteristic also explains the high count of isolated nodes.

6.2.4.4 Results summary

In a random mobility context we can conclude that the technique of reducing the considered neighborhood does not help achieve better quality solutions. It is however possible that the results would be completely different with more realistic mobility models in which groups of nodes are moving in the same directions (e.g. cars on highways or city street, pedestrian in urban streets). However, as previously stated, we consider this mobility model the worst possible scenario and thus a good indicator concerning the quality of a given optimization.

We have seen that the high degree variant produces almost the same compact backbones than the genuine algorithm but increases the stability. Its performances for the other quality measures are also very close to those of the original version. As a consequence, the absence of trade-off induces that this variant should be considered a replacement for the original algorithm.

The forced stability version of the Blackbone2 algorithm is the variant that benefits from the best stability. Even if the backbone size overhead with this technique exists, it can still be considered a good solution when a more stable backbone is required. Moreover, as all Blackbone2 variants, its theoretical computation time, $\mathcal{O}(\Delta^2)$, allows it to be deployed on a wide panel of devices. However, even though this variant obtained the best stability results among the other variant, its stability performances are still far below those of the Dai and Wu algorithm.

The Dai and Wu algorithm obtains impressive results concerning the stability of the backbone and the connected component ratio. However this algorithm may not be suitable when in dense networks for two main reasons. First, the percentage of nodes required to create the backbone is higher than the Blackbone2 variants in most of the cases. Second, the theoretical computation time is very high, $\mathcal{O}(k\Delta^4)$, with Δ the average density, which may be a problem with not powerful devices. However, having more stable backbones than those from the Blackbone2 variants is leading to the reduction of the availability-related quality metrics. Indeed, the number of isolated nodes together with the high ratio values proves that this algorithm does not adapt quickly to the topology changes.

6.2.5 Results analysis

We empirically studied the validity of two distributed algorithms and some variants via extensive simulations. For the latter work, we also studied the impact of the mobility on various quality criteria.

The first version Blackbone obtains very promising results provided the network density is not too high. Moreover, these results do not rely on the synchronization of the network in contrary to WuLi(*). The poor results in high density situations are most probably due to the robustness property which is too constraining and thus adds too many nodes into the backbone.

Blackbone2 obtains very encouraging results in static networks as it surpasses all the other compared algorithms. It is not based on the robustness property anymore and as a consequence, its time efficient design allows Blackbone2 to be deployed on real devices, even if the convergence of the algorithm is its major drawback.

Two interesting variants have been proposed in the last work. The first one is the high degree variant that produces almost the same compact backbones than the genuine algorithm but increases the stability. The second one is the forced stability version of the Blackbone2 algorithm, which benefits from the best stability. Even if a small size overhead exists with this technique, it can be considered a good solution when a more stable backbone is required.

6.3 Recap of key points

- Finding an optimal solution for the minimum dominating set is quickly non tractable when the size of the input is increasing.
- The centralized algorithm based on DC programming and DCA is proved to be more efficient than the best $0-1$ solver, ILOG CPLEX, to find sub-optimal solutions in reasonable time.
- A greedy approximation algorithm is also compared to our DCA. Its solution are always bigger but its computation time outperform CPLEX and our DCA on average. In low density graphs however, the greedy approach requires much more time than the other approaches.
- The first version of Blackbone obtains good results in low and moderate network density graphs. The construction rule on which Blackbone is based perform poorly when the graph density increases because it is too constraining.
- Blackbone2 fixes Blackbone's problem and thus obtain very promising results outperforming all the other compared algorithms.
- Blackbone2 and many variants have been extensively tested for mobile networks. Variants that were reducing the considered sub-graph to increase the stability performed poorly, probably as a consequence of the randomness of the selected mobility model.
- The high degree variant of Blackbone2 produces almost the same compact backbones than the genuine algorithm but increases the stability.
- The forced stability version of the Blackbone2 algorithm benefits from the best stability. It can be considered a good solution when a more stable virtual backbone is required although it induces a small size overhead.

Chapter 7

Conclusions and perspectives

Contents

7.1 Conclusions **134**
7.1.1 Centralized algorithm 134
7.1.2 Distributed algorithms in static networks 134
7.1.3 Distributed algorithms in dynamic networks 135
7.2 Perspectives **135**

7.1 Conclusions

In this thesis we studied various aspects of the virtual backbone design, from an Operational Research point of view with a centralized algorithm based on mathematical programming to more practical solutions with distributed and localized algorithms.

The contributions of the dissertation to the addressed problem can be summarized as below.

