Doctor of Computer Science
Pontifical Catholic University of Paraná State

Specialization

Computer Networking and Telecomunications

presented by

Tânia Lúcia Monteiro

Submitted for the degree of

Doctor of Computer Science of Pontifical Catholic University of Paraná State

Distributed Channel Allocation Algorithms for Access Point Based Return Channel in IDTV

Commitee:

This work was partially supported by CAPES and CNPq (Brazil), under Grants RH-TVD: AUX-PE-RH-TVD 249/2008, 473222/2009-4 and 304761/2009-5.

To Daniel...

Acknowledgements

I would like to start by expressing my warm acknowledgements to my thesis co-director, Prof. Manoel C. Penna, for believing from the beginning that I would be able to realize this project. His research supervision, his support, his hard way to conduct me and his attention to push me back on track when I was loosing my focus, was essential over the last five years. Next I extend my acknowledgements to my thesis director, Prof. Marcelo E. Pellenz, for the ability to conduct our research in a very careful and efficient away, showing me up all the time an example to be followed. The care you both took with my guidance inspire me to be a researcher very committed to the quality of my works. Thank you!

My acknowledgements also go to the entire team of RH-TVD project, from UTFPR, named by the coordinators, Profa. Keiko V. Onno Fonseca, Prof. Richard Demo Souza and Prof. Alexandre Paul, with whom I have had enriching interactions.

Special thanks to Prof. Guy Pujolle that has kindly received me in his special researchers team from LIP6 at UPMC in France, giving me all support to develop my research and moments of interesting discussions on the possibilities of my work.

I thanks very much the collaboration of Prof. Fabrício Enembreck. He provided invaluable help with the understanding of the ADOPT an DPOP algorithms.

Let me specially thanks to my friend and co-supervised student Juliana M. A. Mendes who worked out the details of the implementation of the DOCA and DSCA algorithms.

A special thanks to my colleagues from UPMC, for good feedback on research issues and presentational skill, or simply for providing good atmosphere during our meetings: Sandra Eliza, To Lam, Diana Sakae, and Patrícia Giron.

Following, I thank the Pontifical University Catholic of Paraná, and CAPES for the scholarship. It was fundamental so that I could devote myself fully to my research.

Thanks to my lovely parents, the other members of my close family and to my friends, who were very supportive during these past few years, whom I did not get to see so often. Last but certainly not least, I would like to sincerely thank my husband Daniel for his love, his patience, and for tolerating my too often work-focused mindset, being by my side in every moments. Without you I would not be writing this page.

Finally, I would like to apologize to anyone I might have unforgivably omitted.

Tânia L. Monteiro

Abstract

An important social aspect of the Interactive Digital TV (IDTV) is to provide access to telecommunications networks through the use of the return channel, promoting digital inclusion. IEEE802.11 standards still are competitive technologies for the implementation of the return channel in suburban areas, because their low cost and ease of implementation. Considering an access point (AP) based architecture for the return channel, the channel allocation problem is investigated, in order to minimize interference and improve the network performance. The channel allocation problem is modeled as a DCOP (Distributed Constraint Optimization Problem).

This thesis proposes and investigates four distributed algorithms for channel allocation. The first proposed algorithm, named DCAA-O, is synchronous and parallel which finds the optimal solution to the channel assignment problem for a group of APs. The second algorithm, called DCAA-S, is derived from DCAA-O. It achieves the suboptimal solution with the advantage of significantly reducing the number of exchanged control messages. The third and fourth algorithms, called DOCA and DSCA, reduce the number of messages and also the total amount of control information exchanged between APs. The DOCA algorithm achieves a globally optimal solution and requires a linear number of control messages. The DSCA algorithm, by reducing the size of the exchanged messages, presents a suboptimal solution and is more scalable than DOCA. The algorithms were compared to a recently proposed distributed protocol for channel allocation in WLANs, denoted Local-Coord (LO-A). All algorithms outperform LO-A in terms of solution quality in most scenarios.

Key Words

Wireless Networks, Distributed Optimization, Channel Allocation, DCOP, ADOPT, DPOP.

Table of Contents

Chapter 1

Introduction

Contents

1.1 Motivation

1.1.1 Return Channel for Interactive Digital TV

Emerging as a new scenario of communication, the digital TV presents itself as a new way of communication. Besides providing numerous advantages over analogical systems, such as, better audio and video quality, more programs into a single channel, and receiving signals on mobile devices, it offers to the viewers the opportunity to interact with the TV station. TV users move from an essentially passive activity to a participatory one.

In Interactive Digital TV the action of interactivity represents the ability of a device to interact or allow interaction with its respective user. It should be noted that the existence of interaction is strictly related to the existence of a return channel mediating the interaction [1]. The concept of interactivity can be classified as: *full* where the viewer can control the content of the program one is watching; and *coactive* where features are added allowing viewers to obtain information about sports, weather, news, and also access information about advertised products and even purchase them. Montez and Becker [2] define interactivity in a more fully way, considering the possibility of the viewers interfere in the programming, and not only react to programs transmitted by broadcasters. The concept of interactivity requires the existence of a communication channel capable of providing resources for transmitting content from the viewer to the TV station.

The amount and diversity of users wishing to access interactive services have been expanding, creating a demand for technologies that address their requirements. In this sense, given the limitations of the Brazilian population, in terms of the access to telecommunications networks and digital inclusion, one of the alternatives that is being consolidated is the use of the TV receiver as an efficient and simple terminal to access these services, regardless of the telecommunication network they are connected. Then the Interactive Digital Television is presented as a low cost alternative to this reality, presenting itself as a technological solution that can meet the range of the digitally excluded population, providing Internet access, distance learning, e-government, entertainment, health services, etc. The device capable of receiving signals transmitted by the TV station may be embedded in a digital television or may be an external device. It is called set top box. In the case of an interactive service, the receiver should send data from the viewer to the TV station or to any other destination, using a communication channel, named return or feedback channel.

Technology on which the communication channel (return channel) is implemented defines its transmission characteristics. The solution is not unique and it will depend on resource availability and characteristics of each region, city, operator and user. As well as local population density that may favor or not the most diverse solutions. Among the main possible solutions, we have: PLC, mobile telephony, xDSL, Wimax and Wi-Fi [1].

- PLC (*Power Line Communication*) – This technology allows the use of the electric power network for the digital signals transmission, providing that any energy point may be a network point. It is an attractive solution due to its capillarity and transmission rates. However according to [3] the studies for large-scale deployments have not been conclusive;

- Mobile Telephony – The data transmission over cellular networks is evolving. Today, the 3G systems reach transmission rates about 700kbits/s, and 4G systems, due Mobile Wimax (*Worldwide Interoperability for Microwave Access*) and LTE (*Long-term-evolution*) technologies [4], reach 100Mbits/s for high mobility communication (such as from trains and cars), and 1Gbits/s for low mobility communication (such as pedestrians and stationary users). However, the cost of these services is still

[1]Term licensed by the Wi-Fi Alliance to refer to wireless networks based on IEEE 802.11 standard

prohibitive for most of the population;

- xDSL (*Digital Subscriber Line*) – This technology is widely used as a broadband solution in small businesses and residential environments, enabling high-speed delivery of data, audio, and video in a digital form over the existing telephone infrastructure. There are several types of DSL technology, such as ADSL, SDSL, IDSL, RADSL and VDSL [5, 6]. Various DSL technologies differ from each other in data rate, operating distance, ratio between downstream and upstream speeds, as well as concerning their applications. The DSL technology is considered disadvantageous in relation to the final cost, since access to the technology is associated with the availability of a telephone line and a service provider;

- WiMax – This standard, IEEE 802.16, was designed to provide wireless last-mile broadband access in the Metropolitan Area Network (MAN), delivering performance comparable to traditional cable, DSL, or T1 offerings. Originally covered radio frequencies between 10 GHz to 66 GHz. This frequency band has been enlarged to also include frequencies bellow 10 GHz. Recently, it has been proposed the use of WiMAX for the return channel in digital television systems, but using a new frequency profile around 700 MHz, called WiMAX-700. Meloni et al. [7] provide elements to determine the suitability of WiMAX-700 technology as the return channel for the majority of interactive digital television applications. It also presents some conditions that must be observed for this intended application. Some of them imply a higher investment on the network infrastructure;

- Wi-Fi Networks – Wireless local networks based on IEEE 802.11 standard [8]. These networks operate in unlicensed bands and present a theoretical capacity ranging from 11Mbps to 600Mbps, depending on the chosen standard (802.11a, 802.11b, 802.11g and 802.11n) [9–12]. One important characteristic of these networks is that they provide a flexible technology that enables the deployment in various locations at reduced costs.

1.1.2 Relationship Between Access Technologies and Brazilian Government Goals for the Interactive Digital TV

The Brazilian government is aware that the open television system is one of the most important sources of information dissemination, knowledge, entertainment and responsibility in relation to national culture and citizenship [13]. Due to these aspects, the adoption of the Interactive Digital TV model must necessarily pass through a strategy that is focused on the Brazilian citizen low-income and socially excluded.

As said in the previous subsection, the return channel can be deployed over several technologies (such as PLC, mobile telephony, DSL, Wimax and Wi-Fi). Each one presents different technical features enabling the return channel implementation. However the portion of the population that suffers from the digital exclusion is precisely the one that has less economic resources and in large urban centers is located mostly in areas of difficult access. Usually they have difficulty in accessing telecommunications networks as phone networks and are not able to afford the costs of subscriptions. Due to this reality it is necessary a solution that meets technical and economic requirements, allowing access for all social strata of the population.

Consequently, among the evaluated alternatives, the wireless networks, based on the 802.11b/g standards [8], present themselves as viable strategy for the return channel. The access network solution can be based in public access points or in access points owned by TV operators. The APs are connected to the broadcasters, and the users can establish a connection to these APs through a wireless network interface embedded on the set top box or on the TV set.

1.1.3 Using Unlicensed Band as a Solution to Return Channel

The wireless 802.11b/g standards [10,11] are increasingly popular, thanks to their low cost, capacity, flexibility, coverage, easy implementation and simplified operation. A variety of products with wireless adapters (computers, telephones, sensors, TV, tablets, etc.) has responded to different classes and consumer profiles, and are used in residential, commercial, and public access points networks.

In recent years, these networks have been widely investigated for application in two main scenarios: the wireless LANs - WLANs (*Wireless Local Area Networks*) and the *mesh* networks - WMN (*Wireless Mesh Networks*). They operate in unlicensed portions of the frequency spectrum (2.4 and 5 GHz), named ISM (*Industrial, Scientific and Medical*) band. The standards (802.11b/g), provide 11 frequency channels but only three channels (channels 1, 6 and 11) can be used simultaneously without causing interference (*non-overlapping channels*) [14, 15]. The ability to re-use these channels, benefiting from the whole available spectrum, substantially increases the effective bandwidth and capacity of the wireless network [16].

Considering this standards to provide network access to return channel solution, there are performance-related challenges that must be solved. The major problem is the capacity reduction due to co-channel interference (neighbors networks configured on the same channel) and adjacent channel interference (neighbors networks configured on overlapping channels), that occur between nodes of the network or nodes of neighboring networks. It is known that the level of interference depends mainly on the channels allocation, the geographical distribution of APs (*Access Points*) and users, the number of users associated with each AP and also the traffic load.

1.2 Objectives

The main objective of this thesis is to present the technology based on the 802.11b/g standards [8] as a network solution for deploying a return channel for digital TV system. This work focuses on ensuring the performance of scenarios where the APs may belong to different domains, a very common situation in deploying wireless solutions in very dense areas. We identified that the existing algorithms for the channel assignment problem applied in such scenarios can not guarantee an optimized solution. Additionally, the existing cooperative distributed algorithms require information exchange using a wired network, which is only feasible for networks owned by the same network administrator. It is assumed a dynamic strategy. In this case, a coordination mechanism is necessary to assure adequate channel assignment between communicating APs. The main challenge is the design of the switching algorithm.

Inspired by the search of a solution to ensure an adequate throughput to the 802.11b/g standards networks, we analytically formulate the channel assignment as a DCOP (*Distributed Constraint Optimization Problem*) and propose new distributed interference-aware channel assignment algorithms based on collaborative agents. The adopted strategies are derived from an asynchronous algorithm for coordination of collaborative agents called ADOPT (*Asynchronous Distributed Optimization*) and from a method for distributed constraint optimization based on dynamic programming, named DPOP (*Distributed Pseudo-tree Optimization Procedure*).

1.3 Contributions

This thesis presents two main contributions that to the best of our knowledge, have not been explicitly addressed in prior literature in the context of wireless 802.11b/g standards. First, a novel formulation for the channel assignment problem as a DCOP. Second the proposal of new cooperative and distributed algorithms, which are able to reach the optimal or suboptimal frequency assignment. Two of the proposed algorithms are derived from ADOPT and are named DCAA-O (*Distributed Channel Assignment Algorithm - Optimal*) and DCAA-S (*Distributed Channel Assignment Algorithm - Suboptimal*). The algorithm DCAA-O reaches the optimal channel assignment but presents a drawback in relation to the APs communication. The algorithm DCAA-S (*Distributed Channel Assignment Algorithm - Suboptimal*) solves the problem and is able to achieve a near optimal solution. Next, we propose two algorithms, based on algorithm DPOP. The algorithms DOCA (*Distributed Optimal Channel Assignment*) and DSCA (*Distributed SubOptimal Channel Assignment*) were developed respectively to reduce the total amount of interactions between APs and the size of the exchanged control messages. The algorithm DOCA achieves the optimal solution, requiring fewer interactions between APs. The algorithm DSCA through a heuristic, controls the size of interactions between APs, increasing the scalability presented by DOCA algorithm.

As benchmark algorithms were considered the algorithm DPOP that presents optimal global solution, LO-A [17] because its is a simple scalable distributed algorithm and the algorithm Hsum [14] because its realistic scenarios of interference and channel reuse. Additionally it was included a random channel assignment method, identified as *Random* in the results, which simulates deployments where there is no administration and no local or cooperative mechanisms to control the channel allocation between the APs.

1.4 Outline of this Thesis

The remainder of this thesis is organized as follows:

In Chapter 2, we recall some background knowledge that is necessary to understand our formulation. It is presented the most important features, common deployments and standards of WLAN and WMNs architectures. The mainly characteristics of unlicensed networks are explored, as well as a briefly review of some concepts of pseudo-tree generation and leader election;

Chapter 3 introduces DCOP formulation and describes the algorithms ADOPT and DPOP. These algorithms are distributed algorithms, based in AI, (*Artificial Intelligence*) that use the concept of multiagents. It is presented the used strategy to formulate the analytical interference model and finally describe some important details of the chosen benchmark algorithms;

Chapter 4 is dedicated to model the novel formulation for the wireless channel assignment as a DCOP. The proposed algorithms (DCAA-O, DCAA-S, DOCA and DSCA) are described, as well as the Ddfs-improved algorithm, a new proposal to generate a pseudo-tree. The simulation results and performance analysis also are presented.

Conclusions are drawn in Chapter 5 where we also point out the directions that could be natural extensions of this work.

Chapter 2

Background

Contents

2.1 Introduction

This chapter recalls some background knowledge to help the understanding of our contribution. It presents the most important features of WLANs and WMNs architectures as well as the their most common deployments and standards.

2.2 Wireless Networks Architectures

2.2.1 Wireless Local Area Network (WLAN)

Initially the WLANs provided connections to areas where local wiring systems were not available for the conventional LAN services. Later the solution began to support mobile devices throughout a building or a campus. This is mainly due to the number of available Wi-fi services, and associated consumer products, such as tablets, games devices, wireless IP cameras, etc [18]. Then the WLANs began to be widely used in homes, schools, airports, offices or anywhere that needed a wireless connection [19]. The solution used before in the indoor environments has extended to outdoor environments like public outdoor locations. Nowadays it is possible to describe the WLANs as an wireless network architecture that consist of mobile devices with network adapters (NAs) and access points (APs), which normally are connected to a high speed wired LAN. Most of them use the IEEE 802.11 standards and operate in the unlicensed bands, intended to Industrial, Scientific, and Medical applications (ISM). The WLANs provide wireless connectivity over a certain geographical area, and are formed by one or more APs where each AP has a maximum range of coverage.

2.2.2 Main Features of the ISM Band Networks

As mentioned before the most common deployments of the WLANs are based on the IEEE 802.11 standards that coordinate the operation of the devices in the unlicensed bands (ISM): 902 to 928GHz, 2.4 to 2.5GHz, and 5.725 to 5.875GHz. The IEEE 802.11b and 802.11g standards provide, respectively, WLAN services up 11Mbps and 54Mbps in 2.4GHz. These standards use the modified version of IEEE 802.11 direct sequence spread spectrum. The standard IEEE 802.11a [9], a new version of high rate WLAN standard, also use unlicensed frequency bands and modulation schemes that enable transmission rates up to 54Mbps at the 5.7GHz band. It is further applied to the corporate workspaces. Finally the standard IEEE 802.11n [12], an amendment to the previous IEEE 802.11 standards, that improve network throughput over the two standards 802.11a and 802.11g, with a significant increase in the maximum network data rate from 54Mbits/s to 600Mbits/s, adding multiple-input multiple-output (MIMO), 40MHz channels to the PHY (physical layer), and frame aggregation to the MAC layer.

The unlicensed bands are not unique to 802.11 networks. There may be environments where these bands are shared with other devices such as Bluetooth networks, radios, cordless phones, microwave ovens, etc. Thus the 802.11 b/g standards eventually suffer interference coming from their local environment. The coexistence of different device types in nearby channels can seriously degrade the performance of the WLANs, although the techniques DSSS[1] and FHSS[2] for digital transmission are defined. Due to the natural diffusion of wireless media, the success of the transmission is strongly influenced by the amount of multiple access interference.

The regulatory laws of most countries are controlled by a regulatory body, ITU (*International Telecommunication Union*). In its International Radio Regulations, this authority divides the world into three ITU regions for the purposes of managing the global radio spectrum. Each region has its own set of frequency allocations. Brazil is included in region 2 that covers the Americas, Greenland and some of the eastern Pacific Islands. Regarding region 2 it is established for the IEEE 802.11b/g standards, at least 11 different channels whose carriers are allocated between 2400 and 2483MHz, with channel central frequencies starting at 2412MHz and spaced at 5MHz intervals up to 2462MHz. Each channel has a bandwidth of 22MHz. This assignment allows up to three non-overlapping channels[3] (1, 6 e 11), since they are spaced 25MHz from each other.

The networks of the IEEE 802.11a standard, frequency of 5 GHz, provides 12 different operation channels that do not have spectral overlapping and can be used without the presence of interference between adjacent channels. Although this extension provides a greater amount of non-overlapping channels, its high carrier frequency brings a slight disadvantage compared to our goal: The effective overall range of 802.11a is slightly less than that of 802.11b/g.

If we consider the standard 802.11n, there is the option to double the bandwidth per channel to 40MHz. However, when in 2.4GHz, enabling this option it takes up to 82% of the unlicensed band (2400 to 2483MHz, 11 channels), which may prove to be unfeasible for some environments. To achieve maximum output, a pure 802.11n 5GHz network is recommended, but an 802.11n-only network may be impractical for many users because they need to support legacy equipment that still is 802.11b/g. So this work considers the problem of allocating channel only for frequency bands of the extensions 802.11 b/g. These frequency bands can be used freely, although should be respected the legal limits for power transmission.

2.2.3 Channel Allocation Problems in WLANs

The scheme with the distribution of the frequencies for the 11 channels of the IEEE 8002.11b/g is shown in Figure 2.1. It can be seen that adjacent channels have a cer-

[1]DSSS - *Direct Sequence Spread-Spectrum*: a *spread-spectrum* modulation scheme that generates a redundant pattern of bits for each transmitted bit.

[2]FHSS - *Frequency Hopping Spread*: a *spread-spectrum* modulation scheme that uses a narrowband carrier changing the frequency according to standard known by the transmitter and receiver.

[3]What actually happens is that between channels 1, 6 and 11, there is a small amount of overlapping, however other channels suffer a significant overlapping of the adjacent channel when compared with channels 1, 6 and 11.

tain degree of spectral overlapping. Adjacent channels are channels which overlap in the frequency spectrum. The interference of adjacent channel can occur if there is not enough distance between APs and their allocated channels in the frequency spectrum. In Figure 2.2 it is possible to observe that two APs even operating in non-neighboring channels (channel 1 and channel 3), present some degree of interference.

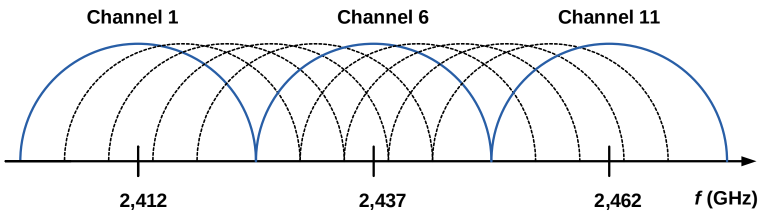


Figure 2.1: Frequency spectrum of the ISM networks - 802.11b/g

Another important factor to be considered is the existence of co-channel interference. Such interference may cause the same effect as the adjacent channel interference, but the circumstances in which it occurs are completely different. A common scenario of a co-channel interference is shown in Figure 2.3. Three APs are operating on the same channel and the signal coverage area (cell) of each one are overlapping causing co-channel interference or collision. This situation defines a collision domain, a serious source of inefficiency in the network [20]. Whenever devices, including access points as well as client stations, can 'see' each other on the same channel they will be forced because of the MAC protocol to wait for others on the same channel to stop transmitting before they can start their transmissions. Some of the most popular MAC protocols have been incorporated in the IEEE 802.11 standard [8].

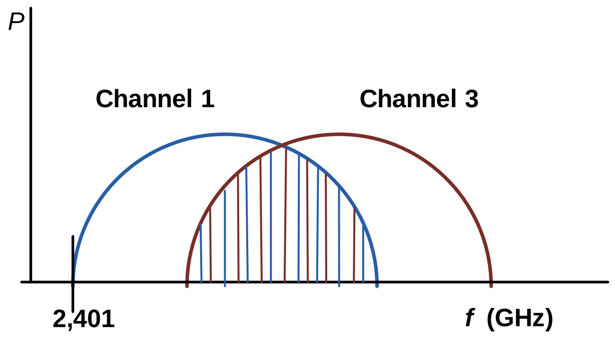


Figure 2.2: Adjacent channel interference between two very close APs.

As a large number of vendors provide APs with an initial default configuration, there may be situations where most of them transmit on the same channel (e.g., channel 6),

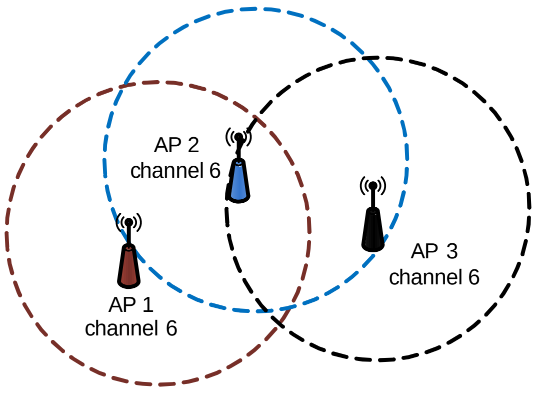


Figure 2.3: Co-channel interference between 3 APs.

and only a reduced fraction of APs use the other two options of non-overlapping channels (e.g., channels 1 and 11). As shown by Jain et al. [21], this is an evidence that many APs overlapped in coverage were not rightly configured to reduce interference, resulting in systems that present low throughput. According to Hua and Zheng [22], there may exist starvation (i.e., very low throughput) in a dense 802.11 wireless network.

The existence of a limited number of non-interfering channels, eventually causes the problem of spectral overlap and severe restrictions for the channel allocation techniques in 802.11 networks. The shortage of available channels limits the number of networks that can coexist in the same region without generating mutual interference.

With the continued growth of the WLANs, the number of APs located in a common area has increased, mainly to extend wireless coverage. This leads to a scenario where channel assignments for APs are made independent of each other, normally subjec to different administrative domains. As this scenario increases, so does the co-channel and adjacent channel interferences. Consequently the assignment of channels to this collection of APs needs to be carefully coordinated, otherwise there may be a loss of performance for the wireless users. In such case the use of a centralized technique is not feasible, and a scalable and distributed solution must be applied. Therefore it is evident that the techniques and mechanisms for channel allocation, should consider, as much as possible, the interferences in the process of channel assignment.

A channel assignment strategy can be classified as static or dynamic. In the static approach, a channel is assigned to the AP permanently or for a long period. However, with the explosive growth in WLAN deployments which operate in the same unlicensed spectrum, any static assignment will likely result in the operation on channels that are also being used by co-located WLANs. The dynamic strategy assumes that the network interface can switch the communication channel by a coordination mechanism, necessary to assure adequate channel matching between communicating nodes. Based on centralized or distributed architectures, cooperative or non-cooperative spectrum allocation, lots of methods have been proposed for dynamic spectrum access, including game theory [23], auction mechanisms [24], local bargaining [25], and graph coloring [26, 27]. A distributed strategy allows a self-partial reorganization of sectors of the network that are suffering further degradation or momentary loss of performance.

2.2.3.1 Previous Work

We start by the technique named LCCS *Least Congested Channel Search* [28]. In this technique, each AP monitors packet transmissions, searching for the most lightly loaded channel. It switches to this channel until the next scan finds a less congested channel. An AP first scans each channel for distinguishable beacons published by neighboring APs. Each beacon contains information of the corresponding APs such as the number of wireless clients. Instead of choosing the channel with the fewest number of associated clients, M. Achanta [28] suggests to use traffic-related information, also obtained from beacons, to select a channel for an AP. In this case, the AP will choose to operate on the channel with the least amount of traffic, irrespective of the number of clients associated with it. However, LCCS fails to capture situations when clients associated to two APs interfere with each other, while the APs do not interfere among themselves. For this reason LCCS cannot take advantage of best opportunities of channel re-use [14]. Finally, it should be noted that the specific fields of a beacon frame, specifying the number of wireless clients associated with each AP and the average amount of traffic, are proprietary to Cisco System Inc. [29].

The study presented by Mishra et al. [14] formulates the channel assignment in WLANs as a weight vertex coloring problem and presents two distributed algorithms that are suitable to a wide range of WLANs technologies. The first algorithm, called by the authors as *Hminmax*, minimizes the maximum impact of interference among all overlap regions between APs, by each AP performing a local optimization and not requiring any collaboration among each other. This technique can be applied to a wireless network formed by APs belonging to different WLANs. The second algorithm, called *Hsum*, minimizes the total impact of all conflict edges, hence minimizing the total effect of interfering APs. This technique is applied to wireless networks under the same administrative domain. According to the authors it can also be applied to situations where APs from different WLANs are willing to coordinate and communicate with each other. In terms of net throughput, *Hsum* outperform *Hminmax*. The weighted coloring [14] *Hminmax/Hsum* in turn outperforms LCCS [28] in terms of throughput.

A client-based model denominated *conflict set coloring* was proposed in [30]. The strategy tries to minimize interference at wireless clients by implicitly modeling its location and distribution with respect to the APs. The algorithm is centralized and is well suited for centrally managed wireless networks.

In the distributed algorithm proposed by Leith and Clifford [31], each AP independently selects a channel whose measured interference power is below a predefined threshold. Their approach does not require direct communication between APs. However, the drawback is that each AP requires a feedback provided by the WLAN about the presence of interference on a chosen channel. The solution is applicable only for the same administrative network domain.

Briggs and Tijmes [32] investigated joint channel and power allocation for public WLANs. Their solution also considers a centralized approach using a modified branch-and-bound algorithm, called *local best-first search.* The cost function is computed using channel spectrum overlapping. Their centralized algorithm is suitable for channel assignment in APs belonging to the same administrative domain.

Chen *et al* [17] proposed two categories of distributed local coordination algorithms for optimizing the channel assignment for APs. The first category employs distributed measurement-based algorithms and local coordination mechanisms. These algorithms were denoted as *No-Coord*, *Local-Coord* (LO-A) and *Global Coord.* The second category employs site specific knowledge and a central network controller. Among them, LO-A is a simple scalable distributed algorithm, featuring a good balance between solution quality and number of exchanged messages between the APs. Each AP locally coordinates with others APs via a wired backbone network for channel switching. The algorithm proved to be an excellent candidate in the context of channel allocation. The LO-A algorithm was shown to outperform those proposed in [14, 30, 31].

In [18] the algorithm termed CACAO (*Client-Assisted Channel Assignment Optimization*) of X. Yue et al., it is proposed a completely distributed channel assignment scheme for uncoordinated WLANs. Their model is based on a single (super) node, representing channel condition information in an AP, gathered by its associated clients. Although their method achieves highest throughput than LCCS [28] and *Hminmax* [14], it performs a penalty around 20% if compared with an optimal based solution.

Considering the convergence to an optimal solution, the proposed algorithms in [14, 17] and [18], although scalable and efficient, sometimes do not converge to a global minimum.

2.2.4 Wireless Mesh Networks (WMNs)

The WMNs represent an important development in wireless LAN technology. They present the potential to strongly increase the area served by a wireless network, by improving the overall spectral efficiency, by selecting multiple hops over high capacity links, rather than single hops over low capacity links.

The WMNs may be considered as a key technology, among various wireless networks, to provide quality services. They are formed of nodes comprised of routers and mesh clients. Each node is not only a host but also a router, allowing packets to be forwarded for other nodes that may not be within direct wireless transmission range of their destinations. The routers form the backbone, normally presenting minimal mobility, making possible network access for mesh and conventional clients. They also provide integration of WMNs with other networks (WLANs, cellular, wireless sensor, Wimax, etc.) because of their gateway and bridge functions. They enable services previously not accessible like a cellular call from a tablet in an outdoor area or the access of private systems from a public wireless network, etc.). Other important characteristics of WMNs is their dynamic ability of self-organization and self-configuration. The nodes in the network automatically establish and maintain mesh connectivity creating an ad-hoc network. These features bring lot of advantages to the WMNs such as robustness, low-cost deployment, simple network maintenance, flexibility and reliable service coverage.

The Wireless Mesh Network solution is quickly emerging as a promising technology and has been used as a more mature technology [33]. The increasing interest is related to its potential for the *last few miles*, and the possibility of significant wireless services in metropolitan area networks. The industry and the research community have given lot of attention to the WMNs due their various qualities and useful inheritance from both the ad-hoc and the traditional wired infrastructure networks. The most common deployments are based on the IEEE 802.11 standard, mainly due its cheap availability. It is known that 802.11 software stack was initially designed for infrastructure WLANs requiring various modification when used in WMNs. The efforts to these modifications are directed towards design of better link layer and channel access protocol. Meanwhile, several IEEE standard groups are actively working to define different specifications for wireless mesh networking techniques. According to [34] the following emerging standards may be identified: IEEE 802.11s, IEEE 802.15.5, IEEE 802.16a, and IEEE 802.20 [4].

The design and deployment of WMNs require a more flexible infrastructure, leading the researchers to considerable challenges like scalability, design of a novel MAC layer, techniques of interference mitigation, heterogeneity amongst standards, etc. What has became clear, since the publication of [35], is that the fundamental problems that affect the performance of WMNs, such as power control, channel assignment, topology control, routing, etc., have only been studied in isolation. As they are closely related to the problem of the wireless interference, in practice joint approaches are likely to provide much better results.

[4]For Draft standards and Public documentations, IEEE 802 Standards Committee Web site *(www.ieee802.org/)*

An important design goal for wireless mesh networks is capacity, which has been addressed by several research studies. Besides the survey presented in [35], where the authors considered the operation and problems layer by layer, some are dedicated to specific problems like improving spatial reuse, energy efficiency, routing and dynamic spectrum access.

It is well known that the major limitation to the performance of WMNs is the wireless interference. Various attempts have been made to model the effects of interference, some using measurement-based models and others using theoretical models. Our approach summarizes some representative in the literature that mitigate the unavoidable consequences of interference by presenting some recently studies that address channel assignment mechanisms to increase the overall spatial reuse. The main idea behind these mechanisms is to try the assignment of non-interfering channels for neighboring wireless transmissions. The actual IEEE 802.11 standard for WLANs, also used for mesh networks, provides several orthogonal channels to facilitate these mechanisms.

2.2.4.1 Previous Work

As an alternative approach to exploiting frequency diversity, a node is equipped with only one radio and will use multiple channels. The studies presented by [36] and [37] includes single-radio multi-channel MAC protocols. In [37] the authors present a protocol called MMAC, that augment the 802.11 MAC protocol, where nodes are synchronized and periodically meet at a common channel to negotiate channels for use in the next phase. In [36] through the protocol AMCP, the authors use a separate frequency channel for channel negotiation and the remaining channels for data packet transmission. This protocol was considered by IEEE 802.11s working group as potential approach to support multi-channel capability in single radio mesh networks [38]. In [39] the authors describe and demonstrate via simulation that these algorithms can yield inefficient allocations. They introduce D1C-CA (*Distance-1 Constrained Channel Assignment*), which assigns channels to a set of links as a function of physical connectivity, contention, the unique gateway node (source or destination of all internet traffic) and edges at distance one. The proposed solution provided bounds of minimum number of needed channels to achieve maximum throughput. The recent survey article on mesh networks [33] summarizes the fundamental problems and design objectives, that affect the performance of WMNs.

2.3 WLANs and WMNs as Network Technologies for Return Channel Implementation

WLAN and WMN architectures are both candidates for deployment of the return channel. The meshed topology, provides good reliability, low upfront investments, market coverage and scalability. However, a user node in a WMN has to transmit relayed traffic as well as its own. Therefore, besides the contention with other nodes for the same destination node (normally a gateway), there is an inevitable contention between its own and the relayed traffic [20]. This type of contention does not occur in WLANs where user nodes are always at one-hop distance from the base station or the access point. The bandwidth problem is further aggravated for multi-hop employments due to interference from adjacent hops on the same path as well as from neighboring paths [21]. For topologies of three or more hops, these problems may quite penalize the network capacity.

According to Sichitiu et al. [20] one of the most contradictory ideas is that the capacity of a WMN exceeds the capacity of a WLAN, based on a similar technology. It is common to find the affirmation that the capacity of the network increases with the number of clients. The intuition to support these claims comes from the possible spatial reuse in WMNs: two nodes at opposite ends of the network can transmit simultaneously without a collision; however, in a multi-hop environment, most of the transmissions are just forwarding of traffic, which effectively eliminates the gain of the spatial reuse.

After considering the characteristics of WLAN and WMN architectures and establish a relationship with the objective of this thesis, we considered WLAN as the more feasible architecture for the implementation of the return channel. The solution is based on TV operators or the service providers installing public outdoor APs in a geographical area, and the users connecting with the APs through a wireless network interface, embedded on the set top box or on the TV set, establishing a return channel.

2.4 Modeling a Wireless Network Topology Through a Graph

Focusing in a WLAN topology we must consider a logical network layout, where the nodes represents the APs and the edges indicate the existence of interference between them. The connection between the APs is defined by their coverage area or broadcast domain. From the definition of the APs and their connections it is possible to obtain an undirected *constraint graph* $\mathbf{G} = (V, E)$, where the set of vertices V represents the APs, and the set of conflict edges E represents the existence of contention between the nodes connected by the edges.

To apply an efficient **Channel Assignment** between APs, we use a distributed multiagent approach [40]. In every situation where multiple agents have to decide on a set of actions to perform, coordination between them is extremely important. Our formulation requires:

- Each AP has a dedicated control interface operating in the same pre-defined channel for messages exchange;

- A logical organization of the network of APs through a constraint graph;
- An organization of the constraint graph through a spanning tree where the root node of the spanning tree is the leader in the negotiation process between APs.

The first step is the construction of a undirect constraint graph **G** to make possible the coordination between the agents. The agents must be prioritized in a total order making use of a *spanning tree*. Then it is necessary to elect a leader (a root node) through a leader election algorithm. Finally a pseudo-tree is constructed, which allows the agents to identify their neighborhood according to a logical ordering, being ready to communicate with each other to obtain a feasible solution. The mapping of the interference problem with multiagents is discussed in the next chapter.

2.5 Pseudo-tree Generation

Following the definitions of Adrian Petcu in [41],

Definition 1 (pseudo-tree) *A pseudo-tree arrangement of a graph G is a rooted tree with same nodes as G and the property that adjacent nodes from the original graph fall in the same branch of the tree.*

This definition allows the pseudo-tree to be a rooted structure with more edges than the original graph **G**. This can be seen in the example of Figure 2.4, graph b. The main idea is to ensure the independence of the nodes, positioned in different branches of the pseudo-tree, when their ancestors are removed. Thus, it is possible to conduct parallel searches on these independent branches, allowing some algorithms to instantiate nodes from the root node (top-down) and other from the leaf nodes (bottom-up), etc. Some authors that have used this technique are [42–47].

2.5.1 Depth-First Search Trees - DFS

A special case of a pseudo-tree, Figure 2.4, graph c, is when all edges of the pseudo-tree belong to the original graph. This special class can be generated by a depth-first search of the graph. Therefore, these are called DFS trees. As defined in [41],

Definition 2 (DFS tree) *Formally the DFS tree ensures an arrangement of a graph* **G** *in a rooted tree with the same nodes and edges as* **G***, with adjacent nodes from the original graph falling in the same branch of the tree.*

The DFS tree has many practical applications in distributing computing networks. It makes possible the identification of the strongly connected devices of a network architecture,

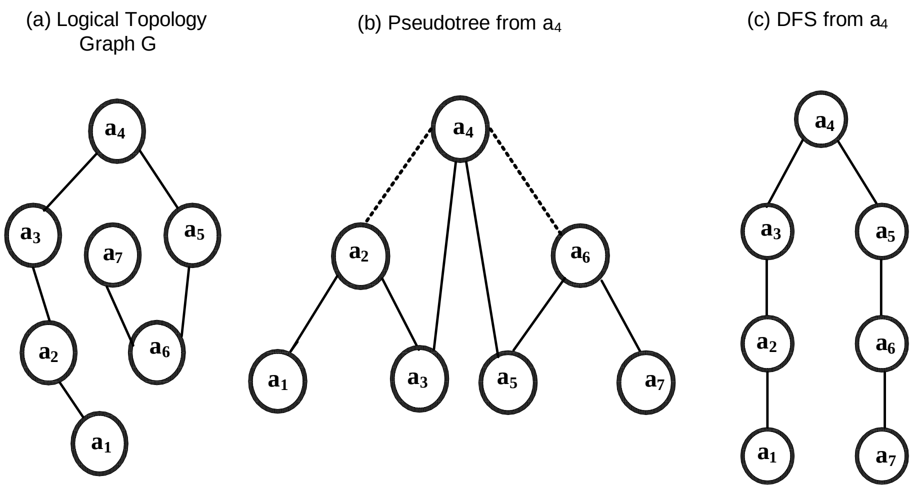


Figure 2.4: A simple constraint graph(a), a possible pseudo-tree (b), and a rooted DFS tree (c). The graph in (c) is also a pseudo-tree, while the graph in (b) is not a DFS tree.

it helps to detect direct cycles, thus avoiding deadlock problems, and also identify the biconnected devices, useful to check whether failure of one device may or may not disconnect a network.

A lot of attention has been dedicated to algorithms that generate a DFS tree in a distributed way. The complexity of distributed DFS computation has a tight relationship with a graph concept called induced width, also know as *treewidth*. Indeed the distributed graph algorithms are time and space exponential in the induced-with along the order of processing [48].

Definition 3 (The induced width) *Given an undirected graph* $\mathbf{G} = (V, E)$*, an ordered graph is a pair* $(\mathbf{G}, d)$*, where* $V = \{v_1, \cdots, v_n\}$ *is the set of nodes, E is a set of arcs over* V *, and* $d = (v_1, \cdots, v_n)$ *is an ordering of the nodes. The nodes adjacent to v that precede it in the ordering are called its parents. The width of a node in an ordered graph is its number of parents. The width of an ordering d, denoted* $w(d)$*, is the maximum width over all nodes.*

Considering Figure 2.4 as an example, the simple constrained graph (a) that is presented over seven nodes, along with two ordering of the graph: d_1 =(graph b) and d_2 =(graph c). By depicting the orderings from bottom to top, so that the first node is at the bottom of the figure and the last node is at the top. The arcs of the graph are depicted by the solid lines. The parents of a_3 along d_1 are $\{a_2, a_4\}$. The width of a_3 along d_1 = is 2, and the width of a_3 along d_2 = is 1. The width of these two orderings are: $w(d_1) = 2$, and $w(d_2) = 1$.

A DFS tree hierarchical strategy makes possible an important approach to solve a distributed constraint optimization problem (DCOP), by dividing the compromise among agents to guarantee the constraints and the computational load. Based on this approach, an agent only has to guarantee the constraints related to its variables and higher variables in the hierarchy. While the responsibility for the guarantee of the other constraints is delegated to its neighboring agents lower in the hierarchy. If compared to variables ordered in a simple way, pseudo-trees provide the advantage of exploiting the topological structure of the problem more efficiently. Parts not too coupled of the DCOP will correspond to two different branches in the pseudo-tree and its computation will be performed in parallel.

In recent years due to the variety of applications, a large number of distributed DFS algorithms have been proposed. The algorithm of Cheung in [49] probes sequentially all the edges of the graph, and therefore requires $2|E|$ messages and time units. Awerbuch [50] improves the time complexity of Cheung's algorithm by using parallel message dispatch but at the cost of increasing the number of exchanged messages. Lakshmanan et al. and Cidon [51, 52] by eliminating some of the parallelism, improve the message complexity, requiring respectively $(4|E| - (|V| - 1))$ and $(3|E|)$ messages and time units. The results of Reif in [53] shows that the DFS problem is inherently sequential in a well-defined sense.

Sharma et al. [54,55] improved the time complexity of the earlier algorithms from $O(|E|)$ to $O(|V|)$ by removing more unnecessary parallelism, eliminating extra messages used by nodes to inform neighbors of their status. The cost was to allow messages to carry more information. The authors ensure that their distributed DFS algorithm uses $2|V|$ messages and $2|V|$ units of time, and that their messages bound improves on the messages complexity of the algorithms of both Lakshmanan [51] and Cidon [52]. The strategy adopted by the authors was to eliminate all real parallelism and put more information in each message.

Makki and Havas [56] presented distributed algorithms for constructing DFS tree in communication networks. According to the authors, their algorithms are more efficient than some previous methods. The algorithms evaluate the communication complexity by the amount of messages sent during the execution, and the time complexity as the maximum time elapsed from the beginning to the end of algorithms execution. It requires $2|V| - 2$ messages and units of time in the worst case when they assume that to deliver a message over a link requires at most one unit of time (time unit is reasonable since time unit represents communication delay rather then actual data transmission time). The improvement comes from dynamic backtracking and elimination of transit nodes, at the expense of a small increase in message length. According to the authors their algorithms provide the basis of fault tolerant algorithms for reconstructing a DFS tree in a interconnected communication of a dynamic environment. Collin and Dolev present in [57] a fault tolerant DFS tree algorithm for a simple model of computation.

For completeness, we specify a distributed DFS algorithm, which is similar to the algorithm of Makki and Havas [56]. The algorithm was named Ddfs-Improved. Our improvement is related to the capacity of each node to discover, during the execution of the algorithm, its complete tree (parent, pseudo-parents and children) at the expense of a small increase of messages length. Since the execution of the algorithm is sequential it is suitable for synchronous and asynchronous methods. The Ddfs-Improved, Algorithm 9, is describe in Chapter 4.

2.6 Leader Election

The problem of choosing a leader from a set of *candidates* is called *Election.* A leader is an agent between a group which all other agents acknowledge as being distinguished to perform some special task [58]. Initially, each agent has a unique identifier chosen from a totally ordered set, and do not know the identifier of any other agent. At this point, it is impossible for an agent to elect a predetermined leader. Then communication between two nonadjacent agents only is possible if they use adjacent nodes to communicate to each other.

The leader election problem is meaningless in the context of anonymous system [58]. Moreover, even if the system is not anonymous, the leader election problem can only be solved for a graph **G** if every agent's identification is unique in **G**, making possible that all identification can be assumed to be totally ordered. In the approaches to leader election that take the leader with the greatest identification, this is a fundamental assumption. Even if this is not the criterion, the ability to compare two candidates' identification to break ties must exist. If it is not possible, any criterion to select a leader might deadlock for the absence of a tie breaker.

It is important to point out that if a graph **G** undergoes a fail or a topology change occurs (in broader terms), the leader election algorithm must be prepared to treat the recovery steps. Failures of several types, may occur during or before the execution of the election algorithms, so messages may be lost. As cited by Hosame H. and Abu-Amara in [59], failures can be *fail-stop*[5] or *Byzantine*[6]. However, the authors propose an algorithm that considers a failure type nominated as *intermittent*: more malicious than fail-stop failure but less malicious then Byzantine failure. Bar-Yehuda et al. [60] solve the same problem for *fail-stop* failure. They also assume that if a channel fails, then it fails before

[5]The fail-stop failure is the most benign of the failure types; a failed processor stops sending messages, and a failed communication channel stops transmitting messages. In the fail-stop failure, once a processor or a channel stops sending messages, it never sends any other message.

[6]Byzantine failure is one of the most malicious of the failure types; processors or channels fail by altering messages, sending false information, and so on.

the execution of the algorithm. To their assumption their algorithm does not tolerate intermittent channel failure.

Since the leader election is not the focus of this work, we will not review extensively the literature in this area. We rely on the method used over DPOP's root election, as in Algorithm 1, presented in [61]. It is available through FRODO framework [62], used to execute some important simulations in this thesis. The Algorithm 1 describes a simple procedure that is based on the propagation of variable values, and starts by each variable assigning a value that can be chosen following different heuristics (line 1.3). One of the most common heuristic is the *most connected* heuristic, based on the assignment of the number of neighbors connected to each agent. An agent, chooses a neighbor as its first child and sends message. Using e.g., the most connected heuristic, the neighbor with the highest degree is chosen, requiring that each agent preliminary sends its degree to all its neighbors. The algorithm finishes when each agent discovers the agent whose value is equal to the maximum, and choose it as the root of the pseudo-tree (lines 1.10 to 1.12). To make sure that only one agent has the highest value, a value tie break should be performed and the agent name may be a criterium. To guarantee that just one agent has the highest value, the agents require the knowledge of an upper bound ϕ_{max} on the diameter of the constraint graph (the maximum distance, in number of edges, between any pair of agents). If the constraint graph is not totally connected, this algorithm will elect one root per connected component of the graph.

Algorithm 1: Root election algorithm

```
1.1  Parameter: $\phi_{max}$ = the maximum distance, in number of edges, between any pair of agents
1.2  Initialize:
1.3  $value_{a_j} \leftarrow$ value that can be chosen following different heuristics
1.4  $max_{value} \leftarrow value_{a_j}$
1.5  for $\phi_{max}$ times do
1.6      Send $max_{value}$ to all neighbors
1.7      Get $(max_{value_{a_1}}, \cdots, max_{value_{a_k}})$ from all neighbors
1.8      $max_{value} \leftarrow \max(max_{value}, max_{value_{a_1}}, \cdots, max_{value_{a_k}})$
1.9  end
1.10 if $max_{value} = value_{a_j}$ then
1.11     $a_j$ is the elected root
1.12 end
```

The election of a leader normally occurs, for instance, in a pre-processing phase when agents initially choose one of then as a root node, or when there is a situation of a malfunctioning agent in the network, and it must be replaced. In practice the leader election algorithm can also be executed at predetermined or configurable time intervals, or the leader may be fixed, or may be a gateway of the network.

Chapter 3

Modeling Channel Allocation Problem

Contents

3.1 Introduction

The objective of a WLAN is to provide wireless connectivity over a certain coverage area surrounding the access point (AP). In practice a large number of access points may be operating in the same region. This environment will lead to a scenario where the *management* of wireless channels will be extremely important because of *co-channel* and *adjacent-channel* interferences. These interferences represent the main factor that significantly reduces network performance. Wireless propagation mechanisms (reflection, refraction and diffraction) are directly related to the location, size, and electrical properties of physical objects in the surroundings [17]. Therefore, the AP location affects the attenuation pattern, the multipath fading and interference levels of the received signal experienced by users or others APs in the neighborhood. This reality creates a challenge for network researches and network managers whose goal is to reach the best performance of the wireless network.

The problem of optimizing the performance of a WLAN with multiple APs, requires that channels and power of all APs have to be set to maximize the signal-to-interference ratio, while retaining a large coverage area. As the number of possible channel and power assignments grows exponentially with the number of APs, we have a NP (non-deterministic polynomial) combinatorial optimization problem [32]. An algorithm for searching the exact solution has obviously a larger complexity, however its complexity can be reduced by making use of the optimization problem properties.

Our goal is to present distributed methods based on multiagents concept [40], allowing any set of APs to coordinate for the search of the best solution for the channel allocation problem. We argue that the DCOP method [43, 63] meets these requirements. We initially adopted a distributed synchronous and parallel solution, that like some conventional distributed approaches as Asynchronous Backtracking (ABT) algorithm for DisCSP (Distributed Constraint Satisfaction Problem) in [64, 65], its extensions in [66] and the Synchronous Branch and Bound (SynchBB) algorithm in [67], ensures the optimal and suboptimal solution.

3.2 Formal DCOP Definition

In the artificial intelligence research area, the process of finding a solution to a number of constraints that imposes conditions over a set of variables is classically denoted *constraint satisfaction* [68]. Formally, a CSP (*Constraint Satisfaction Problem*) is defined by a set of variables whose state must satisfy the constraints, which is solved by constraint satisfaction methods. DisCSP (*Distributed Constraint Satisfaction Problem*) is an elegant formalism developed to address CSPs, where problem solutions are characterized by a designation of satisfactory or unsatisfactory [43, 69]. When the solution requires a degree of quality or cost the problem is modeled as a COP (*Constraint Optimization Problem*). In the last few years research focusing on a general framework for distributed COP, denoted DCOP, has been increased [43, 69, 70].

A DCOP consists of a set of variables that are distributed to a group of collaborative agents as *valued* constraints: constraints that are described as valuable functions that return values in a specific range. The goal is to optimize a global objective function, minimizing the cost of satisfied constraints. A large class of coordination problems can be modeled as a DCOP [43], including distributed planning, distributed scheduling and distributed resource allocation. In this work the wireless channel allocation problem in WLANs is modeled and solved as a DCOP.

A DCOP can be formally defined by considering a set of N agents, $A = \{a_1, a_2, \ldots, a_N\}$, and a set of N values, $D = \{d_1, d_2, \ldots, d_N\}$, where each value d_j is assigned to an agent a_j and belongs to a finite discrete domain $\mathbb{D}_j$. An agent a_j has control of its stored value d_j and knowledge of its domain $\mathbb{D}_j$. Mathematically, a DCOP is modeled by an undirected constraint graph $\mathbf{G} = (V, E)$ where V is the set of vertices (nodes) and E is the set of conflict edges (links). Each agent a_j represents a vertex $v_j \in V$ in the constraint graph. For each pair of agents (a_i, a_j) connected by a conflict edge $e_k = (i, j) \in E$ in the constraint graph, a cost function $f_{ij}(x, y) : \mathbb{D}_i \times \mathbb{D}_j \to \mathbb{R}$ is defined. The values x and y represent the current choices of agents a_i and a_j, respectively. Agents have to coordinate themselves in order to find a set of values that optimize a global function establishing the costs for the constraints. Therefore, the goal of a DCOP algorithm is to find the optimal set, denoted D^*, whose values minimize a global cost function $g(D) = g(d_1, d_2, \ldots, d_N) = \sum_{e_k \in E} f_{ij}(x, y)$. If a pair of constrained agents (a_i, a_j) has different associated domains $\mathbb{D}_j \neq \mathbb{D}_i$, and if agent a_j knows its own domain, then it must know the domain of the connected agent a_i. However, in many DCOP scenarios, discrete and finite domains, $\mathbb{D}_j$, are the same for all agents.

There are basically two classes of algorithms for solving a DCOP. The first class runs directly over the constraint graph, without imposing hierarchical ordering on the communication process among agents. Some examples of these algorithms include *NC-BB*, *ADOPT-NB* and *OPT-APO* [71–73]. The second class, where ADOPT is included, requires predefined ordering for message exchanging among agents during execution. In such cases the agent's ordering is usually obtained making use of a *pseudo-tree* equivalent as in [47]. Figure 3.1 illustrates the agent notation. However, it is important to point out that two non-neighbors agents a_{j1} and a_{j2} may be connected in the constraint graph by a conflict edge if both are linked to a common descendent agent a_k.

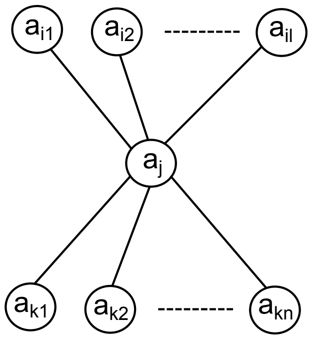


Figure 3.1: Notation of the agents for the description of the problem.

To apply DCOP as a method for **Channel Assignment in Wireless Networks**, a undirected *constraint graph* **G** should be obtained from the physical network topology. Considering, as an example, a WLAN scenario composed of 12 APs, whose physical layout is shown in Figure 3.2. The dotted lines represent that there is interference between the APs. This means that their respective channel allocations will affect the performance of both. Therefore, the graph represented by the connections between APs is the *constraint graph* **G** to be considered by the channel allocation problem. In such case, the set of vertices V of the constraint graph **G** represents the APs, and the set of conflict edges $E \subseteq V^2$ represents the existence of interference between APs, as indicated. In the DCOP context, the APs represent the agents and must be prioritized in a total order, making use of a *pseudo-tree*. An initialization procedure using a predefined common control channel is required allowing each AP to detect the APs in the neighborhood (constraint graph) and also to configure its logical ordering. This can be accomplished making use of Leader Election Algorithm as in Algorithm 1 and a Spanning-Tree algorithm as in Algorithm 9. It is necessary a graph knowledge for collaboration among APs of different administrative domains. This assumption is feasible since by collaborating with each other, the APs can get a more efficient channel allocation for all of them. The initialization procedure may be started time-to-time by any AP which detects the necessity of channel reallocation. This AP represents the root node of the pseudo-tree.

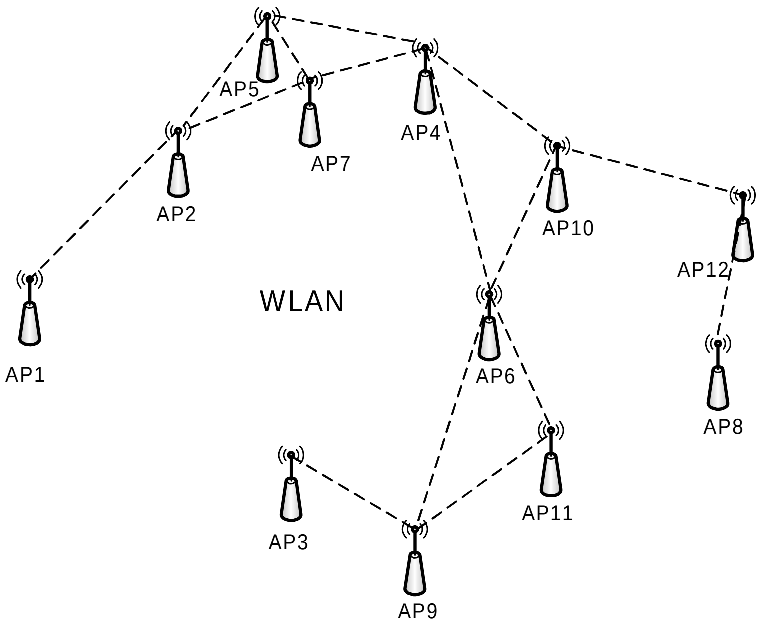


Figure 3.2: Example of a physical WLAN deployment.

3.3 ADOPT Algorithm

Modi *et al.* [43,72,74] pointed out that DCOP is able to model a wide variety of distributed reasoning tasks, but unsuccessfully provides theoretical guarantees on global solution quality while allowing agents to execute their tasks asynchronously. They showed how to circumvent this problem by allowing agents to make local decisions based on conservative cost estimates. This novel approach results in a distributed constraint optimization algorithm called ADOPT (Asynchronous Distributed OPTimization). According to Modi et al., ADOPT is the first algorithm for DCOP that can find the *optimal* solution, or a *suboptimal* solution within a user-specified distance from the optimal, using only localized asynchronous communication and polynomial space at each agent. The basic ideas behind ADOPT include: 1) a novel asynchronous search strategy where solutions may be discarded before they are proved to be suboptimal, 2) an efficient reconstruction of those discarded solutions, and 3) built-in termination detection. The first version of ADOPT presented in [74] was called Simple-ADOPT, and was proved to be optimal. Simple-ADOPT is a backtracking search algorithm that is executed asynchronously and in parallel on every agent, updating lower bounds on global solution quality. The Simple-ADOPT algorithm requires agents to be prioritized in a order, making use of a pseudo-tree (variables are ordered along a pseudo-tree). The first step is the root election, e.g. Algorithm 1. Once the root variable election is complete, the elected one starts a distribute depth-first traversal of the constraint graph, equivalent to the pseudo-tree arrangements described by Freuder and Quinn [47], considering itself as the root. During the pseudo-tree process, each agent a_j constructs its set of variables, defined as follows:

- P_j: parent node of a_j, in the hierarchy of the pseudo-tree (if a_j is not the elected root);
- C_j: the set of children of a node a_j,
- PP_j: the set of pseudo-parents of a node a_j;
- PC_j: the set of pseudo-children of a node a_j (if a_j is not a leaf node);

Consider, for instance, the DCOP described by the constraint graph, **G**, shown in Figure 3.3a, where each agent a_j is depicted as a node and the lines represent the set of conflict edges. The term *parent* is used to refer to an agent's immediate higher priority agent in the ordering and the term *child* to refer to an agent's immediate lower priority agent in the ordering. Let P_j and C_j be the set of parents (nodes a_i) and children (nodes a_k) agents of an agent a_j, respectively. Two agents are neighbors if there is a link between them in the pseudo-tree.

Figure 3.3b shows the total order on agents, obtained by a DFS algorithm [47], for the constraint graph of Figure 3.3a. Given this ordering, an agent communicates its value d_j to all lower priority neighbor agents, as indicated in Figure 3.3c, and communicates a lower bound cost to a unique higher priority neighbor as indicated in Figure 3.3d. Agents communicate using VALUE and VIEW control messages. The flow direction of VALUE and VIEW messages are also indicated in Figures 3.3c and 3.3d, respectively. These graphs are required input parameters for the ADOPT algorithm and they will be denoted as VALUE and VIEW graphs, for short. Let P_j^{value} and C_j^{value} be the set of parents and children of an agent a_j in the VALUE graph, respectively. Likewise, let P_j^{view} and C_j^{view} be the set of parent and children of an agent a_j in the VIEW graph. Taking agent a_5 as an example, from Figures 3.3c and 3.3d we have $P_5^{value} = \{6, 8\}$ and $P_5^{view} = \{6\}$. Therefore agent a_5 receives VALUE messages from agents a_6 and a_8, and sends a VIEW message to agent a_6. It is important to point out that in the ADOPT algorithm, an agent a_j has only one parent in the VIEW graph.

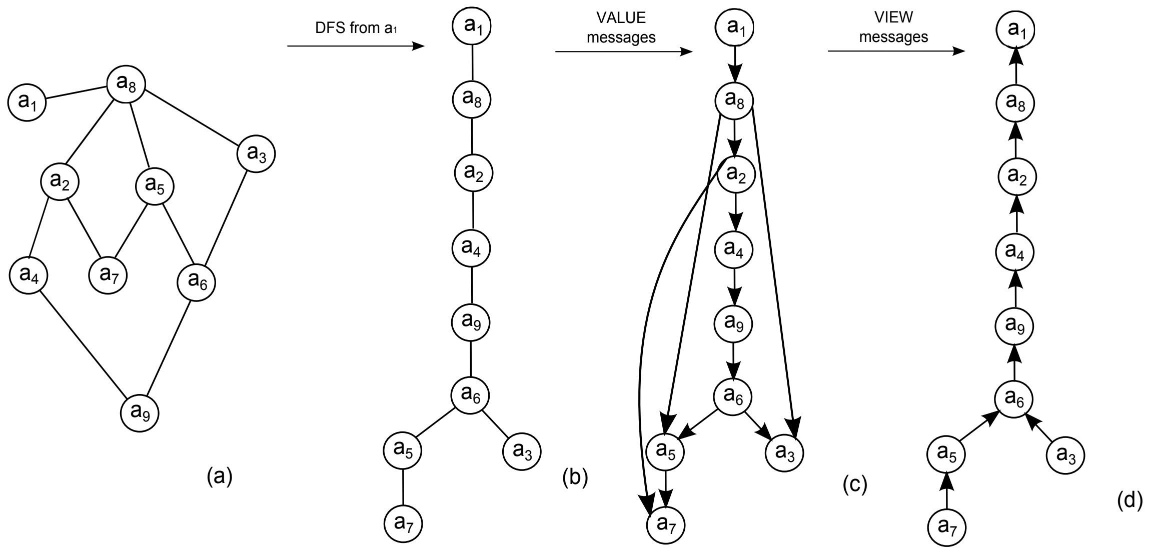


Figure 3.3: DCOP Example: (a) Constraint graph; (b) A constraint graph on totally ordered agents; (c) Graph of VALUE messages; (d) Graph of VIEW messages.

In the following we briefly describe ADOPT algorithm [74] whose pseudo-code is presented in Algorithm 2. A pair agent/value (a_j, d_j) is called a *view*. The variable $\mathbf{Currentvw}_j$ of an agent a_j is the current set of *views* $\{(a_i, d_i)\}$, composed by all linked ancestor's agent/value pairs, which means that $a_i \in P_j^{value}$. We say that two views ($\mathbf{Currentvw}_j$ and $\mathbf{Currentvw}_i$) are compatible if each pair of corresponding entries have the same assignment. It means that for every pair $(a_{k_i}, d_{k_i}) \in \mathbf{Currentvw}_i$ and $(a_{k_j}, d_{k_j}) \in \mathbf{Currentvw}_j$, if $a_{k_i} = a_{k_j}$ then $d_{k_i} = d_{k_j}$.

ADOPT begins by each agent a_j choosing locally and concurrently a value $d_j \in \mathbb{D}_j$. This value is sent to all its linked descendant agents $a_k \in C_j^{value}$, using VALUE messages. Then agents asynchronously wait for and respond to incoming messages. Given a received VALUE message from parent agent $a_i \in P_j^{value}$, agent a_j chooses its local value d_j according to

$$d_j = x \mid \min_{x \in \mathbb{D}_j} \sum_{a_i \in P_j^{value}} f_{ij}(d_i, x) \tag{3.1}$$

where $f_{ij}(\cdot)$ is defined according to the problem of interest. During the algorithm execution, each agent a_j must deal with three distinct cost values, the *local cost*, the *current lower bound cost* and the *estimated lower bound cost.* The *local cost*,

$$lc_j(d_j) = \sum_{a_i \in P_j^{value}} f_{ij}(d_i, d_j), \tag{3.2}$$

is computed as the sum of cost functions for all VALUE messages received from ancestors using the chosen value d_j obtained from (3.1). The *current lower bound cost* for a subtree (initially assigned to zero) is updated according to

$$sc_j(d_j) = \max\left(sc_j(d_j), z_k^*\right), \tag{3.3}$$

where z_k^* is the *estimated lower bound cost* received from children agents in the VIEW graph. At node a_j the cost z_j^* is computed as

$$z_j^* = lc_j(d_j) + sc_j(d_j) \tag{3.4}$$

where d_j is now given by

$$d_j = x \mid \min_{x \in \mathbb{D}_j} lc_j(x) + sc_j(x). \tag{3.5}$$

If an agent a_j does not have any descendant, its *estimated lower bound cost*, z_j^*, is just its local cost $lc_j(d_j)$. Whenever an agent a_j receives a VALUE message from a *linked ancestor* $a_i \in P_j^{value}$, it stores the current received value (a_i, d_i) into its $\mathbf{Currentvw}_j$ variable, which represents the current context of a_j. Agent a_j then reports the estimated lower bound cost z_j^* to its parent agent $a_i \in P_j^{view}$, by a VIEW message. In its turn, parent agent a_i will use the received *estimated lower bound cost* z_j^* in order to compute its *current lower bound cost* for that subtree, $sc_i(d_i)$. The parent agent receives a VIEW message but discards this message when there is a view incompatibility. This situation can happen in two cases: either the parent agent a_i has a more up-to-date current view than its child a_j, or a_j has a more current up-to-date view than its parent a_i. When an agent a_j reports z_j^* to its parent $a_i \in P_j^{view}$, it assumes the value d_j that has minimized z_j^*.

Finally, whenever an agent a_j receives a VALUE message it updates $\mathbf{Currentvw}_j$. By storing only one current view, ADOPT has linear space requirements at each agent. The algorithm reaches a stable state when all agents are waiting for incoming messages. Then the complete assignment chosen by the agents is equal to the optimal assignment set, denoted D^*, whose values minimize a global cost function, $g^* = g(D^*)$.