- An extensive state-of-the-art on the $CDS-$based virtual backbone techniques. Centralized, distributed and robust algorithms are classified independently as their most important characteristics differ.
- Modeling of the $m-$vertex domination and the $l-$level domination constraints for the general minimum $m-$vertex $l-$level dominating set problem. These two contributions extend the original and well-known integer programming model of the minimum dominating set problem.
- A centralized DC Algorithm to solve the minimum $m-$vertex $l-$level dominating set problem. A restart mechanism is also proposed to increase the exploration of the solution space. This approach outperforms the reference commercial solver in both computation time and quality of the solution.
- Two distributed and localized algorithms, Blackbone 1 and 2, designed to create $k-$vertex connected $m-$vertex dominating set virtual backbones in an asynchronous and computer effective fashion. Both versions proposed an original design that permits a more intensive exploration of the solution space and thus obtain more compact solutions compared to state-of-the-art distributed algorithms.
- A set of quality metrics to analyze the behavior of $CDS-$based virtual backbones algorithm in mobile ad hoc networks. Although some of the metrics have already been used for various purposes in different sub-domains, collecting them as a toolbox to analyze various quality aspects of CDS-based virtual backbones is an original contribution of this thesis.

7.1.1 Centralized algorithm

We adopted an Operational Research point of view to deal with the minimum $m-$vertex dominating set problem for we think that theoretical guaranties are important yet not sufficient when dealing with solving NP-Hard optimization problems.

As a consequence, developing a near-optimal algorithm based on DC programming and DCA allowed us to test much bigger instances of graphs than what can usually be optimally solved. We empirically validated our algorithm and showed that it is more robust and efficient than the CPLEX solver from ILOG software via extensive tests on random graphs.

7.1.2 Distributed algorithms in static networks

We proposed and compared two distributed and localized algorithms to create connected dominating set based virtual backbones as we think building such structures in ad hoc networks should be easily implemented and scaled when the network increases. Comparing our results to the state-of-the-art algorithms for the static network case was a necessary step in our validation process.

Both proposed algorithms offered a fundamentally different design than what is found in the literature: the necessary number of steps before convergence is not bounded. Although this characteristic can be considered as a theoretical disadvantage, it enables us to achieve a better exploration of the solution space and thus obtain more compact virtual backbones.

The first version of Blackbone obtained encouraging results in low to moderately dense networks. That lead us to tackle its fundamental problem: only backup routes composed of one relay node were considered. Blackbone2 solved these issues and thus always obtained more compact virtual backbone whatever the network density. In this second version, the computation time has been reduced to the strict minimum when dealing with non-probabilistic approaches: $O(\Delta^2)$ with Δ representing the maximum node degree of the communication graph.

7.1.3 Distributed algorithms in dynamic networks

Simulations with moving network nodes have been extensively studied and the results comfort us in the idea that our algorithm can be implemented on real ad hoc network without inducing too much protocol overhead (computation time and messages). By considering a random mobility model, we proved that the Blackbone2 algorithm is capable of good overall performances even in the worst possible situation.

During this work, many variants of the original algorithm have been tested and some proved to be promising. Considering its performances, the high degree variant can replace the original algorithm without any trade-off. The forced stability version provided encouraging results to increase the stability of the generated virtual backbone and induced a very small size overhead.

All these results were obtained thanks to the different metrics that have been introduced to quantify many different quality aspects of the virtual backbones. This tool box revealed to be quite helpful to analyze more precisely the behavior of the tested algorithms.

7.2 Perspectives

The DC algorithm designed and tested during this work provided us with good results for both the quality of the solutions and the required computation time. Although these results are encouraging, the computed solution are still sub-optimal. Recently, a new cutting edge technique developed for the DCA realm has allowed to reach optimal solutions. We think that including such a promising technique in our next work would permit us to find the optimal solutions for big instances of graphs and thus obtain a decisive advantage compared to the other approaches.

Another short term perspective lies in integrating mobility to our centralized approach. As the DC algorithm relies on the complete network topology to compute its solutions, modeling the stability along with the compactness of the structure can be envisaged and as a consequence, the problem could be solved via multi-objective techniques based on the DC approach.

For the distributed algorithms, we can think of many future work. In a first time, an extension of the Blackbone2 algorithm to create tri-connected or even $k-$connected structures in an computer efficient way can be envisaged. The $l-$level domination also offers a lot of research opportunities even if it may be constrained to specific cases, such as $l = 2$ due to the available local topological information. Finally, an implementation on real devices would complete the validation process and could permit to fine tune our algorithm with respect to real world constraints.

Part II

Appendices

Appendix A

Scripts for DCA

Contents

A.1 AMPL models **140**
A.1.1 Integer programming model for min m-DS 140
A.1.2 Linear programming model for min m-DS 140
A.1.3 DCA model for min m-DS 141
A.2 Random graph generator implementation **142**

A.1 AMPL models

In this appendix we provide the mathematical model we use to solve the different instances of the min $m-$vertex dominating set problem. To such extend we have chosen the AMPL language for its compactness and modularity.