Algorithm 2: Simple-ADOPT

```
2.13 Parameters:
2.14   $\mathbf{Currentvw}_j = \{(a_i, d_i)\}$ : Current view of agent $a_j$
2.15   $d_j$: Current value of $a_j$
2.16   $sc_j(d_j)$: Current lower bound on cost for subtree rooted at child $a_k \in C_j^{view}$
2.17 Initialize: $\mathbf{Currentvw}_j \leftarrow \{\}$; $\forall d_j \in \mathbb{D}_j$: $sc_j(d_j) \leftarrow 0$
2.18 HillClimb;
2.19 WHEN Received_VALUE_Message[$a_i, d_i$]
2.20   Add $(a_i, d_i)$ to $\mathbf{Currentvw}_j$;
2.21   if ($\mathbf{Currentvw}_j$ changed) then
2.22     $\forall d_j \in \mathbb{D}_j$: $sc_j(d_j) \leftarrow 0$
2.23   end
2.24   HillClimb
2.25 end
2.26 WHEN Received_VIEW_Message[$\mathbf{Currentvw}_k, sc_k$]
2.27   if ($\mathbf{Currentvw}_k$ is compatible with $\mathbf{Currentvw}_j$) then
2.28     $sc_j(d_j) \leftarrow \max(sc_j(d_j), sc_k)$;
2.29     if ($sc_j(d_j)$ changed) then
2.30       HillClimb
2.31     end
2.32   end
2.33 end
2.34 Procedure HillClimb
2.35   $\forall d_j \in \mathbb{D}_j$ compute $z_j(d_j) = \sum_{a_i \in P_j^{value}} f_{ij}(d_i, d_j) + sc_j(d_j)$
2.36   $d_j = d \mid \min_d z_j(d)$
2.37   $z_j^* = z_j(d_j)$
2.38   Send_VALUE_Message[$a_j, d_j$] to all $a_k \in C_j^{value}$;
2.39   Send_VIEW_Message[$\mathbf{Currentvw}_j$, $z_j^*$] to $a_i \in P_j^{view}$;
2.40 end
```

3.4 DPOP algorithm

The DPOP algorithm [41,63,69] employs an utility propagation method, based on *dynamic programming* inspired by the sum-product algorithm. It considers a pseudo-tree arrangement of the constraint graph **G**, resulting from the execution of a distributed algorithm as presented by Adrian Petcu in [41]. The pseudo-tree in Figure 3.4 is composed of *tree-edges*, shown as solid lines, and *back-edges*, shown as dashed lines, that are not part of the spanning tree. A path in the graph that is entirely made of *tree-edges* is denoted as a *tree-path* (e.g. a_2 to a_1 in Figure 3.4).

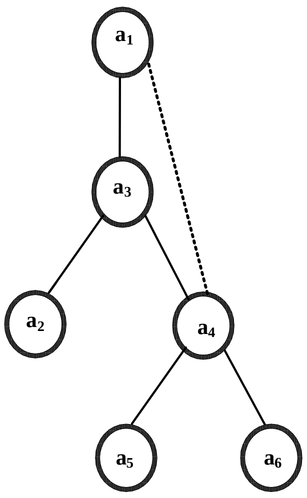


Figure 3.4: Example of a pseudo-tree arrangement.

The principle of DPOP algorithm [41,63,69] operation, whose pseudo-code is presented in Algorithm 3, is described below and the following definitions must be considered:

- Variables:
  - $A = \{a_1, a_2, \dots, a_N\}$: set of agents;
  - $D = \{d_1, d_2, \dots, d_l\}$: agent's values;
  - $D^n = \underbrace{D \times \cdots \times D}_{n}$: the cartesian product;
  - P_j: parent of a node a_j (if a_j is not the elected root), the single node above in the hierarchy of the pseudo-tree that is directly connected to a_j through a tree-edge (e.g. $P_3 = \{a_1\}$ and $P_4 = \{a_3\}$);
  - C_j: the children of a node a_j (if a_j is not a leaf node), the set of nodes below in the pseudo-tree that are directly connected to a_j through tree-edges (e.g. $C_1 = \{a_3\}$ and $C_3 = \{a_2, a_4\}$);
  - PP_j: the pseudo-parents of a node a_j (if a_j is not the elected root), the set of nodes above in the pseudo-tree that are directly connected to a_j through *back-edges* (e.g. $PP_4 = \{a_1\}$);
  - PC_j: the pseudo-children of a node a_j (if a_j is not a leaf node), the set of nodes below in the hierarchy of the pseudo-tree that are directly connected to a_j through *back-edges* (e.g. $PC_1 = \{a_4\}$);

- Sep_j: the separator of node a_j, all ancestors of a_j which are connected to a_j or with descendants of a_j (e.g $Sep_3 \doteq \{a_1\}$ and $Sep_4 \doteq \{a_3, a_1\}$);
- lc_j: local cost for all possible set of combinations between a_j and Sep_j, considered as utility;
- **LocalView**$_j$: a local variable of an agent a_j, contains all possible set of combinations between a_j and Sep_j, and their respective local cost lc;
- X^*: the optimal set of values for a_j, and to its Sep_j;
- **AgentView**: the optimal set of values a_j sends to its child a_k;
- $UTIL_j^{P_j}$: the $UTIL$ message sent by agent a_j to its parent agent P_j:
- $VALUE_{P_j}^{a_j}$: the $VALUE$ message a_j receives from its parent P_j;
- $f(.)$: the cost function defined according to the problem to be solved.

A node a_j can easily find its separator Sep_j considering the union of: separators received from its children, its parent and pseudo-parents, minus itself. Formally,

$$Sep_j \doteq \cup_{a_k \in C_j} Sep_k \cup P_j \cup PP_j \setminus \{a_j\}. \tag{3.6}$$

- Hypercube: set of values in a UTIL message, composed of optimal utility obtained in the subtree for each element of Sep_j. Thus, messages are exponential in the separator size (bounded by the induced width).
- Standard relational operators:
  - Selection (R,c): takes a relation R and the condition c, and yields a new relation with only the tuples in R that satisfy the condition c;
  - $\bigoplus$ - JOIN or sum: takes two relations, and yields a new relation that consists of the tuples combined on all their common variables;
  - $\perp$ - PROJECTION: also know in the literature as elimination, takes a relation and yields a new relation that consists of the tuples with certain column removed.
- Operators created for the algorithm:
  - USelectMIN(R): takes a relation R where each n-tuple corresponds to one line of the hypercube, including the local cost, and extracts different combinations with lowest local cost;
  - Min (R,x): takes a relation R and a specified element $x \in R$, and yields a new relation that consists of the tuples where value of x is the minimum.

- Max (R,x): takes a relation R and a specified element $x \in R$, and yields a new relation that consists of the tuples where value of x is the maximum.
- $\bigoplus_{UTIL_{C_j}} = \bigoplus_k^{C_j} UTIL_k^j$: apply join operator between the UTIL messages received from all children C_j of a node a_j;

Let R_j be a relation of node a_j with its parent P_j and pseudo-parents PP_j given by

$$R_j = \big\{(x_1, x_2, \cdots x_K, x_{K+1}) \quad | \quad (x_1, x_2, \cdots x_K) \in D^K\big\}, \tag{3.7}$$

having

$$K = |P_j \cup PP_j| + 1, \tag{3.8}$$

where x_1 is the value chosen by the local node, $(x_2, x_3 \cdots x_K)$ are the values chosen by Sep_j, and $\{(x_{K+1})\}$ representing the local cost computed as

$$lc_j = \sum_{i=2}^{K} f(x_i, x_1). \tag{3.9}$$

The function $f(.)$ is the cost function defined according to the problem to be solved. We regard the same cost function defined in Section 3.5 that considers normalized overlapping factors as function of the channel spacing, $f_{ij}(x, y) : |m - n| \to \mathbb{R}$.

The variable $\textbf{LocalView}_j$ has its current value according to

$$\textbf{LocalView}_j = \begin{cases} R_j, & \text{for } C_j = 0 \\ \bigoplus_{UTIL_{C_j}} \bigoplus R_j, & \text{for } C_j \neq 0. \end{cases} \tag{3.10}$$

Observe that the variable $\textbf{LocalView}_j$ is represented as

$$\textbf{LocalView}_j = \big\{(x_1, x_2, \cdots x_K, x_{K+1}) \quad | \quad (x_1, x_2, \cdots x_K) \in D^K\big\}, \tag{3.11}$$

having

$$K = |a_j \cup Sep_j| + 1. \tag{3.12}$$

After making $S_j = D^K$ it is defined the operation

$$\text{USelect}(\textbf{LocalView}_j) = \cup_{s \in S_j} \text{Max}\Big(\text{Selection}\Big(\textbf{LocalView}_j, (x_2, \cdots x_K) = s\Big), x_{K+1}\Big). \tag{3.13}$$

The UTIL message an agent a_j sends to its parent agent P_j is computed as

$$UTIL_j^{P_j} = \text{USelect}(\textbf{LocalView}_j) \perp x_1; \tag{3.14}$$

- Functions:

$$\text{BestValue}(\textbf{LocalView}_j, X, Sep_j) \tag{3.15}$$

allows to select the best value for a_j in the variable **LocalView** corresponding to the optimal assigned values in X for the elements in Sep_j, and

$$\text{ChooseValue}(X, Sep_k) \tag{3.16}$$

chooses values in X corresponding to elements of Sep_k.

The algorithm is divided in 3 phases. First the agents elect a leader (root node) and organize themselves in a pseudo-tree structure, a DFS transversal of the constraint graph, to be used in the next two phases, denoted UTIL and VALUE propagations, respectively.

3.4.1 UTIL propagation phase

In the UTIL propagation phase, the UTIL messages are propagated up the tree. The leaf agents immediately send UTIL messages to their parents, only through the tree-edges, so they initiate the process. A child a_k of node a_j will send a vector of the optimal utilities $\upsilon^*_{a_k}(d^l_j)$ (the variable $d^l_j \in \mathbb{D}_j$ where $\mathbb{D}_j = \{d^1_j, d^2_j, \cdots, d^l_j\}$, l identifies the values d_j in $\mathbb{D}_j$). The vector contains the best solution that a_k can obtain given the possible choices of its parent a_j in the domain $\mathbb{D}_j$ and its pseudo-parents in their respective domains $\mathbb{D}_i$.

After a node a_j receives all its children messages, it can compute the optimal values that can be achieved by the entire subtree rooted at a_j, as shown in Algorithm 3 line 3.18. Since all of a_j's subtrees are disjoint, by summing then up, it is possible to compute how much each of its children C_j values gives for the whole subtree rooted at itself (Algorithm 3 line 3.11). Then node a_j can stores its optimal values corresponding to the values received from its children C_j. The UTIL messages depend on the subtree rooted at the respective node, and the constraint between the node and its parent and pseudo-parents. Each node relays its messages according to the UTIL phase of the algorithm.

Considering Figure 3.4 as an example, the UTIL message a node a_2 sends to its parent a_3, depends only on target variable of a_3. If the constraint graph has cycles a message sent from a node to its parent may also depend on variables above the parent P_j. This may happens when there is a *back-edge* connecting the sending node a_k with the *back-edge handler* node a_b, whose variable is d_b. A example is the UTIL message that node a_4 sends to its parent a_3, that also depends on the variables of its pseudo-parent (*back-edge handler*) a_1, that is above its parent a_3.

In Figure 3.4, the UTIL message, $UTIL_4^3$ is computed by node a_4 where the node has to join all messages it receives from its children $C_4\{a_5, a_6\}$, and the relations it has with its parent a_3 and pseudo-parent a_1 (Algorithm 3 lines 3.9 to 3.17). The utilities that the subtree rooted at a_4 can achieve are not dependent only on its parent a_3 as the UTIL message the node a_2 sends to its parent a_3. The agent a_4 is also connected with a_1 through the back-edge (a_1, a_4), then it must take into account this dependency when sending its message to a_3.

Example 1 In Figure 3.4, a_4 computes its $UTIL_4^3$ message for a_3 as in Equation:

$$\mathbf{LocalView}_4^3 = \overbrace{\underbrace{\overbrace{UTIL_5^4 \oplus UTIL_6^4}^{dim=\{a_4\}} \oplus R_4^3}_{dim=\{a_4,a_3\}} \oplus R_4^1}^{dim=\{a_4,a_3,a_1\}}; \quad UTIL_4^3 = \underbrace{\text{USelect}(\mathbf{LocalView}_4)\perp a_4}_{dim=\{a_3,a_1\}}. \tag{3.17}$$

This is where the dynamic programming approach is considered, and a node a_k (e.g. a_4) will compute the optimal value that its subtree can achieve for each value combination of its parent P_j (e.g. a_3) and its *back-edge handler* a_b (e.g. a_1). In Table 3.1 is shown how node a_4 will compute each value combination with its parent a_3 and pseudo-parent (*back-edge handler*) a_1. For each value combination of the tuple $\langle a_3, a_1\rangle$ node a_4 will compute its optimal value (Algorithm 3 line 3.18). The node a_4 will assemble an UTIL message, denoted as a hypercube with dimension $dim = \{a_3, a_1\}$ as in Equation 3.17. It takes into account the possible variable choices of its parent P_3 and also the choices of its back-edge handler a_1. The hypercube is a multi-dimensional matrix, with one dimension for each variable in $Sep_j \doteq \cup_{a_k \in C_j} Sep_k \cup P_j \cup PP_j \setminus \{a_j\}$. To the case of node a_4, $Sep_4 = P_4 \cup PP_4 \setminus a_4$. There are no separator from its children (a_5 and a_6) since they have no back-edge connections.

Table 3.1: UTIL message sent from a_4 to a_3, in Figure 3.4

$a_4 \to a_3$	$a_3 = d_3^1$	$a_3 = d_3^2$	...	$a_3 = d_3^l$
$a_1 = d_1^1$	$v_{a_4}^*(d_1^1, d_3^1)$	$v_{a_4}^*(d_1^1, d_3^2)$	...	$v_{a_4}^*(d_1^1, d_3^l)$
...	...	...	...	...
$a_1 = d_1^l$	$v_{a_4}^*(d_1^l, d_3^1)$	$v_{a_4}^*(d_1^l, d_3^2)$	...	$v_{a_4}^*(d_1^l, d_3^l)$

3.4.2 VALUE propagation phase

In the VALUE propagation phase, messages are propagated down the tree. This phase starts after the root node receives all its children messages. It can compute its optimal

Algorithm 3: DPOP

```
3.1  Procedure DPOP($A, D, \mathbb{R}$):
3.2      $\forall a_j \in A$;
3.3      PHASE 1: DFS arrangement - run token passing mechanism as in [41];
3.4      After execution, $a_j$ knows $P_j, PP_j, C_j, PC_j, Sep_j$ ;
3.5      PHASE 2: UTIL propagation (bottom-up UTIL message propagation starts from the
         leaf nodes)
3.6      $X = \varnothing$; $R_j = null$;
3.7      LocalView$_j = null$;
3.8      Compute $R_j$; // a sub set with all combination values between $a_j$ with $P_j$ and $PP_j$, and the
         computation of $lc_j$
3.9      forall the $a_k \in C_j$ //* for all children of $a_j$ - if $a_j$ is a leaf node, skip this */ do
3.10         ReceiveUTILMessage ($Sep_k, UTIL_k^j$); // wait for $UTIL$ message arrive from $a_k$
3.11         LocalView$_j$ = LocalView$_j \oplus UTIL_k^j$; //Join UTIL messages from $C_j$ as they arrive
3.12     end
3.13     if ($P_j == \varnothing$) // that means that $a_j$ is the root node then
3.14         $X \leftarrow$ Max(LocalView$_j, x_2)\perp x_2$; // Select best value $x_1$ for $a_j$ from max utility values $x_2$
3.15         Send VALUE ($X$) to all $C_j$
3.16         else
3.17         LocalView$_j$ = LocalView$_j \oplus R_j$; //also join relations with parent/pseudo-parents;
3.18         $UTIL_j^{P_j}$ = USelect(LocalView$_j$)$\perp a_j$ //use USelect to select the tuples with max utility
             and $\perp$ to eliminate self out of message to parent;
3.19         SendUTILMessage($Sep_j, UTIL_j^{P_j}$);
3.20     end
3.21     PHASE 3: VALUE propagation (top-down VALUE message propagation)
3.22     wait for $VALUE_{P_j}^j(X)$ message from $P_j$; // is the optimal assignment for the elements in
         $Sep_j$
3.23     $X^* \leftarrow$ BestValue (LocalView$_j, X, Sep_j$); // Selects best $a_j$ value in LocalView
         corresponding to optimal assigned values $X$ for the elements in $Sep_j$
3.24     forall the $a_k \in C_j$ /*for all children of $a_j$ - if $a_j$ is a leaf node, skip this*/ do
3.25         AgentView= $\varnothing$;
3.26         AgentView $\leftarrow$ ChooseValue ($X, Sep_k$); // chooses values in X corresponding to
             elements of $Sep_k$
3.27         Send $VALUE_{P_j}^k$(AgentView) to all $C_j$;
3.28     end
3.29 end
```

solution, based on all the UTIL messages received from its children, as in Algorithm 3 line 3.14. The root node then choose the value that will leads to its optimal solution and informs its children, C_j, sending a VALUE message to all of them. After receiving the VALUE message from its parent P_j, each node is able to recognize the optimal value x_j^* for itself (Algorithm 3 line 3.23) and pass this information (via a VALUE message) to its children (Algorithm 3 lines 3.24 to 3.28). It is important to point out that the $VALUE_{P_j}^{a_j}$ message a node a_j receives from its parent P_j, also contains the values of all the variables that were present in the context of a_j's UTIL message for its parent P_j (Algorithm 3 line 3.27). Based on the example of Figure 3.4, the root node a_1 sends a_3 the message $VALUE_1^3(x_1^*)$, a_3 sends a_4 the message $VALUE_3^4(x_1^*, x_3^*)$ and a_4 sends to its children a_5 and a_6 the respective messages $VALUE_4^5(x_4^*)$ and $VALUE_4^6(x_4^*)$. Then the leaf nodes choose their optimal value. At this moment, the algorithm is finished for the nodes a_1, a_3,a_4, a_5 and a_6.

According to DPOP, the number of UTIL and VALUE messages are directly related to the number of agents n in the problem. Then the UTIL phase requires $n - 1$ bottom-up messages, and the VALUE phase requires $n - 1$ top-down messages (one through each tree-edge). However, the size of messages can reach exponential dimensions. As in [41] the largest UTIL message contains $O(D_{max}^{w_i})$ cost values, where the size D_{max} is the size of the largest domain, and w_i is the *induced width* of the used pseudo-tree, which is bounded above by the number of variables n, and bounded below by the *treewidth* w of the constraint graph. Each VALUE message contains at most w_i optimal assignments to variables. Or as also proposed by [41]: "*The induced width of a graph G along a given DFS arrangement is equal to the size of the largest separator of any node in the DFS arrangement*".

3.5 Interference Model

The *interference model* defines the set of links that can interfere with any given link in the network. Due to the broadcast nature of the wireless links, transmission along a communication link (between a pair of wireless nodes) may interfere with transmissions along other communication links in the network. Two interfering links cannot engage in successful transmission at the same time if they transmit on the same channel. The management of available wireless channels is of extremely importance because *co-channel* and *adjacent-channel* interference can significantly reduce the network performance [75]. The figure of merit for signal reception under interference is the signal-to-interference-plus-noise ratio (SINR) [17]. In the context of wireless networks, the SINR perceived by an AP or a client is a function of the overall concurrent transmissions occurring in the same channel and in partially overlapped adjacent channels. Considering an user u_k associated to an AP a_i ($u_k \in \text{Clients}(a_i)$), operating at a channel c_n. The SINR experienced by a_i is defined as

$$\gamma_{a_i}^{c_n} = \frac{S_{a_i \leftarrow u_k}}{N_{a_i} + I_{a_i}} \tag{3.18}$$

where $S_{a_i \leftarrow u_k}$ denotes the average received signal power from client u_k and N_{a_i} denotes also the noise floor and rogue interference. The interference power I_{a_i} is computed as

$$I_{a_i} = I_{a_i}^a + I_{a_i}^u = \sum_{j \neq i} S_{a_i \leftarrow a_j} + \sum_{u_l \notin a_i} S_{a_i \leftarrow u_l} \tag{3.19}$$

where $I_{a_i}^a$ and $I_{a_i}^u$ denote the interference power from all concurrent transmissions of other APs and users, respectively. The interferers may be operating in partial overlapped channels or even in the same channel. The number of clients associated with each AP and the average traffic volume of each user may be used as scaling factors in computing I_{a_i} [14, 17].

Any channel assignment strategy tries to minimize the interference experienced at the APs, or at the clients or both. Indeed, the SINR is primarily defined by the channel overlapping factor because it impacts directly in the number of concurrent transmissions that may occur in a neighborhood area. In our study we consider only interference among APs in order to demonstrate the proposal effectiveness.

We assume IEEE802.11b/g based WLANs which operate in the ISM (Industrial Scientific Medical) unlicensed band, already described in Chapter 2. As pointed out in the SINR formulation, the interference level is directly proportional to the channel overlapping factor. Therefore we consider a global cost function based on this factor. Based on the approach of [32, 76] and [77], the spectrum overlapping between adjacent channels can be mathematically derived for IEEE802.11b interfaces, as follows. The unfiltered modulated signal has a power spectrum density (Watts/Hz) given by:

$$\mathcal{S}(n,f) = \begin{cases} \left| \frac{\sin[2\pi \mathcal{X}(n,f)]}{2\pi \mathcal{X}(n,f)} \right|^2, & \text{for } \mathcal{X}(n,f) \neq 0 \\ 1, & \text{for } \mathcal{X}(n,f) = 0 \end{cases} \tag{3.20}$$

where $\mathcal{X}(n,f) = \frac{f-2412-5(n-1)}{BW}$, BW is the null-to-null bandwidth and $n \in \{1, 2, \ldots, 11\}$ is the channel number. The radio interface employ IF filtering on both transmit and receive paths in order to reduce sidebands power. This filter has 3dB bandwidth of 17MHz and stopband 50dB down at ±22MHz. The approximated IF filter frequency response is

$$\mathcal{F}(n,f) = \frac{1}{[1 + 2.6\ \mathcal{X}(n,f)]^6}. \tag{3.21}$$

From (3.20) and (3.21) the spectrum overlapping factor $\mathcal{SO}(n,m)$ between channels n and m is computed as:

$$\mathcal{SO}(n,m) = \int_{2200}^{2700} \mathcal{S}(n,f)\mathcal{F}(n,f)\mathcal{S}(m,f)\mathcal{F}(m,f)\ df. \tag{3.22}$$

The normalized values of the overlapping factors $\mathcal{SO}(n, m)$ are presented in Table 3.2 as a function of the channel spacing $|n - m|$. In our strategy, the selected value d_j of an agent a_j represents the channel selection number of the AP$_j$, therefore the domain of d_j is defined as $\mathbb{D}_j = \{1, 2, \ldots, 11\}$. For a pair of connected agents (pair of interfering APs) in the constraint graph, the cost function for a pair of values (d_i, d_j) is defined as $f_{ij}(x, y) : \mathbb{D}_i \times \mathbb{D}_j \to \mathbb{R}$ where the values in $\mathbb{R}$ represent the interference overlap factors, as can be seen in Table 3.2.

The APs, by coordinating themselves, have to find a set of channel allocation values, D^*, that minimize the global cost function and then reduce the total amount of interference. The inverse of the cost function represents the gain or the utility (UTIL) function for the assigned channels between two nodes in the constraint graph.

Table 3.2: Interference Factor

Channel Spacing $\|n - m\|$	Overlapping Factor $(f_{ij}(x, y) : \|n - m\|)$
0	1
1	0.7272
2	0.2714
3	0.0375
4	0.0054
5	0.0008
6	0.0002
7 - 10	0

The global cost function reflects the overall interference degradation experienced by the network and consequently a proportional degradation in throughput. The main advantage in using a cost function in the evaluation of algorithms is that this approach allows us to indirectly infer about the network performance without the need of extensive packet level throughput simulations.

Aiming to demonstrate the proportionality between the overlapping factor and the network throughput, a simple 802.11b simulation scenario with 3 APs (4 clients per AP) was implemented and simulated in OPNET [78] Network Simulator. The main parameters are presented in Table 3.3. A heavy FTP/TCP traffic pattern among server and clients was used. It was considered three channel assignment possibilities, where APs were configured with channels $(1, 2, 3)$, $(1, 2, 11)$ and $(1, 6, 11)$. The average throughput per AP is presented in Figure 3.5, where it is included a 90% confidence interval for the results. As expected, a lower global cost value results in a higher throughput.

Table 3.3: Simulation Settings

Number of Aps	3
Distance between APs (m)	70
Clients per AP	4
Distance between APs and clients (m)	5 (symmetrically distributed)
Data rate (bps)	11 Mbps
Channel settings	(1, 2, 3), (1, 2, 11) and (1, 6, 11)
Transmit power (dBm)	20
Packet reception power threshold (dBm)	-90
BSS identifier	1, 2 and 3
Traffic pattern	FTP/TCP
Simulation time (min)	3
File size (bytes)	constant (15000000)
Traffic direction	100% upload

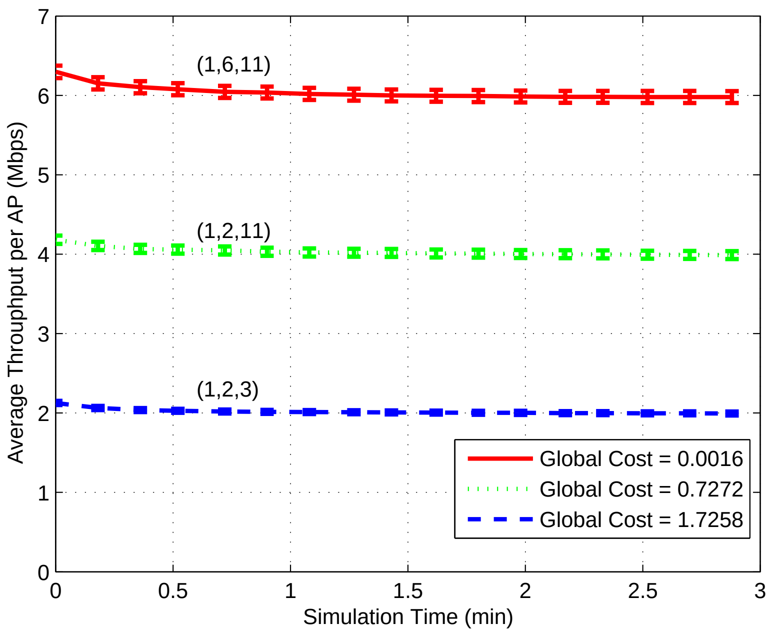


Figure 3.5: Simulated average throughput per AP.

3.6 Benchmark Algorithms

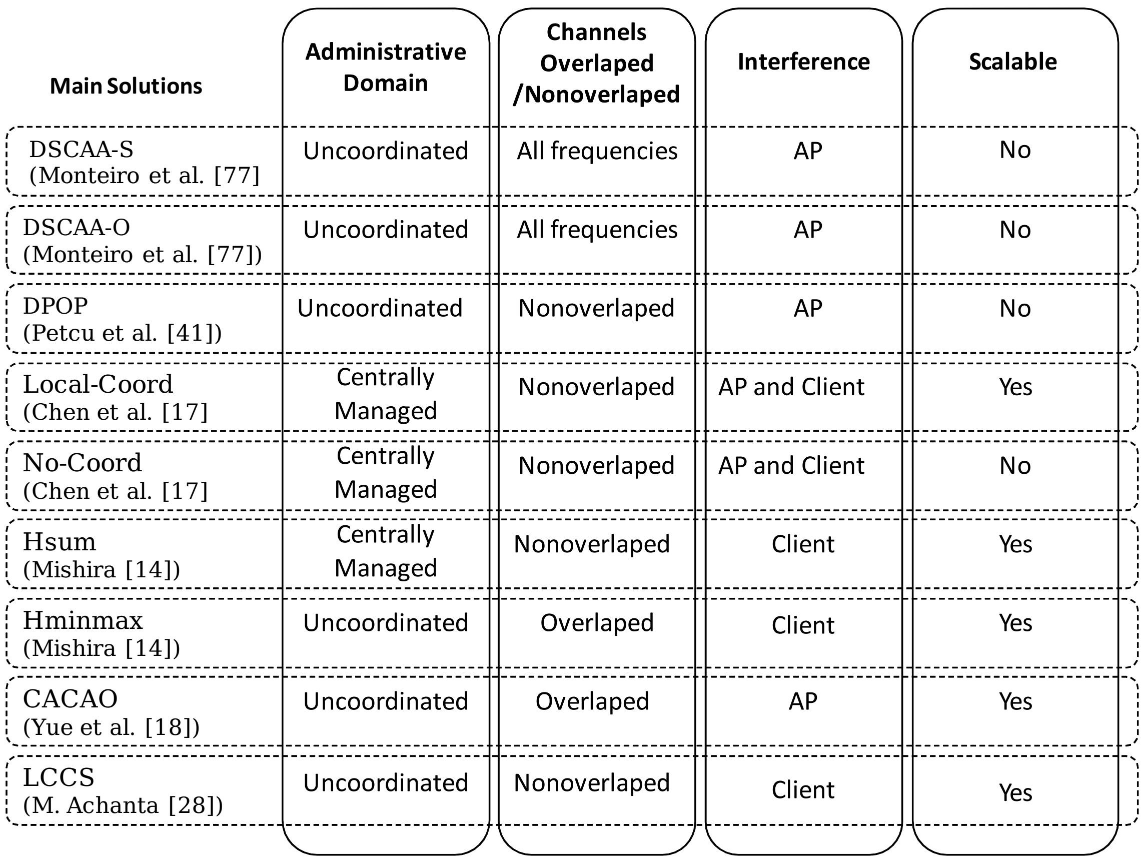
We would like to call attention to some details of the mentioned algorithms in Figure 3.6. Although the references should not be limited here, the methods and the performance of the algorithms were evaluated in the process of choosing the benchmark algorithms for this work. The evaluated aspects were:

1. To which type of environment a channel assignment is applicable (uncoordinated or centrally managed);
2. The type of channels frequencies are used (overlapping or non-overlapping channels);
3. From which perspective interference is modeled (from AP's or clients' point of view);
4. Whether a scheme is scalable (i.e., whether a solution exists for relatively large networks).

According to the considered aspects the adaptive algorithms may be centrally managed or applied to uncoordinated deployments. The solutions for the static algorithms are usually executed only once, so presenting simple complexity. When the planning is done, the channels are established for long term use. Most algorithms consider non-overlapping channels while the most recently tries to utilize both non-overlapping and overlapping ones. Due to high computational complexity, most algorithms use a heuristic view, others try to model interference also from clients' perspectives. As can be expected, the most channel assignment methods in the uncoordinated environments are scalable due to their distributed execution whereas those in the centrally managed environments are not because of their privileged centralized control.

Multiagent (DPOP, DCAA-S and DCAA-O) strategies outperform traditional algorithms (CACAO, LO-A, No-Coord, Hsum, Hminmax and LCCS), but present scalability problems, due to the exponential number or exponential size of the messages exchanged between the APs.

We decided to use Local-Coord (LO-A) as one of the benchmark algorithm because its is a simple "scalable" distributed algorithm, featuring a good balance between solution quality and number of exchanged messages between the APs. The algorithm proved to be an excellent candidate in the context of channels allocation, where messages can be exchanged between APs through a specific interface or a control channel. LO-A, is a measurement-based algorithm recently proposed by Chen et. al in [17] that has an iterative nature, use a physical model rather than a binary model for interference, and mitigate the impact of rogue interference. According to the proposal of LO-A algorithm, all or a subset of clients periodically measure the *in situ* interference power on all frequency channels and report the average measured power to their associated APs. The APs also measure their *in situ* interference power. Then, each AP is able to compute a metric called *weighted interference*, useful to place different weights to himself and its clients. The overhead in taking measurements is negligible because the measurements at APs or clients are performed during their idle time. The algorithm presents a mechanism for iteratively switching frequency channels to reduce the metric *weighted interference* seen in a group



Main Solutions	Administrative Domain	Channels Overlaped /Nonoverlaped	Interference	Scalable
DSCAA-S (Monteiro et al. [77]	Uncoordinated	All frequencies	AP	No
DSCAA-O (Monteiro et al. [77])	Uncoordinated	All frequencies	AP	No
DPOP (Petcu et al. [41])	Uncoordinated	Nonoverlaped	AP	No
Local-Coord (Chen et al. [17]	Centrally Managed	Nonoverlaped	AP and Client	Yes
No-Coord (Chen et al. [17]	Centrally Managed	Nonoverlaped	AP and Client	No
Hsum (Mishira [14])	Centrally Managed	Nonoverlaped	Client	Yes
Hminmax (Mishira [14])	Uncoordinated	Overlaped	Client	Yes
CACAO (Yue et al. [18])	Uncoordinated	Overlaped	AP	Yes
LCCS (M. Achanta [28])	Uncoordinated	Nonoverlaped	Client	Yes

Figure 3.6: Summary of Distributed Strategies for Channel Assignment in 802.11 WLAN

of nearby cells. Suppose M APs, indexed by $\mathbb{M} = \{1, 2 \cdots, M\}$, a *cell* is represented by $\mathbb{Z}_m = \{a_m\} \cup \{u_n \mid u_n \in Clients(a_m)\}$, meaning an AP a_m (or base station) and its associated users u_n, and, $\mathbb{G}_m$ the set of cells interfered by $\mathbb{Z}_m$.