A.1.1 Integer programming model for min m-DS

This first model presents the original problem with a binary variable representing the dominating set. The parameter of the problem are the size of the network (the scalar value *size*), the available wireless connections between the network nodes (the *edges* matrix) and the vertex domination value (the scalar value m).

```
# Problem parameters
param size;
param edges{i in 1..size, j in 1..size};
param m{i in 1..size};

# Variables
var x{i in 1..size} binary;

# Objective function
minimize
   setsize: sum{i in 1..size} x[i];

# Constraints
subject to
   domconst{i in 1..size}: sum{j in 1..size} edges[i,j]*x[j] >= m[i];
```

A.1.2 Linear programming model for min m-DS

The second model is the linear relaxation of the original problem and as such the *binary* keyword does not appear. This model is used to compute the initial point of our DC algorithm.

```
# Problem parameters
param size;
param edges{i in 1..size, j in 1..size};
param m{i in 1..size};

# Variables
var x{i in 1..size};

# Objective function
minimize
   setsize: sum{i in 1..size} x[i];

# Constraints
subject to
   domconst{i in 1..size}: sum{j in 1..size} edges[i,j]*x[j] >= m[i];
```

A.1.3 DCA model for min m-DS

This last model presents the necessary modifications to apply DC programming and DCA to solve the original problem. Two additional parameters are required: a scalar value p representing the weight of the exact penalty function and the solution vector x at iteration step k (the vector x_k).

```
# Problem parameters
param size;
param edges{i in 1..size, j in 1..size};
param m{i in 1..size};

# DCA parameters
param p;
param x_k{i in 1..size};

# Variables
var x{i in 1..size};

# Objective function
minimize
    setsize: sum{i in 1..size} ( x[i] * (1+p) - p * x_k[i] * x[i]);

# Constraints
subject to
    domconst{i in 1..size}: sum{j in 1..size} edges[i,j]*x[j] >= m[i];
```

A.2 Random graph generator implementation

This appendix provide the exact implementation of the random graph generator used for our experiments. It is developed in Java and for the sake of clarity we decided to hide all the source code but the main procedure.

```
package graphGenerator;

import org.miv.graphstream.graph.implementations.DefaultGraph;
import java.util.Random;

public class GraphGenerator {

  public void generateGraph(int nbNodes, int avgDegree, int seed) {

    DefaultGraph gra = new DefaultGraph();
    Random generator = new Random(seed);

    /// Adding nodes to the graph
    for (int i=0; i<nbNodes; i++) {
      gra.addNode(Integer.toString(i));
    }

    /// The matrix containing the edges (adjacency matrix)
    Integer[][] edgesTable = new Integer[nbNodes][nbNodes];

    /// Initialization of the graph
    for (int i=0; i<nbNodes; i++)
      for (int j=0; j<nbNodes; j++)
        edgesTable[i][j]=0;

    for (int i=0; i<nbNodes; i++)
      edgesTable[i][i]=1;

    /// Randomly adding edges
    int nbEdges=0;
    while(nbEdges < avgDegree * nbNodes) {
      int node1 = (int)(generator.nextDouble() * nbNodes);
      int node2 = (int)(generator.nextDouble() * nbNodes);
      if (node1 != node2) {
        if (edgesTable[node1][node2] == 0) {
          gra.addEdge(Integer.toString(node1)+"_"+Integer.toString(node2),
                      Integer.toString(node1), Integer.toString(node2));
          edgesTable[node1][node2] = 1;
          edgesTable[node2][node1] = 1;
          nbEdges += 2;
        }
      }
    }
  }
}
```