An important point considered by the algorithm LO-A is the definition of nearby cells. Since an AP switches its channel, nearby cells may see changes in their weighted interference. So, cell $\mathbb{Z}_n$ is said to be interfered by cell $\mathbb{Z}_m$ (or $\mathbb{Z}_m$ interferes with $\mathbb{Z}_n$) if and only if an AP a_m or a user associated with a_m induces nonnegligible interference at an AP a_n or a user associated with a_n.

The base idea is to capture the overall interference seen in the cell, defined as a weighted sum of the average *in situ* measurements at AP a_m and all clients u_n associated with a_m. At each point in time (predefined, randomly chosen, or determined at runtime), one iteration of channel switching takes place where one or more APs switch their frequency channels according to specific mechanisms, whereas other APs and clients stay on their channels. It is also possible to capture the metric *weighted interference* according two simplified ways, representing practical metrics. The first, called *user-based*, places different weights on the *in situ* interference measurements based on the traffic volume and signal strength at each client. The second, called *AP-based*, includes the interference measurements at AP only. It is viewed as a simplified version of *user-based* form by considering that all users have the same traffic volume and signal strength.

APs and clients measure and average their *in situ* interference between successive iterations. Iterations keep taking place on different APs until the channel allocations converge. The characterization of the convergence point is a procedure that may vary and it is directly related to the topological environment where the algorithm is executed.

Figure 3.7 depicts the cells that see changes in weighted interference before and after AP a_1 switches its channel. Since the max weighted interference seen by cells 1-4 decreases, a_1 can switch to new channel. If a_1 switches from channel 1 to channel 3, only cell $\mathbb{Z}_1$ and the cells interfered by $\mathbb{Z}_1$ ($\mathbb{G}_1 = \{\mathbb{Z}_2, \mathbb{Z}_3, \mathbb{Z}_4\}$) see changes in their weighted interference. Since coordination among APs is confined in a local area, multiple APs that are enough far apart can simultaneously change their channels if a proper inter-AP protocol is employed. The number of APs that can simultaneously change channel may grow with the number of total APs; hence, LO-A is scalable. An other important feature of the algorithm is that deadlock never arises. Algorithm 4 presents the protocol for the distributed implementation of Local-Coord. The procedure to handle locking and unlocking requests is described in Algorithm 4 items (b) and (c), respectively. As an AP can be locked multiple times by different APs, Table 3.4 must be mentioned since it describes the variable ψ_m that denotes the number of times a_m was locked.

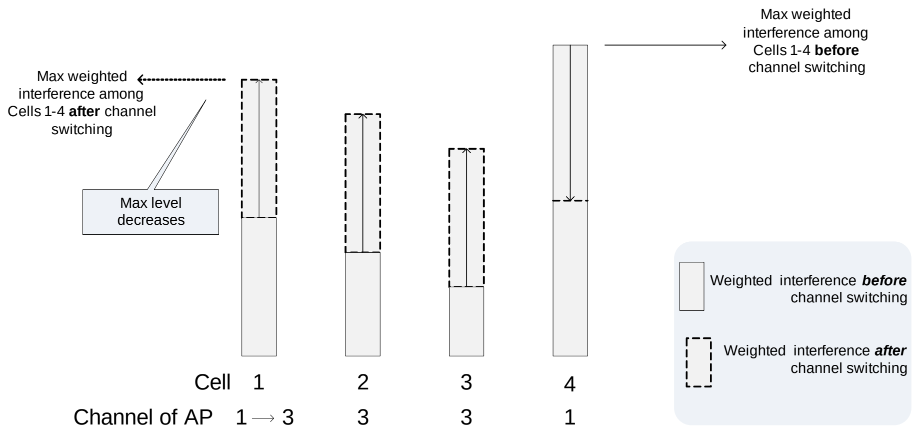

Max weighted interference among Cells 1-4 after channel switching
Max level decreases
Max weighted interference among Cells 1-4 before channel switching
Weighted interference before channel switching
Weighted interference after channel switching
Cell 1 2 3 4
Channel of AP 1 → 3 3 3 1

Figure 3.7: Max weighted interferences seen by cells 1-4 before and after AP a_1 switches from channel 1 to 3

Algorithm 4: Local-Coord - LO-A [17]

```
4.1  (a) a timer triggers $a_m$ to consider initiating channel switching. Then $a_m$ will execute the
     following procedure.
4.2  if $\psi_m = 0$ then
4.3      Phase1: Set $\psi_m = -1$ and send requests to lock all APs indexed by $\mathbb{V}_m$, i.e., $\{a_n : n \in \mathbb{V}_m\}$ ;
4.4      Phase2: Wait for replies from $\{a_n : n \in \mathbb{V}_m\}$;
4.5      if the replies indicate that $\{a_n : n \in \mathbb{V}_m\}$ were all successfully locked by $a_m$ then
4.6          $a_m$ switches its channel from $k$ to $k'$ and stays at $k'$ if max weighted interference seen by
             all $\mathbb{Z}_m$ is satisfied; otherwise, $a_m$ switches back to channel $k$;
4.7          Send message to unlock $\{a_n : n \in \mathbb{V}_m\}$;
4.8          else
4.9          Send message to unlock the APs among $\{a_n : n \in \mathbb{V}_m\}$ that were successfully locked by
             $a_m$. (Do not need to unlock the APs that could not be locked by $a_m$);
4.10     end
4.11     Set $\psi_m = 0$;
4.12 end
4.13 (b) Upon receiving a locking request from $a_m$, $a_n$ will do the following procedure.
4.14 if $\psi_n \neq -1$ then
4.15     Increase $\psi_n$ by one;
4.16     Reply to $a_m$ that $a_n$ was successfully locked by $a_m$;
4.17     else
4.18     Reply to $a_m$ that $a_n$ could not be locked;
4.19 end
4.20 (c) Upon receiving unlocking request from $a_m$, $a_n$ will decrease $\psi_n$ by one.
```

The authors [17] prove the convergence and show that their measurement-based algorithm for frequency allocation in wireless networks performs better than [14, 30, 31, 79, 80]. However to their considered environment, an AP needs to locally coordinate with others APs via a wired backbone network for channel switching. For comparison purposes, LO-A messages are exchanged among APs via a pre-defined wireless control channel.

It was also decided to use the algorithm Hsum [14] as a benchmark algorithm because it presents realistic scenarios of interference and channel reuse, also achieving interference reduction for sparse and dense topologies. The algorithm also has no scalability problems.

To compare the results with the optimal solution we used the DPOP algorithm itself, even with its scalability limitations.

Table 3.4: Variable ψ_m Used by LO-A [17]

ψ_m	Channel switching at a_m	Can a_m be blocked?
-1	a_m is in the process of switching its channel	No
0	a_m can initiate the process of channel switching	Yes
1 or more	a_m cannot initiate the process of channel switching	Yes

Chapter 4

Proposed Algorithms

Contents

4.1 Introduction

This chapter presents the algorithms that were developed to optimize the capacity of the IEEE 802.11 b/g networks by treating the channel allocation problem as a DCOP formulation. The algorithms were derived from Simple-ADOPT [2] [74] and DPOP algorithms [41, 63, 69]. They presented optimal and suboptimal solutions to the channel allocation problem. The suboptimal solutions were developed to reduce the number of messages

[2]The more recent versions of ADOPT [43] were not considered because they assume specific objectives like explicit determination of stopping criterion, requiring additional control messages and storage space for local information compared to Simple-ADOPT.

and the amount of information, exchanged between APs, maintaining the solution very close to the optimal.

The Simple-ADOPT based algorithms are:

- DCAA-O (*Distributed Channel Assignment Algorithm - Optimal*);
- DCAA-S (*Distributed Channel Assignment Algorithm - SubOptimal*).

The algorithms derived from DPOP algorithm are:

- DOCA (*Distributed Optimal Channel Assignment*);
- DSCA (*Distributed SubOptimal Channel Assignment*).

4.1.1 Assumptions

For the implementation of the algorithms, some assumptions were considered:

- The considered scenario refers to one or more stationary wireless networks, composed by APs arbitrarily distributed over an area;
- The locations of the APs do not vary with time;
- An undirected *constraint graph* $\mathbf{G} = (V, E)$ should be obtained from the physical network topology, where the set of vertices V of the constraint graph $\mathbf{G}$ represents the APs, and the set of conflict edges E represents the connectivity between interfering APs. This means that their respective channel allocations will affect the performance of each other due the co-channel or adjacent channel interference.
- An AP can measure the strength of an incoming signal and identify its interfering neighbors by using the *Receive Signal Strength Indicator* (RSSI);
- Each AP employs an additional control interface [10] for messages exchange of the channel assignment algorithms. Initially the APs change messages, between them, to elect a leader and to organize themselves in a pseudotree structure. Then the APs exchange messages during the execution of the distributed algorithms for channel allocation. To ADOPT-based algorithms, the exchanged messages are: VALUE, VIEW and TERMINATE messages. To DPOP-based algorithms: UTIL and VALUE messages;
- The dedicated control interfaces operate in the same pre-defined channel c, which is previously selected by the channel allocation protocol among one of the 11 available channels provided by the IEEE.802.11b/g standards. We assume as a first approach that channel c does not belong to any agent domain ($c \notin \mathbb{D}$);

- The algorithm may be started by any AP that detects performance degradation or increased interference level. Such AP is denoted as root access point;
- In a preliminary step, the root access point initiates a distributed algorithm for constructing the spanning tree;
- The optimal solution values were checked by exhaustive search.

4.2 DCAA Algorithms

Our ADOPT based algorithms are executed synchronously and in parallel on every AP in order to solve the channel assignment problem, searching for the optimal or suboptimal solutions. Their primary goal is to obtain the best channel assignment as well as to minimize the number of exchanged messages among APs. The first proposed algorithm, denoted DCAA-O, ensures the convergence to the optimal channel assignment. The second algorithm, called DCAA-S, is a modified version of DCAA-O, including a heuristic that contributes to efficiency rather than optimality, where the number of exchanged messages grows linearly. As a consequence, the number of exchanged messages is significantly reduced with respect to DCAA-O, at the cost of achieving a suboptimal channel allocation.

4.2.1 DCAA-O

The ADOPT algorithm operates asynchronously and in parallel on every agent. In many multiagent applications this is not a restriction. However, this is an undesirable characteristic for our proposed application since we want to minimize the number of overhead messages among APs. Therefore, we derived DCAA-O from ADOPT keeping the optimality conditions and imposing synchronous message control at agents.

The pseudocode of the DCAA-O method is shown in Algorithm 5. The parameters listed below are also based on the description of ADOPT, however some changes were necessary for the development of our algorithm:

- a_j: Current access point;
- a_1: Root access point;
- A'_i: Set of APs descendent of an ancestor $a_i \in P_j^{value}$;
- $P_j^{value'} = P_j^{value} \cap A'_i$: Subset of P_j^{value} whose nodes are descendent of an $a_i \in P_j^{value}$;
- **Currentvw**$_j = \{(a_i, d_i)\}$, $\forall a_i \in P_j^{value}$: Current view of AP a_j;
- **LocalCosts**$_j = \{[d_j, lc_j(d_j), sc_j(d_j)]\}, \forall d_j \in \mathbb{D}_j$: Temporary local costs of AP a_j for each choice d_j

- $\mathbf{SubTreeValues}_j(d_j) = \{(a_j, d_j), (a_k, d_k), \dots\}$, $\forall a_k \in$ subtree of a_j : Temporary subtree costs;
- $lc_j(d_j)$: Local cost of AP a_j given d_j and $\mathbf{Currentvw}_j$;
- $sc_j(d_j) = \sum_{a_k \in C_j^{view}} z_k(d_j)$: Subtree cost of a_j;
- $z_j(d_j) = lc_j(d_j) + sc_j(d_j)$ where $d_j = x \mid \min\limits_{x \in \mathbb{D}_j} lc_j(x) + sc_j(x)$;
- $z_{j_a}(d_j)$: Lowest *estimated lower bound cost* value;
- $\mathbb{D}_j^{'}$: Reduced domain of $\mathbb{D}_j$.

The choice of the local value d_j is an important point of the distributed algorithm execution. A given node a_j will iterate with its subtree, through its VALUE and VIEW children, for different temporary $d_j \in \mathbb{D}_j$ choices. During the iteration the node a_j builds its variables, $\mathbf{LocalCosts}_j$ and $\mathbf{SubTreeValues}_j$.

After the execution of the algorithm, as presented in Algorithm 5, the root node sends a message identified as *TERMINATE*, containing the optimal allocation solution for all other APs through the VIEW graph.

The DCAA-O algorithm is based on the execution of a procedure called **CutDomainD**, (Algorithm 6). The procedure is an improvement to reduce the number of exchanged messages between agents, avoiding an exhaustive search in the solution space. It reduces domain $\mathbb{D}_j$ to a new solution space $\mathbb{D}_j^{'}$. The logic implementation is based on APs *local cost* and in the lowest *estimated lower bound cost* value, $z_{j_a}(d_j)$, stored until the last iteration. A given node a_j will iterate with its subtree for different temporary $d_j \in \mathbb{D}_j$ choices. After testing the appropriate values $d_j \in \mathbb{D}_j$, selected through the execution of the routine **CutDomainD**, the node is able to identify the value d_j that minimizes the cost of the subtree in the context represented by $z_j(d_j)$. From this time the node a_j sends a VIEW message to its parent $a_i \in P_j^{view}$, that can continue with its procedure for choosing its local channel d_i, without compromising the solution optimality.

4.2.1.1 Algorithm Functioning

In a preliminary step, the root access point initiates a distributed algorithm for constructing the spanning tree [47], which allows the definition of VALUE and VIEW graphs on all nodes. Assuming a_1 as the root node, it must choose an initial value $d_1 \in \mathbb{D}_1$ to its local variable. This value is sent, via a VALUE message, to all its linked descendants in C_1^{value}. Then the root AP waits synchronously for the VIEW messages from its VIEW children in C_1^{view}, in order to update variables $\mathbf{LocalCosts}_1$ and $\mathbf{SubTreeValues}_1$.

Algorithm 5: DCAA-O

```
5.1  Procedure Initialization
5.2      $\textbf{Currentvw}_j = \emptyset$;
5.3      $\textbf{LocalCosts}_j = \emptyset$;
5.4      $\textbf{SubTreeValues}_j(d_j) = \emptyset \ \forall d_j \in \mathbb{D}_j$;
5.5  end
5.6  Procedure RootNode
5.7      forall the $(d_1 \in \mathbb{D}_1)$ do
5.8          Send_VALUE_Message($a_1$,$d_1$) $\forall a_k \in C_1^{value}$;
5.9          forall the $(a_k \in C_1^{view})$ do
5.10             Receive_VIEW_Message($a_k, z_k(d_k), \textbf{SubTreeValues}_k(d_k)$);
5.11             $\textbf{SubTreeValues}_1(d_1) = \textbf{SubTreeValues}_1(d_1) \cup \textbf{SubTreeValues}_k(d_k)$;
5.12             $sc_1(d_1) = sc_1(d_1) + z_k(d_k)$;
5.13         end
5.14         Add $[d_1, 0, sc_1(d_1)]$ to $\textbf{LocalCosts}_1$;
5.15     end
5.16     $d_1 = x \mid \min_{x \in \mathbb{D}_1} sc_1(x)$, where $sc_1(x) \in \textbf{LocalCosts}_1$;
5.17     Send_TERMINATE($\textbf{SubTreeValues}_1(d_1)$) $\forall a_k \in C_1^{view}$;
5.18 end
5.19 Procedure OtherNodes
5.20     while $TERMINATE\_QUEUE = \emptyset$ do
5.21         Receive_VALUE_Message($a_i$,$d_i$);
5.22         Add $(a_i, d_i)$ to $\textbf{Currentvw}_j$;
5.23         forall the $(a_i \in P_j^{value'})$ do
5.24             Receive_VALUE_Message($a_i$,$d_i$);
5.25             Add $(a_i, d_i)$ to $\textbf{Currentvw}_j$;
5.26         end
5.27         forall the $(d_j \in \mathbb{D}_j)$ do
5.28             $lc_j(d_j) = \sum_{a_i \in \textbf{Currentvw}_j} f_{ij}(d_i, d_j)$;
5.29             $sc_j(d_j) = 0$;
5.30             Add $[d_j, lc_j(d_j), sc_j(d_j)]$ to $\textbf{LocalCosts}_j$;
5.31         end
5.32         $\mathbb{D}_j^{'} = \mathbb{D}_j$;
5.33         if $(\boldsymbol{C}_j^{value} \neq \emptyset)$ then
5.34             forall the $(d_j \in \mathbb{D}_j^{'})$ do
5.35                 Send_VALUE_Message($a_j$,$d_j$) $\forall a_k \in C_j^{value}$;
5.36                 forall the $(a_k \in C_j^{view})$ do
5.37                     Receive_VIEW_Message($a_k, z_k(d_k), \textbf{SubTreeValues}_k(d_k)$);
5.38                     $\textbf{SubTreeValues}_j(d_j) = \textbf{SubTreeValues}_j(d_j) \cup \textbf{SubTreeValues}_k(d_k)$;
5.39                     Add $z_k(d_k)$ to $sc_j(d_j)$ in $\textbf{LocalCosts}_j$;
5.40                 end
5.41                 if $(z_j(d_j) = 0)$ then break;
5.42                 $\mathbb{D}_j^{'} = \textbf{CutDomainD}[a_j, \textbf{LocalCosts}_j]$;
5.43             end
5.44         end
5.45         $d_j = x \mid \min_{x \in \mathbb{D}_j} lc_j(x) + sc_j(x)$, where $lc_j, sc_j \in \textbf{LocalCosts}_j$;
5.46         $z_j(d_j) = lc_j(d_j) + sc_j(d_j)$;
5.47         $\textbf{SubTreeValues}_j(d_j) = \textbf{SubTreeValues}_j(d_j) \cup \{(a_j, d_j)\}$;
5.48         Send_VIEW_Message($a_j, z_j(d_j), \textbf{SubTreeValues}_j(d_j)$) to $a_i \in P_j^{view}$;
5.49     end
5.50     Receive_TERMINATE($\textbf{SubTreeValues}_1(d_1)$);
5.51     Set channel $d_j = d_m \mid a_j = a_m, (a_m, d_m) \in \textbf{SubTreeValues}_1(d_1)$;
5.52     Send_TERMINATE($\textbf{SubTreeValues}_1(d_1)$) $\forall a_k \in C_j^{view}$;
5.53 end
```

Algorithm 6: Procedure **CutDomainD** for DCAA-O

```
6.1 Procedure CutDomainD[$a_j$, LocalCosts$_j$]
6.2     if $(d_j = 1)$ then $z_{j_a}(d_j) = z_j(d_j)$;
6.3     else if $(z_j(d_j) < z_{j_a}(d_j))$ then $(z_{j_a}(d_j) = z_j(d_j))$;
6.4     forall the $((d_k > d_j)\ in\ \mathbb{D}_j^{'})$ do
6.5         if $(sc_j(d_j) = 0\ and\ lc_k(d_k) \geq lc_j(d_j))$ then Delete $(d_k)$ in $\mathbb{D}_j^{'}$;
6.6         else if $(lc_k(d_k) \geq z_{j_a}(d_j))$ then Delete $(d_k)$ in $\mathbb{D}_j^{'}$;
6.7     end
6.8     Return $\mathbb{D}_j^{'}$;
6.9 end
```

When the root AP has checked all possible d_1 with its descendants, the node selects the lowest subtree cost value $sc_1(d_1)$ and sends a TERMINATE message, containing the optimal allocation solution, to all APs in C_1^{view} through the VIEW graph. From this moment, each AP a_j learns the best value d_j that minimizes the global cost function, also sends a TERMINATE message to all its APs in C_j^{view}, and terminates the algorithm execution.

At each iteration AP a_j receives a sequence of VALUE messages, defined by the VALUE graph. The first message received from a_i in the current iteration (line 5.21) defines the sequence of the expected messages. The set $P_j^{value'} = P_j^{value} \cap A_i'$ (line 5.23) defines the a_j parents in the VALUE graph from which a_j should receive a VALUE message (line 5.24). For example, considering the VALUE graph in Figure 4.1a, the AP a_7 would receive the first message from its parent a_4. Because $P_7^{value} = \{2, 4, 5\}$, $A_4' = \{2, 5\}$ and $P_7^{value'} = \{2, 5\}$, after receiving a VALUE message from AP a_4 (line 5.21), AP a_j should receive VALUE messages from APs a_2 and a_5 (line 5.24). If the first VALUE message came from its parent AP a_5 then the AP a_7 should be prepared to receive VALUE messages from its parents $P_7^{value'} = \{2, 4, 5\} \cap \{2\} = \{2\}$.

4.2.1.2 Numerical Example

Consider the network topology shown in Figure 4.2 with 4 APs and also assume an available set of channels $\mathbb{D} = \{1, 6, 11\}$ and the adjacent channel interference factors in Table 4.1 [1]. We assume that AP a_1 is the root node. The AP a_1 chooses channel 1 as the first value d_1 and sends its value to all linked descendants in $C_1^{value} = \{a_3, a_4\}$, as shown in Figure 4.2a. The AP a_3 synchronously receives the VALUE message from its parent, $P_3^{value} = \{a_1\}$, and sets its **Currentvw**$_3$ $= \{(a_1, 1)\}$. Then a_3 calculates its local costs (*interference factors*) considering its **Currentvw**$_3$ and the cost values defined in Table 4.1. Given that $lc_j(d_j) =$

[1] In this numerical example we consider a different set of adjacent channel interference factors than that shown in Table 3.2, only for the sake of a simpler presentation.

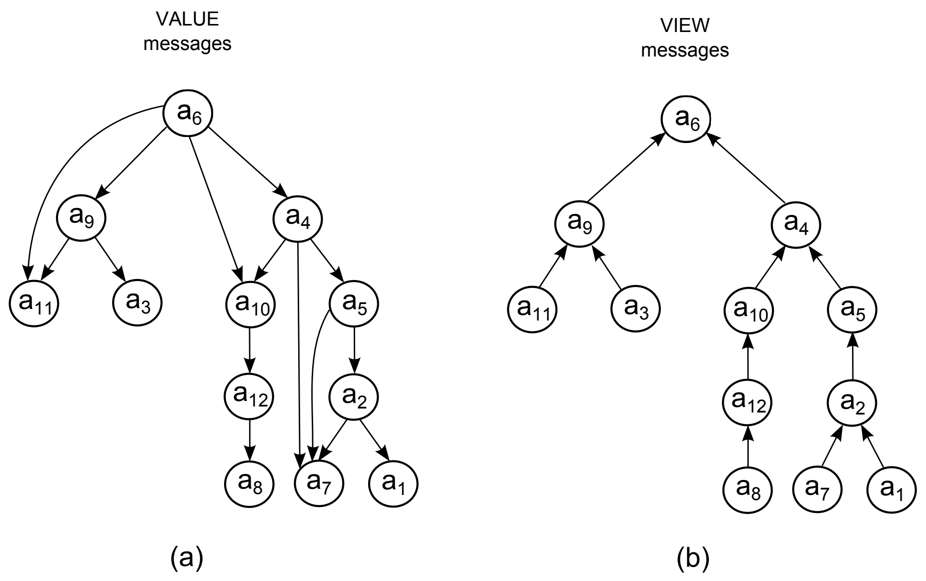


Figure 4.1: Graphs based on the physical model of Figure 3.2, with node a_6 being the root node; (a) Graph of VALUE messages; (b) Graph of VIEW messages

$\sum_{a_i \in \mathbf{Currentvw}_j} f_{ij}(d_i, d_j)$ we have $f_{13}(1,1) = 10000$, $f_{13}(1,6) = 8$ and $f_{13}(1,11) = 0$. These values will be stored in its variable $\mathbf{LocalCosts}_3 = \{[1, 10000, 0], [6, 8, 0], [11, 0, 0]\}$. After calculating its local costs, the AP a_3 chooses $d_3 = 1$, the first option in $\mathbb{D}_3$ and sends this value to all linked descendants in $C_3^{value} = \{a_2, a_4\}$. The AP a_2 receives the VALUE message from its parent $P_2^{value} = \{a_3\}$, stores the received value $d_3 = 1$ in its $\mathbf{Currentvw}_2 = \{(a_3, 1)\}$ and calculates its local costs considering its $\mathbf{Currentvw}_2$. These values are stored in its variable $\mathbf{LocalCosts}_2 = \{[1, 10000, 0], [6, 8, 0], [11, 0, 0]\}$, since we have $f_{32}(1,1) = 10000$, $f_{32}(1,6) = 8$ and $f_{32}(1,11) = 0$. As AP a_2 do not have descendants, it chooses as its current channel choice d_j, the one that minimizes the value of its current local cost, lc_2. In this example, the value $d_2 = 11$ is selected because it is the best choice according to its $\mathbf{Currentvw}_2$, since $f_{32}(1,11) = 0$. After that AP a_2 sends a VIEW message to its VIEW parent, $P_2^{view} = \{a_3\}$. The VIEW message contains the variable $\mathbf{SubTreeValues}_2$ with its selected channel $d_2 = 11$ and the value of its estimated local cost $z_2(d_2) = 0$, as shown in Figure 4.2a.

Table 4.1: Interference Factors for the Numerical Example of DCAA algorithms

Channel Spacing	Overlap Factor
0	10000
5	8
10	0

The same procedure will be executed in parallel at AP a_4. After receiving the VALUE messages from its parents $P_4^{value} = \{a_1, a_3\}$, the AP a_4 stores the values in its $\mathbf{Currentvw}_4 = \{(a_1, 1), (a_3, 1)\}$ and computes its local costs considering its $\mathbf{Currentvw}_4$. These values will be stored in its variable $\mathbf{LocalCosts}_4 = \{[1, 20000, 0], [6, 16, 0], [11, 0, 0]\}$, since we have $f_{14}(1,1) + f_{34}(1,1) = 20000$, $f_{14}(1,6) + f_{34}(1,6) = 16$ and $f_{14}(1,11) + f_{34}(1,11) = 0$. As AP a_4 does not have descendants, it will choose as its current channel choice the one that minimizes the value of its current local cost, lc_4. Then AP a_4 sends a VIEW message to its VIEW parent in $P_4^{view} = \{a_3\}$, with the chosen channel $d_4 = 11$ through its variable $\mathbf{SubTreeValues}_4$ and the value of its estimated local cost $z_4(d_4) = 0$.

Until now, the AP a_3 was synchronously waiting for the VIEW messages of its VIEW children, $C_3^{view} = \{a_2, a_4\}$. The AP a_3 receives all its messages and stores the received values in its variables $\mathbf{SubTreeValues}_3$ and $\mathbf{LocalCosts}_3$. Only after receiving the messages of all its VIEW children, the AP a_3 repeats the same procedure to the next values $\mathbb{D}_3'$, selected by the procedure **CutDomainD**, from its initial domain $\mathbb{D}_3$. The steps for choices $d_3 = 6$ and $d_3 = 11$ are represented in Figures 4.2(b) and 4.2(c), respectively. Then the AP a_3 is ready to choose the value d_3 in its variable $\mathbf{LocalCosts}_3 = \{[1, 10000, 0], [6, 8, 16], [11, 0, 16]\}$ that minimizes its costs $z_3(d_3)$. At this moment the AP chooses $d_3 = 11$, since this channel minimizes the sum of its local cost lc_3 with the cost of its subtrees sc_3, as $z_3 = 0 + 16 = 16$. Then AP a_3 sends a VIEW message to its VIEW parent, $P_3^{view} = \{a_1\}$, as shown in Figure 4.2(c).

The AP a_1 receives the VIEW message of its VIEW children, $C_1^{view} = \{a_3\}$, and continues the procedure execution to its next values in $\mathbb{D}_1'$, since the cost of its subtree is $\neq 0$ ($z_3 = 16$). After testing its channel options in $\mathbb{D}_1'$ and choosing the iteration that indicates the best global cost, *i.e.* the minimum value z_1 in its variable $\mathbf{LocalCosts}_1$, AP a_1 sends a TERMINATE message with its variable $\mathbf{SubTreeValues}_1 = \{[1, (a_2, 1), (a_3, 11), (a_4, 6)]\}$ containing the best channel allocation to all its APs in C_j^{view} through the VIEW graph, which can be shown to be the optimum solution for this scenario.

4.2.1.3 Algorithm Correctness

Correctness of the algorithm follows correctness proof for an asynchronous and distributed algorithm. The DCAA-O algorithm, Algorithm 5, is a synchronous distributed variant of an asynchronous distributed algorithm. During the execution, each AP sends messages at its stable state. The algorithm reaches a stable state when all APs, except the root node, receive a TERMINATE message. At this moment the complete assignment chosen by the APs is equal to the optimal channel assignment. We describe the proof in Appendix A.

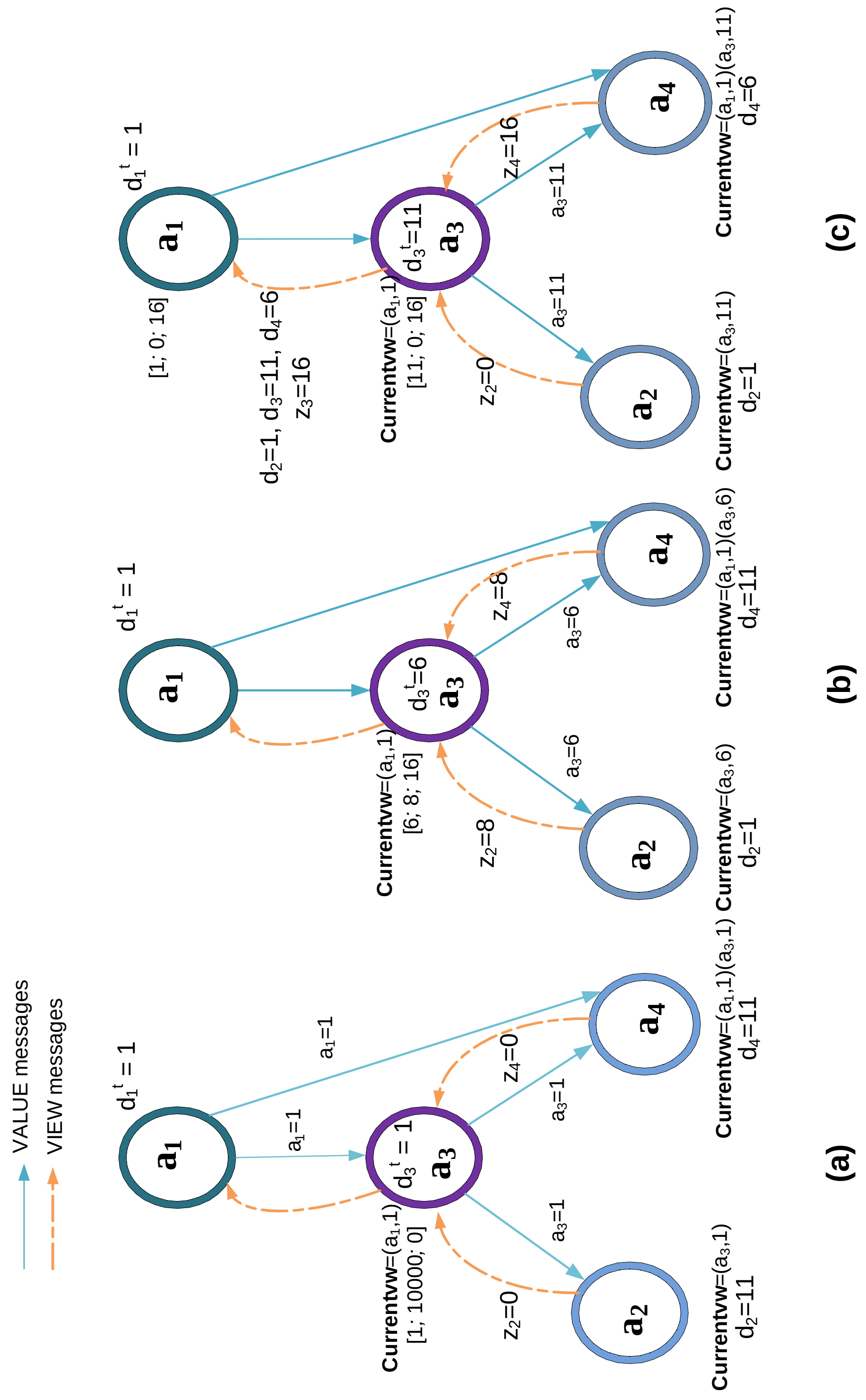


Figure 4.2: Example of DCAA-O execution.