Appendix B

Simulation parameters for OMNeT++

Contents

B.1 omnetpp.ini 144
B.2 params.ini 145

B.1 omnetpp.ini

This appendix proposes the complete set of parameters used for the OMNeT++ simulator and the Mobility Framework during all our simulations. Another set of parameters related to the tested algorithm and the simulation run are stored in a separate file named *params.ini*. An example of such a file can be found in B.2.

```
[General]
network = sim
sim-time-limit = 200s

[Parameters]
include params.ini

#####################################################
#         Parameters for the simulation             #
#####################################################
sim.playgroundSizeX = 1300
sim.playgroundSizeY = 750

#####################################################
#         Parameters for the ChannelControl         #
#####################################################
sim.channelcontrol.carrierFrequency = 868e+6
# max transmission power [mW]
sim.channelcontrol.pMax  = 3
# signal attenuation threshold [dBm]
sim.channelcontrol.sat   = -110
sim.channelcontrol.alpha = 3.83986
sim.channelcontrol.sendDirect = 0
sim.channelcontrol.useTorus = 0

######################################################
#         Parameters for the Mobility Module         #
######################################################
# starting position for the hosts "-1" means random
sim.host[*].mobility.x=-1
sim.host[*].mobility.y=-1

description = "ConstSpeedMobility"
**.host*.mobilityType = "ConstSpeedMobility"
**.host*.mobility.updateInterval = .1

######################################################
#         Parameters for the Application Layer       #
######################################################
sim.host[*].appl.headerLength=0

######################################################
#         Parameters for the Network Layer           #
######################################################
```

```
sim.host[*].net.headerLength=24

##########################################################
#         Parameters for the MAC Layer                    #
##########################################################
sim.host[*].nic.mac.queueLength=5
sim.host[*].nic.mac.headerLength=24
sim.host[*].nic.mac.busyRSSI=-97
sim.host[*].nic.mac.slotDuration=0.01
sim.host[*].nic.mac.difs=0.006
sim.host[*].nic.mac.maxTxAttempts=14
sim.host[*].nic.mac.defaultChannel = 0
sim.host[*].nic.mac.bitrate = 15360
sim.host[*].nic.mac.contentionWindow = 31

#########################################################
#         Parameters for the radio                       #
#########################################################
sim.host[*].nic.radio.swSleep = 0
sim.host[*].nic.radio.swSend = 0.001
sim.host[*].nic.radio.swRecv = 0.003

##########################################################
#         Parameters for the Physical Layer               #
##########################################################
sim.host[*].nic.snrEval.publishRSSIAlways = 0
sim.host[*].nic.snrEval.headerLength=16
# transmission power [mW]
sim.host[*].nic.snrEval.transmitterPower=3
sim.host[*].nic.snrEval.carrierFrequency=868E+6
sim.host[*].nic.snrEval.thermalNoise=-120
sim.host[*].nic.snrEval.sensitivity=-110
sim.host[*].nic.snrEval.pathLossAlpha=3.5
sim.host[*].nic.decider.debug = 0
sim.host[*].nic.decider.snrThresholdLevel=10;[dB]
```

B.2 params.ini

```
# Number of network nodes
sim.numHosts=100

# Speed of nodes
**.host*.mobility.speed=0

# Vertex domination and vertex connectivity values
sim.host[*].net.domination = 1
sim.host[*].net.connectivity = 1
```

References

[1] Cédric Adjih, Emmanuel Baccelli, Thomas Clausen, and Philippe Jacquet. On the robustness and stability of connected dominating sets. Rapport de recherche 5609, INRIA, sept 1997.

[2] Cedric Adjih, Philippe Jacquet, and Laurent Viennot. Computing connected dominated sets with multipoint relays. *Ad Hoc & Sensor Wireless Networks*, 1(1-2), 2005.

[3] Shoukat Ali, Anthony A. Maciejewski, Howard Jay Siegel, and Jong-Kook Kim. Measuring the robustness of a resource allocation. *IEEE Trans. Parallel Distrib. Syst.*, 15 (7):630–641, 2004.

[4] Khaled M. Alzoubi, Peng-Jun Wan, and Ophir Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *MobiHoc*, pages 157–164. ACM, 2002. ISBN 1-58113-501-7. URL http://doi.acm.org/10.1145/513800.513820.

[5] Le Thi Hoai An and Pham Dinh Tao. Solving a class of linearly constrained indefinite quadratic problems by dc algorithms. *Journal of Global Optimization*, 11(3):253–285, 1997.

[6] Le Thi Hoai An and Pham Dinh Tao. A continuous approach for large-scale constrained quadratic zero-one programming. (in honor of professor elster, founder of the journal optimization). *Optimization*, 45(3):1–28, 2001.

[7] Le Thi Hoai An and Pham Dinh Tao. D.c. programming approach for multicommodity network optimization problems with step increasing cost functions. *J. of Global Optimization*, 22(1-4):205–232, 2002. ISSN 0925-5001. doi: http://dx.doi.org/10.1023/A:1013867331662.

[8] Le Thi Hoai An and Pham Dinh Tao. Large scale molecular optimization from distances matrices by a dc optimization approach. *SIAM Journal of Optimization*, 14(1):77–116, 2003.

[9] Le Thi Hoai An and Pham Dinh Tao. The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems. *Annals of Operations Research*, 133(1-4):23–46, 2005.

[10] Le Thi Hoai An and Pham Dinh Tao. A continuous approach for the concave cost supply problem via dc programming and dca. *Discrete Applied Mathematics*, 156(3):325–338, 2008.

[11] Le Thi Hoai An, Pham Dinh Tao, and Le Dung Muu. Exact penalty in dc programming. *Vietnam Journal of Mathematics*, 27(2):169–178, 1999.

[12] Le Thi Hoai An, Nguyen Trong Phuc, and Pham Dinh Tao. A continuous dc programming approach to the strategic supply chain design problem from qualified partner set. *European Journal of Operational Research*, 183(3):1001–1012, 2007.

[13] David A. Bader and Guojing Cong. A fast, parallel spanning tree algorithm for symmetric multiprocessors (smps). *J. Parallel Distrib. Comput.*, 65(9):994–1006, 2005. ISSN 0743-7315. doi: http://dx.doi.org/10.1016/j.jpdc.2005.03.011.

[14] S Basagni, M Mastrogiovanni, A Panconesi, and C Petrioli. Localized protocols for ad hoc clustering and backbone formation: a performance comparison. *Parallel and Distributed Systems, IEEE Transactions on DOI - 10.1109/TPDS.2006.52*, 17(4):292–306, 2006.

[15] Christian Bettstetter. Smooth is better than sharp: a random mobility model for simulation of wireless networks. In *MSWIM '01: Proceedings of the 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 19–27, New York, NY, USA, 2001. ACM. ISBN 1-58113-378-2. doi: http://doi.acm.org/10.1145/381591.381600.

[16] Douglas M. Blough, Mauro Leoncini, Giovanni Resta, and Paolo Santi. The lit K-neigh protocol for symmetric topology control in ad hoc networks. In *MobiHoc*, pages 141–152. ACM, 2003. ISBN 1-58113-684-6. URL http://doi.acm.org/10.1145/778415.778433.

[17] Jeremy Blum, Min Ding, Andrew Thaeler, and Xiuzhen Cheng. Connected dominating set in sensor networks and manets. *Handbook of combinatorial optimization*, page 329, 2005.

[18] Jeremy Blum, Min Ding, Andrew Thaeler, and Xiuzhen Cheng. Connected dominating set in sensor networks and manets. *Handbook of combinatorial optimization*, page 329, 2005.

[19] Jeremy Blum, Min Ding, Andrew Thaeler, and Xiuzhen Cheng. Connected dominating set in sensor networks and manets. *Handbook of combinatorial optimization*, page 329, 2005.

[20] A. B. Bondi. Characteristics of scalability and their impact on performance. *Proceedings of the 2nd International Workshop on Software and Performance*, pages 195–203, 2000.

[21] B. Bresar, M. A. Henning, and D. F. Rall. Rainbow domination in graphs. *Taiwanese J. of Math.*, 12(213-225), 2008.

[22] Breu and Kirkpatrick. Unit disk graph recognition is NP-hard. *CGTA: Computational Geometry: Theory and Applications*, 9, 1998.

[23] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In

MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking, pages 85–97, New York, NY, USA, 1998. ACM. ISBN 1-58113-035-X. doi: http://doi.acm.org/10.1145/288235.288256.

[24] Matthias R. Brust, Adrian Andronache, and Steffen Rothkugel. WACA: A hierarchical weighted clustering algorithm optimized for mobile hybrid networks. *CoRR*, abs/0706.1080, 2007. URL http://arxiv.org/abs/0706.1080. informal publication.

[25] Sergiy Butenko, Xiuzhen Cheng, Ding zhu Du, and Panos M. Pardalos. On the construction of virtual backbone for ad hoc wireless network, January 17 2003. URL http://citeseer.ist.psu.edu/507779.html.

[26] Arnaud Casteigts and Serge Chaumette. Dynamicity aware graph relabeling systems (DA-GRS), A local computation based model to describe manet algorithms. In S. Q. Zheng, editor, *International Conference on Parallel and Distributed Computing Systems (PDCS'05)*, pages 231–236, Phoenix, AZ, USA, November 2005. IASTED/ACTA Press.

[27] Mainak Chatterjee, Sajal K. Das, and Damla Turgut. WCA: A weighted clustering algorithm for mobile ad hoc networks. *Cluster Computing*, 5(2):193–204, 2002.

[28] D. Cheriton and R. E. Tarjan. Finding minimum spanning trees. *SIAM J. Comput.*, 5: 724–742, 1976.

[29] Cidon and Mokryn. Propagation and leader election in a multihop broadcast environment. In *DISC: International Symposium on Distributed Computing*. LNCS, 1998.

[30] Decheng Dai and Changyuan Yu. A $5+\epsilon$-approximation algorithm for minimum weighted dominating set in unit disk graph. *Theoretical Computer Science*, 410:756–765, 2009.

[31] Fei Dai and Jie Wu. An extended localized algorithm for connected dominating set formation in ad hoc wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, PDS-15(10):908–920, October 2004.

[32] Fei Dai and Jie Wu. On constructing k-connected k-dominating set in wireless ad hoc and sensor networks. *Journal of parallel and distributed computing*, 66(7):947–958, 2006.

[33] Leon Danon, Jordi Duch, Albert Diaz-Guilera, and Alex Arenas. Comparing community structure identification. page 09008, October 18 2005. ISSN 1742-5468. URL http://arxiv.org/abs/cond-mat/0505245. Comment: 10 pages, 3 figures, 1 table. v2: condensed, updated version as appears in JSTAT.

[34] Antoine Dutot, Frédéric Guinand, Damien Olivier, and Yoann Pigné. Graphstream: A tool for bridging the gap between complex systems and dynamic graphs. In *EPNACS: Emergent Properties in Natural and Artificial Complex Systems*, 2007.