4.2.2 DCAA-S

The DCAA-S algorithm is based on DCAA-O. It considers that each AP a_j, except the root AP, only selects values d_j in $\mathbb{D}_j$ that provide the smallest local cost value $lc_j(d_j) = \sum_{a_i \in \mathbf{Currentvw}_j} f_{ij}(d_i, d_j)$, accounting for the interference received at a_j from its parents in P_j^{value}. This simple strategy modifies the domain $\mathbb{D}_j$ to a reduced domain $\mathbb{H}_j$. The reduced domain $\mathbb{H}_j$ is obtained by each AP executing the additional procedure denoted **CurrentChannelDomain**. The pseudo-code of the procedure **CurrentChannelDomain** is presented in Algorithm 7. The AP a_j must consider the redefined domain $\mathbb{H}_j$ for calculating its local cost from subtree, $sc_j(d_j)$. The DCAA-S algorithm reaches a stable state with a reduced number of exchanged messages, obtaining a global cost value very close to the optimal solution, when not optimal.

The algorithm DCAA-S employs the same pseudo-code of DCAA-O shown in Algorithm 5, with some minor changes:

- lines 5.27 to 5.31 should be replaced by the instruction with the procedure call:

  $\mathbb{H}_j$ = **CurrentChannelDomain**[a_j, **Currentvw**$_j$] described in Algorithm 7;

- in line 5.32 the considered domain $\mathbb{D}_j$ must be replaced by the new domain $\mathbb{H}_j$ that is created by the procedure **CurrentChannelDomain**.

Algorithm 7: Procedure **CurrentChannelDomain** for DCAA-S

```
7.1 Procedure CurrentChannelDomain[$a_j$, Currentvw$_j$]
7.2     forall the ($d_j \in \mathbb{D}_j$) do
7.3         $lc_j(d_j) = \sum_{a_i \in \mathbf{Currentvw}_j} f_{ij}(d_i, d_j)$;
7.4         $sc_j(d_j) = 0$;
7.5         Add[$d_j, lc_j(d_j), sc_j(d_j)$] to LocalCosts$_j$;
7.6     end
7.7     Add $d_j$ to $\mathbb{H}_j$, $\forall\ d_j = x \mid \min_x \mathbf{LocalCosts} = \{[lc_j(x)]\}$;
7.8     Return $\mathbb{H}_j$;
7.9 end
```

The DCAA-O and DCAA-S algorithms were implemented and simulated in a distributed way, using a library of discrete event simulation denoted *SIMPATICA* [81], based on the actors/messages paradigm. According to this paradigm, a simulation model is composed by a set of actors (or tasks) that communicate among them using messages. This library allowed us to simulate a distributed synchronous environment through three kinds of entities: task, queue and message. Each AP was implemented as a task and has three queues for receiving messages, denoted VIEW_QUEUE, VALUE_QUEUE, and TERMINATE_QUEUE, respectively.

4.2.3 Simulation Results

We evaluated the performance of DCAA-O and DCAA-S under different random generated network topologies that represent the constraint graph between APs. We considered as performance metrics the average number of exchanged messages among APs to achieve the best solution and the respective value resulting from the global cost function. As discussed in Section 3.5, the final value reflects the overall interference degradation experienced by the network and consequently a proportional degradation in throughput. The main advantage in using a cost function in the evaluation of algorithms is that this approach allows us to indirectly infer about the network performance without the need of extensive packet level throughput simulations.

4.2.3.1 Evaluation Scenarios

We consider network topologies (constraint graphs) with $|V| = 9$ and $|V| = 16$ nodes. Additionally, for each network topology were considered restriction levels through link densities $\{1, 2, 3, 4\}$ for $|V| = 9$ nodes and $\{1, 2, 3, 4, 5\}$ for $|V| = 16$ nodes. It is possible to represent the restriction levels of a topology as follow:

- The *density* l_d of a graph G = (V,E) measures how many edges are in set E compared to the maximum possible number of edges between vertices in set V. An undirected graph can have at most $|V| \times (|V| - 1)/2$ edges, so the density of an undirected graph is $l_d = 2 \times |E|/(|V| \times (|V| - 1))$, [82];

- The average degree AD of a graph G is another measure of how many edges are in set E compared to number of vertices in set V. Because each edge is incident to two vertices and counts in the degree of both vertices, the average degree of an undirected graph is $AD = 2 \times |E|/|V|$, [82];

We decide to follow the same strategy used by Modi et al. in [74], based on the concept of graph density. The authors consider that a graph with *link density* l_d has $(l_d \cdot V)$ links, where V is the number of vertices in the graph and $l_d \leq (V - 1)/2$.

The number of nodes used to construct the topologies were based on the scalability presented by the algorithms during the simulations. For a given number of nodes, a total of 30 random topologies were generated for each density, and the considered results are the average over these topologies. As a first scenario, we allowed the APs to select the corresponding channel from an available set of three channels: $\{1, 6, 11\}$, *non-overlapping* channels. In the second scenario the APs can select one among all 11 available channels.

We used the interference factors as in Table 3.2. As the simulation results represent sample mean values, we include in the graphs the 90% confidence interval.

4.2.3.2 Performance Analysis

The performance of DCAA-O and DCAA-S was evaluated in terms of the average number of exchanged messages among APs and the final global cost value. As explained in Section 3.6, the LO-A algorithm was used as a benchmark, and was implemented as Algorithm 4. LO-A does not have a deterministic convergence criterion, and is said to converge when it stops improving the cost function. As there is any specific criterion of convergence in [17], we performed a large set of experiments in order to arrive at a reasonable one. We observed that when LO-A does not improve the cost function during at least 50 consecutive iterations, the algorithm can be said to have converged.

Before addressing the performance of the algorithms DCAA-O, DCAA-S and LO-A, we present the improvement of the DCAA-O algorithm that reduces the number of exchanged messages between APs. The values are compared with the exhaustive search method. It is possible to observe in Table 4.2 that the proposed algorithm after the pruning criteria described in Section 4.2.1, presented around 90% less exchanged messages for convergence (topologies of 9 APs and 11 available channels), especially for topologies with lower densities, but still less to topologies with high densities (around 29% less). The same conclusion also applies for scenarios with l6 APs and 3 available channels. It is important to point that the solution optimality was guaranteed for all tested scenarios.

Table 4.2: DCAA-O after implementation of pruning criteria

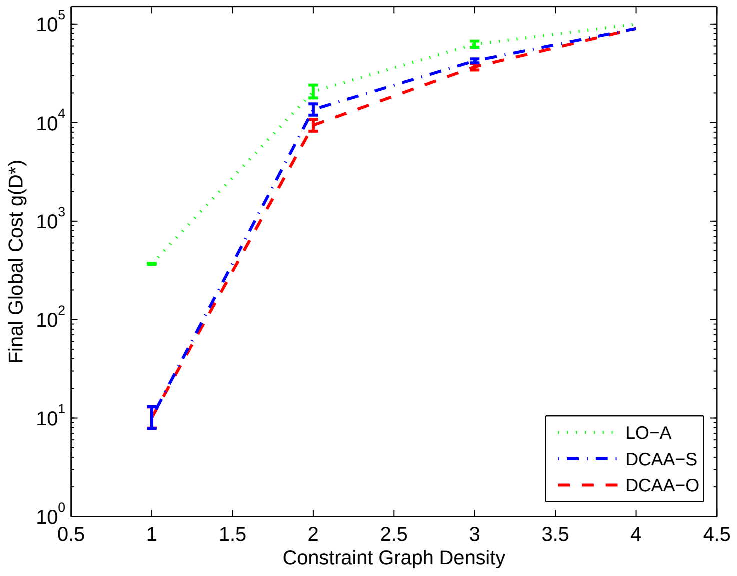
9 APs	3 channels	11 channels
Link Density	Messages Reduction(%)	Messages Reduction(%)
1	66.27	96.46
2	31.41	93.00
3	11.06	58.38
4	5.44	28.64

16 APs	3 channels	-
Link Density	Messages Reduction(%)	-
1	91.18	-
2	61.19	-
3	23.29	-
4	11.84	-
5	9.33	-

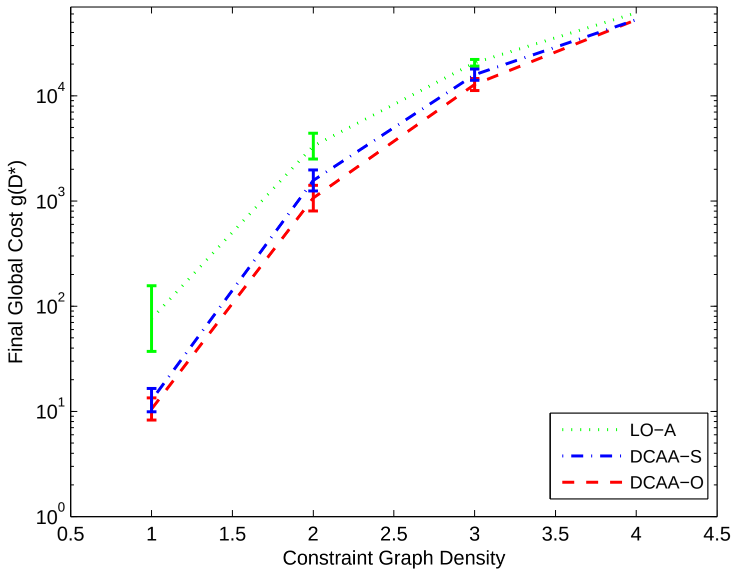
The performance of DCAA-O, DCAA-S and LO-A for random topologies formed by 9 APs, with 3 and 11 available channels, are shown in Figures 4.3 (a) and (b). The values related to DCAA-O and DCAA-S algorithms present the final global cost value given by the global function $z_j(d_j) = lc_j(d_j) + sc_j(d_j)$ where $d_j = x \mid \min_{x \in \mathbb{D}_j} lc_j(x) + sc_j(x)$. The global cost function considers the sum of the local cost $lc_j(d_j)$ of an AP a_j given d_j and its subtree cost $sc_j(d_j) = \sum_{a_k \in C_j^{view}} z_k(d_j)$. The random topologies were generated with link densities ranging from 1 to 4. As we can see from the figures, in both cases DCAA-O presents the smallest final global cost value, and achieves the optimal solution. DCAA-S performs very close to DCAA-O, while LO-A presents the worst performance among them. The difference in performance among the algorithms decreases with the increase in link density (or network connectivity). The LO-A algorithm employs a local coordination strategy, what can lead to a local minimum solution. In a full mesh graph (complete graph) all nodes directly get consensus when a specific node takes a decision (channel change). Therefore, based on this observation, when the graph connectivity increases, the LO-A solution approaches the optimal one because the possibility to converge to a local minimum is greatly reduced.

Figures 4.4 (a) and (b) consider the same scenarios as in Figures 4.3 (a) and (b), but the comparison is in terms of the average number of exchanged messages among APs. As we can see from the figures, DCAA-O demands the largest amount of exchanged messages. DCAA-S, through its heuristic, demands much less messages exchanges than DCAA-O, the difference being of some orders of magnitude. DCAA-S also demands less messages exchanges than LO-A. It is important to note that DCAA-S, while being clearly the algorithm that demands the least messages exchanges in these scenarios (9 APs, with 3 or 11 available channels), also achieves a final global cost very close to the optimum one achieved by DCAA-O. Therefore, such reduction in message exchanges does not come at the cost of a relevant decrease in performance.

For the random topologies formed by 16 APs, we compare only the performance of DCAA-S and LO-A, since DCAA-O has a considerably high cost in terms of exchanged messages for this scenario. Moreover, as the network has more APs, we consider link densities ranging from 1 to 5. Figure 4.5 shows the final global cost achieved by each algorithm, where 3 and 11 channels are available. As we can see, DCAA-S always outperforms LO-A. Figure 4.6 presents the average number of exchanged messages for each algorithm. For 3 available channels DCAA-S requires much less messages than LO-A (more than a order of magnitude less). For the case of 11 available channels and a weakly connected network (link densities 1 and 2), LO-A required less exchanged messages. However, for a more connected network (link densities 3, 4 and 5), where the complexity of the channel allocation problem increases, DCAA-S required considerably less messages exchanges than LO-A.



(a) 9 APs, 3 channels

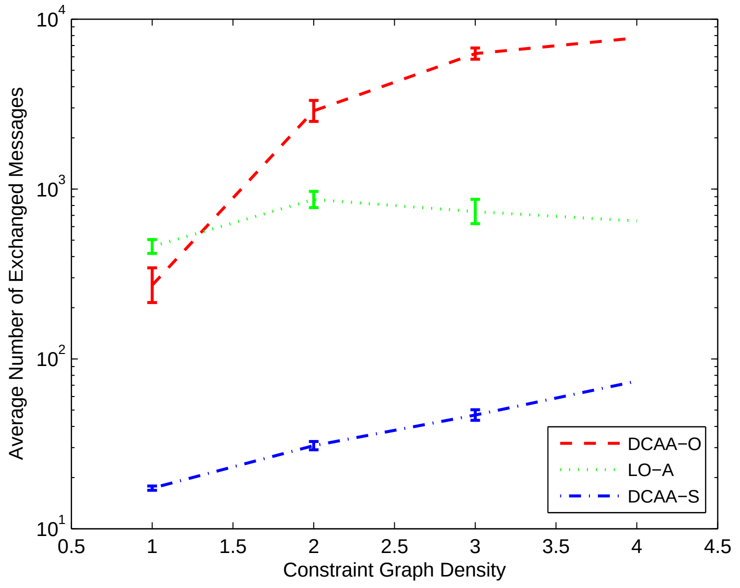


(b) 9 APs, 11 channels

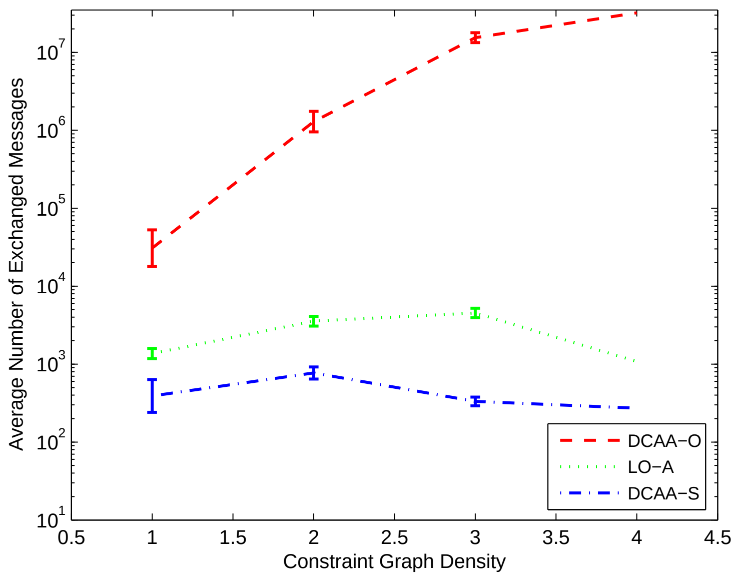
Figure 4.3: Final global cost for LO-A, DCAA-S, and DCAA-O.

4.3 DOCA Algorithm

In Section 4.2 we proposed a new algorithm to solve the channel assignment problem where the solution was derived from a modified version of a polynomial space algorithm



(a) 9 APs, 3 channels



(b) 9 APs, 11 channels

Figure 4.4: Average number of exchanged messages for DCAA-O, LO-A and DCAA-S.

for DCOP named ADOPT. The algorithm, called DCAA (*Distributed Channel Assignment Algorithm*), is able to reach the optimal (DCAA-O) or sub-optimal (DCAA-S) solution for

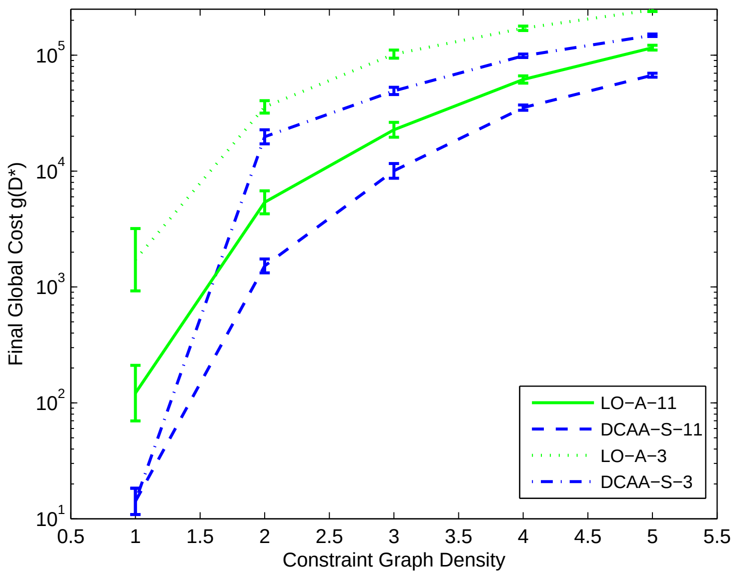


Figure 4.5: Final global cost value for DCAA-S and LO-A (16 APs, 3 and 11 channels)

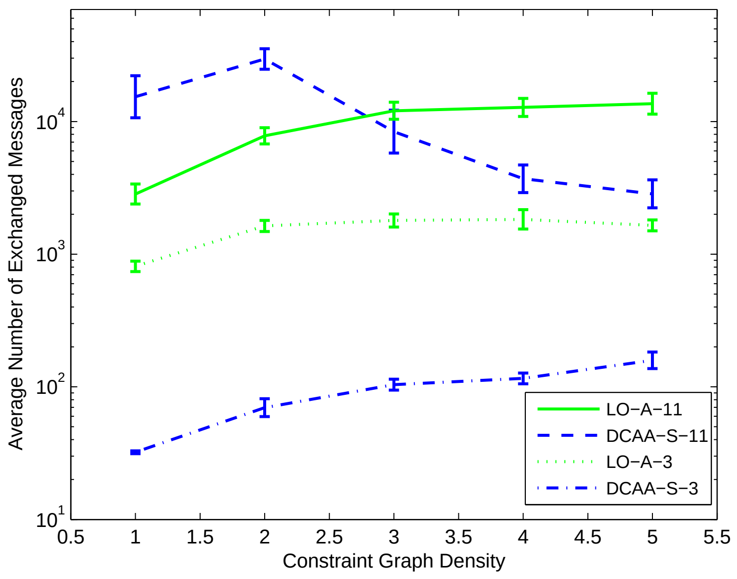


Figure 4.6: Average number of exchanged messages for DCAA-S and LO-A (16 APs, 3 and 11 channels)

the channel assignment problem. However, the ADOPT-based strategy presents a drawback concerning to the total number of exchanged messages, especially for more connected topologies. In order to improve the performance of the ADOPT-based strategy, we present in this section a cooperative channel allocation strategy using an adaptation of the algorithm DPOP [41, 63, 69]. DPOP is an algorithm based on dynamic programming that operates on a variable ordering given by a DFS arrangement of the constraint graph [41]. One of its main advantage is that DPOP requiring only a linear number of exchanged messages, reduces exponentially the network overhead in relation to ADOPT. However its complexity is related to the size of its UTIL messages that is bounded exponentially by the induced width of the chosen DFS ordering. The goal is to find a best channel allocation solution and reduce the number of control messages exchanged between APs, presented by the DCAA algorithm, to improve the overall network performance.

The algorithm is named DOCA (*Distributed Optimal Channel Assignment*). DOCA algorithm uses the same basic notation of the DPOP algorithm for parent, pseudo-parents and child nodes. It starts by electing a leader (a root node) through a leader election algorithm as in Algorithm 1. After choosing a leader, DOCA executes the three phases: DFS tree creation, UTIL propagation and VALUE propagation. Its concept of optimality differs from DPOP. The utility propagation value considered by DPOP algorithm to reach the global optimal solution was changed by the minimum *expected* cost with no changes in DPOP's logical methods shown in Chapter 3.

The parameters listed below are also based on the description of DPOP [41, 63, 69], however some changes were necessary for the development of DOCA algorithm:

- Varibales:
  - $\mathbb{A} = \{a_1, a_2, \ldots, a_N\}$: set of APs;
  - a_j: Current access point;
  - a_1: Root access point;
  - $X = \{x_1, x_2, \ldots, x_l\}$: APs's variables;
  - $\mathbb{D} = \{d_1, d_2, \ldots, d_l\}$: AP's values (available channel set);
  - P_j: parent of a node a_j, the single node above in the hierarchy of the pseudotree that is directly connected to a_j through a tree-edge;
  - C_j: the children of a node a_j, the set of nodes below in the pseudotree that are directly connected to a_j through tree-edges;
  - PP_j: the pseudo-parents of a node a_j, the set of nodes above in the pseudotree that are directly connected to a_j through *back-edges*;
  - PC_j: the pseudo-children of a node a_j, the set of nodes below in the hierarchy of the pseudotree that are directly connected to a_j through *back-edges*;

  - Sep_j: the separator of node a_j, all ancestors of a_j which are connected with a_j or with descendants of a_j.
  - lc_j: local cost for all possible set of combinations between a_j and Sep_j, considered as utility;;
  - **LocalView**$_j$: a local variable of an AP a_j, contains all possible set of combinations between a_j and Sep_j, and their respective local cost lc_j;
  - D^*: Best set of assigned channels for nodes in A that produces the lowest *estimated lower bound cost* value for the whole problem;
  - **AgentView**: the optimal set of values a node a_j sends to its child a_k;
  - $UTIL_i^{P_j}$: the *UTIL* message sent by a_j to its parent agent P_j;
  - $VALUE_{P_j}^{a_j}$: the *VALUE* message a_j receives from its parent P_j.

- Hypercube: set of values in a UTIL message, composed of minimal local cost values obtained in the subtree for each element of Sep_j. Thus, messages are exponential in the separator size (bounded by the induced width).

- Operators created for DOCA algorithm:
  - USelectMIN(R) - **UTIL SELECTION**: takes a relation R where each n-tuple corresponds to one line of the hypercube, including the local cost, and extracts different combinations with the lowest local cost;
  - Min (R, x): takes a relation R and a specified element $x \in R$, and yields a new relation that consists of the tuples where value of x is the minimum;

Let $S_j = D^K$ and $K = |a_j \cup Sep_j| + 1$, the operation USelectMIN is given by

$$\text{USelectMIN}(\textbf{LocalView}_j) = \cup_{s \in S_j} \text{Min}\Big(textSelection\Big(\textbf{LocalView}_j, (d_2, \cdots x_K) = s\Big), x_{K+1}\Big). \tag{4.1}$$

- Functions:

$$\text{BestValue}(\textbf{LocalView}_j, D^*, Sep_j) \tag{4.2}$$

allows to select the best value for a_j in the variable **LocalView** corresponding to the optimal assigned values in D^* for the elements in Sep_j, and

$$\text{ChooseValue}(D^*, Sep_k) \tag{4.3}$$

chooses values in D^* corresponding to elements of Sep_k.

Following we present a simple numerical example to help understanding of the DOCA algorithm, presented in Algorithm 8.

4.3.1 Numerical Example of DOCA Algorithm

The problem considers a DFS tree with 4 APs, as can be seen in Figure 4.7. It is assumed the available set of channels $\mathbb{D} = \{1, 6, 11\}$ and interference factors as in Table 4.3[2]. The AP a_1 is considered the root node.

The algorithm starts by the root node initializing the execution of a DFS algorithm as in [41]. After execution of the DFS algorithm, a node a_j knows its parent P_j, pseudo-parents PP_j, children C_j, pseudo-children PC_j and its separator Sep_j, well defined in Section 3.4.

The leaf nodes a_2 and a_3 starts the execution of phase two, *UTIL* propagation phase. The details of the values set constructed during this phase, for these nodes, are shown in Figure 4.7 (a) and (b). According to the *UTIL* propagation phase, each leaf node creates a relation R_j with its parent an pseudo-parents. In this example the leaf nodes do not have pseudo-parents, so a_2 creates a relation R_2 and a_3 creates a relation R_3 with their parent a_4. During the construction of their relations it is calculated the local cost for all the tuples $\{(x_2, x_4)|(x_2, x_4) \in D^{P_2+1}\}$ and $\{(x_3, x_4)|(x_3, x_4) \in D^{P_3+1}\}$. These values are respectively stored in their local variables **LocalView**$_2$ and **LocalView**$_3$. Then considering the set of values in their **LocalView**, the leaf nodes execute the operation USelectMIN, to choose combinations with the lowest cost, with their parent. Finally they execute the operation *projection* ($\perp$), to eliminate themselves out. The resulting set of values represents the best channel options in relation to their domains $\mathbb{D}_2$, $\mathbb{D}_3$ and the domain $\mathbb{D}_4$ of their parent. Then APs a_2 and a_3 send to their parent their respective $UTIL_{a_2}^{a_4}$ and $UTIL_{a_3}^{a_4}$ messages. The values in both UTIL messages, represent the best channel options, those with the lowest cost for the APs' parent.

AP a_4 waits until it receives the UTIL messages from its children. It calculates the relation R_4 with its parent a_1. Operations of this node are presented in Figure 4.7 (c). After receiving the UTIL messages from its children, the AP executes the operation join ($\oplus$) between its relation R_4 and the received UTIL messages. The result set of values are stored in its **LocalView**$_4$. It is important to call attention that it is executed the sum of the local cost values, to the same value assumption of a_4 in its relation R_4, and in the UTIL messages $UTIL_{a_2}^{a_4}$ and $UTIL_{a_3}^{a_4}$. As an example, when AP a_4 assumes the value 6 from its domain, for the tuple $\langle 1, 6\rangle$ the local cost in its **LocalView**$_4$ will be (10+10+10)=30. Once the variable **LocalView**$_4$ is constructed, the next operations are the same as the leaf nodes (USelectMIN and PROJECTION). Then AP a_4 sends its UTIL message $UTIL_{a_4}^{a_1}$ to

[2]In this numerical example we consider a different set of interference factors than that shown in Table 3.2, only for the sake of a simpler presentation.

its parent a_1.

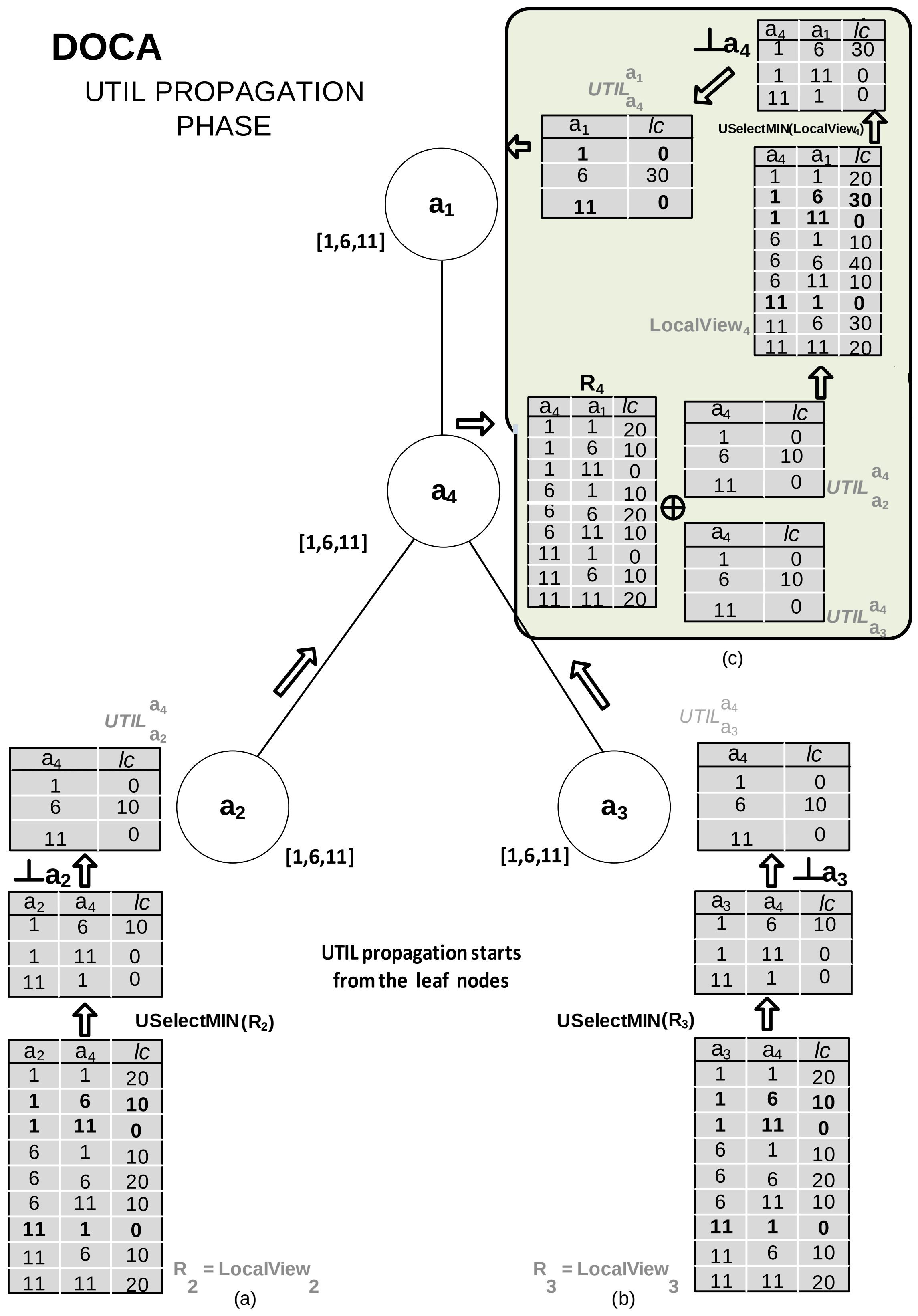
The root AP a_1, receives the UTIL message from AP a_4. As it does not have a parent it can immediately chooses its value d_1, that produces the minimum cost for the whole channel assignment problem, $d_1 = \{1\}$, shown in Figure 4.8 (d). The values $d_1 = \{1\}$ or $d_1 = \{11\}$ may be chosen by the AP a_1 since their local cost are the same, equal to zero, in the example. The algorithm always assume the first option. Then AP a_1 initiates the execution of the third phase of the DOCA algorithm, *VALUE* propagation phase. The AP, via a message $VALUE_{a_1}^{a_4}(\{1\})$, informs its child a_4, of its choice. Then AP a_4 restores from its variable $\mathbf{LocalView}_4$, the value d_4 for itself, that presents the best local cost when its parent a_1 assumes the value $d_1 = \{1\}$, Figure 4.8 (e) point this computation. The AP a_4 chooses as its best value $d_4 = \{11\}$. The process continues with a_4 sending the messages, $VALUE_{a_4}^{a_2}(\{11\})$ and $VALUE_{a_4}^{a_3}(\{11\})$ to its children a_2 and a_3. After receiving their parent best assumption, nodes a_2 and a_3 restore their optimal values for $a_4 = \{11\}$, i.e a_2 assumes for itself the value $d_2 = \{1\}$ and a_3 assumes $d_3 = \{1\}$, Figure 4.8 (f) and (g). Then the algorithm finishes with the optimal solution $a_1 = \{1\}, a_2 = \{1\}, a_3 = \{1\}, a_4 = \{11\}$ presenting the final global cost equal to zero.

Table 4.3: Interference Factors for the Numerical Example of DOCA Algorithm

Channel Spacing	Overlap Factor
0	20
5	10
10	0

4.3.2 Simulation Results

The simulations evaluated the performance of LO-A, DCAA and DOCA algorithms, concerning to scenarios of random topologies with $|V| = 9$ and $|V| = 10$ nodes with each AP having average degree $AD = 4$, $AD = 6$ and $AD = 8$. As ADOPT-based strategy, the number of nodes used to construct the topologies of the DPOP-based scenarios were based on the scalability presented by the algorithms during the simulations. We assume that each AP can select one among all 11 available channels. The comparison metrics are the number of exchanged messages to achieve convergence and the final global cost. The convergence criterion of LO-A algorithm was the same used in Section 4.2. For DCAA algorithms it was used the DCAA-S algorithm since it includes a change that contributes to efficiency, where the number of exchanged messages grows only linearly. The simulations of DOCA were done using FRODO platform [62] integrated with Matlab®. FRODO is an open-source multithread framework for distributed optimization. The wireless network topologies and the corresponding interference constraint graphs were created using Matlab.