[35] A. Ephremides, J. E. Wieselthier, and D. J. Baker. A design concept for reliable mobile radio networks with frequency hopping signaling. *Proceedings of IEEE*, 75(1):56–73, 1987.

[36] P. Erdős and A. Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.

[37] Henning Fernau, Joachim Kneis, Dieter Kratsch, Alexander Langer, Mathieu Liedloff, Daniel Raible, and Peter Rossmanith. An exact algorithm for the maximum leaf spanning tree problem. In Jianer Chen and Fedor V. Fomin, editors, *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, volume 5917 of *Lecture Notes in Computer Science*, pages 161–172. Springer, 2009. ISBN 978-3-642-11268-3. URL http://dx.doi.org/10.1007/978-3-642-11269-0.

[38] F. V. Fomin, D. Kratsch, and G. J. Woedinger. Exact (exponential) algorithms for the dominating set problem. *Proceedings of WG*, 3353:245–256, 2004.

[39] Tetsuya Fujie. An exact algorithm for the maximum leaf spanning tree problem. *Computers & OR*, 30(13):1931–1944, 2003. URL http://dx.doi.org/10.1016/S0305-0548(02)00117-X.

[40] Michael R. Garey and David S. Johnson. *Computers and Intractability.* W.H. Freeman and Co., New York, 1979. ISBN 0-7167-1045-5 (pbk.).

[41] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman Co, 1990.

[42] S. Gaspers, D. Kratsch, and Mathieu Liedloff. Exponential time algorithms for the minimum dominating set problem. *Proceedings of SWAT*, 4059:148–159, 2006.

[43] Mario Gerla, Christoph Lindemann, and Ant Rowstron. P2p manet's - new research issues. *Perspectives Workshop: Peer- to-Peer Mobile Ad Hoc Networks - New Research Issues, number 05152 in Dagstuhl Seminar Proceedings*, 2005.

[44] Guha and Khuller. Approximation algorithms for connected dominating sets. In *ESA: Annual European Symposium on Algorithms*, 1996.

[45] Sudipto Guha and Samir Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.

[46] Indranil Gupta, Denis Riordan, and Srinivas Sampalli. Cluster-head election using fuzzy logic for wireless sensor networks. In *CNSR*, pages 255–260. IEEE Computer Society, 2005. ISBN 0-7695-2333-1. URL http://doi.ieeecomputersociety.org/10.1109/CNSR.2005.27.

[47] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Trans. Info. Theory*, 46(2):388–404, 2000.

[48] Shay Halperin and Uri Zwick. Optimal randomized erew pram algorithms for finding spanning forests. In *J. Algorithms*, pages 438–447, 2000.

[49] F. Harary and T. W. Haynes. Double domination in graphs. *Ars Combinatoria*, 50 (201-213), 2000.

[50] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. *Fundamentals of Domination in Graphs*, volume 208 of *Monographs and Textbooks in Pure and Applied Mathematics*. Marcel Dekker, 1998.

[51] S. T. Hedetniemi and R. C. Laskar. Bibliography on domination in graphs and some basic definitions of domination parameters. *Discrete Mathematics*, 86(1-3):257–277, 1990.

[52] Xiaobing Hou, David Tipper, and Shuju Wu. A gossip-based energy conservation protocol for wireless ad hoc and sensor networks. 2006. ISSN 1573-7705. URL http://dx.doi.org/10.1007/s10922-006-9037-6.

[53] H.T.Friis. A note on a simple transmission formula. *Proc. IRE*, 34:254–256, 05 1946.

[54] M.L. Huson and A. Sen. Broadcast scheduling algorithms for radio networks. In IEEE MILCOM '95, Conference Record, editor, *Military Communications Conference*, volume 2, pages 647 – 651, 1995.

[55] Amit P. Jardosh, Elizabeth M. Belding-royer, Kevin C. Almeroth, and Subhash Suri. Real-world environment models for mobile network evaluation. *IEEE Journal on Selected Areas in Communications*, 23:622–632, 2005.

[56] Soheir M. Khamis, Semeh S. Daoud, and Hanaa A. E. Essa. A randomized algorithm for determining dominating set in graphs of maximum degree five. *Theoretical Computer Science*, 410:5122–5127, 2009.

[57] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.

[58] Hailan Li. Routing in mobile ad-hoc wireless networks, January 01 1999. URL http://digitalcommons.fau.edu/dissertations/AAI1395579.

[59] C. S. Liao and G. J. Change. k-tuple domination in graphs. *Information Processing Letters*, pages 45–50, 2003.

[60] Mathieu Liedloff. Finding a dominating set on bipartite graphs. *Information Processing Letters*, 107:154–157, 2008.