(a) R_2 = LocalView$_2$

a_2	a_4	*lc*
1	1	20
1	**6**	**10**
1	**11**	**0**
6	1	10
6	6	20
6	11	10
11	**1**	**0**
11	6	10
11	11	20

USelectMIN(R_2)

a_2	a_4	*lc*
1	6	10
1	11	0
11	1	0

$\perp a_2$ → $UTIL_{a_2}^{a_4}$

a_4	*lc*
1	0
6	10
11	0

(b) R_3 = LocalView$_3$

a_3	a_4	*lc*
1	1	20
1	**6**	**10**
1	**11**	**0**
6	1	10
6	6	20
6	11	10
11	**1**	**0**
11	6	10
11	11	20

USelectMIN(R_3)

a_3	a_4	*lc*
1	6	10
1	11	0
11	1	0

$\perp a_3$ → $UTIL_{a_3}^{a_4}$

a_4	*lc*
1	0
6	10
11	0

(c) R_4

a_4	a_1	*lc*
1	1	20
1	6	10
1	11	0
6	1	10
6	6	20
6	11	10
11	1	0
11	6	10
11	11	20

$\oplus$ $UTIL_{a_2}^{a_4}$

a_4	*lc*
1	0
6	10
11	0

$UTIL_{a_3}^{a_4}$

a_4	*lc*
1	0
6	10
11	0

LocalView$_4$

a_4	a_1	*lc*
1	1	20
1	**6**	**30**
1	**11**	**0**
6	1	10
6	6	40
6	11	10
11	**1**	**0**
11	6	30
11	11	20

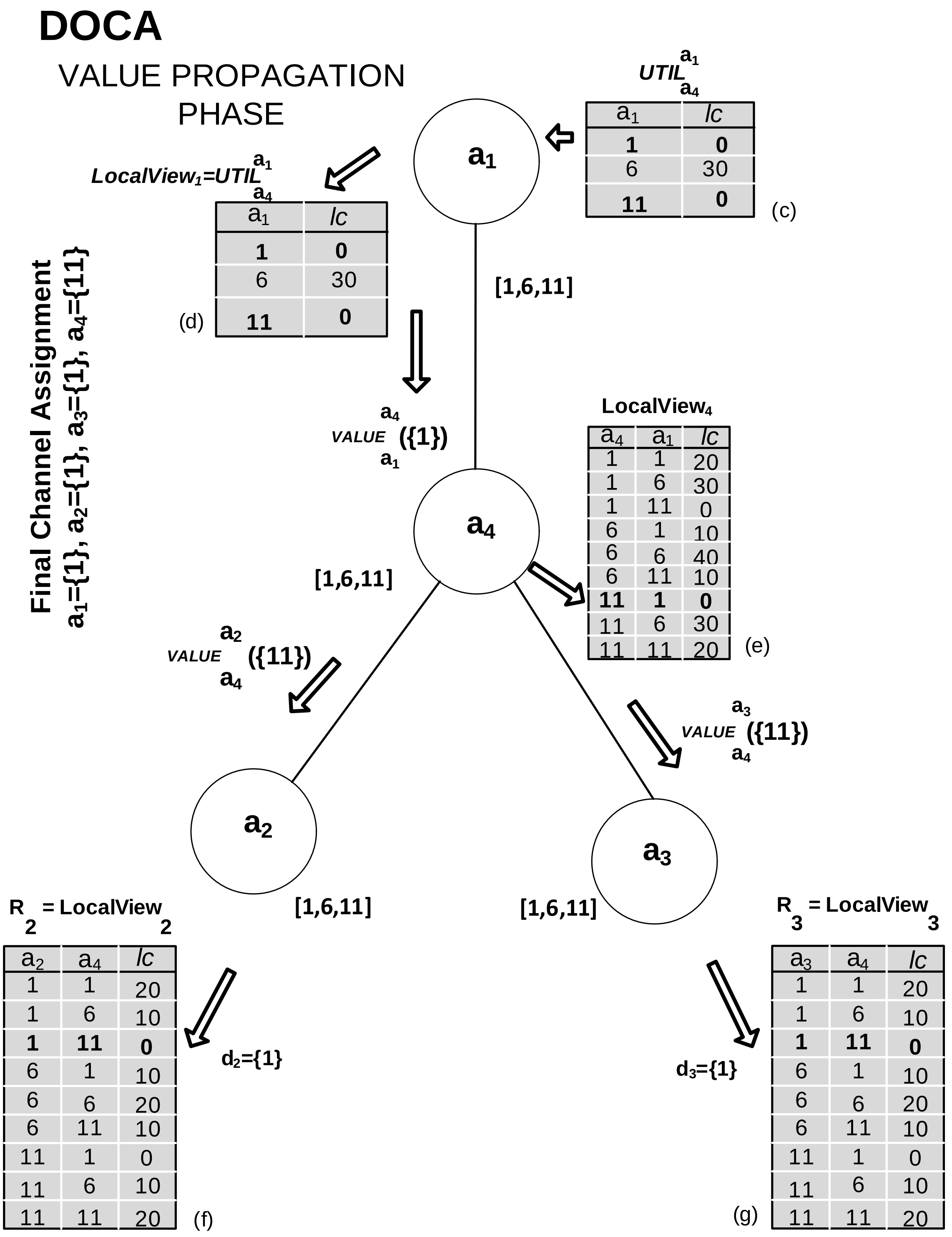
USelectMIN(LocalView$_4$)

a_4	a_1	*lc*
1	6	30
1	11	0
11	1	0

$\perp a_4$ → $UTIL_{a_4}^{a_1}$

a_1	*lc*
1	**0**
6	30
11	**0**

Figure 4.7: A DFS tree. Relations R_2, R_3 and R_4 are presented in (a), (b) and (c); Projections of a_2 and a_3 out of their relations with a_4 (a) and (b); In (c) a_4 joins the UTIL messages from a_2 and a_3 with its relation R_4; Node a_4 projects itself out, and sends $UTIL_{a_4}^{a_1}$ to a_1 in (c).



(c) $UTIL_{a_4}^{a_1}$

a_1	*lc*
1	**0**
6	30
11	**0**

(d) $LocalView_1 = UTIL_{a_4}^{a_1}$

a_1	*lc*
1	**0**
6	30
11	**0**

(e) $LocalView_4$

a_4	a_1	*lc*
1	1	20
1	6	30
1	11	0
6	1	10
6	6	40
6	11	10
11	**1**	**0**
11	6	30
11	11	20

(f) $R_2 = LocalView_2$

a_2	a_4	*lc*
1	1	20
1	6	10
1	**11**	**0**
6	1	10
6	6	20
6	11	10
11	1	0
11	6	10
11	11	20

(g) $R_3 = LocalView_3$

a_3	a_4	*lc*
1	1	20
1	6	10
1	**11**	**0**
6	1	10
6	6	20
6	11	10
11	1	0
11	6	10
11	11	20

Figure 4.8: The root node a_1 sends a VALUE message $VALUE_{a_1}^{a_4}$ to its child a_4 (d); In (e) a_4, based on the best value of its parent, chooses its best value in $LocalView_4$; Node a_4 sends VALUE messages to its children a_2 and a_3 (e); Nodes a_2 and a_3 after receiving the VALUE message from their parent a_4 choose their best values (f) and (g).

Algorithm 8: DOCA - Distributed Optimal Channel Assignment

```
8.1  Procedure DOCA($A, D, \mathbb{R}$):
8.2      $\forall a_j \in A$;
8.3      PHASE 1: DFS arrangement - run token passing mechanism as in [41];
8.4      After execution, $a_j$ knows $P_j, PP_j, C_j, PC_j, Sep_j$ ;
8.5      PHASE 2: UTIL propagation (bottom-up UTIL message propagation starts from the
         leaf nodes)
8.6      $D^* = \varnothing$; $R_j = null$;
8.7      LocalView$_j$ = $null$;
8.8      Compute $R_j$; // a sub set with all combination values between $a_j$ with $P_j$ and $PP_j$, and the
         computation of $lc_j$
8.9      forall the $a_k \in C_j$ //* for all children of $a_j$ - if $a_j$ is a leaf node, skip this */ do
8.10         ReceiveUTILMessage ($Sep_k, UTIL_k^j$); // wait for $UTIL$ message arrive from $a_k$
8.11         LocalView$_j$ = LocalView$_j \oplus UTIL_k^j$; //Join UTIL messages from $C_j$ as they arrive
8.12     end
8.13     if ($P_j == \varnothing$) // that means that $a_j$ is the root node then
8.14         $D^* \leftarrow$ Min(LocalView$_j, d_2$)$\perp d_2$; // Select best value $d_1$ for $a_j$ from minimum local cost
             values $d_2$
8.15         Send VALUE ($D^*$) to all $C_j$
8.16         else
8.17         LocalView$_j$ = LocalView$_j \oplus R_j$; //also join relations with parent/pseudo-parents;
8.18         $UTIL_j^{P_j}$ = USelectMIN(LocalView$_j$)$\perp a_j$ //use USelectMIN to select the tuples with
             minimum local cost and $\perp$ to eliminate self out of message to parent;
8.19         SendUTILMessage ($Sep_j, UTIL_j^{P_j}$);
8.20     end
8.21     PHASE 3: VALUE propagation (top-down VALUE message propagation)
8.22     wait for $VALUE_{P_j}^j(D^*)$ message from $P_j$; // the optimal assignment for the elements in $Sep_j$
8.23     $D^* \leftarrow$ BestValue(LocalView$_j, D^*, Sep_j$); // Selects best $a_j$ value in LocalView
         corresponding to optimal assigned values $D^*$ for the elements in $Sep_j$
8.24     forall the $a_k \in C_j$ /*for all children of $a_j$ - if $a_j$ is a leaf node, skip this*/ do
8.25         AgentView= $\varnothing$;
8.26         AgentView $\leftarrow$ ChooseValue ($D^*, Sep_k$); // chooses values in $D^*$ corresponding to
             elements of $Sep_k$
8.27         Send $VALUE_{P_j}^k$(AgentView) to all $C_j$;
8.28     end
8.29 end
```

For each scenario a set of 100 topologies, created randomly, were considered. The results represent average values, presenting negligible variation when considering 90% confidence interval.

4.3.2.1 Performance Analysis

Considering the evaluated scenarios, Table 4.4 presents the average final global costs. It is possible to observe that the LO-A and DCAA-S algorithms converge sometimes to a local minimum. Only DOCA algorithm converged to an optimal solution in 100% of the simulations. The DCAA algorithm has shown good performance in terms of convergence, when compared to LO-A, but not always converging to an optimal solution.

Table 4.4: Average Final Global Cost

	9 APs			10 APs		
	AD=4	AD=6	AD=8	AD=4	AD=6	AD=8
LO-A	0.339	2.074	6.166	0.339	1.956	5.967
DCAA-S	0.199	1.510	5.550	0.179	1.407	4.345
DOCA	0.038	1.282	5.291	0.039	0.369	3.408
Optimal	0.038	1.282	5.291	0.039	0.369	3.408

The difference between the final global cost values presented by LO-A and DCAA algorithms, in relation to that presented by DOCA algorithm, is pronounced for more connected graphs (9 and 10 APs with AD=6 and AD=8). To this cases the complexity of the channel allocation problem increases.

Figure 4.9 presents the analysis of the number of exchanged messages to achieve convergence. It is important to point out that DOCA presents a number of exchanged messages about one order of magnitude lower than LO-A and DCAA, while achieving optimal solution.

Next we investigate the scalability of DOCA, considering as principal metrics, the number and the size of the exchanged messages to DFS, UTIL and VALUE phases. Figure 4.10 presents the required number of exchanged messages for different topology sizes and degrees. It should be noted that for control messages (UTIL+VALUE), the algorithm scales linearly with the number of APs and does not depend on the average degree of the topology. This behavior is based on the definition of the algorithm DPOP where the number of *UTIL* and *VALUE* messages are directly related to the number of agents n in the problem. DOCA algorithm follows the same principle. For DFS arrangement the algorithm does not present the same number of messages, for different degrees of connectivity, showing that this phase depends on the number of edges which in turn is related to the number of

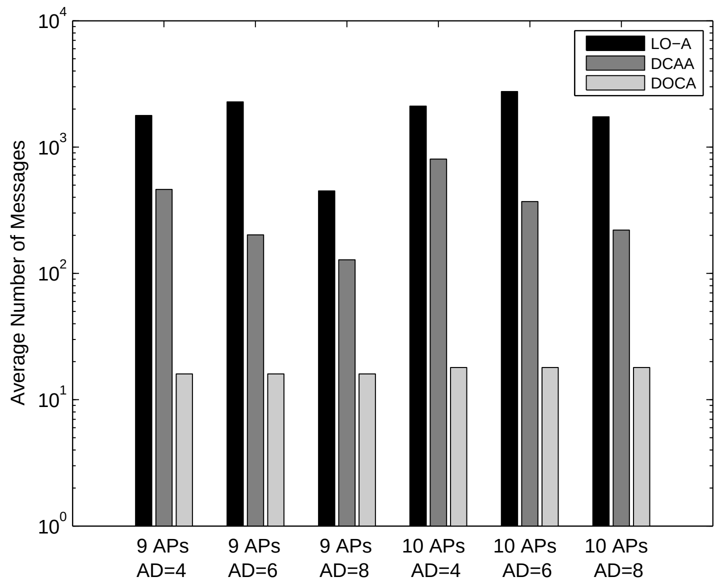


Figure 4.9: Average number of exchanged control messages.

nodes and average degree of the topology. Due to this characteristic, we decided to present an DFS algorithm where the number of exchanged messages between the APs, during the construction of the ordered tree, depends exclusively on the number of vertices $|V|$ of the network . This algorithm is presented in the next section.

Figure 4.11 presents the total number of bytes of the DFS+UTIL+VALUE messages needed to be exchanged by the protocol. It is possible to observe the variation of the total message sizes, whose maximum value depends on the width of the pseudotrees chosen by the algorithm. For the case with 20 nodes and $AD = 4$, it is required the exchange of approximately 24KBytes. However for more complex topologies, the performance of the algorithm is compromised, limiting its scalability. In an attempt to solve this problem we developed the algorithm DSCA, that includes a heuristic to control the size of the hypercube processed by an AP and consequently the size of its UTIL message.

4.4 DSCA Algorithm

Based on the scalability problem presented by DOCA algorithm for topologies greater than 20 nodes, we developed an algorithm denoted *Distributed SubOptimal Channel Assignment* (DSCA). The algorithm is based on the algorithm DOCA, having as its main goal the control of the UTIL messages size. As already mentioned, the UTIL messages size are exponentially bounded by the induced width of the DFS ordering chosen. Through DSCA

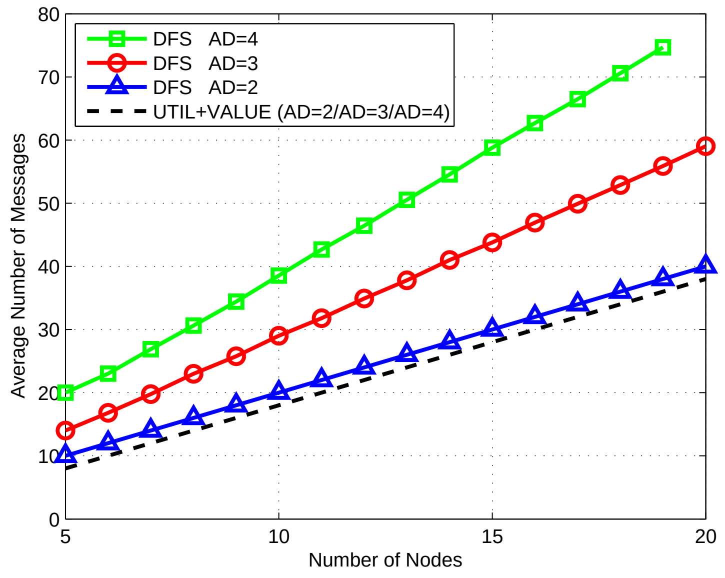


Figure 4.10: Number of exchanged messages.

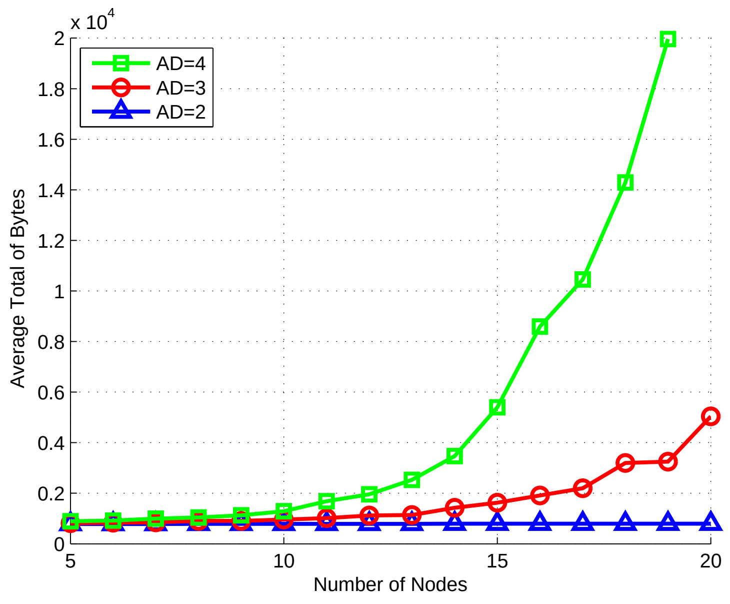


Figure 4.11: Number of exchanged bytes.

algorithm we present a heuristic to control the size of the hypercube processed in an AP and consequently the size of UTIL message each AP sends to its parent, thus ensuring linearity concerning to AP's local processing.

By analyzing previous results, regarding to the number of exchanged messages during the DFS tree creation (e.g. Figure 4.10), we decided to use a new DFS method. The idea was to use a new strategy that was not directly related to the degree of the network. So, inspired by the method presented by Makki and Havas [56], that requires $2|V| - 2$ exchanged messages to create a pseudotree, we developed an improved algorithm termed Ddfs-Improved. The algorithm Ddfs-Improved and the proposed DSCA-heuristic are described below.

4.4.1 Ddfs-Improved Algorithm

To generate a DFS tree by a method that is not dependent on the degree of the network, we developed a distributed DFS tree algorithm denoted Ddfs-Improved. The algorithm is based on the algorithm of Makki and Havas [56]. Our improvement is related to the capacity of each node discover their parents, pseudo-parents and children. Essential information to the execution of the DSCA algorithm. As [56] we used FORWARD and RETURN messages and dynamic backtracking. Each node has a copy of the procedure Ddfs-Improved, whose pseudocode is given in Algorithm 9. For understanding the pseudocode some variable definitions are necessary:

- *split point*: a node, when visited, has two or more unvisited neighbors or it is the root node. Nodes which are not split points may be bypassed by RETURN messages.
- messages: our messages comprise five components: message originator, message type (FORWARD or RETURN), the set *visited* of visited nodes, *splitpoint*, the previous *splitpoint* and the set *returnpath* of nodes from whom a node has received RETURN messages.

Comparing with the messages used in [56] the extra information is the set *returnpath* included as a component of the RETURN message. This component was included as a strategy to find the pseudo-parents of a node, as can be seen in Algorithm 9 (lines 9.40, 9.53 and 9.54). We used the same theoretic notation and avoided considerations of nonessential details of the authors implementations in [56].

Each node that receives a message executes the Ddfs-Improved algorithm. When a node receives a FORWARD message, it executes the initialization code for local variables. Thus, in each case, a FORWARD message is sent to a neighbor (if there is an unvisited neighbor connected to this node), else a RETURN message is sent to an ancestor (if this node is not a root). At the end the algorithm is able to find the pseudo-parents of a node by excluding from the variable *neighbors*: the parent, *childset* and the nodes in the set of *returnpath*. Then algorithm terminates. The Ddfs-Improved procedure constructs

Algorithm 9: Ddfs-Improved

```
Procedure
    Initialize
    $neighbors, jchildset, pseudo-parents, parent, returnnode, unvisited, i, D1, D2, D3$; {local variables}
    $messagetype, originator, visited, splitpoint, returnpath$; {message variables}
    begin
    if $messagetype = FORWARD$ then
        {node initialization}
        $visited \leftarrow visited \cup \{j\}$; {this node is visisted}
        $parent \leftarrow originator$;
        $childset \leftarrow \varnothing$;
        $D1_j \leftarrow \varnothing, D2_j \leftarrow \varnothing, D3_j \leftarrow \varnothing, pseudo-parents_j \leftarrow \varnothing$;
        $returnpath \leftarrow \varnothing$;
        $returnnode \leftarrow splitpoint$; {save preferred return address}
        {initialization complete}
    end
    $unvisited \leftarrow neighbors - (visited \cap neighbors)$;
    if $messagetype = RETURN$ then
        $returnpath_j \leftarrow returnpath_j \cup returnpath_k$;
    end
    if $unvisited \neq \varnothing$ then
        {unvisited neighbors remain};
        $i \leftarrow$ Next($unvisited$); {get next unvisited neighbor}
        $unvisited \leftarrow unvisited - \{i\}$; {remove it from unvisited}
        $childset \leftarrow childset \cup \{i\}$; {add it to childset}
        if $unvisited \neq \varnothing$ then
            {this node $j$ is a split point};
            $splitpoint \leftarrow j$ {we need return to this point};
            else
            {This node j is not a split point }
            $splitpoint \leftarrow returnnode$; {we will try to return to an ancestor}
        end
        issue FORWARD message to node i;
        else
        {time backtrack};
        if $j = parent$ then
            stop; {j is the root};
            else
            {we need to backtrack};
            $returnpath_j \leftarrow returnpath_j \cup \{j\}$;
            if $returnnode \in neighbors$ then
                {appropriate ancestor is a neighbor};
                issue RETURN message to node $returnnode$;
                else
                {previous split point was not adjacent};
                issue RETURN message to node $parent$;
            end
        end
    end
    end
    $D1_j = neighbors_j - parent_j$; {remove parent from neigbors}
    $D2_j = D1_j - childset$; {remove childset}
    $D3_j = D2_j - returnpath_j$; {remove returnpath}
    $pseudo-parents_j = D3_j$;
end
```

the depth-first search tree for the distributed methods following the order that nodes are visited during the execution of the algorithm. The DFS-tree is stored in a standard way: at termination the root node is informed and each node knows its parent, children and pseudo-parents in the tree. Such information is stored locally. Each node has local variables *parent*, *childset* and *pseudo-parents* for storing the DFS-tree, plus a variable *unvisited* for storing its unvisited neighbors. Each node also knows its set of *neighbors*. The process starts with the root node giving to itself a FORWARD message containing initial values of $\varnothing$ for *visited* and *returnpath* and itself for *originator* and *splitpoint*. We do not considered this message in the total amount of exchanged messages. We assume that no link or process failure occurs during the execution of the algorithm.

With Ddfs-Improved algorithm we improve over the previous Makki and Havas [56] algorithm the capacity of each node discover its complete tree (parent, pseudo-parents and children) at the expense of a small increase on RETURN message length. Since the execution of the algorithm is sequential it is suitable for synchronous and asynchronous methods.

4.4.2 DSCA Heuristic

To advance in relation to the scalability problem presented by DOCA algorithm, it was necessary to create a heuristic to control local processing, as well as the size of the UTIL message to be sent by a child to its parent. DSCA (*Distributed SuOptimal Channel Assignment*) heuristic by limiting the size of the hypercube to be processed in the UTIL propagation phase, guaranties the condition to improve a scalable solution. The used criteria are:

- Use a maximum size *Utildim* (a configurable parameter) to limit the size of the hypercube.

- Consider from all possible value combinations between a node a_j and its Sep_j, only tuples $(d_1, d_2, \cdots d_{K+1}, lc)$, where $K = |a_j \cup Sep_j| + 1$, that provides local cost lower or equal a established cost limit (also configurable). A *Utildim* number of selected tuples are stored in a local variable called $\mathbf{TempUTIL}_j^{P_j}$ and the remaining tuples are stored in $\mathbf{DepUTIL}_j^{P_j}$. Then the hypercube is construct from the set of values $\mathbf{TempPUTIL}_j^{P_j}$. The tuples in $\mathbf{DepUTIL}_j^{P_j}$ may be used if in the VALUE propagation phase, the local node a_j does not find an assignment for itself in $\mathbf{TempUTIL}_j^{P_j}$, based on the values received from its Sep_j.

- The function

$$\text{BestValueHEUR}(\mathbf{TempUTIL}_j^{P_j}, \mathbf{DepUTIL}_j^{P_j}, D^*, Sep_j) \tag{4.4}$$

  allows to select the best value for a_j in the variable $\mathbf{TempUTIL}_j^{P_j}$ or in the variable $\mathbf{DepUTIL}_j^{P_j}$ corresponding to the optimal assigned values in D^* for the elements in Sep_j.

Algorithm 10 shows the changes, made to the algorithm DOCA, so that it could be possible to control the size of UTIL exchanged between APs. The procedure LinearityControl (line 10.31) computes the specified criteria, including the modifications that contribute to increase the scalability rather than optimality. The size of UTIL messages grows linearly, and is significantly reduced with respect to algorithm DOCA, at the cost of achieving a suboptimal channel allocation.

4.4.3 Numerical Example of DSCA

The example considers the DFS tree with 4 APs as seen in Figure 4.12. It is assumed an available set of channels $\mathbb{D} = \{1, 2, 3\}$ and interference factors as in Table 4.5. This numerical example considers a different set of interference factors than that shown in Table 3.2, only for the sake of a simpler presentation. The AP a_1 is the root node, and as initialization parameters are considered *Utildim*= 1 and local cost equal zero (lc = 0). Meaning that to this example in the UTIL propagations phase, a node can chooses only one option between its set of combinations, respecting that the local cost must be zero. These parameters were chosen for simplifying the example.

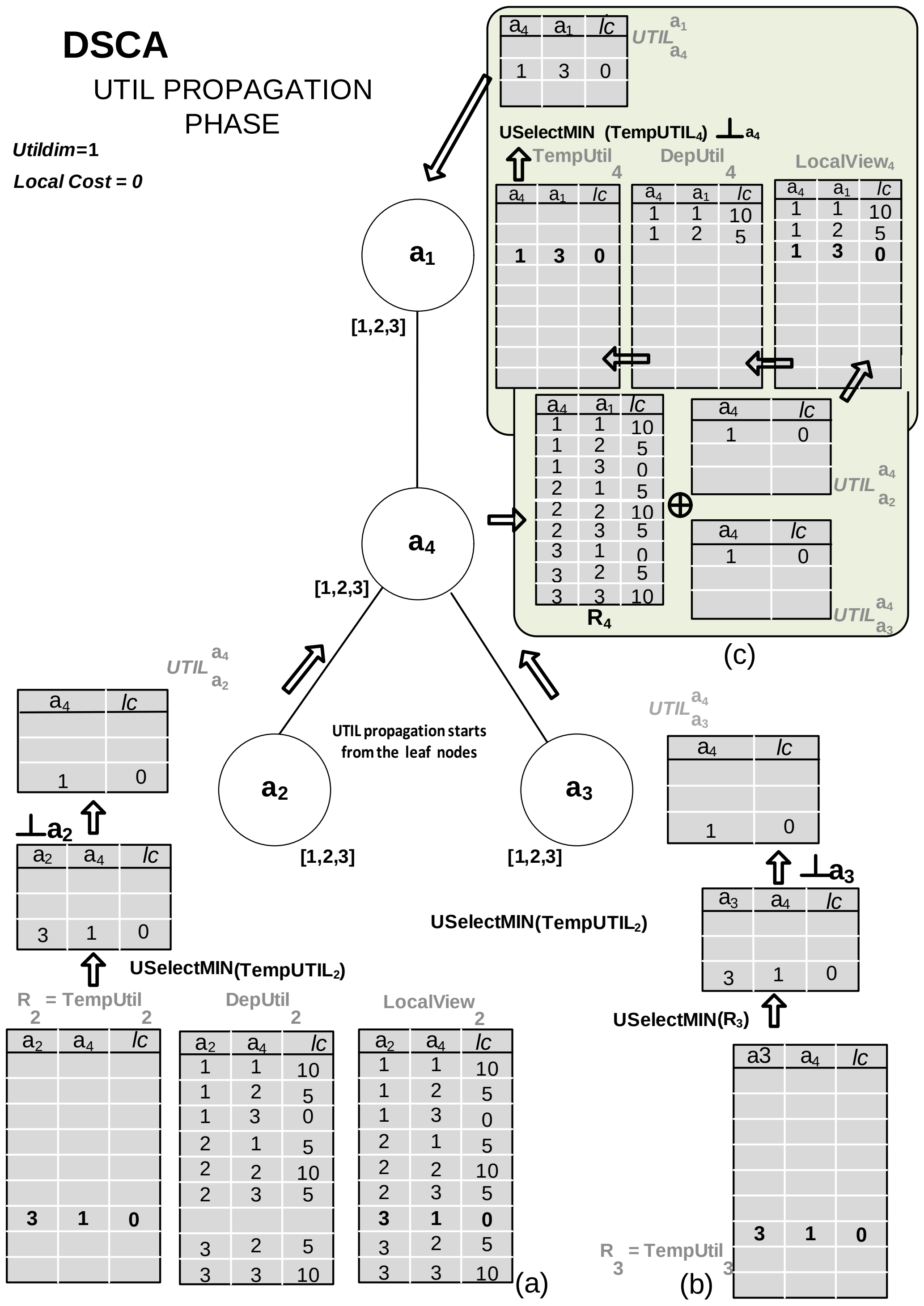
The details of the set of values constructed during the UTIL propagation phase, for the leaf nodes (a_2) and (a_3), is shown in Figure 4.12 (a) and (b). According to the UTIL propagation phase, each leaf node creates a relation R_j with its parent an pseudo-parents. During the construction of their relations it is calculated the local cost for all the tuples. These values are respectively stored in their local variables $\mathbf{LocalView}_2$ and $\mathbf{LocalView}_3$. Then considering the set of values in their **LocalView**, each leaf node executes the function LinearityControl to select its combinations respecting the heuristic parameters ($Utildim = 1$ and $lc = 0$). The nodes a_2 and a_3 choose respectively only one option in their variables $\mathbf{LocalView}_2$ and $\mathbf{LocalView}_3$, and it is stored in their variables $\mathbf{TempUTIL}_2$ and $\mathbf{TempUTIL}_3$. The remaining set of values are stored in the variables $\mathbf{DepUTIL}_2$ and $\mathbf{DepUTIL}_3$. From now on the nodes do not need anymore the variables $\mathbf{LocalView}_2$ and $\mathbf{LocalView}_3$. Then leaf nodes execute the operation USelectMIN, to choose combinations with the lowest cost, with their parent a_4. To this example as there is only one combination in variables $\mathbf{TempUTIL}_2$ and $\mathbf{TempUTIL}_3$, this is the one chosen. Finally they execute the operation PROJECTION ($\perp$), to eliminate themselves out. Then APs a_2 and a_3 send to their parent their respective $UTIL_{a_2}^{a_4}$ and $UTIL_{a_3}^{a_4}$ messages.

AP a_4 waits until it receives the UTIL messages from its children. It calculates the relation R_4 with its parent a_1. Operations of this node are presented in Figure 4.12 (c). After receiving the UTIL messages from its children, the AP executes the operation JOIN ($\oplus$) between its relation R_4 and the received UTIL messages. The result set of values are stored in its $\mathbf{LocalView}_4$. It is important to call attention to this point of the algorithm. The join operation will consider only the set of combinations where a_4 assumes value equal 1. The only one value that is present in R_4 and in the UTIL messages received from their children. As a consequence there is a reduction in the size of the variable $\mathbf{LocalView}_4$. Then the function LinearityControl is executed and the variables $\mathbf{TempUTIL}_4$ and $\mathbf{DepUTIL}_4$ are also created following the heuristic parameters. Once the variables $\mathbf{TempUTIL}_4$ and $\mathbf{DepUTIL}_4$ are constructed, the next operations are the same as the leaf nodes (USelectMIN and PROJECTION). Then AP a_4 sends its UTIL message $UTIL_{a_4}^{a_1}$ to its parent a_1.

The root AP a_1, receives the UTIL message from AP a_4. As it does not have a parent it immediately chooses its value d_1, that produces the minimum cost for the whole channel assignment problem, $d_1 = \{3\}$, shown in Figure 4.13 (d). Then AP a_1 initiates the execution of the third phase of the DSCA algorithm, *VALUE* propagation phase. The AP, via a message $VALUE_{a_1}^{a_4}(\{3\})$, informs its child a_4, of its choice. Then AP a_4 by executing the procedures BestValueHEUR restores from its variables $\mathbf{TempUTIL}_4$ or $\mathbf{DepUTIL}_4$ the value d_4 for itself, that presents the best local cost when its parent a_1 assumes the value $d_1 = \{3\}$, Figure 4.13 (e) point this computation. The AP a_4 chooses in $\mathbf{TempUTIL}_4$ as its best value $d_4 = \{1\}$. The process continues with a_4 executing the procedure ChooseValue and sending the messages, $VALUE_{a_4}^{a_2}(\{1\})$ and $VALUE_{a_4}^{a_3}(\{1\})$ to its children a_2 and a_3. After receiving their parent best assumption, nodes a_2 and a_3 restore their optimal values for $a_4 = \{1\}$, i.e a_2 assumes for itself the value $d_2 = \{3\}$ and a_3 assumes $d_3 = \{3\}$, Figure 4.13 (f) and (g). Then the algorithm finishes with the optimal solution $a_1 = \{3\}, a_2 = \{3\}, a_3 = \{3\}, a_4 = \{1\}$ presenting the final global cost equal to zero.