[61] JH Lin, CR Dow, and SF Hwang. A distributed virtual backbone development scheme for ad-hoc wireless networks. *Wireless Personal Communications*, 27(3):215–233, 2003.

[62] Haiyun Luo, Fan Ye, Jerry Cheng, Songwu Lu, and Lixia Zhang. TTDD: Two-tier data dissemination in large-scale wireless sensor networks. *Wireless Networks*, 11(1-2): 161–175, 2005. URL http://dx.doi.org/10.1007/s11276-004-4753-x.

[63] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematics, Statistics and Probability, Vol. 1*, pages 281–296, 1967.

[64] Nesetril, Milkova, and Nesetrilova. Otakar boruvka on minimum spanning tree problem: Translation of both the 1926 papers, comments, history. *DMATH: Discrete Mathematics*, 233, 2001.

[65] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks, August 11 2003. URL http://arxiv.org/abs/cond-mat/0308217. Comment: 16 pages, 13 figures.

[66] Yoann Pigné. *Modélisation et traitement décentralisé des graphes dynamiques – Application aux réseaux mobiles ad hoc*. PhD thesis, Université du Havres, 2008.

[67] A. Piyatumrong, Pascal Bouvry, Frédéric Guinand, and K. Lavangnananda. Trusted spanning tree for delay tolerant MANETs. In Cheng-Zhong Xu and Minyi Guo, editors, *2008 IEEE/IPIP International Conference on Embedded and Ubiquitous Computing (EUC 2008), Shanghai, China, December 17-20, 2008, Volume II: Workshops*, pages 293–299. IEEE Computer Society, 2008. ISBN 978-0-7695-3492-3. URL http://dx.doi.org/10.1109/EUC.2008.45.

[68] Apivadee Piyatumrong, Patricia Ruiz, Pascal Bouvry, Frédéric Guinand, and Kittichai Lavagnananda. Token traversal strategies of a distributed spanning forest algorithm in delay tolerant MANETs, 2009. URL HAL:http://hal.archives-ouvertes.fr/hal-00455459/en/.

[69] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[70] Amir Qayyum, Laurent Viennot, and Anis Laouiti. Multipoint relaying for flooding broadcast messages in mobile wireless networks. In *HICSS*, page 298, 2002.

[71] T. S. Rappaport. *Wireless communications, principles and practice*. Prentice Hall, 1996.

[72] E. Sampathkumar. The global domination number of a graph. *Journal of Mathematical and Physical Sciences*, 23(5):377–385, 1983.

[73] Paolo Santi. *Topology Control in Wireless Ad Hoc and Sensor Networks*. John Wiley & Sons, NJ, August 2005. $99.95 0-470-09453-2.

[74] Shang, Wan, Yao, and Hu. Algorithms for minimum m-connected k-tuple dominating set problem. *TCS: Theoretical Computer Science*, 381, 2007.

[75] Stojmenovic, Seddigh, and Zunic. Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. *IEEETPDS: IEEE Transactions on Parallel and Distributed Systems*, 13, 2002.

[76] I Stojmenovic and J Wu. Broadcasting and activity-scheduling in ad hoc networks. *Mobile Ad Hoc Networking: Edited by Stefano Basagni...[et Al.]*, page 205, 2004.

[77] Gordon L. Stüber. *Principles of mobile communication (2nd ed.)*. Kluwer Academic Publishers, Norwell, MA, USA, 2001. ISBN 0-7923-7998-5.

[78] Haixia Tan, Weilin Zeng, and Lichun Bao. PATM: Priority-based adaptive topology management for efficient routing in ad hoc networks. In Vaidy S. Sunderam, G. Dick van Albada, Peter M. A. Sloot, and Jack Dongarra, editors, *Computational Science - ICCS 2005, 5th International Conference, Atlanta, GA, USA, May 22-25, 2005, Proceedings, Part II*, volume 3515 of *Lecture Notes in Computer Science*, pages 485–492. Springer, 2005. ISBN 3-540-26043-9. URL http://dx.doi.org/10.1007/11428848_64.

[79] Pham Dinh Tao and Le Thi Hoai An. Convex analysis approach to d.c. programming: Theory, algorithms and applications. *Acta Mathematica Vietnamica*, 22(1):289–355, 1997.

[80] Pham Dinh Tao and Le Thi Hoai An. Dc optimization algorithms for solving the trust region subproblem. *SIAM Journal of Optimization*, 8:476–505, 1998.

[81] My T Thai, N Zhang, R Tiwari, and X Xu. On approximation algorithms of k-connected m-dominating sets in disk graphs. *Theoretical Computer Science*, 385(1-3):49–59, 2007.

[82] Jing Tian, Joerg Haehner, Christian Becker, Illya Stepanov, and Kurt Rothermel. Graph-based mobility model for mobile ad hoc network simulation. In *SS '02: Proceedings of the 35th Annual Simulation Symposium*, page 337, Washington, DC, USA, 2002. IEEE Computer Society.