4.4.4 Simulation Results

The performance of the algorithm DSCA were evaluated under a group of more complex random network topologies. We considered three performance metrics: the number of messages, total amount of information exchanged and best global cost solution. In this sense, were used topologies with $|V| = \{10, 20, 30, \cdots, 100\}$ nodes with each AP having average degree $AD = \{3, 4, 5, 6\}$. It was assumed that each AP could select one among of



$UTIL_{a_4}^{a_1}$

a_4	a_1	lc
1	3	0

$TempUtil_4$

a_4	a_1	lc
1	**3**	**0**

$DepUtil_4$

a_4	a_1	lc
1	1	10
1	2	5

$LocalView_4$

a_4	a_1	lc
1	1	10
1	2	5
1	**3**	**0**

R_4

a_4	a_1	lc
1	1	10
1	2	5
1	3	0
2	1	5
2	2	10
2	3	5
3	1	0
3	2	5
3	3	10

$UTIL_{a_2}^{a_4}$

a_4	lc
1	0

$UTIL_{a_3}^{a_4}$

a_4	lc
1	0

$UTIL_{a_2}^{a_4}$

a_4	lc
1	0

$\perp a_2$

a_2	a_4	lc
3	1	0

$R_2 = TempUtil_2$

a_2	a_4	lc
3	**1**	**0**

$DepUtil_2$

a_2	a_4	lc
1	1	10
1	2	5
1	3	0
2	1	5
2	2	10
2	3	5
3	2	5
3	3	10

$LocalView_2$

a_2	a_4	lc
1	1	10
1	2	5
1	3	0
2	1	5
2	2	10
2	3	5
3	**1**	**0**
3	2	5
3	3	10

$UTIL_{a_3}^{a_4}$

a_4	lc
1	0

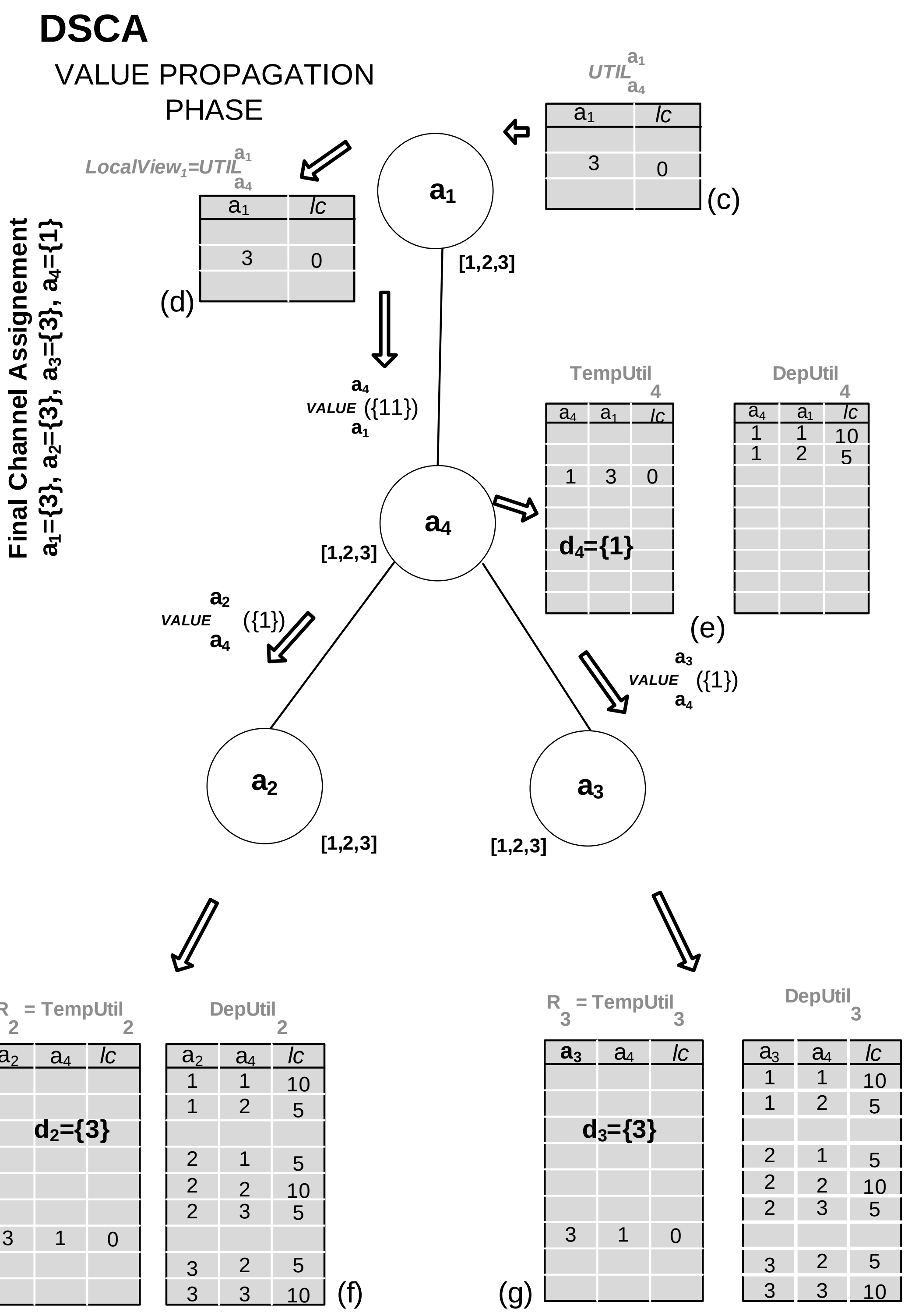
$\perp a_3$

a_3	a_4	lc
3	1	0

$R_3 = TempUtil_3$

a3	a_4	lc
3	**1**	**0**

Figure 4.12: Relations R_2, R_3 and R_4 are presented in (a), (b) and (c); Projections of a_2 and a_3 out of their relations with a_4 (a) and (b); In (c) a_4 joins the UTIL messages from a_2 and a_3 with its relation R_4; Node a_4 creates its $TempUTIL_4$ and $DepUTIL_4$ based on heuristic parameters.



(c) $UTIL_{a_4}^{a_1}$

a_1	*lc*
3	0

(d) $LocalView_1=UTIL_{a_4}^{a_1}$

a_1	*lc*
3	0

(e) $TempUtil_4$

a_4	a_1	*lc*
1	3	0
$d_4=\{1\}$		

$DepUtil_4$

a_4	a_1	*lc*
1	1	10
1	2	5

(f) $R_2 = TempUtil_2$

a_2	a_4	*lc*
$d_2=\{3\}$		
3	1	0

$DepUtil_2$

a_2	a_4	*lc*
1	1	10
1	2	5
2	1	5
2	2	10
2	3	5
3	2	5
3	3	10

(g) $R_3 = TempUtil_3$

a_3	a_4	*lc*
$d_3=\{3\}$		
3	1	0

$DepUtil_3$

a_3	a_4	*lc*
1	1	10
1	2	5
2	1	5
2	2	10
2	3	5
3	2	5
3	3	10

Figure 4.13: The root node a_1 sends a VALUE message $VALUE_{a_1}^{a_4}$ to its child a_4 (d); In (e) a_4, based on the best value of its parent, chooses its best value in $TempUTIL_4$; Node a_4 sends VALUE messages to its children a_2 and a_3 (e); Nodes a_2 and a_3 after receiving the VALUE message from their parent a_4 choose their best values in $TempUTIL_2$ (f) and $TempUTIL_4$ (g).

Algorithm 10: DSCA - Distributed SubOptimal Channel Assignment

```
10.1  Procedure DSCA($A, D, \mathbb{R}$):
10.2      $\forall a_j \in A$;
10.3      PHASE 1: DFS arrangement - run token passing mechanism as in [41];
10.4      After execution, $a_j$ knows $P_j, PP_j, C_j, PC_j, Sep_j$ ;
10.5      PHASE 2: UTIL propagation (bottom-up UTIL message propagation starts from the
          leaf nodes)
10.6      $D^* = \varnothing$; $R_j = null$;
10.7      $Utildim = n$; //to limit the size of $\textbf{TempUTIL}_j^{P_j}$
10.8      $\textbf{LocalView}_j = null$;
10.9      Compute $R_j$; // a sub set with all combination values between $a_j$ with $P_j$ and $PP_j$, and the
          computation of $lc_j$
10.10     forall the $a_k \in C_j$ //* for all children of $a_j$ - if $a_j$ is a leaf node, skip this */ do
10.11         ReceiveUTILMessage ($Sep_k, UTIL_k^j$); // wait for $UTIL$ message arrive from $a_k$
10.12         $\textbf{LocalView}_j = \textbf{LocalView}_j \oplus UTIL_k^j$; //Join UTIL messages from $C_j$ as they arrive
10.13     end
10.14     if ($P_j == \varnothing$) // that means that $a_j$ is the root node then
10.15         $D^* \leftarrow$ Min($\textbf{LocalView}_j, d_2$)$\perp d_2$; // Select best value $d_1$ for $a_j$ from minimum local cost
              values $d_2$
10.16         Send VALUE ($D^*$) to all $C_j$;
10.17         else
10.18         $\textbf{LocalView}_j = \textbf{LocalView}_j \oplus R_j$; //also join relations with parent/pseudo-parents;
10.19         [$\textbf{TempUTIL}_j^{P_j}, \textbf{DepUTIL}_j^{P_j}$] = LinearityControl[$\textbf{LocalView}_j, Utildim$];
10.20         $UTIL_j^{P_j}$ =USelectMIN($TempUTIL_j^{P_j}$)$\perp a_j$ //use USelectMIN to select the tuples with
              minimum local cost and $\perp$ to eliminate self out of message to parent;
10.21         SendUTILMessage ($Sep_j, UTIL_j^{P_j}$);
10.22     end
10.23     PHASE 3: VALUE propagation (top-down VALUE message propagation)
10.24     wait for $VALUE_{P_j}^j(D^*)$ message from $P_j$; // the optimal assignment for the elements in
          $Sep_j$
10.25     $D^* \leftarrow$ BestValueHEUR ($\textbf{TempUTIL}_j^{P_j}, \textbf{DepUTIL}_j^{P_j}, D^*, Sep_j$); // Selects best $a_j$ value in
          $\textbf{TempUTIL}_j^{P_j}$ or in $\textbf{DepUTIL}_j^{P_j}$ corresponding to the optimal assigned values $D^*$ for the
          elements in $Sep_j$
10.26     forall the $a_k \in C_j$ /*for all children of $a_j$ - if $a_j$ is a leaf node, skip this*/ do
10.27         AgentView= $\varnothing$;
10.28         AgentView $\leftarrow$ ChooseValue ($D^*, Sep_k$); // chooses values in $D^*$ corresponding to
              elements of $Sep_k$
10.29         Send $VALUE_{P_j}^k$(AgentView) to all $C_j$;
10.30     end
10.31     Procedure LinearityControl[$\textbf{LocalView}_j, Utildim$]
10.32         Initialize: $\textbf{TempUTIL}_j^{P_j} = null$, $\textbf{DepUTIL}_j^{P_j} = null$, $media = 0$;
10.33         $lc_{max}$ =Max($\textbf{LocalView}_j, d_{K+1})_{d_{K+1}}$; // select max local cost between the tuples in
              $\textbf{LocalView}_j$
10.34         $lc_{min}$ =Min($\textbf{LocalView}_j, d_{K+1})_{d_{K+1}}$; // select min local cost between the tuples in
              $\textbf{LocalView}_j$
10.35         $media = (lc_{max} + lc_{min})/2$;
10.36         while $\textbf{LocalView}_j$ do
10.37             $\textbf{TempUTIL}_j^{P_j} \leftarrow \textbf{LocalView}_j$; // $\forall$ tuples in $\textbf{LocalView}_j$ that have $lc \leq media$
                  until $\textbf{TempUTIL}_j^{P_j} < Utildim$
10.38             $\textbf{DepUTIL}_j^{P_j} \leftarrow \textbf{LocalView}_j$; // $\forall$ tuples in $\textbf{LocalView}_j$ that have $lc > media$ or
                  $\textbf{TempUTIL}_j^{P_j} > Utildim$
10.39         end
10.40     end
10.41     Return [$\textbf{TempUTIL}_j^{P_j}, \textbf{DepUTIL}_j^{P_j}$];
10.42 end
```

the three channels $\{1, 2, 3\}$ and interference factors as in Table 4.5 [3]. For each scenario we report the average over a set of 100 randomly generated topologies.

As benchmark algorithms, were used the algorithms LO-A [17], Hsum [14] and also an additional method denoted as random channel assignment method, identified as *Random* in the results. The Random channel assignment method simulates deployments where there is no administration and no local or cooperative mechanisms to control the channel allocation between the APs. For this method channels are randomly selected once for each AP. The simulations of all phases of the DSCA algorithm were done using the software Matlab®, as well as the creation of the wireless network topologies and the corresponding interference constraint graphs.

Table 4.5: Interference Factors for the Numerical Example of DSCA Algorithm

Channel Spacing $\lvert n-m \rvert$	Overlap Factor $f_{ij}(x,y) : \lvert n-m \rvert$
0	10
1	5
2	0

4.4.4.1 Performance Analysis

The first values to call attention is the number of exchanged messages between APs, shown in Figure 4.14. The results of DSCA algorithm related to the number of control messages (UTIL+VALUE), continues to be similar to algorithm DOCA. The algorithm DSCA also scales linearly with the number of APs and does not depend on the average degree of the topology. However for DFS arrangement the results are different from DOCA algorithm. The algorithm DSCA produces identical number of messages for different APs' average degrees for the topologies with the same number of APs. The DFS phase no longer depends on the number of edges and average degree of the topology. It is related to the changes applied by our proposed algorithm, Ddfs-Improved, that requires $2|V|-2$ exchanged messages to create a pseudo-tree.

In Figure 4.15 is shown the number of exchanged message for the higher APs' average degree ($AD = 6$) of all topologies. We compare the results of the DSCA and LO-A algorithms. It is possible to observe that the values presented by the DSCA algorithm, as shown in previous tests, Figure 4.9, continue to present a number of exchanged messages about one order of magnitude lower than LO-A. The results show that although there

[3]In this numerical example we consider a different set of interference factors than that shown in Table 3.2

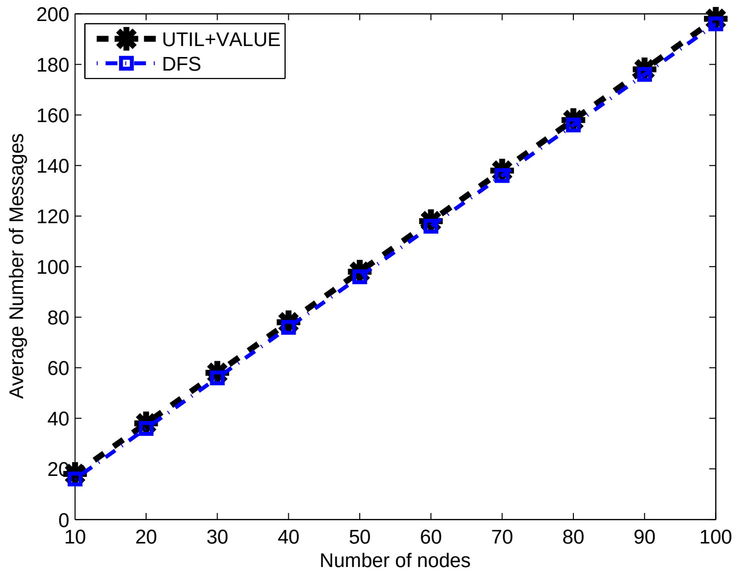


Figure 4.14: Number of exchanged messages for DSCA, AD=3,4,5,6.

was an increase in the number of nodes of the topologies, the behavior of the algorithm DSCA related to the number of exchanged messages between APs remained the same. It is possible to say that the DSCA algorithm got a significant scalability increase relative to the number of nodes.

Considering the final global cost values shown in Figure 4.16 for the higher APs' average degree $(AD = 6)$ of all topologies, the DSCA algorithm achieves the smallest final global cost, in relation to the algorithms LO-A, Hsum and the random configuration method $(Random)$, which means that it achieves the best solution. The results show that even prioritizing the scalability through control of the information size, exchanged between APs, the algorithm can find best final results. Showing to be better in approximately 19% when compared to the algorithm LO-A, 31% in relation to the algorithm Hsum and finally 43% better the Random assignment method. Moreover, the same behavior can be observed through Figure 4.17, where are compared the results for APs' average degrees $AD = \{3, 4, 5, 6\}$ for the topologies of $|V| = 50$ and $|V| = 100$ APs . The algorithm DSCA always outperforms the algorithms LO-A, Hsum and $Random$ for simpler networks ($|V| = 50$ and $AD = 6$) to more connected network ($|V| = 100$ and $AD = 6$), where the complexity of the channel allocation problem increases.

In DOCA algorithm as in DPOP, the size of the UTIL messages can reach exponential dimensions, since UTIL exchanged message between APs are directed related to the size of the largest domain D_{max} and to the size of the largest separator Sep_k of any node in

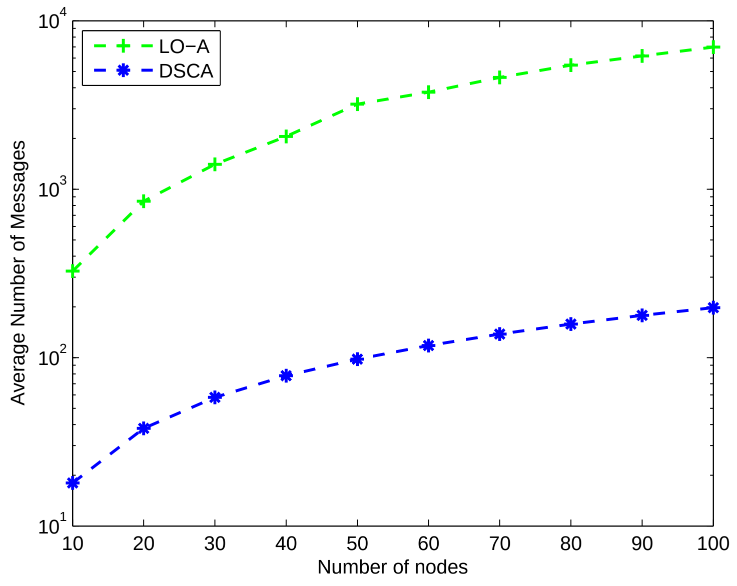


Figure 4.15: Average number of exchanged control messages for DSCA and LO-A (AD=6).

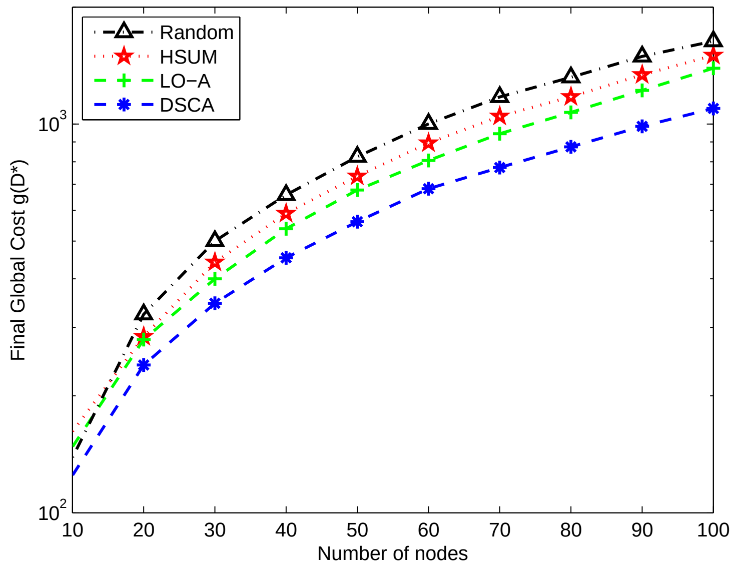


Figure 4.16: Final global cost value for DCAA-S, LO-A, Hsum and Random (AD=6).

the DFS arrangement. Due to this behavior our heuristic is based on the configurable variable *Utildim*. This variable will determine de maximum size of the hypercube that will be exchanged by the UTIL propagation phase.

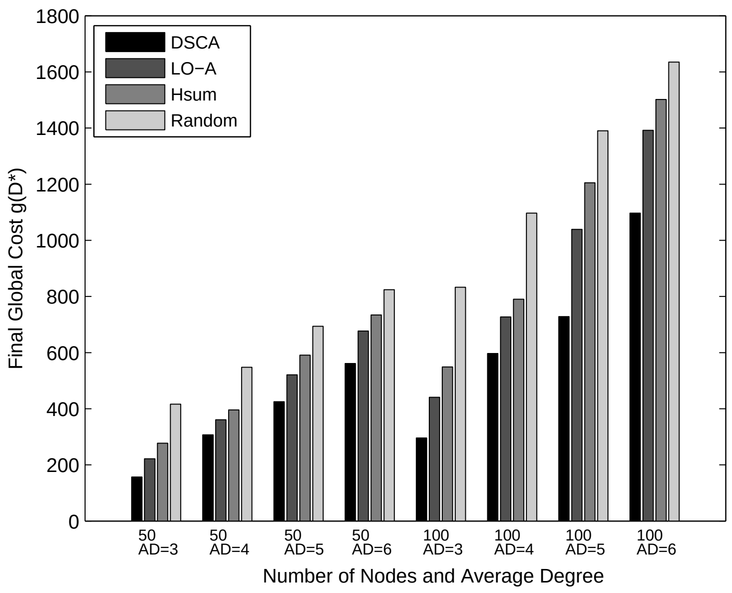


Figure 4.17: Final global cost value for DCAA-S, LO-A, Hsum and Random ($|V| = 50$ and $|V| = 100$, AD=3,4,5 and 6).

By considering L the maximum number of channels the APs can assign and AD a possible AP's average degree. The value of the variable *Utildim* is defined as L^{AD}. This value represents the total number of possible channel assignments in scenarios where each AP can select among L channels and have average degree of AD. Taking an example of a scenario where the APs may choose one of tree channels and have average degree AD=9, the maximum value the variable *Utildim* may assume is *Utildim*$=3^9$=19683, i.e. there are a maximum of 19683 possible channel assignments. Our strategy was to set an appropriate value to the variable *Utildim*. The value had to present an suitable relationship between scalability and environment processing in which the tests were performed. For our scenarios, the best ratio between number of channels and the average degree the APs could assume were *Utildim*$= 3^4$ ($c = 3$ and $AD = 4$). The value was chosen after some computational tests. The experiments were executed on a 2.1 GHz, dual core computer, memory of 2.0 GB with Matlab 7.11 (R2010b).

After choosing the appropriate value to the variable *Utildim* it is necessary to adopt a criterium to select the set of tuples in the variable **LocalView**. Only these set of tuples will be included in the hypercube, and will provide local cost, lower or equal to a pre-determined cost limit. This criterium also is configurable. In Algorithm 10 line 10.31, the procedure **LinearityControl** considers only values that are little or equal the variable *media*. This variable represents the average between the highest and lowest cost computed among all possible combinations in the variable **LocalView**.

In Table 4.6 is shown, for all tested scenarios, the average values concerning to the exchanged messages between APs and their size in bytes. To present the increase in scalability through our heuristic, we consider the number of MTUs necessary to be exchanged by each AP to find their global solution. We considered a MTU (*Maximum transmission unit*) of 1500 bytes as the largest protocol unit for the MAC level.

Table 4.6: Total of Messages and Total of Bytes

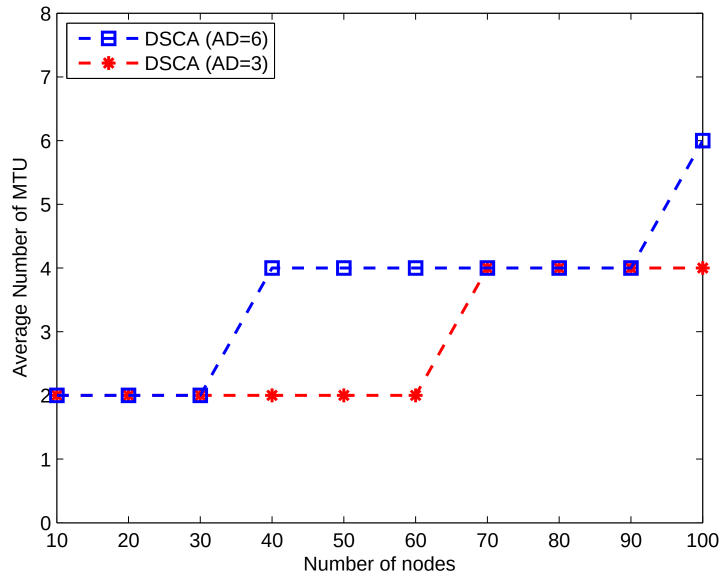
	Algorithm DSCA							
Nodes	Total of Messages				Total of Bytes			
	AD=3	AD=4	AD=5	AD=6	AD=3	AD=4	AD=5	AD=6
10	18	18	18	18	6133	5062	8370	9220
20	38	38	38	38	21424	19506	28471	32898
30	58	58	58	58	48439	44834	60642	72428
40	78	78	78	78	74124	75929	104212	125048
50	98	98	98	98	112943	115359	150548	187131
60	118	118	118	118	155876	159546	215464	257433
70	138	138	138	138	210607	214809	296874	329340
80	158	158	158	158	269400	269074	360916	430043
90	178	178	178	178	319178	324717	447607	517853
100	198	198	198	198	384618	400487	490319	616464

	Algorithm LO-A							
Nodes	Total of Messages				Total of Bytes			
	AD=3	AD=4	AD=5	AD=6	AD=3	AD=4	AD=5	AD=6
10	306	324	333	326	4892	5190	5325	5217
20	683	761	878	848	10933	12172	14055	13560
30	1161	1429	1483	1404	18570	22856	23734	22463
40	1644	2056	2203	2298	26309	32893	35250	36775
50	2202	2658	2958	3201	35237	42524	47321	51218
60	2813	3430	3647	3768	45004	54885	58356	60289
70	3338	4063	4424	4610	53406	65008	70777	73753
80	3767	4768	5201	5455	60276	76294	83210	87276
90	4500	5374	6069	6180	71993	85978	97105	98886
100	4854	6105	6751	6984	77661	97676	108019	111740

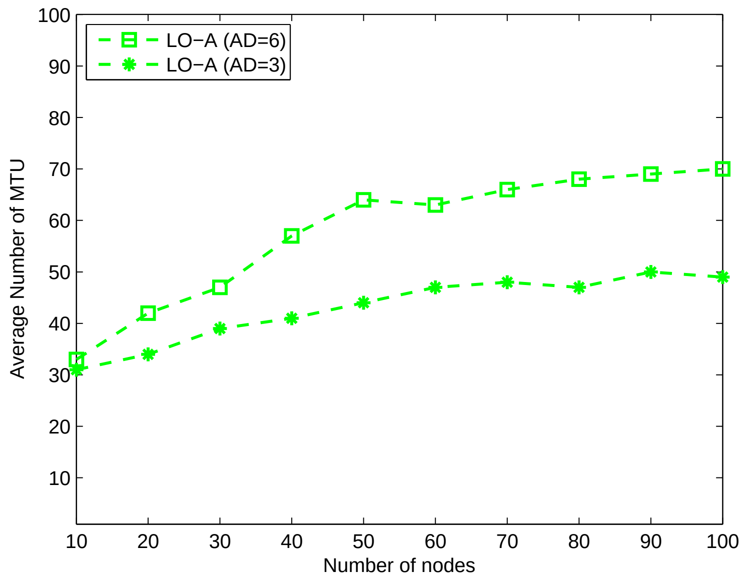
In relation to DSCA algorithm, each node sends in average 2 control messages, until the end of the algorithm. Considering the total size of the exchanged messages and their quantity, it is possible to get the number of bytes per message. Dividing this value by the number of bytes of the adopted MTU, we get the number of MTUs per message. As each AP sends 2 messages it is possible to determine the total amount of MTUs that each node will send to the network to get its final solution. Taking as an example the scenario of 100 nodes with APs' average degree AD=3 in Table 4.6, it is possible to determine that each AP sends in average 2 messages by dividing 198/100 (total number of messages/nodes). Then we calculate the size of each message by dividing 384618/198 (total of bytes/total of messages), the result leads us to find the number of necessaries MTUs to each message, 1942/1500 (total of bytes per message/MTU). In this example, it is necessary 1.2 MTUs per message. To best represent the value, we considered that is necessary 4 MTUs per AP (2×1.2). In DSCA algorithm, the number of the MTUs per AP, is directed related to the size of each exchanged message.

By analyzing the results of the LO-A algorithm in Table 4.6, it is possible to see that it changes more messages than the DSCA algorithm. These messages are smaller than the size of the adopted MTU. For each message it is possible to consider only one MTU. To LO-A algorithm, the number of MTUs per AP, is directed related to the number of exchanged message per AP.

Figure 4.18 presents the results of the smallest and the biggest ($AD = 3$ and $AD = 6$) APs' average degrees for all the topologies $|V| = \{10, 20, 30, \cdots, 100\}$. Comparing the results of the algorithms DSCA and LO-A, respectively presented in Figure 4.18 (a) and (b), it is possible to observe that the values of DSCA algorithm are negligible when compared with LO-A algorithms. Even for more complex networks the DSCA algorithm provided lower numbers of MTUs transmitted per node on the network. It shows that the used heuristic to control the size of the UTIL messages showed to be efficient. Our method was able to control the exponential growth of the UTIL messages, thus ensuring a gain in scalability.



(a)



(b)

Figure 4.18: Average number of MTU by Node for DSCA and LO-A.

Chapter 5

Conclusion

Contents

This thesis addressed the problem of channel allocation in WLANs, motivated by a return channel architecture for IDTV (Interactive Digital TV) implemented using an access point (AP) based architecture. The channel allocation problem was investigated in order to minimize interference and improve network performance.

The channel allocation problem was modeled as a DCOP (*Distributed Constraint Optimization Problem*), that uses multiagent approach. An undirected constraint graph is employed to model the physical network topology. In DCOP context, APs represent agents and must be prioritized in a total order, making use of a spanning tree. An initialization procedure using a predefined common control channel is used, allowing each AP to identify other APs in the neighborhood (constraint graph) and also to configure its logical ordering. The APs of different administrative domains, by collaborating with each other, can get a more efficient channel allocation for all of them.

This thesis presents two main contributions. The first one is a novel formulation for the channel assignment problem as a DCOP. The second is the proposal and investigation of four distributed algorithms for channel allocation, which are able to reach the optimal or suboptimal frequency assignment. The proposed methods aim to minimize the co-channel and adjacent-channel interference in WLANs.

Two algorithms were proposed by considering ADOPT's approach, named DCAA-O and DCAA-S. The DCAA-O algorithm finds the optimal solution to the channel assignment problem for a group of APs. The algorithm presented the advantage of requiring polynomial space at each agent, but the drawback of producing exponential number of small messages, resulting in large communications overheads.

The second algorithm, called DCAA-S, was derived from DCAA-O. It includes a heuristic that contributes to efficiency rather than optimality. As a consequence, the number of exchanged messages was significantly reduced with respect to DCAA-O, presented to be some orders of magnitude lower, at the cost of achieving a suboptimal channel allocation. The algorithm while demanding clearly the least messages exchanges, also achieved a final global cost very close to the optimum one achieved by DCAA-O. Therefore, such reduction in message exchanges did not come at the cost of a relevant decrease in performance.

Considering the ADOPT based strategy, we conclude that the proposed algorithms converge to the optimal or suboptimal solution. However, based on the evaluated scenarios, to more complex topologies even the suboptimal approach presented drawbacks in relation to scalability.

The third and fourth algorithms, called DOCA and DSCA, were based in the dynamic programming algorithm called DPOP. According to the number of control messages both algorithms scales linearly with the number of APs and did not depend on the average degree of the constraint graph, thus generating low communication overhead. For DFS arrangement DOCA algorithm depends on the number of nodes and average degree of the topology. We solved this problem in DSCA algorithm by proposing an improved algorithm called Ddfs-Improved. The DOCA algorithm achieves a global optimal solution. However, DOCA's complexity is given by the size of the largest UTIL message it produces, which is exponential in the induced width of the DFS ordering used. This behavior compromises its scalability especially in environments where there are memory constraints. The DSCA algorithm was developed as an alternative to solve the problem, without losing quality in relation to the overall global cost value. The DSCA proposed heuristic controls the size of the UTIL messages exchanged between the APs. The performance of the DOCA algorithm limited in solving the channel allocation problem, was improved by DSCA algorithm increasing by five times scalability.