[83] J. M. M. van Rooij and H. L. Bodlaender. Design by measure and conquer: A faster exact algorithm for dominating set. *Proceedings of STACS*, pages 657–668, 2008.

[84] F Wang, My T Thai, and DZ Du. On the construction of 2-connected virtual backbone in wireless networks. *IEEE Transactions on wireless Communications*, 2007.

[85] J. Wu and H. Li. On calculating connected dominating sets for efficient routing in ad hoc wireless networks. *Proc. Third ACM Int'l Workshop Discrete Algorithms and Methods for Mobile Computing and Comm.*, Dial M 1999:7–17, 8 1999.

[86] Jie Wu. An enhanced approach to determine a small forward node set based on multipoint relays. *IEEE VEHICULAR TECHNOLOGY CONFERENCE*, 4:2774–2777, 2003.

[87] Jie Wu and Fei Dai. A generic distributed broadcast scheme in ad hoc wireless networks. *IEEE Trans. Computers*, 53(10):1343–1354, 2004. URL http://doi.ieeecomputersociety.org/10.1109/TC.2004.69.

[88] Jie Wu, Michaela Cardei, Fei Dai, and Shuhui Yang. Extended dominating set in ah hoc networks using cooperative communication. *NETWORKING*, (LNCS 3462):1393–1396, 2005.

[89] Yiwei Wu and Yingshu Li. Construction algorithms for k-connected m-dominating sets in wireless sensor networks. *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 83–90, 2008.

[90] Yiwei Wu, Feng Wang, My T Thai, and Yingshu Li. Constructing k-connected m-dominating sets in wireless sensor networks. *Military Communications Conference, 2007. MILCOM 2007. IEEE DOI - 10.1109/MILCOM.2007.4454774*, pages 1–7, 2007.

Publications

[ICWN2009] J. Schleich, P. Bouvry and H.A. Le Thi, Decentralized Fault-tolerant Connected Dominating Set Algorithm for Mobile Ad hoc Networks, International Conference on Wireless Networks, 2009

[MobiWAC2009] J. Schleich, G. Danoy, P. Bouvry and H.A. Le Thi, Blackbone2, an Efficient Deterministic Algorithm for creating 2-Connected m-dominating Set-based Backbones in Ad Hoc Networks, Proc. 7th ACM International Symposium on Mobility Management and Wireless Access, 2009

[MobiWAC:2010] J. Schleich, G. Danoy, P. Bouvry and H.A. Le Thi, On quantifying the quality of CDS-based virtual backbones in mobile ad hoc networks, Proc. 8th ACM International Symposium on Mobility Management and Wireless Access, 2010

[JOCO:2010] J. Schleich, H.A. Le Thi and P. Bouvry , Solving the Minimum m-Dominating Set problem by a Continuous Optimization Approach based on DC Programming and DCA, Journal of Combinatorial Optimization, 2010 (submitted)

Index

A

Ad hoc network . . . 9

Approximation algorithm . . . 47, 51, 74

B

Blackbone . . . 88, 117

Blackbone2 . . . 95, 120, 124

Broadcast storm problem . . . 25

C

CDS-based virtual backbone (CDSbVB) . 35, 40

Centralized algorithm . . . 47, 51, 76, 108

Cluster-based virtual backbone (CbVB) . . 34, 39

Communication graph . . . 16, 17

Connected Dominating Set . . . 36

Cut vertex . . . 96

D

DC algorithm (DCA) . . . 77, 83

Deterministic algorithm . . . 49

Difference of convex function (DC) . . . 77, 82

Distributed algorithm . . . 47, 53, 86, 115

Dominating Set . . . 36, 70

Dominating set . . . 80

Dynamic graph . . . 17, 124

E

Exact algorithm . . . 46, 51, 74

G

Greedy algorithm . . . 61, 62

I

Independent set . . . 56

K

k-vertex connectivity . . . 72, 96

L

l-level domination . . . 72, 81

Localized algorithm . . . 48

M

m,l-Dominating Set (m,l-DS) . . . 72

m-vertex domination . . . 71, 80, 98

Maximum Independent Set (MIS) . . . 56, 61

Mobility model . . . 19

Multi-Point Relay algorithm (MPR) . . . 57

N

Network . . . 16

Network Simulator (OMNeT++) . . . 144

Q

Quality criteria . . . 37

R

Range assignment . . . 16

Reliability . . . 38

Robustness . . . 38, 49, 58, 88

S

Self-pruning algorithm . . . 54

Spanning tree . . . 32

Stochastic algorithm . . . 49, 60

T

Topology control (TC) . . . 25

Tree-based virtual backbone (TbVB) . 31, 39

U

Unit disk graph (UDG) . . . 17

V

Virtual backbone (VB) . . . 26, 30

W

Wireless channel . . . 15