In relation to the DPOP based strategy, the third algorithm, DOCA, converges to an optimal solution but is limited in its scalability. The algorithm DSCA by reducing the size of the exchanged messages, presents a suboptimal solution and a better scalability relative to DOCA.

5.1 Summary of contributions

- A global cost function for the DCOP formulation that was mathematically defined from the overlapping factor between interfering APs;
- The channel allocation problem was modeled as a DCOP that uses a multiagent approach;
- The ADOPT based algorithms:
  - DCAA-O (*Distributed Channel Assignment Algorithm - Optimal*), which finds the optimal solution to the channel allocation problem to a group of APs;
  - DCAA-S (*Distributed Channel Assignment Algorithm - Suboptimal*), derived from DCAA-O. It achieves the suboptimal solution reducing the number of exchanged control messages.
- The DPOP based algorithms:
  - DOCA (*Distributed Optimal Channel Assignment*), achieves a global optimal solution, requiring a linear number of messages;
  - DSCA (*Distributed SubOptimal Channel Assignment*) derived from DOCA, reduces the size of the exchanged messages, increases the scalability presented by the algorithm DOCA.
- The Ddfs-Improved Algorithm, a distributed DFS tree algorithm where each node knows its complete tree.

5.2 Future Works

During the evolution of this thesis, were detected interesting possibilities for future research, such as:

- Include other factors in the global cost function as APs power level to topology control in the process of negotiation;
- Apply DCOP method to other networks, as WSNs (*Wireless Sensor Network*) to control power consumption;
- Explore scenarios with dynamic access to spectrum;
- Better explore the DFS algorithms to be applied in the DCOP strategy;

- Propose new heuristics to improve scalability of DOCA algorithm;
- Apply the proposed solutions in a hardware platform;
- Explore cooperation between agents and DCOP for a large class of coordination problems in wireless networks.

5.3 Publications

- Conferences:
  - T. Monteiro, M. Pellenz, M. Penna, F. Enembreck, and R. Souza, "On Optimal Distributed Channel Allocation for Access Points in WLANs", IEEE Networking 2011 - Performance Evaluation of Cognitive Radio Networks - Workshop (PE-CRN), May 13, 2011;
  - T. Monteiro, M. Pellenz, C. Penna, F. Enembreck, R. Souza, and G. Pujolle, "A Multi-Agent Approach to Optimal Channel Assignment in WLANs", IEEE Networking 2012, Apr 01, 2012;
  - T. Monteiro, M. Pellenz, C. Penna, F. Enembreck, R. Souza, and G. Pujolle, "An Optimal Channel Assignment Strategy for WLANs using Distributed Optimization", IEEE/IFIP NOMS 2012, Apr 16, 2012;
- Journal:
  - T. Monteiro, M. Pellenz, M. Penna, F. Enembreck, and R. Souza, and G. Pujolle, "Channel allocation algorithms for WLANs using distributed optimization", AEU - International Journal of Electronics and Communications November 2011, vol. 6827, pp. 73-84, Sept 2011.
- Undergraduate research project(PIBIC)
  - J. Mendes, M. Pellenz, T. Monteiro, "Distributed Algorithm for Topology Control in RSSI", Pontifical Catholic University of Paraná State - PUC-PR, Nov 2012.

Appendix A

APPENDIX

DCAA-O Correctness - Algorithm 5

Lemma 1 states that in a stable state, an AP's estimate of channel assignment, results in the minimum possible cost for the scenario.

Assuming:

- $A = \bigcup_{i=1} |\hat{V}| \{(a_i, d_i) : a_i \in V, d_i \in \mathbb{D}_i\}$ a channel allocation in a graph **G**;
- Ω^*_G the set of all channel allocations with minimum cost in a graph **G**;
- DC^G_j the set of j's descendants in a graph **G**, and;
- $SVALUE_j$ a sub-graph of a $VALUE$ graph, construct by a set of vertex: $\{j\} \cup DC^{VALUE}_j$.

At the end of each interaction, the set $\mathbf{SubtreeValues}_j(d_j) \cup \mathbf{Currentvw}_j$ corresponds to channels allocation in the constraint graph, whose vertex are defined by $\{j\} \cup DC^{VALUE}_j \cup P^{VALUE}_j$.

Lemma 1: $\forall a_j \in V, a_j \neq a_1$, $\mathbf{SubTreeValues}_j(d_j) \cup \mathbf{Currentvw}_j \in \Omega^*_{SVALUE_j}$

Basic case: Hypothesis - $C^{VIEW}_j = \emptyset$,

1. The set $\mathbf{LocalCosts}_j$ stores the triple $(d_j, lc_j(d_j), sc_j(d_j))$ for all $d_j \in \mathbb{D}_j$ (lines 5.27 to 5.31);

2. The variable $lc_j(d_j)$ stores the sum of the interference costs of temporary allocation (a_j, d_j) related to each allocation $(a_i, d_i) \in \mathbf{Currentvw}_j$ (line 5.28);

3. As $C_j^{View} = \emptyset$ by hypothesis, the allocation $\textbf{SubTreeValues}_k(d_k) = \emptyset$, $\forall a_k \in C_j^{VIEW}$;

4. $sc_j(d_j) = 0$ (lines 5.29 and 5.33, $C_j^{VIEW} = \emptyset$);

5. As d_j is chosen to minimize $lc_j + sc_j$ in $\textbf{LocalCost}_j$, (line 5.45), it is possible to conclude that the channel allocation defined by $\textbf{Currentvw}_j \cup \{(a_j, d_j)\}$ present the minimum interference cost in graph $SVALUE_k$;

6. $\textbf{SubTreeValues}_j(d_j) = \textbf{SubTreeValues}_k(d_k) \cup \{(a_j, d_j)\}$ (line 5.47);

7. $\textbf{SubTreeValues}_j(d_j) \cup \textbf{Currentvw}_j \in \Omega^*_{SVALUE_j}$.

Inductive step: Hypothesis - $\forall a_k \in C_j^{VIEW}$, $\textbf{SubTreeValues}_k(d_k) \cup \textbf{Currentvw}_k \in \Omega^*_{SVALUE_k}$,

1. The set $\textbf{LocalCosts}_j$ stores the triple $(d_j, lc_j(d_j), sc_j(d_j)$ for all $d_j \in \mathbb{D}_j$ (lines 5.27 to 5.31);

2. The variable $lc_j(d_j)$ stores the sum of the interference costs of temporary allocation (a_j, d_j) related to each allocation $(a_i, d_i) \in \textbf{Currentvw}_j$ (line 5.28);

3. By hypothesis the channel allocation $\textbf{SubTreeValues}_k(d_k)$, $\forall a_k \in C_j^{VIEW}$ present the minimum interference cost in graph $SVALUE_k$;

4. The variable $sc_j(d_j)$ stores the sum of the interference costs of temporary allocation (a_j, d_j) related to $\textbf{SubTreeValues}_k(d_k)$, $\forall a_k \in C_j^{VIEW}$ (lines 5.36 to 5.40);

5. As d_j is chosen to minimize $lc_j + sc_j$ in $\textbf{LocalCost}_j$, (line 5.45), it is possible to conclude that the channel allocation defined by $\textbf{Currentvw}_j \cup \{(a_j, d_j)\} \cup \textbf{SubTreeValues}_k(d_k)$ present the minimum interference cost in graph $SVALUE_k$;

6. $\textbf{SubTreeValues}_j(d_j) = \textbf{SubTreeValues}_k(d_k) \cup \{(a_j, d_j)\}$ (line 5.47);

7. $\textbf{SubTreeValues}_j(d_j) \cup \textbf{Currentvw}_j \in \Omega^*_{SVALUE_j}$.

Theorem 1: $\textbf{SubTreeValues}_1(d_1) \in \Omega^*_G$

- The variable $sc_1(d_1)$ stores the sum of the interference costs of temporary allocation (a_1, d_1) related to each of the allocations $\{(a_k, d_k)\} \in C_1^{VIEW}$, (lines 5.7 to 5.15);

- By Lemma 1, the allocation $\mathbf{SubTreeValues}_k(d_k)$, $\forall a_k \in C_j^{VIEW}$ present the minimum interference cost in graph $SVALUE_k$;
- From 1 and 2 we can deduce that $sc_1(d_1)$ stores the sum of the interference costs of temporary allocation (a_1, d_1) related to $\mathbf{SubTreeValues}_k(d_k)$, $\forall a_k \in C_1^{VIEW}$(line 5.11);
- Then, $\mathbf{SubTreeValues}_1(d_1) = \mathbf{SubTreeValues}_1(d_1) \cup \{(a_j, d_j)\}$;
- And $\mathbf{SubTreeValues}_1(d_1) \in \Omega^*_{SVALUE_1}$;
- By construction $SVALUE_1 = \mathbf{G}$;
- $\mathbf{SubTreeValues}_1(d_1) \in \Omega^*_G$.

List of Figures

List of Tables

List of Abbreviations

ABT	Asynchronous Backtracking
ADOPT	Asynchronous Distributed OPTimization
AI	Artificial Intelligence
AP	Access Point
ADSL	Asymmetric Digital Subscriber Line
DCOP	Distributed Constraint Optimization Problem
DCAA-O	Distributed Channel Assignment Algorithm - Optimal
DCAA-S	Distributed Channel Assignment Algorithm - Suboptimal
DFS	Depth-First Search Trees
DisCSP	Distributed Constraint Satisfaction Problem
DOCA	Distributed Optimal Channel Assignment
DSCA	Distributed SuOptimal Channel Assignment
DSL	Digital Subscriber Line
DSSS	Direct Sequence Spread-Spectrum
FHSS	Frequency Hopping Spread
FRODO	An Open-Source framework for Distributed Constraint Optimization
IDSL	ISDN Digital Subscriber Line
IDTV	Interactive Digital TV
ISM	Industrial, Scientific and Medical
ITU	International Telecommunication Union
LAN	Local Area Network

LCCS	Least Congested Channel Search
LO-A	Local-Coord Algorithm
LTE	Long-term-evolution
MIMO	Multiple-input Multiple-output
MTU	Maximum transmission unit
NA	Network Adapter
OPNET	A software tool for network modeling and simulation
PIBIC	Undergraduate research project
PLC	Power Line Communication
RADSL	Rate-Adaptive Digital Subscriber Line
SDSL	Symmetric Digital Subscriber Line
SINR	Signal-to-interference-plus-noise Ratio
SynchBB	Synchronous Branch and Bound
Wimax	Worldwide Interoperability for Microwave Access
WLANs	Wireless Local Area Network
WMN	Wireless Mesh Network
WSN	Wireless Sensor Network
Wi-Fi	Wireless Fidelity
xDSL	xDigital Subscriber Line
VDSL	Very-high-bit-rate Digital Subscriber Line

List of Symbols

A	a set of agents
$\mathbb{A} = \{a_1, a_2, \cdots, a_n\}$	set of APs
AD	the average degree of a grapg G
A'_i	a set of APs descendent of an ancestor $a_i \in P_j^{value}$
AgentView	the optimal set of values a node a_j sends to its child a_k
a_b	a back-edge handler
a_i	the ith agent
a_j	the jth agent
a_k	the kth descendent agent
(a_i, a_j)	a pair of agents
(a_j, d_j)	a view of an agent a_j (the current context)
$\{(a_i, d_i)\}$	the current view of an agent a_j
BW	the null-to-null bandwidth
$\mathbf{Currentvw}_i$	the current set of views $\{(a_k, d_k)\}$ of an agent a_i, where $a_k \in P_i^{value}$
$\mathbf{Currentvw}_j$	the current set of views $\{(a_i, d_i)\}$ of an agent a_j, where $a_i \in P_j^{value}$
C_j	the set of children of an agent a_j
C_j^{value}	the set of children of an agent a_j in the VALUE graph
C_j^{view}	the set of children of an agent a_j in the VIEW graph
c_n	an operating channel

D	a set of N channels to be assigned to agents
$D = \{d_1, d_2, \ldots, d_l\}$	the agent's variables
$\mathbb{D} = \{d_1, d_2, \ldots, d_l\}$	APs values (available channel set)
$D^n = \underbrace{D \times \cdots \times D}_{n}$	the cartesian product
D^K	the finite domain D where $K = \lvert a_j \cup Sep_j \rvert + 1$
$\mathbb{D}_{x_i}$	the finite domain of the ith agent
$\mathbb{D}_j$	the finite domain of the jth agent
$\mathbb{D}_j^{'}$	a reduced domain of $\mathbb{D}_j$
D^*	the optimal set of values which minimize a global cost function $g(D)$
D_{max}	the size of the largest domain
d_i	value assigned to the ith agent
d_j	value assigned to the jth agent
d_j^*	the best channel option of an agent a_j
d_b	the variable of a back-edge handler node a_b
dim	dimension of a matrix
E	the set of conflict edges (links)
e_k	an conflict edge $e_k \in E$
$\mathcal{F}(n, f)$	the approximated IF filter frequency response
f_{ij}	a cost function
$f_{ij}(d_i, d_j)$	cost function applied to the values of d_i (from a_i) and d_j (from a_j)
LocalView$_j$	the current set $\big\{(x_1, x_2, \cdots x_K, x_{K+1}) \quad \vert \quad (x_1, x_2, \cdots x_K) \in D^K\big\}$ of an agent a_j, where $K = \lvert a_j \cup Sep_j \rvert + 1$
$\mathbf{G}$	an undirect constraint graph
$\mathbb{G}_m$	the set of cells interfered by $\mathbb{Z}_m$
$\gamma_{a_i}^{c_n}$	the SINR experienced by an AP a_i
g^*	the best global cost value of a function g
$\mathbb{H}_j$	a reduced domain of $\mathbb{D}_j$
I_{a_i}	the interference power
$I_{a_i}^a$	the interference power from all concurrent transmissions of other APs
$I_{a_i}^u$	the interference power from all concurrent transmissions of other users

(i,j)	a conflict edge between agents a_i and a_j
$K = \|a_j \cup Sep_j\| + 1$	the domain of each set of elements in the variable $\textbf{LocalView}_j$
L	maximum number of channels the APs can assign
l_d	the link density
lc_j	local cost for all possible set of combinations between a_j and Sep_j
$lc_j(d_j)$	the local cost of an AP a_j using the chosen value d_j
lc_{max}	the greatest local cost in a relation
lc_{min}	the minimum local cost in a relation
$Max(R,x)$	the Max operator, selects tuples in R where value of x is the maximum
$Min(R,x)$	the Max operator, selects tuples in R where value of x is the minimum
N	a number of elements
N_{a_i}	the noise floor and rogue interference
O	the order of, Big-O notation
P_k	the parent of an agent a_k
P_j	the parent node of a_j, in the hierarchy of the pseudo-tree
PC_j	the set of pseudo-children of a node a_j
PP_j	the set of pseudo-parents of a node a_j
P_j^{value}	the set of parents of an agent a_j in the VALUE graph
$P_j^{value'}$	the subset of P_j^{value} whose nodes are descendent of an $a_i \in P_j^{value}$
P_j^{view}	the parent of an agent a_j in the VIEW graph
ϕ_{max}	an upper bound on the diameter of the constraint graph (Algorithm 1)
$\mathbb{R}$	the finite domain of interference, overlap factors values
R_j	a relation of an agent a_j with its Sep_j
$Selection(R,c)$	the operator SELECTION selects tuples in R that satisfy the condition c
Sep_j	the separator of node a_j, all ancestors of a_j which are connected to a_j or with descendants of a_j
Sep_k	the separators received from children agents
$\mathcal{SO}(n,m)$	the spectrum overlapping factor between channels n and m
$sc_j(d_j)$	the current lower bound cost
sc_k	the current lower bound cost received from children agents in the VIEW graph

$Utildim$	a parameter to limit de size of the hypercube
$S_{a_i \leftarrow u_k}$	the average power received from client u_k
$S_{a_i \leftarrow a_j}$	the average power received from an AP a_j
$\mathcal{S}(n, f)$	the unfiltered modulated signal
$S_j = D^K$	the finite domain D where $K = \|a_j \cup Sep_j\| + 1$
$splintpoint$	the node when visited that has two or more unvisited nodes
$UTIL_k^j$	the *UTIL* message sent by agent a_k to agent a_j
$UTIL_j^{P_j\,tmp}$	a limited dimension of $UTIL_j^{P_j}$ sent by agent a_j to its parent P_j
$UTIL_j^{P_j\,dep}$	a limited dimension of $UTIL_j^{P_j}$ locally stored by an agent a_j
$v_{a_k}^*(d_j^n)$	a vector of optimal utilities a child a_k sends a node a_j
$USelect(R)$	an SELECTION operator to select n-tuples in R and extracts different combinations with highest utility
$USelectMIN$	an SELECTION operator to select n-tuples in R and extracts different combinations with lowest local cost
u_k	an user associated to an AP
u_n	an user associated to an AP
V	a set of vertices (nodes)
$\mathbb{V}_m$	the set of the indexes of cells that interfere with $\mathbb{Z}_m$
$VALUE_{P_j}^{a_j}$	a message a node a_j receives from its parent P_j
v_j	a vertex $v_j \in V$
X^*	the optimal set of utility values which maximize a global cost function
$X = \{x_1, x_2, \cdots, x_l\}$	APs variables
(x, y)	the current choices of agents a_i and a_j
$x_1, \cdots, x_k$	the variables in the scope of a relation R
x_n	an element of the finite domain D_n
w	the *treewidth*
w_i	the *induced width* of the pseudo-tree
$\mathbb{Z}_m$	a group of nearby cells
$z_{j_a}(d_j)$	the lowest *estimated lower bound cost* value
z_j^*	the estimate lower bound cost of the agent owning the decision variable d_j

z_k^*	the estimated lower bound cost received from children agents in the VIEW graph
$\oplus$	the JOIN or sum operator
$\bigoplus_{UTIL_{C_j}} = \bigoplus_k^{C_j} UTIL_k^j$	join operator between the UTIL messages received from all children C_j of a node a_j
$\perp$	the PROJECTION operator on relations
$\langle a_i, a_j \rangle$	a tuple a_i and a_j
ψ_m	number of times an AP_m was locked

References

[1] "Fórum do sistema brasileiro de tv digital terrestre, normas brasileiras de tv digital." http://forumsbtvd.org.br/acervo-online/normas-brasileiras-de-tv-digital/.

[2] C. Montez and V. Becker, *TV digital interativa: conceitos, desafios e perspectivas para o Brasil.* Editora da UFSC, 2005.

[3] "Plc forum." http://www.plcforum.org/.

[4] "Itu global standard for international mobile telecommunications - imt-advanced." Circular letter, ITU-R, March 2008.

[5] "International telecommunication union (itu) study group 15's g.9 series recommendations." http://www.itu.int/ITU-T/recommendations/.

[6] S. Palm, "xdsl tutorial and standards update," in *International Conference on Consumer Electronics, 2000. ICCE. Digest of Technical Papers.*, pp. 68–69, 2000.

[7] A. Budri, J. Goncalves, and L. Meloni, "Wimax simulation models for return channel in digital television systems," in *Telecommunications Symposium, 2006 International*, pp. 688 –693, September 2006.

[8] IEEE802.11std, "Telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications (includes ieee std 802.11, 1999 edition; ieee std 802.11a.-1999; ieee std 802.11b.-1999; ieee std 802.11b.-1999/cor 1-2001; and ieee std 802.11d.-2001)," *ISO/IEC Standard for Information Technology*, p. i, 2005.

[9] "Wireless lan medium access control (mac) and physical layer (phy) specifications: High-speed physical layer in the 5ghz band." IEEE 802.11 Standard, 1999.

[10] "Wireless lan medium access control (mac) and physical layer (phy) specifications: Higher-speed physical layer extension in the 2.4ghz band." IEEE 802.11 Standard, 1999.

[11] "Wireless lan medium access control (mac) and physical layer (phy) specifications: Further higher data rate extension in the 2.4ghz band." IEEE 802.11 Standard, 2003.

[12] "Ieee std 802.11n-2009 - ieee standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac)and physical layer (phy) specifications amendment 5: Enhancements for higher throughput." IEEE 802.11 Standard, 2009.

[13] "Ministério das comunicacões." http://www.mc.gov.br/.

[14] A. Mishra, S. Banerjee, and W. Arbaugh, "Weighted coloring based channel assignment for wlans," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, pp. 19–31, July 2005.

[15] B. Aoun, R. Boutaba, and G. Kenward, "Analysis of capacity improvements in multi-radio wireless mesh network," in *Proceedings of IEEE Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, (Melbourne, Vic.), pp. 543–547, IEEE, 2006.

[16] K. Ramachandran, E. M. Belding, K. C. Almeroth, and M. M. Buddhikot, "Interference-aware channel assignment in multi-radio wireless mesh networks," *IEEE INFOCOM*, 2007.

[17] J. Chen, G. de Veciana, and T. Rappaport, "Site-specific knowledge and interference measurement for improving frequency allocations in wireless networks," *Vehicular Technology, IEEE Transactions on*, vol. 58, no. 5, pp. 2366–2377, jun 2009.

[18] X. Yue, C.-F. Wong, and S. H. G. Chan, "Cacao: Distributed client-assisted channel assignment optimization for uncoordinated wlans," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 9, pp. 1433–1440, september 2011.

[19] "The wi-fi alliance." http://wi-fi.org, 1999.

[20] J. Jun and M. L. Sichitiu, "The nominal capacity of wireless mesh networks," *Wireless Commun*, vol. 10, pp. 8–14, 2003.

[21] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," *Wirel. Netw.*, vol. 11, no. 4, pp. 471–487, july 2005.

[22] C. Hua and R. Zheng, "Starvation modeling and identification in dense 802.11 wireless community networks," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pp. 1022 –1030, april 2008.

[23] N. Nie and C. Comaniciu, "Adaptive channel allocation spectrum etiquette for cognitive radio networks," in *Proc. of IEEE DySPAN*, pp. 269–278, 2005.

[24] J. Huang, R. A. Berry, and M. L. Honig, "Auction-based spectrum sharing," *Mob. Netw. Appl.*, vol. 11, pp. 405–418, June 2006.

[25] L. Cao and H. Zeng, "Distributed spectrum allocation via local bargaining," in *Proc. of IEEE DySPAN*, pp. 475–486, 2005.

[26] H. Zheng, "Collaboration and fairness in opportunistic spectrum access," in *Proc. of IEEE International Conference on Communications*, pp. 3132–3136, 2005.

[27] C. Peng, H. Zheng, and B. Y. Zhao, "Utilization and fairness in spectrum assignment for opportunistic spectrum access," *Mob. Netw. Appl.*, vol. 11, pp. 555–576, August 2006.

[28] M. Achanta, "Method and apparatus for least congested channel scan for wireless access points." International patent WO/2006/042217, april 2006.

[29] "Cisco system,inc." http://www.cisco.com/.

[30] A. Mishra, V. Brik, S. Banerjee, A. Srinivasan, and W. Arbaugh, "A client-driven approach for channel management in wireless lans," in *IEEE Infocom*, 2006.

[31] D. J. Leith and P. Clifford, "A self-managed distributed channel selection algorithm for wlans," in *Proceedings of 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pp. 1–9, 2006.

[32] K. Briggs and M. Tijmes, "Optimal channel allocation for wireless cities," *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, pp. 1–5, 2009.

[33] P. Pathak and R. Dutta, "A survey of network design problems and joint design approaches in wireless mesh networks," *Communications Surveys Tutorials, IEEE*, vol. 13, no. 3, pp. 396–428, quarter 2011.

[34] R. Bruno, M. Conti, and E. Gregori, "Mesh networks: commodity multihop ad hoc networks," *Communications Magazine, IEEE*, vol. 43, no. 3, pp. 123–131, march 2005.

[35] I. Akyildiz and X. Wang, "A survey on wireless mesh networks," *Communications Magazine, IEEE*, vol. 43, no. 9, pp. S23–S30, september 2005.

[36] J. Shi, T. Salonidis, and E. W. Knightly, "Starvation mitigation through multi-channel coordination in csma multi-hop wireless networks," in *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '06, (New York, NY, USA), pp. 214–225, ACM, 2006.

[37] J. So and N. H. Vaidya, "Multi-channel mac for ad hoc networks: handling multi-channel hidden terminals using a single transceiver," in *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '04, (New York, NY, USA), pp. 222–233, ACM, 2004.

[38] "Ieee, draft amendment: Ess mesh networking, ieee p802.11s draft 1.00." IEEE P802.11s/D8.0, December 2010.

[39] E. Aryafar, O. Gurewitz, and E. Knightly, "Distance-1 constrained channel assignment in single radio wireless mesh networks," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pp. 762–770, april 2008.

[40] K. Sycara, A. Pannu, M. Williamson, D. Zeng, and K. Decker, "Distributed intelligent agents," *IEEE Intelligent Systems*, vol. 11, pp. 36–46, 1996.

[41] A. Petcu, *A Class of Algorithms for Distributed Constraint Optimization.* Phd. thesis no. 3942, Swiss Federal Institute of Technology (EPFL), Lausanne (Switzerland), 2007.

[42] R. Dechter and R. Mateescu, "And/or search spaces for graphical models," *Artificial Intelligence*, pp. 73–106, 2007.

[43] P. J. Modi, W. M. Shen, M. Tambe, and M. Yokoo, "Adopt: asynchronous distributed constraint optimization with quality guarantees." Artif. Intelligence, 2006.

[44] R. J. Bayardo and D. P. Miranker, "On the space-time trade-off in solving constraint satisfaction problems," in *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1*, IJCAI'95, (San Francisco, CA, USA), pp. 558–562, Morgan Kaufmann Publishers Inc., 1995.

[45] Z. Collin, R. Dechter, and S. Katz, "On the feasibility of distributed constraint satisfaction," in *Proceedings of the 12th international joint conference on Artificial intelligence - Volume 1*, IJCAI'91, (San Francisco, CA, USA), pp. 318–324, Morgan Kaufmann Publishers Inc., 1991.

[46] Z. Collin, R. Dechter, and S. Katz, *Self-stabilizing Distributed Constraint Satisfaction.* Technical report // Department of Computer Science, Technion Israel Institute of Technology, Information and Computer Science, University of California, Irvine, 1991.

[47] E. C. Freuder and M. J. Quinn, "Taking advantage of stable sets of variables in constraint satisfaction problems," in *Proceedings of the 9th international joint conference on Artificial intelligence - Volume 2*, (San Francisco, CA, USA), pp. 1076–1078, Morgan Kaufmann Publishers Inc., 1985.

[48] R. Dechter, *Reasoning with Graphical Models.* University of California, October 2007.

[49] T. Y. Cheung, "Graph traversal techniques and the maximum flow problem in distributed computation," *IEEE Trans. Softw. Eng.*, vol. 9, pp. 504–512, july 1983.

[50] B. Awerbuch, "A new distributed depth-first search algorithm." Information Processing Letters, 1985.

[51] K. B. Lakshmanan, N. Meenakshi, and K. Thulasiraman, "A time-optimal message-efficient distributed algorithm for depth-first-search." Information Processing Letters, 1987.

[52] I. Cidon, "Yet another distributed depth-first-search algorithm." Information Processing Letters, 1987/88.

[53] J. Reif, "Depth-first search is inherently sequential." Information Processing Letters, 1985.

[54] M. B. Sharma, S. S. Iyengar, and N. Mandyam, “An efficient distributed depth-first-search algorithm.” Information Processing Letters, 1989.

[55] M. B. Sharma, S. S. Iyengar, and N. K. Mandyam, “Corrections to a distributed depth-first search algorith.” Information Processing Letters, 1990.

[56] S. Makki, G. Havas, U. of Queensland. Dept. of Computer Science, and U. of Queensland. Key Centre for Software Technology, *Distributed Algorithms for Constructing a Depth-first-search Tree.* Technical report (University of Queensland. Dept. of Computer Science), Department of Computer Science, University of Queensland, 1993.

[57] Z. Collin and S. Dolev, “Self-stabilizing depth-first-search algorithm.” Information Processing Letters, 1994.

[58] V. Barbosa, *An Introduction to Distributed Algorithms.* Mit Press, 1996.

[59] H. H. Abu-Amara, “Fault-tolerant distributed algorithm for election in complete networks,” *IEEE Trans. Comput.*, vol. 37, no. 4, pp. 449–453, april 1988.

[60] R. Bar-Yehuda, S. Kutten, Y. Wolfstahl, and S. Zaks, “Making distributed spanning tree algorithms fault-resilient,” in *STACS 87* (F. Brandenburg, G. Vidal-Naquet, and M. Wirsing, eds.), vol. 247 of *Lecture Notes in Computer Science*, pp. 432–444, Springer Berlin / Heidelberg, 1987.

[61] T. Léauté, *Distributed Constraint Optimization: Privacy Guarantees and Stochastic Uncertainty.* Phd thesis, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, November 11 2011.

[62] T. Léauté, B. Ottens, and R. Szymanek, “FRODO 2.0: An open-source framework for distributed constraint optimization,” in *Proceedings of the IJCAI’09 Distributed Constraint Reasoning Workshop (DCR’09)*, (Pasadena, California, USA), pp. 160–164, July 2009. http://frodo2.sourceforge.net.

[63] A. Petcu and B. Faltings, “A scalable method for multiagent constraint optimization,” in *Proceedings of the 19th international joint conference on Artificial intelligence*, (San Francisco, CA, USA), pp. 266–271, Morgan Kaufmann Publishers Inc., 2005.

[64] M. Yokoo, *Distributed constraint satisfaction: foundations of cooperation in multi-agent systems.* London, UK: Springer-Verlag, 2001.

[65] M. Yokoo and K. Hirayama, "Distributed constraint sastifaction algorithm for complex local problems," in *Proceedings of International Conference on Multiagent Systems*, 1998.

[66] K. Hirayama and M. Yokoo, "An approach to over-constrained distributed constraint satisfaction problems - distributed hierarchichal constraint satisfaction," in *Proceedings of Internationak Conference on Multiagent Systems*, 2000.

[67] K. Hirayama and M. Yokoo, "Distributed partial constraint satisfaction problem," in *Principles and Practice of Constraint Programming*, pp. 222–236, 1997.

[68] F. Rossi, P. V. Beek, and T. Walsh, "Handbook of constraint programming." Elsevier, 2006.

[69] A. Petcu and B. Faltings, "Odpop: An algorithm for open/distributed constraint optimization," in *AAAI*, pp. 703–708, 2006.

[70] A. Meisels, "Distributed search by constrained agents." Springer, 2008.

[71] A. Chechetka and K. Sycara, "No-commitment branch and bound search for distributed constraint optimization," in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, AAMAS '06, (New York, NY, USA), pp. 1427–1429, ACM, 2006.

[72] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo, "Solving distributed constraint optimization problems optimally, efficiently an asynchronously," 2003. Extension of : An Asynchronous, Complete Method for General Distributed Constraint Optimization.

[73] R. Mailler and V. Lesser, "Solving distributed constraint optimization problems using cooperative mediation," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '04, (Washington, DC, USA), pp. 438–445, IEEE Computer Society, 2004.

[74] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo, "An asynchronous complete method for distributed constraint optimization," in *AAMAS*, pp. 161–168, 2003.

[75] T. Rappaport, *Wireless Communications: Principles and Practice.* Upper Saddle River, NJ, USA: Prentice Hall PTR, 2nd ed., 2001.

[76] T. Monteiro, M. Pellenz, M. Penna, F. Enembreck, and R. Souza, “On optimal distributed channel allocation for access points in wlans,” in *NETWORKING 2011 Workshops*, vol. 6827 of *Lecture Notes in Computer Science*, pp. 73–84, Springer Berlin / Heidelberg, 2011.

[77] T. L. Monteiro, M. E. Pellenz, M. C. Penna, F. Enembreck, R. D. Souza, and G. Pujolle, “Channel allocation algorithms for wlans using distributed optimization,” *AEU - International Journal of Electronics and Communications*, vol. 66, no. 6, no. 6, pp. 480 – 490, 2012.

[78] “Opnet technologies, inc..” www.opnet.com, 1986.

[79] K. K. Leung, “Frequency assignment for ieee 802.11 wireless networks,” in *Proc. 58th IEEE Vehicular Technology Conference*, pp. 1422–1426, 2003.

[80] Y. Lee, K. Kim, and Y. Choi, “Optimization of ap placement and channel assignment in wireless lans,” in *Proceedings of 27th Annual IEEE Conference on Local Computer Networks (LCN)*, pp. 831–836, 2002.

[81] “Library simpatica.” Available at http://www.ppgia.pucpr.br/ maziero/doku.php/software:simulation, 2009.

[82] D. Jungnickel, *Graphs, Networks and Algorithms.* Springer Berlin Heidelberg, 2008.